Albert-Ludwigs-Universität Freiburg
Institut für Informatik und Gesellschaft

Dissertation zur Erlangung des
Doktorgrades der Fakultät für Angewandte Wissenschaften der
Albert-Ludwigs-Universität Freiburg im Breisgau

Sicherheit in Geschäftsprozessen

Prozessrekonstruktion als Werkzeug nachgelagerter Prozessanalysen

vorgelegt von

Thomas Stocker

aus Bötzingen

November 2014

Dekan:	Prof. Dr. Georg Lausen, Albert-Ludwigs-Universität Freiburg
Erstreferent:	Prof. Dr. Dr. h.c. Günter Müller, Albert-Ludwigs-Universität Freiburg
Zweitreferent:	Prof. Dr. Bernd Becker, Albert-Ludwigs-Universität Freiburg
Datum der Disputation:	07. Januar 2015

Zusammenfassung

Die Standardisierung und Automatisierung betrieblicher Abläufe eröffnet Unternehmen reichhaltige Möglichkeiten zur Rationalisierung ihrer Prozesse und deren Kontrolle. Dabei besteht die Notwendigkeit, Prozesse kontinuierlich an sich verändernde Rahmenbedingungen wie Gesetze, Branchenstandards oder interne Richtlinien anzupassen und in diesem Spannungsfeld die Sicherheit i.S. der Einhaltung verbindlicher Vorgaben für deren Ablauf zu wahren. Hinsichtlich Nachweisen für die Regelkonformität von Prozessen wird typischerweise ein zu starker Fokus auf präventive Verfahren gelegt. Unvollständige Spezifikation, unsachgemäße Verwendung und Schwachstellen von Prozessen können zu abweichendem Prozessverhalten führen und damit Verletzungen verbindlicher Vorgaben zur Wahrung der Sicherheit begünstigen. Zur Ableitung faktenbasierter Aussagen hinsichtlich der Sicherheit von Geschäftsprozessen ist deshalb die Erkennung und Bewertung von Abweichungen im Rahmen nachgelagerter (ex-post) Analysen unerlässlich.

Der Fokus dieser Arbeit liegt auf der nachgelagerten Analyse von Sicherheit. Dazu wird die Verwendung von Process Mining vorgeschlagen, welches bekannte Konzepte des Data Mining auf Prozessebene adaptiert und damit die Untersuchung ablaufbezogener Fragestellungen ermöglicht. Erstmals wird dabei der Nutzen dieser Technologie systematisch im Hinblick auf sicherheitsspezifische Anforderungen evaluiert. Ferner wird der Stand der Technik hinsichtlich Verfahren zur Erlangung eines detaillierten Verständnisses der Prozessausführung um folgende Methoden erweitert:

- GENET^{+}: Ein Verfahren zur Rekonstruktion von Kontrollflussmodellen, die tatsächliches Prozessverhalten gemäß aufgezeichneter Ablaufdaten präzise darstellen.
- GENET*: Eine Erweiterung von GENET^{+} zur zusätzlichen Berücksichtigung von Datenflüssen bei der Rekonstruktion.
- APClustering: Eine Methode zur Erfassung der Prozessdynamik i.S. unterschiedlicher Ausführungsphasen von Prozessen im zeitlichen Verlauf.

Auf diese Weise kann im Rahmen nachgelagerter Sicherheitsanalysen von Geschäftsprozessen tatsächliches Verhalten und das Zustandekommen von Abweichungen in rekonstruierten Modellen nachvollzogen und die Reaktion von Prozessen auf externe Ereignisse wie Ausnahmesituationen oder Anpassungsoperationen beurteilt werden. Die Evaluation der entwickelten Verfahren berücksichtigt deren formale Korrektheit (Adäquanz der theoretischen Basis), die Berechnungskomplexität zugehöriger Algorithmen und die praktische Durchführbarkeit. Dazu werden Ablaufdaten auf Basis von Prozessmodellen unterschiedlicher Komplexität generiert und in Experimenten zum empirischen Nachweis der Funktionstüchtigkeit der Verfahren verwendet.

Abstract

Standardization and automation of internal procedures allow companies to run their processes more efficiently and reach business goals with optimal resource allocation and better control. At the same time, processes have to be continuously adapted to changing circumstances (laws, best-practices, internal policies) without violating security in terms of adherence to processing guidelines. Typically, the compliance of business processes to guidelines is assured with preventive mechanisms. However, incomplete or faulty process specifications, inappropriate usage as well as vulnerabilities and ineffective controls can cause deviations from planned behavior and ultimately lead to security breaches. Thus, reliable statements about the security of business processes require detection and evaluation of deviations from an ex-post perspective.

The focus of this work is on the a posteriori analysis of business process security and compliance. In this context, it puts forward Process Mining which adapts Data Mining concepts to processes and allows to analyze recorded process data in order to detect and visualize hidden relations. For the first time, the benefit of this technology is systematically evaluated wrt. security-specific requirements. State of the art in this direction is extended by the following three approaches:

- GENET^{+}: A discovery-method which is able to produce control-flow models showing actual process behavior on basis of recorded process events in a precise way.
- GENET*: An extension of GENET^{+} for the consideration of control-flow and data-flow information in a unified way.
- APClustering: A method to gather process dynamics in the sense of different execution phases along time.

Altogether, these mechanisms allow for a deep understanding of real process behavior, help analysts in observing reactions of processes to external events (emergency situations, adjustment operations) and finding root causes for deviating process behavior. Evaluation of developed mechanisms considers formal correctness (adequacy of theoretical basis), computing complexity of corresponding algorithms and feasibility. For this, a set of synthetic process logs is generated using sample models with different degree of complexity. Intense experiments on basis of these logs empirically show characteristics and functionality of proposed approaches.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Sicherheit im Geschäftsprozessmanagement

Ein integraler Bestandteil unternehmerischen Erfolgs besteht in der strategischen Koordination verschiedener, in ihrer Summe wertschöpfender, betrieblicher Abläufe. Diese betreffen die einzelnen Prozesse entlang einer Wertschöpfungskette. Zielgerichteter Ressourceneinsatz innerhalb der Bestandteile dieser Kette ist eine zentrale Voraussetzung für die Wettbewerbsfähigkeit eines Unternehmens.

Geschäftsprozesse definieren die Bestandteile betrieblicher Abläufe zur Erreichung spezifischer Ziele (z.B. Abwicklung eines Bestellvorgangs) und beschreiben dabei, wie Ressourcen (Personen, Material, Information) eingesetzt bzw. verwendet werden, um diese Ziele möglichst effizient und gemäß geltender Richtlinien und Verfahrensvorschriften zu erreichen. Aufgrund des positiven Einflusses effizienter Geschäftsprozesse auf die Wirtschaftlichkeit von Unternehmen und ihres enthaltenen Know-hows, werden sie längst als wesentliche Vermögenswerte gesehen. Eine prozessorientierte Sichtweise erleichtert Messungen bzgl. des Beitrags einzelner Abläufe zum Unternehmenserfolg, damit verbundene Kosten, sowie die Identifikation von Schwachstellen, Engpässen und fehlerhafter Ressourcenallokation. Durch die Fokussierung auf Abläufe und die Einbindung verschiedener Abteilungen, Ressorts und Managementebenen, entsteht eine ganzheitliche Sicht auf die unternehmerische Aktivität, deren Struktur und Zusammenhang.

Die Ausgestaltung und Steuerung von Prozessen wird begleitet von der Notwendigkeit, verbindliche regulatorische Vorgaben auf Prozessebene zu berücksichtigen. Gesetzliche Regelungen, Branchenstandards, Datenschutzrichtlinien, Verträge und unternehmensinterne Vorgaben haben dabei erheblichen Einfluss auf die Prozessausführung. Wirtschaftsskandale der letzten Jahre (bspw. Enron, WorldCom[1]) haben die Verletzlichkeit von Unternehmen durch schwerwiegende Sicherheitsmängel in internen Arbeitsabläufen gezeigt. Prozessteilnehmer mit weitreichenden Kenntnissen dieser Schwachstellen haben gerade im Finanzwesen kreative Wege gefunden, Kontrollsysteme zu umgehen und durch Veruntreuung und Manipulation mitunter Milliardenverluste verursacht (Société Générale: 4 Mrd. Euro, Schweizer UBS: 4 Mio. Euro). Gemäß einer aktuellen globalen Studie

[1] Vgl. Barizo 2014.

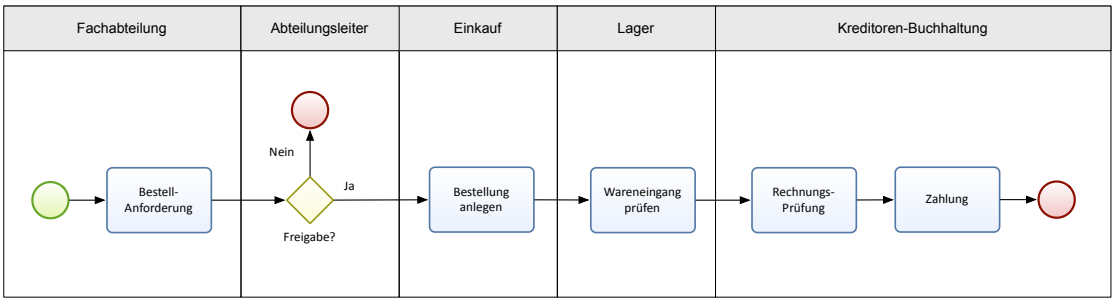


Abbildung 1.1.: Geplanter Ablauf eines Bestellvorgangs.

(5100 Unternehmen in 99 Ländern) sehen sich nahezu alle Unternehmen von Wirtschaftskriminalität bedroht, rund ein Drittel sieht sich Folgen konkreter Delikte ausgesetzt. Betroffen sind meist Kernprozesse wie Zahlungsabwicklung, Auftragsvergabe und Vertrieb[2]. Der dadurch verursachte Schaden kann für Unternehmen existenzbedrohende Ausmaße annehmen und wird in Deutschland vom Bundeskriminalamt für 2013 mit ca. 4 Mrd. Euro beziffert[3].

Sicherheit in Geschäftsprozessen kann auf unterschiedliche Art und Weise definiert werden. Typischerweise stehen diesbezüglich Anforderungen der Zugangskontrolle (engl. *access control*) und deren Durchsetzung anhand von Mechanismen auf der Ebene der IT-Infrastruktur im Vordergrund. Diese Arbeit fokussiert Anforderungen auf höherer Ebene und führt Sicherheit in Geschäftsprozessen auf die Einhaltung verbindlicher Prozessvorgaben zurück, die der Verhinderung wirtschaftskrimineller Delikte und der Wahrung von Prozessvorgaben dienen und Unternehmen vor finanziellen Schäden, geschäftsschädigenden Wirkungen wie Reputationsverlust, aber auch Sanktionen aufgrund der Verletzung gesetzlicher Vorgaben (*Non-Compliance Risiko*) schützen sollen. In diesem Zusammenhang befasst sie sich mit Methoden zum effektiven Nachweis der Sicherheit von Geschäftsprozessen aus einer ex-post Perspektive, d.h. nach der eigentlichen Prozessausführung und auf Basis von Spuren, die diese in involvierten Informationssystemen hinterlässt. Durch die Entwicklung von Ansätzen der Prozessrekonstruktion und der Erkennung von Prozessveränderungen im zeitlichen Verlauf trägt sie zur Visualisierung und detaillierten Erfassung tatsächlichen Prozessverhaltens im Rahmen nachgelagerter Prozessprüfungen bei.

Im Folgenden wird anhand einer Einführung in das Geschäftsprozessmanagement zunächst der Kontext der Ausgestaltung und Durchführung von Geschäftsprozessen innerhalb von Unternehmen vermittelt. Danach definiert dieses Kapitel die unterschiedlichen Betrachtungsebenen von Sicherheit im betrieblichen Umfeld und identifiziert anschließend relevante Sicherheitsanforderungen auf Prozessebene, sowie Möglichkeiten zu deren Nachweis. Dabei wird insbesondere auf die Notwendigkeit nachgelagerter Prozessanalysen eingegangen. Die Problemstellung der Arbeit wird in Kapitel 2 aus Defiziten existierender Methoden zur ex-post Analyse von Geschäftsprozessen abgeleitet.

[2] Vgl. PwC 2014.
[3] Vgl. Bundeskriminalamt 2013.

1.1 Geschäftsprozessmanagement

Ausgehend von einer traditionell datenzentrierten Ausrichtung von Informationssystemen zur Unterstützung von Geschäftsprozessen[4] ist eine Hinwendung zu prozessorientierten Sichtweisen erkennbar, die mutmaßlich Globalisierungsaspekten und der damit verbundenen Notwendigkeit, Abläufe zu standardisieren und sich verändernden Rahmenbedingungen anzupassen, geschuldet ist[5]. Die Konzeption von Standardsoftware für die Steuerung betrieblicher Abläufe folgt dieser Entwicklung. So nahm bereits das 1993 vorgestellte SAP R/3 ERP-System mit dem integrierten *Business Process Reference Model*, eine prozessorientierte Sichtweise ein, erlaubte jedoch keine explizite Prozesssteuerung. Methoden der Prozessautomation haben ab Mitte der 90er Jahre eine Reihe von Systemen hervorgebracht, die den Ablauf von Geschäftsprozessen durch eine automatisierte Durchführung von Teilschritten unterstützen und den Begriff „Workflow Management" geprägt haben[6].

Das Geschäftsprozessmanagement (engl.: *Business Process Management*, kurz BPM) erweitert Workflowkonzepte um eine ganzheitliche Betrachtung von Prozessen im betrieblichen Kontext. Während Workflow Management als unterstützende Technologie aufgefasst werden kann, ist BPM eine prozessorientierte Managementdisziplin mit dem zugrundeliegenden Kalkül, durch effizientes Management und systematischer Optimierung von Prozessen zum Unternehmenserfolg beizutragen[7]. Dabei wird neben der (teil-)automatisierten Ausführung von Prozessen deren gesamter „Lebenszyklus" betrachtet, der bei der Spezifikation (Modellierung des geplanten Prozessverhaltens) beginnt, die Implementierung in der Systemlandschaft des Unternehmens einschließt und eine der Ausführung nachgelagerte Diagnose zur Aufdeckung von Optimierungspotenzialen beinhaltet (siehe Abb. 1.2). Im Gegensatz zu einem Workflowsystem beinhalten Systeme zur Unterstützung des Geschäftsprozessmanagements (BPMS) typischerweise nicht nur Werkzeuge zur automatisierten Ausführung von Geschäftsprozessen, sondern auch zur Erstellung von Modellen für das Soll-Verhalten und der Diagnose aufgezeichneter Daten abgeschlossener Durchläufe.

Ein Großteil der DAX-Unternehmen ist sich heute des Optimierungspotentials durch Standardisierung von Abläufen und der Notwendigkeit eines integrierten Managementkonzepts für Geschäftsprozesse bewusst (26 von 30 im Jahr 2009[9]). Das steigende Inter-

[4] Vgl. Weske 2012.
[5] Vgl. Dumas, Rosa et al. 2013.
[6] Vgl. Aalst 2003; Curtis et al. 1992; Jablonski et al. 1996.
[7] Vgl. Hill et al. 2008.
[8] Die Zielsetzung des BPM-Lifecycle-Modells besteht in der Veranschaulichung unterschiedlicher „Lebensphasen" von Geschäftsprozessen. Nach deren Spezifikation in der Design-Phase werden Prozesse in betriebliche Informationssysteme implementiert und dadurch ausführbar gemacht (Implementation). Der Phase tatsächlicher Prozessausführungen (Enactment) folgt ein nachgelagerter Schritt, in dem das Prozessverhalten nachvollzogen und beurteilt wird. Dies kann zu Anpassungen im geplanten Prozessverhalten führen, wodurch der Kreis geschlossen wird.
[9] Vgl. Best et al. 2010.

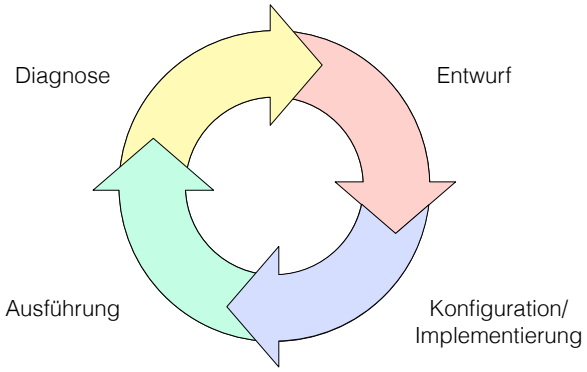


Abbildung 1.2.: BPM-Lebenszyklus von Geschäftsprozessen[8].

esse an Technologien zum Management von Geschäftsprozessen ist laut aktueller Studien auf globaler Ebene seit Jahren ungebrochen[10,11].

1.2 Sicherheit im betrieblichen Umfeld

Die allgemeine Sensibilisierung für Wirtschaftskriminalität hat zu einer Verschärfung der Gesetzeslage geführt. Mittlerweile sehen sich Unternehmen einer Fülle von Regularien und Handlungsempfehlungen gegenüber, die im Zuge der *Corporate Governance* in die Grundprinzipien für Leitung und Überwachung integriert werden müssen. Den mutmaßlich größten Effekt hatte diesbezüglich das amerikanische Bundesgesetz SOX (Sarbanes-Oxley Act of 2002), das zur Sicherung der Verlässlichkeit veröffentlichter Finanzdaten u.a. die Existenz eines internen Kontrollsystems (IKS)[12] für die Rechnungslegung und von unabhängigen Prüfern bestätigte Nachweise für dessen Effektivität forderte. Neben zahlreichen länderspezifischen Adaptierungen (J-SOX in Japan, CLERP 9 in Australien) wurden Teile dieser Bestimmungen auch in Deutschland adaptiert und u.a. in das Abschlussprüferaufsichtsgesetz (APAG), das Berufsaufsichtsreformgesetz (BilReG) und das Bilanzrechtsreformgesetz (BARefG) eingebracht[13].

Die Etablierung eines IKS erfordert eine enge Kopplung des Prozessmanagements mit dem Risiko- und Compliancemanagement. Während Risikomanagement allgemein auf die Identifikation, Bewertung und Behandlung insbesondere bestandsgefährdender Risiken abzielt, steht beim Compliancemanagement rechts- und regelkonformes Verhalten im

[10] Vgl. C. Wolf et al. 2012.

[11] Die Entwicklung des Workflowparadigmas hin zu BPM und dessen Akzeptanz ist auf akademischer Seite an der Durchsetzung renommierter Konferenzen mit prozessbezogenen Themen (EE-Track der ACM-SAC Konferenz) und dem steigenden Erfolg spezifischer Konferenzen wie der seit 2003 durchgeführten BPM (Vgl. Daniel et al. 2013) erkennbar.

[12] Vgl. Gabler Wirtschaftslexikon 2014.

[13] Die Implementierung von Inhalten des SOX-Acts in deutsche Gesetze zur Überwachung von Wirtschaftsprüfungs-Aktivitäten diente vorrangig deren fundamentalen Reform mit dem Ziel einer Zulassung der PCAOB, einer im Zuge von SOX neu gegründeten Behörde zur Koordination der Ablösung des bisherigen Eigenregulierungs-Konzepts von Wirtschaftsprüfern durch ein System externer, unabhängiger Prüfungen.

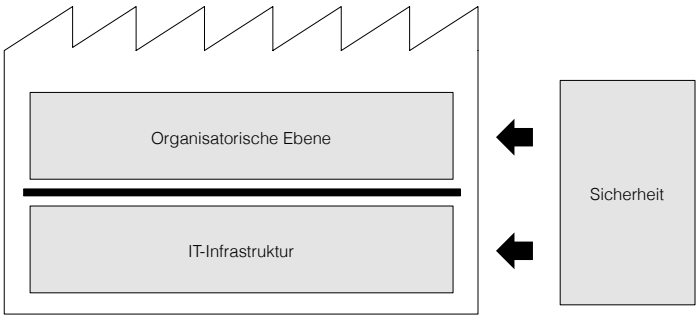


Abbildung 1.3.: Ebenen der Unternehmensarchitektur.

Vordergrund[14,15]. Aus der Perspektive der Sicherheit bezieht sich ein Risiko stets auf die Nichteinhaltung einer konkreten Eigenschaft, z.B. unautorisierten Datenzugriff. Identifizierte Risiken können entweder akzeptiert (aufgrund zu geringer Eintrittswahrscheinlichkeit oder zu gering erwartetem Schaden), ausgelagert (z.B. durch eine Versicherung) oder behandelt werden. Behandelte Risiken sind für das IKS relevant, d.h. es muss eine adäquate Kontrolle zur Reduktion der Eintrittswahrscheinlichkeit implementiert werden. Im Beispiel nicht erfasster Rechnungen kann eine Kontrolle in periodischen, manuellen Prüfungen auf Stichprobenbasis bestehen. Denkbar sind ebenfalls IT-gestützte, automatische Kontrollmechanismen, die bspw. unerlaubte Zugriffe auf Datenquellen unterbinden. Kontrollen müssen in jedem Fall effektiv sein, d.h. das Risiko nachvollziehbar mindern.

Kontrollen können auf unterschiedlichen Ebenen implementiert werden. Grundsätzlich kann die Architektur eines Unternehmens in die IT-Infrastruktur und die organisatorische Ebene unterteilt werden, wobei Anforderungen der Sicherheit auf beiden Ebenen betrachtet werden müssen (siehe Abb. 1.3). Während die IT-Infrastruktur des Unternehmens aus IT-Systemen wie Datei-Server, Datenbank- oder ERP-Systemen besteht und in erster Linie zur Realisierung und Unterstützung betrieblicher Abläufe dient, bezieht sich die organisatorische Ebene auf die wertschöpfende Aktivität des Unternehmens. Merkmal dieser Ebene ist die Fähigkeit, Geschäftsprozesse so effizient und effektiv zu gestalten, dass die Wettbewerbsfähigkeit des Unternehmens gewahrt wird und dabei stets genügend Spielraum bleibt, um auf veränderte Marktbedingungen rechtzeitig und angemessen zu reagieren.

Neben grundlegenden Anforderungen bzgl. der Verfügbarkeit von Daten, Systemen und Kontrollmechanismen, bezieht sich Sicherheit auf der Ebene der IT-Infrastruktur hauptsächlich auf die Informationssicherheit. Mechanismen zur fehlerfreien, abhörsicheren Datenübertragung (Verschlüsselungskonzepte), zur persistenten und manipulationsresistenten Speicherung sowie zur Nachvollziehbarkeit von Aktivitäten innerhalb von IT-

[14] Vgl. Ozip-Philippsen 2013.

[15] Ein Risiko ist durch eine Eintrittswahrscheinlichkeit, verbunden mit einer Schadenshöhe, gekennzeichnet. Zur Identifikation von Einzelrisiken können Risikokataloge herangezogen werden, die für unterschiedliche Standardprozesse (z.B. zur Materialbeschaffung) spezifische Risiken beinhalten. FMEA oder das Fehler-Prozess-Matrix Modell sind Beispiele für systematische Vorgehensweisen zur Identifikation von Risiken/Schwachstellen. Neben der Bestimmung potentieller Fehlerursachen, Fehlerarten und deren Auswirkungen, wird dabei auch die Eintrittswahrscheinlichkeit und die Schadenshöhe abgeschätzt.

Systemen kommt dabei eine tragende Rolle zu. Die Notwendigkeit, beim Entwurf von sichereren IT-Systemen und Kontrollmechanismen nicht nur externe Angreifer zu berücksichtigen, ergibt sich aus der Dimension und Charakteristik vergangener Betrugsfälle[16] durch Innentäter. Die Erkennung betrügerischer Handlungen interner Angreifer, die Wissen über interne Abläufe für kriminelle Zwecke nutzen, ist auf IT-Infrastruktur-Ebene allerdings oft schwer erkennbar, da Sicherheitskontrollen nicht selten durch rudimentäre „Angriffe“ wie die Verwendung von auf Notizzetteln hinterlegten Zugangsdaten oder die Ausnutzung bekannter, jedoch nicht abgefangener Systemschwachstellen umgangen werden.

Eine ergänzende Betrachtung von Prozessen auf organisatorischer Ebene kann durch eine aggregierte Sichtweise auf interne Abläufe zusätzliche Informationen zur Aufklärung von Verdachtsmomenten bieten. Dies kann eindrücklich am Fall der Société Générale 2008 verdeutlicht werden, bei dem der finanzielle Schaden durch eine mit 50 Mrd. Euro eingegangene Spekulation des Investmentbankers Kerviel über die Entwicklung europäischer Börsen zustande kam. Kerviel hatte über Jahre durch das Umgehen von Sicherheitsmechanismen und die Initiierung von Scheingeschäften Handelspositionen aufgebaut, die ihm Spekulationsgeschäfte in Milliardenhöhe ermöglichten, gleichzeitig aber den Marktwert der Bank bei weitem überschritten. Eine der verwendeten Methoden beruhte darauf, für hochriskante Spekulationen jeweils gegenteilig ausgerichtete Transaktionen zu hinterlegen, die einen nivellierenden Effekt auf das durch das System berechnete Risiko hatten[17]. Durch den Anschein stets abgesicherter Wertpapiergeschäfte war es Kerviel möglich, unter dem Radar des Risikomanagements zu operieren. Der Abschlussbericht[18] einer internen Kommission zur Aufdeckung der Aktivitäten von Kerviel legte charakteristische Handlungsabfolgen der betrügerischen Aktivität offen, die auf der Ebene der IT-Infrastruktur verborgen blieben, auf Prozessebene jedoch nachvollzogen werden konnten, also bspw. die vielfache Initiierung von Wertpapiergeschäften ohne erfolgte Bestätigung oder die Absicherung von Geschäften mit weiteren Scheingeschäften nach dem Prinzip eines Schneeballsystems.

Allgemein kann durch die Korrelation von Einzelereignissen (siehe Abb. 1.4) eine umfassende Sicht auf Abläufe gewonnen und Sicherheitsbetrachtungen auf der Ebene der IT-Infrastruktur durch prozessbezogene Aspekte ergänzt werden. Auf diese Weise kann dazu beigetragen werden, sicherheitskritische Aktivitäten auf höherer Abstraktionsebene zu identifizieren und durch unterstützendes Wissen über involvierte Akteure und beteiligte Prozessschritte zielgerichtet entsprechende Maßnahmen einzuleiten. Diese Arbeit fokussiert Sicherheitsanforderungen auf Prozessebene[19] und abstrahiert dabei bewusst von spezifischen Anforderungen der IT-Infrastruktur.

[16] Vgl. Barizo 2014.

[17] Ein Beispiel für ein solches Vorgehen wäre der Kauf eines Aktienindex-Terminkontrakts zur Sicherung eines vorteilhaften Preises bei fallenden Kursen (z.B. DAX) und eines Scheinkontrakts zur Absicherung gegen steigende Kurse, was bei entsprechendem Kapitaleinsatz das Risiko ausgleicht.

[18] Vgl. Société Générale - General Inspection Services 2008.

[19] Vgl. Atluri et al. 2008; Herrmann et al. 1999.

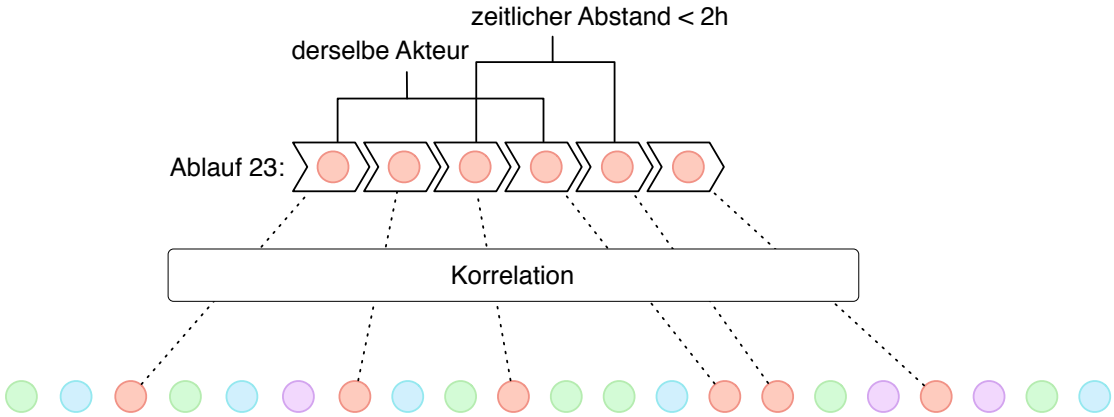


Abbildung 1.4.: Korrelation von Einzelereignissen zum Prozessablauf.

1.3 Sicherheitsanforderungen auf Prozessebene

Neben rein funktionalen Erfordernissen und Anforderungen bzgl. der Performanz von Geschäftsprozessen sind diese aus Sicherheitsperspektive mit den drei grundlegenden Schutzzielen der Informationssicherheit (engl.: *CIA-triad*) konfrontiert. Neben dem Schutz von Informationen gegen unberechtigten Zugriff (*Vertraulichkeit*) und unberechtigter Manipulation (*Integrität*) betrifft dies auch die *Verfügbarkeit* zum Zeitpunkt des Zugriffs. Durch die Betrachtung dieser Schutzziele entlang unterschiedlicher Prozessaspekte, können konkrete Sicherheitseigenschaften von Geschäftsprozessen auf systematische Art und Weise abgeleitet werden[20].

In diesem Zusammenhang sind drei Prozessaspekte wesentlich[21]. Während sich der *funktionale und verhaltensbezogene Aspekt* auf kausale und temporale Zusammenhänge zwischen Prozessaktivitäten und der strukturellen Komposition von Teilprozessen bezieht, betrachtet der *organisationelle Aspekt* Organisationseinheiten und deren Beteiligung in der Prozessausführung. Beim *Informationellen Aspekt* steht die Nutzung von Datenelementen im Vordergrund.

Funktionaler und verhaltensbezogener Aspekt. Der funktionale Aspekt bezieht sich auf die operative Zielsetzung des Prozesses und den Zusammenhang beteiligter Subprozesse. Bei Geschäftsprozessen betrifft dies vorrangig das durch den erfolgreichen Ablauf erreichte Geschäftsziel und schließt Informationen über einzubeziehende Abteilungen und Ressourcen ein. Hinsichtlich des verhaltensbezogenen Aspekts ist der Kontrollfluss eines Prozesses fundamental, da er sich auf die Existenz oder Abwesenheit, sowie Reihenfolge von Aktivitäten im Prozessverlauf bezieht. Die Spezifikation des Kontrollflusses kann auf unterschiedliche Art und Weise erfolgen[22]. Möglich sind textuelle Beschreibungen in Form von Regeln, die bestimmte Ablaufstrukturen vorgeben. Diese Art des *rule-based*

[20] Vgl. Müller et al. 2010.
[21] Vgl. Curtis et al. 1992; Jablonski et al. 1996.
[22] Vgl. Lu et al. 2007.

modeling[23] empfiehlt sich in Situationen hochflexibler Abläufe, die nicht zu stark eingeengt werden dürfen. Meist wird jedoch ein grafischer Ansatz für die Definition des Kontrollflusses verfolgt, der explizit gültige Pfade definiert und dabei auch Verzweigungen und Bedingungen für die Wahl bestimmter Pfade berücksichtigt. Gängige Metamodelle sind UML[24], Ereignisgesteuerte Prozessketten (EPKs)[25], BPMN[26] oder Petrinetze[27]. Innerhalb von Workflowsystemen können spezifizierte *de-jure* Prozessmodelle für die automatische Ablaufsteuerung verwendet werden.

Neben der grundsätzlichen Anforderung, dass Prozesse zur Laufzeit die anvisierten Geschäftsziele realisieren, beziehen sich Sicherheitseigenschaften des funktionalen und verhaltensbezogenen Aspekts hauptsächlich auf den Zusammenhang von Aktivitäten, fordern also z.B. bei Kreditanträgen, dass stets eine Solvenzprüfung erfolgen muss und dass in jedem Fall erst danach die Konditionen ausgehandelt werden. Viele solcher Anforderungen können durch die explizite Definition des Kontrollflusses in Form von Prozessmodellen abgedeckt werden. Eigenschaften aus dem Bereich der Nutzungskontrolle[28] erlauben die Definition zusätzlicher Vor- und Nachbedingungen für die Ausführung von Aktivitäten oder den Zugriff auf Ressourcen. Bedingungen wie „Die Unterschrift des Kunden muss nach Aushändigung der Vertragsunterlagen innerhalb von 7 Tagen erfolgen (um Zinsrisiken zu minimieren)“ beschreiben temporale Beziehungen von Aktivitäten unabhängig von konkreten Pfaden. Möglich sind auch Constraints bzgl. der Nachhaltung von Information (engl.: *data retention*) oder die Betrachtung der Häufigkeit von Zugriffen[29].

Organisationeller Aspekt. Im Zentrum des organisationellen Aspekts steht die Zugangskontrolle, i.e. die Regulierung von Berechtigungen einzelner Akteure (Personen oder IT-Systeme) innerhalb von Prozessen und die Gewährung von Zugriff auf entsprechende Ressourcen[30]. Berechtigungskonzepte innerhalb von Unternehmen arbeiten typischerweise rollenbasiert. Für unterschiedliche Aufgaben/Funktionen werden entsprechende Rollen gebildet, die selbst wiederum in Hierarchien strukturiert werden und Rechtevererbung einschließen können. In Mehrbenutzersystemen können RBAC-Systeme[31] dazu eingesetzt werden, rollenbasierte Berechtigungskonzepte umzusetzen. Workflowsysteme nutzen diese Information für *user-task assignments*, i.e. die Zuteilung von Einzelschritten im Prozessverlauf zu Akteuren mit entsprechenden Befugnissen und ausreichender Kapazität. Die Feststellung der Identität von Akteuren (Authentifizierung) ist meist Teil der unterstützenden IT-Infrastruktur und wird hier nicht betrachtet.

Zur Vermeidung von Interessenkonflikten können bei der Zuteilung von Rollen zu einzelnen Prozessaktivitäten Anforderungen zur Funktionstrennung (engl.: *segregation of duties*) definiert werden. Prinzipien der Funktionstrennung beziehen sich allgemein auf

[23] Vgl. Knolmayer et al. 2000.
[24] Vgl. ISO/IEC 19505-1 2012; ISO/IEC 19505-2 2012.
[25] Vgl. Keller et al. 1992.
[26] Vgl. ISO/IEC 19510 2013.
[27] Vgl. Aalst 1998.
[28] Vgl. Park et al. 2002.
[29] Vgl. Pretschner et al. 2006.
[30] Vgl. Sandhu et al. 1994.
[31] Vgl. Ferraiolo et al. 2009; Alturi et al. 2011.

die explizite Trennung von Funktionen und Befugnissen zur Vermeidung von Interessenkonflikten und der Reduktion von Betrugsrisiken[32]. Dabei sollte „ein und dieselbe Person oder Stellengruppe grundsätzlich nie alle Phasen eines Geschäftsvorfalls alleine durchführen und kontrollieren können, ohne dass eine andere Person in den Geschäftsvorfall eingreift, also z.B. gleichzeitig Entscheidungen über Vermögenswerte treffen, Belege ausstellen, Güter verwalten, darüber Buch führen und seine Arbeit selbst nachprüfen“[33].

Im *strukturellen* Fall werden Segregationen bei der Gestaltung der Organisationsstruktur (Aufbauorganisation eines Unternehmens) berücksichtigt. Die Definition geeigneter Rollen- und Berechtigungskonzepte soll dabei sicherstellen, dass Prozesse den Anforderungen der Funktionstrennung genügen. Die Mindestanforderungen an das Risikomanagement für Kreditinstitute (MaRisk[34]) fordern beispielsweise, dass Kreditentscheidungen nicht allein durch den Vertrieb getroffen werden dürfen, sondern die Marktfolge involviert werden muss. Der geplante Prozessablauf muss also so gestaltet werden, dass beide Bereiche beteiligt sind.

Rollenhierarchien können sehr komplex sein und in realen Ausführungen (auch aufgrund von Rechtedelegationen) dazu führen, dass Einzelpersonen durch dass Agieren in unterschiedlichen Rollen dennoch die beabsichtigte Trennung umgehen können. Daher wird auf *individueller* Ebene die Funktionstrennung durch das *Mehraugen-Prinzip* implementiert. Dabei wird gefordert, dass stets mehr als eine Person in einem Ablauf beteiligt ist.

Informationeller Aspekt. Der informationelle Aspekt betrifft im Prozessverlauf verwendete Datenelemente, deren Eigenschaften und Einfluss auf den Kontrollfluss des Prozesses. Bei einem Datenelement kann es sich je nach Art und Detailgrad der Prozessspezifikation um Datenbanken, Dokumente oder Variablen handeln. Durch die Betrachtung von Datenelementen können Anforderungen der Zugangskontrolle angereichert werden, um bestimmte Akteure bspw. aufgrund einer Kreditsumme oder der Höhe eines Rechnungsbetrags in den Prozess zu involvieren oder davon auszuschließen. Eine weitere Anwendung besteht in der Annotation von Verzweigungsbedingungen mit datenbezogenen Constraints, um die Wahl von Folgepfaden zu determinieren. Ausnahme- und Sonderfälle, aber auch Anforderungen der Sicherheit (z.B. in Form einer zusätzlichen Unterschrift bei hohen Kreditbeträgen) können auf diese Weise explizit modelliert werden.

Sicherheitsanforderungen bzgl. verarbeiteter Information sind in realen Ausführungskontexten allerdings facettenreicher. In die Entscheidung darüber, ob eine Person berechtigt ist, bestimmte Aktivitäten auszuführen, kann bspw. auch die Historie bereits getätigter Aktivitäten einfließen. Durch die Definition sog. Konfliktklassen kann in einem *Chinese Wall* Sicherheitsmodell[35] so vermieden werden, dass Geschäftsbereiche, die miteinander in Konflikt stehen, interferieren und Interessenkonflikte entstehen (z.B. wenn ein Finanzberater konkurrierende Unternehmen betreut).

Die Nutzung von Cloud-Technologien erfordert ebenfalls strikte Isolationsgarantien. Geschäftsprozesse unterschiedlicher Klienten dürfen keine Berührungspunkte haben und In-

[32] Vgl. Botha et al. 2001; Atluri et al. 2008.
[33] Vgl. Deutsches Institut für Interne Revision e.V. 2014.
[34] Vgl. BA 54-FR 2210-2012/0002 2011.
[35] Vgl. Brewer et al. 1989.

formationen nicht zwischen diesen Domänen fließen. Isolation kann aber auch innerhalb von Geschäftsprozessen betrachtet werden, wenn sensitive Information nur an bestimmte an der Prozessausführung beteiligte Parteien fließen darf (und nicht an Unberechtigte innerhalb des eigenen Unternehmens oder Externe bei unternehmensübergreifenden Prozessen). Neben der Informationsübertragung durch direkten Zugriff können im Rahmen der Informationsflusskontrolle dabei auch sog. „verdeckte Kanäle“ (engl.: *covert channels*) betrachtet werden, die eine indirekte Informationsübertragung erlauben[36]. Beispiele für verdeckte Kanäle sind *Implizite Flüsse*, *Storage-Channels* und *Timing-Channels*. Implizite Flüsse lassen z.B. durch die Beobachtung von nicht geheimen Aktivitäten im Prozessverlauf Rückschlüsse auf interne (geheime) Zustände zu[37].

Die Notwendigkeit verdeckte Kanäle zu betrachten ergibt sich hauptsächlich aus hoch sicherheitsbedürftigen Kontexten (z.B. Militär) und wurde bereits Mitte der 80er Jahre in dem amerikanischen TCSEC-Standard (Orange Book) für die Bewertung und Zertifizierung der Sicherheit von Computersystemen[38] berücksichtigt. Zugehörige Sicherheitseigenschaften sind geprägt von dem Begriff der Nicht-Interferenz (engl.: *non-interference*). Schwachstellen in Prozessen, die ungewollte direkte oder indirekte Informationsübertragung ermöglichen, werden als Informationslecks (engl.: *information-leaks*) bezeichnet.

Sicherheitsbezogene Prozessanforderungen können aufbauend auf vorherigen Ausführungen in folgende Klassen unterteilt werden:

- **Autorisierung.** Durchsetzung von Zugriffskontrollkonzepten zur Sicherstellung ausschließlich autorisierter Aktionen und Zugriffe auf Daten.
- **Nutzungskontrolle.** Vor- und Nachbedingungen für die Gewährung von Zugriff auf Daten oder die Ausführung von Prozessaktivitäten.
- **Funktionstrennung.** Trennung von Aufgaben/Funktionen zur Vermeidung von Interessenkonflikten. Strukturell anhand geeigneter Zugriffskontrollkonzepte, individuell durch Mehraugenprinzip.
- **Isolation.** Dynamische Trennung konfliktbehafteter Domänen durch Chinese-Wall. Vermeidung von Informationsflüssen über direkte oder indirekte Kanäle mithilfe der Informationsflusskontrolle.

1.4 Regelkonformität von Geschäftsprozessen

Regelkonformität von Geschäftsprozessen kann, analog zur klassischen Sicherheit, grundsätzlich präventiv oder detektivisch betrachtet werden[39]. Im präventiven Fall wird durch

[36] Vgl. Denning et al. 1977.

[37] Bei einem Storage-Channel beruhen Rückschlüsse auf der Charakteristik beobachtbarer Ausgaben. Unterscheiden sich z.B. bei einem Online-Login die Serversignaturen bei existierenden Benutzern und nicht existierenden Benutzern, können Benutzernamen durch wiederholte Anfragen identifiziert werden. Im Fall von Timing-Channels können aus dem zeitlichen Verhalten von Prozessen Rückschlüsse auf interne Zustände gezogen werden. Im aktuellen Beispiel könnte ein Timing-Channel durch unterschiedliche Antwortzeiten bei existierenden Benutzern oder nicht existierenden Benutzern entstehen.

[38] Vgl. DoD 5200.28-STD 1985.

[39] Vgl. Böhr et al. 2013.

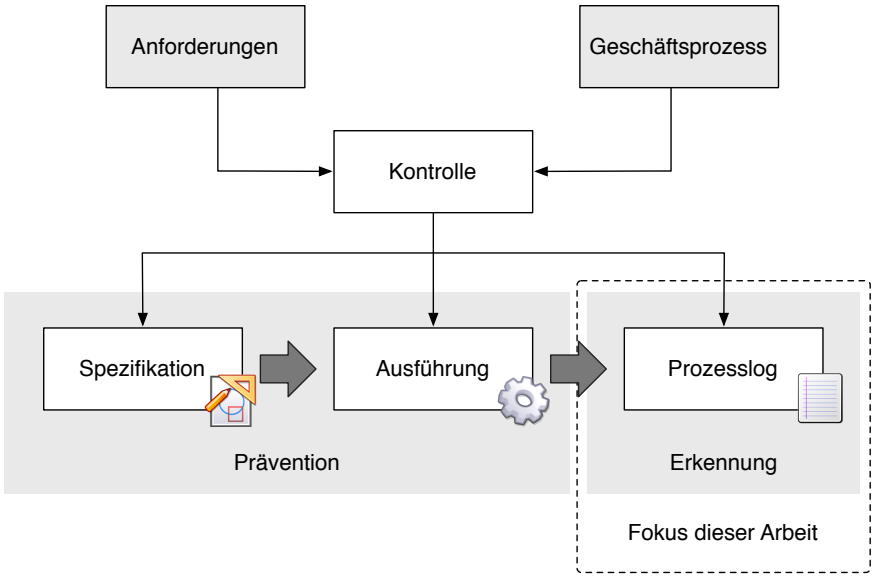


Abbildung 1.5.: Zeitpunkte der Prozesskontrolle.

möglichst präzise Spezifikation das Auftreten von Abweichungen minimiert und verbleibenden Restrisiken während der Ausführung durch geeignete Kontrollen begegnet. Angewandt auf das Lebenszyklusmodell für Geschäftsprozesse (siehe Abb. 1.2), betrifft dies die Designphase und schließt ein Prozessmonitoring während der Laufzeit ein. Nachgelagerte Analysen dienen dem Zweck, Erkenntnisse aus vergangen Prozessausführungen zu gewinnen und Aussagen über die Einhaltung von Vorgaben zu treffen. Dabei müssen Aufzeichnungen einzelner Ereignisse mit Prozessbezug aggregiert und interpretiert werden. Die Zielsetzung nachgelagerter Analysen ist nicht die Verhinderung von nicht konformem Verhalten, sondern dessen Erkennung.

Präventive Prozesskontrolle

Prozessspezifikationen beschreiben das geplante Verhalten von Geschäftsprozessen. Neben der generellen Zielsetzung enthält die Spezifikation Informationen über die Ausgestaltung einzelner Aktivitäten (z.B. Lieferantenauswahl), beteiligte Akteure (Personen oder Systemaufgaben), sowie Unternehmenseinheiten (z.B. Einkauf) und deren Zusammenhang. Die Berücksichtigung plausibler und korrekter Abläufe, valide Terminierung und die Einbeziehung verbindlicher (Sicherheits-)Anforderungen spielen dabei eine wesentliche Rolle. Grundsätzlich orientiert sich der Detailgrad von Prozessspezifikationen an der Bedeutung betreffender Prozesse für das Unternehmen (Kernprozesse bzw. monetärer Einflussfaktor).

Spezifikationen können in rein textueller Form vorliegen oder grafische Modellierung einschließen. Prozessmodelle enthalten hierbei zumindest den Kontrollfluss eines Prozesses (i.e. die Gesamtheit gültiger Aktivitätssequenzen), können aber abhängig von verwendeten Metamodellen auch weitere Aspekte einbeziehen (Datenverwendung und -fluss, Akteure etc.). Prozessspezifikationen sind nicht nur hilfreich für die unternehmensinterne

Kommunikation. In vielen Fällen kann eine adäquate Berücksichtigung von Bedrohungen schon durch die Überprüfung von Prozessspezifikationen und -modellen festgestellt werden.

Obwohl Spezifikationen maßgeblich für die tatsächliche Ausführung von Prozessen sind, können sie nicht in jedem Fall vollständig umgesetzt werden. Ist die Prozessausführung keiner automatisierten Steuerung unterworfen, ist nicht abschätzbar, inwiefern das tatsächliche Verhalten eines Prozesses seiner Spezifikation folgt. Diese kann dann lediglich als Handlungsempfehlung aufgefasst werden. Die Einhaltung verbindlicher Vorgaben hängt also von der Bereitschaft beteiligter Akteure ab, sich spezifikationsgemäß zu verhalten und nicht absichtlich vom geplanten Ablauf abzuweichen. Wird ein korrekter Prozessablauf nicht durch geeignete Steuerung oder sonstige Kontrollmechanismen sichergestellt, ist die Prozessspezifikation nicht vertrauenswürdig und kann nicht als Grundlage für Prozessanalysen dienen.

Doch auch im Fall automatisierter Geschäftsprozesse ist die alleinige Betrachtung der Prozessspezifikation für die Beurteilung der Regelkonformität nicht ausreichend[40]. Die Einhaltung von Vorgaben hängt von der Vollständigkeit des internen Kontrollsystems und der korrekten Implementierung von Einzelkontrollen ab, die ggf. aktiv in konkrete Abläufe eingreifen. Im Folgenden werden häufige Fehlannahmen des Geschäftsprozessmanagements beschrieben, die in der Praxis zu einem starken Fokus auf präventive Kontrollmechanismen führen und damit abweichendes Fehlverhalten begünstigen.

1. **„Die Implementierung von Prozessen erfolgt lückenlos."** Für die korrekte Implementierung von Geschäftsprozessen in der IT-Landschaft eines Unternehmens sind detaillierte, sprachlich eindeutige Spezifikationen erforderlich, die alle für den korrekten Ablauf notwendigen Prozessaspekte berücksichtigen. Dazu gehören Berechtigungen bzw. verbindliche Vorgaben für die Auswahl geeigneter Akteure, Datenverwendung und ggf. Vor-/Nachbedingungen für einzelne Ausführungsschritte oder Ressourcenzugriffe. Unvollständige oder fehlerhafte Spezifikationen erschweren die Umsetzung und führen im schlimmsten Fall zu mangelhafter Abdeckung realer Risiken oder der Existenz von Schwachstellen.

   Generell erfordert eine lückenlose Realisierung eines Prozesses die Abdeckung einzelner Erfordernisse durch geeignete Instrumente der IT. Prozessorientierte Informationssysteme mit geeigneten Steuerungsmechanismen (sog. PAIS) können z.B. dafür eingesetzt werden, bestimmte in der Spezifikation festgeschriebene Pfade zu erzwingen und so Anforderungen bzgl. des Kontrollflusses gerecht zu werden. Andernfalls muss dafür Sorge getragen werden, dass durch entsprechende Systemanpassungen eine Prozessausführung im Sinne der Spezifikation erreicht wird. Solche Maßnahmen sind meist mit hohem Aufwand verbunden, da Standardlösungen durch unternehmensspezifische Arbeitsvorgänge nicht ohne Weiteres anwendbar sind. Ob ein „Nachrüsten"[41] spezieller Erfordernisse im Zuge von Customizing-Projekten sinnvoll ist, hängt davon ab, ob der dadurch erzielte Nutzen in einem sinnvollen Verhältnis zu den damit verbundenen Kosten steht. Die Berücksichtigung von Anforderungen der Zugriffskontrolle oder Datensparsamkeit erfordert die Möglichkeit aktiver Eingriffe in die Arbeitsweise beteiligter Informationssysteme.

[40] Vgl. Müller et al. 2013.

Dabei müssen u.U. Berechtigungskonzepte unterschiedlicher Systeme kombiniert werden. Auch Kommunikationsprobleme und fehlende Möglichkeiten der IT, den Prozess wie geplant zu unterstützen (verschlüsselte Kommunikation, sichere Speicherung, zeitliche Steuerung von Abläufen), können dazu führen, dass Prozesse nicht spezifikationsgemäß umgesetzt werden.

2. **„Prozesse sind konstant."** Die einmalige Spezifikation und Umsetzung eines Prozesses wird realen Gegebenheiten nicht gerecht. Flexibilität und die Fähigkeit, sich schnell an neue Umstände anzupassen sind wesentliche Erfolgsfaktoren von Unternehmen. Sich verändernde Rahmenbedingungen (Marktgegebenheiten, neue Gesetzeslagen, Branchenstandards etc.) müssen adaptiert und in Prozessen realisiert werden. Der Erhalt der eigenen Wettbewerbsfähigkeit ist dabei eng verknüpft mit der kontinuierlichen Identifikation und dem Ausschöpfen von Optimierungspotentialen. Dem Umstand, dass Prozesse einer fortlaufenden Dynamik unterworfen sind, wird auch im Lebenszyklusmodell Rechnung getragen. Nicht dokumentierte Änderungen führen zu inaktuellen Prozessspezifikationen, die nicht länger als vertrauenswürdig gelten können.

3. **„Die Prozessausführung folgt der Spezifikation."** Eine vollständige Spezifikation von Geschäftsprozessen ist mit erheblichem Aufwand verbunden und nicht in jedem Fall ist es (wirtschaftlich) sinnvoll, jeden Spezialfall zu berücksichtigen und entsprechend automatisiert durchzusetzen. Stattdessen wird häufig nur das typische Prozessverhalten automatisiert. Ausnahmefälle können ohnehin nur schwer bestimmt, vorhergesagt oder gar verhindert werden. Im Zuge der Risikoakzeptanz werden Einzelrisiken bewusst hingenommen und nicht durch spezielle Mechanismen abgeschwächt. Für selten auftretende Spezialfälle wird meist auf Ad-hoc-Prozesse zurückgegriffen, deren genauer Ablauf nicht spezifiziert ist und von berechtigten Akteuren im Einzelfall bestimmt werden kann.

   Um einen gewissen Grad an Flexibilität zu erhalten und auch in Ausnahmefällen die Fortführung und Beendigung von Prozessen zu ermöglichen, wird deren Steuerung weniger restriktiv gestaltet oder an bestimmten Stellen innerhalb von Prozessen nicht regelkonformes Verhalten explizit toleriert. Die Notwendigkeit solcher Maßnahmen ergibt sich generell aus Notfallsituationen oder der Anforderung, schnell auf unerwartete Situationen zu reagieren, in denen die Fortführung des Prozesses höher gewichtet wird als seine Konformität. Als Beispiel kann die Durchführung lebensrettender medizinischen Handlungen in Krankenhäusern ohne explizite Berechtigung (z.B. aufgrund eines Herzstillstands) oder die Vermeidung des Stillstands von Fließbändern oder Fertigungsketten in der Industrie gelten. Nicht zu unterschätzen ist auch die Kreativität von Prozessbeteiligten, neue (effizientere) Wege der Prozessausführung zu finden und diese gemäß den technischen Gegebenheiten umzusetzen. Dabei kann es sich tatsächlich um ungenutztes Optimierungspotenzial, aber auch um gezielte Manipulation handeln.

[41] Im SAP-Umfeld kann dies z.B. durch den ergänzenden Einsatz einer Workflow-Engine geschehen, die einzelnen Prozessschritten SAP-Transaktionen zuordnet, sowie automatisch passende Bearbeiter auswählt und zur Erledigung auffordert.

4. **„Die Effektivität präventiver Kontrollen ist gesichert."** Bei der Spezifikation und Umsetzung von Prozessen muss im Rahmen einer Anforderungsanalyse nicht nur deren Ressourcen-, sondern auch deren Sicherheitsbedarf bestimmt werden, d.h. Risiken müssen identifiziert und beschrieben werden. Sind daraus abgeleitete Leitlinien für die Prozessausführung unvollständig, konfliktbehaftet oder werden relevante Risiken vernachlässigt oder nicht ausreichend durch adäquate Kontrollen abgedeckt, entstehen Schwachstellen, die (sicherheitskritische) Abweichungen begünstigen. Die Effektivität installierter Kontrollen zur Sicherstellung regelkonformen Prozessverlaufs kann grundsätzlich nicht im Voraus bestimmt, sondern muss stets nachträglich mithilfe von Stichproben oder Methoden der Massendatenanalyse ermittelt werden.

   Fehlkonfigurationen und -funktionen, sowie Schwachstellen und Sicherheitslücken beteiligter IT-Systeme können dazu führen, dass Prozesse nicht zielgerichtet ausgeführt oder von Beteiligten zu wirtschaftskriminellen Zwecken missbraucht werden (z.B. durch nicht erfolgte oder vorgetäuschte Identitätsbestätigungen).

Methoden zur Prüfung von Geschäftsprozessen während der Laufzeit leisten einen wichtigen Beitrag zur Sicherheit von Geschäftsprozessen. Ausführungsmonitore, die Prozessausführungen in Echtzeit überwachen, können diese bspw. unterbrechen, wenn deren Fortführung zu Verletzungen verbindlicher Vorgaben führt. Diese Vorgehensweise ist in realen Umgebungen allerdings nicht immer sinnvoll, da ein Abbruch ggf. negative Auswirkungen auf parallel laufende und davon abhängige Prozesse hat (vgl. *ripple effect*[42]) oder ein Abbruch generell „teurer" als eine fehlerhafte Fortführung ist (z.B. bei zeitkritischen Abläufen). Neben risikobasierten Ansätzen zur Entscheidung über erzwungene Abbrüche[43], hat sich das *break-glass* Prinzip als Alternative etabliert[44]. Dabei wird der Ablauf zwar zunächst unterbrochen, kann aber mit der Konsequenz intensivierter Protokollierung folgender Aktivitäten (zur Sicherstellung der Zurechenbarkeit) dennoch fortgeführt werden. Ursprünglich zur Vermeidung gesundheitlicher Schäden im Fall medizinischer Notfälle gedacht, kann dieses Prinzip auch effektiv zur Abwendung finanziellen oder materiellen Schadens in anderen Domänen angewandt werden.

Aufgrund des damit verbundenen Aufwands und der teils negativen Auswirkungen auf die Prozessausführung, kommen präventive Kontrollen nicht immer zum Einsatz. Das detaillierte Protokollieren von Datenbank-Änderungen zu Sicherheitszwecken etwa, kann zu Performanceeinbußen bei der Ausführung von Prozessen führen und wird deshalb nicht selten im Zuge von Kosten/Nutzen-Abwägungen deaktiviert. Dabei wird das Risiko abweichender Ausführungspfade akzeptiert und die Konformitätsprüfung bewusst in nachgelagerte Analysen verschoben. Die Dynamik, die Prozesse während ihrer Ausführung aufgrund der eingeräumten Flexibilität und wiederholter Anpassungen entwickeln, kann nicht im Voraus abgeschätzt, sondern muss aus einem ex-post Standpunkt beurteilt werden. Der Vergleich abgeschlossener Prozessausführungen mit der Spezifikation oder dokumentierten Anpassungsoperationen liefert Hinweise auf die Konformität von Geschäftsprozessen. Dadurch lässt sich auch feststellen, inwieweit die Spezifikation korrekt umgesetzt wurde. Ob installierte Kontrollen erfolgreich umgangen wurden, lässt

[42] Vgl. Fdhila et al. 2012.
[43] Vgl. Ni et al. 2010; Accorsi, Sato et al. 2008.
[44] Vgl. Brucker et al. 2009.

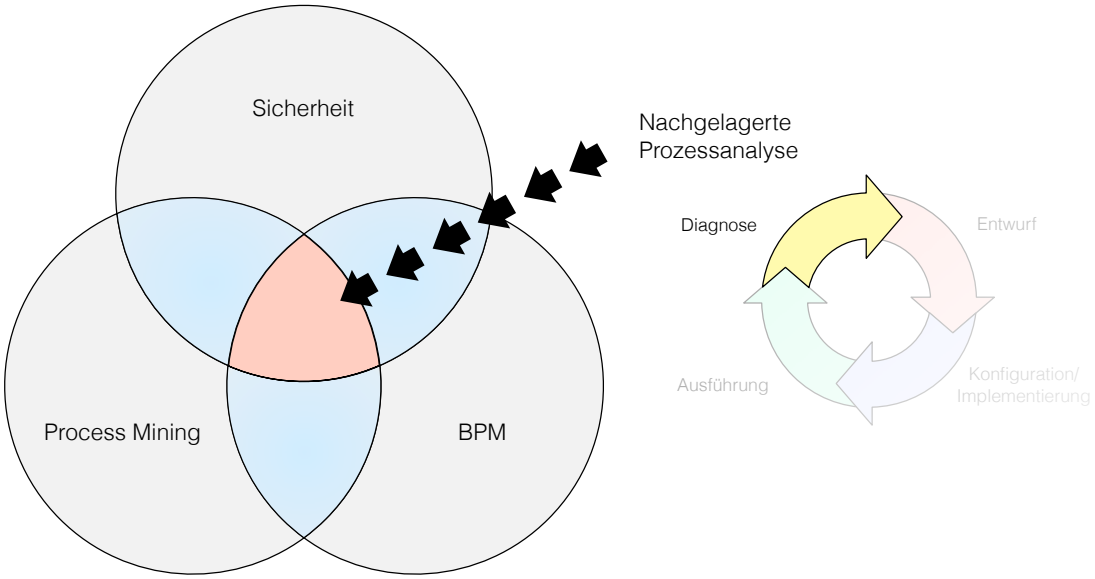


Abbildung 1.6.: Thematische Einordnung der Arbeit.

sich ebenfalls nur durch detaillierte Analysen von Aufzeichnungen (Prozesslogs) beantworten.

1.5 Beitrag und Aufbau der Arbeit

Diese Arbeit betrachtet die Sicherheit von Geschäftsprozessen aus einer nachgelagerten Perspektive. Für die Beurteilung der Prozesskonformität wird die Verwendung der Process Mining-Technologie vorgeschlagen, die aufgrund der Bereitstellung von Methoden zur Analyse aufgezeichneter Prozessdaten für diesen Zweck besonders geeignet ist. Die Korrelation während der Ausführung generierter Datenartefakte von Prozessen und deren Interpretation auf Prozessebene ermöglicht die Berücksichtigung aller relevanten Prozessaspekte und bietet eine solide Grundlage für sicherheitsorientierte Analysen.

Der Beitrag der Arbeit besteht zum einen in der systematischen Evaluierung des Mehrwerts von Process Mining innerhalb nachgelagerter, sicherheitsorientierter Prozessanalysen und andererseits in der Entwicklung von neuen Methoden zur Visualisierung tatsächlichen Prozessverhaltens auf Basis von Prozesslogs. Durch die explizite Berücksichtigung sicherheitsspezifischer Erfordernisse erreichen die entwickelten Ansätze eine hohe Übereinstimmung zwischen rekonstruierten Modellen und Prozesslogs. Dadurch wird eine Grundlage für die Erlangung eines detaillierten Prozessverständnisses und die Ableitung faktenbasierter Aussagen hinsichtlich des Zustandekommens von Abweichungen im Rahmen nachgelagerter Prozessanalysen geschaffen. Thematisch lässt sich die Arbeit im Überschneidungsbereich von BPM, Sicherheit und Process Mining verorten, wobei die Problemstellung und Anforderungsanalyse aus der Perspektive der Sicherheit in Geschäftsprozessen erfolgt (siehe Abb. 1.6). Im Einzelnen leistet die Arbeit Beiträge in folgenden Bereichen:

- Mit der systematischen Betrachtung von Process Mining vor dem Hintergrund sicherheitsorientierter Prozessdiagnosen wird das Potential dieser Technologie im Rahmen nachgelagerter Prozessanalysen erörtert.
- Die GENET$^+$-Methode zur Rekonstruktion hochpräziser Prozessmodelle aus Prozesslogs trägt effektiv zur Erlangung eines Verständnisses tatsächlicher Prozessausführung und dem Zusammenhang einzelner Aktivitäten im Rahmen von Prozessanalysen bei.
- Mithilfe des GENET*-Ansatzes zur kombinierten Kontroll-/Datenfluss-Rekonstruktion können erstmals Modelle realen Prozessverhaltens gebildet werden, die mehrere Aspekte der Prozessausführung gleichzeitig und in präziser Form darstellen.
- Das APClustering-Verfahren zur Erkennung dynamischer Ausführungscharakteristika kann dazu dienen die Veränderung von Geschäftsprozessen im zeitlichen Verlauf intuitiv nachzuvollziehen, sowie die Präzision rekonstruierter Modelle und damit deren Aussagekraft zu erhöhen.
- Die Weiterentwicklung des IFnet Formalismus und dessen Verwendung als Ziel-Metamodell rekonstruierter Modelle schafft eine Grundlage für weiterführende Analysen auf rekonstruierten Modellen.

Die Struktur der Arbeit kann Abb. 1.7 entnommen werden. Abschnitte mit gestrichelter Umrandung sind nicht notwendig für das Verständnis der Kerninhalte aber hilfreich zur Einordnung und Ergänzung. Die Arbeit beginnt in Abschnitt 2 mit einer Betrachtung des Potentials von Datenanalysen zur Beurteilung der Sicherheit von Geschäftsprozessen. Dabei wird Process Mining als probates Mittel in Zusammenhang mit der sicherheitsorientierten Analyse von Prozessdaten identifiziert. Nach der Erläuterung von Möglichkeiten zum Einsatz dieser Technologie innerhalb nachgelagerter Prozessanalysen (Abschnitt 2.2.1 und Abschnitt 2.2.2) wird der Forschungsbedarf im Bereich der Rekonstruktion (Visualisierung tatsächlichen Prozessverhaltens) in Abschnitt 2.3 offen gelegt und sicherheitsorientierte Anforderungen für die Prozessrekonstruktion abgeleitet. Die im Rahmen dieser Arbeit verfolgte Problemstellung ergibt sich aus identifizierten Defiziten der Process Mining-Technologie (Abschnitt 2.4).

Der Hauptteil dieser Arbeit teilt sich in zwei Bereiche auf. Der erste Teil betrachtet die Rekonstruktion präziser Prozessmodelle und betrachtet und bewertet dafür zunächst eingehend existierende Verfahren (Abschnitt 3). Ein kurzer Abriss zur Entwicklung und Funktionsweise einer besonders geeigneten Familie von Ansätzen in Abschnitt 4.2 und Abschnitt 4.3 leitet Abschnitt 4.5 ein, in dem das Rekonstruktionsverfahren GENET$^+$ vorgestellt wird. Dessen Adäquanz wird in Abschnitt 4.6 formal und empirisch nachgewiesen. Der in Abschnitt 5 vorgeschlagene Ansatz GENET* stellt eine Ergänzung dar, die durch die zusätzliche Betrachtung von Datenflüssen eine kombinierte Rekonstruktion erlaubt. Abschnitt 5.1 enthält zudem einen Exkurs zum IFnet Formalismus, der als Metamodell genutzt wird und ebenfalls eigener Vorarbeiten entspringt.

Der zweite Teil (Abschnitt 6) widmet sich der Behandlung der Prozessdynamik und schlägt nach der Vorstellung verwandter Arbeiten (Abschnitt 6.1) in Abschnitt 6.2 das APClustering-Verfahren zur Erkennung unterschiedlicher Prozessmodelle im zeitlichen Verlauf vor, welches in Abschnitt 6.3 empirisch evaluiert wird. In Abschnitt 7 wird nach

einer Zusammenfassung die Bedeutung der Arbeit in Form von Beitrag und Anwendungsmöglichkeiten erläutert. Die Arbeit wird in Abschnitt 7.2 schließlich durch die Darlegung denkbarer Erweiterungen abgerundet.

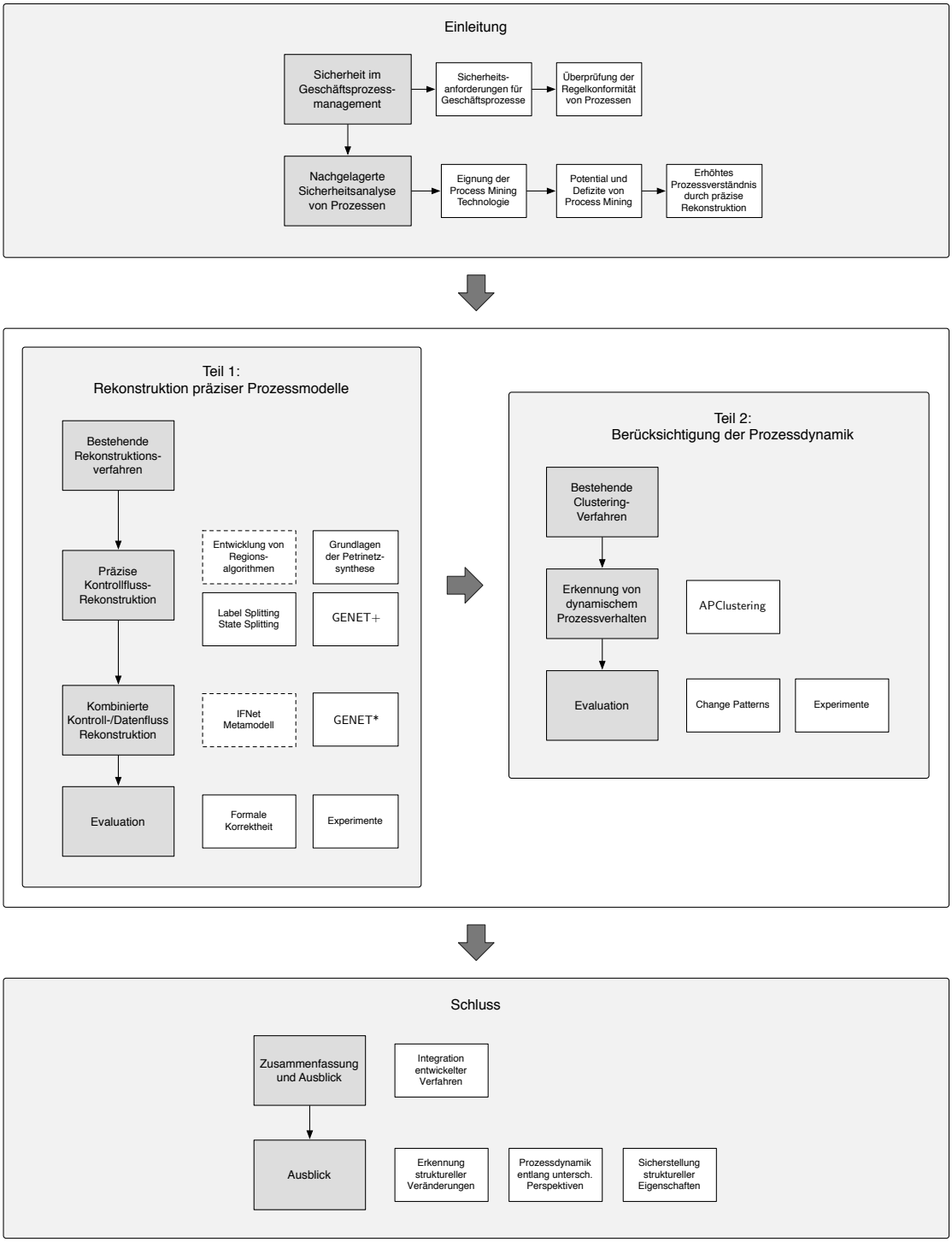


Abbildung 1.7.: Struktur der Dissertation.

Kapitel 2

Nachgelagerte Sicherheitsanalyse von Geschäftsprozessen

Dieses Kapitel hebt die Bedeutung nachgelagerter Analysen und insbesondere Methoden der Massendatenanalyse für die Beurteilung der Sicherheit von Geschäftsprozessen hervor. Dabei wird die Process Mining-Technologie als methodischer Gegenstand der Arbeit, sowie der Zusammenhang und Überschneidungsbereich von Sicherheit, Geschäftsprozessmanagement und Process Mining als Betrachtungsgegenstand der Arbeit motiviert. Nach einer detaillierten Betrachtung der Einsatzmöglichkeiten konkreter Verfahren in Abschnitt 2.2.1 und Abschnitt 2.2.2 wird das Potential von Process Mining in Abschnitt 2.3 zusammengefasst. Bestehende Defizite der Technologie, die ihre Nutzbarkeit innerhalb von Prozessanalysen beschränken, werden in Abschnitt 2.4 herausgearbeitet. Die Problemstellung dieser Arbeit basiert auf dem daraus abgeleiteten Forschungsbedarf und bezieht sich auf die Erweiterung des Standes der Technik um Methoden, die Prozessrekonstruktion innerhalb nachgelagerter Sicherheitsanalysen nutzbar machen.

2.1 Zuverlässige Nachweise durch nachgelagerte Prozessanalysen

Die Integration von Komponenten zur nachträglichen Auswertung von Prozessen prägen den BPM-Teilbereich BPA (Business Process Analysis), der sich auf die Analyse von Geschäftsprozessen bezieht und innerhalb der letzten Jahre immer stärker an Bedeutung gewonnen hat[1]. Darunter fallen hauptsächlich Methoden aus den Bereichen Reporting und Business Intelligence (BI), aber auch Audits des internen Kontrollsystems (IKS-Audit), die z.B. aufgrund von SOX-Anforderungen in regelmäßigen Abständen durchgeführt werden und die Wirksamkeit des IKS evaluieren. Die Überprüfung interner Kontrollsysteme ist Aufgabe der internen Revision, die abhängig von Branche, Unternehmensgröße und länderspezifischen Regelungen selbst wiederum in periodischen Abständen von externen Prüfern auditiert wird. Audits können die IT-Infrastruktur betreffen (IT-Audits) oder

[1] Vgl. Sinur 2002.

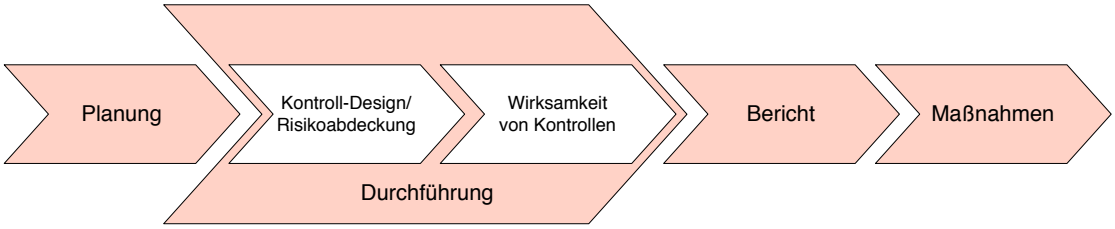


Abbildung 2.1.: Ablauf von Prozessaudits.

die unterstützte Geschäftstätigkeit selbst. In beiden Fällen gibt es verbindliche Standards zu deren Durchführung[2]. Das übergeordnete Ziel besteht in der Vermeidung bzw. frühzeitigen Erkennung krimineller Aktivitäten.

Zeiträume und Periodizität von Prozessaudits werden typischerweise von Auditplänen bestimmt; ihr Ablauf ist im Wesentlichen durch vier Schritte gekennzeichnet (siehe Abb. 2.1). In der Planungsphase wird zunächst die Zielsetzung und der Anwendungsbereich (einzelne Prozesse, Organisationseinheiten etc.) des Audits festgelegt und auf Grundlage relevanter Qualitätsanforderungen, interner Arbeitsanweisungen und Normen ein Fragenkatalog angefertigt. Sofern frühere Audits einen Handlungsbedarf offengelegt haben, finden auch Fragestellungen zur Beseitigung herausgearbeiteter Mängel Berücksichtigung.

Die Durchführung ist gekennzeichnet durch die Überprüfung, ob relevante Prozessrisiken mit adäquaten Kontrollen abgedeckt und ob diese effektiv sind. Dafür benötigen Auditoren ein tiefgreifendes Verständnis des geplanten Prozessablaufs und potentieller Risiken. Die Sammlung relevanter Daten aus diversen Informationsquellen, sowie deren Aufarbeitung und Bewertung im Hinblick auf konkrete Fragestellungen ist notwendig. Wichtige Informationsquellen sind Prozessdokumentationen, Arbeits- und Verfahrensanweisungen, sowie Dokumentationen des Risikomanagements und der Kontrollmechanismen. Typischerweise wird zusätzlich mithilfe von Interviews auf das Wissen von Prozessverantwortlichen und -beteiligten zurückgegriffen. Eine Kontrolle gilt als angemessen, wenn deren Wirksamkeit nachvollzogen werden kann und wenn sie geeignet ist, das entsprechende Risiko nachvollziehbar zu mindern. Zentrale Fragen betreffen den ausführenden Akteur (Systemkontrolle, qualifizierte Mitarbeiter), modale Eigenschaften (Zeitpunkt und Periodizität) und Voraussetzungen für die Durchführung der Kontrolle (Stelle im Prozess, datenbezogene Vorbedingungen).

Die Arbeitsweise von Kontrollen kann durch deren explizite Durchführung anhand speziell vorbereiteter Testläufe innerhalb beteiligter Systeme nachvollzogen werden. Zur Überprüfung der Wirksamkeit von Kontrollen kann die Dokumentation erfolgter Kontrolldurchführungen überprüft werden. Dabei kann es sich bspw. um geleistete Unterschriften von beteiligten Personen, digitale Aufzeichnungen automatisch erfolgter Kontrollhandlungen oder den Nachvollzug vollständiger Prozessausführungen handeln. Aufgrund der meist stichprobenbasierten Vorgehensweise bei der Überprüfung einzelner Prozessabläufe kann

[2]SAS 70 (Vgl. AICPA SAS70 1992) bzw. SSAE 16 (Vgl. AICPA SSAE16 2011) für unabhängige Auditoren, Richtlinien der IIA für Innenrevisoren

jedoch stets nur eine unvollständige Sicht auf die Prozessausführung[3] gewonnen und die Kontroll-Effektivität sowie Regelkonformität lediglich approximiert werden.

Belastbare Analyseergebnisse durch Massendatenanalyse

Die Fähigkeit moderner Informationssysteme, Aktivitäten betrieblicher Abläufe lückenlos zu dokumentieren, ermöglicht den Einsatz dedizierter Software für die Unterstützung von Auditprozessen, sog. CAATs (Computer Aided Audit Tools)[4]. Die Schaffung einer globalen Sicht auf Daten unterschiedlicher Informationssysteme innerhalb von Unternehmen mithilfe von Integrationsansätzen wie ETL[5] schafft dabei eine belastbare Ausgangsbasis für das Berichtswesen, Controlling und Qualitätsmanagement. Die Notwendigkeit der Betrachtung digitaler Spuren von Prozessen als Ausgangsbasis für die Analyse ergibt sich unmittelbar aus der Anforderung an Audits, für die Ableitung von Aussagen zur Sicherheit und der Feststellung der Regelkonformität von Systemen und Prozessen stets die verlässlichste verfügbare Datenbasis zu wählen[6].

Aus der Perspektive der Sicherheit sind diesbezüglich hauptsächlich Methoden interessant, die eine erschöpfende Analyse der Spuren zulassen, welche ein Prozess während seiner Laufzeit hinterlässt. Auf diese Weise kann die Aussagekraft von Analyseergebnissen im Vergleich zur Durchsicht von Dokumenten zum Prozessablauf und der Funktionsweise von Kontrollen erheblich erhöht und eine belastbare Basis für faktenbasierte Aussagen hinsichtlich der Einhaltung von Sicherheitsanforderungen geschaffen werden. Statistische Methoden zur Verdichtung erhobener Daten, die durch Filterung, Aggregation, Sortierung oder grafischer Aufbereitung Wesentliches erfassbar machen, können Auditoren zusätzlich dabei unterstützen, das Ist-Prozessverhalten und dessen Bezug zum Soll-Verhalten zu erfassen. Für die Evaluation bestehender Spezifikationen und die Ableitung möglicher Verbesserungen sind solche Betrachtungen essentiell.

Eine grundlegende Voraussetzung für die Anwendung von Methoden der Massendatenanalyse ist die Verlässlichkeit protokollierter Prozessdaten i.S. der Manipulationssicherheit. Eine Voraussetzung dafür ist die Vertrauenswürdigkeit der genutzten IT-Infrastruktur, die anhand spezieller Kriterien zur Beurteilung und Handlungsempfehlungen für die Gewährleistung der Sicherheit von Informationssystemen (z.B. ITSEC[7],

[3] Vgl. Carlin et al. 2007.
[4] Vgl. Sayana 2003.
[5] ETL (Extrakt, Transform, Load) steht für einen dreistufigen Daten-Integrationsvorgang, der im Datenbankbereich hauptsächlich zur Realisierung von Data-Warehousing eingesetzt wird. Daten unterschiedlicher Quellsysteme werden extrahiert und in einem einheitlichen Format zusammengeführt. Ein Data-Warehouse kann als integrierte Datenbasis aufgefasst werden, die Mehrwertdienste für verschiedene Anwendungen (Berichterstattung, Performanz-Messung etc.) bereitgestellt.
[6] Vgl. ITAF 2011.
[7] „Die europäischen ITSEC(Information Technology Security Evaluation Criteria) Kriterien wurden als harmonisierte Kriterien der Länder Großbritannien, Frankreich, Niederlande und Deutschland erarbeitet [...] und legen [...] Funktionsklassen mit Sicherheitsanforderungen für spezifische Anwendungsklassen fest.“ (Vgl. Eckert 2003)

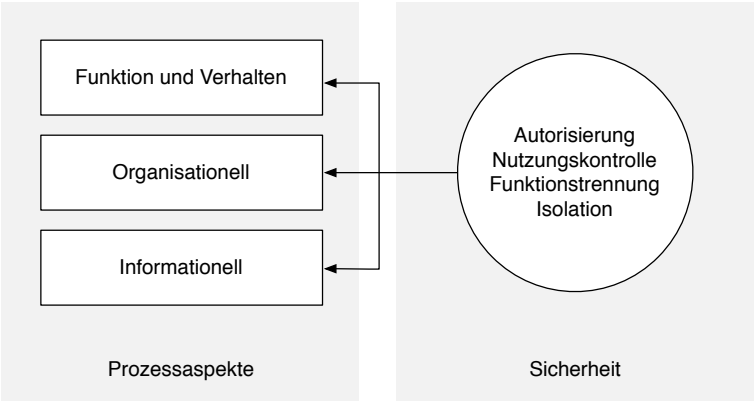


Abbildung 2.2.: Rahmen für die Sicherheitsanalyse von Geschäftsprozessen.

ISO/IEC 2700-Familie[8] oder IT-Grundschutz nach BSI[9]) fest- und sichergestellt werden kann.

Da sich Sicherheitsanforderungen auf unterschiedliche Aspekte der Prozessausführung beziehen können (vgl. Abschnitt 1.3), stellt die Berücksichtigung von Autorisierung, Nutzungskontrolle, Funktionstrennung und Isolation entlang der in Abb. 2.2 visualisierten Aspekte eine zentrale Voraussetzung für eine ganzheitliche, nachgelagerte Sicherheitsanalyse von Geschäftsprozessen dar.

Zur Erstellung von Statistiken und Kennzahlen kommen häufig Data Mining[10] oder Analyseverfahren wie OLAP (Online Analytical Processing) zum Einsatz. Dabei werden Daten in multidimensionale Matrizen organisiert, wobei Dimensionen z.B. für unterschiedliche Produkte, Geschäftsbereiche, Verkaufsbereiche etc. stehen können. Auf diese Weise können Aggregate (z.B. „Durchschnittlicher Umsatz in Bereich A für Produkt B“) effizient berechnet werden. Mithilfe von Data Mining lassen sich Zusammenhänge in Daten erkennen, was zu einem erhöhten Verständnis der Datenbasis und einer besseren Einschätzung des Optimierungs-Potentials führt. Es existiert eine Reihe von Data Mining Anwendungen, die speziell auf die Erfordernisse von Risikomanagement und Auditierung zugeschnitten sind (vgl. CaseWare IDEA®[11], ACL[12]). Neben vielfältigen Werkzeugen zur Darstellung spezifischer Kennzahlen sind Methoden der *Outlier Detection* zur Identifikation atypischer Ausprägungen statistischer Variablen im Hinblick auf die Sicherheit von besonderem Interesse.

Auf diese Weise können Einzelereignisse oder Fragmente in Datenbanksystemen in Beziehung gesetzt werden, deren Bezug zu konkreten Prozessaktivitäten gemäß Spezifikation und einzelner Prozessausführungen (Instanzen) bleibt jedoch verborgen. Die Beantwortung von Fragestellungen bzgl. des Prozessablaufs und insbesondere der Verifikation von

[8] Vgl. ISO/IEC 27001 2013.
[9] Das Bundesamt für Sicherheit in der Informationstechnik stellt Leitfäden für ganzheitliche Konzepte für Informationssicherheit zur Verfügung. Speziell kleine und mittlere Unternehmen sollten dadurch unterstützt werden, den wachsenden Anforderungen an die Informationssicherheit gerecht zu werden.
[10] Vgl. Hand et al. 2001.
[11] Vgl. Caseware Analytics 2014.
[12] Vgl. ACL Services Limited 2014.

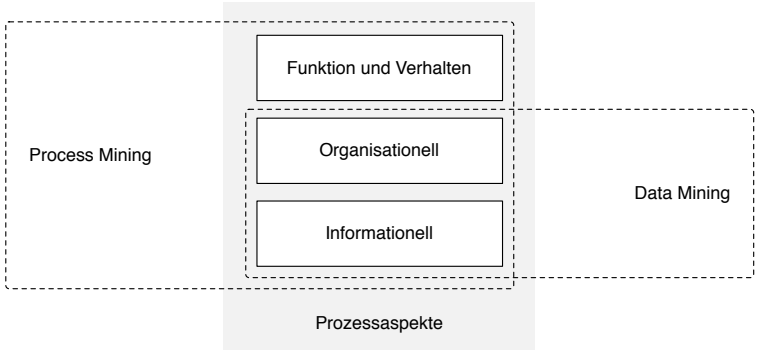


Abbildung 2.3.: Datenanalyse auf unterschiedlichen Ebenen.

Sicherheitseigenschaften auf Prozessebene sind daher nicht oder nur mit erheblichem Aufwand (z.B. durch entsprechend gestaltete SQL-Abfragen) beantwortbar. Beispiele für ablaufbezogene Fragestellungen sind:

- „Erfolgt für jede Eingangsrechnung eine Bestätigung des Kostenstellenleiters?“
- „Gibt es Fälle, in denen ein Mitarbeiter in mehr als 10 Prozessinstanzen gleichzeitig aktiv ist?“
- „Was ist die durchschnittliche Zahl von Aktivitäten pro Ausführung?“

Die Entwicklung von CAATs, die Data Mining Werkzeuge um eine Prozessperspektive erweitern, um so komplexe Analysen auf höherer Ebene durchführen und Kennzahlen für die Konformität tatsächlichen Prozessverhaltens ableiten zu können, zeigt, dass die Einschränkung von Methoden zur Analyse digitaler Spuren von Prozessen, die rein auf Datenebene arbeiten, auf industrieller Seite erkannt wurde[13]. Aus theoretischer Perspektive lassen sich solche Methoden unter dem Oberbegriff *Process Mining* zusammenfassen. Process Mining kann im weitesten Sinne als „Data Mining zugeschnitten auf Prozesse“ verstanden werden und ist durch eine explizite Prozesssicht gekennzeichnet, die neben dem informationellen und dem organisationellen auch die Betrachtung des funktionalen und verhaltensbezogenen Aspekts von Prozessen ermöglicht (vgl. Abb. 2.3). Die Erschließung der Prozessebene entsteht dabei durch die Interpretation von Datensätzen als Einzelereignisse im Prozessverlauf, die jeweils für das Stattfinden einer Aktivität stehen und deren Korrelation anhand der Zugehörigkeit zu einzelnen in sich abgeschlossenen Prozessinstanzen. Ausgangsbasis für Ansätze aus diesem Bereich ist stets ein Prozesslog, der alle Ereignisse eines konkreten Prozesses enthält.

2.2 Process Mining als Werkzeug der Sicherheitsanalyse

Process Mining ist eine Brücke zwischen der Schicht der Informationsverarbeitung und des Geschäftsprozessmanagements. Pionierarbeiten dieser jungen Disziplin[14] sind stark

[13] Perceptive Process Analytics, QPR Process Analyzer, Disco (Vgl. Günther und Rozinat 2012)

[14] Erwähnenswert sind in diesem Zusammenhang vor allem die Arbeiten von Agrawal et al. (Vgl. Agrawal et al. 1998), Datta (Vgl. Datta 1998), sowie Cook und Wolf (Vgl. Cook und A. L. Wolf 1998b).

geprägt von Ansätzen zur Rekonstruktion von Kontrollflussmodellen (*Process Discovery*). Traditionell liegt der Schwerpunkt auf der Rekonstruktion strukturierter Prozessmodelle, die das tatsächliche Prozessverhalten „sichtbar“ machen[15]. Mit der Entwicklung von Mechanismen zur Überprüfung der Konformität stattgefundener Prozessausführungen mit de-jure Modellen oder formalisierten Anforderungen[16] (*Conformance Checking*), wurde ein weiterer Bereich erschlossen, der eine Anwendung von Process Mining für nachgelagerte Prozessanalysen nahe legt[17]. Während Conformance Checking direkt zur Verifikation von Sicherheitseigenschaften genutzt werden kann, liegt das Potential von Process Discovery in der intuitiven Erfassbarkeit tatsächlichen Prozessverhaltens, was die Identifikation von Differenzen zum geplanten Prozessverhalten vereinfacht. Im Folgenden wird detailliert auf Process Mining und den Nutzen konkreter Ansätze für die Beurteilung der Sicherheit von Geschäftsprozessen eingegangen.

Process Mining

Ansätze aus dem Bereich des Process Mining werden gemeinhin in folgende drei Bereiche untergliedert:

1. **Rekonstruktion (Process Discovery)** Bei der Prozessrekonstruktion werden aufgezeichnete Prozessinstanzen genutzt, um strukturierte Prozessmodelle zu bilden. Dadurch kann das reale Prozessverhalten in einer kompakten Form repräsentiert werden (*de-facto* Modelle), wodurch die Evaluierung von de-jure Modellen erleichtert wird.

2. **Konformitätsprüfung (Conformance Checking)** Ausgangsbasis für diese Form der Analyse stellen Prozessmodelle für das idealisierte Prozessverhalten oder mathematische Beschreibungen einzuhaltender Anforderungen dar. Durch eine Überprüfung auf Abweichungen des tatsächlichen Prozessverhaltens kann festgestellt werden, ob vorgeschriebene Prozessschritte (z.B. zur Durchsetzung von Freigaberichtlinien) umgangen oder Anforderungen bzgl. der Funktionstrennung (z.B. 4-Augen-Prinzip) eingehalten wurden.

3. **Optimierung (Enhancement)** Das Ziel der Prozesserweiterung besteht darin, Ergebnisse aus den Bereichen Prozesskonformität und Rekonstruktion dazu zu nutzen, de-jure Modelle anzupassen und bzgl. Performanz (Durchlaufzeiten oder Ressourcenauslastung) zu optimieren. Dabei können Gegebenheiten identifiziert werden, die sich negativ auf die Gesamtperformanz eines Prozesses auswirken, z.B. bestimmte Pfadcharakteristika (Pfade für Sonderfälle involvieren deutlich mehr Aktivitäten), temporale Eigenschaften (längere Laufzeiten in Urlaubszeit) oder organisationelle Zusammenhänge (Abteilung B benötigt für denselben Ablauf doppelt so lange wie Abteilung A).

Trotz ähnlicher Terminologie bezieht sich der Teilbereich der Konformitätsprüfung nicht auf die Sicherheit (Regelkonformität) von Geschäftsprozessen im Sinne der Zielsetzung

[15] Vgl. Dongen, Alves de Medeiros et al. 2009.
[16] Vgl. Cook und A. L. Wolf 1999.
[17] Vgl. Aalst, Hee et al. 2010.

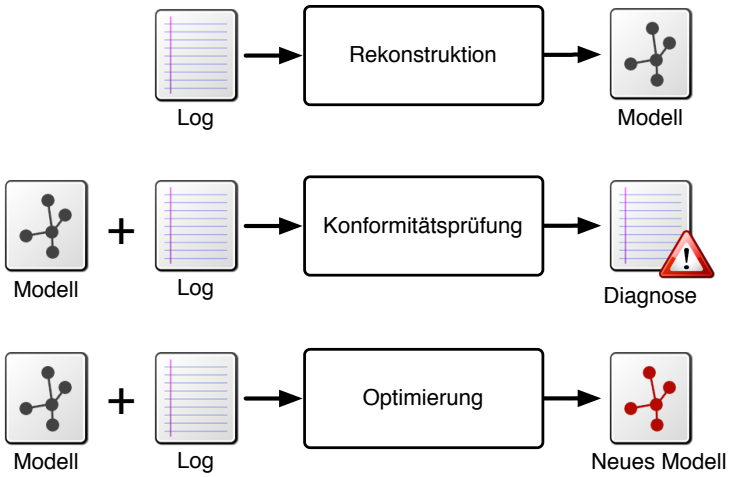


Abbildung 2.4.: Process Mining.

dieser Arbeit. Dennoch eignen sich Ansätze dieses Bereiches zur Feststellung der Einhaltung verbindlicher Prozessvorgaben (siehe Abschnitt 2.2.2).

Für die Prozessbetrachtung und die Kategorisierung von Ansätzen werden neben den drei Bereichen aus Abb. 2.4 zusätzlich *vier Perspektiven* betrachtet[18]. Im Zentrum der *Kontrollfluss-Perspektive* stehen Prozessaktivitäten, deren Unterscheidung, Anordnung und Zusammenhang. Ansätze der Prozessrekonstruktion in dieser Perspektive generieren Modelle, die den Kontrollfluss eines Prozesses auf Basis real aufgetretener Ausführungen darstellen. Die *organisationelle Perspektive* betrachtet Akteure und deren Interaktion. Ansätze dieser Perspektive identifizieren z.B. Rollen oder Befugnisse oder visualisieren Arbeitsabläufe i.S. der Kooperation von Individuen oder Abteilungen. Dadurch kann ein Verständnis über die Unterteilung und Delegation von Aufgaben erlangt werden. Die *Fall-Perspektive* konzentriert sich auf Eigenschaften von Prozessinstanzen, also in sich abgeschlossenen Prozessabläufen. Dabei kann es sich bspw. um Werte von Datenelementen innerhalb eines Falls oder die Frequentierung bestimmter Aktivitäten handeln. Zeitliche Aspekte werden in der *temporalen Perspektive* erfasst. Extrahierte Modelle dieser Perspektive stellen z.B. die mittlere Dauer von Prozessaktivitäten dar.

Die Sammlung und Konsolidierung prozessrelevanter Information muss in einem vorgelagerten Datenerhebungsprozess stattfinden. Dieser Prozess ist vom Unternehmenskontext und den Anforderungen an die Analyse abhängig. Die korrekte Erhebung und Aggregation relevanter Prozessdaten ist wesentlich für die Schaffung einer verlässlichen Datenbasis und stellt in praktischen Anwendungsfällen die zentrale Herausforderung dar. Während mittlerweile Informationssysteme mit expliziter Prozessorientierung (PAIS) am Markt erscheinen, die den Ablauf von Geschäftsprozessen anhand zuvor definierter Prozessmodelle steuern[19], ist die tabellenorientierte und formulargesteuerte Sichtweise auf Unternehmensabläufe immer noch Standard und macht die Extraktion von relevanten Daten für den jeweils betrachteten Prozess aus den beteiligten Transaktionssystemen und deren Beschreibung in Form von Prozessinstanzen notwendig.

[18] Vgl. Aalst 2011.
[19] Vgl. Dumas, Aalst et al. 2005.

Ereignisse innerhalb von Prozesslogs stehen in Relation zu Aktivitäten im zugrundeliegenden Prozessmodell. Jedes Ereignis ist zudem einer Instanz zugeordnet. Abhängig vom Detailgrad aufgezeichneter Prozessdaten können Prozesslogs zusätzliche Informationen zu ausführenden Subjekten, den Zeitstempel der Ausführung von Aktivitäten oder Charakteristika im Prozess verarbeiteter Datenobjekte umfassen (vgl. Tab. 2.1). Die vorliegende Arbeit abstrahiert von der notwendigen Datenerhebung und geht von der Existenz eines vertrauenswürdigen Prozesslogs aus, der relevante Prozessinformation in unverfälschter Form (Integrität) und in zeitlich korrekter Reihenfolge enthält.

Zeitstempel	**Instanz**	**Aktivität**	**Akteur**	**Attribut**
2014-07-01 09:24:11	1	Bedarfserfassung	Nutzer 1433	-
2014-07-02 14:34:21	1	Bestellnummer anlegen	System	Belegnr.: 23496347
2014-07-04 17:12:47	2	Prüfung Wareneingang	Nutzer 7235	Belegnr.: 18238433

Tabelle 2.1.: Schematische Darstellung eines Prozesslogs.

Während Conformance Checking und Process Discovery eine Fülle unterschiedlicher Ansätze zur Extraktion von Information aus Prozesslogs zur Verfügung stellen, bezieht sich Enhancement auf die Frage, inwiefern aus dieser Information Erkenntnisse zur Prozessoptimierung abgeleitet werden können. Typischerweise werden dafür keine neuen Verfahren vorgeschlagen, sondern diskutiert, wie bestehende Verfahren angewandt und kombiniert werden können. Im Folgenden werden konkrete Process Mining Verfahren vorgestellt, die sich für einen Einsatz in nachgelagerten, sicherheitsorientierten Prozessanalysen eignen. Dabei wird insbesondere auf deren Unterstützung innerhalb der Process Mining Werkzeuge ProM[20] und Disco[21], sowie des Analysetools SWAT[22] eingegangen.

2.2.1 Process Discovery: Rekonstruktion von Modellen tatsächlichen Prozessverhaltens

Methoden der Prozessrekonstruktion sind hilfreich, wenn entweder keine Prozessmodelle für den geplanten Prozessverlauf existieren oder durch die Erstellung von de-facto Modellen auf Basis von Prozesslogs ein intuitives Verständnis des tatsächlichen Prozessverhaltens erlangt werden soll. Dabei kann entweder das gesamte Verhalten betrachtet werden oder Mengen von Pfaden, die aus Analysesicht besonders interessant sind.

Im Folgenden werden mögliche Anwendungen der Prozessrekonstruktion auf Basis eines vereinfachten Kreditantragsprozesses (siehe Abb. 2.13 (a)) veranschaulicht. Nach der

[20] ProM (Vgl. Aalst, Dongen et al. 2009) ist eine Umgebung zur Erprobung wissenschaftlicher Ansätze aus dem Process Mining-Bereich. Mithilfe der erweiterbaren Architektur soll eine möglichst einfache Integration ermöglicht werden. ProM ist quelloffen (GPL) und enthält mittlerweile über 200 verschiedene Plugins.

[21] Disco (Vgl. Günther und Rozinat 2012) ist ein kommerzielles Tool zur Analyse von Prozesslogs der Firma Fluxicon, einem Spin-Off der Technischen Universität Eindhoven (TUe), NL.

[22] SWAT (Vgl. Accorsi, Holderer et al. 2014) ist ein Werkzeug zur sicherheitsorientierten Analyse von Geschäftsprozessen. Neben Methoden zur modellbasierten Analyse von Prozessen (de-jure Modelle) auf Basis von Informationsflüssen, ermöglicht SWAT auch die Verifikation von Sicherheitseigenschaften auf Prozesslogs.

Ablauf	Aktivitätssequenz
Ablauf 1	[Request Loan] → [Check Status] → [Prepare Form] → [Sign] → [Ack]
Ablauf 2	[Request Loan] → [Prepare Form] → [Check Status] → [Sign] → [Ack]
Ablauf 3	[Request Loan] → [Check Status] → [Prepare Form] → [Refuse]
Ablauf 4	[Request Loan] → [Prepare Form] → [Check Status] → [Refuse]
Ablauf 5	[Request Loan] → [Check Status] → [Prepare Form] → [Sign]
Ablauf 6	[Request Loan] → [Prepare Form] → [Check Status] → [Sign]
Ablauf 7	[Request Loan] → [Prepare Form] → [Sign]
Ablauf 8	[Request Loan] → [Prepare Form] → [Refuse]

Tabelle 2.2.: Prozesslog zur Veranschaulichung der Modellrekonstruktion.

Kreditanfrage ([Request Loan]) werden der Finanzstatus des Kunden geprüft ([Check Status]) und die Vertragsunterlagen vorbereitet ([Prepare Form]). Diese Aktivitäten können parallel ausgeführt werden, bevor dann auf Basis des Prüfungsergebnisses die Anfrage entweder akzeptiert ([Sign]) oder abgelehnt ([Refuse]) wird. Im Fall besonders hoher Kreditsummen muss die Anfrage von einer zusätzlichen Person geprüft und befürwortet werden ([Ack]). Die Optionalität dieser Zusatzprüfung wird mithilfe einer *stillen Transition* modelliert, die keinen Bezug zu einer tatsächlich ausgeführten Aktivität hat, sondern lediglich zu Routingzwecken dient. Tab. 2.2 zeigt einen beispielhaften Prozesslog, der neben dem geplanten Verhalten auch abweichende Pfade enthält, bei denen die Aktivität [Check Status] ausgelassen wird (Ablauf 7 und Ablauf 8).

Ein Großteil der Rekonstruktionsansätze bezieht sich auf den Kontrollfluss von Prozessen, versucht also kausale Zusammenhänge zwischen Prozessaktivitäten zu erkennen. Es existieren jedoch auch Verfahren zur Rekonstruktion von Modellen, die der Visualisierung des organisationellen und informationellen Aspekts der Prozessausführung dienen. Im Folgenden wird getrennt auf Methoden beider Bereiche und ihre Eignung für die sicherheitsorientierte Betrachtung von Geschäftsprozessen eingegangen.

Rekonstruktion der Prozessstruktur

Modelle, die den Zusammenhang von Aktivitäten darstellen, können grundsätzlich in strukturbasierte Verfahren und *Fuzzy Mining* unterteilt werden. Strukturbasierte Verfahren versuchen, die einem Prozesslog zugrundeliegende Struktur zu rekonstruieren. Dabei werden Metamodelle verwendet, die die Definition struktureller Elemente wie Entscheidungsknoten (inklusive oder exklusive ODER-Verknüpfungen), Parallelität (UND-Verknüpfungen) und Schleifen unterstützen. Auf diese Weise können Abhängigkeiten von Aktivitäten (z.B. dass A und B nie gleichzeitig innerhalb eines Pfades auftreten) explizit wiedergegeben werden. Im Gegensatz zur isolierten Betrachtung einzelner Varianten bietet die Betrachtung solcher Strukturmodelle die Möglichkeit, tatsächliches Verhalten grafisch zu erfassen und Stellen im Prozess zu identifizieren, an denen von geplantem Verhalten abgewichen wird. Im Fall rekonstruierter Petrinetze kann durch sog. *token games* intuitiv durch das reale Prozessverhalten *navigiert* werden. Beginnend in einem

Initialzustand, kann auf diese Weise Transition für Transition geschaltet und das Zustandekommen von Instanzen nachvollzogen werden. Darüber hinaus können Vergleiche von de-jure und de-facto Modellen bspw. zusätzliche oder fehlende Aktivitäten identifizieren und damit ebenfalls zum Verständnis von Abweichungen beitragen. Für den Vergleich von Prozessmodellen existieren vielfältige Verfahren[23]. Die Identifikation von Auffälligkeiten und Umständen, die in detaillierteren Betrachtungen erörtert werden müssen, stellt dabei die Grundlage weiterer Analyseschritte dar.

Abhängig von ihrem Verwendungszweck kann die Aussagekraft rekonstruierter Modelle unterschiedlich bewertet werden. Traditionell besteht das Ziel der Prozessrekonstruktion darin, vom Prozesslog zu abstrahieren und so eine möglichst kompakte Repräsentation des gängigsten Prozessverhaltens abzuleiten. Seltene Pfade werden dabei typischerweise als nicht repräsentativ bzw. als Störquelle (*noise*) betrachtet und vernachlässigt. Diese Vorgehensweise ist für sicherheitsorientierte Betrachtungen nur bedingt hilfreich, da der Eindruck des tatsächlichen Prozessverhaltens durch die von der Abstraktion bedingte Unschärfe verfälscht werden kann. Meist existieren keine Garantien dafür, wie viel Verhalten gemäß Prozesslog Eingang in das Modell finden und wie viel zusätzliches Verhalten dieses enthält. Das Ergebnis der Anwendung eines heuristischen Ansatzes zur Rekonstruktion der Prozessstruktur (Details in Abschnitt 3.4) auf den Log aus Tab. 2.2 findet sich in Abb. 2.13(b)[24]. Obwohl das Modell einen Großteil des Verhaltens gemäß Prozesslog enthält, wird die Optionalität zusätzlicher Bestätigungen (Aktivität [Ack]) vernachlässigt, was den Generalisierungseffekt veranschaulicht.

Verfahren, die weniger Generalisierung vornehmen und näher am Prozessverhalten gemäß Log bleiben, produzieren typischerweise komplexere Modelle (siehe Abb. 2.13(d)). Die Interpretation solcher Modelle ist dennoch nicht trivial, denn es kann im Einzelfall nicht unterschieden werden, ob für eine Sequenz im Modell eine Entsprechung im Prozesslog existiert. Eine detaillierte Betrachtung existierender strukturbasierter Rekonstruktionsverfahren und ihre Bewertung hinsichtlich Qualität und Tauglichkeit im Rahmen von sicherheitsorientierten Prozessanalysen kann Abschnitt 3.6 entnommen werden.

Im Gegensatz zur Erkennung der Prozessstruktur, beziehen sich Methoden des *Fuzzy Mining*[25] auf direkte Folgebeziehungen von Aktivitäten und deren Häufigkeit im Prozesslog. Innerhalb der Visualisierung existiert eine Verbindung von einer Aktivität A zu einer anderen Aktivität B, wenn mindestens ein Ausführungspfad im Log existiert, der einen Übergang von A nach B enthält. Die Stärke der Verbindung leitet sich aus ihrer Häufigkeit im Log ab. Ein Vorteil dieser Methode ist, dass mit der Verwendung von Schwellwerten für die Mindest-Häufigkeit von Verbindungen auf einfache Weise Modelle unterschiedlichen Detailgrades erstellt werden können. Obwohl diese keine strukturellen Elemente enthalten, werden sie bevorzugt in Software-Tools verwendet, die Process Mining Ansätze für Prozessanalysen bereitstellen[26]. Die Häufigkeit von Folgebeziehungen ist in vielen Fällen zur Ableitung von Folgeschritten innerhalb der Analyse zwar ausreichend; zur Erlangung eines detaillierten Prozessverständnisses sind Fuzzymodelle allerdings nur

[23] Vgl. Humm et al. 2012; Alves de Medeiros, Aalst et al. 2008; Weidlich et al. 2012; Rittgen 2011; Becker et al. 2012.
[24] [Prom: Mining - Heuristics Miner (Default) + Konvertierung in Petrinetz]
[25] Vgl. Günther und Aalst 2007.
[26] Perceptive Process Analytics, QPR Process Analyzer, Disco (Vgl. Günther und Rozinat 2012)

bedingt geeignet, da der Kontrollfluss des zugrundeliegenden Strukturmodells Abhängigkeiten zwischen Prozessaktivitäten bedingen kann, die auf diese Weise nicht erfasst werden können[27]. Auch wenn die Parallelität der Aktivitäten [Check Status] und [Prepare Form] im Fuzzymodell aus Abb. 2.13(c) offensichtlich erscheint, ist sie nicht explizit. Innerhalb komplexerer Modelle kann diese Information nicht mehr oder nur sehr schwer gewonnen werden. Generell sind Parallelität oder wechselseitiger Ausschluss Beispiele für Abhängigkeiten, die in Fuzzymodellen nicht ohne weiteres erkennbar sind.

Es kann festgehalten werden, dass Fuzzymodelle zwar ein gewisses Verständnis für den Prozessablauf vermitteln; für detailliertere Analysen jedoch auf strukturbasierte Verfahren zurück gegriffen werden sollte. Hinsichtlich verfügbarer Ansätze in diesem Bereich existiert ein Trade-Off zwischen intuitiver Erfassbarkeit und Präzision rekonstruierter Strukturmodelle.

Betrachtung weiterer Prozessaspekte

Prozesslogs können neben den durchgeführten Aktivitäten auch Informationen zu beteiligten Akteuren, verwendeten Datenelementen oder temporale Eigenschaften wie die Dauer von Aktivitäten enthalten. Die Rekonstruktion der Prozessstruktur wird ergänzt durch Methoden zur Erstellung von Modellen für die Visualisierung von Aspekten der organisationellen, temporalen und datenbezogenen Perspektive, die bei der Sicherheitsanalyse von Prozessen wertvolle Hinweise für die Ursachenforschung bei abweichendem Prozessverhalten liefern können.

Organisationelle Perspektive. Rekonstruktion innerhalb der organisationellen Perspektive bezieht sich auf die Darstellung von Beziehungen zwischen Akteuren, die in der Prozessausführung beteiligt sind. Eine hervorgehobene Rolle nehmen hierbei Methoden zur Erstellung Sozialer Netzwerke ein[28]. Die Analyse Sozialer Netze ermöglicht Untersuchungen zur Rolle von Akteuren und deren „Wichtigkeit“ für die Ausführung eines Prozesses. Dabei wird typischerweise die Anzahl der Verbindungen zu anderen Akteuren betrachtet. Bzgl. der Art der Beziehung zwischen Akteuren existieren unterschiedliche Metriken:

1. Bei der *handover of work* Metrik werden Kausalitätsbeziehungen betrachtet, die immer entstehen, wenn nach der Ausführung einer Aktivität von Akteur O_1 entweder direkt oder indirekt eine Aktivität von einem anderen Akteur O_2 ausgeführt wird.

2. *subcontracting* bezieht sich auf die Vergabe von Teil-Arbeitsschritten eines Akteurs O_1 an einen anderen Akteur O_2. Diese Beziehung tritt immer auf, wenn eine Aktivität von O_2 zwischen zwei Aktivitäten von O_1 stattfindet.

3. Eine Beziehung kann sich auch daraus ergeben, wie oft verschiedene Akteure in denselben Prozessausführungen beteiligt sind. Die *working together* Metrik bezieht sich deshalb auf die Anzahl solcher Instanzen (siehe Abb. 2.5).

[27] Vgl. Jans, Depaire et al. 2011.
[28] Vgl. Aalst, Reijers et al. 2005.

4. Um herauszufinden, welche Akteure ähnliche Tätigkeiten (i.S. von Prozessaktivitäten) ausüben, bietet sich die *similar work* Metrik an. Für die Definition der typischen Tätigkeit werden Profile erstellt, deren Ähnlichkeit für die Ableitung von Verbindungen zwischen Akteuren und deren Intensität verwendet wird.

5. Prozesslogs können Ereignisse enthalten, die nicht nur für die Ausführung oder Fertigstellung von Aktivitäten stehen, sondern genauere Informationen über den Bearbeitungsmodus liefern. Dabei kann es sich um die Zuteilung einer Aktivität an einen bestimmten Akteur handeln, deren tatsächlicher Bearbeitungsstart in einem weiteren Ereignis erfasst wird. Weitere Eventtypen können zur Definition von Delegationen, der Pausierung, Weiterführung und Beendigung von Aktivitäten verwendet werden[29].

   Metriken die sich auf Eventtypen beziehen, können z.B. explizit Delegationen betrachten und daraus Informationen über die Hierarchie von Prozessbeteiligten ableiten.

Organisationelle Modelle eignen sich dazu, kollaborierende Akteure zu identifizieren[30]. Wirtschaftskriminelle Delikte innerhalb betrieblicher Abläufe sind oft nur durch die Zusammenarbeit mehrerer Individuen möglich, die sich gegenseitig decken und durch geschicktes Zusammenspiel betrügerische Aktivitäten vertuschen. Die Betrachtung von Sozialen Netzen unter Verwendung unterschiedlicher Metriken für die Art der Kollaboration von Akteuren kann aus Sicherheitssicht bei der Aufdeckung solcher Aktivitäten hilfreich sein. Insbesondere, wenn nicht klar ist, ob ein bereits bekannter Täter allein gehandelt hat oder auf Unterstützer zurück greifen konnte, können Analysen hinsichtlich seiner Verbindungen zu anderen Personen Klarheit schaffen oder zumindest unterstützend wirken.

Temporale Perspektive. Die Analyse des zeitlichen Verhaltens von Prozessen ist eine Kernfunktion existierender Tools zur Prozessanalyse. Aus Statistiken zur Dauer einzelner Aktivitäten oder ganzer Instanzen können wertvolle Informationen zur Abschätzung der Performanz und Ableitung von Anpassungen zur Optimierung des Prozessverlaufs gewonnen werden. Mittlerweile bieten manche Process Mining-Umgebungen (darunter Disco und ProM) erweiterte Visualisierungskomponenten, die den Prozessverlauf anhand von Marken visualisieren, die durch ein Fuzzymodell bewegt werden und so eine intuitive Erkennung von „Flaschenhälsen“ ermöglichen, bei denen die Ausführung verlangsamt wird (Marken bewegen sich aufgrund längerer Übergangszeiten zwischen Aktivitäten langsamer) oder zeitweise still steht (Häufung von Marken an bestimmten Aktivitäten). Abb. 2.6 zeigt die Bewegtbild-Animation eines Prozessverlaufs, in dem deutlich erkennbar ist, dass die ersten Prozessaktivitäten schnell aufeinander folgen, aber bis zur Bezahlung verhältnismäßig viel Zeit vergeht.

[29] Das MXML-Logformat (Vgl. Dongen und Aalst 2005a) enthält einen *life-cycle* für Prozessaktivitäten, der unterschiedliche Eventtypen enthält. Ob die Erstellung eines detaillierten Prozesslogs inkl. Eventtypen möglich ist, hängt davon ab, ob diese Information aus entsprechenden Informationssystemen extrahiert werden kann. In vielen Fällen beschränkt sich die Extraktion auf Zeitstempel zur Fertigstellung von Aktivitäten, maximal auf die Differenzierung von Start und Ende der Ausführung.

[30] Vgl. Aalst und Song 2004.

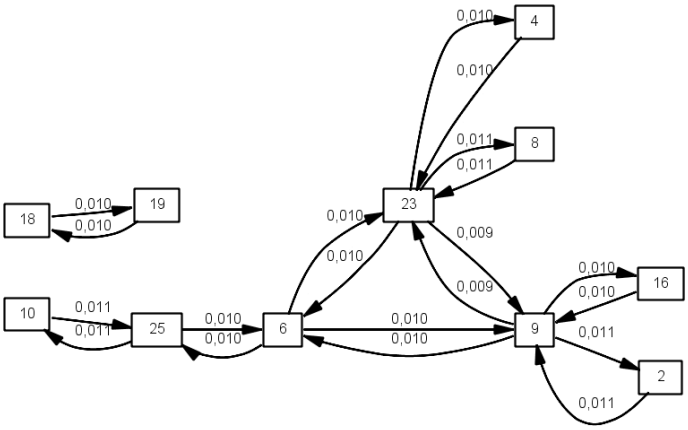


Abbildung 2.5.: Rekonstruiertes Soziales Netzwerk bzgl. der Zusammenarbeit von Akteuren.

Aus Sicherheitssicht ist die Identifikation verhältnismäßig kurzer oder langer Ausführungspfade besonders interessant, da Prozessinstanzen, die betrügerische Aktivitäten enthalten, in vielen Fällen sehr kurz ausfallen oder wie im Fall der Société Générale (siehe Abschnitt 1.3) scheinbar endlos fortgeführt werden. Modelle zur Visualisierung zeitlicher Prozesseigenschaften können Auditoren dabei helfen, den Betrachtungsraum gemäß der anvisierten Fragestellung zu reduzieren und zielgerichtet zu arbeiten. Dabei wird zwar nicht zwangsläufig eine Verbindung zu einer der in Abschnitt 1.3 aufgestellten Sicherheitseigenschaften hergestellt, doch können auf diese Weise Konstellationen entdeckt werden, die sicherheitskritische Schwachstellen begünstigen (z.B. mangelhafte Verfügbarkeit benötigter Ressourcen aufgrund von Denial-of-Service Attacken).

Datenbezogene Perspektive Die Betrachtung des Prozessablaufs aus Datenperspektive ermöglicht die Erfassung der Verwendung und des Flusses von Daten. Statistische Informationen darüber, welche Instanzen verhältnismäßig hohe Werte von Datenelementen (z.B. Kreditsummen) beinhalten und welche Akteure dabei beteiligt sind, können Hinweise auf die Regelkonformität geben. Insbesondere bei Prozessen, deren Transaktionen hohe Kapitaleinsätze beinhalten, existieren typischerweise strikte Richtlinien für die Implementierung von Kontrollen und die Involvierung bestimmter Personengruppen (Funktionstrennung).

Mithilfe von *Decision Mining*[31] können auf Basis eines rekonstruierten Strukturmodells Regeln für Datenelemente an Entscheidungspunkten im Prozessmodell (siehe Abb. 2.7) abgeleitet werden. An Entscheidungsknoten existieren mehrere Alternativen für die weitere Prozessausführung. Wird die Wahl von Folgepfaden durch Charakteristika auf datenbezogener-, organisationeller oder temporaler Ebene determiniert, können mithilfe von Decision Mining zugrundeliegende Regeln approximiert werden. Dafür werden Techniken des Maschinellen Lernens[32] angewandt, konkret Verfahren zur automatischen

[31] Vgl. Rozinat et al. 2006; Leoni und Aalst 2013b.
[32] Vgl. Mitchell 1997.

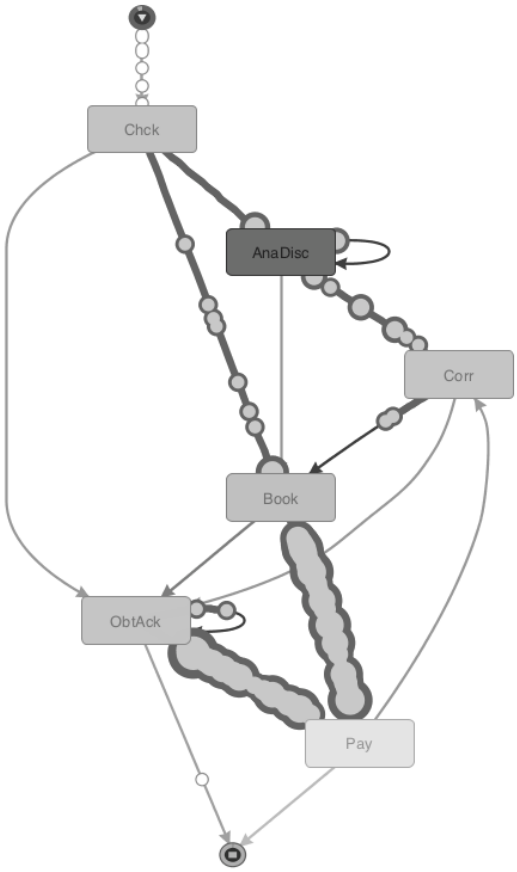


Abbildung 2.6.: Visualisierung temporalen Prozessverhaltens anhand beweglicher Marken im Fuzzymodell.

Klassifikation. Für jeden identifizierten Entscheidungspunkt wird ein Entscheidungsbaum konstruiert, der alle Datenelemente involviert, die bis zu diesem Zeitpunkt verwendet und verändert wurden[33]. I.S. des Klassifikationsproblems entsprechen die Blätter des Baums den Klassen, i.e. den möglichen Entscheidungen, die in diesem Fall den Folgepfaden entsprechen. Regeln für die Wahl von Pfaden werden durch die Kombination von Pfaden im Entscheidungsbaum gewonnen (siehe Beispiel in Abb. 2.7). Idealerweise stehen diese Regeln in einem sinnvollen Zusammenhang mit existierenden Richtlinien.

Die Anwendung von Decision Mining auf komplette Prozesslogs ist selten sinnvoll, da die Komplexität abgeleiteter Regeln mit der Anzahl von Datenelementen und der Größe zugehöriger Wertebereiche stark zunimmt. Dadurch werden sie in vielen Fällen schwer interpretierbar, weshalb nicht immer ein Beitrag zur Erfassung des Prozessverhaltens geliefert wird. Anwendungen auf Teilmengen von Ausführungspfaden können jedoch sehr hilfreich sein.

[33] Details zur Konstruktion von Entscheidungsbäumen und zur Lösung von Klassifikationsproblemen können [Mitchell (1997)] und [Witten et al. (1999)] entnommen werden.

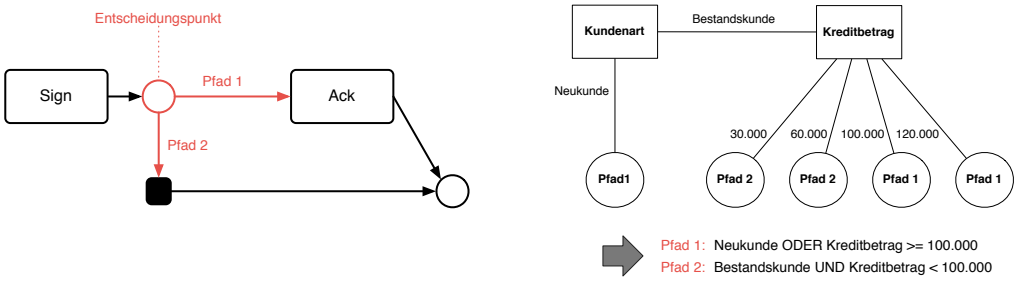


Abbildung 2.7.: Entscheidungspunkt in Strukturmodell und Entscheidungsbaum zur Bestimmung von Regeln für die Wahl von Folgepfaden.

Eine weitere Methode zur Visualisierung von Datenflüssen besteht in der Konstruktion sog. *Propagation Graphs* (PGs), die ursprünglich zur Analyse von Datenflüssen in Programmcode entwickelt wurden[34]. Die zentrale Annahme des Ansatzes von Accorsi et al.[35] ist, dass im Prozesslog für jede Aktivität Informationen zu Eingabe-, sowie Ausgabe-Daten vorhanden sind. Knoten innerhalb des PGs entsprechen Prozessaktivitäten. Verbindungen zwischen Knoten existieren, wenn eine Aktivität Daten als Eingabe erwartet, die von einer anderen Aktivität ausgegeben werden. Auf diese Weise können direkte Informationsflüsse abgebildet werden[36]. Unter Verwendung einer dedizierten Sprache für die Definition von informationsflussorientierten Richtlinien können kritische bzw. unerlaubte Flüsse im Graph identifiziert werden.

Zusammenfassend besteht das Potential der Prozessrekonstruktion hauptsächlich in der Bereitstellung von Modellen, die ein detailliertes Verständnis der tatsächlichen Prozessausführung und Vergleiche mit dem geplanten Verhalten ermöglichen.

2.2.2 Conformance Checking: Überprüfung der Prozesskonformität

Methoden der Prozesskonformität messen die Übereinstimmung eines Prozesslogs mit einem de-jure Modell oder verifizieren formalisierte Eigenschaften bzgl. des Ablaufs von Prozessinstanzen. Insofern eignen sie sich prinzipiell zur Identifikation von Abweichungen realen Prozessverhaltens zu geplantem Prozessverhalten und der Erkennung von Verletzungen einzuhaltender Sicherheitseigenschaften.

Abweichungen von der Prozessspezifikation repräsentieren nicht vorgesehene Ausführungspfade und sind deshalb im Rahmen von Prozessaudits von besonderem Interesse. Eine gängige Methode, abweichende Ausführungspfade zu bestimmen, besteht darin, Pfade aus dem Prozesslog im de-jure Modell „abzuspielen“. Mithilfe dieses *Replays* können nicht spezifikationsgemäße Pfade identifiziert werden. Ein besonders flexibler Ansatz[37], der neben verschiedenen Metriken für die Übereinstimmung von Prozessmodell (in Form eines Petrinetzes) und Prozesslog auch eine Partitionierung des Logs in konforme und

[34] Vgl. Livshits et al. 2009.
[35] Vgl. Accorsi, Wonnemann und Stocker 2011.
[36] Die Betrachtung indirekter Informationsübertragung über verdeckte Kanäle ist in diesem Zusammenhang zwar denkbar, wird aber nicht explizit betrachtet
[37] Vgl. Rozinat et al. 2008.

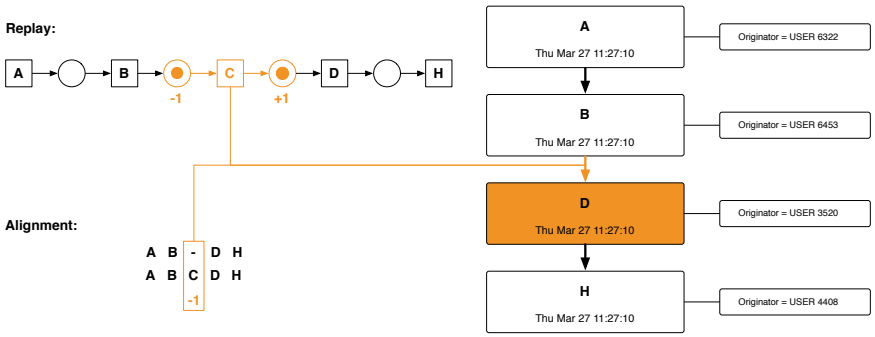


Abbildung 2.8.: Visualisierung einer identifizierten Kontrollflussabweichung durch Replay und Alignment.

nicht konforme Pfade erlaubt, wurde in das *Conformance Checker* ProM-Plugin integriert.

Abb. 2.8 zeigt beispielhaft die Visualisierung einer Abweichung in ProM, bei dem die obligatorische Aktivität C übersprungen und stattdessen nach B direkt D ausgeführt wurde. Innerhalb des Petrinetzes wird für jeden abgespielten Pfad die Anzahl fehlender und zusätzlicher Marken gezählt, in diesem Fall eine fehlende Marke in der Stelle vor C und eine zusätzliche Marke in der Stelle vor D. Auf Basis dieser Information wird ein Maß für die Übereinstimmung des Prozesslogs mit dem de-jure Modell berechnet (*token-based fitness*, Vgl. Abschnitt 3.1). Im Prozesslog enthaltene Information wie Zeitstempel der aufgetretenen Aktivitäten oder beteiligte Akteure kann im Fall kritischer Abweichungen unterstützend für die Ursachenklärung verwendet werden.

Neben Replayansätzen können Abweichungen auch durch *Alignment* bestimmt werden. Dabei wird für jeden Pfad im Prozesslog ein Modellpfad gesucht, der diesem entspricht. Existiert kein solcher Pfad, wird analog zu Verfahren aus der Bioinformatik zur Gruppierung ähnlicher DNA-Sequenzen versucht, einen möglichst ähnlichen Modellpfad zu finden. Dies geschieht durch die paarweise Zuordnung von Aktionen (*moves*) im Modell und Aktionen im Prozesslog. Weichen Aktionen von Log und Modell voneinander ab, wird dies mit *Kosten* sanktioniert. Für jeden Pfad im Prozesslog findet stets das kostenminimale Alignment Eingang in die Berechnung des Ähnlichkeitsmaßes. Das Alignment in Abb. 2.8 zeigt nach zwei übereinstimmenden Aktionen eine Paarung, innerhalb derer keine Aktion im Prozesslog stattfindet, während im Modell C ausgeführt wird. Der Vorteil von Alignments im Vergleich zu Replay besteht darin, dass die Quantifizierung der „Nähe“ eines Prozesslogs zu einem Modell durch die Kosten von Alignments einfacher bestimmt werden kann und auf Benutzerebene leichter interpretierbar ist als die Bedeutung fehlender oder zusätzlicher Marken.

Bisherige Ansätze und Lösungen[38] beziehen sich bei der Übereinstimmung eines Prozesslogs mit einem Prozessmodell meist ausschließlich auf den Kontrollfluss. Situationen, in denen die Wahl von Folgepfaden von Datenbedingungen bestimmt wird, können so nicht

[38] Vgl. Adriansyah et al. 2011; Aalst et al. 2012; Rozinat et al. 2008; Adriansyah et al. 2010; Cook und A. L. Wolf 1999.

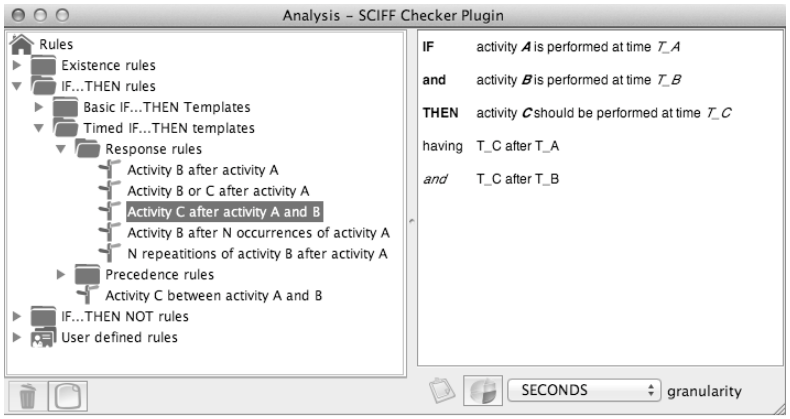


Abbildung 2.9.: Regeleditor des SCIFF-Plugins in ProM und SWAT.

berücksichtigt werden. Neuere Alignmentansätze[39] betrachten deshalb mehrere Dimensionen der Prozessausführung (Kontrollfluss, Daten und Akteure) gleichzeitig und erlauben damit eine facettenreiche Konformitätsprüfung, die realen Gegebenheiten gerecht wird. Auf diese Weise kann bspw. überprüft werden, ob Sonderfälle korrekt behandelt wurden (z.B. eine intensivere Solvenzprüfung bei Neukunden).

Voraussetzung für die Anwendung von Replay- und Alignment-Techniken ist die Verfügbarkeit eines formalen Modells für den Kontrollfluss des zu analysierenden Prozesses. Die Transformation von informalen Modellen (z.B. BPMN) in Petrinetze ist zwar nicht trivial, eröffnet Auditoren jedoch die Möglichkeit konformes Prozessverhalten eindeutig von nicht konformem zu unterscheiden. Ein weiterer Vorteil solcher Modelle besteht in ihrer Anwendbarkeit in *Variantenanalysen*. Im Rahmen einer Variantenanalyse können unterschiedliche Arten der Prozessausführung betrachtet und in Zusammenhang mit der Prozessspezifikation bewertet werden[40]. Dieser Form der Prozessanalyse kommt im Rahmen von Analyseprojekten typischerweise eine besondere Bedeutung zu, da die Zahl der Varianten, kombiniert mit deren Frequenz eine aggregierte Sichtweise schafft, die einen Eindruck von der Gesamtvarianz eines Prozesses liefert und damit eine grobe Einschätzung der Compliance und Sicherheit ermöglicht. Fällt die Zahl unterschiedlicher Varianten hoch aus, ist die manuelle Prüfung jeder einzelnen Variante auf Konformität mühsam und fehleranfällig. Replaying ermöglicht in diesem Zusammenhang eine schnelle Identifikation konformer und damit die Isolation „interessanter“ nicht konformer Varianten.

Bzgl. der Konformität eines Prozesslogs zu formalisierten Anforderungen existieren verschiedene auf Model-Checking-Techniken basierende Verfahren. Der in dem ProM-Plugin *LTL-Checker*[41] implementierte Ansatz[42] baut auf LTL auf, um temporale Aussagen über Zustände in Pfaden zu treffen. Dabei wird eine Sprache realisiert, die LTL um Konstrukte für den Zugriff auf prozessspezifische Datenfelder und deren Verwendung in temporalen

[39] Vgl. Leoni, Aalst und Dongen 2012; Leoni und Aalst 2013a.
[40] *[Disco: Cases] [ProM 5.2: Mining - Log Summary, Exports - Grouped MXML Log (same sequences) - Log Summary]*
[41] *[ProM 5.2: Analysis - LTL Checker]*
[42] Vgl. Aalst, Beer et al. 2005.

Formeln erlaubt[43]. Das Plugin bietet eine Reihe von Mustern, die für die Überprüfung von (Sicherheits-)Eigenschaften genutzt werden können. Diese bestehen aus parametrisierbaren Formeln, die Aussagen über die Charakteristik von Pfaden treffen (z.B. dass nach einer bestimmten Aktivität stets eine weitere folgen muss). Der SCIFF-Ansatz[44] arbeitet auf Basis von Prolog und stellt einen Regeleditor bereit (siehe Abb. 2.9), der eine intuitive Erstellung von Regeln erlaubt, die intern in Prolog-Statements übersetzt und auf dem gegebenen Prozesslog überprüft werden. Während die Ausführungspfade im Log als Fakten der Datenbasis dienen, entsprechen die Regeln Anfragen, die für jeden Pfad stets mit Ja (Regel gilt) oder Nein (Regel gilt nicht) beantwortet werden. Diese beiden Ansätze eignen sich hervorragend um die Einhaltung von Anforderungen der Nutzungskontrolle und Funktionstrennung zu überprüfen. Beispiele für verifizierbare Sicherheitsanforderungen sind:

Funktionstrennungs-Anforderungen:

- *Existieren Pfade, bei denen lediglich eine Person beteiligt ist?*
- *Werden die Aktivitäten A und B von unterschiedlichen Personen durchgeführt?*
- *Wird jede Kreditanfrage von einem Sachbearbeiter des Backoffice geprüft?*

Nutzungskontroll-Anforderungen:

- *Wird bei Abweichungen in Eingangsrechnungen der Einkauf involviert?*
- *Erfolgt bei Kreditsummen >= 500.000 Euro eine Zusatzgenehmigung?*
- *Erfolgt der Vertragsabschluss spätestens 14 Tage nach der Aushändigung der Vertragsunterlagen?*
- *Wird der Abschlussbericht einer Bilanzprüfung stets vom Prüfer selbst erstellt?*

Für die Überprüfung von Richtlinien zur Zugangskontrolle kann in ProM zunächst die *User-Task-Matrix*[45] herangezogen werden, die für jede Aktivitäts-Akteur-Paarung angibt, ob diese im Prozesslog zu finden ist (siehe Abb. 2.10). Für systematische Analysen komplexer Rollenhierarchien ist diese Methode aufgrund des erheblichen manuellen Aufwands allerdings nicht geeignet. Baumgrass et al. stellen jedoch Methoden für die direkte Überprüfung von RBAC-Modellen auf Prozesslogs zur Verfügung[46]. Dafür wird ein Organisations-Modell benötigt, welches die Akteure eines Systems und deren Rollenzugehörigkeit beschreibt. Berechtigungen werden für definierte Rollen auf Aktivitätsebene spezifiziert und intern auf LTL-Formeln zurückgeführt, die dann mithilfe obiger Model-Checking-Ansätze überprüft werden[47]. Dieser Ansatz eignet sich außerdem für die Extraktion von Artefakten für das Role Mining-Problem[48]. Role Mining bezeichnet einen bottom-up-Ansatz für die Erstellung von Rollenhierarchien auf Basis notwendiger Berechtigungen für die Durchführung von Einzelaktivitäten. Das zugrundeliegende Kalkül besteht in der Realisierung des *least privilege* Konzepts, nach dem Benutzer möglichst

[43] Vgl. Beer 2004.
[44] *[ProM 5.2: Analysis - SCIFF Checket Plugin] [SWAT: Analysis - SCIFF]*
[45] *[ProM 5.2: Analysis - Originator by Task Matrix]*
[46] Vgl. Baumgrass, Baier et al. 2011.
[47] *[Prom 6.0: RBAC-to-LTL Plugin]*
[48] Vgl. Baumgrass, Strembeck und Rinderle-Ma 2011.

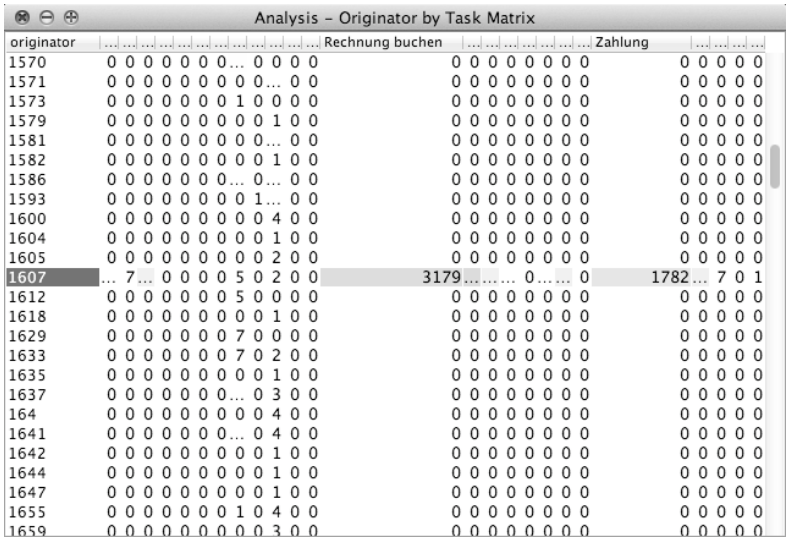


Abbildung 2.10.: User-Task-Matrix in ProM.

wenige Berechtigungen erhalten sollen, die über die für ihre übliche Tätigkeit benötigten Rechte hinausgehen. Methoden zum Vergleich bestehender Rollenkonzepte mit aus Prozesslogs generierten[49] erlauben die Ableitung von Anpassungsoperationen, die neben effektiverem Rechtemanagement durch geeignete Einschränkung von Rechten einen Beitrag zur Erhöhung der Sicherheit leisten.

Die wesentliche Stärke von Conformance Checking-Methoden besteht in der Fähigkeit, Sicherheitseigenschaften formal zu erfassen und auf Prozesslogs zu verifizieren.

2.3 Nutzen und Defizite der Process Mining-Technologie

Existierende Ansätze des Process Mining können effektiv im Rahmen von Prozessaudits eingesetzt werden, um die tatsächliche Prozessausführung nachzuvollziehen und konkrete Sicherheitseigenschaften zu überprüfen. Der Nutzen von Process Mining zur Reduktion von (Sicherheits-)Risiken in Geschäftsprozessen und insbesondere zur Aufdeckung geschäftsschädigender Handlungen von Prozessbeteiligten konnte in zahlreichen Publikationen nachgewiesen werden[50]. Die noch junge Technologie findet zunehmend Beachtung und Anwendung in praktischen Prozessprüfungen[51]. Allerdings liegt der Fokus der Process Mining-Technologie traditionell nicht auf der Sicherheit von Geschäftsprozessen. Daraus ergeben sich Defizite hinsichtlich der Abdeckung relevanter Sicherheitseigenschaften und der Präzision rekonstruierter Modelle.

[49] Vgl. Baumgrass und Strembeck 2012.

[50] Vgl. Aalst, Hee et al. 2010; Jans, Depaire et al. 2011; Jans, Lybaert et al. 2009; Jans und Alles 2010; Lima Bezerra et al. 2009; Aalst 2005.

[51] Process Mining wird im Bereich der internen Revision als innovative Technologie betrachtet. 2011 wurde der „Audit Innovation Award“ der Fachkonferenz für Revisionsmethodik und Strategische Weiterentwicklung (Audit Challenge 2011) für die Anwendung von Process Mining im internen Prüfungswesen vergeben (Vgl. Frankfurt School of Finance & Management 2011).

Entwicklungspotential

Die in vorherigen Abschnitten erläuterten Möglichkeiten zur Anwendung konkreter Ansätze für die sicherheitsorientierte Betrachtung von Prozessen führen in diesem Zusammenhang zu einer Reihe denkbarer Anwendungsbereiche für Process Mining. Im Wesentlichen können folgende Bereiche durch die Verwendung von Process Mining profitieren:

- **Prozessverständnis.** Die Analyse digitaler Spuren in Form von Prozesslogs ermöglicht die Identifikation des Ist-Zustands von Prozessen und die Erkennung von Prozessvarianten und Abweichungen. Die Visualisierung von Abläufen erlaubt dabei ein intuitives Verständnis von Prozessen. Gewonnene Erkenntnisse zum realen Prozessverlauf erlauben eine fokussiertere Vorbereitung der eigentlichen Prüfungstätigkeit.

- **Prozessprüfung.** Process Mining-Methoden senken allgemein den manuellen Aufwand für die Betrachtung von Prozessdaten und die Selektierung relevanter Ausführungspfade. Durch die Betrachtung des gesamten Prozessverhaltens wird eine hohe Aussagekraft durchgeführter Analysen erreicht. Die Anwendung von Methoden der Konformitätsprüfung erlaubt durch die Verifikation formalisierter Anforderungen dabei eine effiziente Analyse. Dadurch kann die Wirksamkeit interner Kontrollen effektiv überprüft und der Compliancestatus von Prozessen festgestellt werden.

- **Continuous Monitoring.** Im Zuge des erhöhten Automatisierungsgrades von Prozessen und der Verfügbarkeit von Echtzeit-Daten bzgl. deren Durchführung gewinnen kurzperiodische Prüfungen immer mehr an Bedeutung. Im Gegensatz zu jährlichen Prüfungen können so aktuelle Aussagen über die Prozesskonformität gewonnen werden, was den Vergleich von Einzelergebnissen und die Durchführung von Trendanalysen ermöglicht.

- **Prozessoptimierung.** Analysen auf Prozessebene mithilfe von Process Mining erlauben die Ableitung von Maßnahmen für Prozessoptimierungen (Performanz, Regelkonformität) auf Basis objektiver Fakten. Die Möglichkeit, verschiedene Aspekte der Prozessausführung zu fokussieren, führt zu einer differenzierten Beurteilung der Effizienz und Effektivität von Prozessen und der Identifikation von „Flaschenhälsen“. Fragestellungen, die dem Leitprinzip „Wie sieht der Prozess aus wenn...“ folgen, sind in dieser Hinsicht besonders effektiv und erlauben den Vergleich von Produktgruppen, Lieferanten, Sachbearbeitergruppen etc.

Abb. 2.11 zeigt die Anwendung von Process Mining im Kontext von Prozessspezifikation und -ausführung. Die direkte Anwendung von Process Mining auf Prozesslogs durch Conformance Checking-Methoden, ermöglicht eine direkte Feststellung der Prozesskonformität, insbesondere im Hinblick auf die Gültigkeit von (Sicherheits-)Anforderungen. Process Discovery-Methoden dienen in erster Linie dazu, ein detailliertes Prozessverständnis zu erlangen, den Gesamt-Analyseprozess durch gewonnene Erkenntnisse zu leiten und Maßnahmen für die Anpassung von Prozessen abzuleiten.

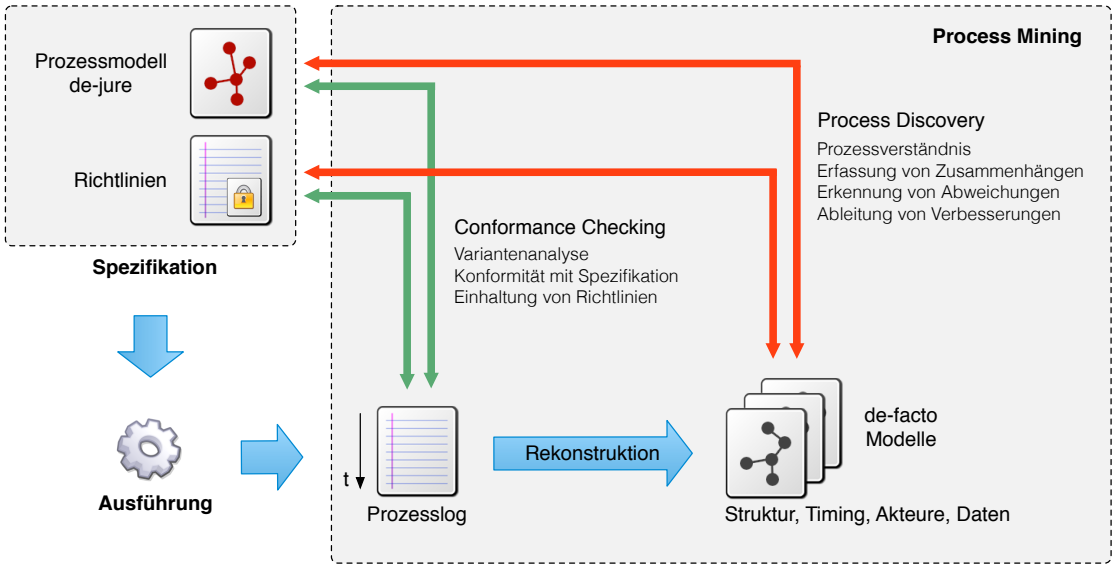


Abbildung 2.11.: Process Mining im Kontext sicherheitsorientierter Prozessanalyse.

Defizite

Process Mining eignet sich zwar prinzipiell für sicherheitsorientierte Prozessbetrachtungen, ist jedoch nicht primär darauf ausgelegt. Dies führt in vielen Fällen zu hohem Aufwand für Anpassungen und erfordert manuelle Arbeitsschritte. Sowohl für die Konformitätsprüfung als auch für die Prozessrekonstruktion müssen Spezifikationen, die im betrieblichen Umfeld typischerweise informaler Natur sind, in mathematisch eindeutige Repräsentationen überführt werden. Dies betrifft Prozessmodelle und Anforderungen gleichermaßen. Analyseergebnisse, die auf *low-level*-Ebene (Petrinetze) gewonnen werden, müssen auf höherer Ebene interpretiert werden (z.B. BPMN-Modell). Generell ergibt sich eine Diskrepanz zwischen der Geschäftsebene, auf der Prozesse und Anforderungen definiert werden und der technischen Ebene, auf der die Analyse stattfindet. Existierende Tools für die Anwendung von Process Mining-Techniken schließen diese Lücke nicht und erfordern typischerweise hohes technisches Knowhow (Petrinetze, LTL etc.)[52]. Darüber hinaus tragen folgende Defizite wesentlich zur Minderung des Mehrwerts der Process Mining-Technologie für sicherheitsorientierte Prozessprüfungen bei:

Intra-Instanz vs. Inter-Instanz. Existierende Methoden zur Verifikation formalisierter Prozessanforderungen beziehen sich ausschließlich auf sog. *Intra-Instanz* Eigenschaften. Zwar können dabei detaillierte temporale Eigenschaften bzgl. der Ausprägung einzelner Ausführungspfade betrachtet werden, nicht jedoch der Zusammenhang von Prozessinstanzen. Die gleichzeitige Betrachtung mehrerer Prozessinstanzen innerhalb überprüfter

[52] Mit dem Tool SWAT (Vgl. Accorsi, Holderer et al. 2014) wird mit der Bereitstellung von Methoden zur intuitiven Definition von Sicherheitseigenschaften und deren Verifikation ein Vorstoß in diese Richtung gewagt. Dieser Aspekt des eigenen Beitrags zum Thema „Sicherheit in Geschäftsprozessen“ wird in dieser Arbeit nicht explizit betrachtet, kann jedoch als Transferleistung aufgefasst werden.

Eigenschaften ist für die Beurteilung der Sicherheit von Prozessen jedoch äusserst sinnvoll. Insbesondere Isolationseigenschaften wie Chinese Wall, bei der die Durchführbarkeit von Aktivitäten im Prozessverlauf nicht nur von zuvor ausgeführten Aktivitäten innerhalb derselben Instanz, sondern auch innerhalb bereits abgeschlossener, vergangener Instanzen abhängt, erfordern diese Betrachtung explizit. Warner und Atluri[53] haben bereits 2006 eine Sprache zur Definition von *Inter-Instanz* Anforderungen vorgeschlagen, mit der die Formulierung von Aussagen wie „Eine Person darf nie in mehr als 5 Prozessinstanzen gleichzeitig beteiligt sein“ möglich sind. Derzeit können solche Anforderungen jedoch nicht überprüft werden.

Indirekte Informationsflüsse. Während Anforderungen hinsichtlich direkter Informationsflüsse (Zugriffskontrolle und PGs) prinzipiell überprüfbar sind, existieren derzeit keine Methoden zur Identifikation verdeckter Kanäle. Implizite Flüsse, die sich in der Struktur von Prozessen manifestieren, spielen eine Rolle bei der *a-priori* Analyse von Strukturmodellen zum geplanten Prozessverhalten. Diesbezüglich konnte die Gültigkeit gängiger Nicht-Interferenz Eigenschaften (BNDC) auf die Abwesenheit bestimmter Petrinetzkonstrukte zurückgeführt werden[54]. Im Wesentlichen handelt es sich dabei um Situationen, in denen durch die Nichtausführbarkeit einer Aktivität auf das Stattfinden einer anderen Aktivität geschlossen werden kann. Die Anwendung der Informationsflusskontrolle auf rekonstruierte Strukturmodelle ist grundsätzlich denkbar, erlaubt jedoch lediglich die Identifikation von Stellen im Prozess, an denen Information geflossen *sein könnte*. Die Quantifizierung geflossener Information ist nicht trivial. Hinzu kommt, dass rekonstruierte Strukturmodelle nicht eindeutig sind und je nach Modell möglicherweise unterschiedliche Schwachstellen gefunden werden. Die Aussagekraft solcher Analysen ist damit eher gering.

Die Überprüfung von Timing Channels durch die Analyse von Instanzlaufzeiten in Kombination mit Werten verwendeter Datenelemente hingegen erscheint denkbar. Unterschiedliche Pfade sind mithilfe von Process Mining erkennbar und die Analyse von Durchlaufzeiten ermöglicht die Feststellung deren typischer Laufzeiten. Signifikante Unterschiede in der Durchlaufzeit von Pfaden, die i.S. von Timing Channels als sicherheitskritisch eingestuft werden, können auf diese Weise gefunden werden. In Web-Anwendungen stellt diese Art der verdeckten Informationsübertragung zwar eine reale Gefahr für die Sicherheit dar[55], die Erkennung verdeckter Kanäle ist jedoch selten Teil von Prozessaudits; entsprechende Methoden existieren bislang nicht.

Unscharfe Strukturmodelle. Bzgl. der Erlangung eines tiefgreifenden Verständnisses des Prozessverhaltens und des Zustandekommens unerwünschter Abweichungen von Prozessspezifikationen, spielt die Genauigkeit der Prozessrekonstruktion eine große Rolle. Während die Erstellung von Modellen in anderen Perspektiven unproblematisch erscheint, existieren bisher keine Methoden zur Rekonstruktion präziser Strukturmodelle. Unschärfen, die sowohl durch die Einführung von Pfaden, die nicht Teil des Prozesslogs sind, als auch durch die Vernachlässigung von Pfaden im Prozesslog zustande kommt,

[53] Vgl. Warner et al. 2006.
[54] Vgl. Accorsi und Lehmann 2012; Accorsi und Wonnemann 2011; Busi et al. 2009.
[55] Vgl. Jager et al. 2012.

sind dabei meist inhärenter Natur, da ein Großteil der Rekonstruktionsansätze bewusst vom Prozessverhalten gemäß Log abstrahiert. Generell ist bei rekonstruierten Strukturmodellen nicht klar, für welche Modellpfade eine Entsprechung im Prozesslog existiert. Aus Sicherheitssicht stellt diese Unschärfe ein Problem dar, da Feststellungen bzgl. des tatsächlichen Prozessverhaltens und damit auch Ableitungen von Maßnahmen zu Verbesserungen/Korrekturen beeinträchtigt werden können.

Auch wenn aus o.g. Gründen bisher nur wenige Verfahren zur Überprüfung von Sicherheitseigenschaften auf rekonstruierten Strukturmodellen zum Einsatz kommen, steht bereits fest, dass mit existierenden Methoden keine Modelle mit ausreichender Qualität bereitgestellt werden können. Weder die Existenz von *false-positives* (erkannte Schwachstellen, die durch Modellpfade hervorgerufen werden, die nicht Teil des Prozesslogs und damit irrelevant sind), noch von *false-negatives* (nicht erkannte Schwachstellen aufgrund fehlenden Prozessverhaltens im Modell) kann ausgeschlossen werden.

Prozessdynamik. Die Dynamik von Geschäftsprozessen, also deren Veränderung im zeitlichen Verlauf, kann bisher nur bedingt nachvollzogen werden. Zwar existiert eine Reihe von Verfahren zur Gruppierung von Ausführungspfaden im Prozesslog, dies geschieht jedoch meist ohne die Berücksichtigung der zeitlichen Dimension. Die Erfassung der Prozessdynamik ist allerdings aus verschiedenen Gründen sinnvoll. Zum einen ist die Prozessdynamik ein Bestandteil des Prozessverhaltens, das für ein Verständnis der Prozessausführung über längere Zeiträume notwendig ist. Zum anderen werden Unschärfen bei der Rekonstruktion von Strukturmodellen durch die Vernachlässigung dieses Aspekts begünstigt. Existieren für einen gegebenen Prozesslog bspw. unterschiedliche Ursprungsmodelle führt die Subsumierung von Pfaden beider Modelle in ein einziges Strukturmodell für das tatsächliche Verhalten ggf. zu Abstraktionen ohne Bezug zum Prozesslog, die wiederum zu Fehlinterpretationen führen können. Wird bei zwei Ursprungsmodellen in einem Modell bspw. eine Aktivität immer ausgeführt, im anderen jedoch nie, führt das im rekonstruierten Modell zu einer Optionalität der Aktivität, was weder dem einen, noch dem anderen Modell gerecht wird.

Beschränkte Sichtweise rekonstruierter Modelle. Process Mining ermöglicht durch vielfältige Ansätze eine facettenreiche Analyse von Prozessdaten. Allerdings können unterschiedliche Perspektiven (Kontrollfluss, Akteure, Datenfluss) bisher meist nur isoliert betrachtet werden. Rekonstruierte Strukturmodelle können mithilfe von Decision Mining mit Bedingungen für Entscheidungspunkte angereichert werden. Der Ansatz von Weiters und Ribeiro[56] bietet Optionen zur Rekonstruktion multidimensionaler Modelle, Eine kombinierte Betrachtung von Kontroll- und Datenfluss i.S. der Verwendung von Datenelementen im Prozessverlauf wird aber bisher nicht vorgenommen. Der Vorteil einer integrierten Betrachtung gegenüber der Kombination von Ergebnissen auf Basis isolierter Betrachtungen liegt in der ganzheitlicheren Erfassung des Prozessverhaltens. Dadurch kann das Zustandekommen bestimmter Ausführungspfade besser nachvollzogen und Wechselwirkungen zwischen Kontroll- und Datenfluss besser erfasst werden.

[56] Vgl. Weijters und Ribeiro 2011.

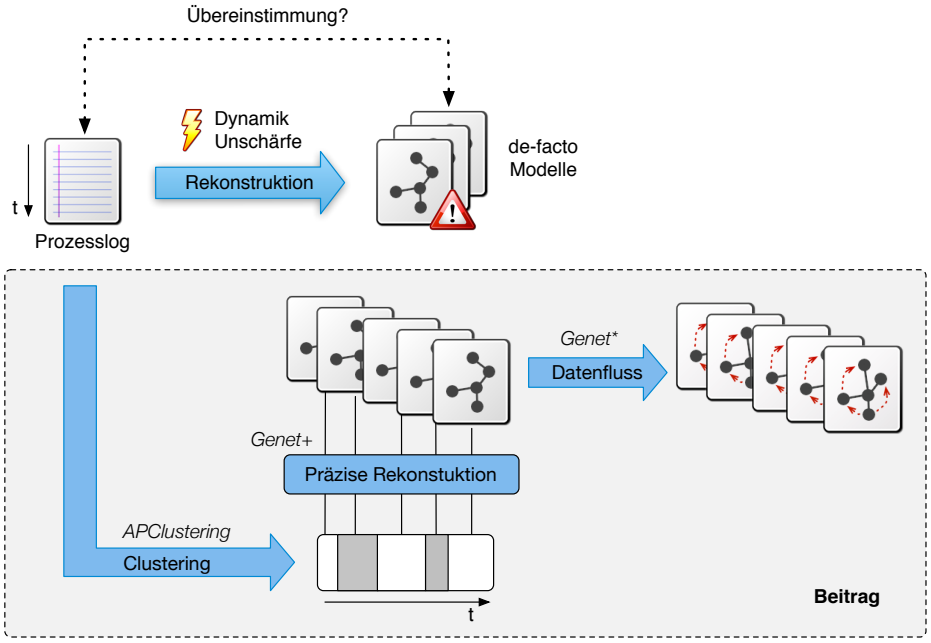


Abbildung 2.12.: Beitrag der Arbeit.

2.4 Erhöhtes Prozessverständnis durch präzise Rekonstruktion

Hinsichtlich der Anwendbarkeit der Prozessrekonstruktion im Rahmen prozessorientierter Sicherheitsanalysen für die Beurteilung der Regelkonformität und Sicherheit von Geschäftsprozessen sind die folgenden Defizite wesentlich:

(AA1) Unpräzise rekonstruierte Strukturmodelle.

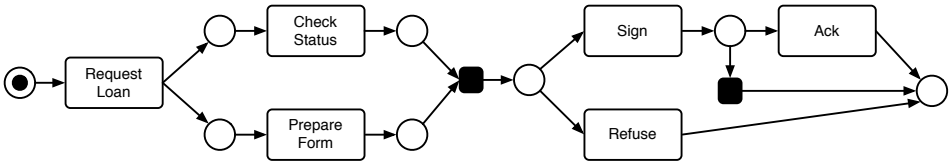
(AA2) Ungenügende Erfassung der Prozessdynamik.

(AA3) Eindimensionalität rekonstruierter Modelle.

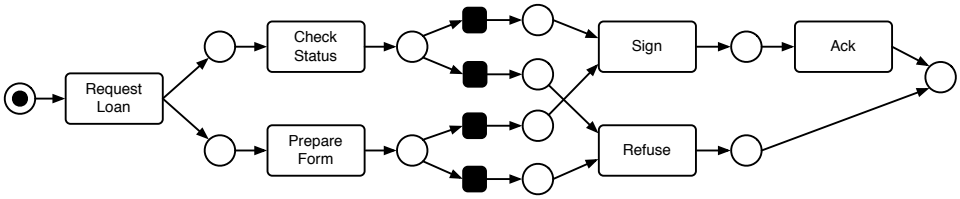
Durch die Vernachlässigung der Prozessdynamik und die Unschärfe existierender Mechanismen kann keine vollständige Übereinstimmung von Prozesslog und Prozessmodell erreicht werden (siehe Abb. 2.12). Die Verwendung rekonstruierter Prozessmodelle zur Erlangung eines detaillierten Prozessverständnisses unter Ausnutzung bisheriger Ansätze ist aus diesem Grund nicht sinnvoll, da durch fehlende Akkuratesse Fehlinterpretationen auf Analyseebene begünstigt werden.

Diese Arbeit befasst sich intensiv mit der Rekonstruktion präziser Strukturmodelle, sowie deren Anreicherung mit Datenflussinformation (Teil I) und der Erfassung der Prozessdynamik (Teil II). Das übergeordnete Ziel besteht darin, Prozessverhalten auf präzise Weise zu visualisieren, um auf diese Weise Auditoren ein intuitives Erfassen tatsächlichen Prozessverhaltens und Verständnis des Zustandekommens von Abweichungen zu ermöglichen. Damit wird eine Grundlage für die Anwendung von Methoden der Prozessrekonstruktion innerhalb nachgelagerter Sicherheitsanalysen von Geschäftsprozessen und die Entwicklung sicherheitsspezifischer Analysetools gelegt.

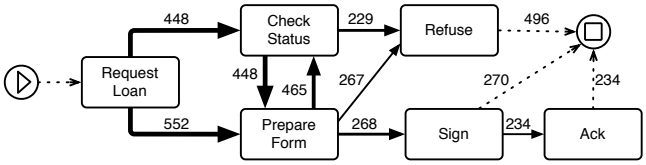
In Abschnitt 3.6 wird die Eignung unterschiedlicher Rekonstruktionsmechanismen für die Erstellung präziser Strukturmodelle aus technischer Sicht beurteilt (i.e. die Unschärfe mithilfe geeigneter Metriken konkretisiert). Mit GENET^{+} wird dann ein Verfahren vorgeschlagen, welches die Rekonstruktion von Strukturmodellen erlaubt, die das tatsächliche Prozessverhalten gemäß Prozesslog präzise abbilden. Die Flexibilität dieses Ansatzes wird genutzt, um mit GENET* neben dem Kontrollfluss auch den Datenfluss zu erfassen. Abschnitt 6 konzentriert sich auf den Aspekt der Dynamik und stellt APClustering vor, ein Clusteringverfahren zur Bestimmung unterschiedlicher Ausführungsphasen, das zur Unterteilung eines Prozesslogs dienen kann. Abb. 2.12 veranschaulicht den Zusammenhang dieser Ansätze und den dadurch erzielten Mehrwert für die Anwendung von Process Mining zur Sicherheitsanalyse von Geschäftsprozessen. Durch den entwickelten Clusteringansatz kann ein Prozesslog anhand unterschiedlicher Ausführungsphasen partitioniert und anschließend für jede Partition separat ein (Struktur-)Modell rekonstruiert werden. Für die Modellrekonstruktion steht neben den o.g. Möglichkeiten mit dem Resultat dieser Arbeit auch ein Verfahren zur Rekonstruktion eines präzisen Kontroll-/Datenflussmodells zur Verfügung.



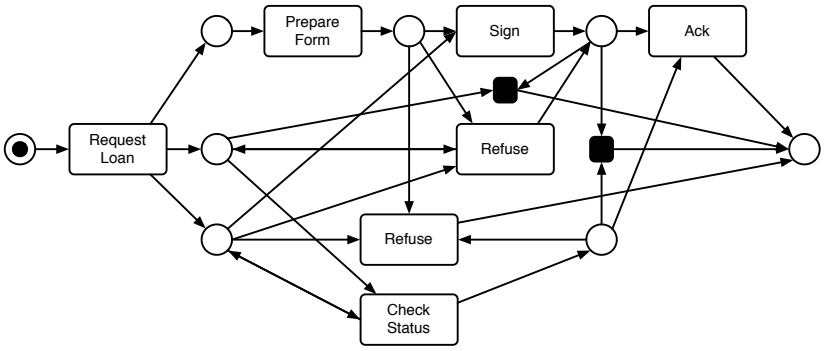
(a) Prozessmodell für geplantes Verhalten.



(b) Rekonstruktionsergebnis (heuristisch).



(c) Rekonstruktionsergebnis (Fuzzy).



(d) Rekonstruktionsergebnis (Synthese).

Abbildung 2.13.: Rekonstruktionsergebnisse unterschiedlicher Ansätze im Vergleich.

Teil I.

Rekonstruktion präziser Prozessmodelle

Kapitel 3

Bestehende Verfahren zur Kontrollfluss-Rekonstruktion

Bezüglich der Rekonstruktion von Prozessen auf Basis von Aufzeichnungen beobachteten Verhaltens gibt es eine Fülle unterschiedlicher Ansätze, die sich in die vier in Abschnitt 2.2 dargelegten Perspektiven einteilen lassen. Mechanismen, die sowohl den Kontroll- als auch den Datenfluss eines Prozesses rekonstruieren, existieren derzeit jedoch nicht. Nachdem Methoden zur Datenfluss-Rekonstruktion bereits in Abschnitt 2.2.1 vorgestellt wurden, betrachtet dieses Kapitel eingehend die vielfältigen Methoden zur Kontrollfluss-Rekonstruktion und deren Eignung im Hinblick auf nachgelagerte Prozessanalysen.

Die Kontrollfluss-Rekonstruktion konzentriert sich auf die Struktur eines einem Prozesslog zugrundeliegenden Modells. Grundsätzlich geht es dabei nicht um das Finden *des einen* Prozessmodells (unter bestimmten Rahmenbedingungen kann es unendlich viele mögliche Modelle geben), sondern um die Konstruktion eines möglichst adäquaten. Im Folgenden werden zunächst einige Konzepte erläutert, die genutzt werden, um Mechanismen zur Kontrollfluss-Rekonstruktion zu charakterisieren und hinsichtlich ihrer Qualität zu beurteilen. Dazu werden zunächst Bewertungskriterien für die Beurteilung der Qualität rekonstruierter Prozessmodelle vorgestellt und anschließend gruppiert nach dem verwendeten Metamodell für die Darstellung rekonstruierten Prozessverhaltens konkrete Ansätze erläutert und im Bezug auf die sicherheitsorientierte Rekonstruktion bewertet.

Anhand Abb. 3.1 lassen sich neben der Häufigkeit publizierter Ansätze zur Prozessrekonstruktion auch Tendenzen hinsichtlich verwendeter Metamodelle erkennen. Während sich erste Arbeiten hauptsächlich auf gerichtete Graphen und Zustandsautomaten stützen, ist eine deutliche Tendenz zu Petrinetzen zu erkennen. Dieser Trend wird durch weitere Publikationen aus anderen Bereichen des Process Mining bestätigt, was hauptsächlich darauf zurück zu führen ist, dass Petrinetze im Vergleich zu zustandsorientierten Metamodellen (z.B. Transitionssysteme) geeigneter sind, nebenläufiges Verhalten und kausale Beziehungen zwischen einzelnen Aktivitäten innerhalb eines (Geschäfts-)Prozesses darzustellen[1]. Ihre formale Semantik und die Verfügbarkeit eindeutiger Definitionen für Gestalt

[1] Vgl. Cortadella et al. 1995.

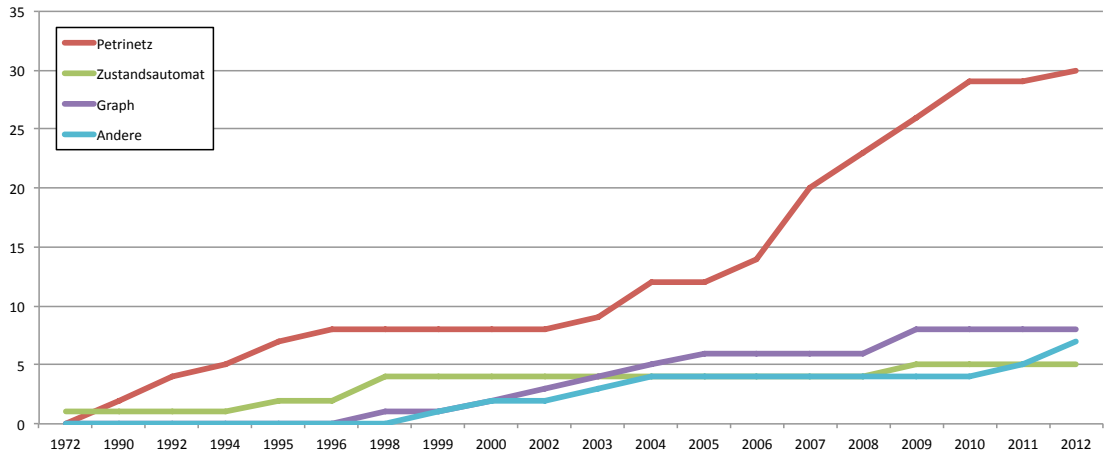


Abbildung 3.1.: Anzahl von Verfahren im zeitlichen Verlauf gruppiert nach verwendeten Metamodellen für die Darstellung rekonstruierten Prozessverhaltens.

und Verhalten verschiedener Petrinetzdialekte ermöglicht zudem vielfältige Analysemöglichkeiten[2]. Fundierte Forschungsergebnisse zu petrinetzspezifischen Eigenschaften wie Beschränktheit oder Erreichbarkeit können auf Geschäftsprozesse angewandt und konkretisiert werden (z.B. valide Terminierung eines Ablaufs). Es existieren sowohl Arbeiten zur Definition von Sicherheitseigenschaften auf Petrinetzen[3] als auch Ansätze zur Verifikation von Zugangs- und Nutzungskontroll-Richtlinien[4], Integritätsbedingungen[5] und Anforderungen der Informationsflusskontrolle[6], sowie der Erkennung von Interessenkonflikten[7]. Dieses Kapitel wird der Vollständigkeit halber einen Überblick über das gesamte Spektrum an Rekonstruktionsmechanismen bieten, sich bei der Bewertung und Auswahl geeigneter Mechanismen im Hinblick auf die Zielsetzung dieser Arbeit jedoch aus o.g. Gründen ausschließlich auf Petrinetze konzentrieren.

Parallel zu dieser analytisch motivierten Bewegung in Richtung Petrinetzen gibt es auch den fortwährenden Versuch, Process Mining mit Metamodellen zu koppeln, die in der Praxis verwendet werden. Die Frage nach dem am häufigsten anzutreffenden Metamodell ist schwer zu beantworten. Diesbezügliche Erhebungen[8] sind oft unvollständig bzw. widersprüchlich und schwanken zwischen UML[9], BPEL[10], BPMN[11] und EPCs[12]. Einer 2011 durchgeführten Umfrage nach zu Folge, präferieren zumindest Prozessverantwortliche und Prozessanalysten (hauptsächlich in Nordamerika und Europa) für die Modellierung von Geschäftsprozessen BPMN[13].

[2] Vgl. Aalst 1996; Aalst 1998; C. A. Ellis et al. 1993.
[3] Vgl. Adam et al. 1998; Huang et al. 2009; Huang et al. 2011; Huang et al. 2013.
[4] Vgl. Jiang et al. 2004; Katt et al. 2009.
[5] Vgl. Zhang, Hong und Xiao 2006.
[6] Vgl. Accorsi und Lehmann 2012; Accorsi und Wonnemann 2011.
[7] Vgl. Zhang, Hong und Liao 2006.
[8] Vgl. Lu et al. 2007.
[9] Vgl. ISO/IEC 19505-1 2012; ISO/IEC 19505-2 2012.
[10] Vgl. OASIS 2007.
[11] Vgl. ISO/IEC 19510 2013.
[12] Vgl. Keller et al. 1992.
[13] Vgl. C. Wolf et al. 2011; Man 2009.

3.1 Bewertungskriterien

Die Bewertung der Qualität eines rekonstruierten Prozessmodells hängt in erster Linie vom eingenommenen Betrachtungswinkel und der Zielsetzung des Rekonstruktionsunterfangens ab. Eine in der Literatur häufig anzutreffende Sichtweise stützt sich dabei auf das sog. *rediscovery problem*. Dabei wird ein Prozesslog mittels Erzeugung valider Pfade eines präexistenten Prozessmodells generiert. Anschließend wird das mithilfe der Rekonstruktion entstandene Prozessmodell daraufhin überprüft, ob es die strukturellen Eigenschaften des Ursprungsmodells korrekt wiedergibt. Eine wichtige Rolle spielen hier mehrfach auftretende Aktivitäten (*duplicate tasks*), die im Prozesslog nicht mehr unterschieden werden können, unsichtbare Aktivitäten (*silent/invisible tasks*, in Petrinetzen *stille Transitionen*), die im Ursprungsmodell lediglich Routingzwecken dienen und Eigenschaften wie *(non) free-choice*, die Verwendung in Wohlgeformtheitsdefinitionen von Prozessen finden.

Werden rekonstruierte Prozessmodelle zur Darstellung tatsächlichen Prozessverhaltens verwendet, spielen Modelle für das geplante Prozessverhalten eine untergeordnete Rolle. Maßgeblich ist in diesem Fall, dass das Verhalten des rekonstruierten Modells dem Verhalten des Prozesslogs entspricht. Dieser Abschnitt betrachtet deshalb ausschließlich das Verhältnis von Prozesslogs und daraus rekonstruierter Modelle. Die Überprüfung einer solchen Übereinstimmung erfolgt typischerweise entlang vier Qualitätskriterien:

1. **Deckung**. Das Deckungskriterium (engl.: *fitness*) bewertet die Qualität eines rekonstruierten Modells anhand des Anteils von Verhalten gemäß Prozesslog, welches vom Modell korrekt wiedergegeben werden kann; also inwiefern das Modellverhalten das Logverhalten inkludiert bzw. abdeckt. Die Übereinstimmung wird bei sog. *tokenbasierter* Deckung mithilfe von *replay*[14] getestet. Dabei werden einzelne, in sich abgeschlossene Abläufe im Log auf dem Modell „abgespielt“. Ein solcher Vorgang ist erfolgreich, wenn der Ablauf von Anfang bis Ende im Modell abgespielt werden kann und dieses dabei in einen gültigen Endzustand überführt. Trifft dies für alle Abläufe zu, spricht man von *perfect fitness* (zu Deutsch: vollständige Deckung).

   Daneben gibt es noch *alignmentbasierte* Deckung, bei der für jeden Ablauf im Log ein möglichst ähnlicher Ablauf im Modell gesucht wird. Diese Vorgehensweise ist im weitesten Sinne als *Pattern Matching* zu verstehen und erinnert stark an das Sequenzalignment der Bioinformatik zur Feststellung von Ähnlichkeiten in biologischen Sequenzen. Typischerweise wird eine Kostenfunktion definiert, die fehlerhafte Übereinstimmungen in Sequenzen sanktioniert; gesucht sind kostenminimale Anordnungen.

2. **Affinität**. Im Gegensatz zur Deckung betrachtet Affinität (engl.: *precision*) die Wesensverwandtschaft von Log und Modell im Sinne zusätzlichen Verhaltens vom Modell, welches über das Prozessverhalten gemäß Log hinausgeht. Abschätzungen dieses Übermaßes können hierbei auch anhand von Alignments erfolgen[15]. Aufgrund der hohen Berechnungskomplexität, bedingt durch die Berechnung des Zustandsraums eines Modells, wurden aber auch Ansätze vorgeschlagen, die stattdessen

[14] Vgl. Adriansyah et al. 2011; Aalst et al. 2012.
[15] Vgl. Adriansyah, Munoz-Gama et al. 2012.

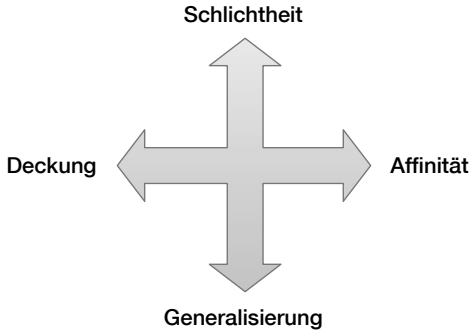


Abbildung 3.2.: Qualitätsdimensionen rekonstruierter Prozessmodelle.

sog. *escaping edges* betrachten (Areale im Zustandsraum eines Modells, in denen es zu Abweichungen vom Logverhalten kommt), diese charakterisieren und quantifizieren[16]. Lässt das Modell kein über das im Prozesslog enthaltene Verhalten zu, spricht man von *perfect precision* (zu Deutsch: völlige Affinität).

3. **Generalisierung**. Dieses Kriterium hat seinen Ursprung in der traditionellen Zielsetzung von Process Mining, nicht das exakte Verhalten eines Prozesslogs im Modell abzubilden, sondern stattdessen das am häufigsten auftretende. Der Generalisierungsfaktor eines Modells gibt an, wie strikt es sich am Verhalten gemäß Prozesslog orientiert. Je höher er ausfällt, umso wahrscheinlicher ist es, dass zukünftiges Verhalten ebenfalls abgedeckt wird[17]. Generalisierung ist schwer zu messen, da sie sich auf unbekanntes Verhalten bezieht. Es existieren aber Ansätze, die anhand der Anzahl unterschiedlicher gewählter Pfade an Kreuzungsstellen im Prozess und deren Frequentierung abzuschätzen versuchen, mit welcher Wahrscheinlichkeit in Zukunft an derselben Stelle ein neuer Pfad gewählt wird.

4. **Schlichtheit**. Die Schlichtheit eines Modells wird hauptsächlich aus Gründen der Gebrauchstauglichkeit bei der manuellen Analyse rekonstruierter Modelle betrachtet. Dabei wird die strukturelle Komplexität des Modells bspw. anhand der Anzahl von Knoten und Verbindungen betrachtet, aber auch mithilfe von Wohlgeformtheitseigenschaften[18]. Diesbezüglich wird häufig auf *Ockhams Rasiermesser* (auch Prinzip der Parsimonie) verwiesen, ein Prinzip, das Sparsamkeit bei der Erklärung eines Sachverhalts fordert.

Die Herausforderung bei der Rekonstruktion ist es, einen sinnvollen Kompromiss zwischen diesen Kriterien zu finden, die in Konkurrenz stehen und isoliert betrachtet meist nicht zu aussagekräftigen Ergebnissen führen[19]. Ausgehend von einem Prozesslog, der ausschließlich Abläufe der Form $\langle A, B, D\rangle$, $\langle A, C, D\rangle$ und $\langle A, B\rangle$ enthält, veranschaulicht Abb. 3.3 (a) ein Modell mit vollständiger Deckung. Einerseits können alle Abläufe vom Modell wiedergegeben werden, andererseits hat dieses Modell eine schlechte Präzision, da es sehr viel zusätzliches Verhalten zulässt. Das Modell in Abb. 3.3 (b) weist zwar

[16] Vgl. Munoz-Gama et al. 2010.
[17] Vgl. Buijs et al. 2012.
[18] Vgl. Rozinat et al. 2008.
[19] Vgl. Buijs et al. 2012.

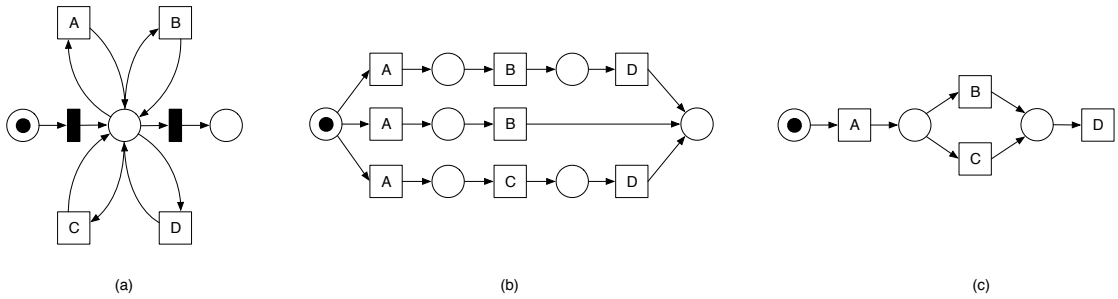


Abbildung 3.3.: Modelle unterschiedlicher Charakterisierungen bzgl. der Qualitätskriterien Deckung, Affinität, Generalisierung und Schlichtheit für einen Prozesslog mit Einträgen der Form $\langle A, B, D\rangle$, $\langle A, C, D\rangle$ und $\langle A, B\rangle$.

sowohl vollständige Deckung als auch völlige Affinität auf, stellt das Prozessverhalten allerdings unnötig komplex dar. Jeder unterschiedliche Ablauf wird sequentiell wiedergegeben, was zu mehrfach auftretenden Aktivitäten führt. Abb. 3.3 (c) zeigt, dass dies im vorliegenden Fall vermeidbar ist und eine kompaktere Repräsentation existiert. Ob dieses Modell tatsächlich ein guter „Kandidat“ ist, hängt im traditionellen Process Mining von der Häufigkeit der im Prozesslog enthaltenen Abläufen ab. Tritt bspw. der Ablauf $\langle A, B\rangle$ lediglich einmal auf, beide anderen Abläufe hingegen 1000 mal, könnte man das Modell in Abb. 3.3 (c) durchaus als gute Repräsentation des realen Prozessverhaltens auffassen.

Abläufe mit geringer Häufigkeit sind entweder das Resultat unvollständiger Aufzeichnung von Prozessverhalten oder sie stehen für tatsächlich stattgefundenes Verhalten. In diesem Fall kann es sich schlicht um Ausnahmefälle handeln, die selten auftreten, jedoch ihre Berechtigung haben oder aber um ungewollte Abweichungen von geplantem Prozessverhalten. In dem Bestreben, einen möglichst kompakten Überblick über das Prozessverhalten zu bieten, werden solche Abläufe als Rauschen (*noise*) aufgefasst, von dem in vielen Fällen abstrahiert wird, um das resultierende Modell nicht unnötig zu verkomplizieren. Die Robustheit gegenüber Rauschen ist deshalb ein etabliertes Qualitätsmerkmal von Rekonstruktionsmechanismen. Im Sicherheitskontext ist es jedoch nicht brauchbar, da gerade seltene Prozessausprägungen interessant für eine nähere Betrachtung sind, da diese *outlier* nicht selten Hinweise auf fehlerhaftes oder gar betrügerisches Verhalten von Prozessteilnehmern hinweisen.

Aus dem Blickwinkel sicherheitsorientierter Betrachtungen sind nicht alle traditionellen Bewertungskonzepte des Process Mining hilfreich für die Beurteilung der Qualität rekonstruierter Prozesse. Aus dem in Abschnitt 2.3 abstrakt beschriebenen Defizit bzgl. der Qualität rekonstruierter Prozessmodelle **(AA1)** können unter Zuhilfenahme vorgestellter Qualitätsmaße auf technischer Ebene folgende konkrete Anforderungen abgeleitet werden:

(KA1) Vollständige Deckung. Anforderung zur Vermeidung von Modellen nicht ausreichender Präzision, die sich in der Unvollständigkeit hinsichtlich tatsächlichem Prozessverhalten manifestiert. Notwendige Voraussetzung um zu gewährleisten, dass das beobachtete Prozessverhalten vollständig im Modell repräsentiert ist. An-

dernfalls vermittelt das Modell einen verfälschten Eindruck der tatsächlichen Prozessausführung und begünstigt Fehlinterpretationen.

(KA2) Völlige Affinität. Anforderung zur Vermeidung unpräziser Modelle; in diesem Fall Abweichung von bzw. fehlender Bezug zu tatsächlichem Prozessverhalten.

(KA3) Keine Generalisierung. Diese Anforderung bezieht sich ebenfalls auf die geforderte Präzision rekonstruierter Modelle. Bei einer Generalisierung entsteht notwendigerweise eine Abstraktion von beobachtetem Prozessverhalten und damit zusätzliches Verhalten, was die Anforderung **(KA2)** verletzt. Eine Implikation von **(KA3)** ist, dass rekonstruierte Modelle nur beschränkte Schleifen enthalten dürfen.

(KA4) Aussagekraft. Im Hinblick auf Schlichtheit steht diese Anforderung für die Notwendigkeit der Erkennung von Kontrollflussstrukturen, die Beziehungen zwischen unterschiedlichen Prozessaktivitäten möglichst explizit darstellen. Werden solche Strukturen nicht erkannt, ergeben sich entweder wenig aussagekräftige Modelle (vgl. Abb. 3.3 (b)) oder die Beziehung zwischen Aktivitäten (z.B. wechselseitiger Ausschluss) können schwerer nachvollzogen werden (vgl. Fuzzymodelle). Um ein möglichst großes Analysespektrum zu ermöglichen, sollten also nach Möglichkeit schlichte Modelle bevorzugt werden. Insofern bezieht sich diese Anforderung auf den Informationsgehalt von Modellen, der durch mangelhafte Qualität geschmälert werden kann. Schlichte Modele haben darüber hinaus die Tendenz, sich positiv auf die intuitive Wahrnehmung eines Prozesses durch einen Anwender auszuwirken.

3.2 Verfahren auf Basis von Graphen

Mit einem Rekonstruktionsansatz basierend auf gerichteten Graphen übertragen Agrawal et al.[20] 1998 erstmals das Rekonstruktionsproblem auf Geschäftsprozesse und legen gleichzeitig den Grundstein für weitere Rekonstruktionsverfahren auf Basis von Graphen. Dabei wird ein Graph $V = (N, E)$, der im Initialzustand ausschließlich Knoten mit einer eindeutigen Zuordnung zu den unterschiedlichen im Prozesslog auftretenden Prozessaktivitäten enthält, sukzessive mit Kanten angereichert. Kanten zwischen Knoten stehen dabei für die kausale Abhängigkeit entsprechender Aktivitäten. Die Extraktion kausaler Abhängigkeiten erfolgt auf Fall-Ebene, d.h. jeder in sich abgeschlossene Prozessablauf im Log wird einzeln abgearbeitet. Für eine gegebene Aktivität a innerhalb eines Falls, wird zunächst für jeden Nachfolger b eine Kante $a \rightarrow b$ im Graph eingefügt. In einem weiteren Schritt wird die transitive Reduktion des Graphen bestimmt, d.h. eine kantenminimale Version von E mit derselben transitiven Hülle berechnet. Kanten, deren Umkehrung ebenfalls im Graph enthalten ist, werden als Indikator für Nebenläufigkeit interpretiert und entfernt. Das Verfahren wurde zunächst auf azyklischen Graphen definiert, da für die transitive Reduktion andernfalls keine eindeutige Lösung existiert. Die Verallgemeinerung auf zyklische Graphen ist jedoch mittels einer Umbenennung mehrfach auftretender Aktivitäten innerhalb eines Falls möglich. Interessanterweise kommt bereits hier eine auf Grenzwerten für die Konfidenz identifizierter Kausalzusammenhänge basierende Strategie für die Behandlung von Rauschen zum Einsatz, die in ähnlicher Art und Weise bei

[20] Vgl. Agrawal et al. 1998.

späteren abstraktionsbasierten Ansätzen (speziell beim α-Algorithmus und Derivaten) Anwendung findet.
Der Ansatz von Hang et al.[21] kann als Erweiterung des Ansatzes von Agrawal et al. gesehen werden, der für Prozessaktivitäten definierte Start- und Endzeitpunkte für deren Bearbeitung unterscheidet, um kausale Abhängigkeiten präziser zu erkennen und neben einer Graph-Struktur zusätzliche Kantenbedingungen in Form boolscher Funktionen extrahiert. Diese Funktionen beschreiben notwendige Bedingungen für die Fortführung von Teilpfaden abhängig von zuvor stattgefundenen Aktivitäten und werden mithilfe von Data-Mining Klassifikationstechniken gewonnen.
Golani und Pinter[22] verwenden ebenfalls Start- und End-Zeitpunkte und leiten Parallelität von Prozessaktivitäten aus sich überschneidenden zeitlichen Intervallen ab. Bei der Konstruktion des Ergebnisgraphen verfolgen sie einen zweistufigen Ansatz, der zunächst für jeden Fall einen separaten Graph erzeugt und diese dann in einem zweiten Schritt vereint. Ebenfalls auf azyklische Graphen beschränkt, nutzt das Verfahren von Silva et al.[23] probabilistische Modelle für die Rekonstruktion und weist Knoten innerhalb eines Graphen bedingte Auftrittswahrscheinlichkeiten bzgl. ihrer Vorgänger zu. Eine interessante Eigenschaft dieses Ansatzes ist, dass resultierende Graphen sog. *hidden nodes* enthalten können, also Knoten, die nicht mit der Ausführung einer Prozessaktivität assoziiert sind, sondern Routingzwecken dienen.
Manila et al.[24] betrachten statt eines Graphen eine Menge von Halbordnungen über der Menge unterschiedlicher Prozessaktivitäten im Log als Ergebnis der Rekonstruktion. Jede Halbordnung kann eine bestimmte Menge von Sequenzen „erzeugen“. In einem iterativen Verfahren wird eine triviale Halbordnung schrittweise verfeinert um einen gegebenen Prozesslog möglichst gut zu beschreiben. In jeder Iteration wird für die aktuell beste Halbordnung zunächst eine Menge von Varianten berechnet und daraus der beste Kandidat für den nächsten Durchlauf gewählt. Die Berechnung terminiert, wenn kein signifikant besserer Kandidat mehr gefunden werden kann. Da das Verfahren zyklische Bezüge vernachlässigt und mehrfach auftretende gleiche Aktivitäten innerhalb eines Prozessablaufs ausschließt, ist es jedoch nur beschränkt einsetzbar.
Greco et al.[25] verfolgen eine andere Strategie und extrahieren statt einem einzigen Modell eine Modellhierarchie. Jedes Modell stellt einen Prozesslog auf einem spezifischen Abstraktionsniveau dar und verkörpert eine Menge von *workflow patterns*, die sich wiederum auf eine Untermenge der Prozessabläufe im Log beziehen. Patterns sind ebenfalls Graphen, die sich jedoch auf Teilaspekte des Prozessverhaltens konzentrieren. Modelle auf höheren Abstraktionsebenen subsumieren das Prozessverhalten von Modellen in tieferen Schichten.
Ein weiterer Ansatz wurde von Folio et al.[26] vorgeschlagen. Durch geschickte Konstruktion und Modifikation eines Abhängigkeitsgraphen, der direkte kausale Zusammenhänge von Prozessaktivitäten (benachbarte Aktivitäten) modelliert, kann sowohl Rauschen als auch mehrfaches Auftreten von Aktivitäten innerhalb von Prozessabläufen behandelt werden.

[21] Vgl. Hwang et al. 2002.
[22] Vgl. Golani et al. 2003; Pinter et al. 2004.
[23] Vgl. Silva et al. 2005.
[24] Vgl. Mannila et al. 2000.
[25] Vgl. Greco et al. 2004; Greco, Guzzo, Manco et al. 2005.
[26] Vgl. Folino et al. 2009.

Um auch Situationen abfangen zu können, in denen keine Zuordnung von Einzelaktivitäten zu Prozessabläufen (Fällen) gegeben ist, verwenden Ferreira et al.[27] einen *Expectation-Maximization*-Ansatz, um Aktivitäten zu gruppieren.

3.3 Verfahren auf Basis von Zustandsautomaten

Die Wurzeln von Mechanismen zur Rekonstruktion von Prozessmodellen in Form von Transitionssystemen sind stark an den Begriff *Synthese* gekoppelt und reichen zurück auf Arbeiten von Nerode[28] (Grundlagen für die Synthese von Zustandsautomaten). Frühe Ansätze haben keinen expliziten Bezug zu Process Mining oder allgemein Geschäftsprozessen. Cook und Wolf[29] betrachten Rekonstruktion im Hinblick auf Softwareprozesse und stellen mit `Rnet`, `Ktail` und `Markov` drei unterschiedliche Methoden für die Erstellung eines endlichen Zustandsautomaten (ZA) vor. `Rnet` verwendet Neuronale Netze, um ein geeignetes Modell mithilfe eines simulierten Lernprozesses zu erstellen. Im Wesentlichen „lernt“ das Netzwerk Wahrscheinlichkeiten für das Auftreten von Ereignissen basierend auf bisher observiertem Verhalten. Eine explizite Steuerung des Lernprozesses zur Erzielung eines möglichst präzisen Ergebnisses ist nicht möglich, sodass resultierende Modelle meist über das „gesehene“ Verhalten hinausgehen. Andererseits macht genau diese Eigenschaft `Rnet` robust gegenüber Rauschen. `Ktail` wählt zur Synthese von ZAs eine ausschließlich algorithmische Vorgehensweise nach dem Vorbild früherer Arbeiten von Biermann et al.[30]. Zustände innerhalb des ZAs werden dabei anhand potentieller Fortführungen (*future*) einer bisherigen Folge von Ereignissen (*history*) definiert. Die Anzahl der Ereignisse die in die Berechnung der Fortführungen einfließen ist parametrisierbar (Parameter k); die Fortführungen der Länge k selbst werden als *(k-)tail* bezeichnet. Abhängig von k und den Eingabesequenzen, können resultierende ZAs nichtdeterministisch sein. Dies kann nur verhindert werden, indem k genügend groß gewählt wird, i.e. entsprechend der maximalen Länge einer Eingabesequenz. Der `Markov`-Ansatz nutzt Markov-Ketten n-ter Ordnung, um Modelle auf Basis von Tabellen zu bilden, die Wahrscheinlichkeiten für das Eintreten folgender Ereignisse basierend auf der bisherigen Ereignishistorie bilden. `Markov` arbeitet ähnlich wie `Ktail`, betrachtet aber statt zukünftigem Verhalten die Historie. Außerdem ist `Markov` durch die Nutzung von Mindestwerten für Übergangswahrscheinlichkeiten robuster gegenüber Rauschen und konnte von Cook et al.[31] um Mechanismen zur zuverlässigen Erkennung von Nebenläufigkeit erweitert werden. Sehr ähnliche Konzepte basierend auf Markov-Ketten werden auch von Datta[32] verwendet.

Methoden zur Rekonstruktion von ZAs oder genereller Transitionssystemen bilden die Grundlage für mehrstufige Ansätze, die aus solchen zustandsbasierten Modellen in einem zweiten Schritt Petrinetze erzeugen (vgl. ZBR-Algorithmen in Abschnitt 4.2). Eine besonders flexible Möglichkeit zur Erzeugung eines Transitionssystems wird von Aalst et

[27] Vgl. Ferreira et al. 2009.
[28] Vgl. Nerode 1958.
[29] Vgl. Cook und A. L. Wolf 1995; Cook und A. L. Wolf 1998a.
[30] Vgl. Biermann et al. 1972.
[31] Vgl. Cook und A. L. Wolf 1998b; Cook, Du et al. 2004.
[32] Vgl. Datta 1998.

al.[33] beschrieben. Durch die Verwendung von sog. Abstraktionen (engl.: *abstractions*), kann die Zustandsbildung und damit der Detailgrad und die Wiedergabetreue des TS direkt beeinflusst werden. Diese Methode wird für den in dieser Arbeit vorgestellten zweistufigen Ansatz zur Rekonstruktion präziser Strukturmodelle genutzt und deshalb gesondert in Abschnitt 4.5.1 betrachtet.

3.4 Verfahren auf Basis von Petrinetzen

Abstraktionsbasierte Ansätze. Der bekannteste Vertreter petrinetzbasierter Rekonstruktionsverfahren ist der α-Algorithmus[34]. Aalst et al. orientieren sich in deren Arbeit sehr stark am *rediscovery*-Problem und schlagen ein Verfahren vor, das auf der Klasse von strukturierten Workflowmodellen (*structured workflow models*) die Übereinstimmung von Ursprungsmodell und rekonstruiertem Modell garantiert. Ursprungsmodelle dürfen dabei keine kurzen Schleifen (Schleifen der Länge 1 und 2) enthalten und auch keine Redundanz in Form von *implicit places*, i.e. Stellen, deren Entfernen das Verhalten des Petrinetzes nicht beeinflusst. Der α-Algorithmus zählt zu den abstraktionsbasierten Verfahren, da er auf der Extraktion von Abhängigkeitsrelationen zwischen Prozessaktivitäten aus dem Log basiert und aus diesen in einem zweiten Schritt Petrinetzstrukturen ableitet. Konkret wird eine Basisrelation (*follows*) für direkte Folgebeziehungen verwendet und drei davon abgeleitete Relationen. Aktivitäten a, b haben einen kausalen Zusammenhang (*causal*) wenn b direkt auf a folgt, aber nicht umgekehrt. Als Parallel (*parallel*) werden Aktivitäten angesehen, wenn sowohl a auf b folgt, als auch umgekehrt. Falls a und b in keiner Folgebeziehung stehen, werden sie als unzusammenhängend (*unrelated*) aufgefasst. Eine zentrale Annahme des Ansatzes ist die Vollständigkeit des Prozesslogs hinsichtlich der *follows*-Relation. D.h. für jede direkte Folgebeziehung von Aktivitäten, die laut Ursprungsmodell existiert, muss mindestens eine Ausführung im Prozesslog vorhanden sein, die diese Beziehung enthält. Diese Annahme ist weniger strikt als die Forderung, dass jeder mögliche Ausführungspfad des Modells im Log enthalten sein muss. Existieren bspw. zwei Aktivitätssequenzen $\langle A, B, C, D, E\rangle$ und $\langle B, A, C, E, D\rangle$, so ist die Existenz von $\langle B, A, C, D, E\rangle$ nicht erforderlich. Diese Vollständigkeitsforderung ist für praktische Anwendungen jedoch sehr restriktiv und kann typischerweise weder garantiert noch überprüft werden. Dennoch ist der α-Algorithmus vor allem in theoretischer Hinsicht bzgl. des *rediscovery*-Problems interessant.

Es existiert eine Reihe von Ansätzen, die den α-Algorithmus erweitern und diverse Schwächen beheben. So schlagen de Medeiros et al.[35] mit α^+ eine Erweiterung vor, die mithilfe einer Anpassung der Vollständigkeits-Forderung und der Modifikation von Ordnungsrelationen auch Urpsrungsmodelle mit kurzen Schleifen rekonstruieren kann. Li et al.[36] (α^*) konzentrieren sich auf die Unterstützung von mehrfach auftretenden Aktivitäten im Ursprungsmodell und ergänzen den ursprünglichen Ansatz um einen entsprechenden

[33] Vgl. Aalst, Rubin et al. 2010.
[34] Vgl. Aalst, Weijters et al. 2004.
[35] Vgl. Alves de Medeiros et al. 2004b; Alves de Medeiros et al. 2004a.
[36] Vgl. Li et al. 2007.

vorgelagerten Erkennungsmechanismus. Wen et al.[37] (α^{++}) befassen sich mit sog. *non-free choice* Konstrukten, die implizite Abhängigkeiten von Aktivitäten zur Folge haben. Solche indirekten Abhängigkeiten können von Methoden die auf direkten Folgebeziehungen operieren nicht erkannt werden, was zu fehlenden Stellen oder Verknüpfungen im rekonstruierten Petrinetz führt. Für die Rekonstruktion von unsichtbaren Aktivitäten, schlagen Wen et al.[38] den $\alpha^{\#}$-Algorithmus vor. Mit dem β-Verfahren konnte außerdem nachgewiesen werden, dass die Betrachtung von expliziten Start- und Endzeitpunkten für einzelne Prozessschritte die Erkennung von Parallelität erheblich erleichtert[39].

Abstraktionsbasierte Verfahren stellen für die Rekonstruktion präziser Modelle sehr restriktive Annahmen an den Informationsgehalt von Prozesslogs, die in den seltensten Fällen (weder unter künstlichen Bedingungen bei der Simulation von Prozessen und noch weniger unter realen Bedingungen) gewährleistet werden können[40]. Erwartungsgemäß sinkt die Qualität generierter Modelle signifikant bei Anwendung auf „unvollständigen" Prozesslogs.

Heuristische Verfahren. Trotz der vielfältigen Erweiterungen kommen abstraktionsbasierte Verfahren bei vorhandenem Rauschen an ihre Grenzen. Heuristische Verfahren berücksichtigen deshalb die Häufigkeit von Folgebeziehungen und leiten daraus die Stärke bzw. die Verlässlichkeit von abgeleiteten Beziehungen ab. Auf diese Weise kann durch geeignete Grenzwerte von seltenem Prozessverhalten abstrahiert und ein Prozessmodell mit der gewünschten Granularität erzeugt werden. Der *little thumb*-Ansatz von Weijters et al.[41] erweitert die Abhängigkeitsrelationen des α-Algorithmus um globalere Parameter wie die Häufigkeit einer Aktivität im Log und leitet daraus zunächst einen Abhängigkeitsgraphen ab, mit dessen Hilfe kausale Abhängigkeiten eindeutiger und von globalerem Standpunkt aus bestimmt werden können. Für die Konstruktion eines Petrinetzes wird auf den α-Algorithmus zurückgegriffen. Weijters et al.[42] bauen auf diese Methode auf, beschreiben jedoch nicht direkt eine Petrinetz Rekonstruktion, sondern beschränken sich auf sog. *Causal Matrices*.

Heuristische Verfahren schneiden hinsichtlich Deckung und Affinität geringfügig besser ab als abstraktionsbasierte Verfahren, produzieren jedoch Modelle mit stillen Transitionen die sich negativ auf ein intuitives Verständnis des Prozesses auswirken.

Genetische Ansätze. Die Problematik vieler bestehender Rekonstruktionsverfahren, durch die Verwendung von Ordnungs- und Abhängigkeitsrelationen zu lokalen Minima zu tendieren, wird von de Medeiros et al.[43] durch eine globale Lernstrategie umgangen, die an Lernprozesse im biologischen Umfeld angelehnt ist und evolutorische Entwicklungsprozesse nachahmt. Für die Bildung des Initialmodells wird auf die heuristischen Techniken

[37] Vgl. Wen, Aalst et al. 2007.
[38] Vgl. Wen, Wang und Sun 2007.
[39] Vgl. Wen, Wang, Aalst et al. 2009.
[40] Vgl. Dongen, Alves de Medeiros et al. 2009.
[41] Vgl. Weijters und Aalst 2003.
[42] Vgl. Weijters, Aalst und Alves de Medeiros 2006.
[43] Vgl. Alves de Medeiros et al. 2004c; Aalst, Alves de Medeiros et al. 2005; Alves de Medeiros 2006; Alves de Medeiros et al. 2007.

von Weijters et al. zurückgegriffen; die Anwendung genetischer Operatoren (*crossover* für die Kombination von Modellen und *mutation* für randomisierte Variantenbildung) auf Individuen einer Population führt zur Bildung neuer Individuen und formt eine neue Population. Das Haltekriterium wird anhand des Deckungskriteriums formuliert. Für die Reduktion der Berechnungskomplexität des Verfahrens schlagen Bratosin et al.[44] eine Erweiterung vor, die für die Berechnung des Deckungsgrades auf ein Sample des Prozesslogs zurückgreift, statt auf die Gesamtpopulation der Ausführungspfade.

Genetische Ansätze erreichen mitunter enorm hohe Deckungsgrade, allerdings muss dafür auch erheblicher Rechenaufwand in Kauf genommen werden. Hinsichtlich der Affinität fallen Testergebnisse oft sehr unterschiedlich aus, da bestehende Verfahren sich bei der Lenkung des evolutorischen Näherungsverfahrens hauptsächlich auf Deckung konzentrieren. Generell gibt es bei dieser Klasse von Ansätzen keine Garantie, dass ein Modell mit hoher Präzision gefunden wird, weshalb eine Erweiterung bestehender Verfahren als nicht sinnvoll erscheint.

Regionsbasierte Ansätze. Eine ganze Klasse von Ansätzen beschäftigt sich mit der Anwendung von Methoden aus der *Petrinetzsynthese* auf das *Process Discovery* Problem. Ausgehend von derselben theoretischen Grundlage in Form von *Regionen* (engl.: *theory of regions*)[45] werden grundsätzlich zwei Richtungen unterschieden. In beiden Fällen besteht das erklärte Ziel darin, eine Petrinetzdarstellung eines Prozesslogs zu generieren, welche vollständige Deckung und minimale Generalisierung aufweist. Transitionen des Petrinetzes entsprechen Aktivitäten im Prozesslog; Stellen werden aus Regionen abgeleitet.

Sprachbasierte Regionsalgorithmen (SBR) definieren Regionen als Lösungen eines linearen Ungleichungssystems, wobei Ungleichungen erforderliche Eigenschaften für den Deckungsgrad des resultierenden Netzes kodieren. In vielen Fällen ist die Menge der Regionen und damit die Zahl der möglichen Stellen im resultierenden Petrinetz unendlich, weshalb verschiedene Methoden vorgeschlagen wurden, um eine endliche Repräsentation (*finite basis*) der Lösungsmenge zu berechnen, die dennoch hinreichend für die Gewährleistung der gewünschten Deckungseigenschaft ist. Bergenthum et al.[46] unterscheiden deshalb zwischen notwendigen Regionen (Basisregionen) und redundanten Regionen, die als Kombination von Basisregionen dargestellt werden können und deshalb keinen Mehrwert liefern. Die Berechnung ist in diesem Fall allerdings exponentiell in der Zahl der betrachteten Ungleichungen, die wiederum linear von der Zahl der Ereignisse (nicht Fälle!) im Log abhängt. Eine zweite Methode der Autoren versucht nur solche Stellen zu generieren, die das Verhalten des Petrinetzes in einer Weise beeinflussen, dass vom Prozesslog abweichendes Verhalten unterbunden wird. Durch eine entsprechende alternative Regionsdefinition (*separating region*) wird auch das Ungleichungssystem angepasst, welches laut den Autoren auch von SIMPLEX-Varianten gelöst werden kann. Wert et al.[47] verwenden mit `Parikh` einen anderen Ansatz von Regionen und damit Stellen, indem sie auf Kausalitätsrelationen zurückgreifen, wie sie in abstraktionsbasierten Rekonstrukti-

[44] Vgl. Bratosin et al. 2010.
[45] Vgl. Badouel und Darondeau 1996.
[46] Vgl. Bergenthum et al. 2007; Bergenthum et al. 2008.
[47] Vgl. Werf et al. 2008.

onsverfahren genutzt werden (vgl. Aalst et al.[48] und Nachfolger). Die Berechnung von Regionen erfolgt anhand der Lösung eines Optimierungsproblems, welches entsprechend formuliert wird. Für Details zu SBR-Algorithmen wird auf den Vergleich konkreter Verfahren von Lorenz et al.[49] verwiesen.

SBR-Ansätze (insbesondere der `Parikh`-Ansatz von Werf et al.[50]) erzielen ebenfalls sehr gute Werte was Deckung betrifft, weisen jedoch bzgl. Affinität deutliche Schwächen auf. Der schwerwiegendste Nachteil dieser Verfahren ist jedoch, dass u.U. Modelle generiert werden, die nicht zusammenhängend sind (also isolierte Knoten enthalten) und nicht klar ist, wie diese Problematik umgangen werden kann[51].

Zustandsbasierten Regionsalgorithmen (ZBR) liegt die Idee zugrunde, eine zustandsbasierte Systemrepräsentation in Form eines Transitionssystems in eine äquivalente ereignisbasierte Form zu überführen. Dabei wird ein Prozesslog unter Ausnutzung von Methoden zur Synthese von Transitionssystemen (siehe Abschnitt 3.3) zunächst als TS dargestellt. Regionen werden als Mengen von Zuständen definiert, die bzgl. der Ereignisse innerhalb des TS bestimmte Eigenschaften garantieren, die eine spätere Umformung in ein Petrinetz erlauben. Äquivalenz wird in konkreten Ansätzen unterschiedlich definiert und umfasst sowohl strukturelle (ausgedrückt mithilfe von Morphismen) als auch verhaltensbasierte Eigenschaften wie Bisimulation[52]. Verfahren dieser Klasse können nachweislich Petrinetze mit hoher Präzision rekonstruieren und sind für diese Arbeit daher besonders interessant. Eine Übersicht bestehender Verfahren und ihrer Entwicklung findet sich in Abschnitt 4.2.

ZBR-Verfahren haben ihren Ursprung in der *Synthese* von Petrinetzen, weshalb sie hinsichtlich der Kriterien **(KA1-4)** hervorragend abschneiden und folglich naheliegende Kandidaten für die Rekonstruktion präziser Modelle sind. Die Existenz von Garantien für die Ableitung eines Petrinetzes auf Basis eines TS welches exakt dasselbe Verhalten aufweist, reduziert die Beurteilung ihrer Eignung für die Rekonstruktion von präzisen Modellen auf die Frage, ob ein TS mit ausreichender Exaktheit aus einem Prozesslog generiert werden kann.

Obwohl solche Verfahren existieren (siehe Abschnitt 3.3), besteht die wesentliche Einschränkung von ZBR-Ansätzen in der sehr restriktiven Annahme, dass die „Sprache“ des Prozesslogs (Wörter entsprechen Aktivitätssequenzen) *präfixabgeschlossen* sein muss. In diesem Fall ist jedes Präfix eines Wortes selbst ein Wort der Sprache. Reale Prozesslogs erfüllen diese Bedingung typischerweise nicht, was die Anwendbarkeit von ZBR-Ansätzen für die Ableitung präziser Modelle schmälert.

[48] Vgl. Aalst, Weijters et al. 2004.
[49] Vgl. Lorenz et al. 2007.
[50] Vgl. Werf et al. 2008.
[51] Vgl. Carmona, Cortadella und Kishinevsky 2010.
[52] Vgl. Milner 1989.

3.5 Alternative Verfahren

Für die Rekonstruktion wohlgeformter Prozessmodelle wählen Herbst et al.[53] eine zweistufige Vorgehensweise, die in einem ersten Schritt zunächst Abhängigkeiten zwischen Ereignissen in einem sog. *stochastic dependency graph* (SDG) festhält und diesen anschließend in ein ADONIS-Modell überführt. Die Erzeugung des SDG orientiert sich an Methoden für Maschinelles Lernen (Hidden Markov Model); ADONIS ist eine Workflow-Definitionssprache, die unterschiedliche Knoten für das Auftreten von Ereignissen und Verzweigungen (UND/XOR) verwendet. Die Stärke dieses Ansatzes liegt in der Fähigkeit, mit mehrfach auftretenden Ereignissen innerhalb von einzelnen Ausführungen zurecht zu kommen. Erweiterungen des initialen Ansatzes erlauben außerdem die Erkennung von Nebenläufigkeit[54].

Schimm[55] schlägt ein iteratives Clusteringverfahren zur Erstellung eines blockstrukturierten Modells vor. Dabei werden Ausführungspfade mit derselben Menge an Ereignissen zunächst in Cluster zusammengeführt, sofern sie dieselbe Charakteristik bzgl. der *happened-before*-Relation aufweisen. Diese Vorgehensweise erinnert stark an das Verfahren von Mannila et el.[56] und der Charakterisierung von Abhängigkeiten anhand von Halbordnungen. Cluster werden solange vereint bis keine Kombination mehr möglich ist. Anschließend wird pro Cluster ein Block erzeugt und diese Blöcke mithilfe von Operatoren für parallele oder alternative Ausführung verbunden. Ähnlichkeit mit dem Ansatz von Schimm hat das *Multi-Phase-Process Mining*[57], welches zunächst für einzelne Ausführungen im Log sog. Instanzgraphen erzeugt und diese dann auf eine Weise kombiniert, die eine direkte Ableitung von EPC-Modellen[58] erlaubt.

Abgestimmt auf die von Aalst et al.[59] zur Modellierung rekonstruierten Prozessverhaltens vorgeschlagenen Kausalnetze (*C-nets*), nutzen Solè und Carmona[60] einen Divide-and-Conquer-Ansatz, der auf der Partitionierung des Prozesslogs basiert. Das Clustering wird so gesteuert, dass C-Net Fragmente, die jeweils für ein Cluster stehen, vereinigt werden können, sodass dabei möglichst wenig generalisiert wird. Der Ansatz nutzt Konzepte einer früheren Arbeit der Autoren, die das Rekonstruktionsproblem auf ein logisches Entscheidungsproblem (*satisfiability modulo theories*) zurückführt[61]. Insgesamt garantiert der Ansatz vollständige Deckung und minimale Generalisierung. Weiters et al.[62] verwenden einen heuristischen Ansatz (vgl. Weijters et al.[63]) für die Rekonstruktion von Kausalnetzen, der sich speziell für unstrukturierte Prozesse eignet, den Methoden von Solé und Carmona jedoch hinsichtlich Deckung unterlegen ist.

[53] Vgl. Herbst 1999b; Herbst 1999a; Herbst et al. 1998.
[54] Vgl. Herbst 2000a; Herbst 2000b; Herbst et al. 2004.
[55] Vgl. Schimm 2003; Schimm 2004.
[56] Vgl. Mannila et al. 2000.
[57] Vgl. Dongen und Aalst 2004; Dongen und Aalst 2005b.
[58] Vgl. Keller et al. 1992.
[59] Vgl. Aalst et al. 2011.
[60] Vgl. Solé et al. 2012a.
[61] Vgl. Solé et al. 2012b.
[62] Vgl. Weijters und Ribeiro 2011.
[63] Vgl. Weijters und Aalst 2003.

	Deckung		Affinität		Schlichtheit
	synthetisch	real	synthetisch	real	
Abstraktionsbasierte Verfahren	– –	– –	–	– –	+
Heuristische Verfahren	– –	–	–	+	○
Genetische Verfahren	+ +	–	–	○	– –
Verfahren basierend auf Negativbeispielen	+ +	–	–	– –	+
SBR Verfahren	+ +	+ +	–	– –	–
ZBR Verfahren	+ +	?	+	?	+

Tabelle 3.1.: Vergleich existierender Rekonstruktionsverfahren anhand der Präzisions-Anforderungen **(KA1-4)**.

3.6 Eignung bestehender Verfahren

Auf Basis bisheriger Arbeiten zum Vergleich bestehender Rekonstruktionsverfahren[64], bietet Tab. 3.1 einen Überblick über verfügbare petrinetzbasierte Rekonstruktionsansätze und ihre Eignung für die Erstellung von Modellen, welche die Anforderungen **(KA1-4)** erfüllen. Dabei wird zwischen der Anwendung der Ansätze auf synthetische oder reale Prozesslogs unterschieden. Die Gegenüberstellung von in Klassen zusammengefassten Rekonstruktionsansätzen abstrahiert bewusst von konkreten Konzepten und Details. Diese können vorherigen Abschnitten entnommen werden.

Wie bereits erwähnt, sind ZBR-Verfahren aufgrund ihrer garantierten Präzision hinsichtlich des Deckungsgrades besonders geeignet für die sicherheitsorientierte Rekonstruktion. Die grundlegende Idee des in dieser Arbeit vorgestellten Ansatzes zur präzisen Kontrollfluss-Rekonstruktion ist, ZBR-Verfahren dahingehend zu erweitern, dass die Annahme der Präfixabgeschlossenheit aufgehoben werden kann. Dies wird durch eine strukturelle Erweiterung von Transitionssystemen erreicht (Details siehe Abschnitt 4).

[64] Vgl. Lorenz et al. 2007; Werf et al. 2008; Tiwari et al. 2008; Weerdt et al. 2012.

	Algorithmisch	Maschinelles Lernen	Data Mining	Genetisch
Graphen	[Agrawal et al. (1998)] [Golani et al. (2003)] [Pinter et al. (2004)] [Folino et al. (2009)]	[Mannila et al. (2000)] [Silva et al. (2005)]	[Hwang et al. (2002)] [Greco et al. (2004)] [Ferreira et al. (2009)] [Greco et al. (2005)]	
Zustandsautomaten	[Biermann et al. (1972)] [Cook et al. (1995)] [Cook et al. (1998a)] [Cook et al. (1998b)] [Cook et al. (2004)] [Datta (1998)] [Folino et al. (2009)]	[Cook et al. (1995)] [Cook et al. (1998a)]		
Petrinetze	[Aalst et al. (2004)] [Alves de Medeiros et al. (2004b)] [Alves de Medeiros et al. (2004a)] [Li et al. (2007)] [Wen et al. (2007)] [Wen et al. (2007)] [Wen et al. (2009)] [Weijters et al. (2003)] [Weijters et al. (2006)] [Carmona et al. (2010)] [Carmona et al. (2013)] [Solé et al. (2012b)] [Solé et al. (2012a)] [Weijters et al. (2011)]	[Goedertier et al. (2007)] [Goedertier et al. (2009)] [Broucke et al. (2012)]	[Goedertier et al. (2007)] [Goedertier et al. (2009)]	[Alves de Medeiros et al. (2004c)] [Aalst et al. (2005)] [Alves de Medeiros (2006)] [Alves de Medeiros et al. (2007)] [Bratosin et al. (2010)]
Andere	[Dongen et al. (2004)] [Dongen et al. (2005b)]	[Herbst (1999b)] [Herbst (1999a)] [Herbst et al. (1998)] [Herbst (2000a)] [Herbst (2000b)] [Herbst et al. (2004)]	[Schimm (2003)] [Schimm (2004)]	

Tabelle 3.2.: Übersicht bestehender Rekonstruktionsverfahren gruppiert nach methodischer Vorgehensweise.

Kapitel 4

Präzise Kontrollfluss-Rekonstruktion

Nach Abschnitt 3.6 eignen sich zustandsbasierte Regionsalgorithmen (ZBR) für die Rekonstruktion von präzisen Modellen, die den Anforderungen **(KA1-4)** genügen. Das trifft insbesondere auf den GENET-Ansatz von Carmona et al.[1] zu, der nicht nur formale Garantien für vollständige Deckung bietet, sondern durch die Möglichkeit nicht nur sichere, sondern k-beschränkte Petrinetze generieren zu können, auch hinsichtlich Anforderung **(KA4)** einen Mehrwert liefert. Dabei wird das Prozessverhalten gemäß Log in Form eines Transitionssystems beschrieben und dieses Transitionssystem anschließend in ein Petrinetz transformiert. Das Verfahren garantiert Verhaltensäquivalenz zwischen dem extrahierten Transitionssystem und dem Erreichbarkeitsgraphen des resultierenden Petrinetzes.

Die Verwendung von Transitionssystemen für die Beschreibung der Logsprache (Ausführungspfade entsprechen Wörtern der Sprache) führt allerdings dazu, dass das nur im präfixabgeschlossenen Fall eine Übereinstimmung der Sprache des Logs und des rekonstruierten Petrinetzes zustande kommt. Sprachen von Transitionssystemen sind stets präfixabgeschlossen, weshalb sie nicht dafür geeignet sind, das Verhalten realer Prozesslogs zu beschreiben, die diese Eigenschaft typischerweise nicht erfüllen. Damit wird die Anwendbarkeit des Ansatzes in realen Umgebungen erheblich eingeschränkt. Dieses Kapitel zeigt, wie GENET erweitert werden kann, um die in Abschnitt 3.1 aufgestellten Anforderungen an Deckung und Affinität zu erfüllen.

Nach der Einführung adäquater Kriterien für die Beurteilung von Deckung und Affinität im Kontext nicht notwendigerweise präfixabgeschlossener Logsprachen (im Folgenden NPC-Sprachen), beschreibt dieses Kapitel zunächst anhand eines an die Praxis angelehnten Beispiels die Problematik existierender Syntheseansätze hinsichtlich der Qualität rekonstruierter Modelle (Abschnitt 4.1). Um eine Grundlage für das Verständnis folgender Abschnitte zu schaffen, folgt mit Abschnitt 4.2 ein Exkurs zur Entwicklung, dem Zusammenhang und aktuellen Stand von ZBR-Mechanismen. Anschließend wird die Arbeitsweise des GENET-Ansatzes von Carmona et al. erläutert (Abschnitt 4.3) und schließlich in Abschnitt 4.5 ein darauf basierendes Verfahren zur Kontrollfluss-Rekonstruktion

[1] Vgl. Carmona, Cortadella und Kishinevsky 2010.

präsentiert, welches hinsichtlich der Anforderungen in Abschnitt 3.1 präzise Modelle bereitstellen kann (GENET$^+$). Dafür wird für die Sprachdefinition eines Prozesslogs kein Transitionssystem verwendet. Stattdessen wird auf Zustandsautomaten zurückgegriffen, die NPC-Sprachen mithilfe von expliziten Endzuständen beschreiben können. Für Petrinetze wird eine alternative Sprachdefinition verwendet, die sich nicht wie üblich auf allgemeine Schaltsequenzen stützt, sondern nur Wörter beinhaltet, die durch abgeschlossene Schaltsequenzen erzeugt werden können (vollständige Sequenzsemantik).

Die Betrachtung von NPC-Sprachen erfordert eine neue Interpretation von Methoden zur Beurteilung der Qualität rekonstruierter Modelle, die typischerweise auf PC-Sprachen (präfixgeschlossene Sprachen) ausgerichtet sind. Generell gründet die Messung der Übereinstimmung eines rekonstruierten Modells und eines Prozesslogs auf der Betrachtung *konformer* Schaltsequenzen mit direktem Bezug zu einem Ablauf im Log und *abweichenden* Schaltsequenzen unter Ausnutzung konkreter Metriken für Deckung und Affinität. Existierende Affinitätsmetriken betrachten neben konformen Schaltsequenzen auch die Menge derer Präfixe als korrekt (implizite Annahme einer PC-Sprache). Sind nicht alle Präfixe Wörter der Logsprache, sind diese Verfahren nicht geeignet, die Affinität eines Modells zu beurteilen. Daher wird hier eine einfache Metrik verwendet, die die Zahl konformer Schaltsequenzen zur Anzahl abweichender Schaltsequenzen ins Verhältnis setzt.

Definition 4.1 (VSS-Affinität). *Gegeben sei ein Prozesslog* $\mathfrak{L} = \{\sigma_1, \ldots, \sigma_n\}$ *und* N *ein Petrinetz mit stillen Transitionen. Seien weiterhin* $deviating(\mathfrak{L}, N) := \{\gamma \in \Gamma^\bullet_N|_R \mid \nexists \sigma \in \mathfrak{L} : \sigma = \gamma\}$ *die abweichenden Schaltsequenzen. Die Affinität von* N *wird definiert als:*

$$\mathcal{A}(\mathfrak{L}, N) = 1 - \frac{|deviating(\mathfrak{L}, N)|}{|\mathcal{L}^\bullet(N)|} \qquad \dashv$$

VSS-Affinität operiert ausschließlich auf abgeschlossenen Schaltsequenzen ($\Gamma^\bullet_N$). Je höher die Affinität des Modells, desto geringer ist der Anteil abweichender abgeschlossener Schaltsequenzen. Hinsichtlich der Deckung arbeiten gängige Verfahren auf Basis von Alignments oder markenbasiert (siehe Abschnitt 3.1). Bei einer sicherheitsorientierten Betrachtung eines rekonstruierten Modells steht die prinzipielle Abspielbarkeit von Abläufen des Prozesslogs im Modell im Vordergrund. Ob dabei Marken im Netz zurückbleiben oder dieses „geleert“ wird, spielt eine untergeordnete Rolle. Hier wird deshalb in Anlehnung an die *Parsing Measure* von Weijters et al.[2] eine Deckungsmetrik verwendet, die den Anteil an Abläufen im Prozesslog angibt, für die eine entsprechende abgeschlossene Schaltsequenz im Modell angegeben werden kann.

Definition 4.2 (VSS-Deckung). *Gegeben sei ein Prozesslog* $\mathfrak{L} = \{\sigma_1, \ldots, \sigma_n\}$ *und* N *ein Petrinetz mit stillen Transitionen. Seien weiterhin* $conform(\mathfrak{L}, N) := \{\gamma \in \Gamma^\bullet_N|_R \mid \exists \sigma \in \mathfrak{L} : \sigma = \gamma\}$ *die konformen Schaltsequenzen. Die Deckung von* N *wird definiert als:*

$$\mathcal{D}(\mathfrak{L}, N) = \frac{|conform(\mathfrak{L}, N)|}{|\mathcal{L}(\mathfrak{L})|} \qquad \dashv$$

[2] Vgl. Weijters, Aalst und Alves de Medeiros 2006.

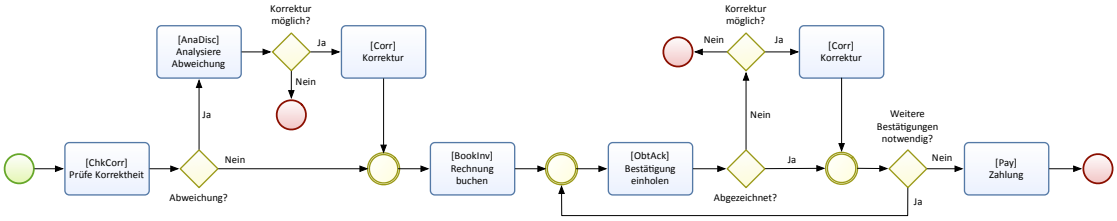


Abbildung 4.1.: Rechnungsprüfungsprozess.

Ablauf	**Aktivitätssequenz**
Ablauf 1	[ChckCorr] → [AnaDisc]
Ablauf 2	[ChckCorr] → [AnaDisc] → [Corr] → [BookInv] → [ObtAck]
Ablauf 3	[ChckCorr] → [BookInv] → [ObtAck] → [Pay]
Ablauf 4	[ChckCorr] → [BookInv] → [ObtAck] → [Corr] → [ObtAck]
Ablauf 5	[ChckCorr] → [BookInv] → [ObtAck] → [Corr] → [ObtAck] → [Corr] → [ObtAck]

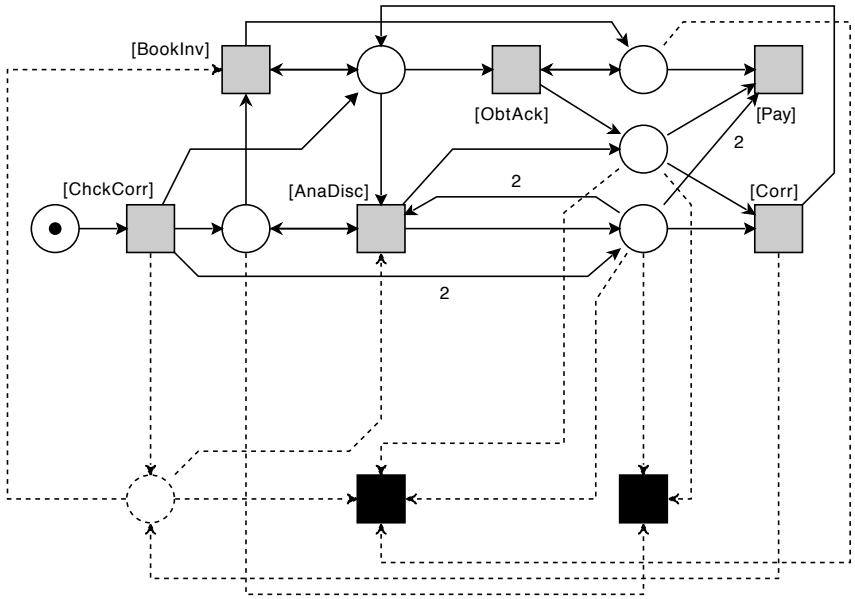
Tabelle 4.1.: Prozesslog $\mathfrak{L}_1$ für den Rechnungsprüfungsprozess.

4.1 Einführendes Beispiel

Zur Veranschaulichung der Beschränkung bestehender ZBR-Verfahren wird eine vereinfachte Version eines Rechnungsprüfungsprozesses aus der Praxis verwendet (siehe Abb. 4.1). Eintreffende Papierrechnungen werden zunächst auf Korrektheit geprüft. Im Fall von Preis- oder Mengenabweichungen wird die Einkaufsabteilung um Klärung des Sachverhalts gebeten. Falls Unklarheiten beseitigt und entsprechende Korrekturmaßnahmen durchgeführt werden können, wird die Rechnung verbucht. Sollte eine Korrektur nicht möglich sein, wird die Rechnung abgelehnt und der Prozess frühzeitig abgebrochen. Bevor die Rechnung eine Zahlungsfreigabe erhält, werden nach der Buchung zunächst erforderliche Bestätigungen von Warenempfängern und verantwortlichen Kostenstellenleitern eingeholt. Werden einzelne Positionen nicht bestätigt (z.B. aufgrund fehlender oder beschädigter Ware), kann eine Korrektur (z.B. Erstellung von Gutschriften, Preisreduzierung etc.) oder ein Abbruch des Gesamtprozesses folgen. Die Schleife innerhalb des Prozesses erlaubt mehrfache Korrekturmaßnahmen bis alle Bestätigungen eingegangen sind.

Unter Verwendung von Abkürzungen für Prozessaktivitäten enthält Tab. 4.1 einen beispielhaften Prozesslog dieses Prozesses. Die Anwendung von `genet` auf $\mathfrak{L}_1$ resultiert im Petrinetz $\mathfrak{P}_1$ aus Abb. 4.2(a), welches zwar alle Abläufe des Prozesslogs generieren kann, andererseits aber auch abweichende Schaltsequenzen enthält (grau hinterlegte Zeilen in Tab. 4.2(b)).

Unter Verwendung herkömmlicher Qualitätsmetriken hätte $\mathfrak{P}_1$ vollständige Deckung und völlige Affinität. Diese Aussage ist jedoch irreführend, da im Prozesslog *Ablauf 1* ein Präfix von *Ablauf 2*, sowie *Ablauf 4* ein Präfix von *Ablauf 5* ist, diese korrekten Präfixe allerdings nicht von Präfixen unterschieden werden, die nicht mit Aktivitätssequenzen im Log in Verbindung gebracht werden können. Dieses Beispiel veranschaulicht die zuvor



(a) Rekonstruiertes Petrinetz $\mathfrak{P}_1$ mithilfe von GENET (ausschließlich massive Kanten) und $\mathfrak{P}_2$ mithilfe von GENET$^+$ (alle Kanten unter Hinzunahme weiterer Stellen/Transitionen) auf Basis von $\mathfrak{L}_1$.

	Schaltsequenz
○	[ChckCorr]
○	[ChckCorr] → [BookInv] → [ObtAck]
•	[ChckCorr] → [BookInv] → [ObtAck] → [Corr] → [ObtAck] → [Corr] → [ObtAck]
•	[ChckCorr] → [BookInv] → [ObtAck] → [Pay]
○	[ChckCorr] → [BookInv] → [ObtAck] → [Corr] → [ObtAck] → [Corr]
○	[ChckCorr] → [AnaDisc] → [Corr]
○	[ChckCorr] → [AnaDisc]
•	[ChckCorr] → [AnaDisc] → [Corr] → [BookInv] → [ObtAck]
○	[ChckCorr] → [BookInv]
○	[ChckCorr] → [BookInv] → [ObtAck] → [Corr] → [ObtAck]
○	[ChckCorr] → [AnaDisc] → [Corr] → [BookInv]
○	[ChckCorr] → [BookInv] → [ObtAck] → [Corr]

(b) Schaltsequenzen des Petrinetzes $\mathfrak{P}_1$.

Abbildung 4.2.: Rekonstruktion von GENET und GENET$^+$ im Vergleich.

getroffene Aussage, dass existierende Affinitätsmetriken bei der Betrachtung von NPC-Sprachen nicht hilfreich sind.

Bezogen auf abgeschlossene Schaltsequenzen (angedeutet mit $\bullet$) hat $\mathfrak{P}_1$ völlige Affinität, aber keine vollständige Deckung. Zwar existiert für jede abgeschlossene Schaltsequenz ein entsprechendes Pendant im Prozesslog, andererseits werden nicht alle Aktivitätssequenzen im Log abgedeckt. Das Modell $\mathfrak{P}_2$ hingegen, welches auf Basis der hier vorgestellten Theorie gewonnen wurde, hat nicht nur völlige Affinität, sondern zusätzlich auch vollständige Deckung. Im Vergleich zu $\mathfrak{P}_1$ beinhaltet es zwei stille Transitionen (schwarze Rechtecke) und eine zusätzliche Stelle, die es dem Netz erlauben, zusätzliche abgeschlossene Schaltsequenzen zu erzeugen, die eine Verbindung zum Prozesslog aufweisen. Die Menge der abgeschlossenen Schaltsequenzen entspricht exakt den Aktivitätssequenzen des Prozesslogs.

4.2 Entwicklung zustandsbasierter Regionsalgorithmen

Die grundlegende Theorie für ZBR-Algorithmen wurde von Ehrenfeucht und Rozenberg[3] begründet, die erstmals das Konzept Region innerhalb sog. *labeled partial 2-structures* (im Folgenden *2-Struktur* oder 2S) definierten. Ihre Arbeit wurde in Folge in vielfältiger Weise erweitert, um unterschiedliche Gegebenheiten i.S. von Metamodellen für die Definition von ereignis- bzw. zustandsbasiertem Verhalten unterstützen zu können.

Eine *2-Struktur* ist eine graphenähnliche Struktur, die aus einer Grundmenge D und einer Äquivalenzrelation $R \subseteq D \times D$ besteht. Für $S = (D, R)$ wird $dom(S) := D$ als alternative Schreibweise gewählt. Eine *initialisierte 2-Struktur* ist ein Paar (S, v), wobei S eine *2-Struktur* ist und $v \in dom(S)$ den initialen Zustand der *2-Struktur* definiert. Anschaulich kann eine *2-Struktur* als eine Menge von gerichteten Graphen dargestellt werden, wobei sich pro Äquivalenzklasse $K \subseteq D \times D$ die Knotenmenge zu D und die Kantenmenge zu K ergibt. Alternativ kann eine *2-Struktur* auch als einziger gerichteter Graph mit markierten Kanten visualisiert werden. Die Markierung ist prinzipiell willkürlich, muss den Äquivalenzklassen jedoch als Unterscheidungsmerkmal dienen. D.h. alle Elemente innerhalb einer Äquivalenzklasse tragen dieselbe Markierung, diese unterscheidet sich jedoch von den Markierungen aller Elemente in anderen Äquivalenzklassen. Abb. 4.3 zeigt die Visualisierung der beispielhaften *2-Struktur* $(D, \mathscr{P})$, wobei $D = \{1, 2, 3, 4\}$ und $\mathscr{P} = \{P_1, P_2, P_3, P_4\}$ die von R induzierten Äquivalenzklassen sind:

$P_1 = \{(1, 2), (3, 2), (3, 4), (4, 3)\}$,
$P_2 = \{(1, 3), (2, 1), (2, 4)\}$,
$P_3 = \{(1, 4), (2, 3), (4, 2)\}$,
$P_4 = \{(3, 1), (4, 1)\}$

Ein Merkmal von *2-Strukturen* ist ihre Vollständigkeit, d.h. jedes Knotenpaar ist durch eine Kante verbunden. Dieses Konzept wird durch eine Erweiterung zu *partiellen markierten 2-Strukturen* (PM2S), welche als 5-Tupel $(D, F, R, \Delta, \delta)$ mit den zusätzlichen Komponenten $F \subseteq D \times D, \Delta$ und $\delta : F \mapsto \Delta$ gekennzeichnet sind, generalisiert. Im

[3] Vgl. Ehrenfeucht et al. 1989a.

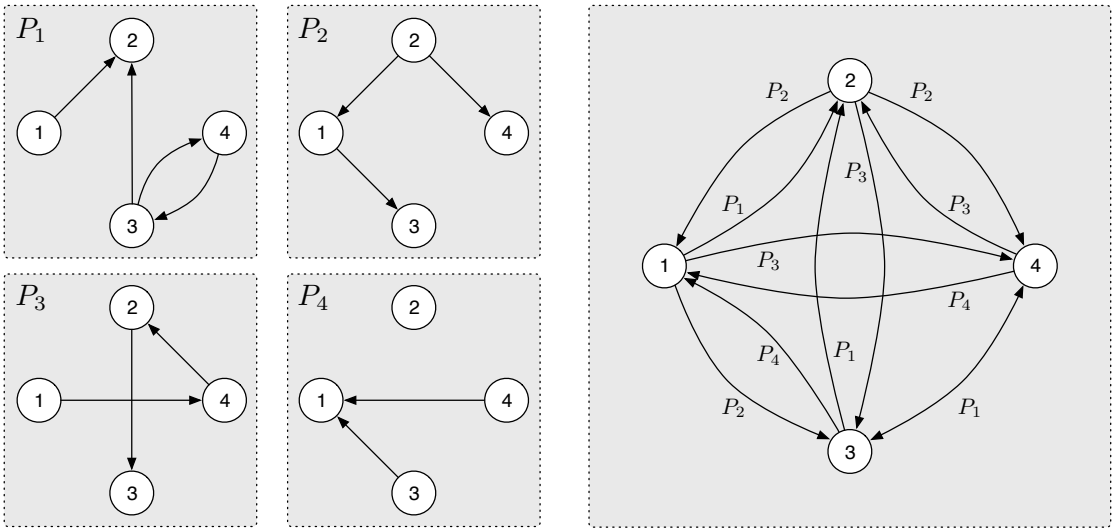


Abbildung 4.3.: Visualisierung einer 2S.

Unterschied zum Ansatz von Ehrenfeucht et al.[4] ist die Äquivalenzrelation R nun auf F definiert, um Unvollständigkeit bzgl. der Verbundenheit von Knoten in D zu ermöglichen. Die Markierungsfunktion δ ordnet jeder Kante einen Bezeichner aus Δ zu. Im Folgenden wird $F_m := \{(x, y) \in F \mid \delta((x, y)) = m\}$ als Abkürzung für alle Kanten mit Markierung m verwendet. Ausgehend von dieser Definition wird eine Region als Knotenmenge definiert welche hinsichtlich der Kantenbeschriftungen eine gewisse Homogenität aufweist.

Definition 4.3 (Region in partieller markierter 2-Struktur). *Sei* $S = (D, F, R, \Delta, \delta)$ *eine partielle markierte 2-Struktur (PM2S).* $Z \subseteq D$ *ist eine Region in* S *gdw.* $\forall (x, y), (u, v) \in F$, $(x, y)R(u, v)$ *impliziert, dass:*

(1) $x \in Z \wedge y \notin Z \implies u \in Z \wedge v \notin Z$, *und*

(2) $x \notin Z \wedge y \in Z \implies u \notin Z \wedge v \in Z$. ⊣

Konkret wird gefordert, dass alle Kanten einer Markierung $m \in \Delta$ entweder

- in die Region hineinführen (`entering`):
  $\forall (x, y) \in F_m : x \notin Z \wedge y \in Z$,
- aus der Region herausführen (`exiting`):
  $\forall (x, y) \in F_m : x \in Z \wedge y \notin Z$, oder
- weder hinein-, noch hinausführen (`no crossing`):
  $(\forall (x, y) \in F_m : x \notin Z \wedge y \notin Z) \vee (\forall (x, y) \in F_m : x \in Z \wedge y \in Z)$

Die Menge aller Regionen innerhalb eines 2S S wird als $\mathscr{R}_S$ bezeichnet und die Funktion $\mathscr{R}_S(d) := \{r \in \mathscr{R}_S \mid d \in r\}$ liefert alle Regionen, die einen Knoten $d \in D$ enthalten.

[4] Vgl. Ehrenfeucht et al. 1990.

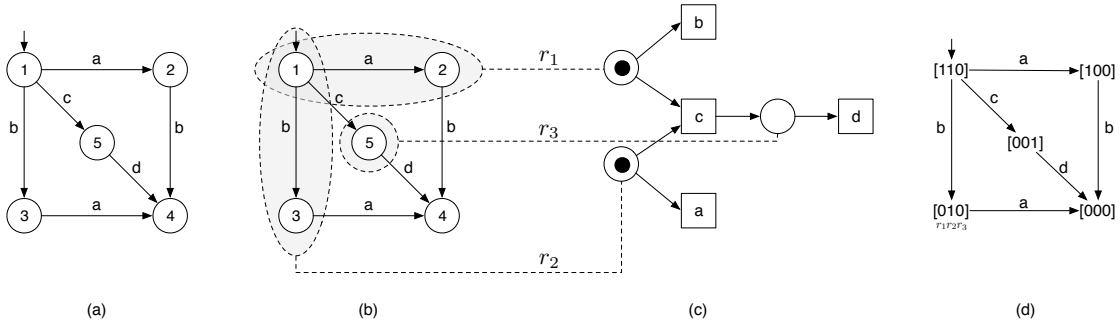


Abbildung 4.4.: Partielle *2-Struktur* (a), zugehörige Regionen (b) und resultierendes Petrinetz (c) mit zugehörigem Zustandsraum (d).

Ehrenfeucht et al.[5] stellen die Verbindung zwischen *2-Strukturen* zu Petrinetzen her, indem sie beweisen, dass sich der Zustandsraum einer speziellen Klasse von Petrinetzen (*elementary net systems*, kurz: ENS) in Form einer initialisierten *2-Struktur* darstellen lässt. Ein ENS ist ein Petrinetz gemäß Def. B.9 mit der Einschränkung, dass jede Stelle maximal eine Marke enthalten kann und jede Kante maximal eine Marke transportiert (Kantengewicht 1). Der zugrundeliegende Ansatz ist, zunächst alle Regionen des PM2S $S = (D, F, R, \Delta, \delta)$ zu berechnen und daraus einen bipartiten Graphen $(V_1, V_2, E_1 \cup E_2)$ zu konstruieren, wobei $V_1 = \{\mathscr{R}_S(d) \,|\, d \in D\}, V_2 = \Delta$ und sich die Kanten zu $E_1 := \{(u, v) \in V_1 \times V_2 \,|\, \exists x \in \mathscr{R}_S(u) \exists y \in D : (x, y) \in F \wedge \delta((x, y)) = v\}$ und $E_2 := \{(v, u) \in V_2 \times V_1 \,|\, \exists x \in \mathscr{R}_S(u) \exists y \in D : (x, y) \in F \wedge \delta((x, y)) = v\}$ ergeben.

Abb. 4.4 enthält ein Beispiel für den entsprechenden Synthesealgorithmus. Jede Region der *2-Struktur* entspricht einer Stelle im resultierenden Petrinetz, Markierungen entsprechen Transitionen. Für jede Markierung, die in eine Region hinein führt, existiert im Petrinetz eine Kante von der entsprechenden Transition zur betreffenden Stelle, umgekehrt für Markierungen, die aus Regionen herausführen. Der initiale Zustand des Petrinetzes wird gemäß dem initialen Zustand der *2-Struktur* gesetzt, i.e. jede Stelle deren zugehörige Region den initialen Zustand der *2-Struktur* enthält, wird mit einer Marke markiert. Diese Vorgehensweise garantiert, dass der Zustandsraum des resultierenden ENS isomorph zur Eingabe *2-Struktur* ist, dient weiteren Arbeiten als Grundlage. Die betreffende Klasse von *2-Strukturen* wird seit der weiteren Konkretisierung dieser Übereinstimmung von Nielsen et al.[6] als *elementary transition systems* (ETS) bezeichnet.

Die Komplexität der Synthese von ENS wurde von Hiraishi[7] untersucht. Nach der Definition hinreichender ENS-Bedingungen für die Isomorphie zu einem gegebenen ETS, konnte gezeigt werden, dass die Überprüfung dieser Eigenschaften NP-vollständig ist. Trotz der Bemerkung des Autoren, dass dies nicht zugleich bedeute, dass das Syntheseproblem als solches ebenfalls NP-vollständig sein müsse, gibt dieses Resultat doch einen Hinweis auf die zugrundeliegende Komplexität der Synthese, die auch in späteren Arbeiten als „schwer“ bezeichnet wird.

[5] Vgl. Ehrenfeucht et al. 1989b.
[6] Vgl. Nielsen et al. 1992.
[7] Vgl. Hiraishi 1994.

Ein grundlegendes Problem bisheriger Syntheseansätze ist die Größe des Netzes, welche im Allgemeinen exponentiell in der Größe des zugrundeliegenden ETS ist. Deshalb schlagen Desel und Reisig[8] ein Verfahren vor, welches nicht die Menge aller Regionen innerhalb eines ETS betrachtet, sondern nur eine hinreichende Untermenge *nichtredundanter* Regionen. Ein auf dieser Theorie basierender praktischer Algorithmus, der solche Regionsmengen in polynomieller Zeit berechnen kann, wurde von Badouel et al.[9] vorgeschlagen (Tool: `synet`[10]). Integriert werden hier außerdem Ergebnisse aus einer Arbeit von Bernardinello et al.[11], in der ein erweitertes Regionskonzept definiert wird (*generalized region*), welches die Synthese von beschränkten Petrinetzen ermöglicht. Interessanterweise konnte von Badouel et al.[12] gezeigt werden, dass im Gegensatz dazu die Synthese von ENS im Allgemeinen NP-vollständig ist (ein von Hiraishi[13] unabhängiges Resultat basierend auf einer 3-SAT Reduktion). Ein Spezialfall bildet eine Ausgangsbasis in Form eines ETS mit Baumstruktur.

Aufbauend auf den Resultaten von Nielsen et al.[14], generalisiert Mukund[15] das Konzept der Regionen auf Petrinetze gemäß Def. B.9. Dafür wird unter Berufung auf Winskel et al.[16] eine spezielle Klasse von Transitionssystemen betrachtet, sog. *PN-transition systems* (PNTS). Ein PNTS erfüllt bestimmte Kriterien hinsichtlich Regionen, die es ermöglichen, den Isomorphismus zwischen dem PNTS und dem Zustandsraum eines Petrinetzes auch unter Berücksichtigung von Kantengewichten zu erhalten.

Diese frühen, hauptsächlich theoretisch gelagerten Ergebnisse wurden von Cortadella et al.[17] erweitert, um nicht nur ETS, sondern allgemeine Transitionssysteme (TS) betrachten zu können (Tool: `petrify`[18]). Um zu garantieren, dass für die korrekte Synthese erforderliche regionsspezifische Eigenschaften innerhalb von TS garantiert werden können, wird mit *label splitting* eine „Korrektur"-Prozedur vorgeschlagen, die auf der Umbenennung von TS-Ereignissen (In ETS: Markierung) beruht. Zudem wird für das resultierende Petrinetz Minimalität bzgl. der Anzahl von Stellen garantiert, d.h. die Wegnahme einer beliebigen Stelle verändert das Verhalten des Netzes so, dass die Äquivalenz seines Zustandsraums mit dem Eingabe-TS nicht mehr gewährleistet ist. Die hierfür zugrundeliegende Idee wird auf die Beobachtung zurückgeführt, dass bestimmte Regionen nicht für die Erstellung eines adäquaten Petrinetzes benötigt werden und damit redundant sind. Wurde bisher die Übereinstimmung des Verhaltens eines Petrinetzes und eines Transitionssystems mithilfe von Isomorphie beschrieben, wird hier erstmals auf ein alternatives Konzept von Milner[19] in Form von Bisimilarität zurückgegriffen. Diese Relaxation vereinfacht die Synthese dahingehend, dass eine zuvor notwendige regionsspezifische Eigenschaft (zur Separierung von TS-Zuständen anhand von Regionen) nicht länger gefordert werden muss.

[8] Vgl. Desel et al. 1996.
[9] Vgl. Badouel et al. 1995; Badouel und Darondeau 1996.
[10] Vgl. Caillaud 2002.
[11] Vgl. Bernardinello et al. 1993.
[12] Vgl. Badouel et al. 1997.
[13] Vgl. Hiraishi 1994.
[14] Vgl. Nielsen et al. 1992.
[15] Vgl. Mukund 1992.
[16] Vgl. Winskel et al. 1995.
[17] Vgl. Cortadella et al. 1995; Cortadella et al. 1998.
[18] Vgl. Cortadella, Kishinevsky, A.Kondratyev et al. 1997.
[19] Vgl. Milner 1989.

Der Ansatz von Cortadella et al. wurde von Carmona et al.[20] weitergeführt und auf die Synthese von beschränkten Petrinetzen mit Kantengewichten erweitert (Tool: `genet`[21]). Beschränkte Petrinetze haben den Vorteil, dass dasselbe Verhalten in vielen Fällen deutlich kompakter als Netz repräsentiert werden kann.

Anwendung für Process Mining

Dass Algorithmen zur Petrinetzsynthese auch im Process Mining-Kontext sinnvoll einsetzbar sind, erkannten zunächst Kindler et al.[22]. Prozesslogs, die bei Process Mining als Eingabe dienen, enthalten allerdings lediglich sequentielle Abfolgen von Ereignissen, die während der Ausführung eines Prozesses aufgetreten sind, und keine Zustandsinformation. Eine Notwendigkeit für die Anwendung von ZBR-Algorithmen besteht also darin, zunächst eine zustandsorientierte Repräsentation des Prozesslogs in Form eines Transitionssystems zu gewinnen. Synthesealgorithmen arbeiten naturgemäß präzise, während Process Mining die eher gegenläufige Idee zugrunde liegt, vom Prozesslog zu abstrahieren und nur das gängigste Prozessverhalten darzustellen. Sprachäquivalenz ist hier eine eher unerwünschte Eigenschaft, da resultierende Petrinetze „zu nahe“ am observierten Prozessverhalten bleiben und durch selten auftretende Ausführungsmuster schnell komplex und damit unübersichtlich werden.

Dingen et al.[23] konzentrieren sich bei der Anwendung des Ansatzes von Desel und Reisig[24] auf die Extraktion eines Transitionssystems und schlagen dafür einen naiven Algorithmus vor, der zunächst für jeden unterschiedlichen Ausführungspfad ein streng sequentielles Transitionssystem erzeugt. Diese Transitionssysteme werden in einem zweiten Schritt mithilfe eines künstlichen gemeinsamen Initialzustandes vereinigt. Ohne eine Abstraktion vorzunehmen, besteht die Aussage dieser Arbeit also in der prinzipiellen Anwendbarkeit regionsbasierter Ansätze auf Prozesslogs.

Eine flexiblere Möglichkeit zur Extraktion eines Transitionssystems aus einem Prozesslog, die auch Raum für Abstraktion bietet, wird von Aalst et al.[25] vorgestellt. Van der Aalst et al. erlauben die Definition eines Zustandes anhand bisher observierter Ereignisse innerhalb eines Traces (Präfix), noch folgender Ereignisse (Postfix) oder einer Kombination aus Präfix und Postfix. Abstraktionen können dann bspw. eingeführt werden, indem die maximal betrachtete Anzahl von Ereignissen im Präfix/Postfix beschränkt wird oder bestimmte Ereignisse gefiltert werden. Für die Synthese selbst stützt sich dieser Ansatz auf `petrify`.

Mit einem Fokus auf Möglichkeiten auch während der Synthese von beobachtetem Verhalten zu abstrahieren und auf Basis der theoretischen Grundlagen von `genet`, schlagen Carmona et al.[26] vor, eine wichtige regionsspezifische Eigenschaft von Transitionssystemen, die für die Garantie der Bisimilarität benötigt wird (*excitation closure*), nicht

[20] Vgl. Carmona, Cortadella, Kishinevsky et al. 2008.
[21] Vgl. Carmona, Cortadella und Kishinevsky 2009.
[22] Vgl. Kindler et al. 2006.
[23] Vgl. Dongen, Busi et al. 2007.
[24] Vgl. Desel et al. 1996.
[25] Vgl. Aalst, Rubin et al. 2010.
[26] Vgl. Carmona, Cortadella und Kishinevsky 2008.

länger zu erzwingen. Diese Maßnahme hat neben der Tatsache, dass das resultierende Petrinetz neben Aktivitätssequenzen des Prozesslogs nun möglicherweise auch Sequenzen akzeptiert, die dort nicht enthalten sind, zur Folge, dass die Generalisierung des Prozessverhaltens gemäß Prozesslog kontrolliert stattfindet. Insbesondere konnte gezeigt werden, dass kein anderes Petrinetz existiert, was weniger zusätzliche Sequenzen enthält. Die Autoren erweitern diese Grundprinzipien ins einer weiteren Arbeit[27] und stellen die Anwendungsbereiche Synthese und Mining gegenüber. Für die Anwendung von `genet` für Process Mining stellen Solé und Carmona schließlich das Tool `rbminer`[28] vor, welches zudem eine Minimierung des Eingabe-TS vornimmt, um die Berechnungskomplexität für große Logs zu reduzieren.

Der Zusammenhang zwischen der in diesem Abschnitt vorgestellten Syntheseansätze auf Basis zustandsbasierter Modelle (TS) untereinander und deren Bezug zu Arbeiten hinsichtlich ihrer Anwendung für Process Mining werden in Abb. 4.5 dargestellt. Arbeiten auf der Grundlagenebene legen das theoretische Fundament i.S. von *2-Strukturen* und Regionen, welches von Syntheseansätzen (Methodenbereich) genutzt wird. Arbeiten auf der Anwendungsebene übertragen diese Konzepte auf Process Mining.

[27] Vgl. Carmona, Cortadella und Kishinevsky 2010.
[28] Vgl. Solé et al. 2010b.

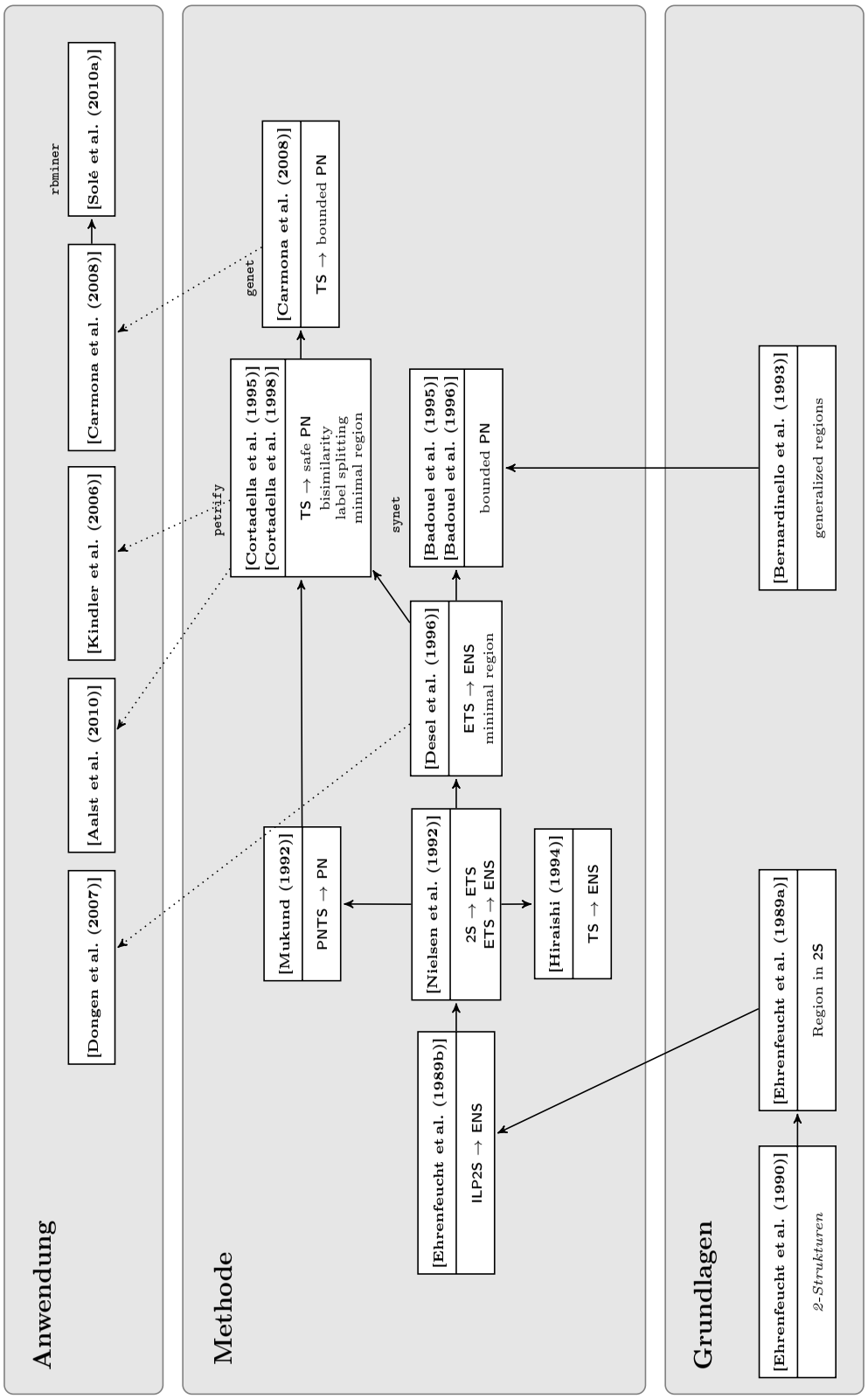


Abbildung 4.5.: Übersicht über Ansätze zur Synthese von Petrinetzen auf Basis zustandsbasierter Modelle.

4.3 Petrinetzsynthese auf Basis von Transitionssystemen

Dieser Abschnitt erläutert die Funktionsweise des GENET-Ansatzes von Carmona et al.[29], welcher die Grundlage für das im Rahmen dieser Arbeit entwickelte Verfahren zur Rekonstruktion präziser Prozessmodelle bildet. Zur Synthese von beschränkten Petrinetzen generalisieren die Autoren das Konzept von Regionen innerhalb von Transitionssystemen und führen dafür den Begriff *Gradient* ein. Bestand bisher eine Region aus einer Menge von TS-Zuständen, wird nun auf Multimengen zurückgegriffen.

Definition 4.4 (Gradient). *Sei (S, E, A, s_{in}) ein Transitionssystem und m eine Multimenge über S. Der Gradient einer Transition $(s, e, s') \in A$ bzgl. m ist definiert als $\Delta_m(s, e, s') = m(s') - m(s)$. Ein Ereignis $e \in E$ hat **konstanten Gradient**, wenn alle mit e ausgezeichneten Transitionen denselben Gradient aufweisen, i.e. $\forall a_1, a_2 \in A_e : \Delta_m(a_1) = \Delta_m(a_2)$.* ⊣

Eine Region besteht aus einer Multimenge von TS-Zuständen, innerhalb derer der Gradient aller Ereignisse konstant ist.

Definition 4.5 (Region). *Sei $(S, E = \{e_1, \ldots, e_n\}, A, s_{in})$ ein Transitionssystem und m eine Multimenge über S. Die Multimenge m ist eine Region gdw. jedes Ereignis konstanten Gradient in m hat, i.e. $\forall e \in E, \exists \tau \in \mathbb{Z}, \forall a \in A_e : \Delta_m(a) = \tau$.* ⊣

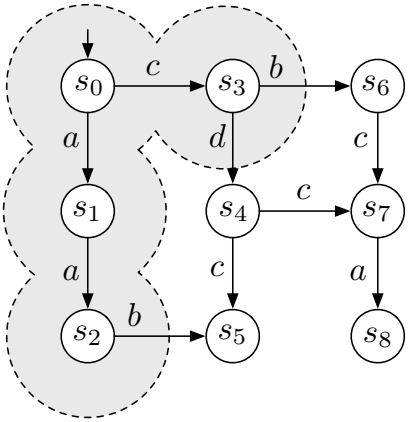
Der Gradient eines Ereignisses e innerhalb einer Region r wird mit $\Delta_r(e)$ bezeichnet. Der *Gradienten-Vektor* von r ist definiert als $\Delta_r := (\Delta_r(e_1), \ldots, \Delta_r(e_n))$. Ansätze zur Synthese von *sicheren* Petrinetzen verwenden Regionen, bei denen der Gradient von Transitionen auf das Intervall $[-1; 1]$ beschränkt ist. Insofern stellt die hier verwendete Regionsdefinition eine Verallgemeinerung dar.

Definition 4.6 (Minimalität und Trivialität von Regionen). *Sei $TS = (S, E, A, s_{in})$ ein Transitionssystem. Eine Region r bzgl. TS ist **trivial** wenn gilt $\exists \tau \in \mathbb{Z}, \forall s \in S : r(s) = \tau$. Triviale Regionen werden mithilfe der Multiplizität τ angegeben mit $\boldsymbol{0}, \boldsymbol{1}, \ldots, \boldsymbol{K}$. Existiert keine Unter-Multimenge für r, die selbst wiederum die Regionseigenschaft erfüllt, so ist r **minimal**, i.e. $\nexists r' \neq \boldsymbol{0} : r' \subset r$.* ⊣

Die Visualisierung in Abb. 4.6 (a) zeigt die 1-beschränkten Regionen eines TS, d.h. alle Regionen, die jeden Zustand des TS maximal einmal enthalten. Anhand der Zustände im grau hinterlegten Bereich, die Region r_1 bilden, lässt sich die durch Regionen implizierte Homogenität hinsichtlich Ereignissen intuitiv nachvollziehen. Analog zu den `exiting`, `entering` und `no crossing` Relationen, die in Abschnitt 4.2 auf partiellen 2S erläutert wurden, kreuzen alle Transition eines TS-Ereignisses die Grenzen der Region auf dieselbe Weise oder haben keine Berührung mit diesen Grenzen. Während Ereignis b stets mit einem Verlassen der Region assoziiert werden kann, führen Transitionen, die mit Ereignis a markiert sind, weder in die Region hinein, noch hinaus. Diese Eigenschaften müssen im allgemeinen Fall für k-beschränkte Regionen nicht gelten. Dabei können Regionen TS-Zustände mehrfach enthalten (Exponenten von Zuständen geben deren Multiplizität innerhalb von Regionen an). Obwohl bzgl. Ereignis a in Abb. 4.6 (b) sowohl Transitionen

[29] Vgl. Carmona, Cortadella und Kishinevsky 2010.

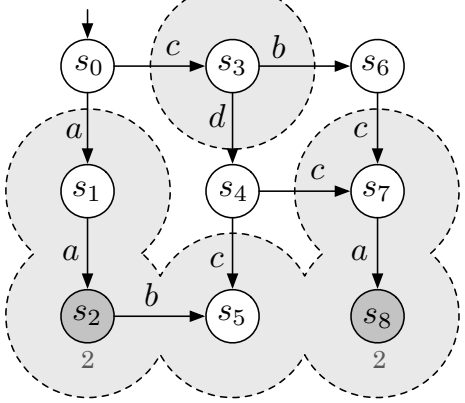
(a)



$r_1 = \{s_0, s_1, s_2, s_3\}$
$r_2 = \{s_4, s_5, s_6, s_7, s_8\}$

$ER(a) = \{s_0, s_1, s_7\}$
$ER(b) = \{s_2, s_3\}$
$ER(c) = \{s_0, s_4, s_6\}$
$ER(d) = \{s_3\}$

(b)



$r_1 = \{s_0, s_1.s_2, s_3\}$
$r_2 = \{s_4, s_5, s_6, s_7, s_8\}$
$r_3 = \{s_1, s_2^2, s_3, s_5, s_7, s_8^2\}$
$r_4 = \{s_0^2, s_1, s_3, s_4^2, s_5, s_6^2, s_7\}$
.....
$r_5 = \{s_0^2, s_1^2, s_2^2, s_3^2, s_4, s_5, s_6, s_7, s_8\}$
$r_6 = \{s_0, s_1, s_2, s_3, s_4^2, s_5^2, s_6^2, s_7^2, s_8^2\}$
$r_7 = \{s_0^2, s_1^2, s_2^2, s_3^2, s_4^2, s_5^2, s_6^2, s_7^2, s_8^2\}$

Abbildung 4.6.: TS mit zugehörigen 1-beschränkten Regionen in (a) und 2-beschränkten Regionen in (b).

existieren, die eine Regionsgrenze kreuzen (von s_0 nach s_1), haben alle mit a beschrifteten Transitionen Gradient 1, was die Regionseigenschaft sicherstellt. Während alle Regionen in Abb. 4.6 (a) minimal sind, enthält die Regionsmenge in Abb. 4.6 (b) nichtminimale Regionen (abgegrenzt unter gestrichelter Linie).

GENET erlaubt die Transformation eines TS gemäß Def. B.5 in ein k-beschränktes PN, dessen Erreichbarkeitsgraph verhaltensäquivalent zu TS ist und $\mathcal{L}(\mathsf{TS}) = \mathcal{L}^{\circ}(\mathsf{PN})$. Die Vorgehensweise umfasst drei Schritte:

1. Berechnung minimaler Regionen.
2. Sicherstellung erforderlicher Eigenschaften für Verhaltensäquivalenz.
3. Petrinetz-Konstruktion auf Basis minimaler Regionen.

Im ersten Schritt wird mit der Bestimmung minimaler Regionen eine Grundlage für die Rekonstruktion präziser Petrinetze gelegt. Abhängig von der Struktur minimaler Regionen kann die Verhaltensäquivalenz des Ausgangs-TS mit dem rekonstruierten PN garantiert werden. In Schritt 2 wird deshalb überprüft, ob genügend adäquate minimale Regionen vorhanden sind. Im negativen Fall, kann dies durch strukturelle Modifikation

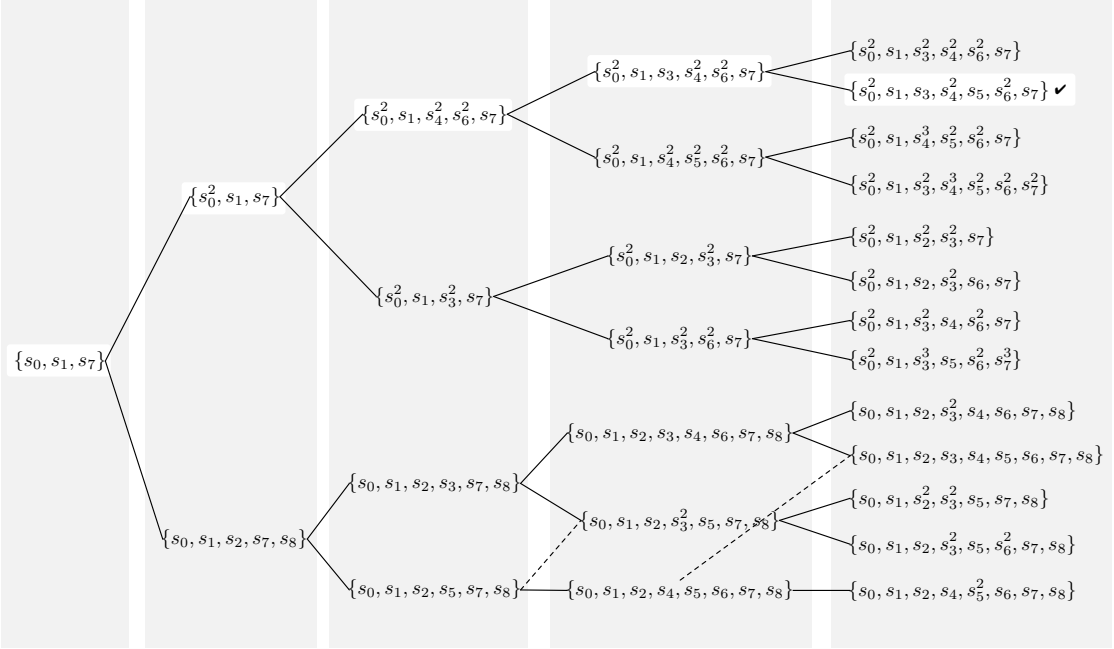


Abbildung 4.7.: Expansionsschritte ausgehend von $ER(a)$ auf Basis des TS in Abb. 4.6.

des TS sichergestellt werden (siehe dazu Ansätze in Abschnitt 4.4). Im letzten Schritt wird dann auf Basis einer Auswahl minimaler Regionen ein Petrinetz konstruiert.

Berechnung minimaler Regionen

Um alle minimalen Regionen innerhalb eines TS zu finden, werden zunächst die *Anreizbereiche* und *Schaltbereiche* von Ereignissen auf die Einhaltung der Regionseigenschaft überprüft. Diese initialen Multimengen werden anschließend schrittweise erweitert (expandiert) um neue Bereiche/Regionen zu finden. Dabei werden in jedem Schritt zusätzliche Zustände hinzugefügt, mit dem Ziel durch diese Erweiterung schließlich die Regionseigenschaft sicherzustellen.

Definition 4.7 (Anreiz- und Schaltbereiche). *Sei (S, E, A, s_{in}) ein Transitionssystem. Der Anreizbereich eines Ereignisses $e \in E$ ist die Menge der Zustände, die mittels e verlassen werden können, i.e.,*

$$ER(e) = \{s \in S \mid \exists s' \in S : (s, e, s') \in E\}.$$

Der Schaltbereich eines Ereignisses $e \in E$ ist die Menge der Zustände, die mittels e erreicht werden können, i.e.,

$$SR(e) = \{s \in S \mid \exists s' \in S : (s', e, s) \in E\}. \qquad \dashv$$

In Abb. 4.6 (a) ergeben sich die Anreiz- und Schaltbereiche von Ereignis a zu $ER(a) = \{s_0, s_1, s_7\}$ und $SR(a) = \{s_1, s_2, s_8\}$.

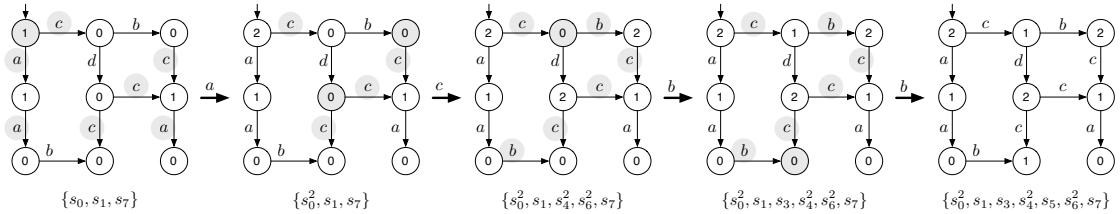


Abbildung 4.8.: Sukzessive Erweiterungen einer Multimenge zur Sicherstellung der Regionseigenschaft.

Abb. 4.7 zeigt einen Ausschnitt der Multimengen die ausgehend vom Anreizbereich des Ereignisses a bei dessen Erweiterung exploriert werden (im folgenden *ER*-Multimenge). Falls eine betrachtete Multimenge keine Region ist, wird sie expandiert, indem zunächst ein Ereignis mit nichtkonstantem Gradient gewählt wird. Durch die Wahl alternativer Multiplizitäten bestimmter Zustände innerhalb des TS wird versucht, Gradientenkonflikte zu lösen und die Multimenge sukzessive in Richtung Region zu entwickeln. Die Auswahl von Zuständen, deren Multiplizität verändert wird, erfolgt auf Basis von Transitionen, die eine Verbindung zu Zuständen haben, die in der aktuell betrachteten Multimenge enthalten sind. Nach Ablauf der Suche werden alle nichtminimalen Regionen entfernt. Nicht in jedem Fall endet ein Explorationspfad in einer Region und nicht jede „Korrektur“ reduziert die Anzahl von Ereignissen mit nichtkonstantem Gradient. Für den zugehörigen Algorithmus wurde jedoch bewiesen, dass alle existierenden minimalen Mengen gefunden werden.

In Abb. 4.8 ist der Pfad aus Abb. 4.7 visualisiert, der zur Region $\{s_0^2, s_1, s_3, s_4^2, s_5, s_6^2, s_7\}$ führt. Obwohl zwischenzeitlich die Zahl von Ereignissen mit nichtkonstantem Gradient steigt, kann durch die Anpassung der Ausgangs-Multimenge die Regionseigenschaft schließlich sichergestellt werden. In jedem der fünf Teilschritte sind Ereignisse mit nichtkonstantem Gradient, sowie Zustände, deren Multiplizität als Reaktion darauf geändert wird, grau hinterlegt.

Sicherstellung erforderlicher Eigenschaften für Verhaltensäquivalenz

Die Verhaltensäquivalenz wird von Carmona et al. i.S. von Bisimilarität interpretiert. Zwei Systeme sind bisimilar, wenn jedes System das Verhalten des jeweils anderen „imitieren“ kann. Für jeden Zustand des einen Systems existiert ein Zustand innerhalb des anderen Systems, der dieselben Aktionen erlaubt, wobei die dabei erreichten Zustände jeweils ebenfalls in einer solchen *Simulations*-Relation stehen. Dafür ist nicht notwendigerweise eine strukturelle Übereinstimmung erforderlich (vgl. Isomorphismen).

Der GENET-Ansatz stellt sicher, das das Eingabe-TS bisimilar zum Erreichbarkeitsgraphen des Petrinetzes ist. Carmona et al.[30] konnten die Bisimilarität eines TS und des Erreichbarkeitsgraphen eines (unter Ausnutzung minimaler Regionen) daraus rekonstruierten Petrinetzes auf die Erfüllung der EC-Eigenschaft zurückführen. Diese Eigenschaft stellt sicher, dass das Schalten einer Transition im PN nur von solchen Zuständen beeinflusst wird, die auch im Ausgangs-TS das Stattfinden des entsprechenden Ereignisses

[30] Vgl. Carmona, Cortadella und Kishinevsky 2010.

zulassen. Andernfalls werden die Vorbedingungen für das Schalten von Transitionen relaxiert. Das Verhalten des PN würde in diesem Fall nicht mehr dem Verhalten des TS entsprechen, sondern dieses lediglich approximieren, was sowohl die Deckung als auch die Affinität des Netzes beeinträchtigt. Für eine formale Definition der EC-Eigenschaft werden zunächst Vorgänger- und Nachfolgeregionen definiert.

Definition 4.8 (Vorgänger- und Nachfolgeregionen). *Sei $TS = (S, E, A, s_{in})$ ein Transitionssystem. Eine Region r in TS ist eine Vorgängerregion von $e \in E$ gdw. sie den Anreizbereich des Ereignisses enthält: $ER(e) \subseteq r$. Eine Region r in TS ist eine Nachfolgeregion von $e \in E$ gdw. sie den Schaltbereich des Ereignisses enthält: $SR(e) \subseteq r$. Mengen von Vorgänger- und Nachfolgeregionen werden mit $^\circ e$ (Vorgänger) bzw. e° (Nachfolge) abgekürzt.* ⊣

Nicht in jedem Fall existiert für jedes Ereignis eine Vorgänger- bzw. Nachfolgeregion. Für Ereignis a gibt es keine Region in Abb. 4.6 (a), die $ER(a)$ enthält. Jedoch existiert mit r_4 eine 2-beschränkte minimale Vorgängerregion für a in Abb. 4.6 (b).

Die EC-Eigenschaft fordert nun, dass sich der Anreizbereich eines Ereignisses aus der Kombination dessen Vorgängerregionen ergibt. Die Kombination erfolgt auf Topsets von Vorgängerregionen, die gerade noch den Anreizbereich eines Ereignisses beinhalten:

Definition 4.9 (ER-Topset). *Sei e ein Ereignis und $r \in {}^\circ e$. Das ER-Topset von r bzgl. e ($\boldsymbol{ET}(r, e)$) ist die Multimenge q für die hinsichtlich eines **Grades** $g \in \mathbb{N}^+$ gilt:*

1. *$q = T_g(r)$ (Topseteigenschaft)*
2. *$ER(e) \subseteq T_g(r)$ (Enthaltensein des Anreizbereichs)*
3. *$ER(e) \nsubseteq T_{g+1}(r)$ (Maximalität)* ⊣

Die Menge der ER-Topsets aller Vorgängerregionen eines Ereignisses ist definiert als $e^* := \{q \mid \exists r \in {}^\circ e : q = \mathbf{ET}(r, e)\}$. Wie bereits erwähnt, werden PN-Stellen auf Basis von Regionen gebildet. Im Falle von k-beschränkten Petrinetzen werden diese nicht unverändert übernommen. Stattdessen wird auf deren ER-Topsets zurück gegriffen. Der Grad von ER-Topsets ($\mathbf{ET\text{-}D}(r, e)$) wird verwendet um das Gewicht eingehender und ausgehender Kanten zu bestimmen. Ein TS erfüllt die EC-Eigenschaft, wenn es ein k gibt, sodass alle Ereignisse geschlossen hinsichtlich Anreizbereichen sind. Regionen erfüllt sind:

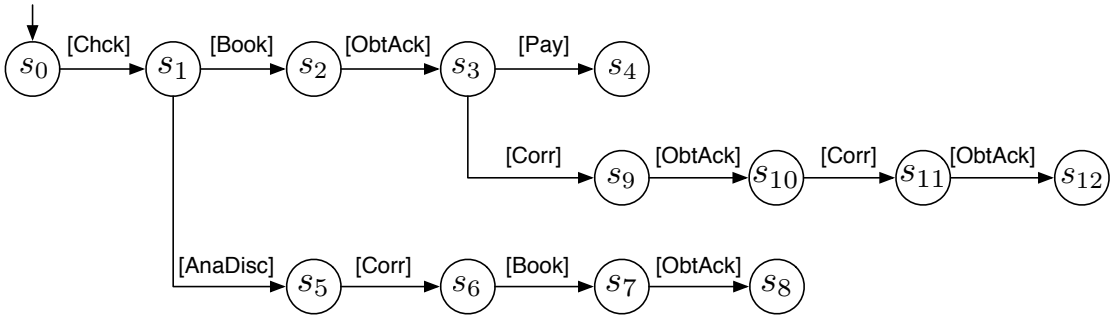
Definition 4.10 (Geschlossenheit hinsichtlich Anreizbereichen). *Ein Transitionssystem (S, E, A, s_{in}) ist unter Ausnutzung von k-beschränkten Regionen geschlossen hinsichtlich Anreizbereichen (k-EC), wenn folgende Bedingungen erfüllt sind:*

1. $\forall e \in E : \bigcap_{q \in e^*} supp(q) = ER(e)$.
2. $\forall e \in E : {}^\circ e \neq \emptyset$. ⊣

Für jedes Ereignis müssen die gemeinsamen Elemente der reduzierten Grundmengen von ER-Topsets der zugehörigen Vorgängerregionen also genau den Anreizbereich des Ereignisses ergeben. Das TS in Abb. 4.6 (b) ist nicht 2-EC, da die obigen Eigenschaften zwar

für c gelten, für alle übrigen Ereignisse jedoch nicht. Für b ergibt sich bspw. die Menge der Vorgängerregionen zu $\{r_1, r_3\}$. Die ER-Topsets entsprechen den Vorgängerregionen und die Kombination der reduzierten Grundmengen ergibt sich zu $\{s_1, s_2, s_3\} \neq \{s_2, s_3\} = ER(b)$. Auch das TS des Rechnungsprüfungsprozesses in Abb. 4.9 (a) ist nicht 1-EC. Jedoch kann die EC-Eigenschaft auf Basis 2-beschränkter Regionen sichergestellt werden. Im Folgenden wird mit (k-)ECTS ein Transitionssystem beschrieben, welches unter Ausnutzung k-beschränkter Regionen geschlossen hinsichtlich Anreizbereichen ist.

Ist es nicht möglich, für eine gegebene Schranke k ein k-ECTS zu finden, kann die Menge minimaler k-beschränkter Regionen durch Veränderung der TS-Struktur so manipuliert werden, dass die EC-Eigenschaft doch noch gewährleistet wird. In Abschnitt 4.4 werden zwei solche Korrekturmethoden vorgestellt.



(a)

$r_1 = \{s_0\}$
$r_2 = \{s_1, s_5, s_6\}$
$r_3 = \{s_3, s_5, s_8, s_{10}, s_{12}\}$
$r_4 = \{s_1, s_2, s_6, s_7, s_9, s_{11}\}$
$r_5 = \{s_1, s_2, s_3, s_9, s_{10}, s_{11}, s_{12}\}$
$r_6 = \{s_2, s_3, s_7, s_8, s_9, s_{10}, s_{11}, s_{12}\}$

$r_7 = \{s_4\}$
$r_8 = \{s_5, s_6, s_7, s_8\}$

(b)

redundant

$r_1 = \{s_0\}$
$r_2 = \{s_1, s_5, s_6\}$
$r_3 = \{s_3, s_5, s_8, s_{10}, s_{12}\}$
$r_4 = \{s_1, s_2, s_6, s_7, s_9, s_{11}\}$
$r_5 = \{s_1^2, s_2^2, s_3^2, s_5, s_9, s_{10}\}$
$r_6 = \{s_2, s_3, s_7, s_8, s_9, s_{10}, s_{11}, s_{12}\}$
$r_7 = \{s_1, s_2^2, s_3^2, s_5, s_7, s_8, s_9, s_{10}\}$
$r_8 = \{s_4\}$
$r_9 = \{s_5, s_6, s_7, s_8\}$
$r_{10} = \{s_2, s_6, s_7^2, s_8, s_9, s_{11}\}$
$r_{11} = \{s_1, s_2, s_3, s_9, s_{10}, s_{11}, s_{12}\}$
$r_{12} = \{s_1, s_6, s_9, s_{10}, s_{11}^2, s_{12}^2\}$
$r_{13} = \{s_6, s_7, s_8, s_9, s_{10}, s_{11}^2, s_{12}^2\}$
$r_{14} = \{s_2^2, s_3^2, s_5, s_7^2, s_8^2, s_9, s_{10}\}$

(c)

Abbildung 4.9.: TS für den Prozesslog $\mathfrak{L}_1$ (a), sowie zugehörige minimale 1-beschränkte Regionen (b) und minimale 2-beschränkte Regionen (b).

Algorithmus 4.1 Petrinetz Konstruktion

Eingabe: $\mathsf{TS} = (S, E, A, s_{in})$, minimale Regionen R
Ausgabe: Petri net $\mathsf{PN} = (R, E, W, M_0)$

```
1: procedure DeriveNet(TS, R)
2:     for all r ∈ R do[Initiale Markierung erzeugen]
3:         M_0[r] = r(s_in)
4:     end for
5:     for all e ∈ E do [Kanten für Transition e hinzufügen]
6:         for all r ∈ °e do [Hinzufügen von eingehenden Kanten]
7:             g = ET-D(r, e)
8:             W = W ∪ {(r -g-> e)}
9:             if (Δ_r(e) > −g) then [Verhinderung von negativen Markierungen]
10:                W = W ∪ {(e -g+Δ_r(e)-> r)}
11:            end if
12:        end for
13:        for all r ∈ e° do [Hinzufügen von ausgehenden Kanten]
14:            W = W ∪ {(e -Δ_r(e)-> r)}
15:        end for
16:    end for
17: end procedure
```

Petrinetz-Konstruktion auf Basis minimaler Regionen

Während jede minimale Region eine Stelle im resultierenden Petrinetz ergibt, entsprechen TS-Ereignisse PN-Transitionen. Der Gradient eines Ereignisses (im Petrinetz eine Transition) innerhalb einer Region wird verwendet, um das Gewicht der Kante zwischen Stelle und entsprechender Transition zu bestimmen. Dabei wird eine Kante zwischen einer Stelle und einer Transition hinzugefügt, wenn die der Stelle entsprechende Region eine Vorgängerregion des Ereignisses ist, das der Transition entspricht. Entsprechend wird für ausgehende Kanten der Transition auf Nachfolgeregionen zurückgegriffen. Alg. 4.1 beschreibt die Vorgehensweise zur Konstruktion des Petrinetzes.

Die Anzahl transportierter Marken entspricht bei eingehenden Kanten dem Grad von *ER*-Topsets (Zeile 8); bei ausgehenden Kanten dem Gradienten des Ereignisses innerhalb der Nachfolgeregion (Zeile 14). Die initiale Markierung einer Stelle (Anzahl der Marken) ergibt sich aus der Multiplizität des Initialzustands innerhalb der Region (Zeile 3). In einzelnen Fällen kann es vorkommen, dass einer Stelle im Preset einer Transition zu viele Marken entnommen werden, um negative Markierungen zu verhindern (Zeile 9). Dann muss ein Teil der Marken wieder in die Stelle zurück gelegt werden (Zeile 10). Das Petrinetz $\mathfrak{P}_1$ in Abb. 4.2(a) wurde auf Basis des Transitionssystems und der 2-beschränkten Regionen in Abb. 4.9 konstruiert. Die Stelle die auf horizontaler Linie zwischen der Transition [AnaDisc] und [Corr] liegt, entspricht der Region r_5. Sie hat eine eingehende Kante von Transition [Chck], weil r_5 eine Nachfolgeregion des Ereignisses [Chck] im TS ist ($SR([\text{Chck}]) = \{s_1\} \subseteq \{s_1^2, s_2^2, s_3^2, s_5, s_9, s_{10}\} = r_5$). Der Gradient von [Chck] in r_5 beträgt 2, also werden der Stelle beim Schalten von [Chck] zwei Marken

hinzugefügt. Die Stelle hat außerdem eine ausgehende Kante zu Transition [AnaDisc], da $ER([\text{AnaDisc}]) = \{s_1\} \subseteq r_5$. Das ER-Topset von [AnaDisc] in r_5 ist $\{s_1^2, s_2^2, s_3^2\}$ und dessen Grad 2. Deshalb werden der Stelle beim Schalten von [AnaDisc] zwei Marken entnommen. Der Gradient von [AnaDisc] in r_5 ist allerdings -1 und damit größer als der negative Grad, weshalb eine Marke wieder in die Stelle zurückgeführt werden muss um die Differenz auszugleichen. Solche Anpassungen sind notwendig, um negative Markierungen im resultierenden Petrinetz zu verhindern.

4.3.1 Redundanz minimaler Regionen

Die Menge minimaler k-beschränkter Regionen innerhalb eines k-ECTS ist hinreichend für die Konstruktion eines präzisen PN. Allerdings können minimale Regionen *redundant* sein. Eine minimale Region ist redundant, wenn genügend andere minimale Regionen zur Verfügung stehen, um die EC-Eigenschaft für ein Ereignis sicherzustellen. Dafür wird in Anlehnung an Def. 4.10 die *Deckung des Anreizbereichs* eines Ereignisses betrachtet:

Definition 4.11 (Deckung des Anreizbereichs). *Sei $TS = (S, E, A, s_{in})$ ein Transitionssystem und R die Menge minimaler Regionen bzgl. TS. Die Deckung des Anreizbereichs eines Ereignisses $e \in E$ hinsichtlich einer Teilmenge der minimalen Regionen $R' \subseteq R$ ist definiert als:*

$$DA(e) = \bigcap_{q \in e^* \cap R'} supp(q)$$

⊣

Reduziert die Hinzunahme einer minimalen Region die Deckung des Anreizbereichs für kein Ereignis, so kann die entsprechende Stelle aus dem resultierenden Petrinetz entfernt werden, ohne dass dessen Sprache dadurch verändert wird. Von den 14 2-beschränkten minimalen Regionen in Abb. 4.9 sind 8 redundant. Region r_7 wird bspw. nicht für die Sicherstellung der EC-Eigenschaft von Ereignis [Corr] benötigt. Die Menge der Vorgängerregionen von [Corr] ergibt sich zu ${}^\circ[\text{Corr}] = \{r_3, r_5, r_7, r_{10}\}$. Bei der Bildung der Schnittmenge der entsprechenden Topsets spielt es keine Rolle, ob r_7 mit einbezogen wird oder nicht. Da r_7 auch für kein anderes Ereignis benötigt wird, ist die Region redundant.

Unter der Annahme, dass für ein gegebenes Ereignis drei Vorgängerregionen existieren, veranschaulicht Abb. 4.10 deren Zusammenhang im Sinne gemeinsamer Elemente zugehöriger reduzierter Grundmengen. Blaue Bereiche enthalten Elemente im Überschneidungsbereich jeweils zweier Regionen, der rote Bereich enthält Elemente, die in allen drei Regionen vorhanden sind, also den Anreizbereich des Ereignisses. Die linke Grafik steht dabei für die Situation, in der keine der Vorgängerregionen redundant ist. Da jeder blaue Bereich mindestens ein Element enthält, wird die Deckung des Anreizbereichs von jeder beteiligten Region reduziert. In der rechten Grafik enthält keiner dieser Bereiche Elemente, was dazu führt, dass jede Vorgängerregionen des Ereignisses redundant ist. Dieser Spezialfall muss gesondert behandelt werden, da bei Entfernung aller redundanter Regionen die EC-Eigenschaft des Ereignisses verletzt wird. Bisher wurden keine Lösungsvorschläge für diese Problematik publiziert. Der im Rahmen dieser Arbeit entwickelte Ansatz bestimmt hierfür zunächst sog. *Konfliktereignisse*, bei denen dieser Spezialfall eintritt und überprüft anschließend für jedes dieser Ereignisse, welche Untermengen der

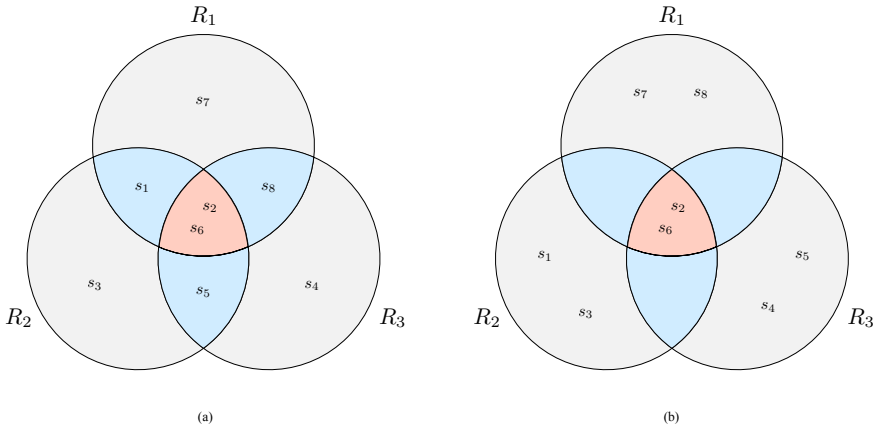


Abbildung 4.10.: Mögliche Zusammenhänge von Vorgängerregionen eines Ereignisses; (a) keine Region ist redundant, (b) alle Regionen sind redundant.

Vorgängerregionen sicher entfernt werden können, d.h. ohne die EC-Eigenschaft zu verletzen. Danach wird für jedes Ereignis eine Regionsmenge ausgewählt, sodass die Kombination der gewählten Mengen aller Ereignisse maximal ist, also am meisten redundante Regionen enthält. Damit wird sichergestellt, dass stets so viele redundante Regionen wie möglich entfernt werden. Eine randomisierte Auswahl ist zwar deutlich schneller, führt aber ggf. dazu, dass das resultierende Petrinetz redundante Stellen enthält.

4.4 TS-Manipulation zur Gewährleistung der EC-Eigenschaft

Laut Carmona et al.[31] existiert immer ein k, sodass das auf Basis eines TS konstruierte PN k-beschränkt ist und dessen Erreichbarkeitsgraph bisimilar zu TS ist. Für Fälle, in denen eine obere Schranke für k existiert, die nicht überschritten werden darf, existieren zwei Verfahren zur Anpassung der TS-Struktur, um dennoch die EC-Eigenschaft zu gewährleisten. Während *Ereignisspaltung* (engl.: *label splitting*) Ereignis-Bezeichner umbenennt[32], dupliziert *Zustandsspaltung* (engl.: *state splitting*) TS-Zustände. Die Zustandsspaltung wurde im Rahmen dieser Arbeit als Alternative zur Ereignisspaltung entwickelt. Dabei treten im resultierenden Petrinetz keine Transitionen mehrfach auf. Stattdessen enthält das Netz zusätzliche stille Transitionen. Im Folgenden werden beide Verfahren erläutert.

4.4.1 Ereignisspaltung

Ist es nicht möglich, für ein TS-Ereignis die EC-Eigenschaft auf Basis von minimalen Regionen zu gewährleisten, wird bei der Ereignisspaltung das Ereignis durch eine Menge neu eingeführter Ereignisse ersetzt. Kanten des entfernten Ereignisses werden mit neuen

[31] Vgl. Carmona, Cortadella und Kishinevsky 2010.
[32] Vgl. Carmona 2012.

Ereignissen ausgezeichnet. Durch die so erreichte Reduktion der Kantenmenge pro Ereignis soll die Gewährleistung der EC-Eigenschaft erleichtert werden. Konkret gibt es zwei verschiedene Strategien:

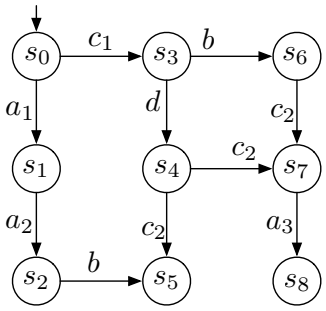
1. **Partitionierung des Anreizbereichs**. Bei dieser Strategie wird überprüft, ob der Anreizbereich eines Ereignisses auf eine Weise partitioniert werden kann, dass mind. eine Partition die EC-Eigenschaft erfüllt. Für Ereignis a in Abb. 4.6 (b) gilt $DA(a) = \{s_0, s_1, s_3, s_4, s_5, s_6, s_7\} \neq \{s_0, s_1, s_7\} = ER(a)$. Betrachtet man jedoch die Partitionierung $ER(a) = \{s_0\} \dot{\cup} \{s_1, s_7\}$ und alternative Ereignisse a_1, a_2 mit $ER(a_1) = \{s_0\}$ und $ER(a_2) = \{s_1, s_7\}$, kann zumindest die EC-Eigenschaft für a_1 sichergestellt werden, da sich die Deckung des Anreizbereichs für a_1 zu $DA(a_1) = \{s_0, s_1, s_2, s_3\} \cap \{s_0, s_4, s_6\} = \{s_0\} = ER(a_1)$ ergibt. Bei dieser Strategie ist allerdings nicht klar, wie verfahren wird, wenn mehrere solcher Ereignisse mit ggf. mehreren möglichen Partitionierungen existieren. Es erscheint sinnvoll, Partitionierungen mit einer geringeren Anzahl von Komponenten zu präferieren, um die Anzahl neu eingeführter Bezeichner zu minimieren. Bei dieser Strategie ist es nicht in jedem Fall möglich, ein Ereignis zur Spaltung auszuwählen. Abb. 4.12 zeigt eine Variante des TS aus Abb. 4.6. Das einzige Ereignis, für das die erforderliche EC-Eigenschaft nicht gilt, ist a_3. Da dessen Anreizbereich allerdings nur ein Element enthält, kann keine weitere Partitionierung erfolgen.

2. **Deckungsorientierte Spaltung**. Bei dieser Strategie wird zunächst eine bei der Suche nach minimalen Regionen aufgetretene Multimenge (*ER*-Multimenge) m ausgewählt. Für ein Ereignis mit nichtkonstantem Gradient innerhalb m werden zugehörige Kanten im TS anhand der unterschiedlichen Gradienten partitioniert und jeweils mit einem eindeutigen neu eingeführten Ereignis versehen. Jeder Spaltungsschritt ersetzt ein Ereignis e des TS mit einer Menge alternativer Ereignisse $e_1, \ldots, e_k$, sodass die Gradienten der Ereignisse e_i bzgl. m nach der Spaltung konstant sind.

Die durch eine Spaltungsoperation herbeigeführte, strukturelle Änderung eines TS beeinflusst die Anzahl und Ausprägung minimaler Regionen. Nach jeder Spaltungsoperation wird deshalb die Menge minimaler Regionen erneut berechnet und geprüft, ob die EC-Eigenschaft gilt. Durch Ereignisspaltung werden keine Regionen „verletzt“, d.h. jede Multimenge, die zuvor die Regionseigenschaft erfüllt hat, tut dies nach der Spaltungsoperation weiterhin[33]. Die Menge minimaler Regionen ist bei mehrfacher Anwendung von Ereignisspaltung monoton steigend. Insbesondere existiert für jedes nicht k-ECTS eine endliche Folge von Spaltungsoperationen, um die EC-Eigenschaft sicherzustellen[34]. Dies ist spätestens dann der Fall, wenn jede Kante einen eindeutigen Bezeichner trägt.

TS-Ereignisse entsprechen den Transitionen des rekonstruierten Petrinetzes. Obwohl einzelne Ereignisse aufgrund von Spaltungsoperationen im Verlauf des Rekonstruktionsverfahrens möglicherweise durch neue Ereignisse ersetzt werden, beziehen sich diese neuen Ereignisse dennoch stets auf dieselbe Ursprungsaktivität im Prozesslog. Die Notation $a_{\blacktriangleleft}$ wird eingeführt, um die Ursprungsaktivität eines Ereignisses a zu bezeichnen. Sofern der Verwendete Formalismus für die Darstellung eines Petrinetzes eine Unterscheidung

[33] Vgl. Carmona 2012.
[34] Vgl. Carmona, Cortadella und Kishinevsky 2010.



$r_1 = \{s_0\}$
$r_2 = \{s_1\}$
$r_3 = \{s_8\}$
$r_4 = \{s_2, s_3\}$
$r_5 = \{s_2, s_5, s_7\}$
$r_6 = \{s_3, s_4, s_6\}$
$r_7 = \{s_4, s_5, s_6, s_7\}$

$$DA(a_3) = supp(r_5) \cap supp(r_7) = \{s_5, s_7\} \neq \{s_7\} = ER(a_3)$$

Abbildung 4.11.: Situation, in der mittels Partitionierung von Anreizbereichen keine weitere Ereignisspaltung mehr möglich ist.

zwischen eindeutigen Namen von Transitionen und nicht notwendigerweise eindeutigen, zusätzlichen Bezeichnern erlaubt, kann a als Name und $a_◀$ als Bezeichner für die entsprechende Transition verwendet werden. Diese Maßnahme kann das Verständnis eines Netzes erheblich erleichtern.

Jede Spaltungsoperation verändert die Menge minimaler Regionen und führt das zugrundeliegende TS näher an die EC-Eigenschaft. Generell ist es wünschenswert, für die Erreichung der EC-Eigenschaft möglichst wenig Spaltungsoperationen durchzuführen, um das PN nicht unnötig „aufzublähen“. Eine aktuelle Arbeit von Carmona verwendet für die minimal mögliche Anzahl zusätzlicher Ereignisse ein Verfahren, das auf einer Kombination aus Graphen-Färbbarkeit und dem NP-vollständigen Mengenüberdeckungsproblem[35] basiert. Praktikable Implementierungen existieren bisher keine; die prinzipielle Machbarkeit konnte jedoch nachgewiesen werden.

Auswahl geeigneter Multimengen

Bei der Auswahl einer geeigneten Multimenge im Fall von deckungsorientierter Spaltung werden zwei Vorgehensweisen unterschieden. Die erste Methode wählt für jedes Ereignis, das die EC-Eigenschaft nicht erfüllt, die *ER*-Multimenge mit der maximalen Anzahl von Ereignissen mit konstantem Gradient aus. Aus dieser Vorauswahl wird dann erneut die Multimenge mit der höchsten Zahl von Ereignissen mit konstantem Gradient gewählt. Auf diese Weise wird versucht, die vorgenommenen Änderungen so minimal und damit den „Preis“ für die Gewährleistung der EC-Eigenschaft so gering wie möglich zu halten.

Eine weitere Methode zur Auswahl einer Multimenge orientiert sich direkt an der Deckung von Anreizbereichen. Im Folgenden wird ein auf Ideen von Carmona aufbauendes Verfahren vorgestellt, welches für jedes Ereignis e, das nicht die EC-Eigenschaft erfüllt, zunächst die Deckung des Anreizbereichs bestimmt und dann für jede zugehörige *ER*-Multimenge den Beitrag misst, den sie zur Gewährleistung der EC-Eigenschaft liefert. Gewählt wird dann die Multimenge, die hinsichtlich aller Ereignisse den höchsten Beitrag liefert.

[35] Vgl. Carmona 2012.

Da e die EC-Eigenschaft nicht erfüllt, gibt es entweder keine Vorgängerregionen ($^\circ e = \emptyset$ und damit $DA(e) = \emptyset$) oder die Kombination der Vorgängerregionen ergibt nicht den Anreizbereich des Ereignisses, sondern enthält zusätzliche Zustände (*Differenzmenge*). Die Differenzmenge ist definiert als Mengenfunktion $ECD(e) = DA(e) \setminus ER(e)$. Der Beitrag einer Multimenge m für die Deckung des Anreizbereichs eines Ereignisses e wird gemessen mit:

$$score(e,m) = \begin{cases} 1 - \frac{|ECD(e) \setminus supp(m)|}{|ECD(e)|} & , {}^\circ e \neq \emptyset \\ \left(1 - \frac{|supp(m) \setminus ER(e)|}{|supp(m)|}\right) * \left(1 - \frac{|ER(e) \setminus supp(m)|}{|ER(e)|}\right) & , {}^\circ e = \emptyset \end{cases}$$

Falls Vorgängerregionen existieren, gibt der Beitrag an, um wie viel Prozent die Differenzmenge von e durch Hinzunahme von m reduziert werden kann. Andernfalls wird die Übereinstimmung von reduzierter Grundmenge und dem Anreizbereich des Ereignisses gemessen. Je mehr Zustände ausschließlich in der Multimenge oder im Anreizbereich liegen, desto niedriger fällt der Wert aus, der in diesem Fall als Eignung einer Multimenge für die Herstellung der EC-Eigenschaft eines Ereignisses aufgefasst wird.

Um den Gesamtbeitrag einer Multimenge hinsichtlich aller Ereignisse zu berechnen, müssen Beiträge auf Ereignisebene akkumuliert werden. Existiert nur ein Ereignis für das die EC-Eigenschaft nicht gilt, entspricht der Gesamtbeitrag dem Beitrag für das Ereignis. Bei zwei Ereignissen ergibt sich der Gesamtbeitrag aus dem arithmetischen Mittel der Beiträge für die einzelnen Ereignisse. Für alle anderen Fälle wird zur Ermittlung des Gesamtbeitrags auf ein Netzdiagramm zurückgegriffen. Ein Netzdiagramm mit Grad n beinhaltet Werte für n unterschiedliche Kategorien, die grafisch mithilfe eines regelmäßigen n-Ecks dargestellt werden. Jede Verbindung vom Mittelpunkt des n-Ecks zu einem Eckpunkt wird genutzt, um den Wert einer Kategorie (im Intervall $[0;1]$) i.S. der Entfernung zum Mittelpunkt abzutragen. Je höher der Wert, desto weiter liegt er vom Mittelpunkt entfernt (bei 1 direkt auf dem Eckpunkt). Hier entsprechen die Kategorien den betrachteten Ereignissen. Werte für Beiträge hinsichtlich einzelner Ereignisse ergeben sich zu $score(e_i, m)$. Ein Netzdiagramm auf Basis einer Menge betrachteter Ereignisse $E = \{e_1, \ldots, e_n\}$ und einer Multimenge m wird mit $\mathbb{D}(E, m)$ bezeichnet. Der mithilfe eines Netzdiagramms ermittelte Gesamtbeitrag einer Multimenge ($\chi(\mathbb{D}(E, m))$) entspricht dem Verhältnis der Fläche des n-Ecks das durch die abgetragenen Werte induziert wird zur Maximalfläche. Je mehr Ereignisse positiv beeinflusst werden, desto größer wird die Fläche und damit das Verhältnis. Die Ereignisse werden innerhalb des Netzdiagramms anhand zugehöriger Beiträge sortiert, um die Eindeutigkeit von Gesamtbeiträgen und damit deren Vergleichbarkeit zu sichern. Allgemein ergibt sich der Gesamtbeitrag einer Multimenge zu:

$$score(m) = \begin{cases} score(e,m) & , E = \{e\} \\ \frac{score(e_1,m) + score(e_2,m)}{2} & , E = \{e_1, e_2\} \\ \chi(\mathbb{D}(E,m)) & , sonst \end{cases}$$

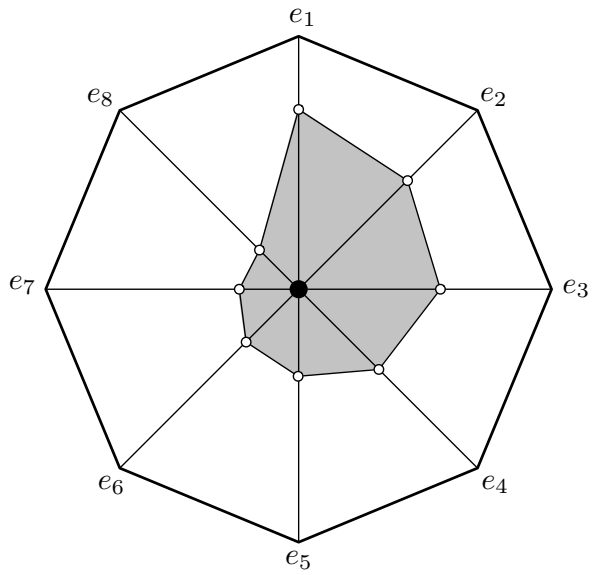


Abbildung 4.12.: Netzdiagramm für den Beitrag einer Multimenge zur Sicherstellung der EC-Eigenschaft 8 verschiedener Ereignisse.

Für die Spaltung wird aus einer gegebenen Menge von Multimengen M schließlich die Multimenge mit maximalem Gesamtbeitrag gewählt: $max_{m \in M} score(m)$.

4.4.2 Zustandsspaltung

Während bei der Ereignisspaltung TS-Ereignisse umbenannt und dadurch entweder direkt die EC-Eigenschaft sichergestellt oder zunächst neue Regionen erzeugt werden, werden bei der Zustandsspaltung neue Zustände und Ereignisse zwischen Quell- und Zielzuständen ausgewählter Transitionen eingeführt[36]. Die so erreichte Separierung von TS-Ereignissen räumt der Methode zur Berechnung minimaler Regionen mehr Freiraum für die Wahl von Gradienten ein und vereinfacht so die Sicherstellung der EC-Eigenschaft. Die Zustandsspaltung vermeidet durch das Hinzufügen zusätzlicher stiller Transitionen die Umbenennung von Ereignissen und eröffnet dadurch bei der Rekonstruktion die Möglichkeit mehrfaches Auftreten von Prozessaktivitäten in rekonstruierten Modellen zu umgehen.

Die prinzipielle Vorgehensweise von Zustandsspaltung ist, durch Anwendung von Spaltungsoperationen auf Transitionen innerhalb des TS konstante Gradienten von Ereignissen bzgl. einer *ER*-Multimenge sicherzustellen und diese Multimenge dadurch in eine Region zu transformieren. Im Folgenden wird nachgewiesen, dass für jede *ER*-Multimenge und jedes Ereignis mit nichtkonstantem Gradient durch (mehrfache) Anwendung von Spaltungsoperationen ein konstanter Gradient erreicht werden kann, ohne dadurch die Regionseigenschaft bisheriger Regionen zu verletzen. Analog zur Ereignisspaltung wird

[36] Bei den vorgestellten Konzepten der Zustandsspaltung handelt es sich um bisher nicht publizierte Ergebnisse einer Kooperation mit Prof. Josep Carmona (UPC Barcelona).

nach jeder Transformation die Menge minimaler Regionen erneut berechnet, da Zustandsspaltung die Minimalität von Regionen beeinträchtigen kann. Bei der Zustandsspaltung bzgl. einer Transition (s, e, t) werden folgende zwei Operationen unterschieden:

1. **Quellspaltung**: Verbindet den Quellzustand s mit einem neu eingeführten Zustand s_{δ_s} über ein neues Ereignis δ_s und fügt eine weitere mit e bezeichnete Transition von s_{delta_s} zum Zielzustand t ein. TS-Transformationen unter Anwendung dieses Spaltungstyps werden mit $\mathsf{TS} \rightharpoonup \mathsf{TS}'$ bezeichnet (siehe Abb. 4.13(b)).
2. **Zielspaltung**: Verbindet den Quelluustand s mit einem neu eingeführten Zustand s_{δ_s} über Ereignis e und fügt eine weitere mit einem neuen Ereignis δ_s bezeichnete Transition von s_{δ_s} zum Zielzustand t ein. TS-Transformationen unter Anwendung dieses Spaltungstyps werden mit $\mathsf{TS} \rightharpoondown \mathsf{TS}'$ bezeichnet (siehe Abb. 4.13(a)).

Die Transformation eines Transitionssystems TS in ein Transitionssystem TS' mithilfe einer Spaltungsoperation wird mit $\mathsf{TS} \twoheadrightarrow \mathsf{TS}'$ oder der Funktion $SPLIT$ bezeichnet. Def. 4.12 enthält eine formale Definition der Spaltungsoperationen.

Definition 4.12. ***(Spaltungsoperationen)*** *Gegeben sei ein TS (S, E, A, s_{in}) und mit $(s, e, t) \in A$ die Transition, auf welche die Spaltungsoperation angewandt wird. Seien weiter s_{δ_s} und δ_s die durch die Operation neu eingeführten Zustände/Ereignisse. Die Spaltung erzeugt ein TS (S', E', A', s_{in}) mit:*

- $S' := S \cup \{s_{\delta_s}\}$
- $E' := E \cup \delta_s$
- $A' := (A \setminus \{(s, e, t)\}) \cup \begin{cases} \{(s, \delta_s, s_{\delta_s}), (s_{\delta_s}, e, t)\} & , \textit{Quellspaltung} \\ \{(s, e, s_{\delta_s}), (s_{\delta_s}, \delta_s, t)\} & , \textit{Zielspaltung} \end{cases}$ ⊣

Um einen Beitrag zur Gewährleistung der EC-Eigenschaft zu leisten, muss es durch Zustandsspaltung möglich sein, bessere Voraussetzungen für die Geschlossenheit von Anreizbereichen von TS-Ereignissen im Sinne der Generierung zusätzlicher minimaler Regionen zu schaffen. Neue δ-Ereignisse werden deshalb auf eine Weise hinzugefügt, die es für eine *ER*-Multimenge ω erlaubt, alternative Gradienten zu wählen, sodass ω zur Region wird. Eine *ER*-Multimenge wird dabei nicht explizit als Region redefiniert. Stattdessen werden die Voraussetzungen dafür geschaffen, dass bei einer erneuten Berechnung minimaler Regionen geeignete Gradienten für die Multimenge gewählt werden können, sodass diese die Regionseigenschaft erfüllt.

Lemma 4.1. ***(Anpassbarkeit von Gradienten)*** *Gegeben sei ein TS (S, E, A, s_{in}) und mit $a = (s, e, t) \in A$ eine Transition. Sei weiter $k \in \mathbb{N}$ eine obere Schranke für die Multiplizität von Zuständen in Multimengen und ω eine k-beschränkte ER-Multimenge. Dann ist es mithilfe von Spaltungsoperationen möglich, für a einen Gradient $\varphi \in [-k; k]$ sicherzustellen.* ⊣

Wird eine Spaltungsoperation auf ein TS zur Gewährleistung eines Gradienten φ innerhalb einer *ER*-Multimenge ω angewandt, wird die dadurch implizierte Transformation mit $SPLIT(\mathsf{TS}, \omega, \varphi)$ bezeichnet.

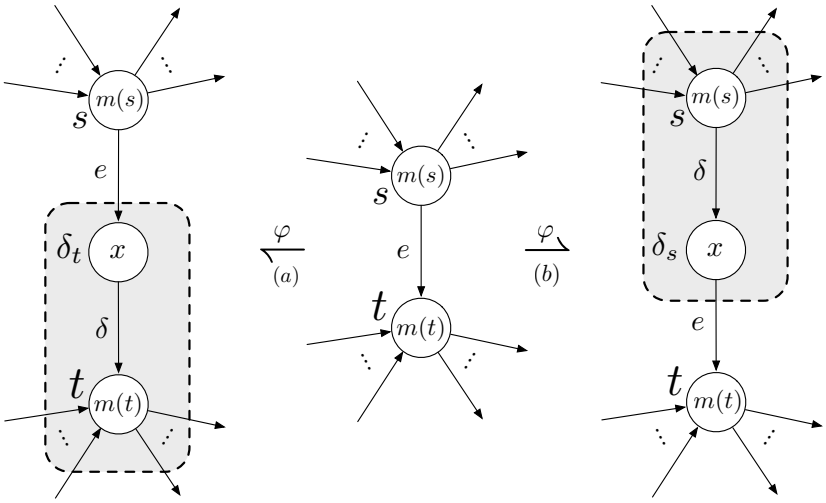


Abbildung 4.13.: Spaltungsoperationen: (a) Zielspaltung, (b) Quellspaltung

Beweis. Sei δ_s das bei der auf a angewandten Spaltung neu hinzugefügte Ereignis. Für jede Multimenge gelten hinsichtlich einer gegebenen Schranke k nach Definition folgende Bedingungen:

(1) Die Multiplizität jedes Zustands ist Null oder positiv und beschränkt durch k, i.e. $k \geq \omega(v) \geq 0, \forall v \in S$.

(2) Der Gradient jedes Ereignisses (und damit auch φ) liegt im Intervall $[-k; k]$.

Für den Beweis von Lemma 4.1 wird nachgewiesen, dass stets für mind. eine Spaltungsoperation die Multiplizität des neu eingeführten Zustands s_{δ_s} so gewählt werden kann, dass die Transition a Gradient φ in ω aufweist.

Im Fall von Quellspaltung muss für die Garantie von $\Delta(a) = \varphi$ die Multiplizität von s_{δ_s} folgendermaßen gewählt werden: $\omega(s_{\delta_s}) = \omega(t) - \varphi$. Aufgrund folgender Ungleichungen ist jedoch klar, dass dies nur unter der Bedingung $\omega(t) \in [\varphi; k]$ möglich ist. Durch Quellspaltung allein kann also nicht in jedem Fall eine adäquate Gradientenanpassung erzielt werden.

$$\begin{aligned} & \omega(t) - \varphi = \omega(s_{\delta_s}) \\ \overset{(\omega(s_{\delta_s}) \geq 0)}{\Longleftrightarrow} \; & \omega(t) - \varphi \geq 0 \\ \Leftrightarrow \; & \omega(t) \geq \varphi \end{aligned}$$

Im Fall von Zielspaltung muss für die Garantie von $\Delta(a) = \varphi$ die Multiplizität von s_{δ_s} folgendermaßen gewählt werden: $\omega(s_{\delta_s}) = \varphi + \omega(s)$. Auch für die Zielspaltung existiert eine Einschränkung. Sie ist nur zielführend unter der Bedingung $\omega(s) \in [-\varphi; k]$:

$$\begin{aligned} & \varphi + \omega(s) = \omega(s_{\delta_s}) \\ \overset{(\omega(s_{\delta_s}) \geq 0)}{\Longleftrightarrow} \quad & \varphi + \omega(s) \geq 0 \\ \Leftrightarrow \quad & \omega(s) \geq -\varphi \end{aligned}$$

Für den Nachweis, dass für jedes φ und jede Schranke k, stets mind. eine Spaltungsoperation anwendbar ist, werden folgende drei Fälle unterschieden:

1. $\varphi = 0$:
   Unter Ausnutzung von (1) folgen $\omega(t) \geq \varphi$ und $\omega(s) \geq -\varphi$ direkt.
   Quell- und Zielspaltung sind anwendbar.

2. $\varphi \in [-k, 0)$:
   Wenn φ negativ ist, ist Zielspaltung nur anwendbar, wenn gilt $\omega(s) \geq -\varphi$. Quellspaltung ist anwendbar wenn $\omega(s) \geq \varphi$. Da $\omega(s)$ stets positiv ist, ist Quellspaltung für jedes negative φ anwendbar.

3. $\varphi \in (0, k]$:
   Wenn φ positiv ist, ist Quellspaltung nur anwendbar, wenn gilt $\omega(t) \geq \varphi$. Zielspaltung ist anwendbar, wenn $\omega(s) \geq -\varphi$. Da $\omega(s)$ stets positiv ist, ist Zielspaltung für jedes positive φ anwendbar.

Für die Multiplizität des neu hinzugefügten Zustands muss gelten $\omega(s_{\delta_s}) \leq k$. Kann für die Anpassung des Gradienten von a keine entsprechende Multiplizität für s_{δ_s} gewählt werden, muss das Ereignis e vom Start- und Zielzustand der Relation a isoliert werden. D.h. es erfolgt für a sowohl eine Quellspaltung als auch eine Zielspaltung und es werden insgesamt zwei neue Zustände (s_{δ_s}, $s_{\delta'_s}$) und zwei neue Ereignisse (δ_s,δ'_s) hinzugefügt. Hierbei entfallen die obigen Bedingungen für die Anwendbarkeit von Quell- bzw. Zielspaltung, da der geforderte Gradient φ nicht lediglich durch eine Spaltungsoperation gewährleistet wird. Als Ersatz für die Relation (s, e, t) werden die zwei Relationen $(s, \delta_s, s_{\delta_s})$, $(s_{\delta_s}, e, s_{\delta'_s})$ und $(s_{\delta'_s}, \delta'_s, t)$ hinzugefügt. Die Wahl der Multiplizitäten von s_{δ_s} und $s_{\delta'_s}$ können nun frei gewählt werden, um den erforderlichen Gradienten φ sicherzustellen. □

Lemma 4.1 kann genutzt werden, um zu zeigen, dass es durch mehrfache Anwendung von Spaltungsoperationen möglich ist, die Voraussetzungen für die Erfüllung der Regionseigenschaft von mind. einer *ER*-Multimenge zu schaffen.

Lemma 4.2. ***(Transformation von ER-Multimengen)*** *Sei $TS = (S, E, A, s_{in})$ ein Transitionssystem und ω eine k-beschränkte ER-Multimenge bzgl. TS, dann kann ω mithilfe (wiederholter) Anwendung von Spaltungsoperationen gemäß Alg. 4.2 in eine k-beschränkte Region transformiert werden.* ⊣

Algorithmus 4.2 Gradientenkorrektur

Eingabe: Transitionssystem $TS = (S, E, A, s_{in})$, *ER*-Multimenge ω

```
1:  procedure STATE-SPLITTING(TS, ω)
2:      Ê := {e ∈ E | ∃a1, a2 ∈ A : Δ_ω(a1) ≠ Δ_ω(a2)}
3:      for all ê ∈ Ê do
4:          φ ← argmax_g |{a ∈ A_e | Δ_ω(a) = g}|
5:          for all a ∈ A_ê do
6:              if Δ_a(ê) ≠ φ then
7:                  SPLIT(TS, a, φ)
8:              end if
9:          end for
10:     end for
11: end procedure
```

Für jedes Ereignis $\hat{e}$ mit nichtkonstantem Gradient in ω wählt Alg. 4.2 als Zielgradient den am häufigsten auftretenden Gradienten aller mit $\hat{e}$ markierten Transitionen φ und benutzt anschließend für jede Transition mit nicht korrektem Gradient entweder Quellspaltung oder Zielspaltung, um Gradient φ sicher zu stellen. Aufgrund von Lemma 4.1 ist klar, dass dies für jeden Gradient $\varphi \in [-k; k]$ möglich ist. Wird Alg. 4.2 für jedes Ereignis mit nichtkonstantem Gradient ausgeführt, kann eine *ER*-Multimenge sukzessive in eine Region transformiert werden.

Satz 4.1. *Sei $TS = (S, E, A, s_{in})$ ein TS, R die zugehörige Menge von Regionen und ω eine k-beschränkte ER-Multimenge von TS. Dann vergrößert die Transformation von ω gemäß Alg. 4.2 die Regionsmenge R.* ⊣

Um die Vergrößerung der Regionsmenge R durch Alg. 4.2 nachzuweisen, wird gezeigt, dass für jede Region $r \in R$ die Multiplizität jedes Zustands s_{δ_*}, der bei der Spaltung hinzugefügt wurde, so gewählt werden kann, dass r weiterhin die Regionseigenschaft erfüllt. Für jedes s_{δ_*} werden die folgenden zwei Fälle unterschieden:

(1) s_{δ_*} wurde aufgrund einer Quellspaltung hinzugefügt, bei der die Transition(s, e, t) durch $(s, \delta_s, s_{\delta_*})$ und (s_{δ_*}, e, t) ersetzt wurde.
In diesem Fall wähle $r(s_{\delta_*}) = r(s)$.

(2) s_{δ_*} wurde aufgrund einer Zielspaltung hinzugefügt, bei der die Transition (s, e, t) durch (s, e, s_{δ_*}) und $(s_{\delta_*}, \delta_s, t)$ ersetzt wurde.
In diesem Fall wähle $r(s_{\delta_*}) = r(t)$.

Diese Vorgehensweise stellt sicher, dass Ereignis e denselben Gradient in r aufweist wie zuvor. Da s_{δ_*} keine weiteren Kanten aufweist, die mit anderen Ereignissen bezeichnet sind, wird für kein anderes Ereignis der Gradient in r beeinträchtigt. Ereignisse, die bei Spaltungsoperationen eingefügt werden, sind eindeutig, deshalb haben sie automatisch konstanten Gradient in r. Insgesamt ist dadurch nachgewiesen, dass jede Region $r \in R$ auch nach der Anwendung von Zustandsspaltung die Regionseigenschaft erfüllt. Satz 4.1 folgt aus der Kombination obiger Argumentation mit der Tatsache, dass die Transformation durch die Zusicherung konstanter Gradienten für alle Ereignisse in ω eine neue Region erzeugt. In diesem Zusammenhang ist erwähnenswert, dass für jede Region $r \in R$

das Hinzufügen von δ-Events die Existenz von Teilmengen $r'_1, \ldots, r'_n$ mit $r'_i \subseteq r$ bedingen kann, die jeweils selbst die Regionseigenschaft erfüllen. Wenn gilt $(\bigcup_{r_i} r'_i) = r$, ist r nicht weiter minimal. Die Gültigkeit von Satz 4.1 bleibt dabei unberührt, da in diesem Fall die Zahl der Regionen sogar noch höher ausfällt.

Bei der Anwendung von Spaltungsoperationen auf eine Transition $s \xrightarrow{e} t$ wird das Ereignis e von Quell- und Zielzustand mithilfe von eindeutigen δ-Ereignissen und -Zuständen isoliert. Dabei stellt $s \xrightarrow{\delta_s} s_{\delta_s} \xrightarrow{e} s_{\delta_t} \xrightarrow{\delta_t} t$ die maximale Spaltung der Transition dar. Im Extremfall sind alle TS-Transitionen maximal gespalten.

Definition 4.13. ***(Maximale Spaltung)*** *Sei $TS = (S_{reg} \dot\cup S_\delta, E_{reg} \dot\cup E_\delta, A, s_{in})$ ein TS und o.B.d.A. E_{reg} sowie S_{reg} die Menge der Ereignisse bzw. Zustände, die bereits vor der Anwendung von Zustandsspaltung Teil von TS waren und E_δ sowie S_δ die Menge der Ereignisse/Zustände, die im Zuge der Spaltung Eingang in TS gefunden haben. TS ist maximal gespalten, wenn gilt:*

$$\forall e \in E_{reg}\, \exists s_1, s_2 \in S_{reg} \wedge e_{\delta_1}, e_{\delta_2} \in E_\delta \wedge s_{\delta_1}, s_{\delta_2} \in S_\delta : \\ \{(s_1, e_{\delta_1}, s_{\delta_1}), (s_{\delta_1}, e, s_{\delta_2}), (s_{\delta_2}, e_{\delta_2}, s_2)\} \subseteq A$$ ⊣

Im Falle maximaler Spaltung gilt die EC-Eigenschaft auf triviale Weise. Der Anreizbereich jedes regulären Ereignisses besteht dann ausschließlich aus δ-Zuständen, die wiederum lediglich eine eingehende Kante, bezeichnet mit einem eindeutigen δ-Ereignis, besitzen. Bei der Wahl derselben Multiplizität für jeden dieser Zustände gibt es kein Ereignis ohne konstanten Gradient. Somit erfüllt der Anreizbereich die Regionseigenschaft und das reguläre Ereignis damit die EC-Eigenschaft. Der Anreizbereich eines neu hinzugekommenen δ-Ereignisses enthält stets einen einzigen Zustand $\hat{s}$, der entweder bereits vor der Spaltung Teil des TS war oder neu hinzugekommen ist. Im ersten Fall erfüllt der Anreizbereich des Ereignisses die Regionseigenschaft, weil alle eingehenden und ausgehenden Transitionen mit eideutigen δ-Ereignissen bezeichnet sind. Dies ist eine direkte Folge maximaler Spaltung. Im zweiten Fall hat der einzige Zustand des Anreizbereichs eine einzige mit einem regulären Ereignis bezeichnete eingehende Transition. Existiert keine weitere Transition im TS mit derselben Bezeichnung, erfüllt der Anreizbereich die Regionseigenschaft direkt. Andernfalls kann er durch die Kombination einer Menge von Regionen hergestellt werden. Sei e_δ das betreffende Ereignis, $ER(e_\delta) = \{s_\delta\}$ dessen Anreizbereich und (s'_δ, e, s_δ) die Transition über die s_δ erreicht werden kann. Für die Gewährleistung der EC-Eigenschaft für e_δ müssen mind. zwei Regionen r_1, r_2 gefunden werden, für die gilt $supp(r_1) \cap supp(r_2) = \{s_\delta\}$. Die erste Region kann mit $r_1 = \{s'_\delta, s_\delta\}$ gewählt werden. Die Regionseigenschaft wird von r_1 erfüllt, da das reguläre Ereignis darin Gradient 0 hat und die Multimenge ansonsten nur eindeutige Ereignisse in die Multimenge hinein- bzw. hinausführen. Region r_1 ist außerdem minimal, da $\{s_\delta\}$ und $\{s'_\delta\}$ keine Regionen sind. Da eine weitere mit e beschriftete Transition $(s'_{\delta_2}, e, s_{\delta_2})$ existiert, kann die zweite Region mit $r_2 = \{s_{\delta_2}, s_\delta\}$ gewählt werden. Analog zur obigen Argumentation für r_1 erfüllt r_2 die Regionseigenschaft und ist minimal.

Lemma 4.3. ***(EC-Eigenschaft maximal gespaltener TS)*** *Sei $TS = (S, E, A, s_{in})$ ein maximal gespaltenes TS. Dann erfüllt TS die EC-Eigenschaft.* ⊣

Da der Anreizbereich jedes Ereignisses entweder selbst die Regionseigenschaft erfüllt oder durch Kombination von Regionen hergestellt werden kann, erfüllt das TS die EC-Eigenschaft und Lemma 4.3 gilt.

Satz 4.2. ***(Effektivität der Zustandsspaltung)*** *Sei $TS = (S, E, A, s_{in})$ ein Transitionssystem, das nicht k-EC ist. Dann gibt es eine endliche Sequenz von Zustandsspaltungsoperationen zur Sicherstellung der EC-Eigenschaft.* ⊣

Die Effektivität der Zustandsspaltung liegt in der Tatsache begründet, dass jede Spaltungsoperation in Richtung maximaler Spaltung steuert und spätestens in diesem Fall die EC-Eigenschaft gesichert ist. Da auf dem Weg dorthin die Menge von (minimalen) Regionen sukzessive erweitert wird (Satz 4.1), kann die EC-Eigenschaft schon zuvor gelten. Weitere Spaltungen sind dann nicht erforderlich.

Für die Auswahl geeigneter *ER*-Multimengen zur Regionstransformation können die in Abschnitt 4.4.1 beschriebenen Verfahren herangezogen werden.

4.5 GENET$^+$: Petrinetzsynthese auf Basis von Zustandsautomaten

Wie bereits in Abschnitt 4.3 demonstriert, lässt sich ein wohlgeformtes TS in ein PN transformieren, dessen Erreichbarkeitsgraph bisimilar zum TS ist und $\mathcal{L}(\mathsf{TS}) = \mathcal{L}^{\circ}(\mathsf{PN})$ gilt. Transitionssysteme, die in einer Bisimilaritätsrelation stehen, erzeugen dieselben Wörter bzw. Pfade. Hinsichtlich der Anforderungen in Abschnitt 3.1 stellt Bisimilarität deshalb zwar prinzipiell ein ausreichendes Qualitätsmerkmal dar, Garantien für die Sprachäquivalenz von Prozesslog und rekonstruiertem Modell existieren jedoch aufgrund der Verwendung von Transitionssystemen nur im präfixabgeschlossen Fall. Dieser Abschnitt stellt mit GENET$^+$ eine Erweiterung des GENET-Verfahrens von Carmona et al.[37] vor, das Sprachäquivalenz auch für NPC-Sprachen garantiert. Petrinetze werden dafür unter Beibehaltung der Bisimilaritätsgarantie auf Basis von Zustandsautomaten konstruiert und nicht wie zuvor auf Transitionssystemen. Abschnitt 4.5.1 thematisiert die Extraktion eines ZAs, der die Sprache des Prozesslogs exakt wiedergeben kann. Abschnitt 4.5.2 stellt die Konstruktion eines PN auf Basis des ZA dar. Für die Sprachdefinition des resultierenden PNS wird auf die vollständige Sequenzsemantik zurückgegriffen (siehe Def. B.10).

4.5.1 Extraktion eines ZAs auf Basis eines Prozesslogs

Die Extraktion eines ZAs aus einem gegebenen Prozesslog erfolgt unter Ausnutzung des Verfahrens von Aalst et al.[38] (vgl. Abschnitt 3.3). Die prinzipielle Vorgehensweise ist, jeden Ausführungspfad separat zu betrachten und basierend auf bisher observierten (*Präfix*) und folgenden (*Postfix*) Ereignissen innerhalb eines Ablaufs, Zustände zu definieren, die dann Eingang in das Ergebnis-TS finden. Bei der Zustandsbildung kann entweder ausschließlich der Präfix/Postfix oder eine Kombination aus beiden betrachtet werden. Dieses

[37] Vgl. Carmona, Cortadella und Kishinevsky 2010.
[38] Vgl. Aalst, Rubin et al. 2010.

Verfahren verfolgt durch Bereitstellung unterschiedlicher Abstraktionen das Ziel, nicht notwendigerweise ein präzises Abbild des Prozesslogs, sondern vielmehr eine vereinfachte Darstellung desselben zu generieren. Das entspricht der grundsätzlichen Ausrichtung von Process Mining, ein möglichst prägnantes Gesamtbild des Prozessverhaltens zu bieten. Insgesamt existieren folgende Abstraktionsmöglichkeiten:

1. **Horizont (P_h)**. Bestimmt die maximale Länge von Präfix/Postfix. Diese Abstraktion ist sehr ähnlich zu den Schwellwerten von `Ktail` und `Markov`, die ebenfalls zur Beschränkung der Betrachtung von vergangenem bzw. zukünftigem Verhalten verwendet werden. Vollständige Präfix-/Postfixbetrachtung wird mit $P_h = \infty$ assoziiert und liefert (sofern keine weiteren Abstraktionen angewandt werden) stets ein TS, welches alle Ausführungen gemäß Prozesslog wiedergeben kann. Wird nur der Präfix verwendet, hat dieses TS einen Startzustand und für jeden unterschiedlichen Pfad einen Endzustand; bei ausschließlicher Postfixbetrachtung tritt der umgekehrte Fall ein. Wird eine Kombination aus Präfix und Postfix gewählt, besteht das TS aus mehreren strikt sequentiellen Komponenten ohne Verbindung untereinander, die jeweils einen spezifischen Pfad symbolisieren.

2. **Filter (P_F)**. Sollten für eine Prozessbetrachtung nicht alle Ereignisse eines Prozesslogs von Interesse sein, kann eine Abstraktion in Form der Reduktion auf Ereignisse stattfinden, die sich auf bestimmte Prozessaktivitäten beziehen, konkret Aktivitäten in F. Der Horizont wird somit auf eine Teilmenge der TS-Ereignisse projiziert. Bei der Filterung von Ereignissen innerhalb eines Ablaufs bleibt die Ordnung der Sequenz erhalten. $P_F = \emptyset$ steht für einen nicht aktiven Filter.

3. **Filterhorizont (P_{F_h})**. Um gefilterte Ereignissequenzen weiter einzuschränken, kann ein Filterhorizont angegeben werden, der die Länge der Sequenz analog zu Abstraktion 1 beschränkt. Auch in diesem Fall impliziert ein Horizont $P_{F_h} = \infty$, dass die Ereignissequenz unangetastet bleibt.

4. **Datenstruktur (P_S)**. Diese Abstraktion gibt vor, in welcher Weise eine Ereignissequenz als Ergebnis der Anwendung obiger Abstraktionen interpretiert werden muss. Die Sequenz wird dabei entweder beibehalten (*sequence*), oder in eine Menge (*set*) oder eine Multimenge (*multi-set*) umgewandelt. Auf diese Weise findet eine weitere Abstraktion statt, bei der die Reihenfolge (*set*, *multi-set*) oder zusätzlich die Häufigkeit (*set*) von Prozessaktivitäten vernachlässigt wird, auf die sich Ereignisse beziehen.

Abb. 4.14 zeigt resultierende Transitionssysteme unter Berücksichtigung unterschiedlicher Abstraktionen auf Basis eines Prozesslogs, der die drei Aktivitätssequenzen $\langle B, C, D\rangle$, $\langle C, B, D\rangle$ und $\langle E, C, B\rangle$ enthält. Zwei Zustände werden als gleich aufgefasst, wenn sich die Sequenzen bisher observierter Ereignisse, die zu den jeweiligen Zuständen geführt haben, entsprechen. Ein Zustand ist durch die im Startzustand beginnende und in ihm endende Sequenz von Ereignissen eindeutig definiert.

Entsprechend der Zielsetzung dieser Arbeit muss für die Extraktion eines möglichst präzisen TS von jeglicher Abstraktion abgesehen werden. Konkret bedeutet das für die Parametrisierung der Methode, dass stets der gesamte Horizont, jedoch keine Filter betrachtet werden. Bzgl. der Datenstruktur kommt nur die *sequence*-Option in Frage. Da

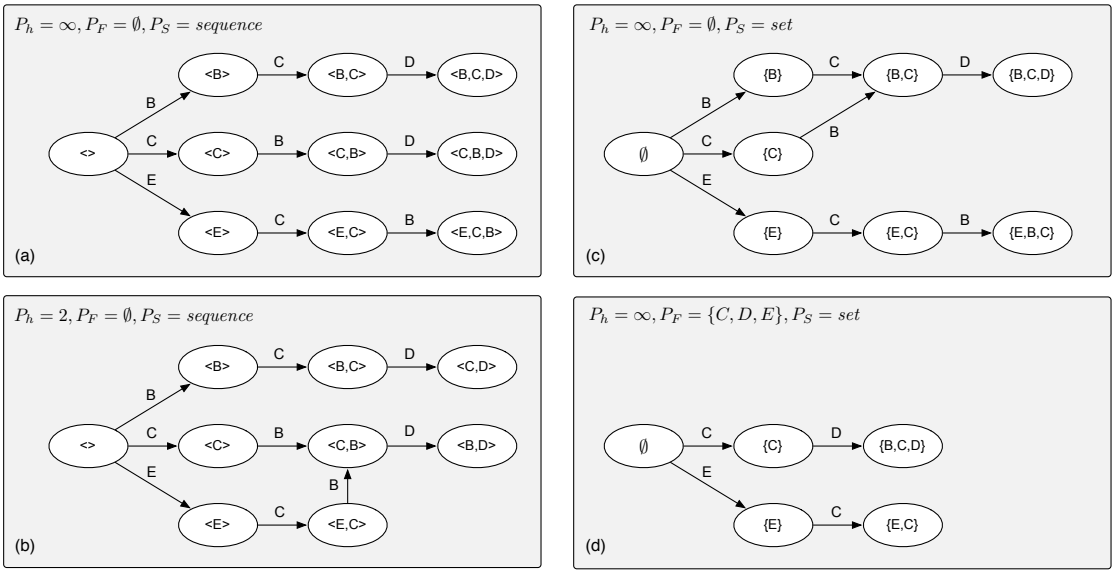


Abbildung 4.14.: Beispiele für die TS-Konstruktion nach Aalst et al.[39].

für die Anwendung von ZBR-Algorithmen ein TS mit eindeutigem Anfangszustand benötigt wird, muss eine Präfixstrategie verfolgt werden. Im Postfix-Fall würden sonst ggf. mehrere Startzustände entstehen. Eine Mischform ist ebenfalls nicht sinnvoll, da bei der Betrachtung des gesamten Horizonts, das TS nicht zusammenhängend wäre. Jede Zusammenhangskomponente würde dabei für einen unterschiedlichen Ablauf im Log stehen und das TS wiederum ggf. mehr als einen Startzustand beinhalten. Die Konstruktion eines TS unter Verwendung dieser Parametrisierung ($P_h = \infty, P_F = \emptyset, P_S = sequence$) wird im Folgenden als PTS-Konstruktion bezeichnet.

Lemma 4.4 (PTS). *Gegeben sei ein durch PTS-Konstruktion entstandenes Transitionssystem $TS = (S, E, A, s_{in})$. Dann ist TS wohlgeformt und es gilt folgendes Pfad-Axiom:*

(A5) $\forall s_1, s_2 \in S : s_{in} \xrightarrow{\sigma} s_1 \wedge s_{in} \xrightarrow{\sigma} s_2 \Rightarrow s_1 = s_2$ *(Pfad-Eindeutigkeit)* ⊣

Aufgrund der hinsichtlich vollständiger Sequenzen interpretierten Präfixbetrachtung und der verwendeten Äquivalenzeigenschaft für TS-Zustände existiert für jeden Zustand stets ein eindeutiger Pfad ausgehend vom Startzustand, der seine Erreichbarkeit sicherstellt. Angenommen, es existieren zwei Zustände $s', s'' \in S$, $s'_1 \neq s'_2$ mit $s_{in} \xrightarrow{\sigma} s'$ und $s_{in} \xrightarrow{\sigma} s''$, dann entsprechen sich die Sequenzen bisher observierter Ereignisse, die zu s' und s'' geführt haben. In diesem Fall wären s' und s'' während der Konstruktion als gleich aufgefasst worden, was der Annahme widerspricht und zugleich (A5) belegt.

Im Falle eines Nichtdeterminismus existiert ein Zustand $s \in S$ mit mehr als einer ausgehenden Kante desselben Bezeichners $e \in E$ mithilfe derer verschiedene Zustände erreicht werden können. O.b.d.A. sei σ der Pfad von s_{in} nach s und $s'_1, s'_2 \in S, s'_1 \neq s'_2$ zwei verschiedene Zustände die von s aus via e erreichbar sind, i.e. $s_{in} \xrightarrow{\sigma'} s'_1$ und $s_{in} \xrightarrow{\sigma'} s'_1$ mit $\sigma' = \sigma \cup e$. Damit sind s'_1 und s'_2 über denselben Pfad erreichbar, was (A5) wi-

derspricht und damit (A1) belegt. Existiert ein Zustand $s \in S$ mit Selbstbezug, i.e. $\exists e \in E : (s, e, s) \in A$, dann wäre s über zwei verschiedene Pfade erreichbar, was laut (A5) nicht möglich ist. Die Ereignismenge des TS entspricht der Menge gesehener Aktivitäten in Log und für jede Aktivität innerhalb einer Aktivitätssequenz wird bei der Konstruktion entweder eine Kante hinzugefügt oder es existiert bereits eine äquivalente Kante. Dadurch ist die Vorkommnis von Ereignissen (A3) sichergestellt. Da für jeden Zustand der Pfad ausgehend vom Startzustand auf Basis einer Aktivitätssequenz im Log gebildet und bei der Konstruktion der vollständige Präfix betrachtet wird, kann kein Zustand isoliert sein (A4). Ein TS, das durch PTS-Konstruktion entstanden ist, wird im Folgenden mit PTS abgekürzt. Der Vorgänger eines Zustandes s innerhalb eines PTS ist aufgrund von Lemma 4.4 eindeutig und wird mit $\overset{\curvearrowleft}{s}$ bezeichnet.

Lemma 4.5. *Gegeben sei ein PTS $T = (S, E, A, s_{in})$. Dann gilt: T ist ein gewurzelter, gerichteter, azyklischer Baum.* ⊣

Die Gültigkeit von Lemma 4.5 folgt direkt aus Lemma 4.4. Die Nichtexistenz von Zyklen innerhalb des TS kann auf naheliegende Weise durch die Verallgemeinerung der Argumentation hinsichtlich Zuständen mit Selbstbezug nachgewiesen werden. Innerhalb eines Baums kann es keinen Knoten mit mehr als einem direkten Vorgänger geben. Gäbe es einen solchen Knoten, wäre dieser über zwei verschiedene Pfade erreichbar, was durch (A5) ausgeschlossen ist. Die Tatsache, dass die ausschließliche Präfixbetrachtung stets in der Existenz eines eindeutigen Startzustands resultiert, vervollständigt die notwendigen Eigenschaften für Lemma 4.5.

Mithilfe der PTS-Konstruktion kann ein TS erzeugt werden, das alle Abläufe im Prozesslog generieren kann. Abb. 4.9 (a) enthält das auf diese Weise gewonnene PTS für den Rechnungsprüfungsprozess in Abb. 4.1. Um die Sprache des PTS auf die im Prozesslog enthaltenen Aktivitätssequenzen zu reduzieren, wird das PTS in einen ZA transformiert. Jeder Zustand, in dem eine Aktivitätssequenz des Prozesslogs endet, wird dabei gemäß Def. 4.14 als Endzustand markiert. Durch ZA-Transformation entstandene ZAs werden im Folgenden als PZAs bezeichnet.

Definition 4.14 (ZA-Transformation). *Gegeben sei ein Log L über dem Alphabet E und ein PTS (S, E, A, s_{in}). Die ZA-Transformation konstruiert einen PZA (S, E, A, s_{in}, F) mit*

$$F := \{s \in S \mid \exists \sigma : s_{in} \xrightarrow{\sigma} s \wedge \bar{\sigma} \in L\}$$ ⊣

4.5.2 Petrinetz-Konstruktion auf Basis eines ZAs

Die grundsätzliche Idee des vorgestellten Verfahrens ist, die Struktur eines PZAs so anzupassen, dass die vollständigen Schaltsequenzen des durch die Anwendung des Verfahrens von Carmona et al. gewonnenen Petrinetzes exakt den unterschiedlichen Aktivitätssequenzen im Prozesslog entsprechen. Die Anpassung erfolgt in zwei Schritten. Bei der *λ-Annotation* (Abschnitt 4.5.2) werden nach der Anwendung des Verfahrens von Carmona et al. zur Sicherstellung der EC-Eigenschaft weitere Zustände und Kanten in den

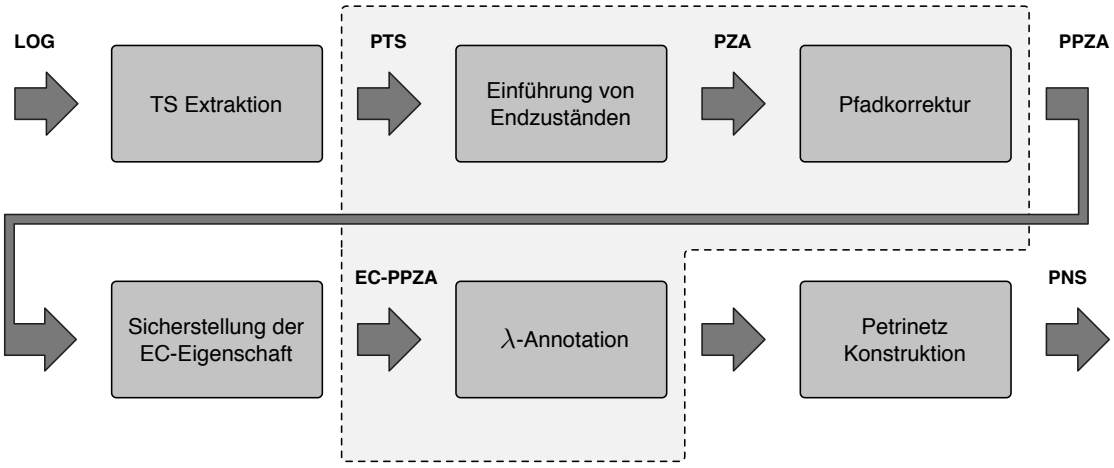


Abbildung 4.15.: Phasen des GENET$^+$-Ansatzes.

PZA eingefügt, die zu zusätzlichen Stellen und stillen Transitionen im Petrinetz führen (vgl. PNS), durch die die Ausführung von Prozessaktivitäten immer dann gestoppt werden kann, wenn die aktuelle Aktivitätssequenz einem Wort der Sprache des Prozesslogs entspricht. Unter der Voraussetzung, dass „innere“ Endzustände (keine Senken) im PZA mithilfe von Regionen isoliert werden können (Parikh-Eindeutigkeit), ist gesichert, dass durch die λ-Annotation die EC-Eigenschaft für keines der schon vor der Annotation vorhandenen Ereignisse verletzt und darüber hinaus auch für neu hinzugefügte Ereignisse gewährleistet wird.

Mithilfe einer *Pfadkorrektur* (Abschnitt 4.5.2) kann Parikh-Eindeutigkeit sichergestellt werden. Dafür werden Ereignisse im PTS (falls nötig) analog zur Ereignisspaltung (Abschnitt 4.4.1) umbenannt. Die Pfadkorrektur kann wie die λ-Annotation nachträglich erfolgen, doch da zur Sicherstellung der EC-Eigenschaft möglicherweise schon zuvor Ereignisspaltungen vorgenommen werden, ist es vorteilhaft, die Pfadkorrektur als Vorverarbeitungsschritt durchzuführen. Während bei der Ereignisspaltung zunächst Ereignisse ausgewählt werden, deren Umbenennung im Hinblick auf die EC-Eigenschaft sinnvoll erscheint, ist bei der Pfadkorrektur direkt klar, welche Ereignisse umbenannt werden *müssen*. Eine vorgelagerte Pfadkorrektur reduziert im Optimalfall die notwendigen Spaltungsoperationen im weiteren Verlauf und führt ansonsten (neben der Einführung zusätzlicher Transitionen im resultierenden PNS) zu keinem Nachteil. Abb. 4.15 gibt einen Überblick über die einzelnen Schritte und die Reihenfolge in der sie ausgeführt werden.

Schritt 1: Pfadkorrektur

Falls der Parikh-Vektor $\hat{\sigma}_s$ eines Pfades $\bar{\sigma}_s$, der zu einem inneren Endzustand s im ZA führt, nicht eindeutig ist, existiert ein weiterer Zustand s', der aus Sicht von Regionen nicht von s unterschieden werden kann. Möglicherweise kann dadurch die EC-Eigenschaft eines der in Schritt 2 eingefügten Ereignisse nicht sichergestellt und damit der Nachweis der Sprachäquivalenz (siehe Abschnitt 4.6.1) nicht erbracht werden. Dort wird für jeden inneren Zustand ein „eindeutiges“ Ereignis gewählt, um diesen Zustand mit einem neu eingeführten Endzustand zu verbinden. Ein Ereignis ist eindeutig, wenn es nur eine Kante

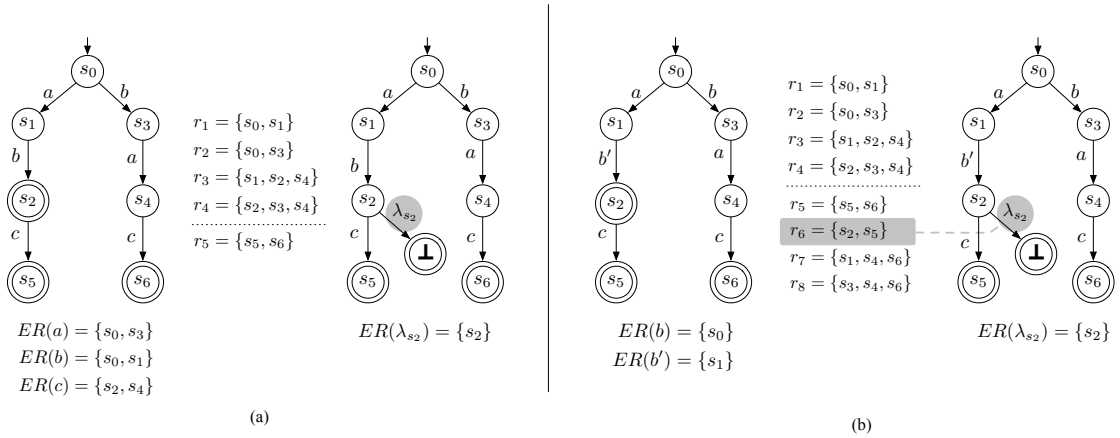


Abbildung 4.16.: (a) PZA mit zugehörigen minimalen Regionen und λ-annotierter Variante, (b) PZA aus (a) nach Pfadkorrektur mit zugehörigen minimalen Regionen und λ-annotierter Variante. Redundante Regionen finden sich in beiden Fällen unter der gestrichelten Linie.

gibt, die mit dem Ereignis beschriftet ist. Abb. 4.16 (a) verdeutlicht diese Situation mit dem inneren Endzustand s_2, dessen Parikh-Vektor $\hat{\sigma}_{s_1} = (1, 1, 0)$ dem Parikh-Vektor des Zustands s_4 entspricht (lexikalische Ordnung der Ereignisse). Der ZA erfüllt zwar die EC-Eigenschaft und ermöglicht deshalb die Herstellbarkeit der Anreizbereiche von Ereignissen mittels minimaler Regionen. Wird jedoch ein neues Ereignis λ_{s_2} benutzt, um s_2 mit dem Zustand $\perp$ zu verbinden, ist die EC-Eigenschaft für λ_{s_2} nicht erfüllt, da ${}^\circ\lambda_{s_2} = \{r_3, r_4\}$ und $supp(r_3) \cap supp(r_4) = \{s_2, s_4\} \neq ER(\lambda_{s_2})$.

Definition 4.15 (Eindeutigkeit von Parikh-Vektoren). *Innerhalb eines PZA (S, E, A, s_{in}, F) sind Parikh-Vektoren eindeutig, wenn gilt:*

(A6) $\forall s_1, s_2 \in S : \hat{\sigma}_{s_1} = \hat{\sigma}_{s_2} \Rightarrow s_1 = s_2$.

$S^\diamond \subseteq S$ bezeichnet die Menge der Zustände ohne eindeutigen Parikh-Vektor. ⊣

Durch eine Umbenennung von Ereignis b gemäß der Ereignisspaltung, kann der Parikh-Vektor von s_2 eindeutig gemacht werden (siehe Abb. 4.16 (b)). Die EC-Eigenschaft der übrigen Ereignisse wird durch solche Umbenennungen nicht verletzt und die Menge minimaler Regionen höchstens vergrößert[40]. In jedem Fall muss eine Neuberechnung der minimalen Regionen stattfinden. Für die Sicherstellung der EC-Eigenschaft des zusätzlichen Ereignisses λ_{s_2} wird eine der neu hinzugekommenen redundanten minimalen Regionen (r_6) benötigt. Die Sicherstellung der Eindeutigkeit eines Parikh-Vektors kann eine Reihe von Umbenennungsschritten auf dem Weg vom betreffenden Zustand hin zum Startzustand erfordern. Alg. 4.3 beschreibt den Vorgang der Pfadkorrektur für einen gegebenen PZA und eine Menge von Zuständen, für die Eindeutigkeit von Parikh-Vektoren gewünscht ist. Dabei wird der neue, eindeutige Bezeichner, der für eine Kante im Zuge der Ereignisspaltung gewählt wird, mit der Funktion α angegeben. Die Eindeutigkeit von Parikh-Vektoren ist nur für innere Endzustände erforderlich, da während der λ-Annotation nur an diesen Stellen zusätzliche Ereignisse angefügt werden. Im Hinblick

[40] Vgl. Carmona 2012.

Algorithmus 4.3 Pfadkorrektur

Eingabe: $\mathsf{PZA} = (S, E, A, s_{in}, F)$

```
1: procedure PFADKORREKTUR(PZA)
2:     while S^◇ ∩ F̄ ≠ ∅ do
3:         Wähle s ∈ S^◇ ∩ F̄ mit Pfad σ̄_s = ⟨e_1, ..., e_k⟩
4:         E = E ∪ {α(e_k)}
5:         A = A \ {(↶s, e_k, s)} ∪ {(↶s, α(e_k), s)}
6:     end while
7: end procedure
```

auf die Implementierung der Pfadkorrektur erscheint es sinnvoll, die betreffenden Zustände mithilfe einer Tiefensuche zu identifizieren, da die Pfadkorrektur eines Zustandes u.U. auch die Eindeutigkeit der Parikh-Vektoren von Nachfolgerknoten im PZA sicherstellt.

Ein PZA mit eindeutigen Parikh-Vektoren wird im Folgenden als PPZA bezeichnet; EC-PPZA wenn zusätzlich die EC-Eigenschaft erfüllt ist.

Schritt 2: λ-Annotation

Um die Voraussetzung dafür zu schaffen, dass im resultierenden Petrinetz das Schalten von Transitionen an geeigneten Stellen gestoppt werden kann, wird jeder Endzustand s, der keine Senke ist, mittels eines eindeutigen Ereignisses λ_s mit einem zusätzlich eingeführten Endzustand ($\perp$) verbunden und selbst nicht mehr als Endzustand aufgefasst. Das PN wird dadurch in einen Zustand überführt, in dem keine weiteren Transitionen mehr geschaltet werden können. Jedes λ-Ereignis führt zu einer stillen Transition, die im Erreichbarkeitsgraphen des PN als Kante in Erscheinung tritt, die von einem inneren Zustand zu einer Senke führt und damit die Ausführung beendet. Die Transformation wird in Def. 4.16 formal beschrieben und als λ-Annotation bezeichnet.

Definition 4.16 (λ-Annotation). *Gegeben sei ein EC-PPZA (S, E, A, s_{in}, F). Seien weiter $S_\circ := \{s \in S \mid \nexists (s, e, s') \in A\}$ dessen Senken und $s_\perp$ der neu hinzugefügte Endzustand. Für jeden Zustand, der eine Verbindung zu $s_\perp$ erhält, wird das zugehörige Ereignis mit λ_s bezeichnet. Die λ-Annotation konstruiert ein EC-PPZA (S', E', A', s_{in}, F') mit*

$$
\begin{aligned}
E' &:= E \cup \{\lambda_s \mid \exists s \in \bar{F}\} \\
S' &:= S \cup \{s_\perp\} \\
A' &:= A \cup \{(s, \lambda_s, s_\perp) \mid s \in \bar{F}\} \\
F' &:= \bar{F} \cup \{s_\perp\}
\end{aligned}
$$

⊣

Abb. 4.17 zeigt den EC-PPZA als Resultat der Anwendung des vorgestellten Verfahrens ausgehend von Prozesslog $\mathfrak{L}_1$ des Rechnungsprüfungsprozesses. Die bei der

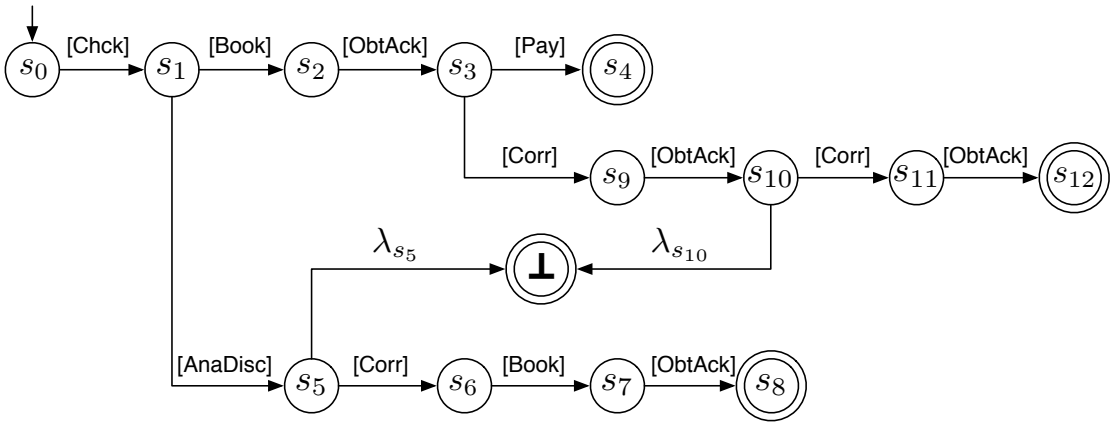


Abbildung 4.17.: EC-PPZA nach λ-Annotation auf Basis des Prozesslogs $\mathfrak{L}_1$. Endzustände sind mit doppelter Umrandung markiert.

PTS-Konstruktion eingeführten Endzustände $s_4, s_5, s_8, s_{10}, s_{12}$ haben eindeutige Parikh-Vektoren; die Pfadkorrektur hat also keinen Effekt. Der entsprechende PPZA ist 2-EC, was keine Korrekturmaßnahmen wie Ereignisspaltung erforderlich macht. Die anschließende λ-Annotation verbindet die Zustände s_5 und s_{10} mit einem neuen Zustand $\perp$, da die zugehörigen Abläufe im Log (Ablauf 1 und Ablauf 4) in einem inneren Zustand enden. Die im Schritt zur Sicherstellung der EC-Eigenschaft generierte Regionsmenge wird durch die Annotation um eine Region, die lediglich aus dem Zustand $s_\perp$ besteht, erweitert. Bestehende Regionen werden nicht beeinträchtigt. Auch die Menge nichtredundanter Regionen wird durch die Annotation nicht verändert. Die Region $s_\perp$ ist immer redundant, da sie für kein Ereignis eine Vorgängerregion sein kann und deshalb auch nichts zur EC-Eigenschaft eines Ereignisses beitragen kann. Formale Nachweise für diese Feststellung und die Korrektheit des Annotationsverfahrens allgemein können Abschnitt 4.6.1 entnommen werden. Abschnitt 4.6.4 diskutiert den Unterscheid resultierender Netze auf Basis des bestehenden Verfahrens von Carmona et al. (Abschnitt 4.3) zu Netzen auf Basis der vorgestellten Erweiterung mithilfe einer Reihe von Experimenten. Abhängig davon, ob eine Pfadkorrektur Anwendung gefunden hat, unterscheiden sich die Netze erheblich.

4.6 Evaluation

Dieses Kapitel bewertet den entwickelten Ansatz zur Rekonstruktion präziser Modelle. Während Abschnitt 4.6.1 auf formaler Ebene Nachweise für die Korrektheit von GENET$^+$ zur Kontrollflussrekonstruktion liefert, enthalten folgende Abschnitte Angaben zur Komplexität des GENET$^+$-Verfahrens (Abschnitt 4.6.2) und Experimente, die anhand synthetischer Prozesslogs die Wirkungsweise von GENET$^+$ untersuchen (Abschnitt 4.6.4). Dabei werden die beiden Verfahren zur Sicherstellung der EC-Eigenschaft (Ereignisspaltung und Zustandsspaltung) verglichen.

4.6.1 Formale Korrektheit des GENET$^+$-Verfahrens

Im Folgenden wird bewiesen, dass der Erreichbarkeitsgraph eines PN, welches auf Basis eines unter Ausnutzung von Def. 4.16 gewonnenen EC-PPZAs konstruiert wird, bisimilar zu diesem EC-PPZA ist und deren Sprache äquivalent bzgl. vollständiger Sequenzsemantik ist, i.e. $\mathcal{L}^\bullet(\mathsf{PN}) = \mathcal{L}(\mathsf{EC\text{-}PPZA})$. Um die Bisimilarität nachzuweisen, wird gezeigt, dass die EC-Eigenschaft durch die λ-Annotation nicht verletzt wird. Dazu werden die folgenden Anforderungen nacheinander betrachtet:

(F1) Die λ-Annotation verletzt nicht die EC-Eigenschaft bereits vorhandener Ereignisse.

(F2) Die λ-Annotation garantiert die EC-Eigenschaft neu hinzugefügter Ereignisse.

(F3) Abgeschlossene Schaltsequenzen entsprechen Aktivitätssequenzen im Prozesslog.

Anforderung (F1) und (F2) beziehen sich auf die Neutralität der λ-Annotation hinsichtlich der EC-Eigenschaft. Ist diese gewährleistet, so ist der Erreichbarkeitsgraph des resultierenden Petrinetzes nach der Annotation weiterhin bisimilar zum Eingabe-EC-PPZA und es gilt $\mathcal{L}^\bullet(\mathsf{PN}) \subseteq \mathcal{L}(\mathsf{EC\text{-}PPZA})$. Mit dem Nachweis von (F3) kann dann die Sprachäquivalenz im Sinne vollständiger Sequenzsemantik nachgewiesen werden.

EC-Neutralität der λ-Annotation

Für den Nachweis, dass die λ-Annotation nicht die EC-Eigenschaft beeinträchtigt, wird zunächst auf (F1) eingegangen. Die λ-Annotation führt eine Menge eindeutiger Ereignisse und einen zusätzlichen Endzustand $s_\perp$ ein. Jede Multimenge, die vor der Annotation die Regionseigenschaft erfüllt hat, tut dies weiterhin, denn aufgrund ihrer Eindeutigkeit haben neu eingeführte Ereignisse konstanten Gradient in jeder Region (es gibt stets nur eine Kante, die mit dem Ereignis gekennzeichnet ist). Die Multimenge $\{s_\perp\}$ selbst ist eine minimale Region, da sie ausschließlich mit neu eingeführten eindeutigen Ereignissen verbunden ist und keine weitere Region enthalten kann. Da sich die Menge minimaler Regionen nicht verändert, sondern lediglich vergrößert wird und $\{s_\perp\}$ keine Berührung mit bereits vorhandenen Ereignissen hat, bleibt die EC-Eigenschaft aller vor der λ-annotation vorhandenen Ereignisse bestehen.

Um zu beweisen, dass die EC-Eigenschaft für alle neu eingeführten Ereignisse gewährleistet werden kann, wird der Zusammenhang zwischen Regionen und Parikh-Vektoren in Transitionssystemen (siehe Abschnitt B) genutzt. Ausgehend von einem Transitionssystem $TS = (S, E, A, s_{in})$, lässt sich die Multiplizität eines Zustands s innerhalb einer Region r bzgl. TS als Kombination des Parikh-Vektors eines zu s führenden Pfades σ mit dem Gradientenvektor Δ_r darstellen: $r(s) = r(s_{in}) + \hat{\sigma}_s * \Delta_r^T$[41]. Für die Angabe von Parikh-Vektoren muss stets dieselbe Sortierung von Ereignissen verwendet werden

[41] Vgl. Solé et al. 2013.

(typischerweise lexikalisch). Für den Zustand s_5 in Abb. 4.6 und die zu s_5 führende Aktivitätssequenz $\bar{\sigma} = \langle a, a, b \rangle$ gilt unter Verwendung der Ereignisreihenfolge $[a, b, c, d]$:

$$\begin{aligned} r_3(s_5) &= r_3(s_0) + \hat{\sigma} * \Delta_{r_3} \\ &= 0 + (2, 1, 0, 0) * (1, -1, 1, -1)^T \\ &= 0 + 2 - 1 \\ &= 1 \end{aligned}$$

Ereignis a tritt innerhalb von $\hat{\sigma}$ zweimal auf, Ereignis b einmal, während c und d nicht auftreten. Die Gradienten ergeben sich zu $(1, -1, 1, -1)$. Es ist unerheblich, welche Aktivitätssequenz für die Berechnung der Multiplizität gewählt wird. Die Verwendung der Sequenz $\bar{\sigma}' = \langle c, d, c \rangle$, die auch zu s_5 führt, ergibt ebenfalls Multiplizität 1, da $\hat{\sigma}' * \Delta_{r_3} = (0, 0, 2, 1) * (1, -1, 1, -1)^T = (2, 1, 0, 0) * (1, -1, 1, -1)^T = \hat{\sigma} * \Delta_{r_3}$.

Innerhalb eines PTS, bei dem jeder Zustand $s \in S$ über einen eindeutigen Pfad σ_s erreichbar ist, existiert genau ein (im PTS nicht notwendigerweise eindeutiger) Parikh-Vektor $\hat{\sigma}_s$. Es kann weitere Zustände geben, deren Parikh-Vektor mit $\hat{\sigma}_s$ übereinstimmt. In diesem Fall können diese Zustände aus Sicht von Regionen nicht unterschieden werden. Ist der Parikh-Vektor eines Zustands jedoch eindeutig, kann er mithilfe einer Menge von Regionen von allen anderen Zuständen separiert werden.

Definition 4.17 (Zustands-Topset). *Sei s ein Zustand eines Transitionssystems und $r \in R_s$ eine Region, die s enthält. Das Zustands-Topset von r bezüglich s ($\boldsymbol{ST}(r, s)$) ist die Multimenge q für die hinsichtlich eines Grades $g \in \mathbb{N}^+$ gilt:*

1. $q = T_g(r)$
2. $s \in T_g(r)$
3. $s \notin T_{g+1}(r)$ ⊣

Die Menge der Zustands-Topsets aller Regionen, die einen Zustand s enthalten, ist definiert als $s^* := \{q \mid \exists r \in R_s : q = \mathbf{ST}(r, s)\}$. Das Separierbarkeits-Lemma besagt, dass innerhalb jedes Transitionssystems stets genügend Regionen existieren, um einen Zustand mit eindeutigem Parikh-Vektor durch Kombination von Zustands-Topsets zu isolieren.

Lemma 4.6. ***(Separierbarkeit von Zuständen)*** *Sei $TS = (S, E, A, s_{in})$ ein Transitionssystem und $s \in S$ ein Zustand mit eindeutigem Parikh-Vektor. Dann gilt:*

$$\bigcap_{q \in s^*} supp(q) = \{s\}$$ ⊣

Die Gültigkeit dieses Lemmas wird informal mit der Aussage, dass „Zustände mithilfe von Regionen von allen anderen Zuständen separiert werden können“ umschrieben. Das Separierbarkeits-Lemma kann durch Widerspruch bewiesen werden. Seien s_1, s_2 zwei Zustände innerhalb eines Transitionssystems $TS = (S, E, A, s_{in})$ mit eindeutigen Parikh-Vektoren, die nicht anhand von Regionen separiert werden können. Für jedes Ereignis $e \in A$ und jeden Zustand $s \in S$ mit $s_{in} \xrightarrow{\sigma} s$ wird die Multimenge $r_e(s) = \hat{\sigma}(e)$ betrachtet.

Diese Definition ist gültig, da aufgrund der Eindeutigkeit von Parikh-Vektoren auch $\hat{\sigma}(e)$ eindeutig bestimmt ist. Außerdem erfüllt $r_e(s)$ mit $\Delta_{r_e}(e) = 1$ und $\Delta_{r_e}(a) = 0 \forall a \neq e$ die Regionseigenschaft[42]. Da $s_1 \neq s_2$, existieren zwei Sequenzen σ_1, σ_2 mit $s_{in} \xrightarrow{\sigma_1} s_1$ und $s_{in} \xrightarrow{\sigma_2} s_2$, sowie mind. ein Ereignis $x \in A$, sodass $\hat{\sigma}_1 \neq \hat{\sigma}_2$. Die Region R_x weist also unterschiedliche Multiplizitäten für die Zustände s_1 und s_2 auf, was in Konflikt zur Annahme steht, dass diese Zustände nicht separiert werden können.

Die für den Beweis erforderlichen kanonischen Regionen erfordern einen Mindestgrad $deg_{min} := max_{s \in S} max_{r \in s^*}(deg(r))$. Wird für das resultierende Petrinetz eine Schranke k angegeben, muss im Falle von $deg_{min} > k$ eine Ereignis- bzw. Zustandsspaltung vorgenommen werden, um die Existenz entsprechender k-beschränkter Regionen sicherzustellen. Dies ist in jedem Fall möglich, da durch Spaltungsoperationen der Grad von Regionen gesenkt werden kann. Im Falle vollständiger Zustandsspaltung sind bspw. alle Kantenmarkierungen eindeutig und jede Menge, die lediglich einen Zustand enthält, erfüllt die Regionseigenschaft und ist minimal.

Darüber hinaus scheint es einen starken Zusammenhang zwischen der EC-Eigenschaft und der Separierbarkeit von Zuständen zu geben. Die Vermutung liegt nahe, dass innerhalb eines TSs welches die EC-Eigenschaft erfüllt, stets die für die Separierbarkeit von Zuständen erforderlichen Regionen existieren. Dennoch wird bei der Implementierung des Verfahrens nach der λ-Annotation geprüft, ob die Separierbarkeit gegeben ist und, falls nicht, weitere Spaltungen vorgenommen. Allerdings war dies innerhalb bisheriger Experimente nie notwendig.

Das Separierbarkeits-Lemma kann genutzt werden, um zu beweisen, dass die EC-Eigenschaft auch für Ereignisse und Zustände gilt, die während der λ-Annotation hinzugefügt wurden.

Lemma 4.7. ***(EC-Eigenschaft von λ-Ereignissen)*** *Sei $T = (S, E, A, s_{in})$ ein EC-PPZA und R die zugehörige Menge minimaler Regionen. Sei weiter λ ein bei der λ-Annotation neu hinzugefügtes Ereignis, das von $s_\lambda \in S$ zur Senke $s_\perp$ führt. Dann enthält R genügend Regionen, um auch die EC-Eigenschaft für λ sicherzustellen.* ⊣

Für jedes solche Ereignis λ besteht der Anreizbereich aus genau einem Zustand, der bereits vor der Annotation Bestandteil des EC-PPZA war: $ER(\lambda) = \{s_\lambda\}$. Gemäß Lemma 4.6 und der Eindeutigkeit von Parikh-Vektoren, die durch Pfadkorrektur erreicht wird, kann dieser Zustand mithilfe von Regionen in R von allen übrigen Zuständen separiert werden. Wenn die Separierbarkeit gegeben ist, existieren auch Vorgängerregionen für λ. Diese beiden Folgerungen stellen die notwendigen Anforderung der EC-Eigenschaft (siehe Def. 4.10) sicher.

Satz 4.3. ***(EC-Neutralität der λ-Annotation)*** *Sei T ein EC-PPZA und R die Menge zugehöriger minimaler Regionen. Sei weiter T' das Resultat der λ-Annotation von T. Dann garantiert R auch die EC-Eigenschaft von T'.* ⊣

Gemäß der Argumentation zu Beginn des Abschnitts ist klar, dass die EC-Eigenschaft für bereits vorhandene Ereignisse weiterhin gilt. Für jedes neu hinzugefügte Ereignis kann Lemma 4.7 angewandt werden. Folglich erfüllt R auch die EC-Eigenschaft für T'.

[42] Vgl. Solé et al. 2010a.

Zwar wird die EC-Eigenschaft durch die λ-Annotation nicht verletzt, doch hat sie Einfluss auf die Redundanz minimaler Regionen. Das bedeutet nicht, dass Multimengen, die zuvor Regionen waren, jetzt keine mehr sind, dennoch wird durch die Annotation die Struktur des zugrundeliegenden EC-PPZAs verändert, sodass die Zahl redundanter minimaler Regionen ggf. reduziert wird. Im Beispiel in Abb. 4.16 wird nach der Annotation eine der redundanten minimalen Region (grau hinterlegt) benötigt, um die EC-Eigenschaft für Ereignis λ_{s_2} zu gewährleisten, da ${}^\circ\lambda_{s_2} = \{r_3, r_4\}$ und $supp(r_3) \cap supp(r_4) = \{s_2, s_4\} \neq \{s_2\}$ aber $\{s_2, s_4\} \cap \{s_2, s_5\} = \{s_2\}$. Region r_6 ist insofern *essentiell* für λ_{s_2} und damit nicht redundant. Die Reduktion redundanter Regionen verursacht die Einführung zusätzlicher Stellen im resultierenden Petrinetz und kann deshalb (zusammen mit den zusätzlichen stillen Transitionen) als „Preis“ für die Erreichung von Sprachäquivalenz angesehen werden. Nicht in jedem Fall sind weitere Stellen notwendig. Dies hängt gänzlich davon ab, ob Anreizbereiche von λ-Ereignissen mithilfe nichtredundanter minimaler Regionen hergestellt werden können oder nicht.

Vollständigkeit abgeschlossener Aktivitätssequenzen

Der Nachweis von (F3) ergibt sich direkt aus der Betrachtung des Erreichbarkeitsgraphen des resultierenden Petrinetzes und der folgenden Bedingungen:

1. Jede abgeschlossene Schaltsequenz entspricht einer Aktivitätssequenz.
2. Es gibt keine Aktivitätssequenz ohne Bezug zu einer abgeschlossenen Schaltsequenz.

Für Bedingung 1 ist erforderlich, dass alle abgeschlossenen Schaltsequenzen des Petrinetzes in Zuständen enden, deren Entsprechung im annotierten EC-PPZA Endzustände sind, da diese Zustände keine ausgehenden Kanten haben (es kann also keine weitere Transition geschaltet werden) und in ihnen endende Sequenzen einen direkten Bezug zu Aktivitätssequenzen im Prozesslog aufweisen (siehe Def. 4.14). Aufgrund der Bisimilarität zwischen dem Erreichbarkeitsgraphen und dem annotierten EC-PPZA, generieren beide dieselbe Menge von Sequenzen. Angenommen, es gäbe im Petrinetz eine abgeschlossene Schaltsequenz, die in einem Zustand endet, dessen Entsprechung im annotierten EC-PPZA kein Endzustand ist. Dieser Zustand ist eine Senke, die aufgrund der Bisimilarität eine Entsprechung im EC-PPZA finden muss. Da jede Senke im EC-PPZA jedoch eine direkte Verbindung zu einer Aktivitätssequenz hat, kann es keine solche Schaltsequenz geben.

Bedingung 2 fordert, dass keine Aktivitätssequenz ausgelassen wird. Für jede Aktivitätssequenz gibt es im EC-PPZA einen Zustand, bei dessen Erreichen die Sequenz generiert wird. Jeder solche Zustand ist entweder eine Senke oder wird mithilfe eines λ-Ereignisses mit einer Senke verbunden. Da der Erreichbarkeitsgraph und der EC-PPZA dieselben Sequenzen generieren, gibt es im PN eine Schaltsequenz, die der Aktivitätssequenz entspricht. Falls die Schaltsequenz in einer Senke endet ist sie ohnehin abgeschlossen, falls nicht, kann eine Senke durch das Schalten einer stillen Transition erreicht werden. Dies ist aufgrund der Existenz des λ-Ereignisses im EC-PPZA und der Bisimilarität gewährleistet.

4.6.2 Berechnungskomplexität

Die Berechnungskomplexität wird maßgeblich von der gewählten Schranke k beeinflusst. Bei $k = 1$ enthält der potentielle Suchraum für Regionen $2^{|S|}$ Elemente, da jeder Zustand entweder Teil einer Region sein kann oder nicht. Allgemein ergibt sich die Größe des Suchraums für ein gegebenes k zu $(k+1)^{|S|}$. Obwohl dieser Raum nicht notwendigerweise vollständig exploriert wird (tatsächlich meist nur ein Bruchteil dessen), konnten mit der zur Verfügung stehenden Hardware nicht genügend Experimente für einen aussagekräftigen Vergleich zwischen Ergebnissen verschiedener Werte für k generiert werden. Die Problematik besteht darin, dass nicht im Voraus bestimmt werden kann, welche Erhöhung von k erforderlich ist, um eine Komplexitätsreduktion im resultierenden Petrinetz zu erreichen.

Die Gesamtkomplexität des Verfahrens wird durch die Anzahl der Zustände im Ausgangs-Transitionssystem $|S|$ bestimmt, die wiederum von der Zahl unterschiedlicher Ausführungspfade abhängig ist. Im ungünstigsten Fall sind alle n Ausführungspfade eines Prozesslogs verschieden. Die Größe des daraus extrahierten Transitionssystems ist maximal, wenn keine Ausführungspfade mit gemeinsamen Präfixen existieren. Auf diese Weise entsteht für jeden Ausführungspfad eine sequentielle Verkettung von Zuständen, die beim Startzustand beginnt und ansonsten keine Verzweigungen enthält. Bei einer durchschnittlichen Länge der Ausführungspfade im Prozesslog von r, ergibt sich damit die Zahl der Zustände im Transitionssystem zu $|S| = n * r + 1$ und die Zahl der Kanten zu $|A| = n * r$. Der Suchraum für die Bestimmung minimaler Regionen entspricht dann $(k+1)^{n*r+1}$.

Für die Sicherstellung der EC-Eigenschaft müssen möglicherweise Spaltungen im Transitionssystem vorgenommen werden. Im ungünstigsten Fall wird diese Eigenschaft erst bei maximaler Spaltung erreicht. Unter Verwendung von Ereignisspaltung und der worst-case Annahme, dass bei jeder Spaltung stets genau eine Umbenennung erfolgt, sind insgesamt $|A| - |E|$ Spaltungen notwendig. Hinzu kommt der Aufwand für die Auswahl einer geeigneten Multimenge für die Spaltung. Dafür muss bei der deckungsorientierten Spaltung jede Multimenge, die nicht die Regionseigenschaft erfüllt, betrachtet werden. Für die Abschätzung des Auswahlaufwands wird ein Faktor $\epsilon \in [0; 1]$ verwendet, der den Anteil der explorierten Multimengen mit nicht geltender Regionseigenschaft bestimmt. Für die Auswahl fällt pro Multimenge entweder konstanter Aufwand an (Auswahl der Multimenge mit höchster Anzahl von Ereignissen mit konstantem Gradient), oder es muss für jede Multimenge zunächst deren Beitrag zur Deckung des Anreizbereichs von Ereignissen bestimmt werden. Dafür muss jedes Ereignis betrachtet werden, dessen Anreizbereich in der Multimenge enthalten ist. Die Anzahl der Ereignisse verändert sich mit jeder Spaltungsoperation, im Durchschnitt existieren bei einer Spaltung $\frac{|E|+|A|}{2}$ Ereignisse. Wie viele Ereignisse pro Multimenge betrachtet werden müssen, ist schwer abschätzbar. Für die Komplexitätsbetrachtung wird vereinfachend angenommen, dass für jede Multimenge die Hälfte der Ereignisse betrachtet werden muss. Hinsichtlich bisher durchgeführter Experimente scheint diese Annahme plausibel. Die obere Schranke für den Gesamtaufwand ergibt sich bei Ereignisspaltung also zu:

$$(|A| - |E|) * \left[(k+1)^{|S|} + \epsilon * (k+1)^{|S|} * \frac{|E| + |A|}{4} \right]$$

$$\Leftrightarrow (n * r - |E|) * (k+1)^{n*r+1} * \left[1 + \frac{\epsilon * (|E| + n * r)}{4}\right]$$

Damit liegt der Gesamtaufwand für die Rekonstruktion eines präzisen Petrinetzes mithilfe des GENET$^+$-Verfahrens und unter Verwendung von Ereignisspaltung in der Komplexitätsklasse $O(n * k^n)$.

Im Fall von Zustandsspaltung verändert sich die Größe des Transitionssystems bei jeder Spaltung. Bei maximaler Spaltung sind alle regulären Ereignisse durch δ-Ereignisse voneinander getrennt. Unter Beibehaltung der obigen worst-case Annahme für die TS-Struktur entstehen dadurch $n*(2*r-1)$ zusätzliche Kanten und eben so viele zusätzliche δ-Ereignisse und Zustände. Die Anzahl von Kanten, die das Transitionssystem vor einer Spaltung enthält wird mit $|A|+n*(r-\frac{1}{2})$ abgeschätzt. Das entspricht der durchschnittlich zu erwartenden Anzahl von Kanten, gemessen am Ursprungszustand und der Anzahl von Kanten im maximal gespaltenen Zustand (arithmetisches Mittel). Die Abschätzung der Anzahl von Ereignissen und Zuständen erfolgt analog. Die Anzahl benötigter Spaltungen ist maximal (entspricht $n * (2 * r - 1)$), wenn bei jeder Spaltung nur auf einer Kante eine Spaltungsoperation (Quell- oder Zielspaltung) angewandt wird. In diesem Fall sind genau $n*(2*r-1)$ Spaltungen erforderlich. Der Aufwand pro Spaltung ist konstant und der Gesamtaufwand für die Durchführung des GENET$^+$-Verfahrens ergibt sich zu:

$$n * (2 * r - 1) * \left[(k+1)^{|S|+n*(r-\frac{1}{2})} + \epsilon * (k+1)^{|S|+n*(r-\frac{1}{2})} * \frac{|E| + n * (r - \frac{1}{2})}{2}\right]$$

$$n * (2 * r - 1) * (k+1)^{2*n*(r-\frac{1}{4})+1} * \left[1 + \frac{\epsilon * \left(|E| + n * (r - \frac{1}{2})\right)}{2}\right]$$

Aufgrund der durch Zustandsspaltung hervorgerufenen Vergrößerung des Transitionssystems erhöht sich die obere Schranke für den Gesamtaufwand unter Verwendung von Zustandsspaltung. Allerdings wird die maximale Anzahl benötigter Spaltungen enorm selten benötigt. Typischerweise werden im Rahmen einer Spaltung mehrere Kanten gespalten, wodurch die EC-Eigenschaft deutlich schneller erreicht wird. Die teils deutlich kürzeren Laufzeiten des GENET$^+$-Verfahrens bei Zustandsspaltung in Abschnitt 4.6.4 unterstützen diese Behauptung.

Der exponentielle Berechnungsaufwand des GENET$^+$-Verfahrens wird auch in den im Folgenden diskutierten Experimenten deutlich. Mit steigender Komplexität verwendeter Prozesslogs (Anzahl unterschiedlicher Ausführungspfade und Anzahl unterschiedlicher Aktivitäten), steigt die benötigte Zeit für die Durchführung des Verfahrens stark an. Die allgemein hohe Komplexität des Verfahrens ist typisch für regionsbasierte Verfahren und nicht auf die spezifischen Eigenschaften des GENET$^+$-Verfahrens zurückzuführen. Tatsächlich erfordert die erweiterte Vorgehensweise im Vergleich zu GENET keinen signifikanten Mehraufwand (siehe Erläuterungen zu durchgeführten Experimenten). Trotz ihrer hohen Komplexität gibt es zur Rekonstruktion von präzisen Kontrollflussmodellen derzeit keine Alternative zu regionsbasierten Verfahren.

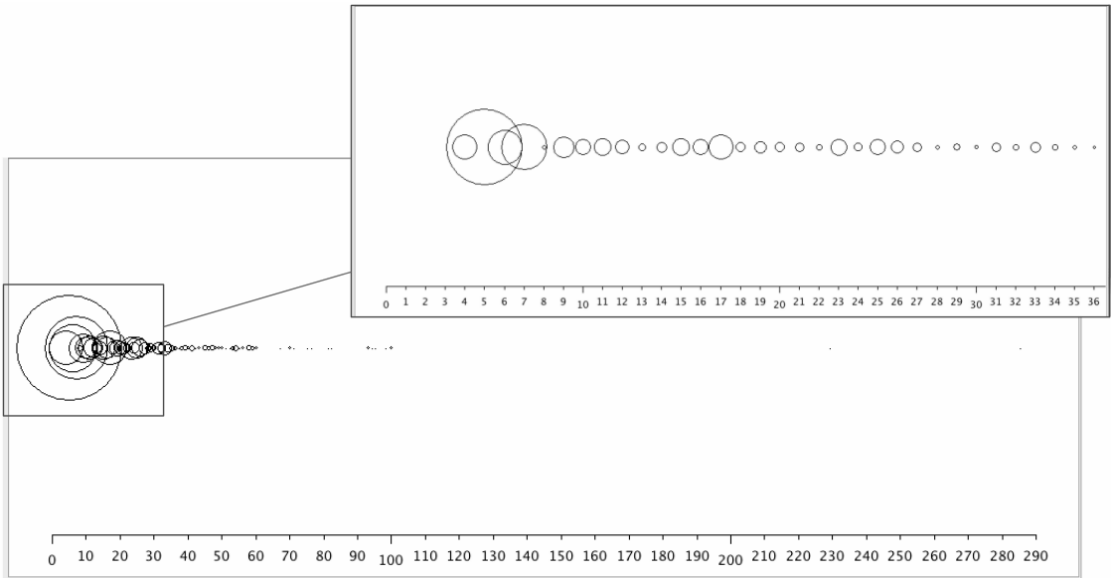


Abbildung 4.18.: Komplexität von Standardprozessen innerhalb der IBM-Websphere Bibliothek (Anzahl der Aktivitäten).

4.6.3 Ausgangsbasis für durchgeführte Experimente

Zur Generierung von Prozesslogs als Basis für Experimente zur Wirkungsweise des entwickelten Ansatzes wurden beispielhafte Prozessmodelle herangezogen, die in ihrer Gesamtheit ein breites Spektrum typischer struktureller Charakteristika abdecken. Dazu zählen Parallelität, die Existenz sich gegenseitig ausschließender Teilpfade, Schleifen, unsichtbare Aktivitäten und Duplikate. Details zu verwendeten Modellen können Abschnitt D entnommen werden. Für schleifenlose Modelle wurde als Prozesslog jeweils die Menge aller möglicher, in sich abgeschlossener, Ausführungspfade gewählt. Bei Modellen mit Schleifen wurden jeweils mehrere Prozesslogs generiert. Zur Synthese von Prozesslogs wurde ausschließlich auf Petrinetze zurückgegriffen. Dabei gilt ein Ablauf als abgeschlossen, wenn ausgehend von der initialen Markierung durch sukzessives Schalten von Transitionen ein Zustand erreicht wird, in dem keine weitere Transition mehr schaltbereit ist. Die so entstandene Aktivitätssequenz wird als Ausführungspfad aufgefasst.

Analysen auf Basis pseudonymisierter und in Form von Petrinetzen vorliegender, industrieller Standard-Prozesse haben ergeben, dass die Komplexität modellierter Geschäftsprozesse meist gering ausfällt. Ca. 65% der 643 Prozessmodelle umfassenden IBM-Websphere Bibliothek sind streng sequentiell, übrige enthalten genau eine Schleife. Trotz der großen Spanne von minimal 4 und maximal 285 Aktivitäten, enthalten 99% der Modelle max. 36 Aktivitäten (Details in Abb. 4.18). Die Komplexität der in Abschnitt D vorgestellten und hier verwendeten Modelle fällt in vielen Fällen höher aus. Insofern kann die Ausgangsbasis der Evaluation als realistisch betrachtet werden.

Die der Synthese von Prozesslogs zugrundeliegenden Modelle wurden angepasst, um die Funktionsweise und Effektivität des entwickelten Verfahrens besser demonstrieren zu können. Alle generierten Prozesslogs enthalten Aktivitätssequenzen, für die ein Präfix

existiert, das selbst wiederum ein akzeptiertes Wort der Logsprache ist. Bei Abwesenheit solcher Präfixe enden alle Aktivitätssequenzen des Prozesslogs in Senken des extrahierten Transitionssystems. Die Sprache des resultierenden Petrinetzes unter Verwendung der vollständigen Sequenzsemantik entspricht der Menge der Aktivitätssequenzen im Prozesslog. In diesem Fall finden die in dieser Arbeit entwickelten Konzepte keine Anwendung.

Tab. 4.2 gibt einen Überblick über die in diesem Abschnitt verwendeten Prozesslogs und ihre Eigenschaften. Sie wurden mithilfe des Tools SecSy[43] erzeugt. Neben der Zahl unterschiedlicher Prozessaktivitäten, die in einzelnen Abläufen (Traces) der aufgeführten Prozesslogs zu finden sind, enthält Tab. 4.2 auch Informationen zur Zahl unterschiedlicher Abläufe und der durchschnittlichen Länge von Traces. Falls das der Synthese zugrundeliegende Modell Schleifen oder Duplikate enthält (vgl. Abschnitt D.1), kann dies bei der Synthese zu Abläufen führen, in denen einzelne Aktivitäten mehrfach enthalten sind. Das Suffix _n im Namen eines Prozesslogs gibt die Zahl generierter Ausführungen im Syntheseprozess an[44].

Spalten 3 und 4 enthalten die Zahl unterschiedlicher Ausführungen und die durchschnittliche Tracelänge. Bei steigender Zahl generierter Ausführungen nimmt bei Modellen, die Schleifen enthalten, die Zahl unterschiedlicher Ausführungen erwartungsgemäß zu (siehe Abb. 4.19). Während für das Prozessmodell a5 der Anstieg minimal ausfällt, nimmt bei den übrigen Modellen die Zahl unterschiedlicher Ausführungen teils stark zu. Als Maß für diese Perspektive der Komplexitätsbetrachtung von Ursprungsmodellen kann die Steigung der Regressionsgeraden herangezogen werden. Für a5 liegt diese bei 0.00203, das Modell l1lSkip weist dagegen einen Wert von 0.359 auf. Wesentlich geringerer Veränderung unterliegt die durchschnittliche Tracelänge, die Abb. 4.20 zu entnehmen ist.

4.6.4 Präzision rekonstruierter Prozessmodelle

Für die Auswahl geeigneter Multimengen bei Ereignis- bzw. Zustandsspaltung zur Sicherstellung der EC-Eigenschaft wurde auf die Methode zurückgegriffen, die sich an der Deckung von Anreizbereichen orientiert (siehe Abschnitt 4.4.1). Auf Basis bisheriger Erfahrungen erzielt diese Methode im Vergleich zur gradientenorientierten Strategie bessere Ergebnisse i.S. weniger Spaltungsoperationen und erzeugt damit kompaktere Netze. Zudem weist sie eine niedrigere Berechnungskomplexität auf, was die Durchführbarkeit einzelner, unten aufgeführter Experimente erst ermöglicht hat. GENET und GENET$^+$ verfolgen eine Greedy-Strategie und nähern sich schrittweise einem lokalen Optimum an, was nicht notwendigerweise zu einem minimalen Petrinetz i.S. der Anzahl von Transitionen, Stellen und Kanten führt. Die Auswahl geeigneter *ER*-Multimengen für Ereignis- bzw. Zustandsspaltungen erfolgt zwar lokal (d.h. innerhalb einer Iteration und nach der Berechnung minimaler Regionen) optimal, es ist jedoch möglich, dass eine Reihe lokal suboptimaler Entscheidungen zu einem insgesamt besseren Ergebnis führen. Die optimale

[43] SecSy ist ein aus eigenen Vorarbeiten entstandenes Tool zur sicherheitsorientierten Synthese von Prozesslogs auf Basis einer Kontrollflussbeschreibung in Form eines Petrinetzes und ergänzenden kontextspezifischen Parametern (Prozessbeteiligte, Zugriffskontroll-Schema, Datenverwendung, ...). Das Tool ist frei verfügbar unter http://www.sourceforge.net/projects/secsy.

[44] Prozesslogs zur Evaluierung des GENET$^+$-Verfahrens, einschließlich der zugrundeliegenden Prozessmodelle, können unter http://prorepo.process-security.de/g/da8c8e95 eingesehen werden.

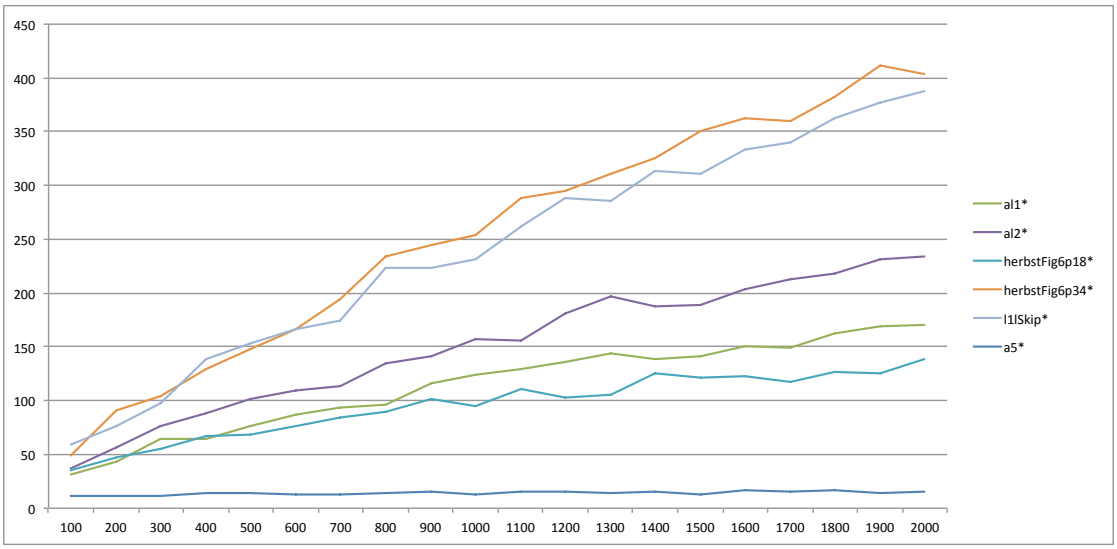


Abbildung 4.19.: Anzahl unterschiedlicher Traces bei steigender Zahl generierter Ausführungen von Prozessmodellen mit unbeschränkten Schleifen.

Sequenz von Spaltungsoperationen im Voraus zu berechnen ist theoretisch zwar möglich, erfordert aber einen enorm hohen Berechnungsaufwand[45].

Im Verlauf einzelner Experimente kann die Zahl berechneter Regionen und generierter *ER*-Multimengen, die für Spaltungsoperationen infrage kommen und damit hinsichtlich ihrer Brauchbarkeit bewertet werden müssen, zu Speicherengpässen bei verwendeten Maschinen führen. Bei der Implementierung des Verfahrens wurde auf sparsame Speichernutzung geachtet und an geeigneten Stellen auf Multi-Threading Konzepte zurück gegriffen, um die Laufzeit der Berechnungen zu optimieren. Bei der Berechnung minimaler Regionen erfolgt die Exploration von Anreizbereichen einzelner Ereignisse jeweils in einem separaten Thread, ebenso die Bewertung von *ER*-Multimengen und die Bestimmung redundanter Regionen. Diese Architektur hat zur Folge, dass Rekonstruktionsergebnisse nicht eindeutig sind und sich aufgrund des Schedulingverhaltens verwendeter Maschinen innerhalb unterschiedlicher Läufe unterscheiden können. Das betrifft bspw. die zuerst observierte (und damit zur Spaltung ausgewählte) *ER*-Multimenge mit bestmöglichem Score. Bei mehrfacher Anwendung desselben Verfahrens auf dieselbe Eingabe können unterschiedliche Ausgaben generiert werden, die zwar jeweils die geforderten Qualitätsgarantien einhalten, jedoch unterschiedlich komplex hinsichtlich ihrer Struktur sind.

Für die unten diskutierten Experimente wurde folgende Hardware verwendet:

(1) Lokaler Arbeitsplatzrechner: 3GHz Intel Core i7 mit 8GB DDR3 Arbeitsspeicher. (In unten aufgeführten Laufzeiten mit dem Zusatz $^{(1)}$ gekennzeichnet).

(2) High Performance Cluster Maschine (bwGrid): HP ProLiant mit 31 Prozessorkernen (jeweils 2,64GHz) und 80GB Arbeitsspeicher.
(In unten aufgeführten Laufzeiten mit dem Zusatz $^{(2)}$ gekennzeichnet).

[45] Vgl. Carmona 2012.

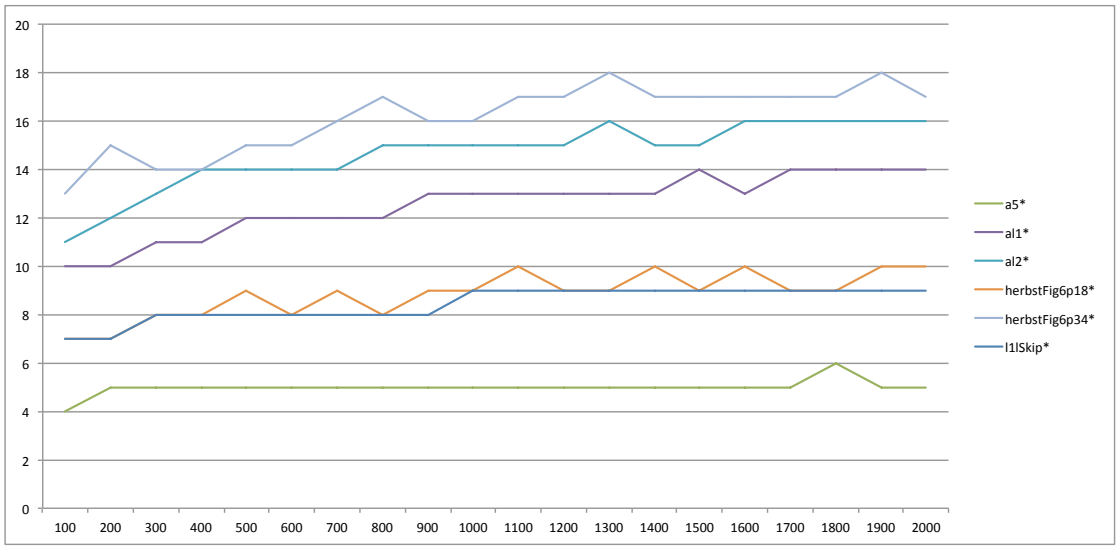


Abbildung 4.20.: Durchschnittliche Tracelänge bei steigender Zahl generierter Ausführungen von Prozessmodellen mit unbeschränkten Schleifen.

Bei Existenz unbeschränkter Schleifen ist zur Bewertung rekonstruierter Modelle ein Vergleich mit dem Ursprungsmodell der generierten Logs nicht sinnvoll. Die Zielsetzung ist vielmehr, ein Modell zu erlangen, welches exakt das beschränkte Verhalten des Prozesslogs abbildet. Ein präzises rekonstruiertes Modell kann folglich nur beschränkte Schleifen enthalten und ist somit in seiner Struktur notwendigerweise komplexer als das Ursprungsmodell. Insofern weicht die Bewertung bewusst wesentlich von verwendeten Verfahren und Kriterien aus dem Bereich Process Discovery (möglichst kompakte Darstellung des gängigsten Verhaltens) und insbesondere dem Rediscovery Problem (größtmögliche Ähnlichkeit zum Ursprungsmodell) ab. Stattdessen wird auf die zuvor definierten Metriken *VSS-Affinität* und *VSS-Deckung* zurückgegriffen. Dabei ist anzumerken, dass unter Verwendung von GENET bzw. GENET$^+$ rekonstruierte Modelle stets völlige Affinität aufweisen. Jede vollständige Aktivitätssequenz kann mit einem Pfad im Erreichbarkeitsgraphen assoziiert werden, der in einem Blatt endet und dabei ein Wort der Sprache des Prozesslogs generiert. Aufgrund der Bisimilarität des Erreichbarkeitsgraphen eines rekonstruierten Petrinetzes zum extrahierten Transitionssystem kann es keinen Pfad im Erreichbarkeitsgraphen geben, der in einem Blatt endet und gleichzeitig nicht mit einem Wort der Logsprache assoziiert werden kann. Jedoch ist es möglich, dass nicht für alle Wörter der Logsprache ein solcher Pfad existiert (keine vollständige Deckung).

Tab. 4.3 und Tab. 4.4 enthalten Anhaltspunkte für die Komplexität rekonstruierter Modelle für GENET und GENET$^+$ im Vergleich. Die Eingabekomplexität kann mithilfe der Kennwerte in Spaltengruppe **TS** abgeschätzt werden. Dabei steht $|S|$ für die Anzahl der Zustände, $|E|$ für die Anzahl der Ereignisse und $|A|$ für die Anzahl der Transitionen innerhalb des auf Basis des Prozesslogs generierten Transitionssystems. Die Anzahl der Ereignisse entspricht dabei der Zahl unterschiedlicher Aktivitäten innerhalb des Prozesslogs. Die Anzahl von Zuständen ist proportional zur Zahl unterschiedlicher Aktivitätssequenzen im Prozesslog. Je komplexer das Ursprungsmodell, desto komplexer fällt auch das Transitionssystem aus, insbesondere bei Existenz von Schleifen. Ob vor der Sicher-

stellung der EC-Eigenschaft eine Pfadkorrektur Anwendung gefunden hat, wird mit der Spalte PC angegeben. Die Spalte $|E|^*$ beziffert für GENET die Anzahl der Ereignisse nach Sicherstellung der EC-Eigenschaft. Durch die Anwendung von Ereignis- bzw. Zustandsspaltung (siehe Abschnitt 4.4.1 und Abschnitt 4.4.2) werden zusätzliche Ereignisse/Zustände eingeführt und treten dadurch mehrfach im resultierenden Petrinetz auf (die Zahl der Transitionen entspricht dabei der Zahl der Ereignisse). Die Anzahl der Stellen $|P|$ ergibt sich aus der Zahl minimaler, nichtredundanter, k-beschränkter Regionen innerhalb des Ausgangs-TS, $|F|$ steht für die Anzahl der Kanten innerhalb des Petrinetzes. Die Ergebnisse für GENET$^+$ sind relativ zu den Ergebnissen von GENET zu verstehen. Für die Rekonstruktion wurden hier ausschließlich 1-beschränkte Regionen berücksichtigt. Dies schmälert die Aussagekraft der durchgeführten Experimente hinsichtlich der Komplexität resultierender Petrinetze nicht, da diese bei höheren Schranken höchstens kompakter (und damit weniger komplex) werden.

Spalte $\mathcal{D}$ enthält den erreichten Wert der VSS-Deckung und zeigt, dass unter Verwendung von GENET kein Modell mit ausreichender Qualität generiert werden kann. Spalte t enthält für beide Ansätze die benötigte Zeit (walltime) für die Durchführung der Rekonstruktion ohne vorherige PTS-Extraktion (bei GENET$^+$ auch die Zeit für Pfadkorrektur und λ-Annotation). Während ein Großteil der Experimente verhältnismäßig kurze Laufzeiten aufweist (unter einer Minute), stechen die Prozesslogs auf Basis des Modells herbstFig6p34* deutlich heraus. Diese Logs weisen nicht nur die höchsten Werte für unterschiedliche Traces (49-104), sondern auch für die durchschnittliche Tracelänge (13-15) auf.

Diskussion durchgeführter Experimente Bzgl. GENET$^+$ kann grundsätzlich festgestellt werden, dass in jedem Fall ein Petrinetz konstruiert werden konnte, welches sowohl völlige Affinität als auch vollständige Deckung aufweist. Um Vergleiche zum GENET-Ansatz zu erleichtern, enthalten Tab. 4.3 und Tab. 4.4 nicht die absolute Anzahl von Stellen, Transitionen und Kanten des resultierenden Petrinetzes, sondern lediglich die Differenz zum entsprechenden, durch GENET erreichten Modell.

Wird keine Pfadkorrektur angewandt, kann das resultierende Netz bei GENET$^+$ aus dem Ergebnis des GENET-Ansatzes abgeleitet werden. Dafür muss lediglich eine λ-Annotation des dort generierten, die EC-Eigenschaft erfüllenden EC-PPZAs erfolgen. Infolgedessen weisen die konstruierten Petrinetze in diesen Fällen nur eine geringfügig höhere Komplexität auf. Die Anzahl der Stellen verändert sich nicht, da die Menge minimaler Regionen durch die Annotation lediglich durch eine weitere, minimale, jedoch redundante Region erweitert wird. Die zusätzlichen Transitionen werden durch neu hinzugefügte, eindeutige Ereignisse bedingt, über die innere Endzustände des EC-PPZAs mit einem zusätzlichen Endzustand verbunden werden. Abhängig davon, in wie vielen minimalen nichtredundanten Regionen diese inneren Endzustände auftreten, erhöht sich die Zahl der Kanten im Petrinetz. Es werden stets mindestens so viele neue Kanten eingeführt wie Transitionen. Im Fall einer notwendigen Pfadkorrektur wird die Ereignismenge des Ursprungs-PZAs verändert und damit ein völlig neuer Startzustand für die Konstruktion eines präzisen Petrinetzes geschaffen. Ein Aufbau auf GENET ist in diesem Fall nicht möglich. Die GENET$^+$-Resultate können sich deshalb sehr von denen des GENET-Ansatzes unterscheiden und führen teils zu erheblich komplexeren Petrinetzen (z.B. Prozesslog a7*

in Tab. 4.3) oder gar zu weniger komplexen Ergebnissen (z.B. Prozesslog al1*_300 in Tab. 4.4).

Nicht bei jedem Prozesslog müssen für die Sicherstellung der EC-Eigenschaft innerhalb des extrahierten PPZA Spaltungsoperationen vorgenommen werden. Solche Spezialfälle (in Tab. 4.3 und Tab. 4.4 hellgrau hinterlegt) führen dazu, dass die resultierenden Petrinetze sehr kompakt sind. Die Zahl der Transitionen entspricht der Zahl der Ereignisse im PPZA und die Zahl der Stellen den minimalen, nichtredundanten Regionen.

Die Berechnungskomplexität hängt generell stark von der Komplexität des Ursprungs-PPZAs und der benötigten Spaltungsoperationen ab. Je mehr Zustände ein PPZA enthält, desto größer wird der Suchraum für minimale Regionen. Mit steigender Anzahl explorierter *ER*-Multimengen steigt auch die benötigte Zeit für die Auswahl einer Multimenge für die Spaltung. Abb. 4.21, Abb. 4.22 und Abb. 4.23 zeigen am Beispiel von herbstFig6p34*_300 und unter Verwendung von Ereignisspaltung, wie sich GENET$^+$ im Verlauf aufeinander folgender Durchgänge, innerhalb derer nach der Berechnung minimaler Regionen jeweils eine Spaltung vorgenommen wird, einer Lösung in Form eines PPZAs mit geltender EC-Eigenschaft annähert. Diese Visualisierungen unterstreichen bisherige Erfahrungen mit dem GENET$^+$-Verfahren, nach denen eine exponentielle Abnahme des Anteils von Ereignissen mit nicht geltender EC-Eigenschaft charakteristisch für GENET$^+$ ist und sich der Aufwand für die Auswahl geeigneter *ER*-Multimengen nach zunächst rasantem Anstieg stark volatil verhält, bis er sich mit zunehmender Abflachung der Kurve in Abb. 4.21 „beruhigt“, auf deutlich niedrigerem Niveau einpendelt und gegen Ende stark abfällt.

Tab. 4.5 ermöglicht einen direkten Vergleich zwischen Ereignisspaltung und Zustandsspaltung. Neben der Komplexität resultierender Netze wird in den Spalten $\#_s$ die Anzahl durchgeführter Spaltungsoperationen abgetragen. Dabei sei angemerkt, dass eine Ereignisspaltung die Zahl der Transitionen um mindestens (und nicht genau) 1 erhöht. Abhängig von der Anzahl unterschiedlicher Gradienten des Spaltungsereignisses innerhalb der als Spaltungsgrundlage gewählten *ER*-Multimenge kann sich durch eine Ereignisspaltung die Zahl oder Transitionen um mehr als 1 erhöhen, allgemein um $k-1$, bei k verschiedenen Gradienten. Bei Prozesslog herbstFig5p19 wurde bspw. durch eine einzige Ereignisspaltung ein existierendes Ereignis durch vier neue Ereignisse ersetzt. Eine Zustandsspaltung stellt bei jeder Anwendung die Regionseigenschaft für mind. eine *ER*-Multimenge sicher und führt dazu u.U. eine ganze Reihe von zusätzlichen Ereignissen und Zuständen ein, um für alle Ereignisse konstante Gradienten sicherzustellen. Mindestens wird jedoch ein Ereignis und ein Zustand eingeführt, was nicht nur die Anzahl von Transitionen im Ergebnisnetz erhöht, sondern auch die Anzahl an Stellen beeinflusst. Dadurch fällt die Komplexität resultierender Netze bei Anwendung von Zustandsspaltung meist deutlich höher aus als bei Ereignisspaltung. Die Wahl der Spaltungsstrategie sollte sich jedoch an der Zielsetzung der Rekonstruktion orientieren. Während Ereignisspaltung kompaktere Netze liefert, die jedoch potentiell Duplikate enthalten, kann diese Problematik mit Zustandsspaltung umgangen werden, allerdings zum Preis komplexerer Modelle mit potentiell hoher Anzahl stiller Transitionen.

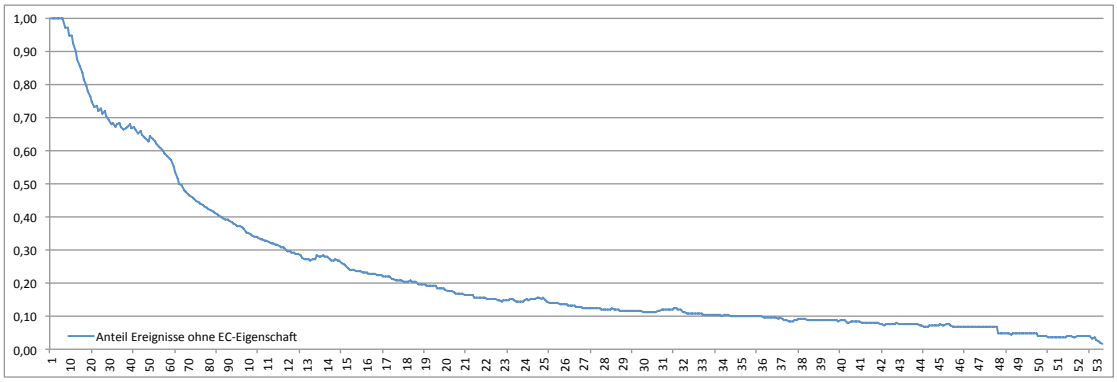


Abbildung 4.21.: Anteil der Ereignisse mit nicht geltender EC-Eigenschaft innerhalb des PPZAs aus Prozesslog herbstFig6p34*_300 im Verlauf sukzessiver Ereignisspaltungen.

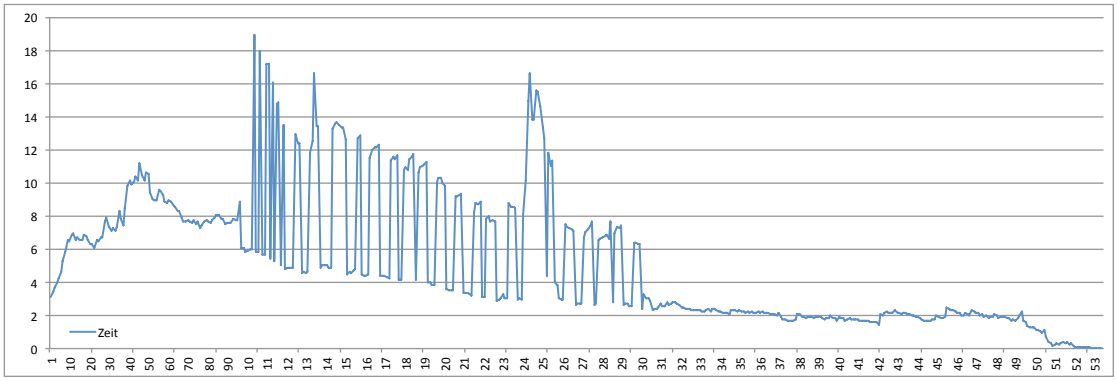


Abbildung 4.22.: Benötigte Zeit (in Sekunden) für die Auswahl einer geeigneten *ER*-Multimenge innerhalb des PPZAs aus Prozesslog herbstFig6p34*_300 im Verlauf sukzessiver Ereignisspaltungen.

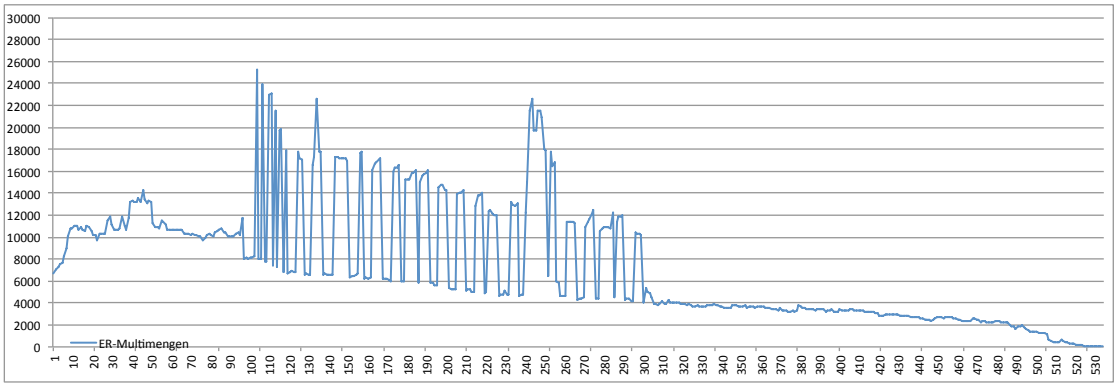


Abbildung 4.23.: Anzahl unterschiedlicher explorierter *ER*-Multimengen des PTS aus Prozesslog herbstFig6p34*_300 im Verlauf sukzessiver Ereignisspaltungen.

Prozesslog	Aktivitäten	Unterschiedliche Traces	Durchschnittl. Tracelänge	Duplikate
a10Skip*	10	7	6	
a6nfc*	6	5	4	
a7*	7	28	4	
a12*	12	7	6	
choice*	10	26	5	
driversLicense*	7	6	3	
herbstFig3p4*	8	14	6	✓
herbstFig5p19	4	6	3	✓
herbstFig6p31*	7	6	3	✓
herbstFig6p38*	3	7	3	✓
herbstFig6p39	3	12	3	✓
herbstFig6p41*	7	30	5	✓
herbstFig6p42	8	20	7	✓
a5*_100	5	11	4	✓
a5*_200	5	11	5	✓
a5*_300	5	11	5	✓
al1*_100	7	31	10	✓
al1*_200	7	43	10	✓
al1*_300	7	64	11	✓
herbstFig6p18*_100	5	35	7	✓
herbstFig6p18*_200	5	47	7	✓
herbstFig6p18*_300	5	55	8	✓
herbstFig6p34*_100	7	49	13	✓
herbstFig6p34*_200	7	91	15	✓
herbstFig6p34*_300	7	104	14	✓
l1lSkip*_100	4	59	7	✓
l1lSkip*_200	4	76	7	✓
l1lSkip*_300	4	97	8	✓

Tabelle 4.2.: Charakteristika verwendeter Prozesslogs.

Prozesslog	TS			GENET					GENET+				
				$\lvert E\rvert^*$						$\lvert E\rvert^*_\Delta$			
	$\lvert S\rvert$	$\lvert E\rvert$	$\lvert A\rvert$	$\lvert T\rvert$	$\lvert P\rvert$	$\lvert F\rvert$	$\mathcal{D}$	t	PC	$\lvert T\rvert_\Delta$	$\lvert P\rvert_\Delta$	$\lvert F\rvert_\Delta$	t
a10Skip*	22	10	21	10	10	25	0,86	8ms$^{(1)}$		1	0	1	9ms$^{(1)}$
a6nfc*	12	6	11	6	6	14	0,6	7ms$^{(1)}$	✓	3	1	12	10ms$^{(1)}$
a7*	45	7	44	7	7	19	0,5	123ms$^{(1)}$	✓	48	25	97	1,85s$^{(1)}$
a12*	25	12	24	12	11	25	0,71	21ms$^{(1)}$		2	0	2	22ms$^{(1)}$
choice*	48	10	47	10	6	19	0,62	18ms$^{(1)}$		10	0	10	20ms$^{(1)}$
driversLicense*	10	7	9	7	7	17	0,33	4ms$^{(1)}$		4	0	8	5ms$^{(1)}$
herbstFig3p4*	30	8	29	8	7	17	0,86	14ms$^{(1)}$		2	0	5	15ms$^{(1)}$
herbstFig5p19	11	4	10	5	5	14	0,67	9ms$^{(1)}$		2	0	2	9ms$^{(1)}$
herbstFig6p31*	10	7	9	7	5	18	0,67	4ms$^{(1)}$		2	0	4	5ms$^{(1)}$
herbstFig6p38*	16	3	15	11	9	34	0,71	103ms$^{(1)}$	✓	2	-1	-5	111ms$^{(1)}$
herbstFig6p39	16	3	15	9	8	20	0,42	140ms$^{(1)}$	✓	10	3	21	204ms$^{(1)}$
herbstFig6p41*	48	7	47	30	25	69	0,5	629ms$^{(1)}$	✓	28	9	43	3,41s$^{(1)}$
herbstFig6p42	52	8	51	39	32	95	0,65	1,51s$^{(1)}$		7	0	9	1,51s$^{(1)}$
a5*_100	26	5	25	11	10	38	0,64	297ms$^{(1)}$		4	0	10	302ms$^{(1)}$
a5*_200	29	5	28	7	8	27	0,73	168ms$^{(1)}$		3	0	9	172ms$^{(1)}$
a5*_300	29	5	28	9	9	29	0,73	125ms$^{(1)}$		3	0	8	126ms$^{(1)}$
al1*_100	128	7	127	121	100	236	0,81	45,58s$^{(1)}$	✓	10	2	2	41,95s$^{(1)}$
al1*_200	143	7	142	134	110	275	0,72	1,24m$^{(1)}$	✓	3	-6	23	3,96m$^{(1)}$
al1*_300	231	7	230	228	186	421	0,72	17,8m$^{(1)}$	✓	-17	-17	196	8,43m$^{(1)}$
herbstFig6p18*_100	88	5	87	87	73	159	0,43	1,77m$^{(2)}$		20	0	20	1,77m$^{(2)}$
herbstFig6p18*_200	83	5	82	79	64	150	0,36	47,45s$^{(2)}$	✓	29	1	37	19,02s$^{(2)}$
herbstFig6p18*_300	107	5	106	100	87	204	0,33	2,71m$^{(2)}$	✓	39	2	43	1,96m$^{(2)}$
herbstFig6p34*_100	260	7	259	257	227	486	0,65	38,77m$^{(2)}$	✓	19	1	17	3,46m$^{(2)}$
herbstFig6p34*_200	545	7	544	542	490	1039	0,59	6,38h$^{(2)}$	✓	39	1	32	26,11m$^{(2)}$
herbstFig6p34*_300	555	7	554	545	493	1048	0,52	10,99h$^{(2)}$	✓	59	8	56	50,49m$^{(2)}$
l1lSkip*_100	143	4	142	136	99	261	0,71	1,4m$^{(1)}$	✓	16	-1	18	29,83s$^{(1)}$
l1lSkip*_200	195	4	194	176	127	368	0,71	1,26m$^{(1)}$	✓	17	5	42	4,09m$^{(1)}$
l1lSkip*_300	232	4	231	223	155	408	0,72	45,5m$^{(1)}$	✓	12	-9	101	28,56m$^{(1)}$

Tabelle 4.3.: Rekonstruktionsergebnisse von GENET und GENET$^+$ im Vergleich (unter Verwendung von Ereignisspaltung).

Prozesslog	TS			GENET					GENET+				
				$\lvert E\rvert^*$						$\lvert E\rvert^*_\Delta$			
	$\lvert S\rvert$	$\lvert E\rvert$	$\lvert A\rvert$	$\lvert T\rvert$	$\lvert P\rvert$	$\lvert F\rvert$	$\mathcal{D}$	t	PC	$\lvert T\rvert_\Delta$	$\lvert P\rvert_\Delta$	$\lvert F\rvert_\Delta$	t
a10Skip*	22	10	21	10	10	25	0,86	15ms(1)		1	0	1	16ms(1)
a6nfc*	12	6	11	6	6	14	0,6	66ms(1)	✓	3	1	12	20ms(1)
a7*	45	7	44	7	7	19	0,5	11ms(1)	✓	65	47	191	1,54s(1)
a12*	25	12	24	12	11	25	0,71	6ms(1)		2	0	2	7ms(1)
choice*	48	10	47	10	6	19	0,62	20ms(1)		10	0	10	21ms(1)
driversLicense*	10	7	9	7	7	17	0,33	18ms(1)		4	0	8	20ms(1)
herbstFig3p4*	30	8	29	8	7	17	0,86	13ms(1)		2	0	5	14ms(1)
herbstFig5p19	11	4	10	12	11	38	0,67	93ms(1)		2	0	4	94ms(1)
herbstFig6p31*	10	7	9	7	5	18	0,67	7ms(1)		2	0	4	8ms(1)
herbstFig6p38*	16	3	15	19	17	66	0,71	201ms(1)	✓	-4	-4	18	129ms(1)
herbstFig6p39	16	3	15	15	14	71	0,42	422ms(1)	✓	7	2	3	152ms(1)
herbstFig6p41*	48	7	47	71	62	237	0,5	3,37s(1)	✓	1	-6	97	3,34s(1)
herbstFig6p42	52	8	51	58	51	242	0,65	1,67s(1)		7	0	15	1,67s(1)
a5*_100	26	5	25	31	29	121	0,64	934ms(1)		4	0	9	940ms(1)
a5*_200	29	5	28	42	38	148	0,73	1,21s(1)		3	0	7	1,22s(1)
a5*_300	29	5	28	18	17	85	0,73	120ms(1)		3	0	10	122ms(1)
al1*_100	128	7	127	164	150	593	0,81	21,12s(1)	✓	-30	-33	96	18,49s(1)
al1*_200	143	7	142	142	124	792	0,72	19,36s(1)	✓	34	18	-47	26,55s(1)
al1*_300	231	7	230	358	323	1273	0,72	1,36m(1)	✓	-81	-85	-93	1,25m(1)
herbstFig6p18*_100	88	5	87	134	125	425	0,43	18,48s(2)		20	0	30	18,56s(2)
herbstFig6p18*_200	83	5	82	118	107	385	0,36	13,73s(2)	✓	30	2	80	16,37s(2)
herbstFig6p18*_300	107	5	106	173	161	546	0,33	44,23s(2)	✓	37	0	56	44,41s(2)
herbstFig6p34*_100	260	7	259	294	270	1153	0,65	1,67m(2)	✓	17	0	38	1,68m(2)
herbstFig6p34*_200	545	7	544	901	855	2958	0,59	1,1h(2)	✓	-37	-74	-276	1,29h(2)
herbstFig6p34*_300	555	7	554	885	839	2903	0,52	1,13h(2)	✓	50	0	71	1,13h(2)
l1lSkip*_100	143	4	142	173	136	948	0,71	15,46s(1)	✓	70	52	-267	15,1s(1)
l1lSkip*_200	195	4	194	294	245	1031	0,71	34,23s(1)	✓	-68	-90	29	21,94s(1)
l1lSkip*_300	232	4	231	397	334	1580	0,72	1,59m(1)	✓	39	13	-293	1,39m(1)

Tabelle 4.4.: Rekonstruktionsergebnisse von GENET und GENET$^+$ im Vergleich (unter Verwendung von Zustandsspaltung).

Prozesslog	TS				GENET+ (ES)					GENET+ (ZS)				
					$\lvert E\rvert^*$					$\lvert E\rvert^*$				
	$\lvert S\rvert$	$\lvert E\rvert$	$\lvert A\rvert$	PC	$\lvert T\rvert$	$\lvert P\rvert$	$\lvert F\rvert$	$\#_s$	t	$\lvert T\rvert$	$\lvert P\rvert$	$\lvert F\rvert$	$\#_s$	t
a10Skip*	22	10	21		11	10	26	0	9ms$^{(1)}$	11	10	26	0	16ms$^{(1)}$
a6nfc*	12	6	11	✓	7	7	26	0	10ms$^{(1)}$	7	7	26	0	20ms$^{(1)}$
a7*	45	7	44	✓	41	32	116	23	1,85s$^{(1)}$	58	54	210	37	1,54s$^{(1)}$
a12*	25	12	24		14	11	27	0	22ms$^{(1)}$	14	11	27	0	7ms$^{(1)}$
choice*	48	10	47		20	6	29	0	20ms$^{(1)}$	20	6	29	0	21ms$^{(1)}$
driversLicense*	10	7	9		11	7	25	0	5ms$^{(1)}$	11	7	25	0	20ms$^{(1)}$
herbstFig3p4*	30	8	29		10	7	22	0	15ms$^{(1)}$	10	7	22	0	14ms$^{(1)}$
herbstFig5p19	11	4	10		7	5	16	1	9ms$^{(1)}$	14	11	42	6	94ms$^{(1)}$
herbstFig6p31*	10	7	9		9	5	22	0	5ms$^{(1)}$	9	5	22	0	8ms$^{(1)}$
herbstFig6p38*	16	3	15	✓	11	8	29	7	111ms$^{(1)}$	13	13	84	7	129ms$^{(1)}$
herbstFig6p39	16	3	15	✓	12	11	41	7	204ms$^{(1)}$	15	16	74	8	152ms$^{(1)}$
herbstFig6p41*	48	7	47	✓	43	34	112	28	3,41s$^{(1)}$	57	56	334	32	3,34s$^{(1)}$
herbstFig6p42	52	8	51		46	32	104	31	1,51s$^{(1)}$	65	51	257	28	1,67s$^{(1)}$
a5*_100	26	5	25		15	10	48	6	302ms$^{(1)}$	35	29	130	19	940ms$^{(1)}$
a5*_200	29	5	28		10	8	36	2	172ms$^{(1)}$	45	38	155	26	1,22s$^{(1)}$
a5*_300	29	5	28		12	9	37	4	126ms$^{(1)}$	21	17	95	6	122ms$^{(1)}$
al1*_100	128	7	127	✓	125	102	238	112	41,95s$^{(1)}$	128	117	689	57	18,49s$^{(1)}$
al1*_200	143	7	142	✓	125	104	298	112	3,96m$^{(1)}$	164	142	745	66	26,55s$^{(1)}$
al1*_300	231	7	230	✓	193	169	617	174	8,43m$^{(1)}$	259	238	1180	96	1,25m$^{(1)}$
herbstFig6p18*_100	88	5	87		107	73	179	80	1,77m$^{(2)}$	154	125	455	69	18,56s$^{(2)}$
herbstFig6p18*_200	83	5	82	✓	78	65	187	71	19,02s$^{(2)}$	118	109	465	65	16,37s$^{(2)}$
herbstFig6p18*_300	107	5	106	✓	102	89	247	93	1,96m$^{(2)}$	210	161	602	105	44,41s$^{(2)}$
herbstFig6p34*_100	260	7	259	✓	259	228	503	238	3,46m$^{(2)}$	311	270	1191	63	1,68m$^{(2)}$
herbstFig6p34*_200	545	7	544	✓	544	491	1071	505	26,11m$^{(2)}$	827	781	2682	398	1,29h$^{(2)}$
herbstFig6p34*_300	555	7	554	✓	554	501	1104	508	50,49m$^{(2)}$	935	839	2974	362	1,13h$^{(2)}$
l1lSkip*_100	143	4	142	✓	135	98	279	116	29,83s$^{(1)}$	226	188	681	129	15,1s$^{(1)}$
l1lSkip*_200	195	4	194	✓	171	132	410	150	4,09m$^{(1)}$	204	155	1060	60	21,94s$^{(1)}$
l1lSkip*_300	232	4	231	✓	208	146	509	188	28,56m$^{(1)}$	409	347	1287	203	1,39m$^{(1)}$

Tabelle 4.5.: Vergleich von Ereignisspaltung (ES) und Zustandsspaltung (ZS).

Kapitel 5

Kombinierte Kontroll-/Datenfluss-Rekonstruktion

Dieses Kapitel beschreibt, wie der in Abschnitt 4.5 vorgestellte GENET$^+$-Ansatz erweitert werden kann, um aufbauend auf einem Prozesslog neben dem Kontroll- auch den Datenfluss präzise abbilden zu können. Dabei ist es für eine präzise Darstellung tatsächlich aufgetretenen Prozessverhaltens nicht ausreichend, ein mithilfe der Theorie in Abschnitt 4.5 rekonstruiertes Strukturmodell nachträglich mit Datenflussinformation anzureichern. Existiert innerhalb zweier aus Kontrollflussperspektive identischer Aktivitätssequenzen im Prozesslog eine Aktivität, die nicht in beiden Sequenzen dieselbe Datenverwendung aufweist, wird diesem Unterschied im rekonstruierten Strukturmodell keine Rechnung getragen; strukturelle Veränderungen sind in diesem Fall notwendig, um diese Sequenzen zu unterscheiden. Diese Problematik wird von dem Beispiel in Abb. 5.1 verdeutlicht.

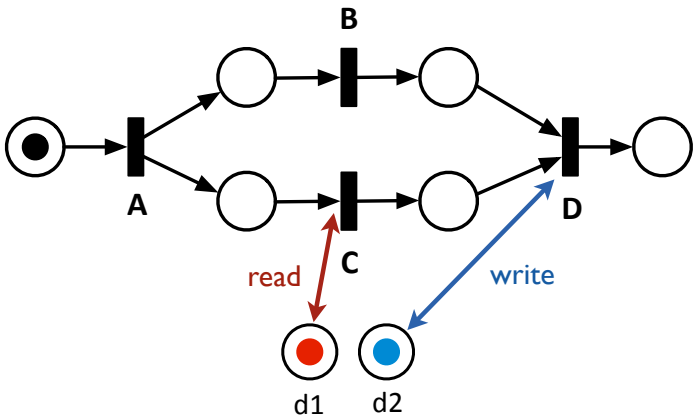


Abbildung 5.1.: Beispiel für die Datenfluss-Annotation eines Petrinetzes mithilfe unterscheidbarer Marken.

Ausgehend von einem einfachen Prozesslog, der lediglich die zwei Aktivitätssequenzen $\langle A, B, C[read\ d1], D[write\ d2]\rangle$ und $\langle A, C[read\ d1], B, D[write\ d2]\rangle$ enthält, beschreibt

das Modell präzise das Logverhalten. Existiert jedoch eine weitere (bzgl. des reinen Kontrollflusses nicht abweichende) Sequenz $\langle A, B, C[read\ d1], D\rangle$, kann das dadurch implizierte, zusätzliche Verhalten nicht im Modell wiedergegeben werden, ohne Transition D zu duplizieren. Allgemein können unterschiedliche Datenverwendungen von Prozessaktivitäten weitgehende, kaskadierende Veränderungen im Kontrollflussmodell nach sich ziehen. Die Erweiterung des GENET$^+$-Verfahrens zur Berücksichtigung des Datenflusses setzt deshalb schon bei der PTS-Konstruktion an und unterscheidet Prozessaktivitäten anhand ihrer Datenverwendung.

Eine Voraussetzung für diese kombinierte Rekonstruktion ist ein geeignetes formales Metamodell, welches nicht nur die Definition von Kontroll- und Datenfluss erlaubt, sondern idealerweise auch Konstrukte bereithält, die eine an die Rekonstruktion angeschlossene, modellbasierte Analyse ermöglichen. Der im Rahmen dieser Arbeit entwickelte IFnet-Formalismus erfüllt diese Anforderungen und eignet sich aufgrund seiner expliziten Ausrichtung auf eine informationsflussorientierte Sicherheitsanalyse als Grundlage für formale Prozessanalysen. Nach einem Exkurs zum IFnet-Formalismus in Abschnitt 5.1 wird in Abschnitt 5.2 das Verfahren GENET* zur kombinierten Kontroll-/Datenfluss-Rekonstruktion vorgestellt.

5.1 Exkurs: Der IFnet-Formalismus

Der IFnet-Formalismus[1] ist ein auf Vorarbeiten von Accorsi und Wonnemann[2] basierender Petrinetzdialekt, der im Vergleich zu bestehenden Ansätzen eine intuitivere Modellierung von Datenflüssen mithilfe unterscheidbarer Marken (deren Typ jeweils für ein Datenelement steht) erlaubt. Mithilfe eines IFnet ist es möglich, den Kontroll- und Datenfluss eines Prozesses unter Berücksichtigung sicherheitsspezifischer Eigenschaften zu definieren. Durch die Verwendung einer zweistufigen Sicherheitsklassifikation (Details siehe unten) erfüllt IFnet die Voraussetzungen für die Anwendung von Methoden der Informationsflusskontrolle, die imstande ist, Sicherheitseigenschaften sowohl hinsichtlich des Kontrollflusses als auch des Datenflusses eines Prozesses zu verifizieren.

Ein IFnet ist im weitesten Sinne ein Petrinetz mit strukturellen, prozessspezifischen Eigenschaften mit zusätzlichen Konstrukten, die sicherheitsorientierte Spezifikationen erlauben. Dazu zählen Prozessbeteiligte (Subjekte) und Sicherheitslevels, die an Subjektbezeichner, Transitionen und Datenelemente angeheftet werden können, um spezifische Sicherheitsanforderungen zu modellieren.

Um zwischen klassifizierten und nicht klassifizierten Elementen unterscheiden zu können, wird der Prozess beim IFnet-Formalismus in zwei logische Sicherheitsdomänen (high für geheim und low for öffentlich) unterteilt. Im Sinne der Informationsflusskontrolle wird ein Prozess als sicher angesehen, wenn kein Ausführungspfad existiert, der einen Informationsfluss von der high-Domäne zur low-Domäne erlaubt. (i.e. die Domänen nicht

[1] Vgl. Stocker et al. 2013.
[2] Vgl. Accorsi und Wonnemann 2011.

interferieren). Nicht-Interferenz (engl.: *non-interference*) ist eine sehr restriktive Sicherheitsnotation, die neben der Vertraulichkeit von Datenelementen auch die Vertraulichkeit (Nichtobservierbarkeit) von Prozessaktivitäten abbilden kann. Neben grundlegenden Sicherheitseigenschaften (z.B. dem Bell-LaPadula Modell für die Vertraulichkeit von Datenelementen) können auf diese Weise auch *multi-level* Modelle abgebildet werden. In diesem Fall muss die halbgeordnete Menge verwendeter Sicherheitsstufen auf geeignete Weise in high und low-Bereiche partitioniert werden. Insofern stellt die Betrachtung von lediglich zwei Domänen keine Restriktion dar.

Definition 5.1 (IFnet). *Ein IFnet ist ein Tupel* $(P, T_R, T_D, F, I, O, C, \mathcal{C}, m_0, A, AC, \mathfrak{G})$ *wobei* $(P, T_R \uplus T_D, F, I, O, C, \mathcal{C}, m_0)$ *ein CPN ist. Neben regulären Transitionen* ($t_R \in T_R$)*, gibt es Deklassifikationstransitionen* ($t_D \in T_D$) *die für sog. downgrades genutzt werden können, also die Überführung von Elementen aus der high- in die low-Domäne.* ⊣

Die Menge der Markentypen ergibt sich zu $\mathcal{C} = \mathcal{C}_c \uplus \{\text{black}\}$. Der Typ „black“ wird verwendet, um den Kontrollfluss eines Netzes zu modellieren, während alle übrigen Typen $\mathcal{C}_c$ Datenelemente repräsentieren, die während der Prozessausführung verwendet werden. Dabei referenzieren Marken desselben Typs dasselbe Datenelement. Bzgl. des Kontrollflusses werden Konsistenzbedingungen eingeführt, die das Schalten von Transitionen an Kontrollflussmarken binden:

- $\forall_{t \in T} \exists_{p \in {}^\bullet t} : supp(I(p,t)) \cap \{\text{black}\} \neq \emptyset$

  Kontrollflussmarke muss beim Schalten einer Transition konsumiert werden.

- $\forall_{t \in T} \exists_{p \in t^\bullet} : supp(O(t,p)) \cap \{\text{black}\} \neq \emptyset$

  Kontrollflussmarke muss beim Schalten einer Transition generiert werden.

Die einzelnen IFnet-Komponenten sind wie folgt definiert:

Analysekontext (AC): Der Analysekontext eines IFnet ist ein Tupel (L, E, U), wobei die Funktion $E : T \to U$ jeder Transition einen Subjektbezeichner aus dem Alphabet U zuordnet und L die Auszeichnung des Netzes darstellt, welche die Sicherheitsklassifikation von Prozessaktivitäten (Transitionen) und Datenelementen, sowie die Berechtigungsstufen von Subjekten definiert. L ist ein 3-Tupel (S_T, S_U, S_C) mit folgenden Eigenschaften:

- **Transitionsklassifikation**: $S_T \to \{\text{high}, \text{low}\}$ bestimmt für jede Prozessaktivität/Transition die Sicherheitsstufe. Befindet sich eine Transition in der high-Domäne, darf das Schalten der Transition für Subjekte der low-Domäne nicht observierter sein.

- **Berechtigungsstufe**: $S_U \to \{\text{high}, \text{low}\}$ gibt für jedes Subjekt $u \in U$ an, ob es der high- oder low-Domäne zugeordnet wird. Subjekte der Domäne low dürfen nur Informationen der Stufe low erhalten; Subjekte innerhalb der Domäne high alle Informationen.

- **Datenklassifikation**: $S_C : C_c \to \{\text{high}, \text{low}\}$ bestimmt für jedes Datenelement/jeden Markentyp ausser „black“ die Sicherheitsstufe.

Datenverwendung: $A : T_R \times \mathcal{C}_c \rightarrow \mathcal{P}(\mathcal{M}_A)$ ist eine Funktion, die für jede reguläre Transition $t \in T_R$ und jeden Markentyp $c \in \mathcal{C}_c$ angibt, auf welche Art und Weise die durch t repräsentierte Prozessaktivität auf das Datenelement zugreift, welches mit dem Markentyp c referenziert wird. Gültige Zugriffsmodi sind definiert als $\mathcal{M}_A = \{read, write, delete, create\}$. Eine Prozessaktivität kann lesend auf bereits existierende Elemente zugreifen (read) oder deren Inhalt modifizieren (write). Des Weiteren können neue Datenelemente erzeugt (create) oder existierende entfernt/gelöscht werden (delete). Die Datenverwendungsfunktion muss folgende Eigenschaften erfüllen:

- $|A(t,c) \cap \{create, delete\}| \leq 1$

  Datenelemente können erzeugt oder entfernt werden, niemals beides gleichzeitig.

- $\forall_{t \in T} \forall_{c \in \mathcal{C}_c} : \text{create} \in A(t,c) \Rightarrow \forall_{p \in {}^\bullet t} : I(p,t)(c) = 0 \wedge \exists_{p \in t^\bullet} : O(t,p)(c) > 0$

  Marken, deren Typ mit einem Datenelement assoziiert wird, welches von einer Transition erzeugt wird, können von dieser nicht konsumiert, müssen jedoch generiert werden.

- $\forall_{t \in T} \forall_{c \in \mathcal{C}_c} : \text{delete} \in A(t,c) \Rightarrow \forall_{p \in t^\bullet} : O(t,p)(c) = 0 \wedge \exists_{p \in {}^\bullet t} : I(p,t)(c) > 0$

  Marken, deren Typ mit einem Datenelement assoziiert wird, welches von einer Transition erzeugt wird, können von dieser nicht generiert, müssen jedoch konsumiert werden.

- $\forall_{t \in T} \forall_{c \in \mathcal{C}_c} : A(t,c) \cap \{\text{delete}, \text{create}\} = \emptyset \Rightarrow \exists_{p \in {}^\bullet t} : I(p,t)(c) > 0 \wedge \exists_{p \in t^\bullet} : O(t,p)(c) > 0$

  Marken, deren Typ mit von einer Transition verwendeten (nicht generiert oder entfernt) Datenelementen assoziiert werden, müssen sowohl konsumiert als auch generiert werden.

Die Funktionen für konsumierte und generierte Markentypen von CPNs werden zur Berücksichtigung von Markentypen auf natürliche Weise erweitert:

- **Konsumierte Marken:**

  $N_c^{\mathcal{M}_A} : T \times \mathcal{P}(\mathcal{M}_A) \rightarrow \mathcal{P}(\mathcal{C})$

  $N_c^{\mathcal{M}_A}(t, M) = \{c \in N_c(t) \mid A(t,c) \supseteq M\}$

  $N_c^{\mathcal{M}_A}(t, M)|_\gamma(t) = N_c^{\mathcal{M}_A}(t, M) \cap \gamma$

  wobei $\gamma \in \mathcal{P}(\mathcal{C})$

- **Generierte Marken:**

  $N_p^{\mathcal{M}_A} : T \times \mathcal{P}(\mathcal{M}_A) \rightarrow \mathcal{P}(\mathcal{C})$

  $N_p^{\mathcal{M}_A}(t, M) = \{c \in N_p(t) \mid A(t,c) \supseteq M\}$

  $N_p^{\mathcal{M}_A}(t, M)|_\gamma(t) = N_p^{\mathcal{M}_A}(t, M) \cap \gamma$

wobei $\gamma \in \mathcal{P}(\mathcal{C})$

Nebenbedingungen: Der IFnet-Formalismus erlaubt die Modellierung von Nebenbedingungen, die für das Schalten einer Transition erfüllt sein müssen. Dafür kann eine Annotation eines Bezeichners für eine Nebenbedingung an eine Transition vorgenommen werden. Vom Inhalt der Bedingung wird abstrahiert, eine Nebenbedingung muss sich jedoch explizit auf eine Menge von Datenelementen beziehen. Formal ist eine Nebenbedingung als Tupel $(p_g, c_g) \in P_g \times \mathcal{P}(\mathcal{C}_c)$ definiert, wobei P_g für die Menge aller Bezeichner für Nebenbedingungen steht. Die Funktion $G : T \to \mathcal{P}(P_g \times \mathcal{C}_c)$ ordnet jeder Transition eine Menge von Nebenbedingungen zu. Dabei bezeichnet $\mathcal{G}$ die Menge aller Nebenbedingungen. Die Berücksichtigung von Nebenbedingungen hat zur Folge, dass die Voraussetzungen für das Schalten einer Transition um die Forderung erweitert werden muss, dass alle der Transition zugeordneten Nebenbedingungen zu „wahr“ evaluieren.

Deklassifikation: Transitionen $t \in T_D$ werden genutzt, um Informationsflüsse von der high- in die low-Domäne explizit zu erlauben. Innerhalb eines Geschäftsprozesses können solche Situationen auftreten, wenn sensible Information innerhalb eines high klassifizierten Dokuments von einem Subjekt der high-Domäne entfernt werden und dieses Dokument dann zur Bearbeitung an Subjekte innerhalb der low-Domäne weitergereicht wird. Ohne die Möglichkeit der Deklassifikation muss jedes von einem Subjekt der high-Domäne bearbeitete Datenelement als high eingestuft werden und darf diese Domäne niemals verlassen. Um die Konsistenz von Klassifikationen zu gewährleisten, müssen Deklassifikationstransitionen folgende Bedingungen erfüllen:

- Es existiert genau eine eingehende und genau eine ausgehende Kante:

  $\forall_{t_D \in T_D} : {}^\bullet t_D = \{i_{t_D}\}$ und $t_D^\bullet = \{o_{t_D}\}$

- Deklassifikationstransitionen müssen effektiv sein, i.e. mind. eine für ein Datenelement stehende Marke konsumieren:

  $\forall_{t_D \in T_D} : N_c|_{\mathcal{C}_c}(_D) \neq \emptyset$

- Die Menge konsumierter Markentypen und generierter Markentypen (mit Ausnahme des Markentyps zur Modellierung des Kontrollflusses) weisen keine gemeinsamen Elemente auf. Ein deklassifiziertes Datenelement wird als Kopie des ursprünglichen Datenelements aufgefasst, welche möglicherweise veränderte Daten enthält.

  $N_c(t_D)|_{\mathcal{C}_c} \cap N_p(t_D)|_{\mathcal{C}_c} = \emptyset$

- Die von Deklassifikationstransitions generierten Markentypen sind exklusiv, d.h. es gibt weder eine reguläre Transition, die eine Marke desselben Type per „create“ erzeugt, noch eine andere Deklassifikationstransition, die eine Marke desselben Typs generiert:

  $$\forall_{t_D \in T_D} : N_p(t_D)|_{\mathcal{C}_c} \cap \left(\bigcup_{t \in T_R} N_p^{\mathcal{M}_A}(t, \{\text{create}\}) \cup \bigcup_{t \in T_D \setminus \{t_D\}} N_p(t) \right) = \emptyset$$

- Für jedes $t_D \in T_D$ existiert eine bijektive Funktion $DF_{t_D} : N_c(t_D)|_{\mathcal{C}_c} \leftrightarrow N_p(t_D)|_{\mathcal{C}_c}$, die jedem konsumierten Markentyp einen generierten Markentyp zuweist, der für das deklassifizierte Datenelement steht. Für jeden konsumierten Markentyp c_1 generiert die Deklassifikationstransition so viele Marken des von DF zugewiesenen Ausangs-Markentyps c_2 wie sie von c_1 konsumiert hat:

  $$\forall_{t_D \in T_D} \forall_{c \in N_c(t_D)|_{\mathcal{C}_c})} : O(t_D, o_{t_D})(DF_{t_D}(c)) = I(i_{t_D}, t_D)(c)$$

- Generierte Marken von Deklassifikationstransitionen tragen die Klassifikation low:

  $\forall_{t_D \in T_D} \forall_{c \in N_P(t_D)|_{\mathcal{C}_c}} : S_C(c) = \mathsf{low}$

- Deklassifikationstransitionen tragen die Klassifikation high. Andernfalls können Subjekte der low-Domäne high-klassifizierte Elemente deklassifizieren.

  $\forall_{t_D \in T_D} : S_T(t_D) = \mathsf{high}$

Zusammenhang von Klassifikation und Berechtigungsstufen: Um konsistente IFnet-Spezifikationen hinsichtlich der Sicherheitseinstufung von Prozessaktivitäten, Datenelementen und Prozessbeteiligten zu gewährleisten, müssen folgende Bedingungen eingehalten werden:

- $\nexists_{t \in T} : S_U(E(t)) = \mathsf{low} \wedge S_T(t) = \mathsf{high}$

  Subjekte der Berechtigungsstufe low dürfen keinen Transitionen mit high-Klassifikation zugeordnet werden.

- $\forall_{c \in \mathcal{C}_c} \forall_{t \in T_R} : \text{create} \in A(t, c) \Rightarrow S_C(c) = S_U(E(t))$

  Die Klassifikation generierter Marken einer Transition entspricht der Berechtigungsstufe des zugeordneten Subjekts.

Abb. 5.2 enthält ein Beispiel für eine Prozessspezifikation mithilfe des IFnet-Formalismus. Bei dem Beispiel handelt es sich um eine vereinfachte Version eines Prozesses für die Bearbeitung einer Anfrage eines Zulieferers bzgl. eines Konstruktionsplans. In vielen Fällen benötigen Zulieferer bestimmte Details hergestellter Güter (Form, Abmessung, Beschaffenheit etc.), um gelieferte Einzelteile optimal abstimmen zu können. Andererseits werden Konstruktionspläne typischerweise nicht herausgegeben, ohne zuvor nicht benötigte oder als klassifiziert eingestufte Informationen zu entfernen. Nach Eingang der Anfrage und der Prüfung der Berechtigung des Lieferanten für das Abrufen von Konstruktionsdetails durch einen Sachbearbeiter, wird die Anfrage an einen in den Verarbeitungsprozess involvierten Ingenieur weitergeleitet. Dieser extrahiert aus dem Konstruktionsplan relevante Details und bereitet ein Dokument für die Weitergabe benötigter Informationen vor, welches schließlich an den Zulieferer weitergereicht wird. Im Hinblick auf den Datenfluss innerhalb des Prozesses werden in Abb. 5.2 (b) insgesamt 5 Markentypen unterschieden. Während „black“ für die Modellierung des Kontrollflusses verwendet wird, stehen die übrigen Markentypen für verwendete Datenelemente, die erst während der Prozessausführung generiert werden. Die Identifikationsnummer des abgefragten Dokuments („blue“) wird bis zur Aktivität „Get Document“ weitergereicht, wo sie mit einer Marke des Typs

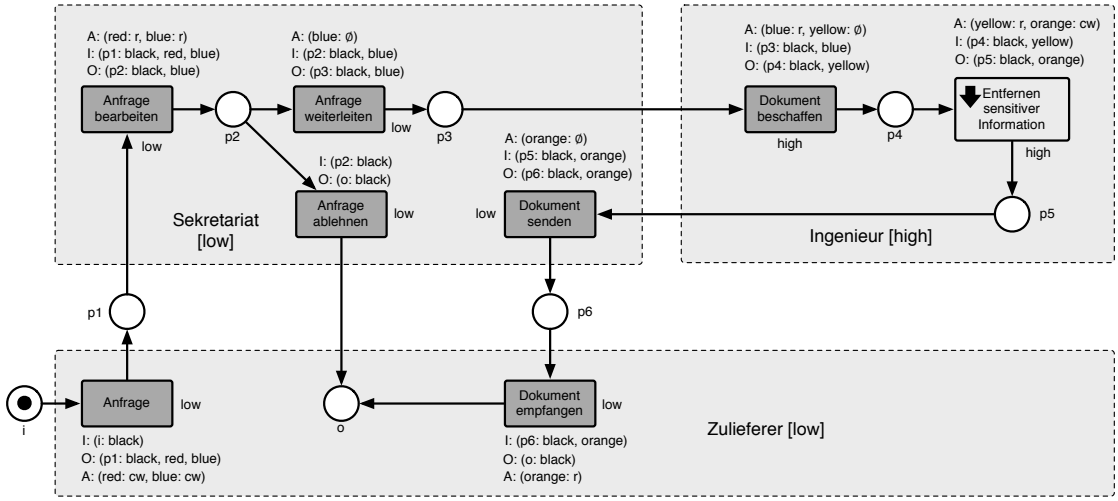


Abbildung 5.2.: Beispielprozess für die Anfrage von Konstruktionsdetails in Form eines IFnet.

„yellow“ ersetzt wird, die für das betreffende Dokument selbst steht. Die Deklassifikationstransition „Remove Classified Information“ wird dann dazu genutzt, die Erstellung einer Variante des Ursprungsdokuments (Marke des Type „orange“) zu modellieren, die keine geheim zu haltende Information mehr enthält. Auf diese Weise kann Information aus der vertraulichen high Domäne in den öffentlichen low-Bereich transferiert werden.

5.2 GENET*: Datenflussannotation rekonstruierter Prozessmodelle

Ein Strukturmodell beschreibt die Gesamtheit der Aktivitätssequenzen eines Prozesslogs, unterscheidet Aktivitäten dabei aber nicht hinsichtlich ihrer Datenverwendung. Für die zusätzliche Berücksichtigung des Datenflusses setzt GENET* bereits bei der PTS-Konstruktion an, die Zustände bisher anhand eindeutiger von der Wurzel zum betreffenden Zustand führenden Aktivitätssequenzen definiert und sich dabei ausschließlich auf den Kontrollfluss beschränkt. Die grundsätzliche Idee von GENET* ist, durch eine explizite Unterscheidung von Ereignissen im PTS anhand ihrer Datenverwendung eine nachgelagerte Datenflussannotation entsprechender Petrinetztransitionen mithilfe unterscheidbarer Marken zu ermöglichen, ohne dabei Änderungen im Kontrollfluss des Prozesses vorzunehmen.

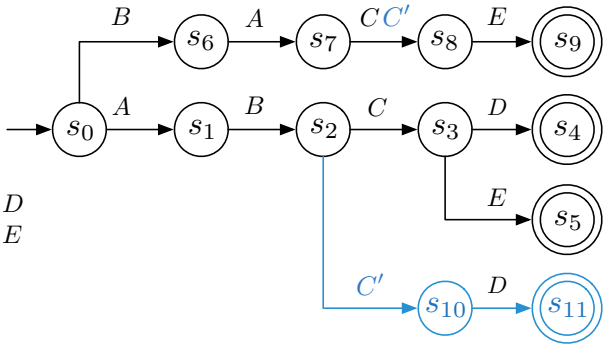
Abb. 5.3 veranschaulicht den Unterschied des erweiterten Konstruktionsverfahrens anhand eines einfachen Prozesslogs mit vier unterschiedlichen Ausführungspfaden. Unter Verwendung der in Abschnitt 4.5.1 vorgestellten PTS-Konstruktion werden lediglich drei unterschiedliche Ausführungspfade betrachtet, da *trace1* und *trace3* als identisch angesehen werden. Zieht man die unterschiedliche Datenverwendung von Aktivität C innerhalb der Ausführungspfade in Betracht, muss das PTS erweitert werden. Diese Unterscheidung erfolgt durch die zusätzliche Aktivität C'. Die durch die Datenbetrachtung veränderten bzw. erweiterten Bereiche des PTS sind blau hervorgehoben.

Prozesslog:

$trace_1 : A[create(d_1)] \rightarrow B \rightarrow C \rightarrow D$
$trace_2 : A[create(d_1)] \rightarrow B \rightarrow C \rightarrow E$
$trace_3 : A[create(d_1)] \rightarrow B \rightarrow C[read(d_1)] \rightarrow D$
$trace_4 : B \rightarrow A[create(d_1)] \rightarrow C[read(d_1)] \rightarrow E$

Datenverwendung:

$A : create(d_1)$
$C' : read(d_1)$



(a) (b)

Abbildung 5.3.: Veranschaulichung des Unterschieds der erweiterten PTS-Konstruktion zur Berücksichtigung von Datenflüssen (hervorgehoben mit blauer Farbe) zum rein kontrollflussorientierten Verfahren (schwarze Farbe).

Die Veränderte PTS-Konstruktion beeinträchtigt die korrekte Funktionsweise von GENET$^+$ nicht. Rekonstruierte Strukturmodelle sind nach wie vor präzise, enthalten jedoch möglicherweise mehrere Transitionen, die derselben Prozessaktivität zugeordnet werden können, sich in ihrer Datenverwendung jedoch unterscheiden.

Für eine korrekte Datenflussannotation des rekonstruierten Prozessmodells muss die Datenverwendung von Prozessaktivitäten spezifiziert werden. Ausgehend von einem PTS $PTS = (S, E, A, s_{in})$, einer Menge von Datenobjekten $\mathcal{O}$ und einer Menge von Datenverwendungsmodi $\mathcal{M} = \{read, write, create, delete\}$, gibt die Funktion $DV_{PTS} : E \times \mathcal{O} \rightarrow \mathcal{P}(\mathcal{M})$ für jede Prozessaktivität $e \in E$ an, auf welche Weise auf einzelne Datenelemente zugegriffen wird. Bzgl. eines rekonstruierten Petrinetzes PN mit einer Transitionsmenge T, entspricht die Datenverwendung einer Transition $t \in T$ der Datenverwendung der Ursprungsaktivität im Prozesslog. Diese Ursprungsaktivität entspricht bei einer Zustandsspaltungsstrategie exakt der Transition selbst. Bei einer Ereignisspaltungsstrategie werden PTS-Ereignisse möglicherweise durch Mengen neuer Ereignisse ersetzt, um die EC-Eigenschaft sicherzustellen. Die Ursprungsaktivität von t ergibt sich in diesem Fall gemäß der in Abschnitt 4.4.1 eingeführten Notation zu $t_{\blacktriangleleft}$. Die Datenverwendung von Transitionen auf Basis von TS rekonstruierten Petrinetzen ist allgemein wie folgt definiert:

$$DV_{PN} : T \times \mathcal{O} \rightarrow \mathcal{P}(\mathcal{M})$$

$$DV_{PN}(t, o) = \begin{cases} DV_{TS}(t, o) & \text{, Zustandsspaltung} \\ DV_{TS}(t_{\blacktriangleleft}, o) & \text{, Ereignisspaltung} \end{cases}$$

Auf Basis der durch ein PNS gegebenen Prozessstruktur und gemäß obiger Definitionen spezifizierter Datenverwendung, kann das PNS durch eine Datenflussannotation in ein IFnet überführt werden, welches eine kombinierte Kontroll- und Datenflusssichtweise auf den rekonstruierten Prozess bietet.

Definition 5.2. ***(Datenflussannotation)*** *Gegeben sei ein aus einem PTS durch Anwendung des GENET⁺-Verfahrens entstandener EC-PPZA* $Z = (S, E, A, s_{in}, F)$. *Sei weiter* $N = (P, T_R, T_S, F, W, m_0)$ *das daraus abgeleitete PNS und* $\mathbf{D}$ *die Menge der verwendeten Datenelemente im Prozesslog. Durch die Datenflussannotation entsteht ein IFnet* $(P', T_R \cup T_S, T_D = \emptyset, F', I, O, C, m'_0, \mathbf{D} \uplus \{black\}, A, AC = \emptyset, \mathfrak{G} = \emptyset)$ *mit folgender Charakteristik:*

(1) $P' := P \cup \{p_d \,|\, d \in \mathbf{D}\}$ *(Spezifische Stellen für Datenelemente)*

(2) $F'_1 := \{(p_d, t) \,|\, p_d \in P' \setminus P\,, t \in T_R,\ DV_N(t, d) \cap \{read, write, delete\} \neq \emptyset\}$
$F'_2 := \{(t, p_d) \,|\, p_d \in P' \setminus P\,, t \in T_R,\ DV_N(t, d) \cap \{read, write, create\} \neq \emptyset\}$
$F' := F \cup F'_1 \cup F'_2$

(3) Für jede Stelle $p \in P'$ *wird der Initialzustand* $m'_0(p)$ *wie folgt definiert:*

- $m'_0(p)(c) = \begin{cases} m_0(p) & , c = black \\ 0 & , sonst \end{cases}$

(4) Für die von der Eingabe- bzw. Ausgabefunktion gelieferten Multimengen gilt:

$$I(p,t)(c) = \begin{cases} W((p,t)) & , c = black \\ 1 & , DV_{PN}(t, c) \neq \emptyset \\ 0 & sonst. \end{cases}$$

$$O(t,p)(c) = \begin{cases} W((t,p)) & , c = black \\ 1 & , DV_{PN}(t, c) \neq \emptyset \\ 0 & sonst. \end{cases}$$

(5) $A(t, c) = DV_{PN}(t, c),\ \forall c \in \mathcal{C}\, \forall t \in T_R$

Für jedes verwendete Datenelement $d \in \mathbf{D}$ wird eine zusätzliche Stelle p_d (Basis von d) eingeführt. Die grundsätzliche Überlegung bzgl. der Modellierung rekonstruierter Datenflüsse ist, dass für d stets genau eine Marke im Netz existiert, deren Typ mit d assoziiert wird und sich diese Marke stets in p_d befindet. Verbindungen zusätzlicher Stellen mit Transitionen werden gemäß der Datenverwendung entsprechender Prozessaktivitäten hergestellt. Nach Verwendung eines Datenelements d (Entnahme der Marke aus p_d durch eine Transition), wird eine Marke gleichen Typs von derselben Transition generiert und in p_d platziert. Datenelemente werden also nach jeder Verwendung „zurück gelegt“. Dadurch ist eine Verwendung desselben Datenelements in späteren Prozessschritten gewährleistet. Einzige Ausnahmen sind:

1. **Erstellen neuer Datenelemente.** Dabei wird ein Marke entsprechenden Typs generiert, aber nicht konsumiert. Es existiert also lediglich eine Verbindung von Transition zur Basis des Datenelements.

2. **Entfernen von Datenelementen.** Dabei wird ein Marke entsprechenden Typs konsumiert, aber nicht generiert. Es existiert also lediglich eine Verbindung von der Basis des Datenelements zur Transition.

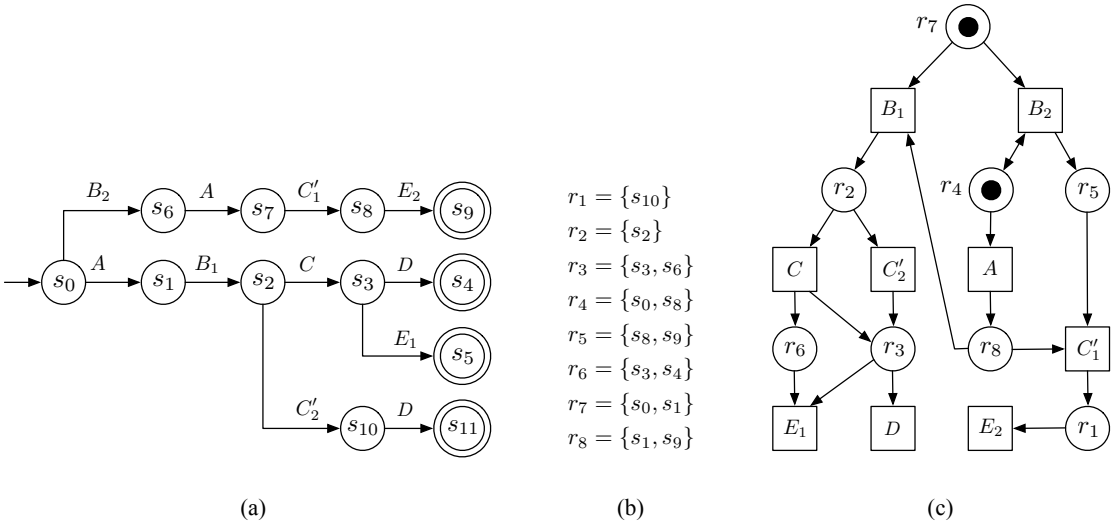


Abbildung 5.4.: (a) EC-Eigenschaft erfüllende Variante des PTS aus Abb. 5.3, (b) zugehörige minimale Regionen, (c) konstruiertes Petrinetz.

Die Anzahl von Kontrollflussmarken, die sich im initialen Zustand in den Stellen des Netzes befinden, werden direkt vom zugrundeliegenden PNS übernommen. Ohne zusätzliche Information zur Bedeutung verwendeter Datenelemente ist nicht klar, anhand welcher Kriterien festgelegt werden soll, ob die Existenz von Marken, deren Typ mit einem Datenelement assoziiert wird, vor Ablauf einer Prozessinstanz notwendig/sinnvoll ist. Deshalb befinden sich solche Marken nicht von Anfang an im Netz und müssen beim Schalten von Transitionen entsprechender Datenverwendung (*create*) explizit erzeugt werden. Diese Strategie erleichtert datenflussorientierte Konsistensprüfungen, bspw.*missing data*, i.e. die Problematik, dass ein Datenelement verwendet wird, zuvor aber nicht erzeugt wurde. Die Datenverwendung von IFnet-Transitionen (A) ergibt sich direkt aus der Datenverwendung des rekonstruierten Netzes DV_{PN}.

Abb. 5.4 enthält die durch die Anwendung von GENET$^+$ entstandene Version des TS in Abb. 5.3 unter Verwendung von Ereignisspaltung. Um die EC-Eigenschaft sicherzustellen, wurden die Ereignisse B, C' und E gespalten. Auf Basis der zugehörigen minimalen Regionen kann schließlich ein präzises Petrinetz konstruiert werden, dessen vollständige Sequenzen exakt den Aktivitätssequenzen im Log entsprechen. Verschiedene Datenverwendungen sind dabei implizit durch die Unterscheidung von Prozessaktivitäten berücksichtigt.

Bei der Annotation der Datenverwendung müssen die durch Spaltung eingeführten Ereignisse wie deren Ursprungsaktivitäten im Log behandelt werden. Die Datenverwendung von Transition C'_1 ergibt sich gemäß obiger Definition zu $DV_{PN}(C'_1) = DV_{PTS}(C'_{1\blacktriangleleft}) = DV_{PTS}(C') = \{read(d_1)\}$. Bei der Überführung des Petrinetzes in Abb. 5.4 in ein IFnet wird eine neue Stelle p_{d_1} eingeführt, die als Basis für Marken dient, deren Typ mit dem Datenelement d_1 assoziiert werden. Im Initialzustand enthält diese Stelle keine Marke, doch mit dem Schalten von Transition A wird eine solche Marke erzeugt und in p_{d_1} abgelegt. Beim Schalten der Transitionen C'_1 und C'_2 kann diese Marke dann verwendet werden (siehe Abb. 5.5).

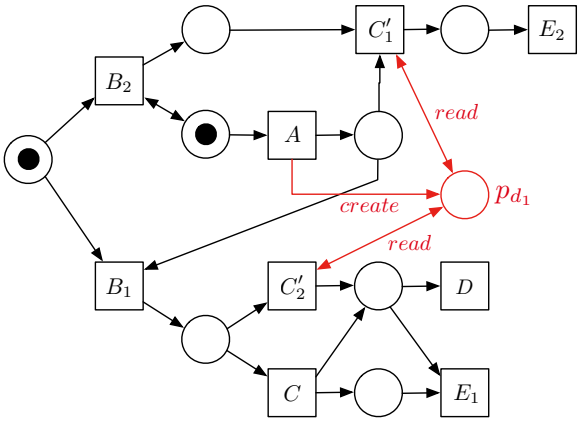


Abbildung 5.5.: Resultat nach Anwendung der Datenflussannotation auf das Netz in Abb. 5.4 und die Datenverwendung in Abb. 5.3.

Durch die Unterscheidung von Prozessaktivitäten anhand ihrer Datenverwendung steigt die strukturelle Komplexität der mittels PTS-Konstruktion erstellten Netze. Die Berechnungskomplexität von GENET$^+$ steigt dadurch nicht notwendigerweise. Verglichen mit einem PTS ohne Berücksichtigung von Datenverwendung hat die erweiterte Variante denselben Effekt wie (mehrfache) Ereignisspaltung, was die Sicherstellung der EC-Eigenschaft sogar vereinfacht. Darüber, ob diese Vereinfachung die durch eine potentiell erhöhte Anzahl von Zuständen bedingte, höhere Berechnungskomplexität hinsichtlich der Berechnung minimaler Regionen kompensiert, lässt sich im Allgemeinen keine Aussage treffen.

Teil II.

Berücksichtigung der Prozessdynamik

Kapitel 6

Erkennung von Prozessveränderungen

Methoden der Prozessrekonstruktion können Modelle aus Prozesslogs erstellen und unter Verwendung der Konzepte aus Abschnitt 4 sogar mit einer für Sicherheitszwecke adäquaten Qualität. Allerdings stellt die Dynamik von Prozessen für Rekonstruktionsmechanismen eine besondere Herausforderung dar. Entgegen der vielfach getroffenen Annahme einer statischen Prozessausführung, besteht die Notwendigkeit, Modelle, die für die Ablaufsteuerung von Prozessen verwendet werden, fortwährend an sich verändernde Rahmenbedingungen anzupassen[1]. Die Ursachen für Veränderungen des Ausführungskontexts von Prozessen sind vielfältig. Neben gesetzlichen Vorgaben, die auf Prozessebene implementiert werden müssen, können auch strategische Entscheidungen eine Umplanung von Abläufen hervorrufen, ebenso wie optimierungsbedingte Anpassungen oder Re-Design-Entscheidungen[2]. Die Möglichkeit der Prozessveränderung ist insofern eine notwendige Voraussetzung für die Flexibilität von Prozessen, als dass Unternehmen ansonsten nicht dynamisch agieren und auf verändernde Geschäftsbedingungen reagieren können. Die Entwicklung adaptiver Prozessmanagement-Technologien (z.B. ADEPT[3] und CBRFlow[4]) und die explizite Betrachtung dynamischer Workflowaspekte in „BPM 2.0“ tragen dieser Notwendigkeit Rechnung[5]. Die Betrachtung der Prozessdynamik ist jedoch nicht nur im Hinblick auf die Aussagekraft rekonstruierter Prozessmodelle sinnvoll. Nachzuvollziehen, wie sich ein Prozess über einen Zeitraum verändert, ist eine Art der Prozessanalyse, die wertvolle Hinweise hinsichtlich Konformität und Optimierungspotential liefern kann. Hinsichtlich der Sicherheit von Geschäftsprozessen kann auf diese Weise z.B. nachvollzogen werden, inwiefern das Eintreten unvorhersehbarer Ereignisse deren Konformität beeinträchtigt oder wie sich die Prozessausführung aufgrund eingeführter Kontrollen verändert hat.

Dieses Kapitel stellt den APClustering-Ansatz vor, der mithilfe tracebasierter Metriken die Dynamik eines Prozesses durch die Erkennung unterschiedlicher Ausführungsphasen nachvollziehbar macht.

[1] Vgl. Schonenberg et al. 2008; C. Ellis et al. 1995.
[2] Vgl. Wil M.P. van der Aalst 2000.
[3] Vgl. Reichert et al. 1998.
[4] Vgl. Weber, Wild et al. 2004.
[5] Vgl. Günther et al. 2007.

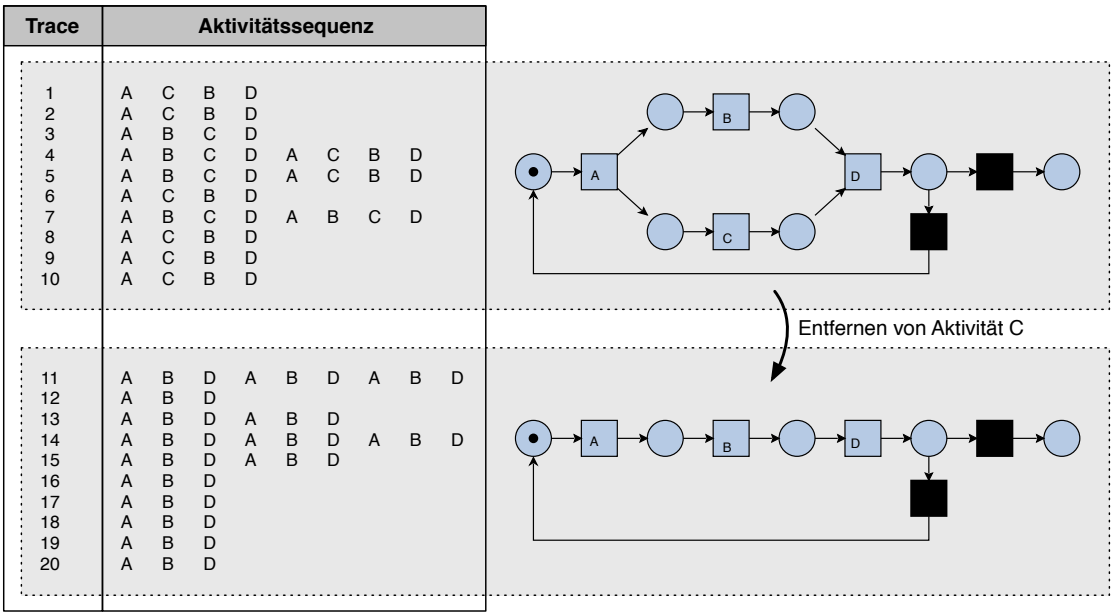


Abbildung 6.1.: Prozesslog mit Bezug zu unterschiedlichen Ursprungsmodellen.

Auf technischer Ebene kann Prozessdynamik als *concept drift* verstanden werden, ein in den Bereichen *Data Mining* und *Künstliche Intelligenz* ausführlich studiertes Paradigma, welches sich allgemein auf den Umstand bezieht, dass Vorhersagen hinsichtlich einer Zielvariablen an Genauigkeit verlieren, wenn sich die statistischen Eigenschaften der Datenbasis verändern. Übertragen auf die Prozessrekonstruktion bedeutet das, dass Veränderungen im geplanten Prozessverlauf die Charakteristik von Ausführungspfaden in Prozesslogs verändern. Rekonstruktionsmechanismen, die stets die Gesamtheit der Ausführungspfade eines Prozesslogs betrachten und mögliche Veränderungen vernachlässigen, subsumieren Pfade unterschiedlicher Ursprungsmodelle und produzieren zwangsläufig inakkurate Modelle. Abb. 6.1 verdeutlicht diese Problematik. Ein präzises Modell auf Basis des dargestellten Prozesslogs erlaubt alle enthaltenen Ausführungspfade und suggeriert auf diese Weise die Optionalität der Aktivität C, die beim Wechsel der Ursprungsmodelle entfernt wurde. Auf diese Weise bezieht sich das subsumierte Verhalten auf keines der verwendeten Ursprungsmodelle, da zu Beginn C stets und nach der Änderung niemals Teil der möglichen Pfade war. Solche „Unschärfen“ können innerhalb von Sicherheitsanalysen zu Fehlinterpretationen führen, die nicht ohne Weiteres aufgedeckt oder vermieden werden können.

Im Process Mining-Bereich wird für den Übergang von einem Ursprungsmodell zu einem anderen zwischen abrupten (*sudden drift*) und sanften Veränderungen (*gradual drift*) unterschieden. Während im ersten Fall der Übergang unvermittelt eintrifft und ab einem bestimmten Zeitpunkt nur noch Pfade des neuen Modells generiert werden, treten bei sanften Veränderungen innerhalb des Übergangszeitraums Pfade beider Ursprungsmodelle im zugehörigen Prozesslog auf. Bisherige Arbeiten zur Erkennung von Prozessveränderungen (auch der hier vorgestellte APClustering-Ansatz) beziehen sich meist auf die Erkennung des Zeitpunktes einer Veränderung (*change point detection*) und konzentrieren sich dabei auf den abrupten Fall. Daneben spielt aber auch die Erkennung der

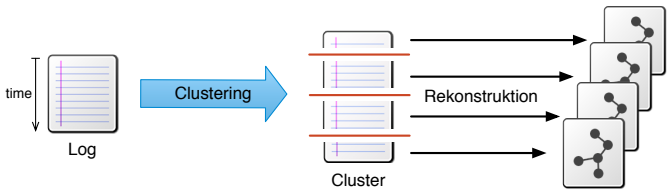


Abbildung 6.2.: Präzise Rekonstruktion durch vorgelagertes Clustering.

Art einer Veränderung und deren Ursprung i.S. der betreffenden Bereiche innerhalb der Struktur eines Prozesses (*change localization*) eine Rolle[6].

Generell muss zwischen externen und internen Prozessveränderungen unterschieden werden. Externe Veränderungen treten ein, wenn ein Ursprungsmodell explizit verändert wird und sich so die potentiell mögliche Menge von Ausführungspfaden ändert. Interne Veränderungen beziehen sich auf sog. Änderungsphasen von Prozessmodellen, die bspw. durch die Bevorzugung einer Teilmenge aller möglichen Pfade über einen bestimmten Zeitraum zustande kommen. Es ist nicht möglich, im Einzelfall zwischen externen und internen Veränderungen zu unterscheiden, da sich die Auswirkungen auf Prozesslog-Charakteristika nicht unterscheiden. Aus diesem Grund arbeiten einige existierende Methoden mit Vorwissen bzgl. der Anzahl unterschiedlicher Ursprungsmodelle, um die Aussagekraft berechneter Cluster zu erhöhen und das Partitionierungsverfahren zu lenken. Das hier vorgestellte Verfahren arbeitet ohne Vorwissen und verfolgt einen interaktiven Ansatz, der dem Benutzer erlaubt, durch verschiedene Phasen der Prozessausführung zu „navigieren“ und so ein Gefühl für die Dynamik eines Prozesses zu entwickeln. Änderungsphasen werden auf Basis unterschiedlicher Metriken hinsichtlich der Distanz zwischen Aktivitätspaaren erkannt und dargestellt.

Die Integration von Mechanismen zur Erkennung von Prozessänderungen in die Prozessrekonstruktion (siehe Abb. 6.2) sieht ein vorgelagertes Clustering (Zerteilen) eines Prozesslogs anhand von Änderungszeitpunkten mit anschließender, separater Rekonstruktion auf Basis einzelner Cluster vor. Das Ziel dabei ist, die Menge der Ausführungspfade im Log zu partitionieren, wobei jede Partition für eine Ausführungsphase des betrachteten Prozesses steht. Separate Rekonstruktionen führen dann im Optimalfall zu präzisen Modellen einzelner Phasen.

6.1 Bestehende Clusteringverfahren

Die Zielsetzung von Clusteringverfahren besteht in einer Partitionierung eines gegebenen Prozesslogs. Innerhalb des Process Mining-Forschungsbereichs werden für dieses Unterfangen meist die Begriffe *Log Clustering*, oder *Trace Clustering* verwendet. Die Notwendigkeit für die Partitionierung von Prozesslogs wird traditionell mit der Problematik begründet, dass speziell bei Prozesslogs eher unstrukturierter Prozesse die Rekonstruktion typischerweise sehr komplexe und damit schwer verständliche, sog. „Spaghetti-Modelle“ hervorbringt. Durch die Partitionierung von Prozesslogs in Cluster verspricht man sich

[6] Vgl. Bose, Aalst et al. 2011.

Verbesserungen in der Qualität und Interpretierbarkeit von auf Basis einzelner Cluster separat rekonstruierter Modelle. Die Beobachtung, dass Prozesse in der Praxis meist unstrukturierter ablaufen als gedacht/geplant[7] ist ein weiterer Grund für die Betrachtung des Log Clustering-Problems.

Ausgehend von einem Prozesslog werden Cluster (bestehend aus Ausführungspfaden) gebildet, die sich im Hinblick auf konkrete Ähnlichkeitsmetriken nahe stehen. Ausführungspfade eines Clusters weisen ähnliches Prozessverhalten auf und definieren dabei sog. Verhaltensmuster (engl.: *behavioral patterns*).

Eine weitere Analyse bzw. Rekonstruktion erfolgt dann für jedes Cluster separat. Metriken zur Ähnlichkeit von Ausführungspfaden beziehen sich dabei entweder auf die Pfade selbst oder auf daraus generierte „Profile“. Ein Profil beschreibt einen Pfad aus einer bestimmten Perspektive und verwendet dafür eine Reihe von „Features“, für die typischerweise jeweils ein numerischer (normalisierter) Wert bestimmt wird. Ein Feature kann sich bspw. auf die unterschiedlichen Aktivitäten eines Ausführungspfades beziehen und als n-dimensionaler Vektor beschrieben werden, der für jede unterschiedliche, im Prozesslog auftretende Aktivität die Anzahl derer Vorkommen innerhalb eines Pfades enthält. Bisherige Arbeiten betrachten hinsichtlich der Ähnlichkeit von Ausführungspfaden hauptsächlich den Kontrollfluss und vernachlässigen dabei temporale Aspekte. Zur Bestimmung von Clustern auf Basis von Featurevektoren kommen gängige Verfahren wie *k-means*, Kohonennetze (unüberwachtes Lernverfahren auf Basis künstlicher neuronaler Netze) oder Methoden der hierarchischen Clusteranalyse, die nicht nur eine Partitionierung liefern, sondern Mengen von Partitionen, die in hierarchischen Beziehungen zueinander stehen können. Hierarchische Clusteringverfahren arbeiten entweder *agglomerativ* (bottom-up) oder *dividiv* (top-down). Während im agglomerativen Fall für jeden Ausführungspfad ein eigenes Cluster erstellt und diese schrittweise vereinigt werden, existiert im dividiven Fall zunächst ein Cluster, das die Gesamtheit aller Ausführungspfade enthält und schrittweise verfeinert/aufgespalten wird.

Tracebasierte Verfahren

Tracebasierte Verfahren arbeiten direkt auf Ausführungspfaden und nicht auf zuvor extrahierten Featurevektoren. Das Verfahren von Veiga und Ferreira[8] beschreibt Cluster in Form von Markov-Ketten. Für eine gegebenes k werden zunächst k Markov-Ketten mit zufällig gewählten Wahrscheinlichkeiten für Zustandsübergänge gebildet. Bis ein Haltekriterium erfüllt ist, werden dann in aufeinander folgenden Schritten jeweils alle Ausführungspfade der Kette mit der höchsten Wahrscheinlichkeit für deren Produktion zugeteilt und die Übergangswahrscheinlichkeiten der Ketten anschließend auf Basis der zugeordneten Pfade aktualisiert. Insofern gruppiert der Ansatz ähnliche Ausführungspfade ohne dabei spezifische Ähnlichkeitsmetriken zu verwenden.

Inspiriert von Verfahren zur Gruppierung ähnlicher DNA-Sequenzen, nutzt der *sequence clustering*-Ansatz von Bose et al.[9] *Alignments* für die Clusterbildung. Ein Alignment von

[7] Vgl. Bose und Aalst 2010.
[8] Vgl. Veiga et al. 2009.
[9] Vgl. Bose und Aalst 2010.

zwei Pfaden t_1 und t_2 entsteht durch die Transformation von t_1 in t_2 durch Hinzunahme, Entfernung oder Austausch einzelner Aktivitäten. Analog zu gängigen Verfahren zur Bestimmung der Editierdistanz ergibt sich die Distanz zwischen beiden Pfaden aus den minimalen Kosten für diese Transformation. Optimale Alignments (mit minimalen Kosten) werden unter Verwendung einer Metrik von Needleman et al.[10] bestimmt. Für die Erstellung von Clustern auf Basis berechneter Distanzen wird dann ein agglomeratives Verfahren verwendet. Ein weiteres tracebasiertes Verfahren von Bose et al.[11] nutzt ebenfalls die Editierdistanz (Levenshtein-Distanz) zwischen als Zeichenketten interpretierten Ausführungspfaden für deren Ähnlichkeitsbestimmung, konzentriert sich dabei aber auf gemeinsame Teilsequenzen.

Featurebasierte Verfahren

Greco et al.[12] beschreiben einen mehrstufigen, iterativen bottom-up-Ansatz. Ausgangspunkt ist der gesamte Prozesslog, auf den ein Rekonstruktionsverfahren angewandt wird. Falls die Qualität des resultierenden Modells nicht ausreichend ist (z.B. bzgl. Deckung oder Affinität), wird die Menge der Ausführungspfade partitioniert und im nächsten Ausführungsschritt für jede der Partitionen analog weitergeführt. Für die Erstellung der Cluster wird das *k-means*-Verfahren auf zuvor extrahierte Features angewandt. Die Anzahl der Cluster ergibt sich dabei aus der Anzahl unterschiedlicher, durch Features charakterisierter Verhaltensmuster. Verfahren wie *multi-phase process mining*[13], deren Arbeitsweise durchaus ähnlich ist, sind keine Clusteringverfahren im engeren Sinne. Zwar verfeinern sie rekonstruierte Modelle ebenfalls schrittweise, nehmen dabei jedoch keine explizite Partitionierung des Prozesslogs vor.

Im Unterschied zur iterativen Vorgehensweise verfolgen Song et al.[14] eine klassische featurebasierte Strategie. Ihr Trace Clustering-Ansatz beinhaltet eine Reihe von Features, die sich sowohl auf den Kontrollfluss (Anzahl von Ereignissen innerhalb von Pfaden und paarweiser Nachfolgerelationen), die Datenverwendung (dem Pfad oder Einzelaktivitäten zugeordnete Datenattribute) und Kontextinformation (beteiligte Akteure), als auch auf temporale Aspekte (Ausführungsdauer von Instanzen, Dauer von Aktivitäten etc.) beziehen. Verwendete Metriken für die Bestimmung der Ähnlichkeit von Featurevektoren beinhalten die euklidische Distanz (geometrische Interpretation), die Hamming-Distanz (Zeichenketten-Interpretation), sowie die Jaccard-Distanz (Mengen-Interpretation). Für die Clusterbildung wird auf oben vorgestellte Verfahren zurückgegriffen (*k-means, Kohonennetze, ...*). Die verwendeten Features decken zwar eine Vielzahl der in Prozesslogs typischerweise vorzufindenden Eigenschaften von Traces ab, sind aber nicht dazu geeignet, Cluster zu erzeugen, die verschiedene zeitlich aufeinander folgende Ausführungsphasen eines Prozesses charakterisieren.

[10] Vgl. Needleman et al. 1970.
[11] Vgl. Bose und Aalst 2009a.
[12] Vgl. Alves de Medeiros, Guzzo et al. 2007; Greco et al. 2006.
[13] Vgl. Dongen und Aalst 2004; Dongen und Aalst 2005b.
[14] Vgl. Song et al. 2008.

Das Verfahren von Bose et al.[15] überträgt den tracebasierten Teilsequenzen-Ansatz derselben Autoren (siehe oben) auf Featurevektoren. Die Idee der *conserved patterns* besteht dabei aus der Betrachtung von Teilsequenzen über Tracegrenzen hinweg. Die Annahme dabei besteht darin, dass Ursprungsmodelle von Instanzen mit hoher Anzahl gemeinsamer Teilsequenzen strukturelle Ähnlichkeiten aufweisen.

Die Arbeiten von Bose et al.[16] und Luengo et al.[17] beschreiben die einzigen existierenden Clusteringansätze mit explizitem Bezug zu Concept-Drift und der Erkennung zeitlich aufeinander folgender Ausführungsphasen eines Prozesses. Luengo und Sepúlveda bauen auf dem Ansatz von Bose et al.[18] auf und führen zur Berücksichtigung temporaler Eigenschaften einen weiteren Featurevektor hinzu, der den Startzeitpunkt von Ausführungspfaden i.S. der vergangenen Zeit seit einem Stichzeitpunkt betrachtet. Allerdings ist nicht klar, ob bei dem Verfahren die Clusteranzahl bereits vor der Partitionierung bestimmt werden muss. Durchgeführte Experimente der Autoren zur Funktionstüchtigkeit des Verfahrens legen diese Vermutung nahe.

Bose et al.[19] folgen der Überlegung, dass sich Änderungen im Prozessmodell in Vorgänger- und Nachfolgerelationen unter Aktivitäten manifestieren und definieren deshalb zunächst Features, die sich auf solche Relationen beziehen. Mithilfe statistischer Hypothesentests (*Kolmogorov-Smirnov*, *Mann-Whitney* und *Hotelling* T^2) wird die Ähnlichkeit von Featurevektoren benachbarter Sequenzen (Samples) bestimmt. Änderungszeitpunkte ergeben sich dann aus Samplegrenzen, bei denen die Ähnlichkeit angrenzender Sequenzen unter einem Schwellwert liegen. Dem in diesem Kapitel vorgestellten Verfahren liegt ein ähnlicher Gedanke zugrunde, doch statt Vorgänger-/Nachfolgerelationen stehen Distanzen zwischen Einzelaktivitäten (Anzahl dazwischenliegender Aktivitäten) innerhalb von Ausführungspfaden im Fokus. Durch die Analyse von Varianzen solcher Distanzen wird auf wahrscheinliche Zeitpunkte für den Übergang von einer Ausführungsphase eines Prozesses zur nächsten geschlossen.

6.2 APClustering: Distanzbasierte Erkennung von Änderungszeitpunkten

Das APClustering-Verfahren versucht Änderungen durch die Beobachtung von Distanzen zwischen Paaren von Prozessaktivitäten im zeitlichen Verlauf zu erkennen. Eine Änderung bezieht sich stets auf ein bislang als „normal“ aufgefasstes Verhalten und wird in diesem Zusammenhang als Änderung der bisher observierten unterschiedlichen Distanzen interpretiert. Dabei können Änderungen nicht nur durch einen Wechsel des zugrundeliegenden Modells für die Ablaufsteuerung eines Prozesses hervorgerufen werden, sondern auch durch unterschiedliche Ausführungsphasen innerhalb desselben Modells. Tritt innerhalb eines Betrachtungszeitraums nur ein Bruchteil des prinzipiell möglichen Verhaltens in Erscheinung, kann dies zu Veränderungen von Distanzen zwischen Aktivitätspaaren

[15] Vgl. Bose und Aalst 2009b.
[16] Vgl. Bose, Aalst et al. 2011.
[17] Vgl. Luengo et al. 2011a; Luengo et al. 2011b.
[18] Vgl. Bose und Aalst 2009b.
[19] Vgl. Bose, Aalst et al. 2011.

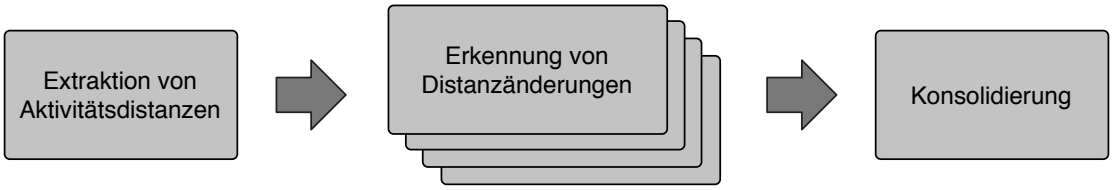


Abbildung 6.3.: Phasen des APClustering-Ansatzes.

führen, die dann möglicherweise als Änderung aufgefasst werden. Im Einzelfall kann die Ursache einer erkannten Änderung also nicht eindeutig bestimmt werden. Diese Beobachtung ist kennzeichnend für die Erkennung von Prozessänderungen auf Basis aufgezeichneten Verhaltens und nicht spezifisch für distanzbasierte Verfahren.

APClustering ist ein dreistufiges Verfahren (siehe Abb. 6.3). In Phase 1 wird in einem Vorverarbeitungsschritt der Prozesslog durchlaufen und eine Matrix erstellt, die für jedes Aktivitätspaar die observierten Distanzen innerhalb jedes Ausführungspfades enthält. Zeitpunkte, bei denen Änderungen bzgl. der bisher üblichen Distanzen beobachtbar sind, werden dann in Schritt 2 für jedes Aktivitätspaar separat berechnet. Dabei entsteht für jedes Aktivitätspaar eine Liste von Zeitpunkten, an denen Distanzänderungen beobachtbar sind. Diese Listen werden schließlich in Schritt 3 aggregiert, um Zeiträume zu identifizieren, innerhalb derer Änderungen als wahrscheinlich betrachtet werden.

6.2.1 Extraktion von Aktivitätsdistanzen

Dieser Abschnitt definiert das Konzept „Distanz“ und stellt einen Algorithmus zur Extraktion von Distanzen zwischen Aktivitätspaaren aus einem Prozesslog gemäß Def. B.2 vor.

Definition 6.1 (Aktivitätsdistanz). *Gegeben sei ein Ausführungspfad* $\sigma = \langle a_1, ..., a_n \rangle$ *mit* $a_i \in A, \forall i \in \{1, \ldots, n\}$. *Die Distanz eines Paares von Aktivitäten* (a_j, a_k), $a_j, a_k \in \sigma$, $k, j \in [1; n], k > j$ *ergibt sich zu* $d_\sigma(a_j, a_k) := k - j - 1$. ⊣

Die Distanz zwischen zwei Aktivitäten innerhalb eines Ausführungspfades wird gemäß Def. 6.1 als Anzahl dazwischen auftretender Aktivitäten definiert. Bezugnehmend auf Pfad 1 aus Abb. 6.1 ergibt sich die Distanz zwischen den Aktivitäten A und C zu 0 und die Distanz zwischen A und B zu 1. Prinzipiell ist es möglich, dass aufgrund von Schleifen oder mehrfach auftretenden Aktivitäten innerhalb des Prozessmodells, welches zur Generierung von Ausführungspfaden verwendet wird, Aktivitäten innerhalb eines Pfades mehrfach auftreten. In diesem Fall existieren möglicherweise auch mehrere Distanzen.

Definition 6.2 (Aktivitätsdistanzen innerhalb von Ausführungspfaden). *Gegeben sei ein Ausführungspfad $\sigma = \langle a_1, ..., a_n \rangle$ mit $\forall i : a_i \in A$. Die Vorkommen eines Aktivitätspaares (a', a'') innerhalb σ ergeben sich zu $P_{a',a''}(\sigma) := \{(i,j) \mid a_i{=}a', a_j{=} a'', a_i, a_j \in \sigma \wedge \{a_{i+1}, \ldots, a_{j-1}\} \cap \{a_i, a_j\} = \emptyset \wedge i < j\}$ und die entsprechenden Distanzen zu*

$$D_\sigma(a', a'') := \bigcup_{(i,j) \in P_{a',a''}(\sigma)} d_\sigma(\sigma[i], \sigma[j])$$

⊣

Grundsätzlich werden nur Distanzen solcher Aktivitäten a_i, a_j betrachtet, zwischen denen keine Aktivität existiert, die a_i oder a_j entspricht. Innerhalb eines Pfades $\sigma = \langle A, B, C, A, D, B, C \rangle$ ergibt sich die Menge von Distanzen zwischen A und B gemäß Def. 6.2 zu $D_\sigma(A, B) = \{0, 1\}$. Die Distanz zwischen dem ersten A und dem letzen B wird vernachlässigt. Die zugrundeliegende Überlegung ist hierbei, dass insbesondere Schleifen die Existenz wiederholter, aufeinander folgender Aktivitätspaare bedingen und eine Distanzbetrachtung über verschiedene Schleifendurchläufe hinweg nicht sinnvoll erscheint. Für das Zustandekommen des Beispielpfades σ könnte bspw. ein möglicher Rücksprung von C nach A innerhalb des Ursprungsmodells von σ verantwortlich sein.

Distanzen von Aktivitätspaaren werden in einer Distanzmatrix gespeichert. Auf dieses hauptsächlich Implementierungszwecken dienliche Konstrukt wird an dieser Stelle dennoch eingegangen, da die relative Größe einer Distanzmatrix zur Abschätzung der Komplexität eines Prozesslogs verwendet werden kann.

Definition 6.3 (Distanzmatrix). *Gegeben sei ein Prozesslog $L = \langle \sigma_1, ..., \sigma_m \rangle$ und mit A die Menge unterschiedlicher Aktivitäten innerhalb von L. Eine Distanzmatrix M^L ist eine 3-dimensionale $A \times A \times m$ Matrix, die für jedes Paar $a', a'' \in A$ und jeden Ausführungspfad σ_i die entsprechende Menge unterschiedlicher Distanzen des Paars enthält, i.e. $M^L_{a',a'',\sigma_i} := D_{\sigma_i}(a', a'')$.* ⊣

Die relative Größe einer Distanzmatrix gibt das Verhältnis der Anzahl von Aktivitätspaaren mit observierten Distanzen zur Anzahl prinzipiell möglicher Aktivitätspaare an und ist ein Maß für die Vollständigkeit der Matrix.

Definition 6.4 (Relative Größe einer Distanzmatrix). *Die relative Größe einer Distanzmatrix M^L misst deren Vollständigkeit im Hinblick auf Aktivitätspaare mit observierten Distanzen und ist folgendermaßen definiert:*

$$rel(M^L) := \frac{|\mathbb{P}^L|}{|A|^2}, \text{ wobei}$$

$$\mathbb{P}^L := \left\{ (a', a'') \mid \left(\bigcup_{\sigma \in L} D_\sigma(a', a'') \right) \neq \emptyset \right\}$$

⊣

Typischerweise treten innerhalb eines Prozesslogs deutlich weniger Aktivitätspaare in Erscheinung als prinzipiell möglich wären und es gilt $rel(M^L) \ll 1$. Die relative Größe einer Distanzmatrix ist abhängig von der Komplexität des Prozesslogs. Die Anzahl von

Algorithmus 6.1 Konstruktion der Distanzmatrix

```
1:  procedure BuildDistanceMatrix(Prozesslog L)
2:      for all σ ∈ L do
3:          for first = 1, |σ| − 1 do
4:              for second = first + 1, |σ| do
5:                  M^L_{σ[first],σ[second],σ} ← M^L_{σ[first],σ[second],σ} ∪ (second − first − 1)
6:                  if σ[first] = σ[second] then break
7:                  end if
8:              end for
9:          end for
10:     end for
11: end procedure
```

Aktivitäten, die im Ursprungsmodell in paralleler Abhängigkeit zueinander stehen spielt bzgl. des Parallelitätsgrades einer Distanzmatrix eine maßgebliche Rolle.

Definition 6.5 (Parallelitätsgrad einer Distanzmatrix). *Der Parallelitätsgrad einer Distanzmatrix M^L misst die Komplexität eines zugrundeliegenden Prozesslogs i.S. paralleler Aktivitäten und ist folgendermaßen definiert:*

$$par(M^L) := \frac{|\mathbb{U}^L|}{|A|^2}, \textit{ wobei}$$

$$\mathbb{U}^L := \left\{(a', a'') \,|\, (a', a'') \in \mathbb{P}^L \wedge (a'', a') \in \mathbb{P}^L\right\} \qquad \dashv$$

Zur Erstellung der Distanzmatrix durchläuft Algorithmus Alg. 6.1 einen gegebenen Prozesslog sequentiell und extrahiert aus jedem Ausführungspfad die Distanzen aller enthaltener Aktivitätspaare. Innerhalb eines Pfades werden mithilfe der Indexe *first* und *second* alle Aktivitätspaare gebildet und deren Distanz bestimmt, wobei mithilfe des Abbruchkriteriums in Zeile 6 sichergestellt wird, dass nur valide Paare gemäß Def. 6.2 gebildet werden. Die resultierende Distanzmatrix für den Prozesslog in Abb. 6.1 wird von Abb. 6.4 visualisiert. Zeile I_1 enthält für jedes Aktivitätspaar ein Intervall, in das observierte Distanzen innerhalb der ersten 10 Pfade fallen, Zeile I_2 entsprechend für die letzten 10 Pfade.

6.2.2 Erkennung von Distanzänderungen

Im Falle struktureller Prozessveränderungen kann sich die Distanz zwischen einzelnen Aktivitätspaaren ändern. In Abb. 6.4 ist deutlich zu erkennen, dass die Entfernung der Aktivität C nach Pfad 10 zu neuen Ausführungspfaden führt, innerhalb derer die Distanzen zwischen zuvor betrachteten Aktivitätspaaren von bisher als üblich betrachteten Werten (gegeben durch die entsprechenden Intervalle) abweichen. Das Ziel des APClustering-Verfahrens besteht darin, Veränderungen von Distanzintervallen aufeinander folgender Sequenzen von Ausführungspfaden zu erkennen. Dabei werden die in Abb. 6.5 visualisierten Intervalländerungen unterschieden. Gruppe (K1) steht für Veränderungen,

Pfad	(D,D)	(D,A)	(D,B)	(D,C)	(A,D)	(A,A)	(A,B)	(A,C)	(B,D)	(B,A)	(B,B)	(B,C)	(C,D)	(C,A)	(C,B)	(C,C)
1	∅	∅	∅	∅	{2}	∅	{1}	{0}	{0}	∅	∅	∅	{1}	∅	{0}	∅
2	∅	∅	∅	∅	{2}	∅	{1}	{0}	{0}	∅	∅	∅	{1}	∅	{0}	∅
3	∅	∅	∅	∅	{2}	∅	{0}	{1}	{1}	∅	∅	{0}	{0}	∅	∅	∅
4	{3}	{0}	{2}	{1}	{2}	{3}	{0, 1}	{0, 1}	{0, 1}	{2}	{4}	{0, 3}	{0, 1}	{1}	{0}	{2}
5	{3}	{0}	{2}	{1}	{2}	{3}	{0, 1}	{0, 1}	{0, 1}	{2}	{4}	{0, 3}	{0, 1}	{1}	{0}	{2}
6	∅	∅	∅	∅	{2}	∅	{1}	{0}	{0}	∅	∅	∅	{1}	∅	{0}	∅
7	{3}	{0}	{1}	{2}	{2}	{3}	{0}	{1}	{1}	{2}	{3}	{0}	{0}	{1}	{2}	{3}
8	∅	∅	∅	∅	{2}	∅	{1}	{0}	{0}	∅	∅	∅	{1}	∅	{0}	∅
9	∅	∅	∅	∅	{2}	∅	{1}	{0}	{0}	∅	∅	∅	{1}	∅	{0}	∅
10	∅	∅	∅	∅	{2}	∅	{1}	{0}	{0}	∅	∅	∅	{1}	∅	{0}	∅
I_1	[3,3]	[0,0]	[1,2]	[1,2]	[2,2]	[2,3]	[0,1]	[0,1]	[0,1]	[2,2]	[3,4]	[0,3]	[0,1]	[1,1]	[0,2]	[2,3]
11	{2}	{0}	{1}	-	{1}	{2}	{0}	-	{0}	{1}	{2}	-	-	-	-	-
12	∅	∅	∅	-	{1}	∅	{0}	-	{0}	∅	∅	-	-	-	-	-
13	{2}	{0}	{1}	-	{1}	{2}	{0}	-	{0}	{1}	{2}	-	-	-	-	-
14	{2}	{0}	{1}	-	{1}	{2}	{0}	-	{0}	{1}	{2}	-	-	-	-	-
15	{2}	{0}	{1}	-	{1}	{2}	{0}	-	{0}	{1}	{2}	-	-	-	-	-
16	∅	∅	∅	-	{1}	∅	{0}	-	{0}	∅	∅	-	-	-	-	-
17	∅	∅	∅	-	{1}	∅	{0}	-	{0}	∅	∅	-	-	-	-	-
18	∅	∅	∅	-	{1}	∅	{0}	-	{0}	∅	∅	-	-	-	-	-
19	∅	∅	∅	-	{1}	∅	{0}	-	{0}	∅	∅	-	-	-	-	-
20	∅	∅	∅	-	{1}	∅	{0}	-	{0}	∅	∅	-	-	-	-	-
I_2	[2,2]	[0,0]	[1,1]	-	[1,1]	[2,2]	[0,0]	-	[0,0]	[1,1]	[2,2]	-	-	-	-	-
Δ	(C4)	-	(C6)	(C9)	(C4)	(C6)	(C6)	(C9)	(C6)	(C4)	(C4)	(C9)	(C9)	(C9)	(C9)	(C9)

Abbildung 6.4.: Distanzmatrix für den Prozesslog in Abb. 6.1.

bei denen entweder die Obergrenze des bisherigen Intervalls überschritten oder dessen Untergrenze unterschritten wird (ein Intervall kann entweder vergrößert (Extension) oder verlagert werden). Während sich Gruppe (K2) auf Kontraktion (Schmälerung) von Intervallen konzentriert, beziehen sich ($C9$) und ($C10$) auf Effekte globalerer Natur. Werden im Rahmen einer Prozessänderung Aktivitäten entfernt oder hinzugefügt, führt dies entweder dazu, dass neue, bisher nicht bekannte Aktivitätspaare auftreten oder für bestimmte Aktivitätspaare ab dem Änderungszeitpunkt keine Distanzen mehr auftreten. Abb. 6.1 enthält konkrete Beispiele unterschiedlicher Intervalländerungen. Im Fall von Aktivitätspaar (A, B) wird das Distanzintervall bspw. von $[0; 1]$ auf $[0]$ reduziert, weil es durch die Hinwegnahme von C innerhalb des Prozesses keine Möglichkeit für das Auftreten einer weiteren Aktivität zwischen A und B gibt (vgl. Abb. 6.1).

Zur Feststellung üblichen Verhaltens i.S. von Distanzen werden für jedes Aktivitätspaar zunächst die ersten ω Ausführungspfade des Prozesslogs betrachtet und dafür ein Distanzintervall gebildet. Diese Menge von Ausführungspfaden bildet das initiale Cluster, welches sukzessive erweitert wird. Solange das Distanzintervall des betrachteten Paares in folgenden Ausführungspfaden nicht verändert wird, wird der aktuell betrachtete Pfad in das aktuelle Cluster übernommen. Dafür wird im Einzelfall ein *Lookahead* der Größe ω konsultiert, der das Distanzintervall der folgenden ω Pfade enthält. Dadurch wird eine Überprüfung der Beständigkeit einer Distanzänderung und damit erst die Erkennung von Intervalländerungen der Kategorie ($K2$) ermöglicht. Im Fall erkannter Änderungen wird das aktuelle Cluster abgeschlossen und das ab diesem Zeitpunk als normal aufgefasste Prozessverhalten durch die Betrachtung der folgenden ω Pfade neu berechnet bzw. vom Lookahead übernommen. Das Verfahren wird dann auf Basis des dadurch entstandenen Folgeclusters analog fortgeführt.

Der Parameter ω wird als Fenstergröße aufgefasst, da er für die Mindestanzahl betrachteter Aktivitätspfade für die Feststellung üblichen Prozessverhaltens verwendet wird und

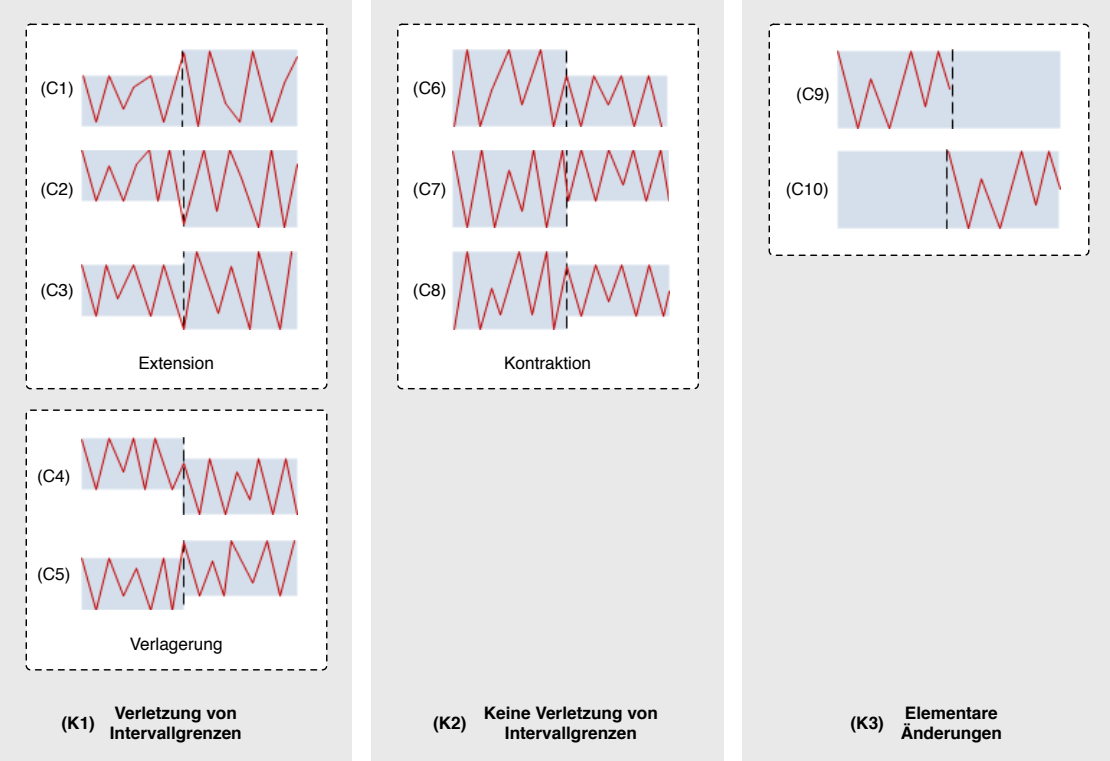


Abbildung 6.5.: Veränderungstypen für Distanzintervalle von Aktivitätspaaren.

damit ein „Fenster“ impliziert, welches mit Fortschreiten des Verfahrens sukzessive über den Prozesslog bewegt und währenddessen erweitert oder neu aufgebaut wird. Während sich die Größe des aktuellen Clusters im Verlauf erhöhen kann, bleibt die Größe des Lookaheads konstant.

Abb. 6.6 verdeutlicht die Arbeitsweise des APClustering-Verfahrens zur Erkennung von Änderungszeitpunkten bzgl. eines einzelnen Aktivitätspaares und beschreibt zwei mögliche Ausgänge (Erweiterung oder neue Clusterbildung) bei der Observierung eines Ausführungspfades t_i. Das initiale Cluster enthält zum visualisierten Zeitpunkt ω viele Pfade.

Für eine detaillierte Erläuterung der Arbeitsweise von APClustering werden zunächst Sequenzen von Ausführungspfaden innerhalb von Prozesslogs (sog. Samples) definiert.

Definition 6.6 (Sample eines Prozesslogs). *Sei* $L = \langle \sigma_1, ..., \sigma_n \rangle$ *ein Prozesslog. Dann ist* $W^L_{s,t} = \langle \sigma_s, ..., \sigma_t \rangle \subseteq L$ *ein Sample von* L *mit Größe* $|W^L_{s,t}| = t - s + 1$. *Die Menge der Distanzen eines Aktivitätspaares* (a', a'') *innerhalb* W *ergeben sich zu* $D_W(a', a'') := \bigcup_{\sigma \in W} D_\sigma(a', a'')$. ⊣

Definition 6.7 (Support von Distanzen innerhalb von Samples). *Der Support einer Distanz* $\mathbf{d}$ *eines Aktivitätspaares* (a', a'') *innerhalb eines Samples* W *ist definiert als*

$$supp_{a',a''}(W, \mathbf{d}) := \sum_{\sigma \in W} |\{(i,j) \in P_{a',a''}(\sigma) \mid d_\sigma(\sigma[i], \sigma[j]) = \mathbf{d}\}|$$

⊣

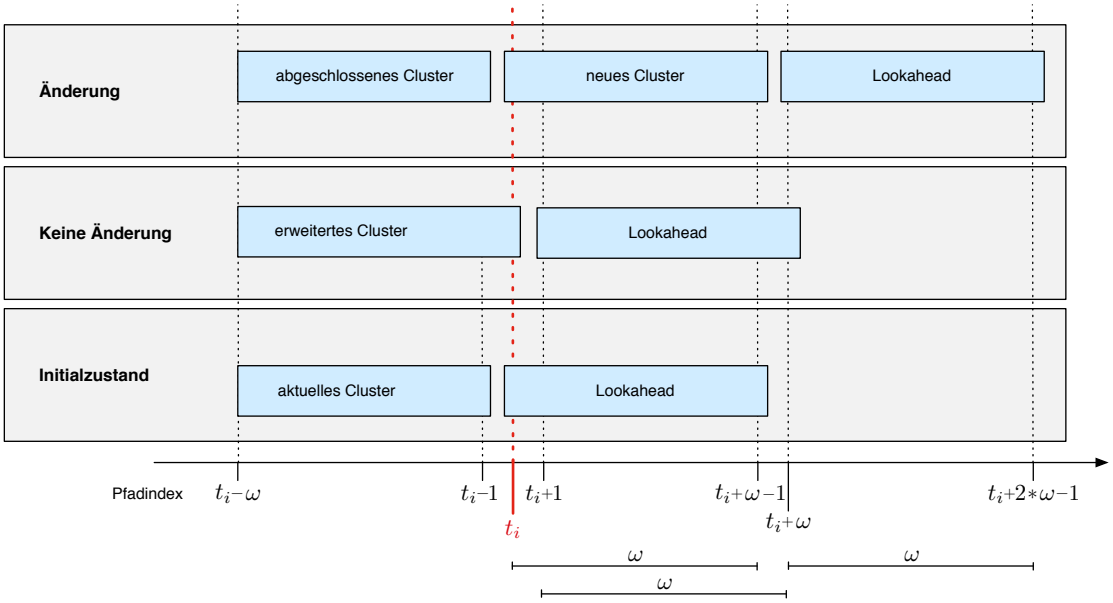


Abbildung 6.6.: Arbeitsweise des APClustering-Verfahrens.

Distanzen von Aktivitätspaaren innerhalb von Samples werden mithilfe von Distanzintervallen (Def. 6.8) erfasst. Dabei wird $I.min$ und $I.max$ als abkürzende Schreibweise für die Unter- bzw. Obergrenze eines Distanzintervalls I verwendet, die Größe von I ergibt sich zu $|I| := I.max - I.min$. Die Relation $\leq$ ist auf Paaren von Distanzintervallen I_1, I_2 wie folgt definiert: $I_1 \leq I_2 \Leftrightarrow I_1.min \geq I_2.min \wedge I_1.max \leq I_2.max$. Für $I_1 < I_2$ gilt zusätzlich $|I_1| < |I_2|$ (i.e. I_1 ist in I_2 enthalten).

Definition 6.8 (Distanzintervall). *Sei $W = \langle \sigma_s, ..., \sigma_t \rangle$ ein Sample eines Prozesslogs. Dann ergibt sich die minimale Distanz zwischen zwei Aktivitäten a' und a'' innerhalb von W zu $min_{a',a''}(W) := min(\bigcup_{\sigma \in W} D_\sigma(a', a''))$; die maximale Distanz entsprechend zu $max_{a',a''}(W) := max(\bigcup_{\sigma \in W} D_\sigma(a', a''))$. Das dadurch implizierte Distanzintervall ist wie folgt definiert:*

$$W[a', a''] := [min_{a',a''}(W); max_{a',a''}(W)] \qquad \dashv$$

Bei der Betrachtung eines Samples $act_{t_i,x} := W_{t_i-x,t_i-1}$, welches alle bisher untersuchten Ausführungspfade enthält, wird für jedes Aktivitätspaar (a', a'') der nächste Pfad t_i dahingehend untersucht, ob er eine Distanz enthält, die auf eine Intervalländerung schließen lässt. Dabei gilt grundsätzlich $x \geq \omega$. Abb. 6.6 visualisiert ein Sample mit $x = \omega$, das in dieser Form im Initialzustand des Verfahrens oder nach der Einführung eines neuen Clusters zur erneuten Bestimmung typischen Prozessverhaltens konstruiert wird.

Wie bereits erwähnt, wird zur Unterscheidung temporärer und persistenter Änderungen ein Lookahead $look_{t_i} := W_{t_i,t_i+\omega}$ verwendet. Es gilt $|act_{t_i,x}| \geq \omega = |look_{t_i}|$. Zur Erkennung von Intervalländerungen werden die Vorkommen eines Aktivitätspaares (a', a'') innerhalb t_i einzeln betrachtet und die Ergebnisse einzelner Überprüfungen aggregiert. Jedes Vorkommen, für das eine Änderung erkannt wurde, wird als Clusterindikator für

(a', a'') in t_i bezeichnet. Für jedes Vorkommen (a_i, a_j) des Paares werden notwendige Vorbedingungen (VB), die sich auf t_i, sowie das aktuelle Cluster und Persistenzbedingungen (PB), die sich ausschließlich auf den Lookahead beziehen, unterschieden:

(K1) Die notwendige Bedingung für eine Änderung dieser Kategorie ist, dass die Distanz von (a_i, a_j) in t_i ausserhalb des Distanzintervalls des bisherigen Clusters liegt, i.e.

$$VB : \boldsymbol{d} = d_{t_i}((a_i, a_j)) \notin act_{t_i,x}[a', a'']$$

Zur Abschwächung der Sensitivität dieser Bedingung kann als Persistenzbedingung der Support der Distanz $\boldsymbol{d}$ innerhalb des Lookaheads betrachtet und eine Schranke τ für die minimale Anzahl deren Vorkommen festgesetzt werden:

$$PB : supp_{a',a''}(look_{t_i}, \boldsymbol{d}) \geq \tau$$

Eine Änderung dieser Art tritt in Erscheinung, sobald die obigen Bedingungen für die Distanz mind. eines Vorkommens von (a', a'') in t_i gelten.

(K2) Änderungen dieser Kategorie beziehen sich auf die Kontraktion eines Distanzintervalls. Dafür muss die Distanz von (a_i, a_j) in t_i innerhalb des Distanzintervalls des aktuellen Clusters liegen (notwendige Vorbedingung) und das Intervall innerhalb des Lookaheads verkleinert werden (Persistenzbedingung):

$$VB : d_{t_i}(a', a'') \in act_{t_i,x}[a', a'']$$

$$PB : look_{t_i}[a', a''] < act_{t_i,x}[a', a'']$$

Eine Änderung dieser Art tritt nur in Erscheinung, wenn die obigen Bedingungen für die Distanzen aller Vorkommen von (a', a'') in t_i gelten.

(K3) Kategorie (K3) enthält zwei Distanzintervalländerungen. Im ersten Fall (A), enthält kein Ausführungspfad des aktuellen Clusters eine Distanz von (a', a''), der Pfad t_i jedoch mindestens eine. Hierbei wird keine Persistenzbedingung definiert und jedes Vorkommen (a_i, a_j) führt zu einem Clusterindikator. Der Zweite Fall (B) beschreibt die umgekehrte Situation, in der ein Distanzintervall für (a', a'') existiert, ab t_i (i.e. im kompletten Lookahead) das Paar jedoch nicht mehr auftritt. In diesem Fall existiert nur ein Clusterindikator. Die Bedingung PB_B stellt sicher, dass nur dann eine Änderung erkannt wird, wenn die Abwesenheit von (a', a'') dauerhaft ist, i.e. der Lookahead keine Distanz des Aktivitätspaares enthält.

$$VB_A : D_{act_{t_i,x}}(a', a'') = \emptyset \wedge D_{t_i}(a', a'') \neq \emptyset$$

$$VB_B : D_{act_{t_i,x}}(a', a'') \neq \emptyset \wedge D_{t_i}(a', a'') = \emptyset$$

$$PB_B : D_{look_{t_i}}(a', a'') = \emptyset$$

Zur einfacheren Verwendung wird im Folgenden für die Clusterindikatoren für (a', a'') innerhalb eines Pfades t_i die abkürzende Schreibweise $P^C_{a',a''}(t_i) \subseteq P_{a',a''}(t_i)$ verwendet. Auf Basis von Clusterindikatoren wird das Clusteringergebnis eines Aktivitätspaares gebildet.

Definition 6.9 (Clusteringergebnis eines Aktivitätspaares). *Gegeben sei ein Prozesslog $L = \langle \sigma_1, ..., \sigma_n \rangle$ und mit A die Menge unterschiedlicher Aktivitäten innerhalb von L. Für $a', a'' \in A$ ist das Clusteringergebnis des Aktivitätspaares (a', a'') definiert als $\mathcal{R}_{a',a''} := \{r_1, \dots r_k\}$, $r_i \in [1; n]$, $r_i \neq r_j$ und enthält die Menge der Indexe von Ausführungspfaden in L, auf Basis derer Distanzen eine Distanzänderung erkannt wurde.* ⊣

Steuerung der Sensitivität des Verfahrens

Distanzen zwischen Aktivitätspaaren reagieren sehr sensitiv auf Veränderungen des Prozessverhaltens. Je kleiner ω gewählt wird, desto mehr potentielle Änderungszeitpunkte werden „erkannt“, da sich kleinste Unterschiede in aufeinanderfolgenden Ausführungspfaden bemerkbar machen. Zur Steuerung des APClustering-Verfahrens werden deshalb eine Reihe von Parametern für gezielte Sensitivitätsanpassungen eingeführt. Bei der Erkennung einer Änderung von Distanzintervallen auf Basis von Clusterindikatoren findet zunächst eine Bewertung statt, anhand derer entschieden wird, ob ein Eintrag im Clusteringergebnis eines Aktivitätspaares stattfindet.

Treten Aktivitätspaare sehr häufig in einzelnen Pfaden in Erscheinung, spricht dies für Zyklen im Ursprungsmodell. In Kombination mit parallelen Beziehungen zwischen einzelnen Aktivitäten, kann dies eine Vielzahl unterschiedlicher Distanzen und damit auch Clusterindikatoren hervorrufen. Mit der Einführung einer Schranke $\psi_{num} \in \mathbb{N}^+$ für die Mindestanzahl von Clusterindikatoren kann der Effekt einer unverhältnismäßig starken Berücksichtigung abgeschwächt werden.

Des Weiteren wird das *Gewicht* von Clusterindikatoren betrachtet. Tritt eine Distanzänderung in Erscheinung, die sich auf die Vergrößerung oder Verkleinerung eines Intervalls bezieht, steigt das Gewicht des Indikators mit der Anzahl von Pfaden innerhalb des Lookaheads, die Distanzen des Aktivitätspaares enthalten. Damit wird die Erkennung von Änderungen verhindert, die im weiteren Verlauf nicht ausreichend durch weitere Vorkommen von Distanzen „gestützt“ werden. Ein „Ausbruch“ aus einem Distanzintervall im aktuell betrachteten Pfad mit einem Lookahead, der keine weitere Distanz des Aktivitätspaares enthält, wäre ein typisches Beispiel eines Indikators mit geringem Gewicht. Analog zum Gewicht von Clusterindikatoren wird die *Stabilität* des aktuellen Clusters gemessen, die mit dem Verhältnis von Pfaden mit Distanzen eines Aktivitätspaares zur Gesamtzahl der Pfade innerhalb des Clusters steigt. Beide Maße, Gewicht und Stabilität werden mithilfe der *Signifikanz* von Samples berechnet:

Definition 6.10 (Signifikanz von Samples). *Gegeben sei ein Sample W und ein Aktivitätspaar (a', a''). Die Signifikanz von W bzgl. (a', a'') ist wie folgt definiert:*

$$sig(W, (a', a'')) := \begin{cases} \frac{|W| - |\{\sigma \in W \mid D_\sigma(a', a'') = \emptyset\}|}{|W|} & , D_W(a', a'') \neq \emptyset \\ 1 & , D_W(a', a'') = \emptyset \end{cases}$$ ⊣

Um die Erkennung von Änderungen des Typs (C9) (siehe Abb. 6.5) nicht zu gefährden, wird das Gewicht zugehöriger Indikatoren in diesem Fall auf 1 gesetzt. Entscheidungen,

ob eine Distanzänderung in das Clusteringergebnis übernommen wird, fallen stets auf Basis von Mengen von Clusterindikatoren. Die Gewichte einzelner Clusterindikatoren werden dazu zu einer *Evidenz* aggregiert.

Definition 6.11 (Clusteringevidenz). *Die Clusteringevidenz eines Aktivitätspaares* (a', a'') *innerhalb eines aktuell betrachteten Pfades* t_i *ist definiert als*

$$ev(t_i, (a', a'')) := \frac{|P^C_{a',a''}(t_i)|}{|P_{a',a''}(t_i)|} * sig(look_{t_i}, (a', a'')) \qquad \dashv$$

Clusterindikatoren innerhalb eines Pfades haben stets dasselbe Gewicht, da sie sich auf denselben Lookahead und dasselbe Aktivitätspaar beziehen. Für die Filterung besonders aussagekräftiger Mengen von Clusterindikatoren wird eine Schranke $\psi_{ev} \in [0; 1]$ für deren Mindestevidenz eingeführt.

Die Aussagekraft eines Clusterindikators kann auch durch die Anzahl der Pfade im Lookahead geschmälert werden. Können gegen Ende der Traversierung des Prozesslogs nur noch Lookaheads gebildet werden, die weniger als ω Pfade enthalten, wird deren Signifikanz um einen *Größenfaktor* α verringert, der sich bzgl. eines Lookaheads zu $\alpha = |look_{t_i}|/\omega$ ergibt. Auf diese Weise können Häufungen nicht signifikanter Änderungen innerhalb der letzten Pfade eines Prozesslogs verhindert werden. Ein weiterer Größenfaktor ergibt sich aus dem Verhältnis der Größe des aktuellen Clusters zur Größe des Lookaheads. Bewegt sich die Distanz eines Aktivitätspaares über weite Strecken des Prozesslogs innerhalb eines Intervalls, scheint es naheliegend, erkannte Änderungen nur dann als solche zu akzeptieren, wenn die Größe des Lookaheads nicht vergleichsweise gering ausfällt. Der *relative Größenfaktor* ergibt sich zu:

$$\alpha_{rel} = \frac{|look_{t_i}|}{|act_{t_i,x}|} + \left(1 - \frac{|look_{t_i}|}{|act_{t_i,x}|}\right) * \theta$$

Mithilfe von $\theta \in [0; 1]$ kann der Einfluss des relativen Größenfaktors gesteuert werden (1 für maximalen Einfluss).

Zusammenfassend existieren folgende Parameter für das APClustering-Verfahren:

1. ψ_{num}: Minimale Anzahl von Clusterindikatoren.
2. ψ_{ev}: Minimale Clusteringevidenz.
3. θ: Einflussfaktor von α_{rel}.

Eine Distanzänderung wird nur dann in das Clusteringergebnis eines Aktivitätspaares übernommen, wenn die Mindestanforderungen gemäß der eingeführten Steuerungsparameter erfüllt sind. Bezugnehmend auf ein aktuelles Cluster $act_{t_i,x}$ mit $look_{t_i}$ und einer Fenstergröße ω müssen dazu folgende Bedingungen erfüllt sein:

(1) $P^C_{a',a''}(t_i) \geq \psi_{num}$

(2) $ev(t_i, (a', a'')) \geq \psi_{ev}$

(3) $\alpha * \alpha_{rel} * ev(t_i, (a', a'')) > sig(act_{t_i,x}, (a', a''))$

Algorithmus 6.2 Berechnung des Clusteringergebnisses für ein Aktivitätspaar

Eingabe: Prozesslog $L := \{t_1, \ldots, t_n\}$, Distanzmatrix M^L, Fenstergröße ω, Aktivitätspaar (a', a'')

```
1:  procedure CLUSTERING(L, M^L, ω, (a', a''))
2:      R_{a',a''} ← ∅; i ← 1; c ← 0
3:      while i < |L| do
4:          if |act_{t_i,c}| ≥ ω then
5:              if |P^C_{a',a''}(t_i)| ≥ ψ_num ∧ ev(t_i, (a', a'')) ≥ ψ_ev then
6:                  if α * α_rel * ev(t_i, (a', a'')) > sig(act_{t_i,x}, (a', a'')) then
7:                      R_{a',a''} ← R_{a',a''} ∪ {i}
8:                      c ← 1
9:                  end if
10:             end if
11:         end if
12:         i ← i + 1
13:     end while
14: end procedure
```

6.2.3 Konsolidierung von Clusteringergebnissen

Nach der Berechnung von Clusteringergebnissen für einzelne Aktivitätspaare werden die enthaltenen Schnittmarken für angenommene Clustergrenzen kombiniert, indem für alle Indexe von Ausführungspfaden die Menge von Aktivitätspaaren gezählt wird, deren Clusteringergebnis an dieser Stelle eine Schnittmarke enthält. Das Ergebnis dieser Summation wird mithilfe spezieller Clustergraphen visualisiert, die neben der Anzahl von Aktivitätspaaren auch Informationen zur Art erkannter Änderungen und der bisherigen Clustergröße einzelner Aktivitätspaare bis zur Schnittmarke enthält. Der visualisierte Wert im Graph ergibt sich für einen gegebenen Pfadindex i zu:

$$CG(i) = \sum_{(a',a'') \in A \times A} \sum_{i \in \mathcal{R}_{a',a''}} i$$

Strukturelle Prozessänderungen sind dabei nicht ausschließlich an vergleichsweise hohen Ausschlägen im Graph erkennbar. Diese kommen zustande, wenn eine Änderung sehr viele Aktivitätspaare betrifft und veränderte Distanzen innerhalb eines oder weniger Ausführungspfade erkennbar sind. Es ist jedoch auch möglich, dass eine Änderung sich erst über eine Menge von Pfaden hinweg bemerkbar macht, z.B. wenn die Änderung nur relativ selten durchlaufene Teilpfade eines Prozesses betrifft. Infolgedessen besteht das Ziel der Analyse nicht darin, Änderungszeitpunkte im Sinne einzelner Pfadindexe zu erkennen, sondern *Änderungsphasen*, innerhalb derer eine durchgeführte Änderungsoperation wahrscheinlich ist.

Abb. 6.7 zeigt den Clustergraph für den Prozesslog aus Abb. 6.1. Unter Verwendung einer Fenstergröße von $\omega = 5$ sind im mittleren Bereich deutlich zwei Ausschläge erkennbar. Der erste Ausschlag steht für Distanzänderungen von vier Aktivitätspaaren mit jeweils einem Clusterindikator. Die grauen Bereiche heben die Pfade des bis zu diesem Zeitpunkt

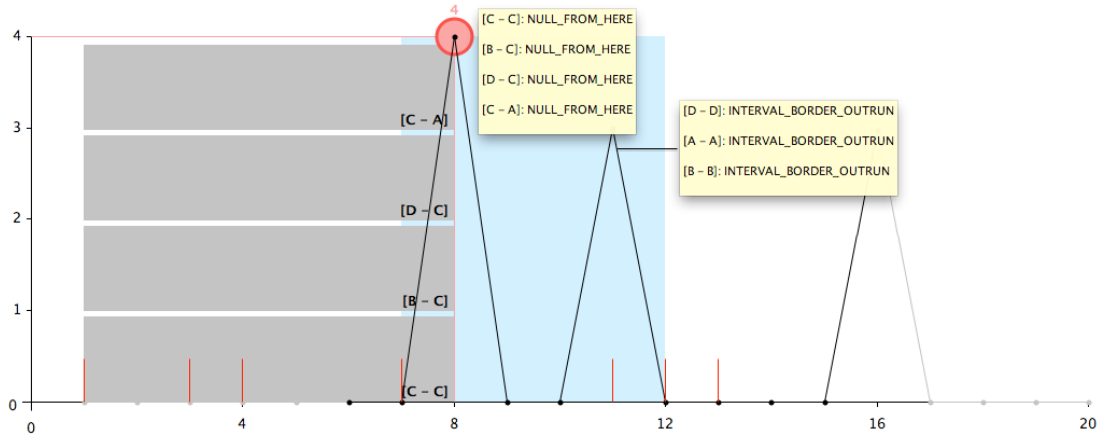


Abbildung 6.7.: Clustergraph für den Prozesslog in Abb. 6.1.

aktuellen Clusters der einzelnen Paare hervor. Für jedes der Paare wurde eine Änderung des Typs (C9) erkannt. Obwohl die tatsächliche Änderung zwischen Pfad 10 und 11 vollzogen wurde, ist diese Änderung schon in Pfad 8 erkennbar, da bereits ab hier kein Pfad mehr existiert, der eine Distanz der Paare enthält. Ein weiteres Beispiel für die frühere Erkennung einer Änderung wäre eine Kontraktion, verursacht durch eine einzige Distanz innerhalb des aktuell untersuchten Pfades in Kombination mit einem Lookahead, der keine Distanzen mehr enthält. Die Kontraktion würde in diesem Fall als persistent betrachtet. Der zweite Ausschlag wird durch Distanzänderungen der Klasse (K1) bedingt, die direkt nach der tatsächlichen Änderung in Pfad 11 erkennbar sind. Die dadurch implizierte Änderungsphase ist blau hinterlegt und umfasst die Pfade 8 bis 11.

Neben Ausschlägen, die durch tatsächliche Veränderungen hervorgerufen werden, können weitere auftreten, die lediglich für die Bevorzugung bestimmter Teilpfade eines Ursprungsmodells stehen. In Abb. 6.7 kommt der Ausschlag bei Pfad 16 durch die Abwesenheit mehrfacher Schleifendurchläufe innerhalb der Pfade 16-20 zustande. Zur Unterscheidung dieser zwei Fälle kann die ebenfalls visualisierte Information darüber, zu welchen Zeitpunkten bisher ungesehene Pfade observiert wurden, herangezogen werden (rote Linien auf Abszisse). Diese heuristische Vorgehensweise ist global orientiert, da sich „ungesehenes Verhalten“ in diesem Fall auf bisheriges Verhalten innerhalb des gesamten Prozesslogs und nicht nur innerhalb aktueller Cluster bezieht. Da der Prozesslog im Beispiel ab Pfad 13 kein neues Verhalten mehr enthält, kann der Ausschlag bei Pfad 16 als *modellinterne* Änderung angesehen werden. Solche Änderungen sind durch Ausführungsphasen eines Prozesses gekennzeichnet, innerhalb derer generierte Pfade auf eine Teilmenge der möglichen Pfade beschränkt sind. Abhängig von der gewählten Fenstergröße variiert die Anzahl von modellinternen Änderungen stark. Je kleiner die Fenstergröße, desto wahrscheinlicher ist das Auftreten solcher Änderungen, da das als „normal“ betrachtete Prozessverhalten dadurch stärker eingeschränkt und die Wahrscheinlichkeit für die Observierung neuen Verhaltens erhöht wird. Interessanterweise scheinen tatsächliche Änderungen unter Verwendung unterschiedlicher Fenstergrößen „stabiler“ zu sein als modellinterne Änderungen.

Bei der Betrachtung von Clustergraphen muss berücksichtigt werden, dass innerhalb der ersten ω Pfade niemals Änderungen erkannt werden, da diese Pfade für die Bestimmung typischen Prozessverhaltens zu Beginn der Änderungserkennung verwendet werden. Ausschläge innerhalb der letzten ω Pfade sind aufgrund reduzierter Lookaheadgröße weniger aussagekräftig als andere. Beide Bereiche sind innerhalb der prototypischen Implementierung des Verfahrens anhand grauer statt schwarzer Punkte im Clustergraph erkennbar.

Erkennung von Änderungsphasen

Die Anzahl und Stärke der Ausschläge und damit auch die Größe der Änderungsphasen sind abhängig von der Komplexität des Ursprungsmodells und der Änderungsoperation. Die Anzahl der Aktivitätspaare, die von Distanzänderungen betroffen sind (Höhe der Ausschläge) und die Anzahl der Pfade, die solche Paare enthalten (Größe der Änderungsphasen), sind dabei die maßgeblichen Einflussfaktoren.

Für die Bestimmung von Änderungsphasen innerhalb eines Prozesslogs $L = \langle \sigma_1, \ldots, \sigma_n \rangle$ wird überprüft, innerhalb welcher Bereiche (i.e. aufeinander folgende Ausführungspfade) verhältnismäßig viele Ausschläge zu verzeichnen sind. Ausgehend von einer Maximalgröße für Änderungsbereiche m_c wird dabei überprüft, für wie viele Aktivitätspaare innerhalb von Samples $W^L_{s,t}$ mit $|W^L_{s,t}| = m_c$ Änderungen erkannt wurden. Genauer wird der Anteil dieser Aktivitätspaare im Vergleich zur Anzahl insgesamt vorkommender unterschiedlicher Aktivitätspaare betrachtet:

$$P_{rel}(W^L_{s,t}) := \frac{|AP_c(W^L_{s,t})|}{|\mathbb{P}^L|}, \; wobei$$

$$AP_c(W^L_{s,t}) := \{(a', a'') \in \mathbb{P}^L \mid \exists i \in \mathcal{R}_{a',a''} : s \leq i \leq t\}$$

Ein Bereich wird als Änderungsbereich aufgefasst, wenn dieser Anteil über einem Grenzwert liegt. Dieser Grenzwert τ_c orientiert sich am maximalen Ausschlag $CG_{max} := max_{i \in \{1,\ldots,n\}} CG(i)$ und wird definiert als

$$\tau_c := \theta_c * \frac{CG_{max}}{|\mathbb{P}^L|}$$

Insofern steht er für den Mindestanteil von Aktivitätspaaren (relativ zum maximalen Ausschlag), für die innerhalb eines Bereichs Änderungen erkannt wurden. Mithilfe von Parameter θ_c kann das Verhältnis von τ zum Anteil an Aktivitätspaaren mit erkannten Änderungen bzgl. des stärksten Ausschlags von CG bestimmt werden. Ein Bereich $W^L_{s,t}$ ist ein Änderungsbereich wenn gilt:

$$P_{rel}(W^L_{s,t}) \geq \tau_c$$

Diese Definition verhindert Mehrfachzählungen von Aktivitätspaaren innerhalb von Änderungsbereichen und stellt eine normalisierte Grenzwertdefinition sicher, die sich an

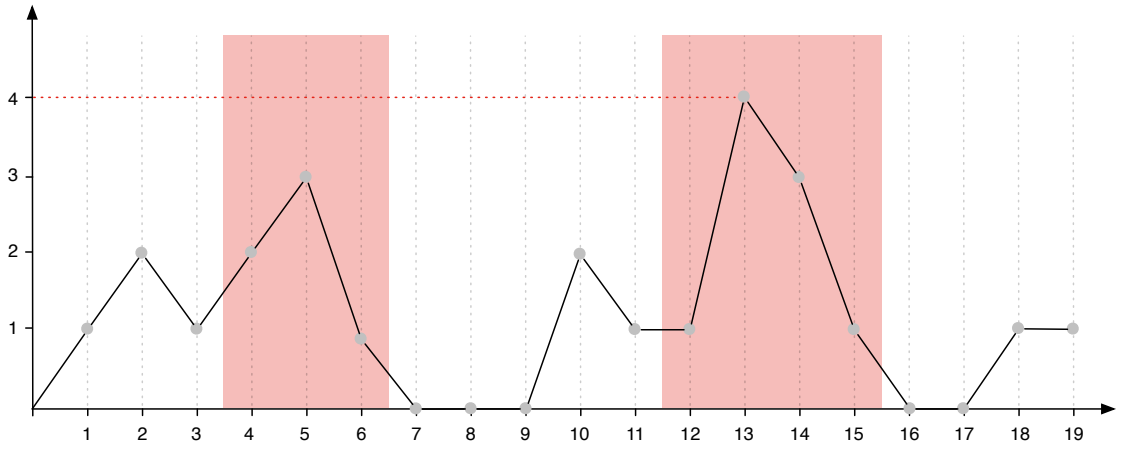


Abbildung 6.8.: Visualisierung von Änderungsbereichen.

observierten Maximalgrößen orientiert. Überlappende Änderungsbereiche werden vereinigt. Abb. 6.8 zeigt erkannte Änderungsbereiche unter Verwendung von $m_c = 2$ und $\tau_c = 1 * (4/10)$ bei 10 unterschiedlichen Aktivitätspaaren. Die Änderungsbereiche $[4;5]$ und $[5;6]$ wurden zu einem Bereich zusammengefasst, ebenso die Bereiche $[12;13]$, $[13;14]$ und $[14;15]$. Die Betrachtung von Clustergraphen unter Verwendung unterschiedlicher Parametrisierungen soll es Betrachtern ermöglichen, Änderungsphasen möglichst präzise zu erkennen. Niedrige Werte für die Mindestgröße von Änderungsbereichen identifizieren dabei ausschließlich Bereiche mit sehr hohen Ausschlägen; höhere Werte schließen auch Bereiche mit höherer Anzahl niedrigerer Ausschläge mit ein.

Die Erkennung von Änderungsbereichen findet stets auf Basis von Clusteringergebnissen statt, die durch die Verwendung einer Fenstergröße ω zustande gekommen sind. Überlegungen zur Erweiterung des Verfahrens schließen die Kombination von Änderungsphasen über verschiedene Fenstergrößen hinweg ein. Bisherige Experimente legen nahe, dass Änderungsphasen sich nicht beliebig verschieben, sondern auch unter Verwendung verschiedener Fenstergrößen hinsichtlich ihrer Position innerhalb des Prozesslogs eine gewisse Konstanz aufweisen.

Das APClustering-Verfahren wurde in erster Linie als intuitive Methode für die Exploration dynamischer Prozesscharakteristika entworfen. Um einer Antwort auf die Frage, inwiefern diese Anforderung erfüllt und Änderungsphasen tatsächlich ohne komplexe Parametrisierung (wie bei existierenden Verfahren) erkannt werden können, näher zu kommen, wurden zahlreiche Experimente durchgeführt, deren Ergebnisse innerhalb des nächsten Abschnitts dargelegt werden.

6.3 Evaluation

Die Evaluation des APClustering-Verfahrens umfasst zunächst eine Betrachtung der Berechnungskomplexität (Abschnitt 6.3.1). Anschließend wird mit der Einführung von Änderungsoperationen Prozessdynamik i.S. beabsichtigter Strukturveränderungen definiert

(Abschnitt 6.3.2) und damit eine Grundlage für die Bewertung des Verfahrens hinsichtlich der Erkennung dieser Veränderungen in Abschnitt 6.3.3 gelegt.

6.3.1 Berechnungskomplexität

Für die Feststellung der Komplexität des APClustering-Verfahrens wird die Erstellung der Distanzmatrix getrennt von der Berechnung von Änderungsphasen betrachtet. Die Distanzmatrix wird durch die Betrachtung aller Ausführungspfade im Prozesslog konstruiert und für alle folgenden Berechnungen von Änderungsphasen verwendet ohne dabei verändert zu werden. Für die Bestimmung von Distanzen zwischen Aktivitätspaaren muss jeder Ausführungspfad im Prozesslog betrachtet und alle darin enthaltenen Vorkommen von Paaren identifiziert werden. Bei einer durchschnittlichen Länge der Ausführungspfade im Prozesslog von r und der Annahme, dass kein Paar mehrfach auftritt (keine Abbrüche in Zeile 6), benötigt Alg. 6.1 dafür $\frac{r*(r-1)}{2}$ Schritte pro Pfad, also insgesamt $n * \frac{r*(r-1)}{2}$ bei n Pfaden im Prozesslog. Der Berechnungsaufwand ist also linear in der Anzahl der Ausführungspfade und quadratisch in der Länge von Pfaden, i.e. $O(n * r^2)$. Da für jedes Aktivitätspaar und jeden Pfad alle Distanzen entsprechender Vorkommen gespeichert werden, kann eine Distanzmatrix viel Speicher benötigen. Ausgehend von der durchschnittlichen Anzahl von Vorkommen eines Aktivitätspaares in Ausführungspfaden b und der Menge unterschiedlicher Prozessaktivitäten A, kann $n*|A|^2*b$ als obere Schranke für die Anzahl gespeicherter Distanzen gelten.

Hinsichtlich der Komplexität der Bestimmung von Änderungsphasen wird zunächst der Aufwand für die Bestimmung von Änderungszeitpunkten betrachtet. Abhängig von der Fenstergröße ω, werden für jedes Aktivitätspaar Zeitpunkte im Prozesslog bestimmt, an denen eine signifikante Veränderung der bisher als üblich betrachteten Distanz beobachtbar ist. Im ungünstigsten Fall wird niemals ein Clusterindikator erkannt und in jedem Schritt der Lookahead zu Rate gezogen um eine Entscheidung über eine Änderung zu treffen. Um die Bestimmung der dafür benötigten Arbeitsschritte zu vereinfachen, wird im Folgenden o.B.d.A. angenommen, dass die Anzahl der Ausführungspfade im Prozesslog einem Vielfachen der Fenstergröße entspricht. In diesem Fall wird nach der Erstellung des initialen Clusters, welches die ersten ω Pfade enthält, bei jeder Betrachtung folgender Pfade ein Lookahead der Größe ω und alle darin befindlichen Distanzen des Aktivitätspaares betrachtet. Am Ende des Prozesslogs verringert sich die Größe des Lookaheads, sodass für die Betrachtung von Lookaheadelementen insgesamt folgender Aufwand entsteht:

$$b * \left((n - 2 * \omega + 1) * \omega + \frac{\omega * (\omega - 1)}{2} \right)$$

$$\Leftrightarrow \frac{b * \omega * (2 * n - 3 * \omega + 1)}{2}$$

Dieser Aufwand ist maximal bei einer Fenstergröße von $\omega = \frac{1}{3} * (n + \frac{1}{2})$ und beträgt dann $\frac{b}{6} * (n^2 + n + \frac{1}{4})$. Hinzu kommt der Aufwand für die Betrachtung jedes Ausführungspfades beim Durchlaufen des Prozesslogs. Unter der worst-case-Annahme, dass für jedes potentiell mögliche Aktivitätspaar Distanzen existieren und in keinem Fall Änderungen erkannt

werden, ergibt sich der Gesamtaufwand für die Bestimmung von Änderungszeitpunkten also zu:

$$\frac{|A|^2 * b}{6} * \left(n^2 + 7 * n + \frac{1}{4}\right)$$

Der Gesamtaufwand ist quadratisch in der Anzahl unterschiedlicher Prozessaktivitäten und quadratisch in der Länge des Prozesslogs. Die durchschnittliche Anzahl von Distanzen pro Aktivitätspaar und Ausführungspfad b hat ebenfalls einen starken Einfluss auf die Gesamtkomplexität. Werden alle drei dieser Parameter bei der Klassifizierung der Komplexität berücksichtigt, liegt der Gesamtaufwand in der Klasse $O(b * |A|^2 * n^2)$.

6.3.2 Ausgangsbasis für durchgeführte Experimente

Untersuchungen hinsichtlich der Erkennung der Prozessdynamik erfordern zunächst eine Definition der betrachteten Prozessveränderungen. Im Rahmen der Evaluation des AP-Clustering-Verfahrens wird dazu auf eine Arbeit von Weber et al.[20] zurückgegriffen, in der sog. Änderungsmuster definiert werden. Jedes Muster steht für eine Änderungsoperation, die im Rahmen von Prozessanpassungen denkbar erscheint. Diese Operationen sind in gewisser Hinsicht „atomar“, da sie sich jeweils auf einen spezifischen Änderungstypus beziehen. Grundsätzlich beziehen sich Änderungsoperationen auf *Prozessfragmente*. Betrachtet werden folgende Muster:

(AP1) Insert: Ergänzung eines Prozessmodells durch ein neues Prozessfragment.

a) **Insert Serial**: Einfügen des Fragments zwischen zwei direkt aufeinander folgende Aktivitäten.

b) **Insert Conditional**: Einfügen des Fragments als Alternative zu einem bestehenden Fragment (XOR-Verzweigung).

c) **Insert Parallel**: Einfügen des Fragments als parallele Ergänzung eines bestehenden Fragments.

(AP2) Delete: Entfernen eines nicht länger benötigten Prozessfragments.

(AP3) Move: Verschieben eines bestehenden Prozessfragments an eine neue Position.

a) **Move Serial**: Verschiebung des Fragments zwischen zwei direkt aufeinander folgende Aktivitäten.

b) **Move Conditional**: Positionierung des Fragments als Alternative zu einem bestehenden Fragment (XOR-Verzweigung).

c) **Move Parallel**: Positionierung des Fragments als parallele Ergänzung eines bestehenden Fragments.

(AP4) Replace: Ersetzen eines veralteten Prozessfragments mit einer neueren (ergänzten) Version.

[20] Vgl. Weber, Rinderle et al. 2013.

(AP5) Swap: Veränderung der Reihenfolge von Prozessaktivitäten (Austausch der Position zweier Fragmente).

(AP6) Extract Sub Process: Ersetzung eines Prozessfragments mit einer Referenzaktivität für dessen Aufruf.

(AP7) Inline Sub Process: Ersetzung einer Referenzaktivität für den Aufruf eines Subprozesses mit dessen Fragment-Beschreibung.

(AP8) Embed Process Fragment in Loop: Mehrfachausführungen eines Prozessfragmentes zulassen, i.e. Umschließen eines existierenden Fragments mit einer Schleife.

(AP9) Parallelize: Parallelisierung bisher sequentieller Fragmente.

Je mehr Aktivitäten die von Änderungsoperationen betreffenden Prozessfragmente enthalten, desto wahrscheinlicher werden Distanzen von Aktivitätspaaren beeinträchtigt. Im Rahmen der Evaluation wurde eine minimalinvasive Strategie verfolgt und stets so wenig Aktivitäten wie möglich in die Änderungen mit einbezogen. Im Folgenden wird jede Änderungsoperation separat betrachtet und die Methode daraufhin überprüft, ob Änderungszeitpunkte/-phasen sichtbar gemacht werden können. Im Gegensatz zu realen Prozessänderungen, die in der Regel mehrere Änderungsoperationen und Prozessaktivitäten umfassen, wird dabei jede Änderungsoperation isoliert betrachtet. Insofern liegt den durchgeführten Experimenten gewissermaßen eine worst-case-Betrachtung zugrunde.

Für die Erstellung von Prozesslogs wurden Prozessmodelle unterschiedlicher Komplexität verwendet und für jede Kombination aus Prozessmodell und Änderungsoperation zunächst Varianten des Ursprungsmodells gebildet. Aufbauend auf diesen Varianten wurden für jede Ursprungsmodell-Varianten-Paarung zwei Prozesslogs mit unterschiedlicher Anzahl von Ausführungspfaden erstellt. Im ersten Fall enthält der Prozesslog zunächst 100 Pfade des Ursprungsmodells und danach 100 Pfade der Variante; im zweiten Fall wurden jeweils 1000 Pfade verwendet. Ursprungsmodelle wurden ausschließlich zur Generierung von Prozesslogs unterschiedlicher Komplexität verwendet[21].

Tab. 6.1 enthält die Merkmale verwendeter Modelle. Bzgl. deren Komplexität ist zunächst die Anzahl der enthaltenen Aktivitäten relevant (bei Existenz von Duplikaten wird die Anzahl unterschiedlicher Aktivitäten in Klammern angegeben). Ein weiteres Komplexitätsmerkmal besteht in der Anzahl unterschiedlicher Ausführungspfade, die von einem Modell generiert werden können. Bei schleifenlosen Modellen ist diese Zahl konstant, andernfalls wird die Steigung der bereits in Abschnitt 4.6 verwendeten Regressionsgeraden angegeben. In beiden Fällen entspricht die Reihenfolge der Modelle deren Komplexität, sodass für Experimente jeweils ein Modell mit *geringer*, *mittlerer* und *hoher* Komplexität zur Verfügung steht.

[21] Prozesslogs zur Evaluierung des APClustering-Verfahrens, einschließlich der zugrundeliegenden Prozessmodelle, können unter http://prorepo.process-security.de/g/681f5fe3 eingesehen werden.

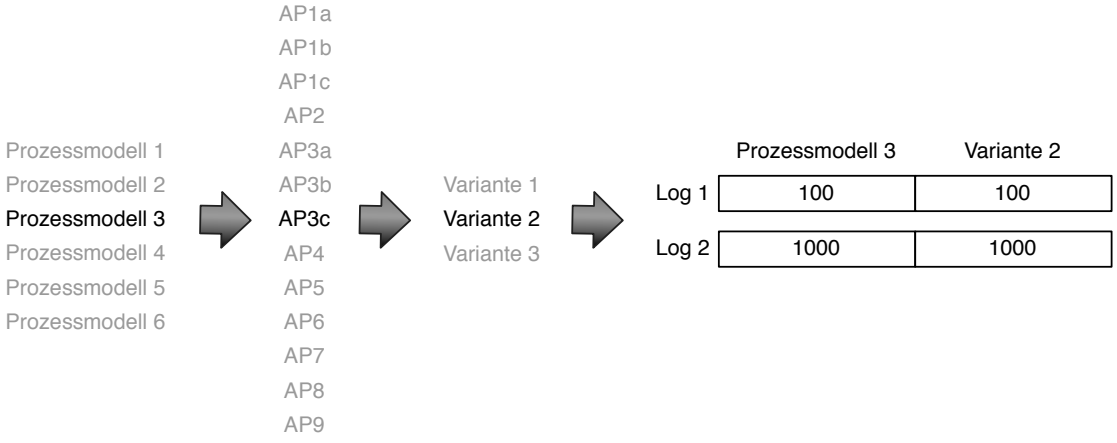


Abbildung 6.9.: Erstellung von Modellvarianten und Prozesslogs.

Netz	Komplexität	Aktivitäten	Pfade	Parallelität	Schleifen	Unsichtbare Aktivitäten
a12*	gering	12	7	✓		✓
herbstFig6p41*	mittel	8(7)	30	✓		✓
herbstFig6p42	hoch	15(8)	17	✓		✓
herbstFig6p18*	gering	5	∞ (0,05)		✓ (arbiträr)	✓
herbstFig6p34*	mittel	11(7)	∞ (0,19)	✓	✓ (strukturiert)	✓
al2*	hoch	11	∞ (0,1)	✓	✓ (arbiträr)	✓

Tabelle 6.1.: Charakteristika verwendeter Petrinetze.

Durch drei unterschiedliche Anwendungen jeder der vorgestellten Änderungsoperationen auf einem gegebenen Ursprungsmodell sollen Beobachtungen auf ein breiteres Fundament und der Einfluss von „Zufallseffekten“ minimiert werden. Insgesamt wurden für jedes Modell $3 * 13$ Varianten erstellt. Die Anzahl der Prozesslogs ergibt sich somit zu $2(\text{Loggröße}) * 6(\text{Ursprungsmodelle}) * (3 * 13)(\text{Varianten}) = 468$. Die Vorstellung jedes einzelnen Experiments würde den Rahmen dieser Arbeit sprengen, deshalb werden im Folgenden Auszüge der Beispiele exemplarisch betrachtet und die Gesamteindrücke im Rahmen eines Fazits subsumiert.

6.3.3 Erkennungsleistung des APClustering-Verfahrens

Zur Beurteilung der Erkennungsleistung wurde das APClustering-Verfahren unter Verwendung der folgenden Parametrisierung auf jeden generierten Prozesslog angewandt:

- $\psi_{num} = 1$
  Für jedes Aktivitätspaar genügt ein Vorkommen mit Distanzänderungen innerhalb eines Pfades für die Erkennung einer Änderung.

- $\psi_{ev} = 0,1$
  Der Mindestwert für die Clusteringevidenz eines Aktivitätspaares wird gering gewählt, um möglichst viele erkannte Änderungen zu berücksichtigen.

- $\theta = 1$
  Um möglichst „stabile“ Änderungen zu erkennen und die Anzahl erkannter Änderungsphasen gegen Ende des Prozesslogs (mit geringer Lookaheadgröße) zu vermeiden, fließt der relative Größenfaktor maximal ein.

- $\theta_c = 0,6$
  Die Existenz hoher Ausschläge im Clustergraph reduziert bei zu hohen Werten für θ_c die Anzahl von Änderungsphasen drastisch. Deshalb wird hier eine moderate Einstellung gewählt.

Für jede Fenstergröße ω wird die *Präzision* des Ergebnisses (erkannte Änderungsphasen) berechnet. Diese ergibt sich aus der Distanz des tatsächlichen Änderungszeitpunktes zur Mitte der nächstgelegenen Änderungsphase, wobei Ergebnisse mit geringerer Anzahl von Änderungsphasen als präziser angesehen werden.

Definition 6.12. *Gegeben sei die Menge erkannter Änderungsphasen* $CP = \{I_1, \ldots, I_n\}$*, wobei jedes* I_i *ein Intervall ist. Dann ergibt sich für* $k \in \{100, 1000\}$ *die Präzision der Änderungserkennung zu:*

$$\frac{\min_{I \in CP}(dist_k(I))}{2 * k} * \frac{1}{|CP|}, \text{ wobei}$$

$$dist_k([l;u]) := \left| k - \left(l + \left\lfloor \frac{u-l}{2} \right\rfloor \right) \right| \qquad \dashv$$

Tab. 6.4 enthält die berechneten Präzisionswerte für den Änderungstyp (AP1a) unter Verwendung einer maximalen Größe von Änderungsbereichen $m_c = 4$. Für jede Modellvariante und jedes $k \in \{100, 1000\}$ wird mit *rel* die relative Größe der Distanzmatrix und mit *par* deren Parallelitätsgrad bis zum Änderungszeitpunkt angegeben. Diese Maße sind hilfreich zur Einschätzung der Komplexität von Prozesslogs vor dem Eintritt einer Änderung. Präzisionen werden zwar für alle Fenstergrößen $[1;100]$ bzw. $[1;1000]$ berechnet, jedoch nicht einzeln abgetragen. Stattdessen werden die berechneten Werte in fünf gleich große Gruppen geteilt (jeweils 20 bzw. 200). Für $k = 100$ ergeben sich die Gruppen $\{\{1, \ldots, 20\}, \{21, \ldots, 40\}, \{41, \ldots, 60\}, \{61, \ldots, 80\}, \{81, \ldots, 100\}\}$ (jeweils Fenstergrößen). Die Präzision innerhalb dieser Gruppen wird als Zufallsvariable aufgefasst, deren Erwartungswert und Varianz in Tab. 6.4 angegeben wird (Varianz hochgestellt). Zur Verbesserung der Lesbarkeit von Präzisionsergebnissen wurden berechnete Werte folgendermaßen eingefärbt:

(rot)	Werte im Intervall $[0;0,2)$
(blau)	Werte im Intervall $[0,2;0,5)$
(grün)	Werte im Intervall $[0,5;0,8)$
(gelb)	Werte im Intervall $[0,8;1,0]$

Sofern sich der Erwartungswert in aufeinander folgenden Gruppen stets verbessert, enthält Spalte ↗ einen Haken. Die Akkuratesse der Änderungserkennung ϑ wird auf Basis der Gruppen 3-5 gebildet und setzt sich aus dem Verhältnis der erreichten Präzision zur maximal möglichen Präzision zusammen. Die Gruppen 1 und 2 werden vernachlässigt, da bei verhältnismäßig kleinen Fenstergrößen die Anzahl erkannter Distanzänderungen typischerweise so groß ausfällt, dass sinnvolle Aussagen über Änderungsphasen nicht möglich sind. Die geringen Werte der entsprechenden Spalten verdeutlichen diese Tatsache. Aus Gründen der Übersichtlichkeit sind die entsprechenden Tabellen für die übrigen Änderungsoperationen dem Anhang (Abschnitt E) zu entnehmen. Änderungsoperation (AP1a) wird weiter zur exemplarischen Veranschaulichung der Bewertung des APClustering-Verfahrens verwendet.

Bei ca. 70% der Prozesslogs (auch in Tab. 6.4) enthält Gruppe 5 den maximalen Präzisionswert und mit Ausnahme von (AP1c) sind die Präzisionswerte für größere Fenstergrößen monoton steigend (meist in ca. 70% der Fälle). Daher liegt die Vermutung nahe, dass eine Vergrößerung der Fenstergröße zu höherer Präzision führt. Dies trifft auf eine Vielzahl der Änderungsoperationen zu, einzige Ausnahmen bilden (AP1b), (AP3b), (AP1c), (AP3c) und (AP9). Während die ersten beiden Ausnahmen das Einfügen einer Aktivität als Alternative (XOR-Verzweigung) zu einem bestehenden Fragment beinhalten, beziehen sich (AP1c) und (AP3c) auf die Positionierung einer Aktivität als parallele Ergänzung eines bestehenden Fragments. Gerade bei Prozesslogs, deren Ursprungsmodelle Schleifen enthalten, ist keine Verbesserung bei größeren Fenstergrößen beobachtbar. Eine naheliegende Erklärung für diesen Sachverhalt ist, dass bei diesen Änderungstypen bei minimalinvasiver Anwendung gerade in ohnehin verhältnismäßig komplexen Prozesslogs (hoher Parallelitätsgrad des Ursprungsmodells) Aktivitätsdistanzen nur unwesentlich verändert und damit schlechter erkannt werden. Dieser Effekt ist auch bei Prozesslogs mit schleifenlosen Ursprungsmodellen beobachtbar. Speziell Variante 2 des Modells a12* und Variante 1 von herbstFig6p42 weisen enorm schlechte Präzision auf. In beiden Fällen lässt sich dies auf die Änderungen zurückführen, die im Ursprungsmodell vorgenommen wurden, die zwar zum Auftreten neuer Distanzen führen, jedoch keine Distanzen bisher observierter Aktivitätspaare verändern. Generell hängt die Exaktheit der Änderungserkennung maßgeblich von der Anzahl an Aktivitätspaaren ab, deren Distanz von der konkreten Änderungsoperation beeinflusst wird.

Um die erreichte Akkuratesse für die Erkennung einzelner Änderungsoperationen kompakter darstellen und interpretieren zu können, wurden Komprimate erzeugt, die für $m_c \in \{4, 6, 10\}$ jeweils Durchschnittswerte zur Subsumierung der Ergebnisse einzelner Varianten enthalten (siehe Tab. 6.2). Mit $rel_{\varnothing}$ und $par_{\varnothing}$ werden durchschnittliche, relative Größe und Parallelitätsgrad zugehöriger Distanzmatrizen angegeben. Auf diese Weise lässt sich der Effekt größerer Änderungsbereiche besser ausmachen und auch der Vergleich zwischen unterschiedlichen Prozessloggrößen wird vereinfacht. Auch in diesen Komprimaten wird deutlich, dass Änderungen in Modellen geringerer Komplexität besser erkannt werden. In einzelnen Fällen wird die Präzision durch Betrachtung größerer Änderungsbereiche verbessert; allgemein fällt der Einfluss unterschiedlicher Größen für Änderungsbereiche hingegen minimal aus. Die Werte $\vartheta_{\varnothing_{100}}$ und $\vartheta_{\varnothing_{1000}}$ stehen für die durchschnittlich erreichte Präzision. Erwartungsgemäß ist auch hier (bei (AP1a)) die höhere Präzision bei der Betrachtung des größeren Logs deutlich sichtbar. Allgemein ist der positive Einfluss größerer Prozesslogs abhängig von der Komplexität des Ursprungsmo-

Änderung	Ohne Schleifen		Mit Schleifen	
	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$
(AP1a)	0,47	0,65	0,36	0,47
(AP1b)	0,38	0,63	0,21	0,27
(AP1c)	0,56	0,76	0,32	0,28
(AP2)	0,74	0,77	0,57	0,60
(AP3a)	0,53	0,75	0,32	0,54
(AP3b)	0,45	0,63	0,36	0,46
(AP3c)	0,46	0,55	0,27	0,33
(AP4)	0,64	0,80	0,43	0,65
(AP5)	0,51	0,61	0,36	0,37
(AP6)	0,63	0,72	0,64	0,77
(AP7)	0,43	0,75	0,44	0,64
(AP8)	0,34	0,66	0,29	0,42
(AP9)	0,50	0,64	0,31	0,29

Tabelle 6.3.: Übersicht der Erkennungsleistung von APClustering.

dells (Anzahl potentiell möglicher Pfade). Je früher (i.S. der Anzahl betrachteter Pfade im Prozesslog) das komplette Verhalten des Ursprungsmodells, oder zumindest ein Großteil dessen, observiert wurde, desto präziser können Abweichungen bestimmt werden. Ist dagegen zu einem Änderungszeitpunkt nur ein Bruchteil des möglichen Prozessverhaltens bekannt, wird die Erkennung zwangsläufig ungenauer, da Distanzintervalle durch häufig wechselnde Pfad-Charakteristiken weniger konstant sind.

Netz	$rel_\varnothing$	$par_\varnothing$	$m_c = 4$		$m_c = 6$		$m_c = 10$		Aggregiert	
			$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$
a12*	0,28	0,03	0,58	0,83+	0,58	0,83+	0,58	0,83+		
herbstFig6p41*	0,43	0,16	0,59	0,66+	0,62	0,83+	0,63	0,87+	0,47	0,65
herbstFig6p42	0,63	0,42	0,22	0,30+	0,21	0,29+	0,27	0,37+		
herbstFig6p18*	0,58	0,47	0,26	0,37+	0,26	0,37+	0,26	0,35+		
herbstFig6p34*	0,69	0,61	0,49	0,56+	0,49	0,56+	0,52	0,56+	0,36	0,47
al2*	0,83	0,67	0,27	0,50+	0,31	0,49+	0,38	0,48+		

Tabelle 6.2.: Exaktheit der Erkennung von Änderungen des Typs (AP1a).

An dieser Stelle wird der Vorteil einer grafischen Betrachtung anhand von Clustergraphen deutlich. Die Zusatzinformation, an welchen Stellen neue Pfade observiert wurden, ist hilfreich zur Bestimmung tatsächlicher Änderungen und dem Filtern sog. interner Änderungen, die nicht durch neues Verhalten hervorgerufen werden. Die Möglichkeit, einzelne Parameter ad-hoc zu verändern und die Effekte der Änderungen grafisch zu erfassen, ist hilfreich zur Elimination wenig aussagekräftiger Ausschläge (speziell im hinteren Bereich) des Clustergraphen.

Tab. 6.3 gibt einen Überblick über die Erkennungsleistung des APClustering-Verfahrens. Für jeden betrachteten Änderungstyp wird die durchschnittlich erreichte Präzision für unterschiedliche Größen von Prozesslogs angegeben. Die Präzisionswerte wurden aus den Ergebnissen für einzelne Modelle aggregiert um einen Vergleich der Leistung hinsichtlich Modellen mit und ohne Schleifen zu ermöglichen. Bei schleifenlosen Modellen fällt die aggregierte Präzision durchweg höher aus und auch die zuvor erläuterte Verbesserung bei der Betrachtung größerer Prozesslogs ist erkennbar.

Eine klare Schwäche distanzbasierter Verfahren ist, dass Änderungen, die keine oder nur minimale Veränderung von Distanzintervallen zur Folge haben, nicht bzw. nur schlecht erkannt werden können. Dies tritt bspw. auf, wenn eine Menge sich gegenseitig ausschließender Aktivitäten durch eine einzelne weitere Aktivität vergrößert wird und Pfade, die eine dieser Aktivitäten enthalten, entweder selten auftreten oder sehr kurz sind. Distanzen zwischen zuvor existenten Aktivitäten ändern sich so nicht, es werden andererseits aber auch nur wenige zusätzliche Distanzen observiert.

Ein inhärenter Nachteil fensterbasierter Verfahren ist, dass zu große Fenstergrößen dazu führen, dass Änderungszeitpunkte überragt und somit nicht mehr erkannt werden können. Andererseits wird dieser Nachteil durch die interaktive Betrachtung von Distanzänderungen aufgewogen. Das Ziel besteht nicht im automatischen Finden der besten Fenstergröße, sondern der möglichst flexiblen „Navigation“ durch Distanzwerte. In vielen Fällen erlaubt das sukzessive Vergrößern initial kleiner Fenstergrößen ein intuitives Erfassen der Prozessdynamik. Das APClustering-Verfahren ist der erste Ansatz mit dieser explizit interaktiven Ausrichtung.

Bei steigender Komplexität von Ursprungsmodellen verschlechtert sich die Erkennungsleistung teils signifikant und auch die Interaktivität leidet durch längere Berechnungszeiten von Änderungsphasen. In solchen Fällen liegt es nahe, statt eines distanzbasierten Verfahrens auf ein statistisches Verfahren zurück zu greifen (z.B. das Verfahren von Bose et al.[22]). Zwar geht dadurch die Interaktivität verloren, jedoch können so auch für komplexe Eingaben bei adäquater Parametrisierung (nicht trivial!) sinnvollere Ergebnisse produziert werden.

Aufgrund der worst-case-Ausrichtung der Experimente kann das Ergebnis in Tab. 6.3 insgesamt als positiv betrachtet werden. Stichprobenbasierte Tests einzelner Prozesslogs unterstreichen die prinzipielle Eignung distanzbasierter Verfahren zur Änderungserkennung. Die Möglichkeit, Distanzveränderungen grafisch zu erfassen, relativiert in vielen Fällen vergleichsweise schlechte Präzisionsergebnisse. Ähnlich systematische Herangehensweisen zur Beurteilung der Akkuratesse von Änderungserkennungen wurden innerhalb bisheriger Arbeiten nicht verwendet.

[22] Vgl. Bose, Aalst et al. 2011.

Netz		Pfade	*rel*	*par*	Präzision ($m_c = 4$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,28	0,03	$0,15^{0,008}$	$0,46^{0,019}$	$0,94^{0,011}$	0,97	0,97	✓	0,96
		2*1000	0,28	0,03	$0,76^{0,145}$	1,00	1,00	1,00	1,00	✓	1,00
	Variante 2	2*100	0,28	0,03	$0,13^{0,002}$	$0,48^{0,005}$	$0,49^{0,002}$	$0,44^{0,007}$	$0,45^{0,006}$		0,46
		2*1000	0,28	0,03	$0,36^{0,068}$	$0,40^{0,004}$	$0,38^{0,002}$	$0,75^{0,048}$	$0,35^{0,205}$		0,50
	Variante 3	2*100	0,28	0,03	$0,22^{0,048}$	$0,23^{0,004}$	0,32	0,32	0,32	✓	0,32
		2*1000	0,28	0,03	$0,73^{0,125}$	$0,92^{0,033}$	1,00	$0,99^{0,005}$	1,00		0,99
herbstFig6p41*	Variante 1	2*100	0,43	0,16	$0,08^{0,004}$	$0,44^{0,113}$	0,87	0,87	0,87	✓	0,87
		2*1000	0,43	0,16	$0,66^{0,196}$	1,00	1,00	1,00	1,00	✓	1,00
	Variante 2	2*100	0,43	0,16	$0,11^{0,005}$	$0,20^{0,004}$	$0,32^{0,002}$	$0,42^{0,006}$	$0,55^{0,029}$	✓	0,43
		2*1000	0,43	0,16	$0,22^{0,026}$	0,50	0,50	0,50	0,50	✓	0,50
	Variante 3	2*100	0,43	0,16	$0,07^{0,001}$	$0,20^{0,007}$	$0,42^{0,007}$	0,48	0,48		0,46
		2*1000	0,43	0,16	$0,51^{0,117}$	$0,49^{0,001}$	$0,50^{0,001}$	$0,49^{0,002}$	0,50		0,49
herbstFig6p42	Variante 1	2*100	0,63	0,42	$0,07^{0,001}$	$0,15^{0,001}$	$0,26^{0,009}$	$0,26^{0,017}$	$0,31^{0,001}$	✓	0,28
		2*1000	0,63	0,42	$0,11^{0,003}$	0,03	0,03	0,03	$0,36^{0,226}$		0,14
	Variante 2	2*100	0,63	0,42	$0,09^{0,001}$	$0,16^{0,002}$	$0,21^{0,003}$	0,17	0,17		0,18
		2*1000	0,63	0,42	$0,19^{0,013}$	$0,26^{0,001}$	$0,32^{0,001}$	$0,39^{0,006}$	$0,49^{0,002}$	✓	0,40
	Variante 3	2*100	0,63	0,42	$0,08^{0,001}$	$0,22^{0,006}$	$0,23^{0,004}$	$0,16^{0,005}$	$0,23^{0,007}$		0,21
		2*1000	0,63	0,42	$0,15^{0,013}$	0,25	0,25	$0,30^{0,010}$	0,50	✓	0,35
herbstFig6p18*	Variante 1	2*100	0,58	0,47	$0,11^{0,002}$	$0,17^{0,001}$	$0,23^{0,003}$	$0,40^{0,011}$	$0,79^{0,054}$	✓	0,47
		2*1000	0,58	0,47	$0,10^{0,002}$	$0,29^{0,014}$	$0,54^{0,005}$	0,50	0,50		0,51
	Variante 2	2*100	0,56	0,44	$0,18^{0,006}$	$0,23^{0,018}$	$0,17^{0,004}$	0,11	0,10		0,13
		2*1000	0,58	0,47	$0,05^{0,003}$	$0,10^{0,009}$	$0,24^{0,004}$	$0,39^{0,056}$	$0,36^{0,003}$		0,33
	Variante 3	2*100	0,58	0,47	$0,14^{0,002}$	$0,14^{0,004}$	$0,18^{0,007}$	$0,26^{0,014}$	$0,13^{0,001}$		0,19
		2*1000	0,58	0,47	$0,15^{0,007}$	$0,26^{0,001}$	0,28	$0,39^{0,004}$	$0,17^{0,031}$		0,28
herbstFig6p34*	Variante 1	2*100	0,69	0,61	$0,15^{0,008}$	$0,22^{0,005}$	$0,24^{0,003}$	$0,33^{0,006}$	$0,88^{0,038}$	✓	0,48
		2*1000	0,69	0,61	$0,10^{0,004}$	$0,17^{0,010}$	$0,23^{0,010}$	0,33	$0,34^{0,001}$	✓	0,30
	Variante 2	2*100	0,69	0,61	$0,14^{0,004}$	$0,24^{0,002}$	$0,32^{0,004}$	$0,34^{0,002}$	$0,43^{0,038}$	✓	0,36
		2*1000	0,69	0,61	$0,11^{0,006}$	$0,21^{0,006}$	$0,28^{0,004}$	$0,42^{0,007}$	$0,52^{0,031}$	✓	0,41
	Variante 3	2*100	0,69	0,61	$0,13^{0,004}$	$0,20^{0,016}$	$0,60^{0,072}$	0,50	$0,74^{0,062}$		0,61
		2*1000	0,69	0,61	$0,15^{0,012}$	$0,32^{0,025}$	$0,89^{0,041}$	1,00	1,00	✓	0,96
al2*	Variante 1	2*100	0,83	0,67	$0,20^{0,015}$	$0,44^{0,010}$	$0,29^{0,006}$	$0,33^{0,005}$	$0,61^{0,050}$		0,41
		2*1000	0,83	0,67	$0,13^{0,019}$	0,06	0,06	0,06	0,06		0,06
	Variante 2	2*100	0,83	0,67	$0,13^{0,004}$	0,14	$0,18^{0,002}$	$0,18^{0,001}$	$0,23^{0,015}$		0,20
		2*1000	0,83	0,67	$0,18^{0,016}$	$0,25^{0,017}$	$0,41^{0,031}$	$0,46^{0,004}$	$0,47^{0,004}$	✓	0,45
	Variante 3	2*100	0,83	0,67	$0,12^{0,003}$	$0,26^{0,006}$	$0,29^{0,004}$	$0,17^{0,001}$	0,16		0,20
		2*1000	0,83	0,67	$0,20^{0,017}$	$0,91^{0,037}$	$0,98^{0,010}$	1,00	1,00	✓	0,99

Tabelle 6.4.: Erreichte Präzision von APClustering für Änderungstyp (AP1a) bei $m_c = 4$.

Kapitel 7

Zusammenfassung und Ausblick

Diese Arbeit beschäftigt sich mit der Frage, inwiefern die Sicherheit von Geschäftsprozessen mithilfe nachgelagerter Prozessanalysen (Prozessaudits) auf Basis von Aufzeichnungen vergangener Ausführungen (Prozesslogs) nachgewiesen werden kann. Nach der Definition von Sicherheit i.S. der Einhaltung konkreter Anforderungen auf funktionaler, organisationeller und informationeller Ebene, sowie der Ableitung von Anforderungen an die nachgelagerte Prozessanalyse allgemein, bewertet sie zunächst das Potential der Process Mining-Technologie. Deren Fokus auf die Extraktion von Information aus Prozesslogs, die Bereitstellung vielfältiger Ansätze zur Visualisierung tatsächlichen Prozessverhaltens, sowie der Überprüfung der Konformität eines Prozesslogs zu einem gegebenen Strukturmodell, macht sie zu einer naheliegenden Option für eine Verwendung innerhalb von Prozessaudits.

Die Arbeit wird dem identifizierten Handlungsbedarf im Hinblick auf die präzise Visualisierung tatsächlichen Prozessverhaltens und Erkennung von Prozessveränderungen im zeitlichen Verlauf durch die Entwicklung adäquater Methoden der Prozessrekonstruktion gerecht. Das GENET^{+}-Verfahren zur Rekonstruktion präziser Strukturmodelle (siehe Abschnitt 4.5) wird durch das GENET*-Verfahren um die zusätzliche Betrachtung von Datenflüssen ergänzt (siehe Abschnitt 5.2). Hinsichtlich der Prozessdynamik erlaubt das APClustering-Verfahren eine intuitive Erfassung struktureller Prozessveränderungen.

Insgesamt trägt die Arbeit dazu bei, das Potential automatisierter Prozessanalysen im Rahmen von Prozessaudits abzuschätzen und diese durch speziell auf sicherheitsspezifische Anforderungen ausgerichtete Methoden zu unterstützen. Dabei steht die Erfassung tatsächlichen Prozessverhaltens und dessen Vergleich mit geplantem Verhalten im Vordergrund. Die einzelnen Beiträge der Arbeit lassen sich wie folgt zusammenfassen:

- Die systematische Herleitung von Sicherheitsanforderungen für Geschäftsprozesse und Ableitung funktionaler Anforderungen an Mechanismen zur nachgelagerten Sicherheitsanalyse, sowie deren Interpretation auf der technischen Ebene von Process Mining bildet eine Grundlage für die Entwicklung von Methoden, die der Zielsetzung des Nachweises oder der Verbesserung der Sicherheit von Geschäftsprozessen folgen.

- Im Rahmen dieser Arbeit wird gezeigt, dass die Process Mining-Technologie effektiv zur Evaluation des Ist-Zustands von Prozessen eingesetzt werden kann. Die Stärke der Technologie liegt derzeit bei Verfahren zur Erstellung statistischer Informationen zum Prozessverlauf und zum Nachweis konkreter Sicherheitseigenschaften auf Prozesslogs. Im Rahmen von Prozessaudits kann damit eine deutlich höhere Aussagekraft getroffener Feststellungen zum Sicherheitsstatus von Prozessen erreicht werden als mit stichprobenbasierten Verfahren. Methoden der Prozessrekonstruktion liefern zwar einen Beitrag zur Visualisierung tatsächlichen Prozessverhaltens und damit zu dessen intuitivem Verständnis, allerdings entsteht dabei eine Unschärfe, die bei der Analyse zu Fehlinterpretationen führen kann.

- Der entwickelte Ansatz GENET$^+$ kann zur Rekonstruktion von Strukturmodellen verwendet werden, die den tatsächlichen Prozessverlauf exakt abbilden und sich damit zur Erlangung eines detaillierten Verständnisses der Prozessausführung eignen. Im Gegensatz zu existierenden Methoden ist bei deren Betrachtung klar, dass für jeden vollständigen Pfad im Modell eine Entsprechung im Prozesslog existiert und das Verhalten des Modells weder Verhalten gemäß Prozesslog vernachlässigt, noch zusätzliches Verhalten hinzufügt. Die Problematik möglicher Fehlinterpretationen kann somit umgangen werden.

  Durch Zustandsspaltung wird der Stand der Technik regionsbasierter Prozessrekonstruktion um eine alternative Strategie zur Sicherstellung der EC-Eigenschaft erweitert, die es Anwendern erlaubt, mehrfach auftretende Prozessaktivitäten in Ergebnismodellen zu vermeiden.

- Mit GENET* wird erstmals ein Verfahren zur kombinierten Kontroll-/Datenfluss-Rekonstruktion definiert. Dadurch wird das Analysespektrum um die gleichzeitige Betrachtung unterschiedlicher Prozessaspekte erweitert und eine Grundlage für eine ganzheitliche Betrachtung der Prozessausführung geschaffen. Die Möglichkeit, Datenflüsse zu berücksichtigen, ergibt sich dabei aus der Flexibilität regionsbasierter Rekonstruktionsverfahren, die in ihrem zweistufigen Aufbau begründet liegt (Zwischenschritt über Transitionssysteme).

- Das APClustering-Verfahren zur Erkennung der Prozessdynamik erlaubt die intuitive Erfassung unterschiedlicher Ausführungsphasen von Prozessen im zeitlichen Verlauf. Der Nutzen im Rahmen von Sicherheitsanalysen ist vielfältig. Im Fall aufgetretener Notfallsituationen oder sonstiger unvorhergesehener Ausnahmesituationen kann die Betrachtung der Prozessdynamik dazu beitragen, die Robustheit der Prozessausführung hinsichtlich solcher Ereignisse zu beurteilen. Veränderungen der Ausführung in besagten Zeiträumen zeigen den Einfluss von Ereignissen sowie die „Reaktion“ des Prozesses auf diese. Werden zu bestimmten Zeitpunkten Kontrollmechanismen für die Sicherstellung gewünschter Eigenschaften des Prozessverlaufs implementiert, ist eine Veränderung der Prozessausführung ab diesem Zeitpunkt zu erwarten. Die Wirkung der Kontrolle auf den Prozess kann also durch die Betrachtung der Dynamik nachvollzogen werden.

  Die Betrachtung der Prozessdynamik stellt darüber hinaus eine Voraussetzung für die Rekonstruktion präziser Modelle dar. Deren Vernachlässigung führt dazu, dass

Verhalten unterschiedlicher, der Ausführung zugrunde liegender Modelle subsumiert wird, was eine präzise Rekonstruktion unmöglich macht (vgl. Abschnitt 6).

In ihrer Gesamtheit erlauben die entwickelten Ansätze eine präzise Erfassung tatsächlichen Prozessverhaltens. Vergleiche rekonstruierter Modelle mit Strukturmodellen geplanten Prozessverhaltens erlauben die Identifikation „kritischer“ Stellen innerhalb von Prozessen, an denen die reale Ausführung von geplantem Verhalten abweicht. Das Zustandekommen von Ausführungspfaden, die einer genaueren Untersuchung unterzogen werden, kann in hochpräzisen Modellen nachvollzogen werden, was die Findung von Ursachen für abweichende Pfade unterstützt.

Die Effektivität der entwickelten Methoden konnte im Rahmen der Arbeit erfolgreich evaluiert werden. Im Fall der Rekonstruktion von Strukturmodellen konnte formal nachgewiesen werden, dass für jeden Prozesslog stets ein präzises Modell gefunden werden kann. Empirische Analysen auf Basis simulierter Prozessdaten liefern zudem starke Hinweise dafür, dass die Kosten hoher Präzision i.S. erhöhter Komplexität (Laufzeit der Algorithmen und Struktur rekonstruierter Modelle) moderat ausfallen. In beiden Fällen weichen diese nicht signifikant von dem als Grundlage verwendeten Verfahren ab. Aufgrund der Tendenz präziser Strukturmodelle zu hoher Komplexität (Anzahl der Elemente) empfiehlt sich die Analyse präziser Modelle speziell bei der Betrachtung von Ausschnitten des Prozessverhaltens. Für das APClustering-Verfahren konnte gezeigt werden, dass selbst unter ungünstigen Voraussetzungen (worst-case-Ausrichtung durchgeführter Experimente) eine Vielzahl von Änderungsoperationen, die typischerweise im Hinblick auf strukturelle Veränderungen von Prozessen betrachtet werden, erkannt werden können.

7.1 Integration entwickelter Verfahren

Diese Arbeit liefert mit ihren Beiträgen eine solide Grundlage für zukünftige Forschungsarbeiten im Bereich nachgelagerter Sicherheitsanalyse von Geschäftsprozessen und die Entwicklung dedizierter Software-Tools für Prozessaudits. Die Kombination der im Rahmen dieser Arbeit entwickelten Ansätze mit existierenden Methoden zur sicherheitsorientierten Analyse von Geschäftsprozessen ermöglicht die Entwicklung von Software, die neben der Verifikation von Sicherheitseigenschaften auf Prozesslogs auch Methoden zur Erlangung eines Verständnisses der Prozessdynamik und der Prozessstruktur, die der Ausführung zugrunde liegt, enthält.

Ein Grundstein für den Transfer von Beiträgen und Leistungen dieser Arbeit wurde mit dem Start der Entwicklung einer Software zur sicherheitsorientierten Analyse von Geschäftsprozessen am Institut für Informatik und Gesellschaft (IIG) der Universität Freiburg gelegt. Erkenntnisse bzgl. der Eignung von Process Mining haben den Entwurfsprozess von SWAT (Security Workflow Analysis Toolkit)[1] stark geprägt. Neben der Bereitstellung von Methoden zur Prozessanalyse besteht die Zielsetzung von SWAT darin, Process Mining für Auditoren nutzbar zu machen und dabei die Lücke zwischen der technischen Ebene, auf der entsprechende Methoden operieren und der betriebswirtschaftlichen Ebene, auf der diese interpretiert werden müssen, zu schließen.

[1] Vgl. Accorsi, Holderer et al. 2014.

Aktuell enthält SWAT sowohl Methoden für die modellbasierte a-priori Analyse von Geschäftsprozessen als auch für die Verifikation von Sicherheitseigenschaften auf Prozesslogs. Die Beta-Phase der Entwicklung soll mit der Integration von Mechanismen zur Rekonstruktion präziser Strukturmodelle und der Erkennung der Prozessdynamik abgeschlossen werden. Implementierungen für die im Rahmen dieser Arbeit entwickelten Konzepte stellen in diesem Zusammenhang eine belastbare Ausgangsbasis dar (vgl. Abschnitt C). Die Gewährleistung einer intuitiven Benutzung innerhalb von SWAT erfordert allerdings die Entwicklung von grafischen Benutzeroberflächen für die Parametrisierung verwendeter Methoden und die Visualisierung von Ergebnissen.

Im Hinblick auf die Komplexität rekonstruierter Modelle erscheinen Methoden zur Beschränkung der Sichtweise auf Prozessdaten und die Konzentration auf Modell-Ausschnitte unter Beibehaltung der Präzision sinnvoll. Mithilfe geeigneter Filter könnte Auditoren auf diese Weise ein selektiver Zugang zum visualisierten Gesamtverhalten gewährt werden (z.B. durch Beschränkung auf Pfade, die bestimmte Akteure involvieren oder spezielle Charakteristiken bzgl. der beinhalteten Aktivitäten aufweisen). Für den Vergleich rekonstruierter Modelle mit Modellen für den geplanten Prozessverlauf besteht ein Bedarf an Methoden für die Identifikation von Unterschieden und deren Visualisierung. Dadurch könnten Auditoren dabei unterstützt werden, Abweichungen zu erkennen und deren Zustandekommen nachzuvollziehen.

7.2 Mögliche Erweiterungen

Die theoretische Grundlage der im Rahmen dieser Arbeit entwickelten Verfahren bietet ausreichenden Spielraum für Erweiterungen und die weitere Anpassung an sicherheitsspezifische Erfordernisse. Im Folgenden werden drei Möglichkeiten skizziert.

7.2.1 Erkennung struktureller Prozessveränderungen

Mithilfe des APClustering-Verfahrens können unterschiedliche Ausführungsphasen von Prozessen im zeitlichen Verlauf erkannt werden. Konkret werden dabei i.S. der *change point detection* Zeitpunkte erkannt, bei denen das Prozessverhalten signifikant von bisher als „normal“ betrachtetem Verhalten abweicht. Auf diese Weise kann nachvollzogen werden, welchen Schwankungen das Prozessverhalten unterliegt.

Steht unter der Annahme zeitlich aufeinander folgender Ursprungsmodelle für die Generierung von Ausführungspfaden die Erkennung von Zeitpunkten für den Übergang von einem Modell auf ein anderes im Vordergrund, kommt das APClustering-Verfahren an seine Grenzen. Wie bereits in Abschnitt 6 erwähnt, kann beim Übergang von einer Phase zu einer anderen nicht entschieden werden, ob dem geänderten Verhalten tatsächlich eine strukturelle Änderung zugrunde liegt oder ob sich die Prozessausführung ab dem Änderungszeitpunkt lediglich auf einen Teil des bisher möglichen Verhaltens beschränkt und sich deshalb Distanzen zwischen Aktivitätspaaren verändern. Enthält ein Ursprungsmodell bspw. eine optionale Aktivität, kann durch eine genügend große Menge aufeinander folgender Ausführungspfade im Prozesslog, die diese Aktivität nicht ausführen, der Eindruck einer Änderung entstehen, obwohl das Modell nicht verändert wurde.

Zur besseren Erkennung struktureller Veränderungen könnte das Clusteringverfahren um eine detaillierte Betrachtung von Pfaden benachbarter Ausführungsphasen erweitert werden. Enthält die Folgephase lediglich Pfade, die bereits Teil des vorherigen Ursprungsmodells waren, kann dies als Indiz für eine nicht strukturell bedingte Veränderung gewertet und die Einführung eines Änderungszeitpunkts verhindert werden. Auch die Rekonstruktion eines Strukturmodells für jede Ausführungsphase könnte dabei helfen, Änderungszeitpunkte präziser zu erfassen und Unterschiede zwischen Ausführungsphasen i.S. veränderter Kontrollflusskonstrukte zu identifizieren.

Insofern könnte neben der Erkennung von Änderungszeitpunkten auch ein Beitrag zur *change localization* erbracht werden, bei der die Frage, wo innerhalb eines Ursprungsmodells eine Änderung stattgefunden hat, im Vordergrund steht. Aktivitätspaare, deren Distanz sich ab einem Änderungszeitpunkt verändert, können als Grundlage für solche Betrachtungen dienen. Fehlende oder bisher unbekannte Aktivitäten in Kombination mit der Information, in welchen Pfaden sie auftreten und wo sich diese Pfade im Ursprungsmodell befinden, können weitere Hinweise auf den Ursprung und die Art der Änderung liefern.

7.2.2 Prozessdynamik hinsichtlich unterschiedlicher Perspektiven

Die Erkennung unterschiedlicher Ausführungsphasen mithilfe des APClustering-Verfahrens basiert momentan ausschließlich auf dem Kontrollfluss von Prozessen, da sich Änderungen innerhalb dieser Perspektive in Distanzen zwischen Aktivitätspaaren manifestieren. Bisher existieren keine Verfahren, die Änderungen auf organisationeller, temporaler oder informationeller Ebene berücksichtigen. Für die Schaffung einer ganzheitlichen Sicht auf die Dynamik von Geschäftsprozessen erscheint dies jedoch sinnvoll.

Unter Beibehaltung des fensterbasierten Ansatzes für die Erkennung von Änderungen könnte das vorgestellte Verfahren um zusätzliche Metriken erweitert werden, die typisches Prozessverhalten entlang weiterer Perspektiven charakterisieren. Denkbar erscheint die Bildung von Mengen für Akteure, die bestimmte Aktivitäten typischerweise ausführen und Intervalle bzw. Verteilungen für deren typische Dauer. Auf diese Weise könnten auf Basis signifikanter Differenzen innerhalb dieser Perspektiven ebenfalls Indikatoren für Änderungen abgeleitet werden. Die Bestimmung von Änderungszeitpunkten kann dabei für unterschiedliche Perspektiven getrennt oder integriert (evtl. durch Hinzunahme von Gewichtungen einzelner Perspektiven) erfolgen.

7.2.3 Sicherstellung struktureller Eigenschaften bei der Rekonstruktion

Die Konzentration auf nichtredundante, minimale Regionen dient der Rekonstruktion schlichter Modelle, die möglichst wenig Konstrukte enthalten, die nicht zur minimalen Definition des tatsächlichen Prozessverhaltens benötigt werden. Es scheint jedoch denkbar, das Rekonstruktionsverfahren dahingehend zu leiten, dass bewusst solche Regionen ausgewählt werden, die im Ergebnismodell bestimmte strukturelle Eigenschaften sicherstellen. Ein Beispiel für eine gewünschte Eigenschaft sind blockstrukturierte Modelle.

Obwohl diesbezüglich unterschiedliche Definitionen existieren[2], bezieht sich diese Eigenschaft meist darauf, eine Synchronisation von *Splits* und *Joins* zu fordern. Während sich Splits sich auf strukturelle Konstrukte in Prozessmodellen beziehen, die eine Verzweigung initiieren und damit Pfade entweder in eine UND oder eine ODER Beziehung zueinander setzen, führen Joins diese Pfade wieder zusammen, d.h. sie synchronisieren diese. Wird das geplante Prozessverhalten mithilfe blockstrukturierter Modelle beschrieben, kann es sinnvoll sein, diese Eigenschaft auch bei rekonstruierten Modellen zu erzwingen, um Vergleiche zwischen Soll- und Ist-Zustand zu erleichtern.

Für eine Realisierung auf technischer Ebene müsste dafür zunächst festgestellt werden, welche Regionscharakteristika für Blockstrukturen relevant sind, inwiefern dies bei der Rekonstruktion berücksichtigt werden kann und ob der Ablauf des Verfahrens grundlegend verändert werden muss.

[2] Vgl. Weske 2012; Hofstede, Aalst et al. 2010; Dumas, Aalst et al. 2005.

Anhang A

Veröffentlichungen

Teile dieser Arbeit (insbesondere zur Eignung von Process Mining-Verfahren für die Sicherheitsanalyse von Geschäftsprozessen und der Erkennung der Prozessdynamik) wurden veröffentlicht und finden sich in folgenden Beiträgen:

Accorsi, Rafael, Julius Holderer, Thomas Stocker und Richard M. Zahoransky (2014). "Security Workflow Analysis Toolkit". In: *Sicherheit*. Hrsg. von Stefan Katzenbeisser, Volkmar Lotz und Edgar R. Weippl. Bd. 228. LNI. GI, S. 433–442. ISBN: 978-3-88579-622-0.

Accorsi, Rafael und Thomas Stocker (2008). "Automated Privacy Audits Based on Pruning of Log Data". In: *EDOCW*. Hrsg. von Marten van Sinderen, oão Paulo A. Almeida, Luís Ferreira Pires und Maarten Steen. IEEE Computer Society, S. 175–182. ISBN: 978-0-7695-3720-7.

— (2009). "On Frameworks for the Visualization of Privacy Policies Implications". In: *W3C Workshop on Access Control Application Scenarios*. Luxemburg.

— (2011). "Discovering Workflow Changes with Time-Based Trace Clustering". In: *SIMPDA*. Hrsg. von Karl Aberer, Ernesto Damiani und Tharam S. Dillon. Bd. 116. Lecture Notes in Business Information Processing. Springer, S. 154–168. ISBN: 978-3-642-34043-7.

— (2012). "On the Exploitation of Process Mining for Security Audits: The Conformance Checking Case". In: *SAC*. Hrsg. von Sascha Ossowski und Paola Lecca. ACM, S. 1709–1716. ISBN: 978-1-4503-0857-1.

— (2013). "SecSy: Synthesizing Smart Process Event Logs". In: *EMISA*. Hrsg. von Reinhard Jung und Manfred Reichert. Bd. 222. LNI. GI, S. 71–84. ISBN: 978-3-88579-616-9.

Accorsi, Rafael, Thomas Stocker und Günter Müller (2013). "On the Exploitation of Process Mining for Security Audits: The Process Discovery Case". In: *SAC*. Hrsg. von Sung Y. Shin und José Carlos Maldonado. ACM, S. 1462–1468. ISBN: 978-1-4503-1656-9.

Accorsi, Rafael, Claus Wonnemann und Thomas Stocker (2011). "Towards Forensic Data Flow Analysis of Business Process Logs". In: *IMF*. Hrsg. von Holger Morgenstern, Ralf Ehlert, Sandra Frings, Oliver Göbel, Detlef Günther, Stefan Kiltz, Jens Nedon und Dirk Schadt. IEEE Computer Society, S. 3–20. ISBN: 978-0-7695-4403-8.

Stocker, Thomas (2011a). "Data Flow-Oriented Process Mining to Support Security Audits". In: *ICSOC Workshops*. Hrsg. von George Pallis, Mohamed Jmaiel, Anis Charfi, Sven Graupner, Yücel Karabulut, Sam Guinea, Florian Rosenberg, Quan Z. Sheng, Cesare Pautasso und Sonia Ben Mokhtar. Bd. 7221. Lecture Notes in Computer Science. Springer, S. 171–176. ISBN: 978-3-642-31874-0.

— (2011b). "Time-Based Trace Clustering for Evolution-Aware Security Audits". In: *Business Process Management Workshops (2)*. Hrsg. von Florian Daniel, Kamel Barkaoui und Schahram Dustdar. Bd. 100. Lecture Notes in Business Information Processing. Springer, S. 471–476. ISBN: 978-3-642-28114-3.

Stocker, Thomas, Rafael Accorsi und Tobias Rother (2013). "Computergestützte Prozessauditierung mit Process Mining". In: *HMD - Praxis Wirtschaftsinformatik* 292.

Stocker, Thomas und Frank Böhr (2013). "IF-Net: A Meta-Model for Security-Oriented Process Specification". In: *STM*. Hrsg. von Rafael Accorsi und Silvio Ranise. Bd. 8203. Lecture Notes in Computer Science. Springer, S. 191–206. ISBN: 978-3-642-41097-0.

Anhang B

Mathematische Grundlagen

Dieser Abschnitt enthält mathematische Grundlagen, die notwendig für das Verständnis der in der Arbeit vorgestellten Konzepte sind.

Definition B.1 (Multimenge). *Eine Multimenge m über einer Grundmenge $\mathcal{S}$ ist eine Abbildung $m : \mathcal{S} \to \mathbb{N}_0$, wobei für ein $s \in S$, $m(s)$ die Kardinalität von s in m angibt. Mit $deg(m) := max_{s \in S}(m(s))$ wird der Grad der Multimenge definiert; $\mathcal{S}^+$ bezeichnet die Menge aller Multimengen über $\mathcal{S}$.* ⊣

Zur einfachen Darstellung von Multimengen wird folgende Notation eingeführt: $m = \{e_1^{k_{e_1}}, \cdots, e_n^{k_{e_n}}\}$ mit $m(e_i) = k_{e_i}, \forall i \in \{1, \ldots, n\}$. Die reduzierte Grundmenge (engl. *support*) einer Multimenge m wird definiert als $\text{supp}(m) = \{s \in \mathcal{S} |\ m(s) > 0\}$. Die Relation $\subseteq$ zwischen zwei Multimengen m' und m'' wird wie folgt definiert: $m'_M \subseteq m''_M \Leftrightarrow \forall_{s \in \mathcal{S}} m'(s) \leq m''(s)$. Entsprechend impliziert $m'_M \subset m''_M$, dass gilt $m'_M \subseteq m''_M$ und $m'_M \neq m''_M$. Beim Vergleich von Multimengen mit Mengen werden Mengen als Multimengen aufgefasst. Eine Multimenge m ist k-beschränkt wenn gilt $\forall s \in S : m(s) \leq k$. Das *k-Topset* einer Multimenge m, bezeichnet mit $T_k(m)$, ist definiert als:

$$T_k(m)(s) = \begin{cases} m(s), & \textit{wenn } m(s) \geq k, \\ 0, & \textit{sonst.} \end{cases}$$

B.1 Prozesslogs

Informal ist ein Prozesslog eine durch die Ausführung eines Prozesses entstandene Sequenz von Ausführungspfaden. Jeder Ausführungspfad steht für eine Instanz des Prozesses, der alle zugehörigen Ereignisse enthält. Ereignisse beziehen sich auf die Durchführung spezifischer Aktivitäten innerhalb des Prozesses. Die Reihenfolge von Ereignissen innerhalb von Ausführungspfaden ergibt sich aus den jeweiligen Zeitpunkten ihres Auftretens. Ausführungspfade werden innerhalb dieser Arbeit auch als Abläufe oder Traces bezeichnet.

Eine Minimalanforderung für die Erstellung eines Prozesslogs ist neben der Existenz eines Zeitstempels für aufgezeichnete Ereignisse eine Äquivalenzrelation, mithilfe derer Ereignisse im Hinblick auf Prozessinstanzen partitioniert werden können. Äquivalenzklassen werden typischerweise mit einem eindeutigen Bezeichner, der *Instanz-ID*, versehen.

Definition B.2 (Prozesslog). *Sei A die Menge unterschiedlicher Bezeichner für Aktivitäten, die bei der Prozessausführung auftreten können. Ein Prozesslog (LOG) ist eine Sequenz von Abläufen $\langle \sigma_1, \ldots, \sigma_n \rangle$ mit $\sigma_i \in A^*$. Abläufe sind also Elemente der Verkettungshülle des Alphabets A und Ereignisse entsprechen Bezeichnern für Prozessaktivitäten. Es wird vorausgesetzt, dass sich durch die Zeitstempel von Ereignissen innerhalb eines Ablaufs eine lineare Ordnung (Totalordnung) ergibt und Ereignisse aufsteigend sortiert vorliegen. Die Reihenfolge von Abläufen innerhalb des Prozesslogs ergibt sich aus den Zeitstempeln der jeweils ersten Ereignisse.* ⊣

Abläufe von Prozesslogs gemäß dieser Definition werden auch als *Aktivitätssequenzen* bezeichnet. Für den Zugriff auf das i-te Ereignis eines Ablaufs t wird die Notation $t[i]$ verwendet.

Definition B.3 (Sprache eines Prozesslogs). *Die Sprache eines Logs $\mathfrak{L}$ auf Basis eines Alphabets A ergibt sich zur Menge unterschiedlicher Aktivitätssequenzen über A innerhalb von $\mathfrak{L}$, i.e. $\mathcal{L}(\mathfrak{L}) = \{\sigma \in \mathfrak{L}\} \subseteq A^*$.* ⊣

Sprachen von Prozesslogs sind grundsätzlich endlich, enthalten ausschließlich endliche Wörter und sind nicht notwendigerweise **präfixabgeschlossen**, d.h. nicht jedes Präfix eines Wortes der Sprache ist zwingend selbst ein Wort der Sprache.

Neben Zeitstempeln und Prozessaktivitäten können Ereignisse auch Kontextinformationen wie beteiligte Akteure (Personen, Prozesse, Maschinen, ...) oder weitere Attribute beinhalten, die bspw. organisationelle Aspekte abbilden (Arbeitsgruppe, Nutzerrolle, ...) oder durchgeführte Aktionen präzisieren (z.B. verwendete Datenobjekte/Dokumente).

Definition B.4 (Prozesslog mit Kontextinformation). *Sei A die Menge unterschiedlicher Bezeichner für Aktivitäten, die bei der Prozessausführung auftreten können. Ein Prozesslog mit Kontextinformation (CLOG) ist eine Sequenz von Abläufen $\langle \sigma_1, \ldots, \sigma_n \rangle$, wobei ein Ablauf definiert ist als Sequenz von Ereignissen $\sigma = \langle e_1, \ldots, e_n \rangle$. Ein Ereignis ist ein Tupel $E = (Z_E \in \mathbb{N}, A_E \in A, I_E)$. Für den Zugriff auf Elemente eines Ereignisses e wird $e.time$ (Zeitstempel Z_E), $e.act$ (Aktivität A_E) und $e.att$ (Menge weiterer Kontextattribute I_E) verwendet.* ⊣

Die Projektion eines Ablaufs auf Prozessaktivitäten wird definiert als Sequenz $t|_A = \langle a_1, \ldots, a_n \rangle, a_i = e_i.act$ und beinhaltet statt Ereignissen lediglich die zugehörigen Prozessaktivitäten mit analoger zeitlicher Ordnung. Durch sukzessives Ersetzen der Abläufe eines CLOGs mit deren Projektion auf Prozessaktivitäten entsteht dessen *Reduktion*. Die Sprache eines CLOGs entspricht der Sprache seiner Reduktion.

B.2 Transitionssysteme und Zustandsautomaten

Definition B.5 (Transitionssystem). *Ein Transitionssystem (TS) ist ein Tupel (S, E, A, s_{in}), wobei S eine Menge von Zuständen, E eine Menge von Ereignissen mit $S \cap E = \emptyset$ und $A \subseteq S \times E \times S$ eine Menge bezeichneter Kanten (Transitionen) ist, die Zustände miteinander verbindet. $S_\circ := \{s \in S \mid \nexists (s, e, s') \in A\}$ bezeichnet die Senken im TS, also Zustände ohne ausgehende Kanten. Der Initialzustand wird mit $s_{in} \in S$ angegeben.* ⊣

Ein TS heißt *endlich*, wenn S und E endlich sind. Im Rahmen dieser Arbeit werden ausschließlich endliche Transitionssysteme betrachtet. $A_e := \{(s, e, s') \mid (s, e, s') \in A\}$ wird als Abkürzung für die Kantenmenge verwendet, die den Bezeichner e tragen und $E_s := \{e \in E \mid (s, e, s') \in A \vee (s', e, s) \in A\}$ für die Ereignisse, die durch eingehende oder ausgehende Kanten mit einem Zustand s verbunden sind. Ein Zustand s ist *erreichbar* von einem anderen Zustand s' aus, wenn eine Sequenz von Transitionen $\sigma = \langle (s, e_1, s_1), \ldots, (s_k, e_k, s') \rangle$ existiert. Für Erreichbarkeit wird im allgemeinen Fall die Notation $s \xrightarrow{*} s'$ verwendet oder spezifisch $s \xrightarrow{\sigma} s'$. Die Menge der Pfade, die vom Initialzustand aus zu einem Zustand s führen, wird mit σ_s bezeichnet. Falls es genau einen Pfad gibt, wird σ_s vereinfachend als einzelnes Element aufgefasst. Für die Projektion einer Transitionssequenz auf die entsprechenden Bezeichner/Ereignisse wird die Notation $\bar{\sigma} := \langle e_1, \ldots, e_k \rangle$ verwendet. Der *Parikh-Vektor* von $\bar{\sigma}$ ist ein Vektor $\hat{\sigma} \in \mathbb{N}^{|E|}$, wobei $\hat{\sigma}(e)$ die Häufigkeit eines Ereignisses $e \in E$ innerhalb $\bar{\sigma}$ beschreibt. Existiert für einen Zustand s genau ein Vorgängerzustand, so wird dieser mit $\overset{\curvearrowleft}{s}$ bezeichnet.

Innerhalb dieser Arbeit werden *wohlgeformte* Transitionssysteme betrachtet, die folgende Axiome erfüllen:

(A1) $\forall s \in S, e \in E : |\{s' \mid \exists (s, e, s') \in A\}| < 2$ (Determinismus)

(A2) $\forall (s, e, s') \in A : s \neq s'$ (keine Zustände mit Sebstbezug)

(A3) $\forall e \in E : \exists (s, e, s') \in A$ (Vorkommnis aller Ereignisse)

(A4) $\forall s \in S \setminus \{s_{in}\} : s_{in} \xrightarrow{*} s$ (Erreichbarkeit aller Zustände)

Definition B.6 (Sprache eines Transitionssystems). *Die Sprache eines Transitionssystems $TS = (S, E, A, s_{in})$ wird auf Basis der vom Startzustand aus konstruierbaren Transitionssequenzen definiert, i.e. $\mathcal{L}(TS) := \{\bar{\sigma} \mid \exists s \in S : s_{in} \xrightarrow{\sigma} s\}$.* ⊣

Eine direkte Implikation dieser Definition ist, dass die Sprache eines TS stets präfixabgeschlossen ist, d.h. jedes Präfix eines Wortes der Sprache ist selbst ein Wort der Sprache. Im Rahmen dieser Arbeit werden solche Sprachen als „PC-Sprachen“ bezeichnet.

Definition B.7 (Zustandsautomat). *Ein Zustandsautomat (ZA) ist ein Tupel (S, E, A, s_{in}, F), dessen Träger (S, E, A, s_{in}) ein endliches TS und $F \subseteq S$ die Menge der Endzustände (engl.: accepting-states) ist. Mit $\bar{F} := F \cap S_\circ$ wird die Menge „innerer“ Endzustände bezeichnet. Ein ZA ist wohlgeformt, wenn dessen Träger wohlgeformt ist.* ⊣

Die Einführung von Endzuständen hat zwar keinen Einfluss auf die Struktur eines TS, erlaubt jedoch eine alternative Sprachdefinition, bei der die Sprache des TS nicht notwendigerweise präfixabgeschlossen ist:

Definition B.8 (Sprache eines Zustandsautomaten). *Sei $ZA = (S, E, A, s_{in}, F)$ ein Zustandsautomat. Die Sprache von ZA ergibt sich aus der Menge konstruierbarer Transitionssequenzen ausgehend vom Initialzustand s_{in}, die in einem Endzustand enden, i.e. $\mathcal{L}(ZA) := \{\bar{\sigma} \mid \exists s \in F : s_{in} \xrightarrow{\sigma} s\}$.* ⊣

B.3 Petrinetze

Definition B.9 (Petrinetz). *Ein Petrinetz (PN) ist ein Tupel (P, T, F, W, m_0), wobei P eine endliche Menge von Stellen und T eine endliche Menge von Transitionen ist, wobei $P \cap T = \emptyset$ gilt. $F \subseteq (P \times T) \cup (T \times P)$ ist die Flussrelation, die Stellen mit Transitionen (oder umgekehrt) verbindet und $W : F \to \mathbb{N}$ die Kantengewichtungsfunktion. $m_0 \in \mathbb{N}^P$ ist die initiale Markierung des Netzes und definiert dessen Ausgangszustand im Sinne der Anzahl von in Stellen enthaltenen Marken.* ⊣

Durch das *Schalten* von Transitionen können Zustände ineinander überführt werden. Dabei werden Eingangsstellen (*Preset*) der schaltenden Transition Marken gemäß den Gewichten der betreffenden Flussrelationen (auch „Kanten“) entnommen und Ausgangsstellen (*Postset*) Marken gemäß den Gewichten betreffender Flussrelationen hinzugefügt. Für das Preset einer Transition t wird die abkürzende Schreibweise $\bullet t := \{p \in P \mid \exists (p, t) \in F\}$ eingeführt, $t\bullet := \{p \in P \mid \exists (t, p) \in F\}$ entsprechend für das Postset. Eine Transition t ist *schaltbereit* (engl.: *enabled*) innerhalb einer Markierung m, wenn jede Stelle p in $\bullet t$ genügend Marken hinsichtlich der verbindenden Kanten enthält: $\forall p \in \bullet t : m(p) \geq W((p, t))$. Die Funktion $\delta : \mathbb{N}^P \to \mathcal{P}(T)$ liefert die Menge aller schaltbereiten Transitionen für eine gegebene Markierung innerhalb eines Petrinetzes. Das *Schalten* einer schaltbereiten Transition t in einer Markierung m führt zu einer neuen Markierung m'. Für Zustandsübergänge dieser Art wird die Notation $m \xrightarrow{\mathsf{t}} m'$ verwendet, wobei gilt: $\forall p \in P : m'(p) = m(p) - W((p, t)) + W((t, p))$. Eine Transitionssequenz $\gamma = \langle t_1, \ldots, t_n \rangle$ heißt *Schaltsequenz*, wenn es Markierungen $m_0, m_1, \ldots, m_n$ gibt, sodass $m_{i-1} \xrightarrow{\mathsf{t_i}} m_i \, \forall i \in [1; n]$. Die Sequenz γ heißt *abgeschlossen*, wenn es keine schaltbereite Transition in m_i gibt. Die Schreibweise $[\gamma\rangle$ wird verwendet, um die Markierung zu beschreiben, die durch das sukzessive Schalten aller Transitionen in γ erreicht wird. Die Menge aller erreichbaren Zustände eines Petrinetzes N, ausgehend von einer Markierung m_0 wird mit $[N, m_0\rangle$ bezeichnet; Γ_N beschreibt die Menge aller Schaltsequenzen, $\Gamma_N^\bullet := \{\gamma \in \Gamma_N \mid \delta([\gamma\rangle) = \emptyset\}$ die Menge aller abgeschlossenen Schaltsequenzen.

Der *Erreichbarkeitsgraph* eines Petrinetzes PN (abkürzend: RG(PN)) ist ein TS, dessen Zustände erreichbaren Markierungen und dessen Ereignisse Transitionen entsprechen. Eine Kante (m, t, m') existiert gdw. $m \xrightarrow{\mathsf{t}} m'$. Ein Petrinetz $N = (P, T, F, W, m_0)$ wird als *(k-)beschränkt* bezeichnet, wenn eine obere Schranke k für die Anzahl von in Stellen enthaltenen Marken innerhalb aller erreichbaren Markierungen existiert, i.e. $\forall p \in P, \forall m \in [N, m_0\rangle : m(p) \leq k$. Die Beschränktheit eines Petrinetzes ist also auf die

Endlichkeit seines Erreichbarkeitsgraphen zurückzuführen. Ein 1-beschränktes Petrinetz wird als *sicher* bezeichnet.

Definition B.10 (Sprache eines Petrinetzes). *Die Sprache eines Petrinetzes PN wird auf Basis möglicher Schaltsequenzen definiert. Dabei wird zwischen der* ***Sequenzsemantik*** *und der* ***vollständigen Sequenzsemantik*** *unterschieden. Bei Ersterer entspricht jede mögliche Schaltsequenz einem Wort der Sprache des Netzes, i.e. $\mathcal{L}^{\circ}(PN) := \Gamma_{PN}$. Diese Definition entspricht der gebräuchlichen Sprachdefinition für Petrinetze auf Basis von Erreichbarkeitsgraphen: $\mathcal{L}(PN) := \mathcal{L}(RG(PN))$. Die vollständige Sequenzsemantik reduziert die Sprache des Netzes auf solche Wörter, die durch abgeschlossene Schaltsequenzen zustande kommen, i.e. $\mathcal{L}^{\bullet}(PN) := \Gamma^{\bullet}_{PN}$.* ⊣

Werden Petrinetze für die Modellierung von (Geschäfts-)Prozessen verwendet, wird häufig zwischen *regulären* und *stillen* Transitionen unterschieden. Stille Transitionen stehen nicht für die Ausführung einer Prozessaktivität und werden lediglich zu Routingzwecken eingesetzt oder um eine kompaktere Repräsentation des Netzes zu erreichen. Ein typisches Beispiel sind optionale Prozessaktivitäten, die als Alternative zwischen einer stillen Transition und der Prozessaktivität selbst modelliert werden können.

Definition B.11 (Petrinetz mit stillen Transitionen). *Ein Petrinetz mit stillen Transitionen (PNS) ist ein Tupel (P, T_R, T_S, F, W, m_0), wobei $(P, T_R \cup T_S, F, W, m_0)$ ein Petrinetz ist. T_R ist die Menge regulärer, T_S die Menge stiller Transitionen.* ⊣

Die Reduktion einer Schaltsequenz $\gamma = \langle t_1, \ldots, t_n \rangle$ mit $t_i \in T_R \cup T_S$ auf reguläre Transitionen wird definiert als $\gamma|_R = \langle t'_1, \ldots, t'_m \rangle$, wobei $t'_i \in \gamma \cap T_R$ und unter Beibehaltung der Ordnungsrelation bzgl. Elementen aus $\gamma|_R$. Die Reduktion aller Schaltsequenzen eines Petrinetzes Γ_N ist $\Gamma_N|_R$, analog die Reduktion abgeschlossener Schaltsequenzen $\Gamma^{\bullet}_N|_R$.

Bei der Betrachtung (abgeschlossener) Schaltsequenzen zur Definition der Sprache von Petrinetzen, werden stille Transitionen vernachlässigt.

Definition B.12 (Sprache eines Petrinetzes mit stillen Transitionen). *Die Sprache eines Petrinetzes mit stillen Transitionen PNS ergibt sich aus Wörtern, die auf Basis von Schaltsequenzen generiert werden können, wobei lediglich reguläre Transitionen berücksichtigt werden. Die Sprachdefinition aus Def. B.10 wird folgendermaßen angepasst:*

$$\mathcal{L}^{\circ}(PNS) := \Gamma_{PNS}|_R$$
$$\mathcal{L}^{\bullet}(PNS) := \Gamma^{\bullet}_{PNS}|_R$$ ⊣

Definition B.13 (Petrinetz mit unterscheidbaren Marken). *Ein Petrinetz mit unterscheidbaren („gefärbten") Marken CPN ist ein Tupel* $(P, T, F, I, O, C, \mathcal{C}, m_0)$*, wobei* P *eine endliche Menge von Stellen und* T *eine endliche Menge von Transitionen ist, wobei* $P \cap T = \emptyset$ *gilt.* $F \subseteq (P \times T) \cup (T \times P)$ *ist die Flussrelation, die Stellen mit Transitionen (oder umgekehrt) verbindet. Marken werden anhand ihres Typs unterschieden, dabei ist* $\mathcal{C}$ *die Menge der Markentypen. Die Kapazität von Stellen für Marken eines bestimmten Typs werden mithilfe der Funktion* $C : P \times \mathcal{C} \to \mathbb{N}_0 \cup \infty$ *angegeben, wobei sich die Gesamtkapazität einer Stelle sich zu* $\zeta(p) = \sum_{c \in \mathcal{C}} C(p, c)$ *ergibt. Die Funktionen* I *und* O *werden benutzt, um die Anzahl und den Typ konsumierter (Vorbedingung) bzw. generierter Marken (Nachbedingung) beim Schalten von Transitionen zu definieren. Die Eingabefunktion* $I : (P \times T) \cap F \to \mathcal{C}^+$ *definiert die Vorbedingungen, die Ausgabefunktion* $O : (T \times P) \cap F \to \mathcal{C}^+$ *die Nachbedingungen. Der initiale Zustand des CPN wird mit einer Funktion* $m_0 : P \to \mathcal{C}^+$ *definiert.* ⊣

Um sicherzustellen, dass das CPN keine wirkungslosen Relationen enthält (i.e. Relationen zwischen Stellen und Transitionen, die keine Marken transportieren), müssen I und O folgende Bedingungen erfüllen:

- $\forall (t, p) \in (T \times P) \cap F : O(t, p) \not\equiv 0$
- $\forall (p, t) \in (P \times T) \cap F : I(p, t) \not\equiv 0$

Für konsumierte und generierte Marken wird folgende Schreibweise eingeführt:

- **Konsumierte Marken:**

  $N_c : T \to \mathcal{P}(\mathcal{C})$

  $N_c(t) = \bigcup_{i \in {}^\bullet t}(\text{supp} I(i, t))$

  $N_c|_{\mathcal{C}_c}(t) = N_c(t) \setminus \{\text{black}\}$

- **Generierte Marken:**

  $N_p : T \to \mathcal{P}(\mathcal{C})$

  $N_p(t) = \bigcup_{o \in t^\bullet}(\text{supp} O(t, o))$

  $N_p|_{\mathcal{C}_c}(t) = N_p(t) \setminus \{\text{black}\}$

Markierungen in CPNs: Eine Markierung eines CPN ist definiert als Funktion $m : P \to \mathcal{C}^+$, die den Zustand eines Netzes i.S. der Anzahl und des Typs von Marken angibt, die in den Stellen des Netzes enthalten sind. Die Relation $\leq$ zwischen zwei Markierungen m' und m'' ist folgendermaßen definiert: $m' \leq m'' \Leftrightarrow \forall_{p \in P} : m'(p) \leq m''(p)$.

Schalten von Transitionen in CPNs: Eine Transition $t \in T$ ist schaltbereit in einem CPN mit Markierung m gdw.

- $\forall_{p \in {}^\bullet t} : I(p,t) \leq m(p)$

  (genügend viele Marken mit adäquatem Typ in Eingangsstellen)

- $\forall_{p \in t^\bullet} \forall_{c \in \mathcal{C}} : m(p)(c) + O(p,t)(c) \leq C(p,c)$

  (ausreichende Kapazität in Ausgangsstellen, um die beim Schalten generierten Marken aufnehmen zu können)

Das Schalten einer Transition $t \in T$ innerhalb einer Markierung m führt zu einer Markierung m', wobei gilt:

- $\forall_{p \in {}^\bullet t} : m'(p) = m(p) - I(p,t)$
- $\forall_{p \in t^\bullet} : m'(p) = m(p) + O(t,p)$
- $\forall_{p \in P \setminus \{{}^\bullet t \cup t^\bullet\}} : m'(p) = m(p)$

Die Beschränktheit regulärer Petrinetze wird auf natürliche Weise auf CPNs erweitert. Ein CPN ist k-beschränkt gdw. $[N,m\rangle$ endlich ist, i.e.
$\forall_{m' \in [N,m\rangle} \forall_{p \in P} : \sum_{c \in C} M'(p)(c) \leq k$. Analog zu Def. B.11 können CPNs durch Betrachtung von stillen Transitionen auf CPNSs erweitert werden.

Definition B.14 (Kapazitätsbeschr. Petrinetz mit unterscheidbaren Marken). *Ein kapazitätsbeschränktes Petrinetz mit unterscheidbaren Marken (CCPN) ist ein CPN, für das gilt:* $\forall_{p \in P} : \zeta(p) \in \mathbb{N}$ ⊣

Die maximale Gesamtkapazität der Stellen des CPN ist eine obere Schranke für die maximale Anzahl von Marken, die in einer Stelle enthalten sein können. Ein CCPN ist also k-beschränkt, wobei $k \leq \max_{p \in P}(\zeta(p))$.

Anhang C

Implementierung entwickelter Verfahren

Für die theoretischen Konzepte dieser Arbeit existieren Proof-of-Concept-Implementierungen, die für Experimente zur Evaluierung der GENET$^+$, GENET* und APClustering-Verfahren verwendet wurden. Um weitere Forschungsarbeiten und laufende Projekte am Lehrstuhl für Telematik optimal zu unterstützen, wurde entstandener Java-Programmcode modularisiert. Im Folgenden wird die Funktionalität jedes dadurch entstandenen Projekts kurz skizziert:

JGENET: Proof-of-Concept-Implementierung der entwickelten Verfahren zur Rekonstruktion präziser Prozessmodelle

- Regionskonzepte (Regionen, *ER*-Multimengen etc.)
- Methoden zur Sicherstellung der EC-Eigenschaft innerhalb von TS
- Unterstützung für Ereignis- und Zustandsspaltung
- Heuristiken für die Behandlung redundanter Regionen
- Datenbezogene Extraktion von Transitionssystemen aus Prozesslogs
- Metriken vollständiger Sequenzsemantik (VSS-Deckung und VSS-Affinität)
- Multi-Threading Implementierung des GENET-Ansatzes
- Multi-Threading Implementierung des GENET$^+$-Ansatzes
- JGENET baut auf SEPIA, JAWL und SERAM auf

TORLOC: Proof-of-Concept-Implementierung der entwickelten Verfahren zur Erkennung der Prozessdynamik

- Extraktion von Distanzen für Aktivitätspaare (Distanzmatritzen)
- Implementierung des APClustering-Ansatzes
- Grafisches Frontend für die Anwendung des Verfahrens auf Prozesslogs im Reintextformat, dessen Parametrisierung und die Visualisierung von Clustergraphen.

SEPIA: Bibliothek für Petrinetze

- P/T-Netze
- Netze mit unterscheidbaren Markentypen (CPN)
- IFnet zur Spezifikation von Prozessen mit Sicherheitsmerkmalen
- Analysekontexte für IFnet (Sicherheitsstufen, Freigabelevel)
- Unterstützung grafischer Netzinformation (Koordinaten für Transitionen, Stellen etc.)
- Erstellung von Erreichbarkeitsgraphen
- Traversierung von Petrinetzen (randomisiert und stochastisch)
- PNML standardkonformes Sterilisieren und Parsen von Petrinetzen
- XML-Schemata zur Validierung von Netzen frei verfügbar unter http://ifnml.process-security.de
- Quelloffen verfügbar unter http://sourceforge.net/projects/sepiaframework/

JAWL: Bibliothek für die Verwendung von Prozesslogs

- Parsen und Serialisieren von TXT, MXML und XES Logs
- Transformation von CSV zu MXML
- XES-kompatible Datenflusserweiterung für die standardkonforme Unterstützung kombinierter Kontroll-/Datenfluss-Rekonstruktion.
- Quelloffen verfügbar unter http://sourceforge.net/projects/jawl/

SERAM: Bibliothek für Sicherheitskonzepte

- Konzepte der Zugangskontrolle (ACL, RBAC)
- Unterstützung persistenter Speicherung definierter Modelle (Sterilisierung + Parsen)
- Grafisches Frontend für die Definition von Sicherheitsmodellen
- Replayingmethoden zum Abspielen aufgezeichneter Ausführungspfade aus Prozesslogs auf Modellen.
- Beurteilung der Konformität von Prozesslog und Modell anhand von Metriken vollständiger Sequenzsemantik
- Quelloffen verfügbar unter http://sourceforge.net/projects/seram/

Anhang D

Verwendete Modelle zur Erstellung von Prozesslogs

Dieser Anhang beinhaltet Petrinetze, die als Grundlage für die Generierung von Testdaten zur Evaluierung entwickelter Ansätze genutzt wurden. Größtenteils entstammen diese Modelle der Dissertation von de Medeiros[1]. Auszüge der dort zur Evaluierung eines genetischen Rekonstruktionsverfahrens genutzten Modellkollektion finden sich aufgrund der Abdeckung verschiedener Prozesscharakteristika in vielen Arbeiten aus dem Bereich Process Mining wieder. Um die Funktionsweise des entwickelten Verfahrens zur präzisen Kontrollfluss-Rekonstruktion demonstrieren zu können, wurden einige Modelle um zusätzliche Konstrukte erweitert. Diese erlauben eine vorzeitige Beendigung der Prozessausführung in einer Weise, die innerhalb generierter Prozesslogs die Existenz von Aktivitätssequenzen begünstigt, die ein Präfix enthalten, welches selbst wiederum als Aktivitätssequenz im selben Prozesslog auftritt. Tab. D.1 gibt einen Überblick über die einzelnen Modelle und deren Charakteristika. Angepasste Modelle sind mit * gekennzeichnet; konkrete Veränderungen innerhalb der verwendeten Modelle jeweils mit gestrichelten Linien abgesetzt[2].

Neben sequentiellen Abfolgen von Aktivitäten beinhalten einige Modelle Entscheidungsknoten, an denen eine Auswahl bzgl. unterschiedlicher (exklusiver) Folgepfade getroffen wird. Solche Entscheidungen werden entweder „frei“ getroffen (*free-choice*) oder von zuvor ausgeführten Aktivitäten beeinflusst. Diese Kombination aus Auswahl und Synchronisation wird als *non free-choice* (NFC) bezeichnet. Für Petrinetze wird für die Abwesenheit von NFC-Konstrukten üblicherweise gefordert, dass sich bei Transitionen mit gemeinsamen Vorgängerstellen, die Mengen der Vorgängerstellen entsprechen. Das Modell betaSimplified enthält ein Beispiel für dieses Konstrukt. Die Aktivitäten *Pay For Parking* und *Travel By Train* werden mit den beiden Aktivitäten *Travel By Car* und *Travel By Train* synchronisiert um einen sinnvollen Prozessablauf zu gewährleisten. Nach Ausführung von *Go Home* steht bereits fest, welcher der Folgepfade ausgeführt wird. In einigen

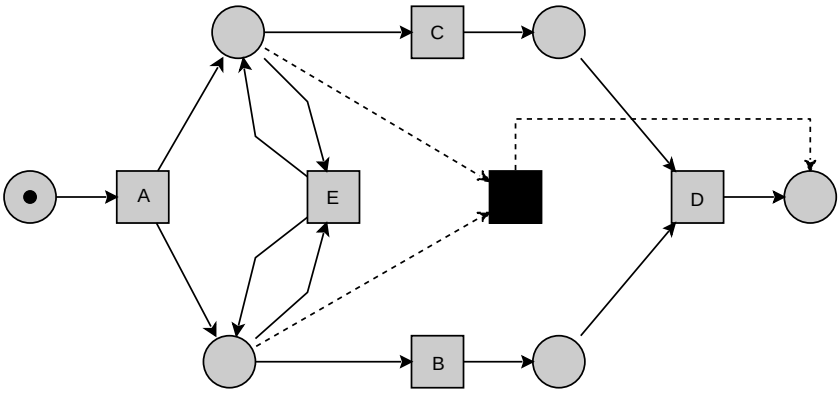
[1] Vgl. Alves de Medeiros 2006.

[2] Prozessmodelle, die innerhalb dieser Arbeit zur Erstellung synthetischer Prozesslogs verwendet wurden, können unter http://prorepo.process-security.de/g/a5c9ecab eingesehen werden.

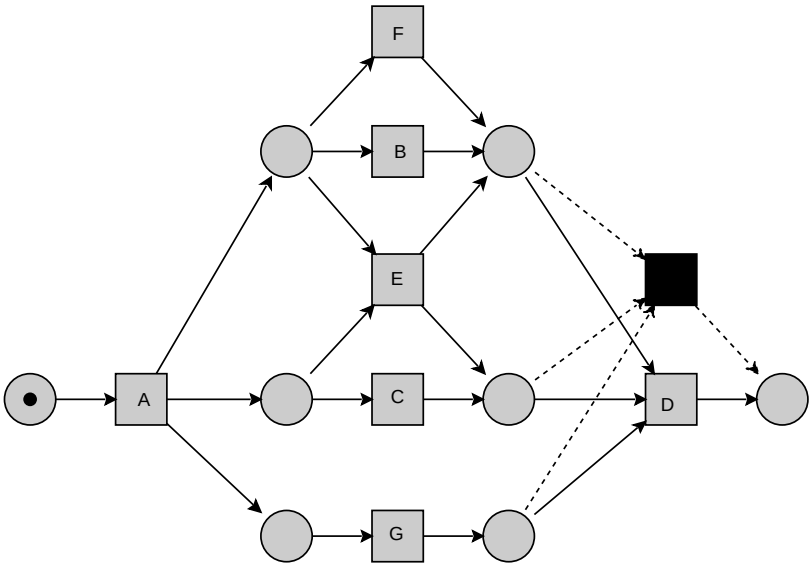
Netz	Auswahl	Parallelität	Schleifen	NFC	Unsichtbare Aktivitäten	Duplikate
a10Skip*	✓	✓			✓	
a5*	✓	✓	✓		✓	
a6nfc*	✓	✓			✓	
a7*	✓	✓			✓	
a12*	✓	✓			✓	
al1*	✓	✓	✓		✓	
choice*	✓				✓	
driversLicense*	✓			✓	✓	
herbstFig3p4*	✓	✓	✓		✓	
herbstFig5p19	✓	✓			✓	✓
herbstFig6p18*	✓		✓		✓	
herbstFig6p31*	✓					✓
herbstFig6p34*	✓	✓	✓		✓	✓
herbstFig6p38*		✓			✓	✓
herbstFig6p39		✓			✓	✓
herbstFig6p41*	✓	✓			✓	✓
herbstFig6p42	✓	✓			✓	✓
l1lSkip*	✓	✓	✓		✓	

Tabelle D.1.: Charakteristika verwendeter Petrinetze.

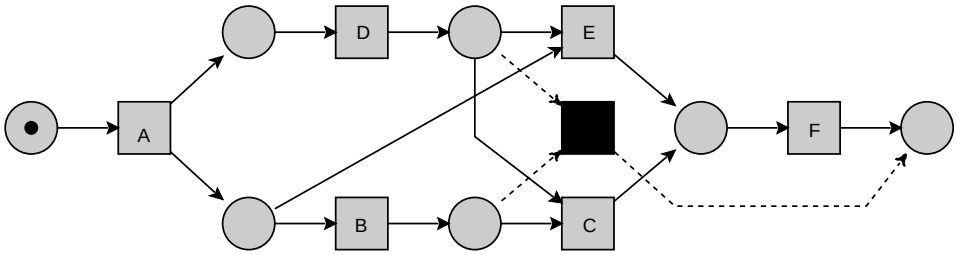
Fällen enthalten verwendete Modelle Schleifen, die mehrfache Ausführungen von Teilpfaden erlauben. Schleifen können gebunden (d.h. es existiert eine obere Schranke für die maximale Zahl an Durchläufen) oder ungebunden sein. Bei der Generierung von Prozesslogs aus gegebenen Modellen ist bei Existenz ungebundener Schleifen die Anzahl unterschiedlicher, in sich abgeschlossener Prozessausführungen typischerweise proportional zur Anzahl generierter Ausführungen. Bei beschränkten Schleifen kann die Zahl und Ausprägung möglicher Ausführungen im Voraus bestimmt werden. Die hier verwendeten Modelle enthalten mit Ausnahme von herbstFig3p4* ausschließlich unbeschränkte Schleifen. Unsichtbare Aktivitäten (in Petrinetzterminologie stille Transitionen) werden häufig zu Routingzwecken eingesetzt, um optionale Teilpfade oder Rücksprünge bzw. Wiederholungen in einfacher Weise zu realisieren. Das Stattfinden einer unsichtbaren Aktivität ist innerhalb eines Prozesslogs ohne Kenntnis des Ursprungsmodells nicht erkennbar. In Modell l1lSkip* werden unsichtbare Aktivitäten zur Realisierung sog. „kurzer Schleifen“ (engl.: *short loops*) bzgl. der Aktivitäten C und B genutzt, Modell herbstFig6p39 enthält ein Beispiel für den Einsatz unsichtbarer Aktivitäten zur Umsetzung optionaler Aktivitäten (A und B). Duplikate treten in Erscheinung, wenn eine Aktivität im Prozessverlauf mehrfach auftreten kann, es aufgrund unterschiedlicher Vor- und Nachbedingungen jedoch entweder nicht möglich oder sinnvoll ist, einen konsistenten Ablauf mithilfe von Schleifen zu gewährleisten. Im Modell betaSimplified tritt die Aktivität *Travel By Car* zweimal auf, zu Beginn und am Ende des Prozesses.



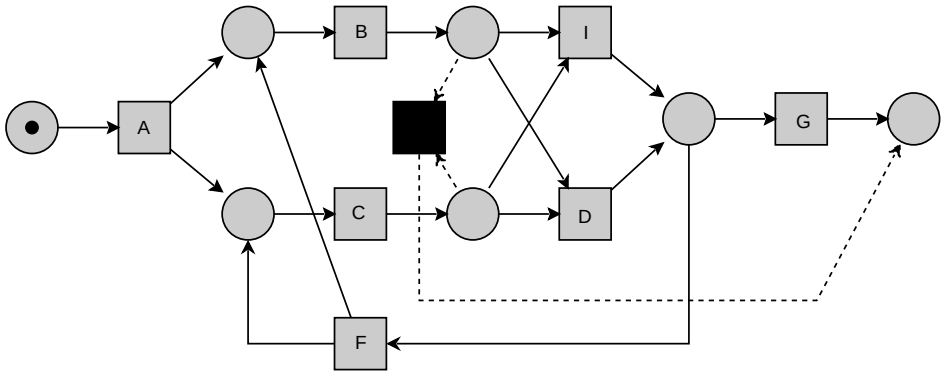
Netz a5*.



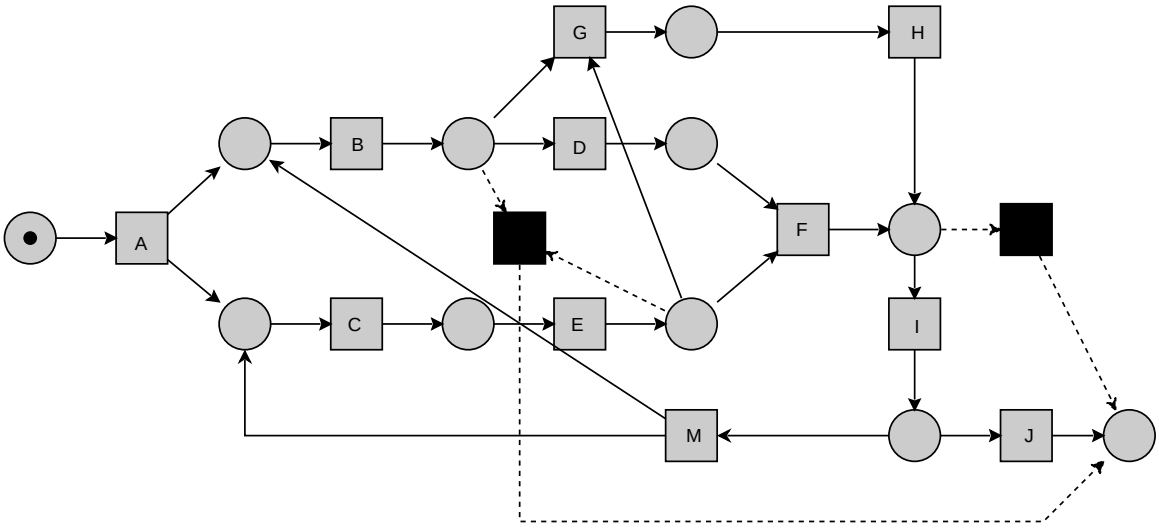
Netz a7*.



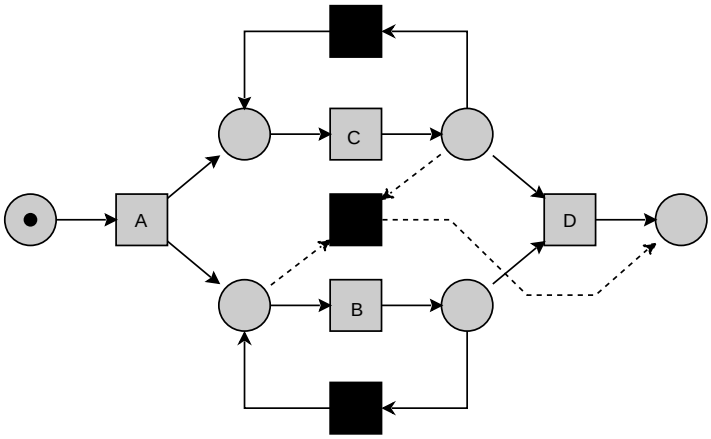
Netz a6nfc*.



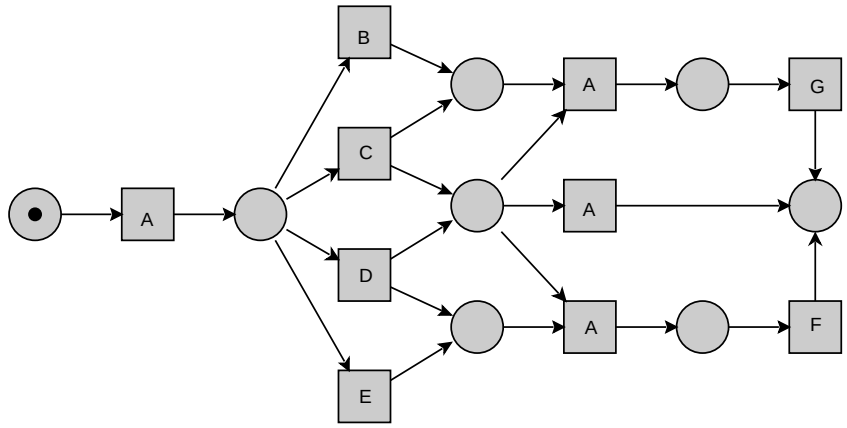
Netz al1*.



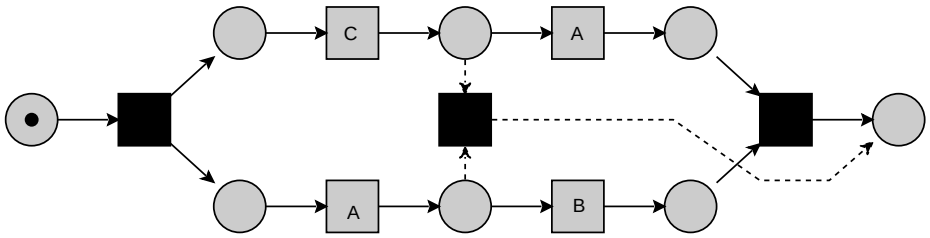
Netz al2*.



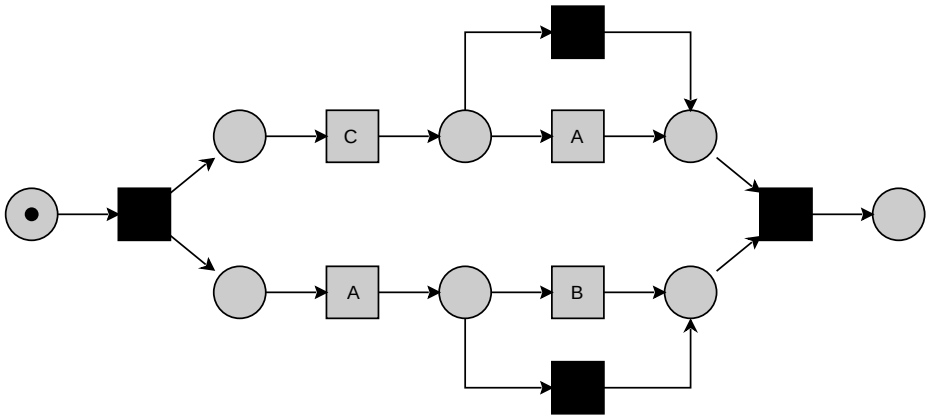
Netz l1lSkip*.



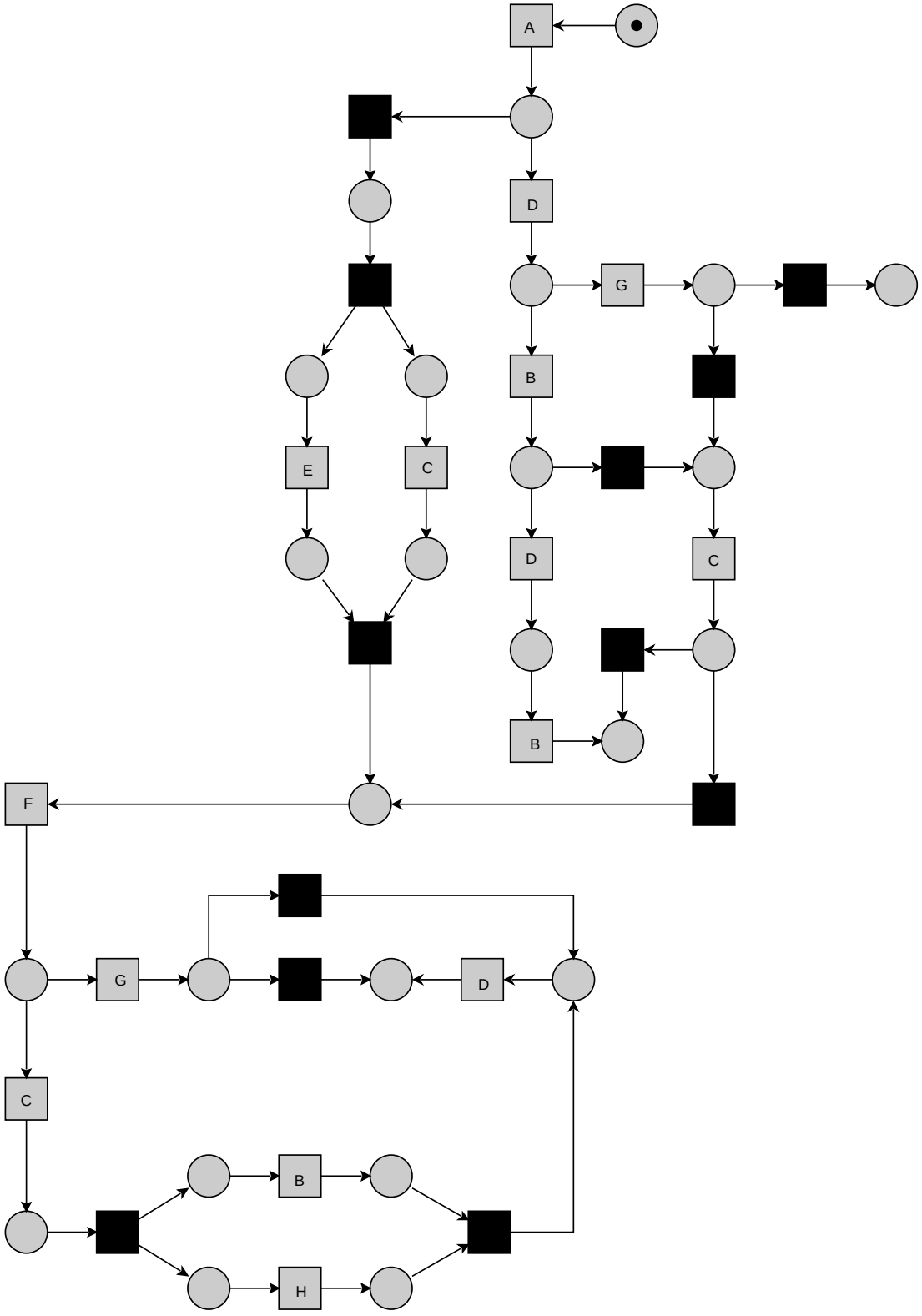
Netz herbstFig6p31*.



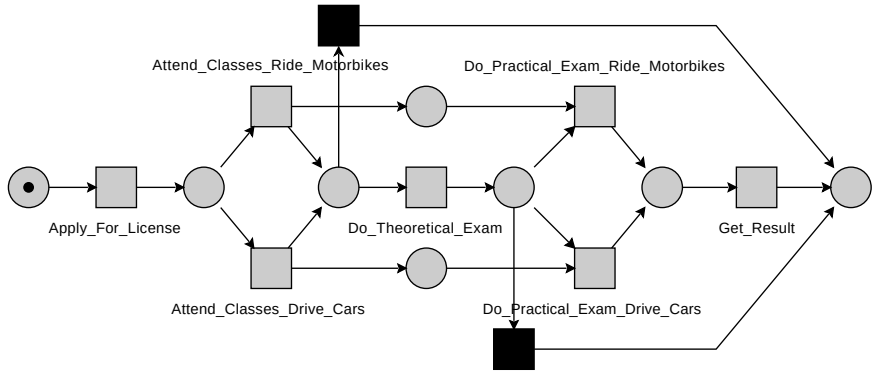
Netz herbstFig6p38*.



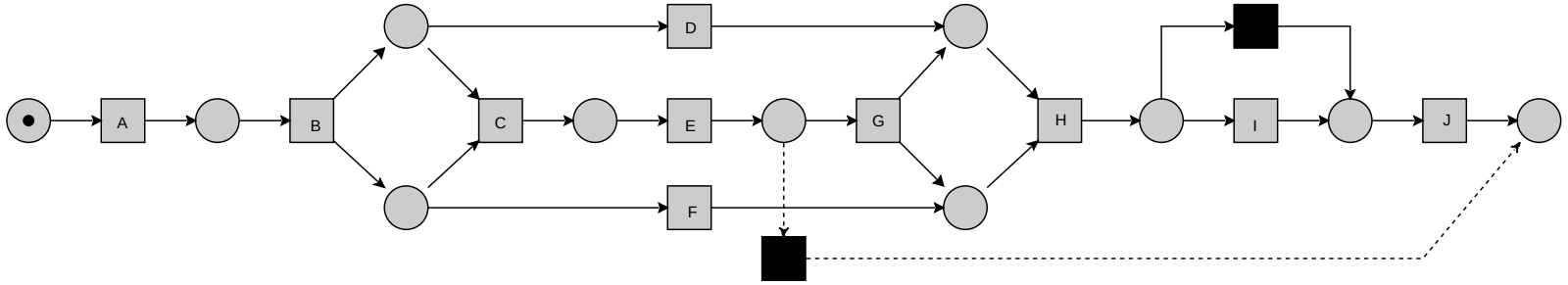
Netz herbstFig6p39.



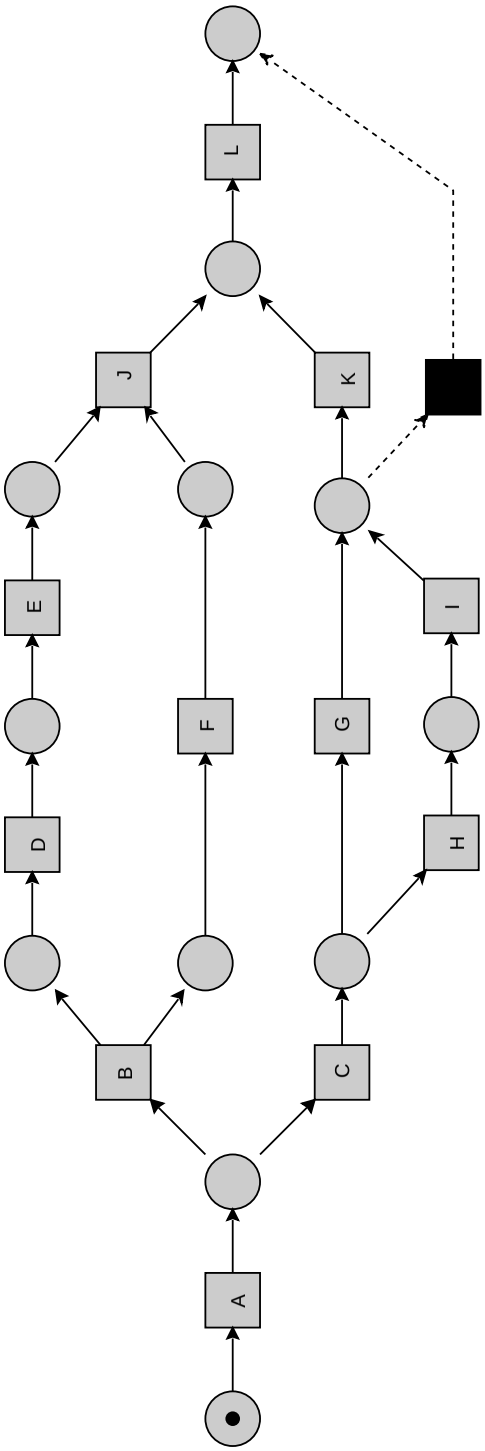
Netz herbstFig6p42.



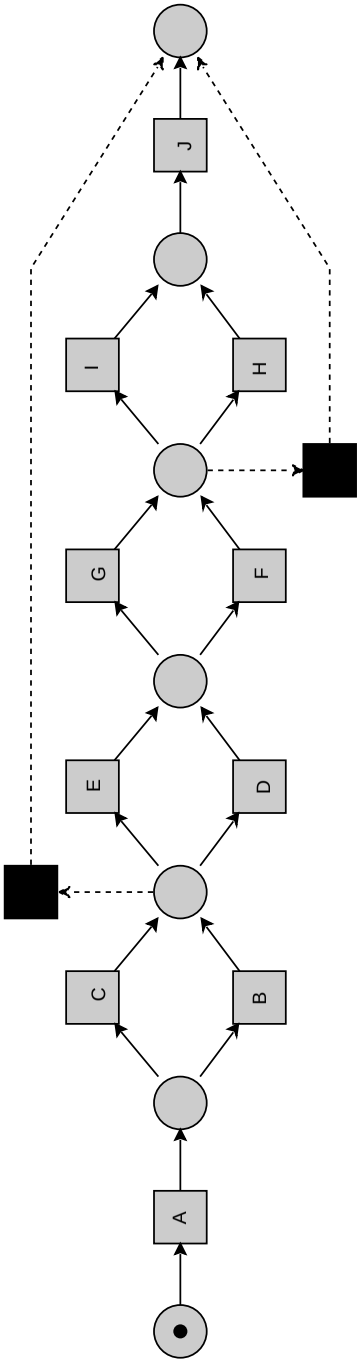
Netz driversLicense*.



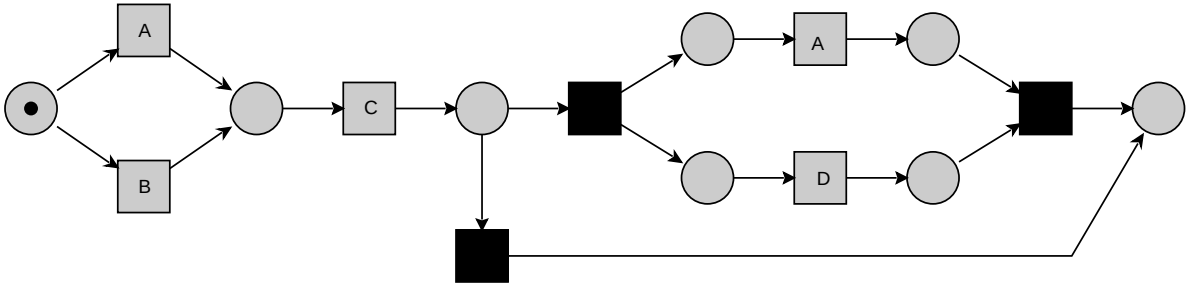
Netz a10Skip*.



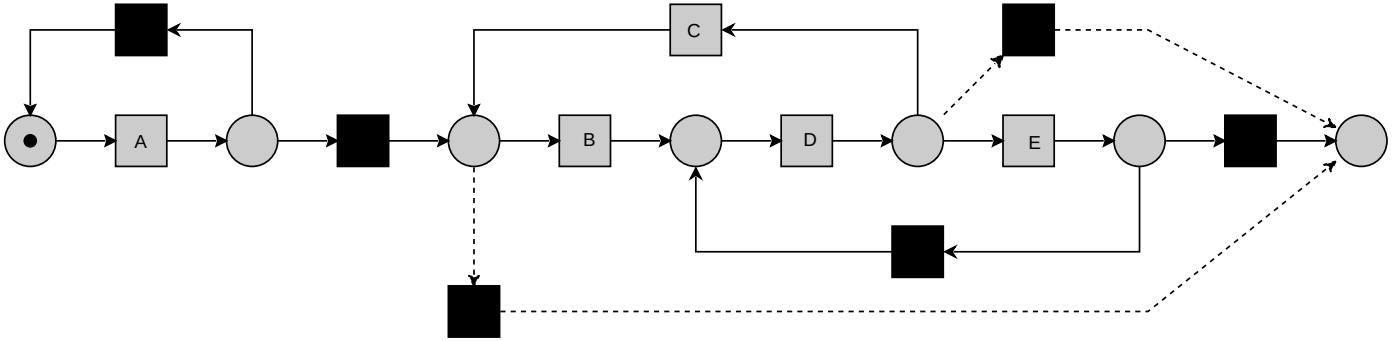
Netz a12*.



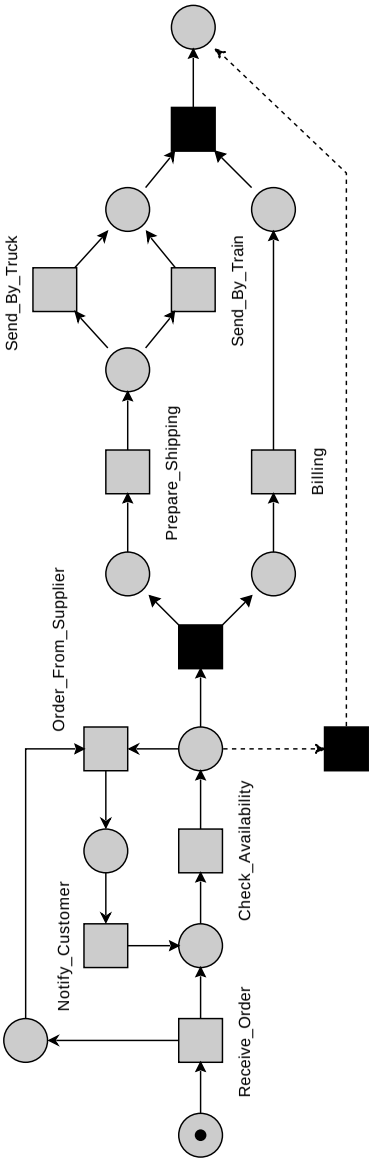
Netz choice*.



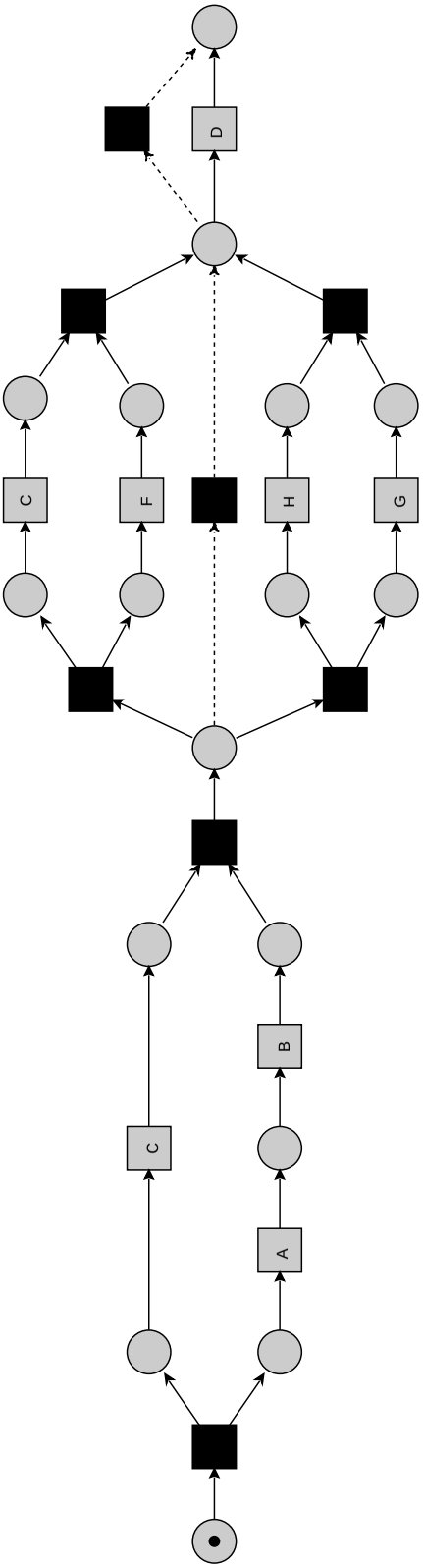
Netz herbstFig5p19.



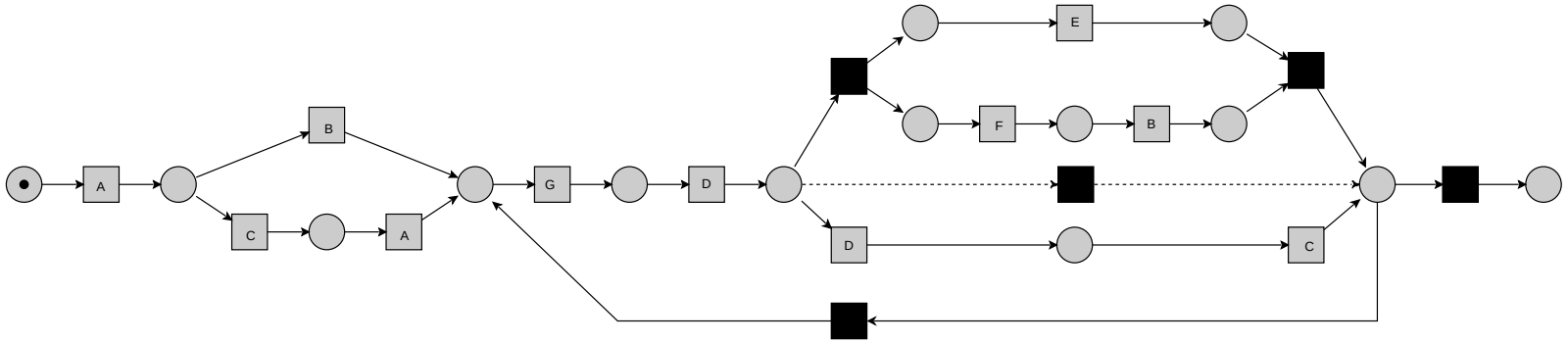
Netz herbstFig6p18*.



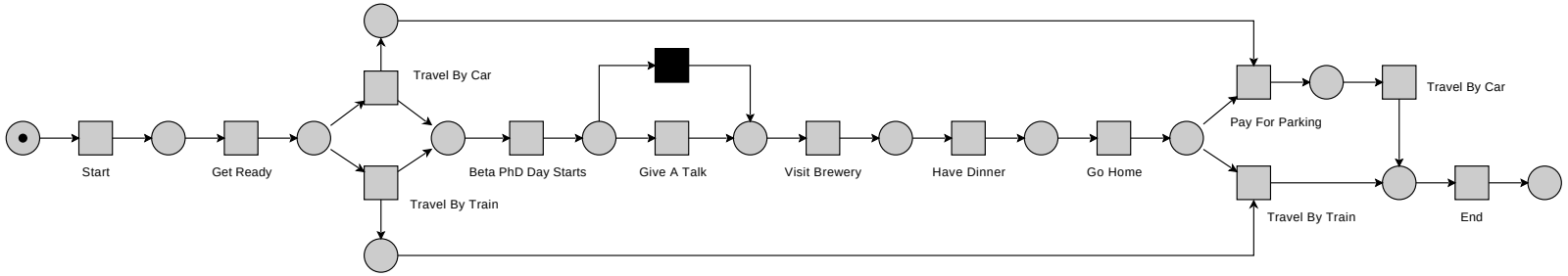
Netz herbstFig3p4*.



Netz herbstFig6p41*.



Netz herbstFig6p34*.



Netz betaSimplified.

Anhang E

Tabellen zur Erkennungsrate des APClustering-Verfahrens

In diesem Anhang finden sich Tabellen zur erreichten Präzision bei der Erkennung von Änderungen der Kategorien (AP1b-AP9).

Netz		Pfade	*rel*	*par*	Präzision ($m_c = 6$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,28	0,03	$0,19^{0,042}$	$0,45^{0,018}$	$0,92^{0,011}$	0,95	0,95	✓	0,94
		2*1000	0,28	0,03	$0,76^{0,145}$	1,00	1,00	1,00	1,00	✓	1,00
	Variante 2	2*100	0,28	0,03	$0,18^{0,037}$	$0,46^{0,005}$	$0,49^{0,002}$	$0,48^{0,002}$	$0,45^{0,005}$		0,47
		2*1000	0,28	0,03	$0,37^{0,069}$	$0,40^{0,004}$	$0,38^{0,002}$	$0,75^{0,048}$	$0,35^{0,205}$		0,50
	Variante 3	2*100	0,28	0,03	$0,21^{0,049}$	$0,23^{0,004}$	$0,32^{0,001}$	0,32	0,32	✓	0,32
		2*1000	0,28	0,03	$0,73^{0,125}$	$0,92^{0,033}$	1,00	$0,99^{0,002}$	1,00		1,00
herbstFig6p41*	Variante 1	2*100	0,43	0,16	$0,16^{0,078}$	$0,40^{0,100}$	0,42	0,42	$0,82^{0,008}$		0,56
		2*1000	0,43	0,16	$0,66^{0,197}$	1,00	1,00	$0,99^{0,002}$	$0,97^{0,013}$		0,99
	Variante 2	2*100	0,43	0,16	$0,15^{0,042}$	$0,23^{0,006}$	$0,31^{0,002}$	$0,42^{0,006}$	$0,55^{0,029}$	✓	0,43
		2*1000	0,43	0,16	$0,34^{0,111}$	0,99	0,99	0,99	0,99	✓	0,99
	Variante 3	2*100	0,43	0,16	$0,10^{0,004}$	$0,21^{0,009}$	$0,76^{0,054}$	0,92	0,94	✓	0,87
		2*1000	0,43	0,16	$0,50^{0,122}$	$0,52^{0,009}$	$0,51^{0,006}$	$0,52^{0,012}$	0,50		0,51
herbstFig6p42	Variante 1	2*100	0,63	0,42	$0,12^{0,012}$	$0,12^{0,002}$	$0,24^{0,002}$	$0,28^{0,023}$	$0,31^{0,001}$	✓	0,27
		2*1000	0,63	0,42	$0,11^{0,004}$	0,03	0,03	0,03	$0,36^{0,226}$		0,14
	Variante 2	2*100	0,63	0,42	$0,17^{0,077}$	0,11	$0,19^{0,006}$	0,15	0,15		0,16
		2*1000	0,63	0,42	$0,18^{0,014}$	$0,26^{0,001}$	$0,31^{0,001}$	$0,39^{0,006}$	$0,49^{0,002}$	✓	0,40
	Variante 3	2*100	0,63	0,42	$0,10^{0,008}$	$0,20^{0,005}$	$0,22^{0,003}$	$0,16^{0,005}$	$0,23^{0,007}$		0,20
		2*1000	0,63	0,42	$0,13^{0,008}$	0,21	$0,22^{0,001}$	$0,29^{0,012}$	0,50	✓	0,34
herbstFig6p18*	Variante 1	2*100	0,58	0,47	$0,10^{0,002}$	$0,15^{0,001}$	$0,19^{0,002}$	$0,29^{0,001}$	$0,78^{0,062}$	✓	0,42
		2*1000	0,58	0,47	$0,10^{0,002}$	$0,29^{0,014}$	$0,54^{0,006}$	0,50	0,50		0,51
	Variante 2	2*100	0,56	0,44	$0,17^{0,005}$	$0,17^{0,003}$	$0,18^{0,005}$	$0,13^{0,002}$	0,10		0,13
		2*1000	0,58	0,47	$0,05^{0,003}$	$0,12^{0,009}$	$0,24^{0,004}$	$0,39^{0,055}$	$0,36^{0,003}$		0,33
	Variante 3	2*100	0,58	0,47	$0,13^{0,002}$	$0,14^{0,005}$	$0,21^{0,007}$	$0,27^{0,012}$	$0,16^{0,008}$		0,21
		2*1000	0,58	0,47	$0,14^{0,007}$	$0,26^{0,001}$	0,28	$0,39^{0,004}$	$0,17^{0,031}$		0,28
herbstFig6p34*	Variante 1	2*100	0,69	0,61	$0,15^{0,008}$	$0,20^{0,003}$	$0,24^{0,002}$	$0,33^{0,005}$	$0,90^{0,030}$	✓	0,49
		2*1000	0,69	0,61	$0,09^{0,004}$	$0,16^{0,010}$	$0,23^{0,010}$	0,33	$0,34^{0,001}$	✓	0,30
	Variante 2	2*100	0,69	0,61	$0,16^{0,003}$	$0,22^{0,003}$	$0,30^{0,002}$	$0,34^{0,002}$	$0,43^{0,038}$	✓	0,36
		2*1000	0,69	0,61	$0,13^{0,009}$	$0,20^{0,003}$	$0,29^{0,004}$	$0,42^{0,007}$	$0,52^{0,031}$	✓	0,41
	Variante 3	2*100	0,69	0,61	$0,12^{0,002}$	$0,20^{0,011}$	$0,60^{0,065}$	0,50	$0,74^{0,062}$		0,61
		2*1000	0,69	0,61	$0,12^{0,007}$	$0,32^{0,025}$	$0,89^{0,041}$	1,00	1,00	✓	0,96
al2*	Variante 1	2*100	0,83	0,67	$0,19^{0,016}$	$0,37^{0,010}$	$0,35^{0,005}$	$0,31^{0,002}$	$0,66^{0,073}$		0,44
		2*1000	0,83	0,67	$0,14^{0,019}$	0,06	0,06	0,06	0,06		0,06
	Variante 2	2*100	0,83	0,67	$0,13^{0,004}$	0,18	$0,20^{0,001}$	$0,18^{0,001}$	$0,25^{0,019}$		0,21
		2*1000	0,83	0,67	$0,17^{0,016}$	$0,24^{0,018}$	$0,41^{0,026}$	$0,43^{0,007}$	$0,45^{0,006}$	✓	0,43
	Variante 3	2*100	0,83	0,67	$0,11^{0,001}$	$0,25^{0,002}$	$0,26^{0,003}$	$0,31^{0,002}$	$0,25^{0,007}$		0,27
		2*1000	0,83	0,67	$0,18^{0,016}$	$0,82^{0,058}$	$0,94^{0,027}$	1,00	$1,00^{0,001}$		0,98

Erreichte Präzision von APClustering für Änderungstyp (AP1a) bei $m_c = 6$.

Netz		Pfade	rel	par	Präzision ($m_c = 10$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,28	0,03	$0,25^{0,069}$	$0,43^{0,005}$	0,95	0,95	0,95	✓	0,95
		2*1000	0,28	0,03	$0,76^{0,142}$	1,00	1,00	1,00	1,00	✓	1,00
	Variante 2	2*100	0,28	0,03	$0,29^{0,093}$	$0,44^{0,008}$	0,49	0,49	$0,45^{0,005}$		0,48
		2*1000	0,28	0,03	$0,38^{0,071}$	$0,40^{0,004}$	$0,38^{0,002}$	$0,75^{0,048}$	$0,35^{0,205}$		0,50
	Variante 3	2*100	0,28	0,03	$0,18^{0,037}$	$0,23^{0,001}$	0,32	0,32	0,32	✓	0,32
		2*1000	0,28	0,03	$0,65^{0,124}$	$0,91^{0,037}$	1,00	$0,99^{0,001}$	1,00		1,00
herbstFig6p41*	Variante 1	2*100	0,43	0,16	$0,29^{0,126}$	$0,32^{0,053}$	$0,52^{0,031}$	0,41	$0,82^{0,009}$		0,58
		2*1000	0,43	0,16	$0,35^{0,046}$	0,50	$0,59^{0,036}$	$0,99^{0,002}$	$0,72^{0,061}$		0,77
	Variante 2	2*100	0,43	0,16	$0,22^{0,074}$	$0,20^{0,005}$	$0,32^{0,002}$	$0,42^{0,006}$	$0,55^{0,029}$		0,43
		2*1000	0,43	0,16	$0,35^{0,115}$	0,99	0,99	0,99	0,99	✓	0,99
	Variante 3	2*100	0,43	0,16	$0,23^{0,070}$	$0,24^{0,010}$	$0,76^{0,054}$	0,92	0,94	✓	0,87
		2*1000	0,43	0,16	$0,63^{0,172}$	$0,80^{0,058}$	$0,76^{0,062}$	$0,81^{0,058}$	0,99		0,85
herbstFig6p42	Variante 1	2*100	0,63	0,42	$0,32^{0,150}$	$0,12^{0,001}$	$0,25^{0,002}$	$0,27^{0,015}$	$0,31^{0,001}$		0,27
		2*1000	0,63	0,42	$0,14^{0,011}$	$0,03^{0,001}$	0,02	0,03	$0,36^{0,226}$		0,14
	Variante 2	2*100	0,63	0,42	$0,23^{0,068}$	$0,16^{0,001}$	$0,22^{0,003}$	$0,28^{0,019}$	$0,39^{0,007}$		0,30
		2*1000	0,63	0,42	$0,15^{0,013}$	$0,23^{0,001}$	$0,31^{0,001}$	$0,39^{0,006}$	$0,49^{0,002}$	✓	0,40
	Variante 3	2*100	0,63	0,42	$0,25^{0,071}$	$0,20^{0,002}$	0,20	0,24	$0,27^{0,002}$		0,24
		2*1000	0,63	0,42	$0,16^{0,012}$	$0,29^{0,004}$	$0,29^{0,002}$	$0,46^{0,079}$	$0,99^{0,001}$	✓	0,58
herbstFig6p18*	Variante 1	2*100	0,58	0,47	$0,13^{0,001}$	0,14	$0,15^{0,002}$	$0,25^{0,001}$	$0,71^{0,069}$	✓	0,37
		2*1000	0,58	0,47	$0,10^{0,003}$	$0,30^{0,012}$	$0,52^{0,010}$	0,50	0,50		0,50
	Variante 2	2*100	0,56	0,44	$0,12^{0,002}$	$0,17^{0,001}$	$0,19^{0,007}$	$0,15^{0,006}$	0,10		0,14
		2*1000	0,58	0,47	$0,06^{0,006}$	$0,14^{0,009}$	$0,22^{0,003}$	$0,38^{0,058}$	$0,36^{0,003}$		0,32
	Variante 3	2*100	0,58	0,47	$0,12^{0,001}$	$0,26^{0,010}$	$0,26^{0,003}$	$0,33^{0,007}$	$0,21^{0,010}$		0,27
		2*1000	0,58	0,47	$0,11^{0,005}$	$0,26^{0,001}$	0,28	0,28	$0,11^{0,013}$		0,23
herbstFig6p34*	Variante 1	2*100	0,69	0,61	$0,18^{0,041}$	$0,20^{0,001}$	$0,23^{0,002}$	$0,33^{0,005}$	$0,90^{0,030}$	✓	0,49
		2*1000	0,69	0,61	$0,08^{0,007}$	$0,16^{0,007}$	$0,23^{0,008}$	0,33	$0,34^{0,001}$	✓	0,30
	Variante 2	2*100	0,69	0,61	$0,22^{0,070}$	$0,19^{0,002}$	$0,28^{0,007}$	$0,42^{0,004}$	$0,52^{0,022}$		0,41
		2*1000	0,69	0,61	$0,11^{0,009}$	$0,19^{0,003}$	$0,29^{0,004}$	$0,42^{0,007}$	$0,52^{0,031}$	✓	0,41
	Variante 3	2*100	0,69	0,61	$0,16^{0,039}$	$0,17^{0,007}$	$0,60^{0,065}$	0,50	$0,86^{0,045}$		0,65
		2*1000	0,69	0,61	$0,10^{0,002}$	$0,30^{0,026}$	$0,89^{0,041}$	1,00	1,00	✓	0,96
al2*	Variante 1	2*100	0,83	0,67	$0,19^{0,010}$	$0,28^{0,006}$	$0,21^{0,001}$	$0,40^{0,007}$	$0,79^{0,058}$		0,47
		2*1000	0,83	0,67	$0,12^{0,016}$	0,06	0,06	0,06	0,06		0,06
	Variante 2	2*100	0,83	0,67	$0,18^{0,037}$	0,16	$0,20^{0,002}$	$0,14^{0,003}$	$0,24^{0,026}$		0,20
		2*1000	0,83	0,67	$0,16^{0,016}$	$0,24^{0,016}$	$0,42^{0,025}$	$0,40^{0,007}$	$0,42^{0,007}$		0,41
	Variante 3	2*100	0,83	0,67	$0,17^{0,005}$	$0,27^{0,002}$	$0,44^{0,007}$	0,48	$0,47^{0,001}$		0,46
		2*1000	0,83	0,67	$0,15^{0,010}$	$0,77^{0,063}$	$0,90^{0,040}$	1,00	$1,00^{0,001}$		0,96

Erreichte Präzision von APClustering für Änderungstyp (AP1a) bei $m_c = 10$.

Netz		Pfade	*rel*	*par*	Präzision ($m_c = 4$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,28	0,03	$0,12^{0,003}$	$0,37^{0,037}$	0,97	0,97	0,97	✓	0,97
		2*1000	0,28	0,03	$0,63^{0,177}$	0,99	0,99	0,99	0,99	✓	0,99
	Variante 2	2*100	0,28	0,03	$0,16^{0,016}$	$0,24^{0,022}$	0,10	0,14	0,15		0,13
		2*1000	0,28	0,03	$0,08^{0,012}$	0,00	0,00	0,00	0,00		0,00
	Variante 3	2*100	0,31	0,03	$0,09^{0,002}$	$0,26^{0,004}$	$0,41^{0,005}$	0,46	0,46	✓	0,45
		2*1000	0,31	0,03	$0,67^{0,175}$	1,00	1,00	1,00	1,00	✓	1,00
herbstFig6p41*	Variante 1	2*100	0,43	0,16	0,07	$0,20^{0,013}$	$0,47^{0,004}$	0,49	0,49	✓	0,48
		2*1000	0,43	0,16	$0,42^{0,150}$	0,98	0,98	0,98	0,98	✓	0,98
	Variante 2	2*100	0,43	0,16	$0,08^{0,002}$	$0,15^{0,005}$	$0,31^{0,018}$	$0,22^{0,013}$	0,16		0,23
		2*1000	0,43	0,16	$0,20^{0,042}$	0,99	0,99	0,99	0,99	✓	0,99
	Variante 3	2*100	0,50	0,19	$0,10^{0,006}$	$0,11^{0,001}$	$0,29^{0,001}$	$0,45^{0,003}$	0,47	✓	0,40
		2*1000	0,50	0,19	$0,31^{0,044}$	$0,61^{0,043}$	$0,50^{0,002}$	0,50	0,50		0,50
herbstFig6p42	Variante 1	2*100	0,63	0,42	$0,08^{0,001}$	0,13	$0,06^{0,001}$	$0,07^{0,001}$	0,10		0,08
		2*1000	0,63	0,42	$0,08^{0,006}$	0,02	0,02	0,02	0,00		0,01
	Variante 2	2*100	0,63	0,42	0,07	$0,13^{0,002}$	$0,25^{0,001}$	$0,32^{0,004}$	0,42	✓	0,33
		2*1000	0,63	0,42	$0,12^{0,006}$	$0,30^{0,002}$	0,33	0,33	0,33	✓	0,33
	Variante 3	2*100	0,63	0,42	$0,08^{0,002}$	$0,15^{0,001}$	$0,34^{0,004}$	0,39	0,39	✓	0,37
		2*1000	0,63	0,42	$0,25^{0,086}$	$0,81^{0,059}$	1,00	1,00	1,00	✓	1,00
herbstFig6p18*	Variante 1	2*100	0,58	0,47	$0,10^{0,001}$	$0,22^{0,001}$	$0,20^{0,005}$	0,15	$0,19^{0,006}$		0,18
		2*1000	0,58	0,47	$0,12^{0,006}$	$0,13^{0,021}$	$0,59^{0,059}$	$0,25^{0,001}$	0,31		0,38
	Variante 2	2*100	0,58	0,47	$0,10^{0,001}$	$0,16^{0,001}$	0,14	0,14	0,14		0,14
		2*1000	0,58	0,47	$0,14^{0,032}$	0,03	0,03	0,03	0,02		0,03
	Variante 3	2*100	0,58	0,47	$0,18^{0,010}$	$0,22^{0,007}$	0,15	$0,17^{0,006}$	$0,20^{0,013}$		0,17
		2*1000	0,58	0,47	$0,06^{0,004}$	$0,14^{0,002}$	0,04	0,07	0,08		0,06
herbstFig6p34*	Variante 1	2*100	0,69	0,61	$0,14^{0,011}$	$0,35^{0,013}$	$0,22^{0,001}$	0,24	$0,28^{0,006}$		0,25
		2*1000	0,69	0,61	$0,09^{0,001}$	$0,20^{0,018}$	$0,34^{0,018}$	$0,29^{0,004}$	$0,35^{0,009}$		0,33
	Variante 2	2*100	0,69	0,61	$0,15^{0,008}$	$0,15^{0,003}$	$0,29^{0,006}$	$0,33^{0,004}$	$0,41^{0,005}$		0,34
		2*1000	0,69	0,61	$0,10^{0,002}$	$0,18^{0,002}$	$0,33^{0,026}$	$0,29^{0,048}$	$0,47^{0,092}$		0,36
	Variante 3	2*100	0,69	0,61	$0,11^{0,002}$	$0,16^{0,001}$	$0,22^{0,002}$	0,17	$0,18^{0,001}$		0,19
		2*1000	0,69	0,61	$0,15^{0,007}$	$0,24^{0,002}$	$0,38^{0,022}$	$0,96^{0,013}$	1,00	✓	0,78
al2*	Variante 1	2*100	0,83	0,67	$0,11^{0,003}$	$0,27^{0,007}$	$0,22^{0,009}$	0,15	0,15		0,17
		2*1000	0,83	0,67	$0,10^{0,011}$	0,02	$0,11^{0,015}$	$0,29^{0,005}$	0,25		0,22
	Variante 2	2*100	0,80	0,64	$0,11^{0,011}$	$0,03^{0,001}$	0,03	0,03	$0,04^{0,001}$		0,03
		2*1000	0,83	0,67	$0,13^{0,011}$	0,02	0,02	0,02	$0,10^{0,033}$		0,05
	Variante 3	2*100	0,80	0,64	$0,08^{0,001}$	0,13	$0,17^{0,001}$	$0,08^{0,001}$	0,10		0,12
		2*1000	0,83	0,67	$0,06^{0,001}$	0,03	0,03	$0,08^{0,007}$	$0,21^{0,001}$		0,11

Erreichte Präzision von APClustering für Änderungstyp (AP1b) bei $m_c = 4$.

Netz		Pfade	rel	par	Präzision ($m_c = 6$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,28	0,03	$0,16^{0,039}$	$0,31^{0,007}$	0,49	0,49	0,49	✓	0,49
		2*1000	0,28	0,03	$0,62^{0,181}$	0,99	0,99	0,99	0,99	✓	0,99
	Variante 2	2*100	0,28	0,03	$0,21^{0,046}$	$0,24^{0,022}$	0,10	0,15	0,15		0,13
		2*1000	0,28	0,03	$0,08^{0,013}$	0,00	0,00	0,00	0,00		0,00
	Variante 3	2*100	0,31	0,03	$0,15^{0,040}$	$0,28^{0,002}$	$0,41^{0,005}$	0,46	0,46	✓	0,45
		2*1000	0,31	0,03	$0,51^{0,129}$	1,00	1,00	1,00	1,00	✓	1,00
herbstFig6p41*	Variante 1	2*100	0,43	0,16	$0,14^{0,042}$	$0,17^{0,007}$	$0,33^{0,002}$	$0,47^{0,002}$	$0,42^{0,006}$		0,41
		2*1000	0,43	0,16	$0,40^{0,143}$	$0,94^{0,018}$	0,98	0,98	0,98	✓	0,98
	Variante 2	2*100	0,43	0,16	$0,14^{0,044}$	$0,14^{0,003}$	$0,35^{0,017}$	$0,27^{0,007}$	0,23		0,28
		2*1000	0,43	0,16	$0,20^{0,042}$	0,99	0,99	0,99	0,99	✓	0,99
	Variante 3	2*100	0,50	0,19	$0,13^{0,042}$	$0,12^{0,001}$	$0,39^{0,012}$	$0,84^{0,026}$	0,91		0,71
		2*1000	0,50	0,19	$0,31^{0,044}$	$0,61^{0,043}$	$0,50^{0,002}$	0,50	0,50		0,50
herbstFig6p42	Variante 1	2*100	0,63	0,42	$0,14^{0,040}$	0,14	$0,32^{0,003}$	0,08	0,10		0,17
		2*1000	0,63	0,42	$0,06^{0,003}$	0,01	0,02	0,02	0,00		0,01
	Variante 2	2*100	0,63	0,42	$0,14^{0,041}$	$0,15^{0,004}$	$0,33^{0,002}$	$0,32^{0,004}$	0,42		0,36
		2*1000	0,63	0,42	$0,13^{0,006}$	$0,30^{0,002}$	0,33	0,33	0,33	✓	0,33
	Variante 3	2*100	0,63	0,42	$0,13^{0,042}$	$0,14^{0,002}$	$0,34^{0,004}$	0,39	0,39	✓	0,37
		2*1000	0,63	0,42	$0,25^{0,086}$	$0,81^{0,059}$	1,00	1,00	1,00	✓	1,00
herbstFig6p18*	Variante 1	2*100	0,58	0,47	0,09	0,19	$0,21^{0,001}$	0,17	$0,21^{0,001}$		0,20
		2*1000	0,58	0,47	$0,10^{0,004}$	$0,15^{0,012}$	$0,59^{0,059}$	$0,25^{0,001}$	0,31		0,38
	Variante 2	2*100	0,58	0,47	$0,11^{0,001}$	$0,17^{0,001}$	0,14	0,14	0,14		0,14
		2*1000	0,58	0,47	$0,10^{0,020}$	0,03	0,03	0,03	0,02		0,03
	Variante 3	2*100	0,58	0,47	$0,11^{0,003}$	$0,24^{0,007}$	$0,21^{0,006}$	$0,20^{0,005}$	$0,22^{0,011}$		0,21
		2*1000	0,58	0,47	$0,09^{0,014}$	$0,13^{0,002}$	$0,05^{0,001}$	0,07	0,08		0,06
herbstFig6p34*	Variante 1	2*100	0,69	0,61	$0,16^{0,041}$	$0,31^{0,012}$	$0,31^{0,010}$	0,24	$0,29^{0,002}$		0,28
		2*1000	0,69	0,61	$0,08^{0,002}$	$0,22^{0,021}$	$0,37^{0,015}$	$0,41^{0,008}$	$0,51^{0,007}$	✓	0,43
	Variante 2	2*100	0,69	0,61	$0,13^{0,005}$	$0,15^{0,005}$	$0,29^{0,005}$	$0,32^{0,003}$	$0,41^{0,005}$	✓	0,34
		2*1000	0,69	0,61	$0,10^{0,003}$	$0,23^{0,002}$	$0,35^{0,011}$	$0,25^{0,017}$	$0,37^{0,032}$		0,33
	Variante 3	2*100	0,69	0,61	$0,11^{0,002}$	$0,16^{0,001}$	$0,23^{0,002}$	0,17	$0,18^{0,001}$		0,19
		2*1000	0,69	0,61	$0,11^{0,003}$	$0,37^{0,027}$	$0,52^{0,004}$	$0,96^{0,013}$	1,00	✓	0,83
al2*	Variante 1	2*100	0,83	0,67	$0,13^{0,006}$	$0,28^{0,008}$	$0,31^{0,003}$	$0,31^{0,001}$	0,25		0,29
		2*1000	0,83	0,67	$0,08^{0,007}$	0,02	$0,14^{0,022}$	$0,29^{0,003}$	0,25		0,23
	Variante 2	2*100	0,80	0,64	$0,19^{0,020}$	$0,06^{0,001}$	0,05	0,05	0,06		0,05
		2*1000	0,83	0,67	$0,13^{0,011}$	0,02	0,02	0,02	$0,10^{0,033}$		0,04
	Variante 3	2*100	0,80	0,64	0,08	$0,13^{0,001}$	$0,23^{0,001}$	$0,19^{0,008}$	0,10		0,18
		2*1000	0,83	0,67	$0,06^{0,001}$	0,03	0,03	$0,08^{0,007}$	$0,20^{0,001}$		0,10

Erreichte Präzision von APClustering für Änderungstyp (AP1b) bei $m_c = 6$.

Netz		Pfade	*rel*	*par*	Präzision ($m_c = 10$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,28	0,03	$0,25^{0,066}$	$0,38^{0,006}$	$0,43^{0,006}$	0,49	0,49	✓	0,47
		2*1000	0,28	0,03	$0,62^{0,181}$	0,99	0,99	0,99	0,99	✓	0,99
	Variante 2	2*100	0,28	0,03	$0,31^{0,097}$	$0,24^{0,022}$	0,10	0,15	0,15		0,13
		2*1000	0,28	0,03	$0,08^{0,007}$	0,00	0,00	0,00	0,00		0,00
	Variante 3	2*100	0,31	0,03	$0,23^{0,068}$	$0,27^{0,004}$	$0,41^{0,005}$	0,46	0,46	✓	0,45
		2*1000	0,31	0,03	$0,31^{0,028}$	0,50	0,50	0,50	0,50	✓	0,50
herbstFig6p41*	Variante 1	2*100	0,43	0,16	$0,30^{0,123}$	$0,18^{0,001}$	$0,39^{0,012}$	0,49	0,49		0,46
		2*1000	0,43	0,16	$0,37^{0,130}$	$0,94^{0,018}$	0,98	0,98	0,98	✓	0,98
	Variante 2	2*100	0,43	0,16	$0,35^{0,141}$	$0,16^{0,001}$	$0,35^{0,018}$	$0,27^{0,008}$	0,23		0,28
		2*1000	0,43	0,16	$0,20^{0,049}$	0,99	0,99	0,99	0,99	✓	0,99
	Variante 3	2*100	0,50	0,19	$0,31^{0,123}$	$0,15^{0,002}$	$0,40^{0,007}$	$0,84^{0,026}$	0,91		0,72
		2*1000	0,50	0,19	$0,32^{0,048}$	$0,61^{0,043}$	$0,50^{0,001}$	0,50	0,50		0,50
herbstFig6p42	Variante 1	2*100	0,63	0,42	$0,25^{0,105}$	$0,14^{0,001}$	$0,31^{0,004}$	$0,16^{0,017}$	0,07		0,18
		2*1000	0,63	0,42	$0,08^{0,012}$	0,02	0,02	0,02	0,00		0,01
	Variante 2	2*100	0,63	0,42	$0,38^{0,161}$	$0,16^{0,001}$	0,39	0,40	0,40		0,40
		2*1000	0,63	0,42	$0,16^{0,019}$	0,33	0,33	0,33	0,33	✓	0,33
	Variante 3	2*100	0,63	0,42	$0,40^{0,152}$	0,13	$0,23^{0,001}$	$0,31^{0,004}$	0,39		0,31
		2*1000	0,63	0,42	$0,19^{0,033}$	$0,80^{0,062}$	1,00	1,00	1,00	✓	1,00
herbstFig6p18*	Variante 1	2*100	0,58	0,47	$0,12^{0,003}$	$0,17^{0,001}$	$0,18^{0,001}$	0,20	$0,22^{0,001}$	✓	0,20
		2*1000	0,58	0,47	$0,09^{0,003}$	$0,14^{0,009}$	$0,59^{0,059}$	$0,25^{0,001}$	0,31		0,38
	Variante 2	2*100	0,58	0,47	0,13	$0,17^{0,001}$	0,14	0,14	0,14		0,14
		2*1000	0,58	0,47	$0,13^{0,017}$	0,03	0,03	0,03	0,03		0,03
	Variante 3	2*100	0,58	0,47	$0,12^{0,001}$	$0,24^{0,006}$	$0,21^{0,006}$	$0,20^{0,005}$	$0,26^{0,017}$		0,22
		2*1000	0,58	0,47	$0,09^{0,014}$	$0,13^{0,002}$	$0,05^{0,001}$	0,07	0,08		0,06
herbstFig6p34*	Variante 1	2*100	0,69	0,61	$0,20^{0,071}$	$0,21^{0,005}$	$0,65^{0,088}$	$0,45^{0,001}$	$0,76^{0,049}$		0,62
		2*1000	0,69	0,61	$0,07^{0,005}$	$0,20^{0,017}$	$0,36^{0,013}$	$0,41^{0,009}$	$0,52^{0,014}$	✓	0,43
	Variante 2	2*100	0,69	0,61	$0,24^{0,102}$	$0,17^{0,005}$	$0,21^{0,003}$	0,21	0,22		0,21
		2*1000	0,69	0,61	$0,09^{0,007}$	$0,22^{0,002}$	$0,35^{0,010}$	$0,25^{0,016}$	$0,39^{0,030}$		0,33
	Variante 3	2*100	0,69	0,61	$0,17^{0,039}$	$0,21^{0,003}$	$0,28^{0,001}$	0,31	$0,29^{0,001}$		0,29
		2*1000	0,69	0,61	$0,08^{0,002}$	$0,36^{0,031}$	$0,51^{0,006}$	$0,96^{0,013}$	1,00	✓	0,82
al2*	Variante 1	2*100	0,83	0,67	$0,18^{0,037}$	$0,21^{0,003}$	$0,29^{0,001}$	0,32	0,32		0,31
		2*1000	0,83	0,67	$0,09^{0,007}$	0,02	$0,13^{0,021}$	$0,29^{0,003}$	0,25		0,22
	Variante 2	2*100	0,80	0,64	$0,17^{0,012}$	$0,09^{0,006}$	0,07	0,07	$0,09^{0,002}$		0,08
		2*1000	0,83	0,67	$0,16^{0,018}$	0,01	0,01	$0,02^{0,001}$	$0,10^{0,033}$		0,04
	Variante 3	2*100	0,80	0,64	$0,16^{0,044}$	$0,12^{0,001}$	0,26	$0,19^{0,009}$	0,10		0,19
		2*1000	0,83	0,67	$0,05^{0,001}$	0,03	0,03	$0,08^{0,006}$	$0,19^{0,001}$		0,10

Erreichte Präzision von APClustering für Änderungstyp (AP1b) bei $m_c = 10$.

Netz		Pfade	*rel*	*par*	Präzision ($m_c = 4$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,28	0,03	$0,12^{0,009}$	$0,42^{0,008}$	$0,68^{0,057}$	0,98	$0,82^{0,074}$		0,83
		2*1000	0,28	0,03	$0,66^{0,128}$	$0,99^{0,001}$	1,00	1,00	1,00		1,00
	Variante 2	2*100	0,28	0,03	$0,12^{0,005}$	$0,33^{0,010}$	$0,52^{0,034}$	$0,51^{0,010}$	$0,76^{0,059}$		0,60
		2*1000	0,28	0,03	$0,86^{0,103}$	1,00	1,00	1,00	1,00	✓	1,00
	Variante 3	2*100	0,28	0,03	$0,12^{0,005}$	$0,36^{0,005}$	$0,59^{0,045}$	$0,90^{0,020}$	0,95	✓	0,81
		2*1000	0,28	0,03	$0,32^{0,032}$	0,50	$0,47^{0,004}$	0,50	0,50		0,49
herbstFig6p41*	Variante 1	2*100	0,43	0,16	$0,06^{0,001}$	$0,13^{0,001}$	$0,28^{0,006}$	$0,38^{0,005}$	0,32		0,33
		2*1000	0,43	0,16	$0,67^{0,175}$	1,00	1,00	1,00	1,00		1,00
	Variante 2	2*100	0,43	0,16	$0,10^{0,006}$	$0,22^{0,006}$	$0,57^{0,065}$	0,95	0,95	✓	0,82
		2*1000	0,43	0,16	$0,29^{0,115}$	0,99	$0,99^{0,002}$	0,99	$0,99^{0,002}$		0,99
	Variante 3	2*100	0,43	0,16	$0,08^{0,002}$	$0,14^{0,004}$	$0,56^{0,065}$	0,47	0,47		0,50
		2*1000	0,43	0,16	$0,35^{0,101}$	$0,26^{0,001}$	$0,32^{0,001}$	0,33	0,33		0,33
herbstFig6p42	Variante 1	2*100	0,63	0,42	$0,10^{0,002}$	$0,23^{0,008}$	0,32	$0,31^{0,008}$	$0,30^{0,005}$		0,31
		2*1000	0,63	0,42	$0,12^{0,027}$	$0,02^{0,003}$	$0,24^{0,032}$	$0,84^{0,065}$	0,98		0,69
	Variante 2	2*100	0,63	0,42	$0,08^{0,001}$	$0,16^{0,004}$	$0,21^{0,002}$	$0,42^{0,095}$	0,95	✓	0,53
		2*1000	0,63	0,42	$0,26^{0,031}$	$0,36^{0,004}$	$0,45^{0,006}$	$0,77^{0,060}$	$0,62^{0,045}$		0,61
	Variante 3	2*100	0,63	0,42	$0,14^{0,026}$	$0,19^{0,007}$	$0,46^{0,002}$	0,32	0,32		0,37
		2*1000	0,63	0,42	$0,23^{0,029}$	$0,70^{0,062}$	$0,75^{0,062}$	0,50	$0,56^{0,027}$		0,60
herbstFig6p18*	Variante 1	2*100	0,41	0,31	$0,22^{0,005}$	$0,15^{0,001}$	0,14	0,14	0,14		0,14
		2*1000	0,43	0,35	$0,16^{0,005}$	$0,27^{0,009}$	$0,46^{0,005}$	$0,49^{0,002}$	$0,36^{0,004}$		0,44
	Variante 2	2*100	0,56	0,42	$0,12^{0,002}$	$0,24^{0,002}$	0,27	$0,08^{0,006}$	$0,26^{0,005}$		0,20
		2*1000	0,58	0,47	$0,15^{0,006}$	$0,22^{0,001}$	0,03	0,05	0,05		0,04
	Variante 3	2*100	0,58	0,47	$0,15^{0,004}$	$0,09^{0,002}$	$0,11^{0,001}$	0,19	0,19		0,16
		2*1000	0,58	0,47	$0,06^{0,008}$	$0,16^{0,002}$	0,21	$0,12^{0,016}$	$0,06^{0,006}$		0,13
herbstFig6p34*	Variante 1	2*100	0,69	0,61	$0,13^{0,005}$	$0,16^{0,001}$	$0,31^{0,009}$	$0,66^{0,053}$	$0,95^{0,011}$	✓	0,64
		2*1000	0,69	0,61	$0,17^{0,012}$	$0,61^{0,045}$	$0,60^{0,095}$	$0,31^{0,001}$	$0,34^{0,002}$		0,42
	Variante 2	2*100	0,69	0,61	$0,20^{0,013}$	$0,17^{0,007}$	$0,34^{0,003}$	$0,42^{0,005}$	$0,46^{0,013}$		0,40
		2*1000	0,69	0,61	$0,09^{0,002}$	$0,21^{0,002}$	$0,32^{0,004}$	$0,38^{0,006}$	$0,99^{0,004}$	✓	0,56
	Variante 3	2*100	0,69	0,61	$0,15^{0,011}$	$0,22^{0,007}$	$0,44^{0,006}$	$0,44^{0,025}$	$0,54^{0,054}$		0,47
		2*1000	0,69	0,61	$0,07^{0,001}$	$0,17^{0,002}$	$0,31^{0,007}$	$0,44^{0,011}$	$0,42^{0,007}$		0,39
al2*	Variante 1	2*100	0,79	0,60	$0,10^{0,001}$	$0,19^{0,002}$	$0,20^{0,002}$	0,16	0,16		0,18
		2*1000	0,83	0,67	$0,05^{0,004}$	$0,16^{0,013}$	$0,21^{0,001}$	$0,30^{0,002}$	$0,39^{0,006}$	✓	0,30
	Variante 2	2*100	0,73	0,58	$0,11^{0,003}$	$0,28^{0,006}$	$0,25^{0,019}$	0,25	0,25		0,25
		2*1000	0,76	0,61	$0,16^{0,019}$	$0,19^{0,007}$	$0,21^{0,001}$	$0,28^{0,002}$	$0,26^{0,002}$		0,25
	Variante 3	2*100	0,80	0,64	$0,10^{0,001}$	$0,23^{0,009}$	$0,35^{0,008}$	$0,35^{0,008}$	$0,32^{0,004}$		0,34
		2*1000	0,83	0,67	$0,09^{0,008}$	0,04	0,04	0,04	0,04		0,04

Erreichte Präzision von APClustering für Änderungstyp (AP1c) bei $m_c = 4$.

Netz		Pfade	*rel*	*par*	Präzision ($m_c = 6$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,28	0,03	$0,17^{0,040}$	$0,42^{0,008}$	0,49	0,49	$0,61^{0,076}$	✓	0,53
		2*1000	0,28	0,03	$0,60^{0,120}$	1,00	1,00	1,00	1,00		1,00
	Variante 2	2*100	0,28	0,03	$0,14^{0,010}$	$0,30^{0,003}$	$0,53^{0,041}$	0,48	$0,89^{0,031}$		0,63
		2*1000	0,28	0,03	$0,81^{0,122}$	1,00	1,00	1,00	1,00	✓	1,00
	Variante 3	2*100	0,28	0,03	$0,13^{0,002}$	$0,33^{0,004}$	$0,49^{0,014}$	$0,91^{0,020}$	0,95	✓	0,78
		2*1000	0,28	0,03	$0,24^{0,017}$	$0,34^{0,001}$	0,33	0,33	0,33		0,33
herbstFig6p41*	Variante 1	2*100	0,43	0,16	$0,12^{0,041}$	$0,12^{0,001}$	$0,30^{0,007}$	0,47	0,47	✓	0,42
		2*1000	0,43	0,16	$0,65^{0,179}$	1,00	1,00	1,00	1,00		1,00
	Variante 2	2*100	0,43	0,16	$0,17^{0,049}$	$0,23^{0,004}$	$0,55^{0,071}$	0,95	0,95	✓	0,81
		2*1000	0,43	0,16	$0,29^{0,115}$	0,99	0,99	0,99	1,00	✓	0,99
	Variante 3	2*100	0,43	0,16	$0,11^{0,013}$	$0,15^{0,007}$	$0,45^{0,032}$	$0,53^{0,027}$	0,92	✓	0,63
		2*1000	0,43	0,16	$0,49^{0,168}$	$0,35^{0,003}$	$0,46^{0,004}$	0,50	0,50		0,49
herbstFig6p42	Variante 1	2*100	0,63	0,42	$0,14^{0,040}$	$0,32^{0,022}$	0,49	$0,32^{0,008}$	$0,29^{0,002}$		0,37
		2*1000	0,63	0,42	$0,12^{0,027}$	$0,02^{0,003}$	$0,24^{0,032}$	$0,84^{0,065}$	0,98		0,69
	Variante 2	2*100	0,63	0,42	$0,10^{0,004}$	$0,15^{0,005}$	$0,19^{0,001}$	$0,42^{0,095}$	0,95	✓	0,52
		2*1000	0,63	0,42	$0,25^{0,035}$	$0,49^{0,001}$	$0,79^{0,067}$	$0,82^{0,056}$	0,99	✓	0,86
	Variante 3	2*100	0,63	0,42	$0,10^{0,003}$	$0,17^{0,002}$	$0,31^{0,001}$	0,32	0,32	✓	0,31
		2*1000	0,63	0,42	$0,22^{0,030}$	$0,70^{0,062}$	$0,75^{0,062}$	0,50	$0,56^{0,027}$		0,60
herbstFig6p18*	Variante 1	2*100	0,41	0,31	$0,20^{0,006}$	$0,16^{0,003}$	0,14	0,14	$0,39^{0,021}$		0,22
		2*1000	0,43	0,35	$0,17^{0,007}$	$0,26^{0,009}$	$0,46^{0,005}$	$0,49^{0,002}$	$0,36^{0,004}$		0,44
	Variante 2	2*100	0,56	0,42	$0,11^{0,001}$	$0,21^{0,003}$	0,27	$0,07^{0,004}$	$0,27^{0,003}$		0,20
		2*1000	0,58	0,47	$0,13^{0,005}$	$0,22^{0,001}$	0,03	0,05	0,05		0,04
	Variante 3	2*100	0,58	0,47	$0,13^{0,003}$	$0,11^{0,002}$	$0,16^{0,001}$	$0,31^{0,002}$	0,31		0,26
		2*1000	0,58	0,47	$0,09^{0,009}$	$0,14^{0,004}$	0,21	$0,12^{0,016}$	$0,06^{0,007}$		0,13
herbstFig6p34*	Variante 1	2*100	0,69	0,61	$0,13^{0,006}$	$0,16^{0,001}$	$0,29^{0,008}$	$0,69^{0,056}$	$0,92^{0,011}$	✓	0,63
		2*1000	0,69	0,61	$0,14^{0,011}$	$0,57^{0,044}$	$0,46^{0,063}$	$0,31^{0,001}$	$0,34^{0,002}$		0,37
	Variante 2	2*100	0,69	0,61	$0,15^{0,006}$	$0,17^{0,007}$	$0,34^{0,003}$	$0,41^{0,005}$	$0,47^{0,007}$	✓	0,41
		2*1000	0,69	0,61	$0,08^{0,002}$	$0,22^{0,003}$	$0,32^{0,003}$	$0,38^{0,006}$	$0,99^{0,004}$	✓	0,56
	Variante 3	2*100	0,69	0,61	$0,14^{0,006}$	$0,21^{0,005}$	$0,41^{0,011}$	$0,44^{0,025}$	$0,54^{0,054}$	✓	0,46
		2*1000	0,69	0,61	$0,06^{0,001}$	$0,16^{0,002}$	$0,31^{0,007}$	$0,47^{0,009}$	$0,43^{0,019}$		0,40
al2*	Variante 1	2*100	0,79	0,60	$0,11^{0,001}$	$0,20^{0,002}$	$0,20^{0,002}$	0,16	0,16		0,18
		2*1000	0,83	0,67	$0,05^{0,004}$	$0,21^{0,023}$	$0,23^{0,001}$	$0,30^{0,002}$	$0,39^{0,006}$	✓	0,30
	Variante 2	2*100	0,73	0,58	$0,11^{0,003}$	$0,25^{0,004}$	$0,24^{0,006}$	0,24	0,24		0,24
		2*1000	0,76	0,61	$0,18^{0,023}$	$0,17^{0,006}$	0,18	$0,22^{0,001}$	$0,22^{0,001}$		0,21
	Variante 3	2*100	0,80	0,64	$0,09^{0,001}$	$0,25^{0,006}$	$0,33^{0,009}$	$0,27^{0,002}$	0,24		0,28
		2*1000	0,83	0,67	$0,07^{0,006}$	0,04	0,04	0,04	0,04		0,04

Erreichte Präzision von APClustering für Änderungstyp (AP1c) bei $m_c = 6$.

Netz		Pfade	rel	par	Präzision ($m_c = 10$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,28	0,03	$0,29^{0,090}$	$0,37^{0,018}$	0,47	0,47	$0,64^{0,051}$	✓	0,53
		2*1000	0,28	0,03	$0,51^{0,111}$	$0,99^{0,002}$	1,00	1,00	1,00		1,00
	Variante 2	2*100	0,28	0,03	$0,20^{0,072}$	$0,38^{0,012}$	$0,55^{0,037}$	0,93	$0,96^{0,001}$	✓	0,81
		2*1000	0,28	0,03	$0,81^{0,123}$	1,00	1,00	1,00	1,00	✓	1,00
	Variante 3	2*100	0,28	0,03	$0,25^{0,068}$	$0,28^{0,005}$	$0,40^{0,010}$	$0,60^{0,047}$	0,47		0,49
		2*1000	0,28	0,03	$0,25^{0,020}$	$0,34^{0,001}$	0,33	0,33	0,33		0,33
herbstFig6p41*	Variante 1	2*100	0,43	0,16	$0,28^{0,097}$	$0,16^{0,002}$	$0,43^{0,033}$	0,90	0,91		0,75
		2*1000	0,43	0,16	$0,50^{0,147}$	1,00	$0,99^{0,002}$	0,99	$0,85^{0,059}$		0,95
	Variante 2	2*100	0,43	0,16	$0,21^{0,041}$	$0,21^{0,005}$	$0,33^{0,008}$	$0,59^{0,087}$	0,95		0,62
		2*1000	0,43	0,16	$0,30^{0,121}$	0,99	0,99	0,99	1,00	✓	0,99
	Variante 3	2*100	0,43	0,16	$0,22^{0,078}$	$0,15^{0,002}$	$0,69^{0,056}$	0,91	0,92		0,84
		2*1000	0,43	0,16	$0,36^{0,086}$	$0,35^{0,003}$	$0,46^{0,004}$	0,50	0,50		0,49
herbstFig6p42	Variante 1	2*100	0,63	0,42	$0,30^{0,123}$	$0,32^{0,016}$	0,49	$0,36^{0,004}$	$0,45^{0,003}$		0,43
		2*1000	0,63	0,42	$0,17^{0,041}$	$0,33^{0,050}$	$0,41^{0,007}$	$0,84^{0,065}$	0,98	✓	0,74
	Variante 2	2*100	0,63	0,42	$0,25^{0,104}$	$0,16^{0,008}$	$0,26^{0,004}$	$0,35^{0,005}$	0,47		0,36
		2*1000	0,63	0,42	$0,25^{0,038}$	$0,49^{0,001}$	$0,78^{0,068}$	$0,82^{0,056}$	0,99	✓	0,86
	Variante 3	2*100	0,63	0,42	$0,21^{0,073}$	$0,12^{0,001}$	$0,23^{0,001}$	0,32	0,32		0,29
		2*1000	0,63	0,42	$0,17^{0,018}$	$0,32^{0,001}$	$0,42^{0,007}$	0,50	$0,50^{0,005}$		0,47
herbstFig6p18*	Variante 1	2*100	0,41	0,31	$0,19^{0,006}$	$0,17^{0,014}$	0,14	0,14	$0,39^{0,021}$		0,22
		2*1000	0,43	0,35	$0,16^{0,004}$	$0,28^{0,007}$	$0,46^{0,005}$	$0,49^{0,002}$	$0,36^{0,004}$		0,44
	Variante 2	2*100	0,56	0,42	$0,13^{0,003}$	$0,22^{0,002}$	0,27	$0,07^{0,004}$	$0,28^{0,003}$		0,20
		2*1000	0,58	0,47	$0,19^{0,009}$	$0,29^{0,001}$	0,03	0,04	0,04		0,04
	Variante 3	2*100	0,58	0,47	$0,13^{0,002}$	$0,15^{0,007}$	$0,29^{0,001}$	0,29	0,30	✓	0,29
		2*1000	0,58	0,47	$0,09^{0,004}$	$0,14^{0,001}$	0,21	$0,13^{0,016}$	$0,07^{0,007}$		0,13
herbstFig6p34*	Variante 1	2*100	0,69	0,61	$0,21^{0,070}$	$0,16^{0,001}$	$0,29^{0,001}$	$0,67^{0,080}$	$0,90^{0,020}$		0,62
		2*1000	0,69	0,61	$0,13^{0,013}$	$0,57^{0,047}$	$0,48^{0,070}$	$0,31^{0,001}$	$0,34^{0,002}$		0,38
	Variante 2	2*100	0,69	0,61	$0,20^{0,071}$	$0,17^{0,006}$	$0,33^{0,003}$	$0,41^{0,005}$	0,49		0,41
		2*1000	0,69	0,61	$0,08^{0,002}$	$0,21^{0,002}$	$0,32^{0,003}$	$0,45^{0,006}$	$0,99^{0,004}$	✓	0,59
	Variante 3	2*100	0,69	0,61	$0,17^{0,037}$	$0,20^{0,003}$	$0,39^{0,014}$	$0,44^{0,025}$	$0,56^{0,049}$	✓	0,46
		2*1000	0,69	0,61	$0,06^{0,005}$	$0,16^{0,001}$	$0,31^{0,008}$	$0,49^{0,005}$	$0,44^{0,018}$		0,41
al2*	Variante 1	2*100	0,79	0,60	$0,15^{0,004}$	$0,22^{0,001}$	0,27	$0,18^{0,002}$	0,16		0,21
		2*1000	0,83	0,67	$0,05^{0,004}$	$0,21^{0,023}$	$0,22^{0,001}$	$0,30^{0,002}$	$0,39^{0,006}$	✓	0,30
	Variante 2	2*100	0,73	0,58	$0,16^{0,040}$	0,12	$0,19^{0,002}$	$0,25^{0,001}$	$0,23^{0,003}$		0,22
		2*1000	0,76	0,61	$0,17^{0,024}$	$0,15^{0,005}$	$0,17^{0,001}$	$0,19^{0,001}$	$0,20^{0,001}$		0,19
	Variante 3	2*100	0,80	0,64	$0,12^{0,002}$	$0,22^{0,006}$	$0,33^{0,006}$	$0,27^{0,002}$	0,24		0,28
		2*1000	0,83	0,67	$0,08^{0,008}$	0,04	0,04	0,04	0,04		0,04

Erreichte Präzision von APClustering für Änderungstyp (AP1c) bei $m_c = 10$.

Netz		Pfade	*rel*	*par*	Präzision ($m_c = 4$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,34	0,03	$0,14^{0,006}$	0,25	$0,39^{0,014}$	$0,82^{0,055}$	$0,94^{0,022}$	✓	0,72
		2*1000	0,34	0,03	$0,38^{0,024}$	$0,76^{0,063}$	$0,74^{0,063}$	$0,60^{0,040}$	$0,94^{0,024}$		0,76
	Variante 2	2*100	0,34	0,03	$0,13^{0,003}$	$0,34^{0,006}$	$0,66^{0,109}$	0,99	0,99	✓	0,88
		2*1000	0,34	0,03	$0,42^{0,025}$	0,50	0,50	0,50	$0,51^{0,006}$	✓	0,50
	Variante 3	2*100	0,34	0,03	$0,14^{0,003}$	$0,41^{0,013}$	0,49	0,49	$0,62^{0,046}$	✓	0,53
		2*1000	0,34	0,03	$0,50^{0,064}$	$0,63^{0,048}$	$0,61^{0,042}$	0,50	$0,51^{0,004}$		0,54
herbstFig6p41*	Variante 1	2*100	0,57	0,21	$0,08^{0,001}$	$0,31^{0,022}$	$0,51^{0,046}$	0,99	1,00	✓	0,83
		2*1000	0,57	0,21	$0,25^{0,040}$	$0,39^{0,011}$	$0,44^{0,017}$	$0,47^{0,004}$	$0,77^{0,062}$	✓	0,56
	Variante 2	2*100	0,57	0,21	$0,07^{0,001}$	$0,26^{0,016}$	$0,40^{0,007}$	0,50	$0,64^{0,050}$	✓	0,51
		2*1000	0,57	0,21	$0,54^{0,190}$	$0,99^{0,002}$	1,00	$0,98^{0,007}$	$0,97^{0,012}$		0,99
	Variante 3	2*100	0,57	0,21	$0,09^{0,002}$	$0,41^{0,048}$	$0,30^{0,005}$	$0,70^{0,060}$	$0,92^{0,032}$		0,64
		2*1000	0,57	0,21	$0,40^{0,045}$	$0,54^{0,020}$	0,50	$0,54^{0,019}$	$0,99^{0,002}$		0,68
herbstFig6p42	Variante 1	2*100	0,80	0,54	$0,13^{0,007}$	$0,22^{0,005}$	$0,66^{0,032}$	0,78	$0,91^{0,004}$	✓	0,78
		2*1000	0,80	0,54	$0,54^{0,165}$	0,97	0,97	0,97	0,97	✓	0,97
	Variante 2	2*100	0,80	0,54	$0,11^{0,004}$	$0,31^{0,015}$	$0,25^{0,033}$	$0,41^{0,026}$	$0,38^{0,030}$		0,35
		2*1000	0,80	0,54	$0,23^{0,030}$	$0,30^{0,002}$	$0,25^{0,001}$	$0,27^{0,001}$	$0,27^{0,005}$		0,26
	Variante 3	2*100	0,80	0,54	$0,12^{0,004}$	$0,76^{0,117}$	1,00	0,99	0,99		1,00
		2*1000	0,80	0,54	$0,45^{0,075}$	1,00	1,00	1,00	$1,00^{0,001}$		1,00
herbstFig6p18*	Variante 1	2*100	0,84	0,68	$0,26^{0,010}$	$0,57^{0,034}$	1,00	1,00	1,00		1,00
		2*1000	0,84	0,68	$0,13^{0,003}$	$0,29^{0,009}$	$0,33^{0,001}$	$0,34^{0,002}$	$0,40^{0,025}$	✓	0,36
	Variante 2	2*100	0,84	0,68	$0,11^{0,001}$	$0,27^{0,003}$	$0,25^{0,002}$	$0,27^{0,002}$	$0,53^{0,086}$		0,35
		2*1000	0,84	0,68	$0,27^{0,040}$	$0,58^{0,069}$	$0,93^{0,029}$	$0,75^{0,062}$	$0,82^{0,057}$		0,83
	Variante 3	2*100	0,84	0,68	$0,18^{0,010}$	$0,66^{0,069}$	0,98	0,98	0,98	✓	0,98
		2*1000	0,84	0,68	$0,12^{0,008}$	$0,18^{0,001}$	$0,29^{0,002}$	$0,35^{0,011}$	$0,49^{0,018}$	✓	0,37
herbstFig6p34*	Variante 1	2*100	0,90	0,80	$0,15^{0,010}$	$0,25^{0,021}$	$0,65^{0,050}$	$0,60^{0,072}$	$0,72^{0,061}$		0,66
		2*1000	0,90	0,80	$0,09^{0,004}$	$0,16^{0,001}$	0,24	0,33	$0,46^{0,050}$	✓	0,35
	Variante 2	2*100	0,90	0,80	$0,16^{0,006}$	$0,16^{0,007}$	$0,30^{0,023}$	$0,39^{0,051}$	$0,32^{0,001}$		0,34
		2*1000	0,90	0,80	$0,08^{0,002}$	$0,19^{0,001}$	$0,29^{0,006}$	0,33	0,33		0,32
	Variante 3	2*100	0,90	0,80	$0,24^{0,021}$	$0,41^{0,009}$	$0,46^{0,020}$	$0,58^{0,057}$	$0,62^{0,075}$	✓	0,55
		2*1000	0,90	0,80	$0,80^{0,099}$	$0,99^{0,002}$	1,00	1,00	$0,99^{0,001}$		1,00
al2*	Variante 1	2*100	1,00	0,81	$0,15^{0,006}$	$0,14^{0,002}$	$0,20^{0,004}$	$0,31^{0,001}$	$0,56^{0,079}$		0,36
		2*1000	1,00	0,81	$0,34^{0,049}$	$0,66^{0,054}$	0,50	$0,81^{0,057}$	$0,64^{0,051}$		0,65
	Variante 2	2*100	1,00	0,81	$0,11^{0,002}$	$0,41^{0,014}$	0,50	0,50	0,50		0,50
		2*1000	1,00	0,81	$0,64^{0,176}$	1,00	1,00	1,00	1,00	✓	1,00
	Variante 3	2*100	1,00	0,81	$0,14^{0,005}$	$0,22^{0,002}$	$0,18^{0,018}$	$0,32^{0,047}$	$0,28^{0,049}$		0,26
		2*1000	1,00	0,81	$0,20^{0,013}$	$0,35^{0,004}$	0,50	0,50	0,50		0,50

Erreichte Präzision von APClustering für Änderungstyp (AP2) bei $m_c = 4$.

Netz		Pfade	rel	par	Präzision ($m_c = 6$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,34	0,03	$0,14^{0,008}$	0,25	$0,39^{0,014}$	$0,82^{0,054}$	$0,94^{0,022}$	✓	0,72
		2*1000	0,34	0,03	$0,31^{0,013}$	$0,83^{0,055}$	$0,75^{0,062}$	$0,60^{0,040}$	$0,94^{0,024}$		0,76
	Variante 2	2*100	0,34	0,03	$0,18^{0,039}$	$0,40^{0,011}$	$0,69^{0,096}$	0,99	0,99		0,89
		2*1000	0,34	0,03	$0,77^{0,132}$	1,00	1,00	1,00	1,00	✓	1,00
	Variante 3	2*100	0,34	0,03	$0,21^{0,037}$	$0,69^{0,092}$	0,97	0,97	0,98		0,97
		2*1000	0,34	0,03	$0,83^{0,115}$	1,00	1,00	1,00	1,00		1,00
herbstFig6p41*	Variante 1	2*100	0,57	0,21	$0,12^{0,041}$	$0,31^{0,020}$	$0,46^{0,058}$	0,99	1,00	✓	0,82
		2*1000	0,57	0,21	$0,24^{0,039}$	$0,39^{0,011}$	$0,43^{0,017}$	$0,47^{0,005}$	$0,72^{0,063}$	✓	0,54
	Variante 2	2*100	0,57	0,21	$0,12^{0,041}$	$0,26^{0,015}$	$0,40^{0,007}$	0,50	$0,69^{0,056}$	✓	0,53
		2*1000	0,57	0,21	$0,54^{0,191}$	$0,96^{0,018}$	1,00	$0,98^{0,008}$	$0,98^{0,010}$		0,99
	Variante 3	2*100	0,57	0,21	$0,14^{0,040}$	$0,25^{0,011}$	$0,32^{0,003}$	$0,68^{0,057}$	$0,88^{0,039}$	✓	0,63
		2*1000	0,57	0,21	$0,40^{0,047}$	$0,60^{0,040}$	0,50	$0,54^{0,019}$	$0,99^{0,002}$		0,68
herbstFig6p42	Variante 1	2*100	0,80	0,54	$0,14^{0,009}$	$0,18^{0,004}$	$0,66^{0,032}$	0,77	$0,91^{0,004}$	✓	0,78
		2*1000	0,80	0,54	$0,49^{0,147}$	0,97	0,97	0,97	0,97	✓	0,97
	Variante 2	2*100	0,80	0,54	$0,17^{0,044}$	$0,30^{0,016}$	0,46	$0,42^{0,016}$	$0,43^{0,006}$		0,44
		2*1000	0,80	0,54	$0,24^{0,033}$	$0,42^{0,007}$	$0,33^{0,002}$	$0,38^{0,005}$	$0,35^{0,003}$		0,35
	Variante 3	2*100	0,80	0,54	$0,14^{0,040}$	$0,74^{0,130}$	1,00	0,99	0,99		0,99
		2*1000	0,80	0,54	$0,45^{0,078}$	1,00	1,00	1,00	$0,99^{0,004}$		1,00
herbstFig6p18*	Variante 1	2*100	0,84	0,68	$0,24^{0,009}$	$0,57^{0,031}$	1,00	1,00	1,00		1,00
		2*1000	0,84	0,68	$0,12^{0,003}$	$0,28^{0,008}$	$0,33^{0,001}$	$0,34^{0,002}$	$0,41^{0,028}$	✓	0,36
	Variante 2	2*100	0,84	0,68	$0,10^{0,001}$	$0,19^{0,002}$	$0,22^{0,001}$	$0,25^{0,001}$	$0,54^{0,083}$	✓	0,34
		2*1000	0,84	0,68	$0,24^{0,035}$	$0,56^{0,069}$	$0,93^{0,030}$	$0,68^{0,058}$	$0,79^{0,061}$		0,80
	Variante 3	2*100	0,84	0,68	$0,17^{0,006}$	$0,67^{0,065}$	0,98	0,98	0,97		0,97
		2*1000	0,84	0,68	$0,10^{0,003}$	$0,18^{0,001}$	$0,28^{0,002}$	$0,35^{0,011}$	$0,49^{0,020}$	✓	0,37
herbstFig6p34*	Variante 1	2*100	0,90	0,80	$0,17^{0,039}$	$0,22^{0,018}$	$0,65^{0,050}$	$0,61^{0,068}$	$0,71^{0,066}$		0,66
		2*1000	0,90	0,80	$0,08^{0,004}$	$0,16^{0,001}$	0,24	0,33	$0,53^{0,040}$	✓	0,37
	Variante 2	2*100	0,90	0,80	$0,17^{0,039}$	$0,15^{0,002}$	$0,29^{0,023}$	$0,41^{0,050}$	$0,32^{0,002}$		0,34
		2*1000	0,90	0,80	$0,07^{0,002}$	$0,19^{0,001}$	$0,29^{0,006}$	0,33	0,33		0,32
	Variante 3	2*100	0,90	0,80	$0,23^{0,017}$	$0,38^{0,009}$	$0,44^{0,042}$	$0,53^{0,056}$	$0,69^{0,073}$	✓	0,55
		2*1000	0,90	0,80	$0,76^{0,120}$	$0,99^{0,002}$	1,00	1,00	$0,99^{0,001}$		1,00
al2*	Variante 1	2*100	1,00	0,81	$0,14^{0,003}$	$0,16^{0,002}$	$0,25^{0,001}$	0,33	$0,42^{0,040}$	✓	0,33
		2*1000	1,00	0,81	$0,32^{0,047}$	$0,66^{0,054}$	0,50	$0,81^{0,057}$	$0,65^{0,052}$		0,65
	Variante 2	2*100	1,00	0,81	$0,10^{0,001}$	$0,40^{0,015}$	$0,46^{0,004}$	0,50	0,50		0,48
		2*1000	1,00	0,81	$0,57^{0,170}$	1,00	1,00	1,00	1,00		1,00
	Variante 3	2*100	1,00	0,81	$0,13^{0,004}$	$0,22^{0,001}$	$0,18^{0,018}$	$0,32^{0,047}$	$0,27^{0,046}$		0,26
		2*1000	1,00	0,81	$0,20^{0,013}$	$0,35^{0,004}$	0,50	0,50	0,50		0,50

Erreichte Präzision von APClustering für Änderungstyp (AP2) bei $m_c = 6$.

Netz		Pfade	*rel*	*par*	Präzision ($m_c = 10$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,34	0,03	$0,24^{0,070}$	0,25	$0,39^{0,014}$	$0,75^{0,052}$	$0,86^{0,046}$	✓	0,66
		2*1000	0,34	0,03	$0,31^{0,015}$	$0,83^{0,055}$	$0,75^{0,062}$	$0,60^{0,040}$	$0,94^{0,024}$		0,76
	Variante 2	2*100	0,34	0,03	$0,23^{0,068}$	$0,40^{0,010}$	$0,69^{0,096}$	0,99	0,99		0,89
		2*1000	0,34	0,03	$0,78^{0,133}$	1,00	1,00	1,00	1,00		1,00
	Variante 3	2*100	0,34	0,03	$0,27^{0,063}$	$0,71^{0,084}$	0,99	0,99	0,98		0,98
		2*1000	0,34	0,03	$0,83^{0,114}$	1,00	1,00	1,00	1,00		1,00
herbstFig6p41*	Variante 1	2*100	0,57	0,21	$0,21^{0,075}$	$0,21^{0,006}$	$0,40^{0,069}$	0,99	0,99		0,80
		2*1000	0,57	0,21	$0,26^{0,045}$	$0,39^{0,011}$	$0,45^{0,011}$	$0,38^{0,006}$	$0,60^{0,077}$		0,48
	Variante 2	2*100	0,57	0,21	$0,29^{0,128}$	$0,33^{0,021}$	$0,68^{0,055}$	0,95	0,96	✓	0,86
		2*1000	0,57	0,21	$0,55^{0,192}$	$0,96^{0,018}$	1,00	$0,98^{0,008}$	$0,96^{0,018}$		0,98
	Variante 3	2*100	0,57	0,21	$0,31^{0,120}$	$0,21^{0,008}$	$0,26^{0,005}$	$0,68^{0,057}$	$0,89^{0,040}$		0,61
		2*1000	0,57	0,21	$0,41^{0,053}$	$0,61^{0,043}$	0,50	$0,54^{0,018}$	$1,00^{0,001}$		0,68
herbstFig6p42	Variante 1	2*100	0,80	0,54	$0,33^{0,116}$	$0,19^{0,006}$	$0,66^{0,036}$	0,77	$0,92^{0,004}$		0,78
		2*1000	0,80	0,54	$0,45^{0,139}$	0,97	0,97	0,97	0,97	✓	0,97
	Variante 2	2*100	0,80	0,54	$0,30^{0,124}$	$0,23^{0,007}$	$0,33^{0,005}$	$0,39^{0,008}$	$0,42^{0,009}$		0,38
		2*1000	0,80	0,54	$0,23^{0,042}$	$0,40^{0,007}$	$0,34^{0,001}$	$0,36^{0,004}$	$0,35^{0,008}$		0,35
	Variante 3	2*100	0,80	0,54	$0,34^{0,144}$	$0,73^{0,128}$	0,99	0,99	0,98		0,99
		2*1000	0,80	0,54	$0,45^{0,081}$	1,00	1,00	1,00	$0,99^{0,005}$		0,99
herbstFig6p18*	Variante 1	2*100	0,84	0,68	$0,24^{0,010}$	$0,57^{0,031}$	$0,96^{0,013}$	1,00	0,99		0,98
		2*1000	0,84	0,68	$0,11^{0,003}$	$0,31^{0,013}$	$0,45^{0,005}$	$0,44^{0,006}$	$0,54^{0,019}$		0,48
	Variante 2	2*100	0,84	0,68	$0,11^{0,001}$	$0,18^{0,001}$	$0,29^{0,003}$	$0,32^{0,003}$	$0,62^{0,054}$	✓	0,41
		2*1000	0,84	0,68	$0,22^{0,030}$	$0,50^{0,051}$	$0,90^{0,038}$	$0,61^{0,043}$	$0,77^{0,061}$		0,76
	Variante 3	2*100	0,84	0,68	$0,17^{0,007}$	$0,68^{0,064}$	0,98	0,98	0,96		0,97
		2*1000	0,84	0,68	$0,09^{0,002}$	$0,18^{0,001}$	$0,28^{0,002}$	$0,35^{0,011}$	$0,50^{0,025}$	✓	0,38
herbstFig6p34*	Variante 1	2*100	0,90	0,80	$0,21^{0,073}$	$0,23^{0,014}$	$0,67^{0,054}$	$0,72^{0,064}$	$0,92^{0,023}$	✓	0,77
		2*1000	0,90	0,80	$0,07^{0,001}$	$0,16^{0,001}$	0,24	0,33	$0,55^{0,043}$	✓	0,38
	Variante 2	2*100	0,90	0,80	$0,21^{0,071}$	$0,16^{0,001}$	$0,29^{0,023}$	$0,44^{0,043}$	0,44		0,39
		2*1000	0,90	0,80	$0,07^{0,005}$	$0,18^{0,001}$	$0,36^{0,017}$	0,50	0,50		0,45
	Variante 3	2*100	0,90	0,80	$0,22^{0,041}$	$0,44^{0,038}$	$0,56^{0,123}$	$0,73^{0,056}$	$0,95^{0,001}$	✓	0,75
		2*1000	0,90	0,80	$0,73^{0,135}$	$0,98^{0,008}$	1,00	1,00	$0,99^{0,001}$		1,00
al2*	Variante 1	2*100	1,00	0,81	$0,18^{0,038}$	$0,18^{0,001}$	$0,45^{0,009}$	$0,41^{0,007}$	$0,54^{0,022}$		0,46
		2*1000	1,00	0,81	$0,33^{0,054}$	$0,66^{0,054}$	0,50	$0,72^{0,061}$	$0,66^{0,054}$		0,62
	Variante 2	2*100	1,00	0,81	$0,22^{0,075}$	$0,26^{0,003}$	$0,30^{0,001}$	$0,46^{0,004}$	0,49	✓	0,42
		2*1000	1,00	0,81	$0,54^{0,186}$	1,00	1,00	1,00	1,00		1,00
	Variante 3	2*100	1,00	0,81	$0,16^{0,038}$	$0,24^{0,005}$	$0,24^{0,026}$	0,48	$0,45^{0,011}$		0,39
		2*1000	1,00	0,81	$0,19^{0,013}$	$0,34^{0,004}$	0,50	0,50	0,50		0,50

Erreichte Präzision von APClustering für Änderungstyp (AP2) bei $m_c = 10$.

Netz		Pfade	rel	par	Präzision ($m_c = 4$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,34	0,03	$0,13^{0,005}$	$0,42^{0,025}$	$0,62^{0,044}$	$0,82^{0,045}$	0,50		0,65
		2*1000	0,34	0,03	$0,74^{0,115}$	$0,89^{0,011}$	$0,80^{0,030}$	1,00	1,00		0,93
	Variante 2	2*100	0,34	0,03	$0,11^{0,002}$	$0,28^{0,005}$	$0,40^{0,007}$	0,50	$0,62^{0,045}$	✓	0,51
		2*1000	0,34	0,03	$0,20^{0,049}$	0,50	0,50	0,50	$0,51^{0,005}$	✓	0,50
	Variante 3	2*100	0,34	0,03	$0,27^{0,051}$	$0,78^{0,087}$	0,99	$0,94^{0,022}$	$0,94^{0,022}$		0,96
		2*1000	0,34	0,03	$0,39^{0,024}$	$0,84^{0,054}$	1,00	$0,79^{0,061}$	$0,54^{0,017}$		0,77
herbstFig6p41*	Variante 1	2*100	0,57	0,21	$0,09^{0,004}$	$0,29^{0,021}$	$0,61^{0,046}$	0,47	$0,61^{0,047}$		0,56
		2*1000	0,57	0,21	$0,46^{0,180}$	$0,82^{0,065}$	$0,62^{0,020}$	$0,93^{0,026}$	$0,75^{0,061}$		0,77
	Variante 2	2*100	0,57	0,21	$0,12^{0,008}$	$0,37^{0,042}$	$0,58^{0,032}$	$0,83^{0,006}$	$0,84^{0,024}$	✓	0,75
		2*1000	0,57	0,21	$0,44^{0,138}$	$0,84^{0,054}$	$0,85^{0,053}$	$0,86^{0,057}$	$0,83^{0,062}$		0,84
	Variante 3	2*100	0,49	0,18	0,07	$0,20^{0,005}$	$0,25^{0,003}$	$0,35^{0,004}$	$0,40^{0,007}$	✓	0,34
		2*1000	0,49	0,18	$0,35^{0,039}$	0,50	0,50	0,50	$0,51^{0,005}$	✓	0,50
herbstFig6p42	Variante 1	2*100	0,77	0,50	$0,10^{0,002}$	$0,21^{0,002}$	$0,30^{0,002}$	0,33	$0,36^{0,021}$	✓	0,33
		2*1000	0,80	0,54	$0,47^{0,150}$	1,00	1,00	1,00	1,00		1,00
	Variante 2	2*100	0,80	0,54	$0,10^{0,002}$	$0,14^{0,001}$	$0,17^{0,001}$	0,16	0,17		0,17
		2*1000	0,80	0,54	$0,19^{0,027}$	$0,32^{0,013}$	$0,37^{0,001}$	$0,37^{0,013}$	$0,42^{0,008}$	✓	0,39
	Variante 3	2*100	0,80	0,54	$0,10^{0,002}$	0,21	$0,25^{0,002}$	$0,22^{0,040}$	$0,43^{0,010}$		0,30
		2*1000	0,80	0,54	$0,17^{0,029}$	$0,12^{0,035}$	$0,13^{0,037}$	$0,35^{0,054}$	$0,99^{0,001}$		0,49
herbstFig6p18*	Variante 1	2*100	0,84	0,68	$0,18^{0,012}$	$0,18^{0,002}$	0,24	$0,30^{0,002}$	$0,64^{0,080}$	✓	0,39
		2*1000	0,84	0,68	$0,57^{0,125}$	1,00	1,00	1,00	1,00		1,00
	Variante 2	2*100	0,84	0,68	$0,12^{0,005}$	$0,27^{0,015}$	0,11	$0,14^{0,001}$	0,16		0,14
		2*1000	0,84	0,68	$0,20^{0,011}$	$0,26^{0,018}$	$0,74^{0,069}$	1,00	1,00	✓	0,91
	Variante 3	2*100	0,43	0,35	$0,24^{0,027}$	$0,20^{0,002}$	$0,40^{0,007}$	0,50	0,49		0,46
		2*1000	0,43	0,35	$0,18^{0,009}$	$0,20^{0,001}$	$0,36^{0,012}$	0,50	0,50	✓	0,45
herbstFig6p34*	Variante 1	2*100	0,90	0,80	$0,14^{0,004}$	$0,27^{0,009}$	$0,29^{0,005}$	$0,48^{0,032}$	$0,63^{0,075}$	✓	0,47
		2*1000	0,90	0,80	$0,11^{0,004}$	$0,26^{0,009}$	$0,48^{0,011}$	$0,47^{0,098}$	$0,60^{0,046}$		0,52
	Variante 2	2*100	0,90	0,80	$0,16^{0,008}$	$0,22^{0,002}$	0,32	$0,35^{0,001}$	$0,44^{0,003}$	✓	0,37
		2*1000	0,90	0,80	$0,11^{0,002}$	$0,26^{0,006}$	$0,44^{0,006}$	$0,63^{0,035}$	$0,92^{0,001}$	✓	0,66
	Variante 3	2*100	0,90	0,80	$0,23^{0,042}$	$0,18^{0,008}$	$0,24^{0,007}$	$0,25^{0,008}$	$0,34^{0,009}$		0,28
		2*1000	0,90	0,80	$0,12^{0,004}$	$0,25^{0,004}$	$0,61^{0,015}$	$0,42^{0,035}$	$0,54^{0,159}$		0,52
al2*	Variante 1	2*100	0,97	0,77	$0,11^{0,003}$	$0,34^{0,025}$	$0,18^{0,002}$	0,23	$0,22^{0,001}$		0,21
		2*1000	1,00	0,81	$0,17^{0,009}$	$0,05^{0,004}$	0,03	$0,20^{0,022}$	$0,46^{0,005}$		0,23
	Variante 2	2*100	0,97	0,77	$0,11^{0,001}$	$0,16^{0,002}$	0,17	0,17	0,17	✓	0,17
		2*1000	1,00	0,81	$0,15^{0,023}$	$0,14^{0,005}$	$0,42^{0,007}$	$0,40^{0,006}$	$0,41^{0,006}$		0,41
	Variante 3	2*100	0,91	0,74	$0,12^{0,004}$	$0,23^{0,002}$	$0,17^{0,001}$	$0,31^{0,009}$	0,40		0,29
		2*1000	0,91	0,74	$0,11^{0,004}$	0,06	0,06	0,06	0,06		0,06

Erreichte Präzision von APClustering für Änderungstyp (AP3a) bei $m_c = 4$.

Netz		Pfade	*rel*	*par*	Präzision ($m_c = 6$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,34	0,03	$0,15^{0,006}$	$0,45^{0,021}$	$0,64^{0,049}$	$0,87^{0,035}$	0,50		0,67
		2*1000	0,34	0,03	$0,74^{0,114}$	$0,89^{0,011}$	$0,80^{0,030}$	1,00	1,00		0,93
	Variante 2	2*100	0,34	0,03	$0,11^{0,002}$	$0,33^{0,012}$	$0,69^{0,062}$	0,99	0,99	✓	0,89
		2*1000	0,34	0,03	$0,51^{0,095}$	$0,78^{0,061}$	$0,94^{0,027}$	1,00	1,00		0,98
	Variante 3	2*100	0,34	0,03	$0,22^{0,019}$	$0,57^{0,064}$	0,49	$0,84^{0,051}$	$0,94^{0,022}$		0,76
		2*1000	0,34	0,03	$0,39^{0,025}$	$0,84^{0,054}$	1,00	$0,79^{0,061}$	$0,54^{0,017}$		0,77
herbstFig6p41*	Variante 1	2*100	0,57	0,21	$0,18^{0,075}$	$0,28^{0,022}$	$0,61^{0,046}$	0,47	$0,52^{0,021}$		0,53
		2*1000	0,57	0,21	$0,46^{0,181}$	$0,82^{0,062}$	$0,60^{0,022}$	$0,98^{0,006}$	$0,91^{0,035}$		0,83
	Variante 2	2*100	0,57	0,21	$0,13^{0,011}$	$0,28^{0,018}$	$0,61^{0,030}$	$0,83^{0,006}$	$0,87^{0,018}$	✓	0,77
		2*1000	0,57	0,21	$0,43^{0,138}$	$0,84^{0,052}$	$0,84^{0,053}$	$0,86^{0,053}$	$0,83^{0,059}$		0,85
	Variante 3	2*100	0,49	0,18	$0,13^{0,041}$	$0,22^{0,006}$	$0,23^{0,001}$	$0,27^{0,001}$	$0,34^{0,009}$	✓	0,28
		2*1000	0,49	0,18	$0,35^{0,039}$	0,50	0,50	0,50	$0,51^{0,006}$	✓	0,50
herbstFig6p42	Variante 1	2*100	0,77	0,50	$0,12^{0,006}$	$0,22^{0,001}$	$0,30^{0,002}$	0,33	$0,36^{0,021}$	✓	0,33
		2*1000	0,80	0,54	$0,47^{0,152}$	1,00	1,00	1,00	1,00		1,00
	Variante 2	2*100	0,80	0,54	$0,11^{0,005}$	0,13	0,19	0,19	$0,21^{0,001}$	✓	0,19
		2*1000	0,80	0,54	$0,22^{0,031}$	$0,32^{0,013}$	$0,37^{0,001}$	$0,37^{0,013}$	$0,42^{0,008}$	✓	0,39
	Variante 3	2*100	0,80	0,54	$0,15^{0,040}$	0,20	$0,25^{0,004}$	$0,21^{0,037}$	$0,44^{0,010}$		0,30
		2*1000	0,80	0,54	$0,14^{0,019}$	$0,12^{0,035}$	$0,13^{0,037}$	$0,35^{0,054}$	$0,99^{0,001}$		0,49
herbstFig6p18*	Variante 1	2*100	0,84	0,68	$0,19^{0,015}$	$0,21^{0,002}$	$0,23^{0,001}$	$0,32^{0,001}$	$0,72^{0,068}$	✓	0,42
		2*1000	0,84	0,68	$0,52^{0,122}$	1,00	1,00	1,00	1,00		1,00
	Variante 2	2*100	0,84	0,68	$0,12^{0,006}$	$0,28^{0,013}$	0,11	$0,14^{0,001}$	0,16		0,14
		2*1000	0,84	0,68	$0,18^{0,012}$	$0,26^{0,017}$	$0,73^{0,074}$	1,00	1,00	✓	0,91
	Variante 3	2*100	0,43	0,35	$0,17^{0,009}$	$0,19^{0,003}$	$0,40^{0,007}$	0,50	0,49		0,46
		2*1000	0,43	0,35	$0,17^{0,008}$	$0,20^{0,001}$	$0,36^{0,012}$	0,50	0,50	✓	0,45
herbstFig6p34*	Variante 1	2*100	0,90	0,80	$0,16^{0,040}$	$0,20^{0,001}$	$0,29^{0,006}$	$0,48^{0,032}$	$0,63^{0,075}$	✓	0,46
		2*1000	0,90	0,80	$0,09^{0,003}$	$0,25^{0,009}$	$0,47^{0,009}$	$0,47^{0,096}$	$0,61^{0,048}$		0,52
	Variante 2	2*100	0,90	0,80	$0,15^{0,005}$	$0,23^{0,001}$	0,31	0,31	$0,42^{0,037}$		0,35
		2*1000	0,90	0,80	$0,11^{0,002}$	$0,26^{0,005}$	$0,43^{0,006}$	$0,63^{0,035}$	$0,92^{0,001}$	✓	0,66
	Variante 3	2*100	0,90	0,80	$0,25^{0,065}$	$0,22^{0,007}$	$0,30^{0,005}$	$0,26^{0,008}$	$0,44^{0,040}$		0,33
		2*1000	0,90	0,80	$0,11^{0,003}$	$0,25^{0,004}$	$0,60^{0,015}$	$0,41^{0,033}$	$0,54^{0,159}$		0,52
al2*	Variante 1	2*100	0,97	0,77	$0,11^{0,001}$	$0,23^{0,003}$	$0,19^{0,001}$	0,23	$0,22^{0,001}$		0,22
		2*1000	1,00	0,81	$0,16^{0,008}$	$0,05^{0,004}$	$0,28^{0,054}$	$0,40^{0,008}$	$0,46^{0,005}$		0,38
	Variante 2	2*100	0,97	0,77	$0,10^{0,001}$	$0,16^{0,001}$	$0,18^{0,001}$	0,17	$0,19^{0,005}$		0,18
		2*1000	1,00	0,81	$0,15^{0,017}$	$0,14^{0,005}$	$0,42^{0,007}$	$0,39^{0,006}$	$0,41^{0,006}$		0,41
	Variante 3	2*100	0,91	0,74	$0,11^{0,002}$	$0,19^{0,001}$	$0,18^{0,001}$	$0,26^{0,004}$	0,32		0,25
		2*1000	0,91	0,74	$0,12^{0,005}$	$0,07^{0,003}$	$0,07^{0,001}$	0,06	0,06		0,06

Erreichte Präzision von APClustering für Änderungstyp (AP3a) bei $m_c = 6$.

Netz		Pfade	rel	par	Präzision ($m_c = 10$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,34	0,03	$0,26^{0,063}$	$0,45^{0,021}$	$0,64^{0,049}$	$0,87^{0,036}$	0,50		0,67
		2*1000	0,34	0,03	$0,75^{0,112}$	$0,89^{0,011}$	$0,80^{0,030}$	1,00	1,00		0,93
	Variante 2	2*100	0,34	0,03	$0,24^{0,069}$	$0,42^{0,013}$	$0,69^{0,062}$	1,00	0,99		0,89
		2*1000	0,34	0,03	$0,46^{0,101}$	$0,79^{0,061}$	$0,95^{0,023}$	1,00	1,00		0,98
	Variante 3	2*100	0,34	0,03	$0,33^{0,088}$	$0,79^{0,059}$	0,99	0,99	0,98		0,99
		2*1000	0,34	0,03	$0,46^{0,044}$	$0,99^{0,004}$	1,00	$0,97^{0,015}$	$0,76^{0,063}$		0,91
herbstFig6p41*	Variante 1	2*100	0,57	0,21	$0,26^{0,103}$	$0,18^{0,002}$	$0,25^{0,001}$	$0,45^{0,003}$	$0,60^{0,049}$		0,43
		2*1000	0,57	0,21	$0,46^{0,179}$	$0,83^{0,055}$	$0,59^{0,023}$	$0,98^{0,004}$	$0,97^{0,011}$		0,85
	Variante 2	2*100	0,57	0,21	$0,24^{0,069}$	$0,26^{0,012}$	$0,51^{0,029}$	$0,83^{0,006}$	$0,87^{0,021}$	✓	0,74
		2*1000	0,57	0,21	$0,44^{0,137}$	$0,87^{0,032}$	$0,91^{0,022}$	$0,93^{0,023}$	$0,92^{0,029}$		0,92
	Variante 3	2*100	0,49	0,18	$0,26^{0,100}$	$0,16^{0,002}$	$0,27^{0,010}$	0,47	0,49		0,41
		2*1000	0,49	0,18	$0,61^{0,178}$	1,00	1,00	1,00	1,00		1,00
herbstFig6p42	Variante 1	2*100	0,77	0,50	$0,26^{0,098}$	$0,24^{0,003}$	0,33	0,33	$0,36^{0,021}$		0,34
		2*1000	0,80	0,54	$0,46^{0,156}$	1,00	1,00	1,00	1,00	✓	1,00
	Variante 2	2*100	0,80	0,54	$0,24^{0,102}$	$0,15^{0,001}$	$0,24^{0,002}$	$0,20^{0,001}$	0,24		0,23
		2*1000	0,80	0,54	$0,24^{0,029}$	$0,36^{0,004}$	0,33	$0,35^{0,002}$	$0,42^{0,009}$		0,37
	Variante 3	2*100	0,80	0,54	$0,22^{0,071}$	$0,20^{0,001}$	$0,27^{0,009}$	$0,40^{0,006}$	$0,45^{0,009}$		0,37
		2*1000	0,80	0,54	$0,18^{0,017}$	$0,12^{0,035}$	$0,13^{0,037}$	$0,35^{0,054}$	$0,99^{0,001}$		0,49
herbstFig6p18*	Variante 1	2*100	0,84	0,68	$0,17^{0,003}$	$0,24^{0,003}$	$0,22^{0,001}$	$0,43^{0,018}$	$0,95^{0,001}$		0,53
		2*1000	0,84	0,68	$0,46^{0,101}$	1,00	1,00	1,00	1,00		1,00
	Variante 2	2*100	0,84	0,68	$0,14^{0,002}$	$0,27^{0,012}$	0,12	$0,14^{0,001}$	0,16		0,14
		2*1000	0,84	0,68	$0,17^{0,012}$	$0,25^{0,014}$	$0,73^{0,075}$	1,00	1,00	✓	0,91
	Variante 3	2*100	0,43	0,35	$0,15^{0,004}$	$0,20^{0,002}$	$0,40^{0,007}$	0,50	0,49		0,46
		2*1000	0,43	0,35	$0,15^{0,007}$	$0,20^{0,001}$	$0,36^{0,012}$	0,50	0,50	✓	0,45
herbstFig6p34*	Variante 1	2*100	0,90	0,80	$0,21^{0,069}$	$0,18^{0,001}$	$0,28^{0,005}$	$0,48^{0,032}$	$0,75^{0,086}$		0,50
		2*1000	0,90	0,80	$0,09^{0,007}$	$0,26^{0,011}$	$0,47^{0,006}$	$0,48^{0,094}$	$0,61^{0,049}$	✓	0,52
	Variante 2	2*100	0,90	0,80	$0,20^{0,036}$	$0,23^{0,001}$	0,31	0,31	$0,43^{0,037}$	✓	0,35
		2*1000	0,90	0,80	$0,11^{0,007}$	$0,25^{0,003}$	$0,42^{0,007}$	$0,63^{0,035}$	$0,92^{0,001}$	✓	0,65
	Variante 3	2*100	0,90	0,80	$0,20^{0,040}$	$0,24^{0,009}$	$0,30^{0,005}$	$0,33^{0,009}$	$0,50^{0,032}$	✓	0,38
		2*1000	0,90	0,80	$0,10^{0,008}$	$0,24^{0,005}$	$0,60^{0,016}$	$0,40^{0,031}$	$0,54^{0,157}$		0,51
al2*	Variante 1	2*100	0,97	0,77	$0,13^{0,002}$	$0,20^{0,006}$	$0,21^{0,001}$	$0,23^{0,001}$	$0,23^{0,001}$		0,23
		2*1000	1,00	0,81	$0,21^{0,023}$	$0,17^{0,001}$	$0,32^{0,039}$	$0,40^{0,008}$	$0,46^{0,005}$		0,39
	Variante 2	2*100	0,97	0,77	$0,16^{0,038}$	$0,15^{0,002}$	$0,24^{0,021}$	0,13	$0,22^{0,025}$		0,19
		2*1000	1,00	0,81	$0,16^{0,014}$	$0,15^{0,005}$	$0,42^{0,006}$	$0,36^{0,004}$	$0,40^{0,006}$		0,39
	Variante 3	2*100	0,91	0,74	$0,16^{0,038}$	$0,20^{0,001}$	0,19	$0,24^{0,004}$	0,31		0,25
		2*1000	0,91	0,74	$0,13^{0,006}$	$0,08^{0,003}$	$0,07^{0,001}$	0,06	0,06		0,06

Erreichte Präzision von APClustering für Änderungstyp (AP3a) bei $m_c = 10$.

Netz		Pfade	*rel*	*par*	Präzision ($m_c = 4$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,34	0,03	$0,12^{0,003}$	$0,17^{0,009}$	$0,25^{0,022}$	$0,35^{0,073}$	0,25		0,28
		2*1000	0,34	0,03	$0,37^{0,028}$	$0,85^{0,029}$	$0,97^{0,003}$	$0,68^{0,045}$	$0,71^{0,061}$		0,79
	Variante 2	2*100	0,34	0,03	$0,14^{0,007}$	$0,28^{0,004}$	$0,26^{0,003}$	$0,26^{0,001}$	$0,50^{0,016}$		0,34
		2*1000	0,34	0,03	$0,62^{0,132}$	1,00	1,00	1,00	1,00	✓	1,00
	Variante 3	2*100	0,34	0,03	$0,15^{0,009}$	$0,24^{0,002}$	0,33	$0,48^{0,004}$	$0,85^{0,052}$	✓	0,55
		2*1000	0,34	0,03	$0,40^{0,035}$	$0,69^{0,059}$	$0,99^{0,005}$	$0,98^{0,008}$	$0,99^{0,005}$		0,99
herbstFig6p41*	Variante 1	2*100	0,57	0,21	$0,12^{0,005}$	$0,24^{0,015}$	0,50	$0,32^{0,019}$	$0,81^{0,105}$		0,54
		2*1000	0,57	0,21	$0,37^{0,131}$	$0,47^{0,018}$	0,50	0,50	$0,53^{0,016}$	✓	0,51
	Variante 2	2*100	0,57	0,21	$0,07^{0,001}$	$0,21^{0,007}$	$0,40^{0,010}$	$0,79^{0,059}$	0,99	✓	0,73
		2*1000	0,57	0,21	$0,43^{0,108}$	$0,39^{0,026}$	$0,40^{0,007}$	$0,35^{0,003}$	$0,51^{0,006}$		0,42
	Variante 3	2*100	0,57	0,21	$0,11^{0,005}$	$0,17^{0,003}$	$0,45^{0,057}$	$0,57^{0,049}$	$0,70^{0,059}$	✓	0,57
		2*1000	0,57	0,21	$0,26^{0,029}$	$0,48^{0,002}$	$0,46^{0,062}$	$0,86^{0,033}$	1,00		0,77
herbstFig6p42	Variante 1	2*100	0,79	0,52	$0,12^{0,009}$	$0,17^{0,009}$	$0,45^{0,002}$	$0,30^{0,001}$	0,31		0,35
		2*1000	0,80	0,54	$0,08^{0,006}$	0,02	0,02	0,02	0,00		0,02
	Variante 2	2*100	0,80	0,54	$0,07^{0,001}$	$0,14^{0,002}$	$0,28^{0,027}$	$0,19^{0,024}$	$0,34^{0,106}$		0,27
		2*1000	0,80	0,54	$0,25^{0,044}$	$0,44^{0,127}$	$0,34^{0,109}$	$0,69^{0,107}$	0,50		0,51
	Variante 3	2*100	0,80	0,54	$0,08^{0,002}$	$0,17^{0,003}$	$0,31^{0,009}$	$0,11^{0,001}$	0,09		0,17
		2*1000	0,80	0,54	$0,14^{0,010}$	$0,07^{0,012}$	$0,09^{0,030}$	$0,52^{0,153}$	$0,48^{0,094}$		0,36
herbstFig6p18*	Variante 1	2*100	0,84	0,68	$0,13^{0,008}$	$0,39^{0,052}$	0,98	0,98	$0,73^{0,060}$		0,90
		2*1000	0,84	0,68	$0,21^{0,010}$	$0,28^{0,004}$	$0,28^{0,010}$	$0,31^{0,001}$	$0,36^{0,028}$		0,32
	Variante 2	2*100	0,84	0,68	$0,12^{0,004}$	$0,28^{0,008}$	$0,27^{0,007}$	0,13	0,13		0,17
		2*1000	0,84	0,68	$0,15^{0,006}$	$0,39^{0,008}$	$0,37^{0,020}$	$0,26^{0,010}$	$0,55^{0,052}$		0,39
	Variante 3	2*100	0,84	0,68	$0,11^{0,004}$	$0,23^{0,003}$	$0,18^{0,004}$	0,19	0,20		0,19
		2*1000	0,84	0,68	$0,11^{0,003}$	$0,16^{0,008}$	$0,12^{0,005}$	$0,22^{0,001}$	0,33		0,23
herbstFig6p34*	Variante 1	2*100	0,90	0,80	$0,41^{0,083}$	$0,25^{0,023}$	$0,32^{0,037}$	$0,44^{0,005}$	$0,73^{0,057}$		0,49
		2*1000	0,90	0,80	$0,14^{0,009}$	$0,36^{0,060}$	$0,45^{0,024}$	$0,49^{0,009}$	$0,71^{0,050}$	✓	0,55
	Variante 2	2*100	0,90	0,80	$0,12^{0,003}$	$0,18^{0,008}$	$0,29^{0,001}$	$0,30^{0,008}$	$0,26^{0,022}$		0,28
		2*1000	0,90	0,80	$0,14^{0,013}$	$0,36^{0,025}$	$0,38^{0,102}$	$0,48^{0,147}$	$0,47^{0,068}$		0,44
	Variante 3	2*100	0,90	0,80	0,08	$0,15^{0,010}$	$0,36^{0,044}$	$0,28^{0,017}$	$0,43^{0,035}$		0,35
		2*1000	0,90	0,80	$0,10^{0,004}$	$0,29^{0,007}$	$0,47^{0,001}$	$0,81^{0,050}$	$0,79^{0,138}$		0,69
al2*	Variante 1	2*100	1,00	0,81	$0,10^{0,001}$	$0,14^{0,001}$	$0,18^{0,001}$	0,18	0,16		0,17
		2*1000	1,00	0,81	$0,12^{0,005}$	$0,09^{0,006}$	0,04	0,07	0,08		0,06
	Variante 2	2*100	1,00	0,81	$0,11^{0,002}$	$0,11^{0,001}$	$0,20^{0,001}$	$0,41^{0,003}$	$0,39^{0,006}$		0,33
		2*1000	1,00	0,81	$0,10^{0,003}$	$0,18^{0,009}$	$0,24^{0,003}$	$0,65^{0,075}$	0,99	✓	0,62
	Variante 3	2*100	0,97	0,77	$0,10^{0,002}$	$0,21^{0,001}$	$0,30^{0,002}$	$0,31^{0,001}$	$0,35^{0,003}$	✓	0,32
		2*1000	1,00	0,81	$0,08^{0,006}$	0,02	$0,14^{0,031}$	0,50	0,50		0,38

Erreichte Präzision von APClustering für Änderungstyp (AP3b) bei $m_c = 4$.

Netz		Pfade	rel	par	Präzision ($m_c = 6$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,34	0,03	$0,13^{0,006}$	$0,26^{0,009}$	$0,33^{0,016}$	$0,37^{0,069}$	0,25		0,32
		2*1000	0,34	0,03	$0,36^{0,030}$	$0,85^{0,029}$	$0,96^{0,007}$	$0,68^{0,045}$	$0,71^{0,061}$		0,78
	Variante 2	2*100	0,34	0,03	$0,18^{0,041}$	$0,31^{0,019}$	$0,26^{0,003}$	$0,26^{0,001}$	$0,52^{0,026}$		0,34
		2*1000	0,34	0,03	$0,62^{0,132}$	1,00	1,00	1,00	1,00	✓	1,00
	Variante 3	2*100	0,34	0,03	$0,18^{0,040}$	$0,25^{0,002}$	0,33	$0,47^{0,003}$	$0,84^{0,051}$	✓	0,54
		2*1000	0,34	0,03	$0,37^{0,031}$	$0,69^{0,059}$	$0,99^{0,004}$	$0,98^{0,008}$	$0,99^{0,005}$		0,99
herbstFig6p41*	Variante 1	2*100	0,57	0,21	$0,15^{0,040}$	$0,23^{0,014}$	0,50	$0,32^{0,014}$	$0,81^{0,105}$		0,55
		2*1000	0,57	0,21	$0,32^{0,070}$	$0,44^{0,007}$	0,50	0,50	$0,54^{0,018}$	✓	0,51
	Variante 2	2*100	0,57	0,21	$0,12^{0,042}$	$0,21^{0,009}$	$0,40^{0,010}$	$0,79^{0,059}$	0,99	✓	0,73
		2*1000	0,57	0,21	$0,42^{0,111}$	$0,39^{0,026}$	$0,40^{0,007}$	$0,35^{0,003}$	$0,52^{0,008}$		0,42
	Variante 3	2*100	0,57	0,21	$0,17^{0,042}$	$0,16^{0,001}$	$0,65^{0,088}$	$0,73^{0,074}$	$0,90^{0,031}$		0,76
		2*1000	0,57	0,21	$0,24^{0,030}$	$0,48^{0,002}$	$0,46^{0,062}$	$0,79^{0,042}$	$0,94^{0,025}$		0,73
herbstFig6p42	Variante 1	2*100	0,79	0,52	$0,12^{0,005}$	$0,16^{0,007}$	$0,44^{0,003}$	0,30	0,31		0,35
		2*1000	0,80	0,54	$0,08^{0,007}$	0,02	0,02	0,02	0,00		0,02
	Variante 2	2*100	0,80	0,54	$0,13^{0,041}$	$0,14^{0,002}$	$0,36^{0,015}$	$0,20^{0,005}$	$0,38^{0,096}$		0,31
		2*1000	0,80	0,54	$0,37^{0,155}$	$0,57^{0,169}$	$0,37^{0,101}$	$0,84^{0,091}$	1,00		0,74
	Variante 3	2*100	0,80	0,54	$0,17^{0,076}$	$0,19^{0,005}$	$0,25^{0,002}$	$0,11^{0,001}$	$0,10^{0,002}$		0,15
		2*1000	0,80	0,54	$0,16^{0,015}$	$0,10^{0,027}$	$0,09^{0,031}$	$0,52^{0,150}$	$0,52^{0,083}$		0,38
herbstFig6p18*	Variante 1	2*100	0,84	0,68	$0,10^{0,003}$	$0,36^{0,058}$	0,98	0,98	$0,71^{0,057}$		0,89
		2*1000	0,84	0,68	$0,26^{0,025}$	$0,40^{0,014}$	$0,43^{0,037}$	$0,46^{0,005}$	$0,53^{0,015}$	✓	0,47
	Variante 2	2*100	0,84	0,68	$0,11^{0,001}$	$0,28^{0,007}$	$0,27^{0,007}$	0,13	0,13		0,17
		2*1000	0,84	0,68	$0,15^{0,006}$	$0,39^{0,008}$	$0,37^{0,020}$	$0,26^{0,011}$	$0,56^{0,050}$		0,40
	Variante 3	2*100	0,84	0,68	$0,11^{0,003}$	$0,19^{0,004}$	$0,16^{0,003}$	$0,21^{0,001}$	0,24		0,20
		2*1000	0,84	0,68	$0,08^{0,001}$	$0,12^{0,005}$	$0,14^{0,010}$	$0,29^{0,002}$	0,50	✓	0,31
herbstFig6p34*	Variante 1	2*100	0,90	0,80	$0,31^{0,059}$	$0,20^{0,004}$	$0,27^{0,006}$	0,48	$0,73^{0,057}$		0,49
		2*1000	0,90	0,80	$0,12^{0,006}$	$0,36^{0,060}$	$0,48^{0,036}$	$0,49^{0,009}$	$0,72^{0,050}$	✓	0,56
	Variante 2	2*100	0,90	0,80	$0,11^{0,002}$	$0,19^{0,006}$	0,30	$0,34^{0,004}$	$0,33^{0,024}$		0,32
		2*1000	0,90	0,80	$0,12^{0,007}$	$0,35^{0,023}$	$0,37^{0,102}$	$0,48^{0,147}$	$0,46^{0,057}$		0,44
	Variante 3	2*100	0,90	0,80	$0,12^{0,041}$	$0,14^{0,004}$	$0,36^{0,043}$	$0,31^{0,015}$	$0,56^{0,040}$		0,41
		2*1000	0,90	0,80	$0,10^{0,004}$	$0,29^{0,007}$	$0,47^{0,001}$	$0,81^{0,050}$	$0,79^{0,138}$		0,69
al2*	Variante 1	2*100	1,00	0,81	$0,10^{0,001}$	$0,13^{0,001}$	$0,17^{0,001}$	0,21	0,21	✓	0,20
		2*1000	1,00	0,81	$0,13^{0,006}$	$0,24^{0,011}$	$0,33^{0,001}$	$0,47^{0,003}$	0,50	✓	0,43
	Variante 2	2*100	1,00	0,81	$0,09^{0,001}$	$0,11^{0,001}$	$0,16^{0,001}$	$0,36^{0,007}$	$0,36^{0,011}$		0,29
		2*1000	1,00	0,81	$0,10^{0,003}$	$0,19^{0,011}$	$0,23^{0,003}$	$0,65^{0,075}$	0,99	✓	0,62
	Variante 3	2*100	0,97	0,77	$0,08^{0,001}$	$0,18^{0,001}$	$0,30^{0,003}$	0,32	$0,35^{0,003}$	✓	0,32
		2*1000	1,00	0,81	$0,10^{0,008}$	0,02	$0,14^{0,031}$	0,50	0,50		0,38

Erreichte Präzision von APClustering für Änderungstyp (AP3b) bei $m_c = 6$.

Netz		Pfade	*rel*	*par*	Präzision ($m_c = 10$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,34	0,03	$0,26^{0,098}$	$0,24^{0,003}$	$0,38^{0,005}$	$0,28^{0,016}$	0,25		0,30
		2*1000	0,34	0,03	$0,36^{0,031}$	$0,85^{0,031}$	$0,96^{0,010}$	$0,67^{0,044}$	$0,71^{0,061}$		0,78
	Variante 2	2*100	0,34	0,03	$0,22^{0,073}$	$0,27^{0,013}$	$0,34^{0,001}$	$0,40^{0,039}$	$0,89^{0,039}$	✓	0,54
		2*1000	0,34	0,03	$0,62^{0,134}$	1,00	1,00	1,00	1,00		1,00
	Variante 3	2*100	0,34	0,03	$0,26^{0,095}$	$0,32^{0,007}$	0,50	0,50	$0,52^{0,011}$	✓	0,50
		2*1000	0,34	0,03	$0,37^{0,030}$	$0,69^{0,059}$	$0,99^{0,004}$	1,00	1,00	✓	1,00
herbstFig6p41*	Variante 1	2*100	0,57	0,21	$0,26^{0,099}$	$0,23^{0,013}$	$0,43^{0,007}$	$0,29^{0,006}$	$0,81^{0,105}$		0,51
		2*1000	0,57	0,21	$0,36^{0,079}$	$0,76^{0,081}$	0,99	0,99	0,99	✓	0,99
	Variante 2	2*100	0,57	0,21	$0,29^{0,095}$	$0,14^{0,001}$	$0,48^{0,030}$	$0,79^{0,061}$	0,99		0,75
		2*1000	0,57	0,21	$0,38^{0,093}$	$0,39^{0,026}$	$0,40^{0,007}$	$0,35^{0,003}$	$0,53^{0,013}$		0,43
	Variante 3	2*100	0,57	0,21	$0,30^{0,123}$	$0,17^{0,001}$	$0,66^{0,088}$	$0,92^{0,031}$	0,98		0,85
		2*1000	0,57	0,21	$0,26^{0,033}$	$0,88^{0,053}$	$0,42^{0,008}$	$0,64^{0,093}$	$0,98^{0,006}$		0,68
herbstFig6p42	Variante 1	2*100	0,79	0,52	$0,32^{0,121}$	$0,16^{0,005}$	$0,44^{0,003}$	$0,44^{0,002}$	0,46		0,45
		2*1000	0,80	0,54	$0,07^{0,006}$	0,02	0,02	0,02	0,00		0,02
	Variante 2	2*100	0,80	0,54	$0,33^{0,149}$	$0,18^{0,003}$	$0,27^{0,003}$	$0,21^{0,005}$	$0,40^{0,090}$		0,29
		2*1000	0,80	0,54	$0,16^{0,019}$	$0,30^{0,020}$	$0,37^{0,100}$	$0,84^{0,091}$	1,00	✓	0,74
	Variante 3	2*100	0,80	0,54	$0,33^{0,148}$	$0,16^{0,003}$	$0,20^{0,001}$	$0,15^{0,003}$	$0,15^{0,006}$		0,17
		2*1000	0,80	0,54	$0,16^{0,021}$	$0,10^{0,027}$	$0,12^{0,038}$	$0,56^{0,131}$	$0,78^{0,070}$		0,48
herbstFig6p18*	Variante 1	2*100	0,84	0,68	$0,16^{0,009}$	$0,30^{0,006}$	0,49	0,49	$0,81^{0,055}$	✓	0,60
		2*1000	0,84	0,68	$0,19^{0,011}$	$0,33^{0,001}$	$0,50^{0,043}$	$0,84^{0,060}$	$0,99^{0,004}$	✓	0,78
	Variante 2	2*100	0,84	0,68	$0,11^{0,001}$	$0,26^{0,007}$	$0,34^{0,004}$	0,19	0,19		0,24
		2*1000	0,84	0,68	$0,14^{0,007}$	$0,39^{0,008}$	$0,38^{0,020}$	$0,27^{0,011}$	$0,61^{0,047}$		0,42
	Variante 3	2*100	0,84	0,68	$0,10^{0,001}$	$0,17^{0,001}$	$0,18^{0,004}$	$0,20^{0,001}$	0,24	✓	0,21
		2*1000	0,84	0,68	$0,09^{0,004}$	$0,09^{0,003}$	$0,20^{0,023}$	0,33	0,50		0,34
herbstFig6p34*	Variante 1	2*100	0,90	0,80	$0,26^{0,068}$	$0,21^{0,007}$	$0,28^{0,005}$	0,48	$0,73^{0,057}$		0,49
		2*1000	0,90	0,80	$0,12^{0,010}$	$0,36^{0,060}$	$0,50^{0,041}$	$0,49^{0,007}$	$0,73^{0,051}$		0,57
	Variante 2	2*100	0,90	0,80	$0,21^{0,076}$	$0,31^{0,015}$	$0,31^{0,002}$	$0,35^{0,003}$	$0,35^{0,022}$	✓	0,34
		2*1000	0,90	0,80	$0,12^{0,010}$	$0,33^{0,020}$	$0,33^{0,070}$	$0,36^{0,038}$	$0,48^{0,052}$		0,39
	Variante 3	2*100	0,90	0,80	$0,16^{0,040}$	$0,17^{0,012}$	$0,29^{0,027}$	$0,32^{0,002}$	$0,67^{0,066}$	✓	0,43
		2*1000	0,90	0,80	$0,10^{0,008}$	$0,28^{0,008}$	$0,47^{0,001}$	$0,81^{0,050}$	$0,79^{0,138}$		0,69
al2*	Variante 1	2*100	1,00	0,81	0,11	$0,14^{0,001}$	$0,18^{0,001}$	0,21	0,21		0,20
		2*1000	1,00	0,81	$0,12^{0,005}$	$0,24^{0,008}$	$0,33^{0,001}$	$0,47^{0,003}$	0,50	✓	0,43
	Variante 2	2*100	1,00	0,81	$0,14^{0,042}$	$0,14^{0,001}$	$0,25^{0,003}$	$0,31^{0,005}$	$0,44^{0,002}$		0,34
		2*1000	1,00	0,81	$0,08^{0,002}$	$0,20^{0,014}$	$0,26^{0,007}$	$0,49^{0,056}$	$0,49^{0,001}$	✓	0,41
	Variante 3	2*100	0,97	0,77	$0,15^{0,039}$	$0,14^{0,001}$	$0,28^{0,004}$	0,32	$0,34^{0,003}$		0,31
		2*1000	1,00	0,81	$0,09^{0,005}$	0,02	$0,13^{0,028}$	0,50	0,50		0,38

Erreichte Präzision von APClustering für Änderungstyp (AP3b) bei $m_c = 10$.

Netz		Pfade	rel	par	Präzision ($m_c = 4$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,34	0,03	$0,10^{0,003}$	$0,16^{0,020}$	0,31	0,31	0,31	✓	0,31
		2*1000	0,34	0,03	$0,41^{0,096}$	$0,49^{0,031}$	$0,49^{0,002}$	0,49	$0,52^{0,015}$		0,50
	Variante 2	2*100	0,34	0,03	$0,12^{0,004}$	$0,34^{0,040}$	$0,51^{0,012}$	1,00	$0,70^{0,060}$		0,74
		2*1000	0,34	0,03	$0,64^{0,116}$	$0,72^{0,062}$	$0,52^{0,007}$	$0,68^{0,058}$	1,00		0,73
	Variante 3	2*100	0,31	0,03	$0,16^{0,015}$	$0,48^{0,001}$	$0,63^{0,050}$	0,98	0,98	✓	0,86
		2*1000	0,31	0,03	$0,31^{0,018}$	0,50	0,50	0,50	$0,64^{0,050}$	✓	0,55
herbstFig6p41*	Variante 1	2*100	0,57	0,21	$0,09^{0,003}$	$0,24^{0,013}$	$0,35^{0,018}$	$0,58^{0,098}$	$0,57^{0,220}$		0,50
		2*1000	0,57	0,21	$0,26^{0,025}$	$0,57^{0,135}$	$0,71^{0,130}$	$0,07^{0,061}$	$0,13^{0,087}$		0,30
	Variante 2	2*100	0,57	0,21	$0,21^{0,056}$	$0,22^{0,006}$	$0,38^{0,059}$	$0,50^{0,048}$	$0,64^{0,051}$	✓	0,50
		2*1000	0,57	0,21	$0,55^{0,200}$	$0,68^{0,061}$	$0,87^{0,048}$	1,00	1,00	✓	0,96
	Variante 3	2*100	0,57	0,21	$0,07^{0,001}$	$0,18^{0,006}$	$0,40^{0,005}$	$0,35^{0,017}$	$0,42^{0,005}$		0,39
		2*1000	0,57	0,21	$0,19^{0,027}$	$0,52^{0,070}$	$0,47^{0,073}$	$0,31^{0,001}$	$0,36^{0,010}$		0,38
herbstFig6p42	Variante 1	2*100	0,80	0,54	$0,09^{0,001}$	$0,21^{0,003}$	$0,28^{0,005}$	0,33	0,33		0,32
		2*1000	0,80	0,54	$0,19^{0,017}$	$0,35^{0,005}$	$0,36^{0,004}$	$0,53^{0,033}$	1,00	✓	0,63
	Variante 2	2*100	0,80	0,54	$0,10^{0,002}$	$0,22^{0,017}$	$0,23^{0,001}$	0,32	$0,35^{0,007}$	✓	0,30
		2*1000	0,80	0,54	$0,07^{0,002}$	0,07	0,12	0,12	0,12		0,12
	Variante 3	2*100	0,80	0,54	$0,09^{0,002}$	$0,23^{0,012}$	$0,25^{0,001}$	0,29	$0,34^{0,007}$	✓	0,29
		2*1000	0,80	0,54	$0,06^{0,004}$	0,03	$0,32^{0,004}$	$0,34^{0,002}$	$0,50^{0,105}$		0,39
herbstFig6p18*	Variante 1	2*100	0,58	0,47	$0,11^{0,004}$	0,05	0,05	0,05	0,05		0,05
		2*1000	0,58	0,47	$0,14^{0,009}$	$0,28^{0,003}$	0,33	0,33	0,33	✓	0,33
	Variante 2	2*100	0,84	0,68	$0,13^{0,004}$	$0,21^{0,013}$	$0,37^{0,021}$	$0,24^{0,016}$	$0,34^{0,048}$		0,32
		2*1000	0,84	0,68	$0,14^{0,010}$	$0,35^{0,014}$	$0,39^{0,017}$	$0,47^{0,006}$	$0,67^{0,057}$	✓	0,51
	Variante 3	2*100	0,58	0,47	$0,15^{0,009}$	$0,18^{0,001}$	$0,22^{0,002}$	0,26	0,26	✓	0,24
		2*1000	0,58	0,47	$0,06^{0,006}$	0,03	0,04	0,04	0,05		0,04
herbstFig6p34*	Variante 1	2*100	0,90	0,80	$0,12^{0,003}$	$0,24^{0,005}$	$0,36^{0,013}$	$0,44^{0,009}$	0,48	✓	0,43
		2*1000	0,90	0,80	$0,11^{0,006}$	$0,36^{0,018}$	$0,62^{0,124}$	$0,72^{0,044}$	0,88	✓	0,74
	Variante 2	2*100	0,90	0,80	$0,15^{0,005}$	$0,19^{0,001}$	$0,21^{0,002}$	0,31	$0,61^{0,061}$	✓	0,38
		2*1000	0,90	0,80	$0,08^{0,003}$	$0,26^{0,041}$	$0,12^{0,001}$	0,16	$0,25^{0,013}$		0,18
	Variante 3	2*100	0,90	0,80	$0,42^{0,158}$	$0,32^{0,042}$	$0,38^{0,044}$	$0,32^{0,003}$	$0,23^{0,001}$		0,31
		2*1000	0,90	0,80	$0,09^{0,002}$	$0,27^{0,008}$	$0,45^{0,028}$	$0,46^{0,053}$	$0,91^{0,004}$	✓	0,61
al2*	Variante 1	2*100	0,91	0,74	$0,10^{0,002}$	$0,24^{0,016}$	$0,22^{0,006}$	0,33	0,32		0,29
		2*1000	0,91	0,74	$0,23^{0,029}$	$0,10^{0,004}$	$0,16^{0,001}$	0,20	0,20		0,19
	Variante 2	2*100	0,83	0,67	$0,24^{0,030}$	0,18	$0,18^{0,001}$	$0,14^{0,001}$	$0,15^{0,002}$		0,16
		2*1000	0,83	0,67	$0,15^{0,035}$	$0,15^{0,033}$	0,14	0,19	$0,23^{0,001}$		0,18
	Variante 3	2*100	1,00	0,81	$0,12^{0,004}$	$0,14^{0,001}$	0,20	0,14	0,14		0,16
		2*1000	0,91	0,74	$0,10^{0,004}$	$0,08^{0,004}$	0,16	$0,14^{0,005}$	$0,06^{0,003}$		0,12

Erreichte Präzision von APClustering für Änderungstyp (AP3c) bei $m_c = 4$.

Netz		Pfade	*rel*	*par*	Präzision ($m_c = 6$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,34	0,03	$0,15^{0,039}$	$0,16^{0,020}$	0,31	0,31	0,31	✓	0,31
		2*1000	0,34	0,03	$0,41^{0,096}$	$0,49^{0,031}$	$0,49^{0,002}$	0,49	$0,52^{0,015}$		0,50
	Variante 2	2*100	0,34	0,03	$0,15^{0,040}$	$0,29^{0,026}$	$0,49^{0,016}$	1,00	$0,69^{0,063}$		0,73
		2*1000	0,34	0,03	$0,58^{0,113}$	$0,68^{0,057}$	$0,51^{0,007}$	$0,68^{0,058}$	1,00		0,73
	Variante 3	2*100	0,31	0,03	$0,14^{0,008}$	$0,38^{0,006}$	$0,48^{0,061}$	$0,88^{0,036}$	0,98	✓	0,78
		2*1000	0,31	0,03	$0,31^{0,019}$	0,50	0,50	0,50	$0,64^{0,050}$	✓	0,55
herbstFig6p41*	Variante 1	2*100	0,57	0,21	$0,14^{0,042}$	$0,23^{0,014}$	$0,40^{0,011}$	$0,66^{0,064}$	$0,57^{0,220}$		0,54
		2*1000	0,57	0,21	$0,27^{0,028}$	$0,57^{0,135}$	$0,71^{0,130}$	$0,07^{0,061}$	$0,13^{0,087}$		0,30
	Variante 2	2*100	0,57	0,21	$0,26^{0,085}$	$0,20^{0,003}$	$0,50^{0,104}$	$0,47^{0,035}$	$0,66^{0,058}$		0,54
		2*1000	0,57	0,21	$0,40^{0,124}$	$0,48^{0,041}$	$0,83^{0,069}$	1,00	1,00	✓	0,94
	Variante 3	2*100	0,57	0,21	$0,16^{0,078}$	$0,17^{0,003}$	$0,40^{0,006}$	$0,33^{0,017}$	$0,44^{0,002}$		0,39
		2*1000	0,57	0,21	$0,27^{0,034}$	$0,71^{0,060}$	$0,81^{0,059}$	$0,89^{0,042}$	$0,70^{0,059}$		0,80
herbstFig6p42	Variante 1	2*100	0,80	0,54	$0,13^{0,041}$	$0,16^{0,003}$	$0,16^{0,001}$	0,16	$0,19^{0,001}$		0,17
		2*1000	0,80	0,54	$0,20^{0,017}$	$0,37^{0,005}$	0,50	$0,56^{0,026}$	1,00	✓	0,68
	Variante 2	2*100	0,80	0,54	$0,11^{0,005}$	$0,17^{0,007}$	$0,23^{0,002}$	0,32	$0,35^{0,009}$	✓	0,30
		2*1000	0,80	0,54	$0,07^{0,002}$	0,07	0,12	0,12	0,12	✓	0,12
	Variante 3	2*100	0,80	0,54	$0,12^{0,008}$	$0,20^{0,003}$	$0,25^{0,001}$	$0,39^{0,010}$	0,49	✓	0,38
		2*1000	0,80	0,54	$0,06^{0,006}$	0,03	$0,48^{0,007}$	$0,53^{0,015}$	$0,66^{0,158}$		0,55
herbstFig6p18*	Variante 1	2*100	0,58	0,47	$0,10^{0,002}$	0,05	0,05	0,05	0,05		0,05
		2*1000	0,58	0,47	$0,14^{0,010}$	$0,28^{0,003}$	0,33	0,33	0,33	✓	0,33
	Variante 2	2*100	0,84	0,68	$0,12^{0,002}$	$0,18^{0,010}$	$0,33^{0,023}$	$0,25^{0,016}$	$0,35^{0,047}$		0,31
		2*1000	0,84	0,68	$0,14^{0,008}$	$0,30^{0,012}$	$0,39^{0,017}$	$0,47^{0,007}$	$0,67^{0,057}$	✓	0,51
	Variante 3	2*100	0,58	0,47	$0,16^{0,011}$	$0,15^{0,001}$	$0,20^{0,003}$	0,26	0,26		0,24
		2*1000	0,58	0,47	$0,06^{0,006}$	0,03	0,04	0,04	0,05		0,04
herbstFig6p34*	Variante 1	2*100	0,90	0,80	$0,13^{0,006}$	$0,23^{0,004}$	$0,28^{0,007}$	$0,46^{0,006}$	0,48	✓	0,41
		2*1000	0,90	0,80	$0,12^{0,004}$	$0,35^{0,016}$	$0,55^{0,105}$	$0,70^{0,047}$	0,88	✓	0,71
	Variante 2	2*100	0,90	0,80	$0,16^{0,005}$	$0,19^{0,001}$	$0,21^{0,002}$	0,31	$0,61^{0,061}$	✓	0,38
		2*1000	0,90	0,80	$0,07^{0,002}$	$0,24^{0,033}$	$0,12^{0,001}$	0,16	$0,25^{0,014}$		0,18
	Variante 3	2*100	0,90	0,80	$0,43^{0,153}$	$0,31^{0,044}$	$0,38^{0,044}$	$0,33^{0,002}$	0,22		0,31
		2*1000	0,90	0,80	$0,08^{0,002}$	$0,26^{0,009}$	$0,44^{0,028}$	$0,48^{0,044}$	$0,91^{0,004}$	✓	0,61
al2*	Variante 1	2*100	0,91	0,74	$0,10^{0,002}$	$0,17^{0,002}$	$0,22^{0,006}$	0,33	0,32		0,29
		2*1000	0,91	0,74	$0,16^{0,013}$	$0,11^{0,005}$	$0,18^{0,001}$	0,20	0,20		0,19
	Variante 2	2*100	0,83	0,67	$0,17^{0,007}$	$0,16^{0,001}$	$0,20^{0,002}$	$0,17^{0,001}$	$0,17^{0,003}$		0,18
		2*1000	0,83	0,67	$0,17^{0,041}$	$0,27^{0,043}$	$0,16^{0,001}$	$0,23^{0,001}$	$0,30^{0,002}$		0,23
	Variante 3	2*100	1,00	0,81	$0,11^{0,002}$	0,12	$0,19^{0,001}$	0,14	0,14		0,15
		2*1000	0,91	0,74	$0,09^{0,003}$	$0,09^{0,003}$	0,15	$0,15^{0,005}$	$0,08^{0,005}$		0,13

Erreichte Präzision von APClustering für Änderungstyp (AP3c) bei $m_c = 6$.

Netz		Pfade	rel	par	Präzision ($m_c = 10$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,34	0,03	$0,27^{0,106}$	$0,16^{0,019}$	0,31	0,31	0,31		0,31
		2*1000	0,34	0,03	$0,42^{0,100}$	$0,49^{0,031}$	$0,49^{0,002}$	0,49	$0,52^{0,015}$		0,50
	Variante 2	2*100	0,34	0,03	$0,23^{0,073}$	$0,30^{0,007}$	$0,34^{0,003}$	$0,62^{0,052}$	$0,91^{0,021}$	✓	0,62
		2*1000	0,34	0,03	$0,51^{0,122}$	$0,65^{0,066}$	$0,51^{0,007}$	$0,68^{0,058}$	1,00		0,73
	Variante 3	2*100	0,31	0,03	$0,33^{0,117}$	$0,45^{0,004}$	$0,63^{0,051}$	0,98	0,99	✓	0,87
		2*1000	0,31	0,03	$0,31^{0,023}$	0,50	0,50	0,50	$0,64^{0,050}$	✓	0,55
herbstFig6p41*	Variante 1	2*100	0,57	0,21	$0,27^{0,099}$	$0,24^{0,006}$	$0,40^{0,011}$	$0,66^{0,064}$	$0,57^{0,220}$		0,54
		2*1000	0,57	0,21	$0,37^{0,050}$	$0,60^{0,123}$	$0,71^{0,128}$	$0,07^{0,061}$	$0,17^{0,143}$		0,32
	Variante 2	2*100	0,57	0,21	$0,29^{0,127}$	$0,21^{0,003}$	$0,42^{0,048}$	$0,51^{0,011}$	$0,66^{0,056}$		0,53
		2*1000	0,57	0,21	$0,33^{0,064}$	$0,48^{0,034}$	$0,73^{0,067}$	1,00	1,00		0,91
	Variante 3	2*100	0,57	0,21	$0,33^{0,118}$	$0,17^{0,001}$	$0,38^{0,012}$	$0,33^{0,017}$	0,45		0,39
		2*1000	0,57	0,21	$0,30^{0,047}$	$0,72^{0,061}$	$0,84^{0,055}$	$0,89^{0,042}$	$0,70^{0,060}$		0,81
herbstFig6p42	Variante 1	2*100	0,80	0,54	$0,26^{0,102}$	$0,12^{0,001}$	0,16	0,17	0,17		0,17
		2*1000	0,80	0,54	$0,20^{0,024}$	$0,37^{0,005}$	0,50	$0,56^{0,025}$	1,00	✓	0,68
	Variante 2	2*100	0,80	0,54	$0,25^{0,103}$	$0,15^{0,004}$	$0,30^{0,001}$	0,32	$0,35^{0,009}$		0,32
		2*1000	0,80	0,54	$0,07^{0,010}$	0,07	0,12	0,12	0,12	✓	0,12
	Variante 3	2*100	0,80	0,54	$0,34^{0,152}$	$0,20^{0,004}$	$0,24^{0,001}$	$0,38^{0,006}$	0,49		0,37
		2*1000	0,80	0,54	$0,16^{0,044}$	$0,44^{0,022}$	$0,49^{0,001}$	$0,53^{0,015}$	$0,57^{0,123}$	✓	0,53
herbstFig6p18*	Variante 1	2*100	0,58	0,47	$0,11^{0,003}$	0,05	$0,09^{0,012}$	$0,09^{0,012}$	0,05		0,07
		2*1000	0,58	0,47	$0,14^{0,010}$	$0,27^{0,004}$	0,33	0,32	0,33		0,33
	Variante 2	2*100	0,84	0,68	$0,13^{0,003}$	$0,17^{0,009}$	$0,34^{0,025}$	$0,23^{0,001}$	$0,39^{0,053}$		0,32
		2*1000	0,84	0,68	$0,12^{0,004}$	$0,25^{0,005}$	$0,40^{0,016}$	$0,47^{0,007}$	$0,67^{0,057}$	✓	0,51
	Variante 3	2*100	0,58	0,47	0,09	0,18	$0,31^{0,012}$	0,42	0,43	✓	0,39
		2*1000	0,58	0,47	$0,06^{0,004}$	0,03	$0,29^{0,010}$	$0,04^{0,001}$	0,05		0,13
herbstFig6p34*	Variante 1	2*100	0,90	0,80	$0,21^{0,071}$	$0,23^{0,004}$	$0,32^{0,005}$	$0,49^{0,002}$	0,48		0,43
		2*1000	0,90	0,80	$0,10^{0,007}$	$0,34^{0,015}$	$0,54^{0,099}$	$0,68^{0,048}$	0,88	✓	0,70
	Variante 2	2*100	0,90	0,80	$0,26^{0,098}$	$0,17^{0,002}$	$0,20^{0,002}$	0,31	$0,69^{0,073}$		0,40
		2*1000	0,90	0,80	$0,07^{0,006}$	$0,19^{0,010}$	$0,14^{0,001}$	0,20	$0,35^{0,037}$		0,23
	Variante 3	2*100	0,90	0,80	$0,37^{0,120}$	$0,33^{0,057}$	$0,34^{0,050}$	$0,33^{0,002}$	0,22		0,29
		2*1000	0,90	0,80	$0,08^{0,007}$	$0,23^{0,007}$	$0,39^{0,018}$	$0,48^{0,044}$	$0,91^{0,004}$	✓	0,60
al2*	Variante 1	2*100	0,91	0,74	$0,14^{0,039}$	$0,17^{0,001}$	$0,23^{0,004}$	0,31	0,31		0,28
		2*1000	0,91	0,74	$0,18^{0,013}$	$0,11^{0,005}$	$0,18^{0,001}$	0,20	0,20		0,19
	Variante 2	2*100	0,83	0,67	$0,17^{0,006}$	$0,17^{0,001}$	$0,19^{0,001}$	$0,21^{0,001}$	$0,19^{0,002}$		0,20
		2*1000	0,83	0,67	$0,17^{0,035}$	$0,34^{0,031}$	0,14	0,19	$0,23^{0,001}$		0,18
	Variante 3	2*100	1,00	0,81	$0,20^{0,074}$	0,12	0,17	$0,15^{0,001}$	0,14		0,15
		2*1000	0,91	0,74	$0,07^{0,002}$	$0,08^{0,002}$	0,16	$0,20^{0,001}$	0,24	✓	0,20

Erreichte Präzision von APClustering für Änderungstyp (AP3c) bei $m_c = 10$.

Netz		Pfade	*rel*	*par*	Präzision ($m_c = 4$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,28	0,03	$0,15^{0,010}$	$0,48^{0,001}$	0,49	$0,79^{0,058}$	$0,94^{0,022}$	✓	0,74
		2*1000	0,28	0,03	$0,78^{0,132}$	1,00	1,00	1,00	1,00	✓	1,00
	Variante 2	2*100	0,28	0,03	$0,16^{0,010}$	$0,41^{0,013}$	$0,50^{0,011}$	0,96	0,96	✓	0,81
		2*1000	0,28	0,03	$0,77^{0,159}$	1,00	1,00	1,00	1,00		1,00
	Variante 3	2*100	0,28	0,03	$0,14^{0,011}$	$0,73^{0,090}$	0,99	0,99	0,99	✓	0,99
		2*1000	0,28	0,03	$0,55^{0,102}$	1,00	1,00	1,00	1,00	✓	1,00
herbstFig6p41*	Variante 1	2*100	0,43	0,16	$0,11^{0,009}$	$0,30^{0,040}$	$0,58^{0,061}$	0,50	$0,60^{0,039}$		0,56
		2*1000	0,43	0,16	$0,26^{0,014}$	0,33	0,33	0,33	$0,34^{0,002}$	✓	0,33
	Variante 2	2*100	0,43	0,16	$0,15^{0,011}$	$0,22^{0,010}$	$0,22^{0,016}$	$0,83^{0,057}$	$0,92^{0,032}$	✓	0,66
		2*1000	0,43	0,16	$0,74^{0,160}$	1,00	1,00	1,00	1,00	✓	1,00
	Variante 3	2*100	0,50	0,19	$0,16^{0,012}$	$0,39^{0,054}$	0,98	0,98	$0,93^{0,021}$		0,97
		2*1000	0,50	0,19	$0,78^{0,149}$	1,00	1,00	1,00	1,00	✓	1,00
herbstFig6p42	Variante 1	2*100	0,63	0,42	$0,12^{0,003}$	$0,24^{0,007}$	0,32	0,32	$0,42^{0,058}$		0,36
		2*1000	0,63	0,42	$0,29^{0,029}$	0,50	$0,90^{0,041}$	1,00	1,00		0,96
	Variante 2	2*100	0,63	0,42	$0,11^{0,002}$	$0,19^{0,001}$	$0,19^{0,001}$	0,20	0,20	✓	0,20
		2*1000	0,63	0,42	$0,06^{0,003}$	0,03	0,03	0,03	0,03		0,03
	Variante 3	2*100	0,63	0,42	$0,09^{0,002}$	$0,36^{0,085}$	$0,31^{0,012}$	$0,33^{0,014}$	$0,30^{0,008}$		0,31
		2*1000	0,63	0,42	$0,30^{0,039}$	$0,77^{0,061}$	0,99	0,99	0,99	✓	0,99
herbstFig6p18*	Variante 1	2*100	0,58	0,47	$0,14^{0,002}$	$0,18^{0,013}$	$0,36^{0,006}$	0,50	$0,42^{0,023}$		0,42
		2*1000	0,58	0,47	$0,20^{0,024}$	$0,31^{0,002}$	$0,39^{0,010}$	0,50	$0,86^{0,050}$	✓	0,58
	Variante 2	2*100	0,56	0,42	$0,11^{0,001}$	$0,21^{0,003}$	$0,33^{0,004}$	$0,35^{0,002}$	$0,45^{0,001}$	✓	0,38
		2*1000	0,58	0,47	$0,26^{0,015}$	$0,73^{0,067}$	1,00	1,00	$0,89^{0,044}$		0,96
	Variante 3	2*100	0,58	0,47	$0,11^{0,003}$	$0,21^{0,004}$	0,32	0,32	$0,52^{0,075}$	✓	0,39
		2*1000	0,58	0,47	$0,25^{0,023}$	$0,88^{0,044}$	1,00	1,00	$0,91^{0,036}$		0,97
herbstFig6p34*	Variante 1	2*100	0,69	0,61	$0,19^{0,016}$	$0,20^{0,004}$	$0,34^{0,005}$	$0,38^{0,013}$	$0,46^{0,003}$	✓	0,40
		2*1000	0,69	0,61	$0,09^{0,004}$	$0,18^{0,004}$	$0,41^{0,006}$	0,49	$0,61^{0,045}$	✓	0,50
	Variante 2	2*100	0,69	0,61	$0,21^{0,023}$	$0,50^{0,028}$	$0,65^{0,048}$	0,95	$0,88^{0,028}$		0,83
		2*1000	0,69	0,61	$0,39^{0,037}$	$0,58^{0,039}$	$0,75^{0,067}$	$0,95^{0,023}$	1,00	✓	0,90
	Variante 3	2*100	0,69	0,61	$0,11^{0,002}$	$0,17^{0,002}$	$0,25^{0,002}$	$0,42^{0,010}$	$0,36^{0,004}$		0,34
		2*1000	0,69	0,61	$0,08^{0,001}$	$0,16^{0,002}$	$0,29^{0,002}$	$0,42^{0,007}$	0,50	✓	0,40
al2*	Variante 1	2*100	0,80	0,64	$0,25^{0,064}$	$0,58^{0,070}$	0,33	0,33	$0,36^{0,007}$		0,34
		2*1000	0,83	0,67	$0,25^{0,028}$	0,50	$0,86^{0,050}$	1,00	1,00	✓	0,95
	Variante 2	2*100	0,83	0,67	$0,15^{0,010}$	$0,17^{0,004}$	$0,24^{0,002}$	0,25	$0,34^{0,049}$	✓	0,27
		2*1000	0,83	0,67	$0,17^{0,008}$	$0,32^{0,001}$	0,33	0,33	0,33		0,33
	Variante 3	2*100	0,83	0,67	$0,09^{0,001}$	$0,36^{0,020}$	0,49	$0,42^{0,013}$	$0,26^{0,003}$		0,39
		2*1000	0,83	0,67	$0,25^{0,014}$	0,33	$0,45^{0,006}$	0,50	$0,50^{0,003}$	✓	0,49

Erreichte Präzision von APClustering für Änderungstyp (AP4) bei $m_c = 4$.

Netz		Pfade	*rel*	*par*	Präzision ($m_c = 6$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,28	0,03	$0,17^{0,010}$	$0,47^{0,004}$	0,49	$0,79^{0,058}$	0,99	✓	0,76
		2*1000	0,28	0,03	$0,78^{0,132}$	1,00	1,00	1,00	1,00		1,00
	Variante 2	2*100	0,28	0,03	$0,17^{0,005}$	$0,40^{0,008}$	$0,49^{0,010}$	0,94	0,95	✓	0,79
		2*1000	0,28	0,03	$0,77^{0,160}$	1,00	1,00	1,00	1,00		1,00
	Variante 3	2*100	0,28	0,03	$0,17^{0,039}$	$0,70^{0,109}$	0,99	0,99	0,99	✓	0,99
		2*1000	0,28	0,03	$0,55^{0,101}$	1,00	1,00	1,00	1,00	✓	1,00
herbstFig6p41*	Variante 1	2*100	0,43	0,16	$0,16^{0,048}$	$0,22^{0,015}$	$0,86^{0,057}$	0,96	$0,89^{0,029}$		0,90
		2*1000	0,43	0,16	$0,73^{0,161}$	1,00	1,00	1,00	$0,99^{0,001}$		1,00
	Variante 2	2*100	0,43	0,16	$0,19^{0,043}$	$0,19^{0,005}$	$0,22^{0,015}$	$0,85^{0,052}$	1,00	✓	0,69
		2*1000	0,43	0,16	$0,74^{0,161}$	1,00	1,00	1,00	1,00	✓	1,00
	Variante 3	2*100	0,50	0,19	$0,19^{0,040}$	$0,40^{0,082}$	1,00	1,00	1,00		1,00
		2*1000	0,50	0,19	$0,77^{0,157}$	1,00	1,00	1,00	1,00	✓	1,00
herbstFig6p42	Variante 1	2*100	0,63	0,42	$0,13^{0,007}$	$0,14^{0,003}$	0,19	$0,30^{0,002}$	0,32	✓	0,27
		2*1000	0,63	0,42	$0,29^{0,029}$	0,50	$0,90^{0,041}$	1,00	1,00		0,96
	Variante 2	2*100	0,63	0,42	$0,13^{0,003}$	$0,18^{0,001}$	$0,19^{0,001}$	0,20	0,20	✓	0,20
		2*1000	0,63	0,42	$0,05^{0,002}$	0,03	0,03	0,03	0,03		0,03
	Variante 3	2*100	0,63	0,42	$0,13^{0,042}$	$0,19^{0,007}$	$0,23^{0,003}$	0,32	0,32	✓	0,29
		2*1000	0,63	0,42	$0,20^{0,016}$	$0,42^{0,007}$	0,50	0,50	0,50		0,50
herbstFig6p18*	Variante 1	2*100	0,58	0,47	$0,13^{0,002}$	$0,29^{0,008}$	$0,37^{0,005}$	0,50	$0,43^{0,023}$		0,43
		2*1000	0,58	0,47	$0,18^{0,025}$	$0,29^{0,004}$	$0,39^{0,011}$	0,50	$0,86^{0,050}$	✓	0,58
	Variante 2	2*100	0,56	0,42	$0,12^{0,001}$	$0,24^{0,003}$	$0,32^{0,005}$	$0,35^{0,002}$	$0,46^{0,001}$	✓	0,38
		2*1000	0,58	0,47	$0,23^{0,013}$	$0,73^{0,067}$	1,00	1,00	$0,84^{0,054}$		0,94
	Variante 3	2*100	0,58	0,47	$0,11^{0,004}$	$0,20^{0,005}$	0,32	0,32	$0,53^{0,072}$	✓	0,39
		2*1000	0,58	0,47	$0,20^{0,017}$	$0,88^{0,044}$	1,00	1,00	$0,91^{0,036}$		0,97
herbstFig6p34*	Variante 1	2*100	0,69	0,61	$0,15^{0,009}$	$0,18^{0,003}$	$0,34^{0,006}$	$0,38^{0,013}$	$0,46^{0,003}$	✓	0,40
		2*1000	0,69	0,61	$0,08^{0,003}$	$0,18^{0,004}$	$0,40^{0,007}$	0,49	$0,62^{0,045}$	✓	0,50
	Variante 2	2*100	0,69	0,61	$0,17^{0,006}$	$0,36^{0,012}$	$0,60^{0,040}$	0,95	$0,88^{0,028}$		0,81
		2*1000	0,69	0,61	$0,33^{0,029}$	$0,52^{0,013}$	$0,70^{0,064}$	$0,94^{0,027}$	1,00	✓	0,88
	Variante 3	2*100	0,69	0,61	$0,10^{0,003}$	$0,15^{0,001}$	$0,25^{0,002}$	$0,38^{0,009}$	$0,40^{0,007}$	✓	0,35
		2*1000	0,69	0,61	$0,07^{0,001}$	$0,15^{0,002}$	$0,29^{0,002}$	$0,42^{0,007}$	0,50	✓	0,40
al2*	Variante 1	2*100	0,80	0,64	$0,22^{0,045}$	$0,48^{0,041}$	0,33	0,33	$0,39^{0,006}$		0,35
		2*1000	0,83	0,67	$0,24^{0,029}$	0,50	$0,86^{0,050}$	1,00	1,00		0,95
	Variante 2	2*100	0,83	0,67	$0,11^{0,002}$	$0,30^{0,002}$	$0,29^{0,002}$	0,32	$0,39^{0,041}$		0,33
		2*1000	0,83	0,67	$0,16^{0,008}$	$0,32^{0,001}$	0,33	0,33	0,33		0,33
	Variante 3	2*100	0,83	0,67	$0,09^{0,001}$	$0,26^{0,006}$	$0,44^{0,006}$	$0,42^{0,013}$	$0,26^{0,003}$		0,37
		2*1000	0,83	0,67	$0,33^{0,037}$	0,50	0,50	0,50	$0,50^{0,003}$		0,50

Erreichte Präzision von APClustering für Änderungstyp (AP4) bei $m_c = 6$.

Netz		Pfade	*rel*	*par*	Präzision ($m_c = 10$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,28	0,03	$0,27^{0,070}$	$0,47^{0,005}$	0,49	$0,79^{0,058}$	0,99	✓	0,76
		2*1000	0,28	0,03	$0,78^{0,127}$	1,00	1,00	1,00	1,00		1,00
	Variante 2	2*100	0,28	0,03	$0,22^{0,039}$	$0,38^{0,013}$	$0,52^{0,020}$	$0,92^{0,010}$	0,95	✓	0,79
		2*1000	0,28	0,03	$0,80^{0,126}$	1,00	1,00	1,00	1,00	✓	1,00
	Variante 3	2*100	0,28	0,03	$0,24^{0,071}$	$0,69^{0,105}$	0,98	0,98	0,97		0,97
		2*1000	0,28	0,03	$0,56^{0,100}$	1,00	1,00	1,00	1,00	✓	1,00
herbstFig6p41*	Variante 1	2*100	0,43	0,16	$0,32^{0,118}$	$0,21^{0,016}$	$0,86^{0,057}$	0,96	$0,93^{0,011}$		0,92
		2*1000	0,43	0,16	$0,72^{0,173}$	1,00	1,00	1,00	1,00		1,00
	Variante 2	2*100	0,43	0,16	$0,28^{0,096}$	$0,20^{0,002}$	$0,26^{0,011}$	$0,83^{0,050}$	0,98		0,69
		2*1000	0,43	0,16	$0,39^{0,041}$	0,50	0,50	0,50	0,50		0,50
	Variante 3	2*100	0,50	0,19	$0,29^{0,101}$	$0,29^{0,044}$	$0,93^{0,040}$	1,00	1,00		0,98
		2*1000	0,50	0,19	$0,76^{0,159}$	1,00	1,00	1,00	1,00	✓	1,00
herbstFig6p42	Variante 1	2*100	0,63	0,42	$0,30^{0,122}$	$0,15^{0,001}$	0,20	$0,29^{0,003}$	$0,40^{0,034}$		0,29
		2*1000	0,63	0,42	$0,29^{0,036}$	0,50	$0,90^{0,041}$	1,00	1,00		0,96
	Variante 2	2*100	0,63	0,42	$0,30^{0,122}$	0,17	$0,18^{0,001}$	0,20	0,20		0,19
		2*1000	0,63	0,42	$0,06^{0,011}$	0,03	0,03	0,03	0,03		0,03
	Variante 3	2*100	0,63	0,42	$0,26^{0,103}$	$0,29^{0,016}$	$0,36^{0,008}$	$0,30^{0,002}$	$0,32^{0,001}$		0,33
		2*1000	0,63	0,42	$0,21^{0,020}$	$0,32^{0,001}$	$0,32^{0,001}$	$0,42^{0,007}$	$0,45^{0,006}$		0,40
herbstFig6p18*	Variante 1	2*100	0,58	0,47	$0,14^{0,003}$	$0,29^{0,008}$	$0,43^{0,005}$	$0,67^{0,056}$	0,98	✓	0,69
		2*1000	0,58	0,47	$0,17^{0,023}$	$0,29^{0,004}$	$0,39^{0,010}$	0,50	$0,86^{0,050}$	✓	0,58
	Variante 2	2*100	0,56	0,42	$0,15^{0,038}$	$0,22^{0,001}$	$0,28^{0,003}$	$0,27^{0,002}$	$0,42^{0,007}$		0,32
		2*1000	0,58	0,47	$0,22^{0,015}$	$0,73^{0,067}$	1,00	1,00	$0,75^{0,062}$		0,92
	Variante 3	2*100	0,58	0,47	$0,13^{0,007}$	$0,18^{0,002}$	0,24	0,24	$0,53^{0,073}$	✓	0,34
		2*1000	0,58	0,47	$0,14^{0,005}$	$0,31^{0,001}$	0,33	0,33	$0,35^{0,008}$	✓	0,34
herbstFig6p34*	Variante 1	2*100	0,69	0,61	$0,21^{0,070}$	$0,17^{0,002}$	$0,33^{0,008}$	$0,27^{0,004}$	$0,56^{0,083}$		0,39
		2*1000	0,69	0,61	$0,08^{0,006}$	$0,18^{0,004}$	$0,40^{0,007}$	$0,49^{0,001}$	$0,65^{0,051}$	✓	0,51
	Variante 2	2*100	0,69	0,61	$0,23^{0,068}$	$0,35^{0,030}$	$0,81^{0,055}$	$0,89^{0,039}$	0,97	✓	0,89
		2*1000	0,69	0,61	$0,30^{0,037}$	$0,50^{0,002}$	$0,59^{0,044}$	$0,90^{0,038}$	1,00	✓	0,83
	Variante 3	2*100	0,69	0,61	$0,24^{0,101}$	$0,16^{0,002}$	$0,20^{0,001}$	$0,26^{0,003}$	$0,45^{0,057}$		0,30
		2*1000	0,69	0,61	$0,06^{0,005}$	$0,15^{0,002}$	$0,29^{0,002}$	$0,42^{0,007}$	0,50	✓	0,40
al2*	Variante 1	2*100	0,80	0,64	$0,21^{0,036}$	$0,43^{0,038}$	$0,30^{0,002}$	$0,31^{0,001}$	$0,42^{0,006}$		0,34
		2*1000	0,83	0,67	$0,25^{0,031}$	0,50	$0,86^{0,050}$	1,00	1,00		0,95
	Variante 2	2*100	0,83	0,67	$0,22^{0,069}$	$0,23^{0,001}$	0,32	0,32	$0,39^{0,041}$		0,34
		2*1000	0,83	0,67	$0,15^{0,009}$	$0,32^{0,001}$	0,33	0,33	0,33	✓	0,33
	Variante 3	2*100	0,83	0,67	$0,14^{0,039}$	$0,20^{0,008}$	$0,42^{0,010}$	$0,42^{0,013}$	$0,26^{0,003}$		0,37
		2*1000	0,83	0,67	$0,32^{0,040}$	0,50	0,50	0,50	$0,50^{0,002}$		0,50

Erreichte Präzision von APClustering für Änderungstyp (AP4) bei $m_c = 10$.

Netz		Pfade	rel	par	Präzision ($m_c = 4$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,34	0,03	$0,15^{0,008}$	$0,32^{0,001}$	$0,43^{0,008}$	$0,45^{0,002}$	$0,52^{0,013}$	✓	0,47
		2*1000	0,34	0,03	$0,86^{0,112}$	$0,95^{0,011}$	0,99	0,99	0,99		0,99
	Variante 2	2*100	0,31	0,03	$0,32^{0,049}$	$0,47^{0,005}$	$0,55^{0,022}$	$0,74^{0,100}$	0,99	✓	0,76
		2*1000	0,31	0,03	$0,54^{0,086}$	$0,44^{0,006}$	$0,38^{0,016}$	$0,52^{0,010}$	$0,68^{0,059}$		0,53
	Variante 3	2*100	0,34	0,03	$0,18^{0,012}$	$0,44^{0,057}$	0,33	$0,48^{0,002}$	$0,62^{0,046}$		0,48
		2*1000	0,34	0,03	$0,43^{0,089}$	0,50	0,50	$0,70^{0,060}$	$0,99^{0,005}$	✓	0,73
herbstFig6p41*	Variante 1	2*100	0,57	0,21	0,08	$0,37^{0,073}$	$0,88^{0,046}$	$0,94^{0,022}$	1,00	✓	0,94
		2*1000	0,57	0,21	$0,31^{0,050}$	0,50	$0,55^{0,022}$	$0,62^{0,045}$	1,00	✓	0,72
	Variante 2	2*100	0,49	0,18	$0,08^{0,001}$	$0,14^{0,002}$	0,20	$0,23^{0,001}$	$0,34^{0,033}$	✓	0,26
		2*1000	0,49	0,18	$0,39^{0,052}$	0,50	0,50	0,50	$0,51^{0,005}$	✓	0,50
	Variante 3	2*100	0,57	0,21	$0,10^{0,004}$	$0,21^{0,011}$	$0,46^{0,005}$	$0,44^{0,010}$	$0,49^{0,103}$		0,46
		2*1000	0,57	0,21	$0,30^{0,052}$	$0,42^{0,007}$	0,50	0,50	$0,52^{0,012}$	✓	0,51
herbstFig6p42	Variante 1	2*100	0,80	0,54	$0,08^{0,001}$	$0,21^{0,006}$	$0,16^{0,014}$	0,06	0,07		0,10
		2*1000	0,80	0,54	$0,05^{0,003}$	0,03	0,04	$0,21^{0,021}$	0,33		0,19
	Variante 2	2*100	0,80	0,54	$0,10^{0,002}$	$0,18^{0,002}$	$0,32^{0,018}$	$0,28^{0,002}$	$0,36^{0,005}$		0,32
		2*1000	0,80	0,54	$0,18^{0,012}$	0,33	0,33	0,33	$0,43^{0,010}$	✓	0,36
	Variante 3	2*100	0,80	0,54	$0,15^{0,017}$	$0,16^{0,008}$	$0,38^{0,009}$	$0,45^{0,006}$	0,50	✓	0,44
		2*1000	0,80	0,54	$0,13^{0,004}$	$0,26^{0,001}$	$0,38^{0,005}$	$0,38^{0,003}$	$0,37^{0,005}$		0,38
herbstFig6p18*	Variante 1	2*100	0,84	0,68	$0,11^{0,003}$	$0,23^{0,007}$	$0,38^{0,017}$	0,49	$0,47^{0,009}$		0,45
		2*1000	0,84	0,68	$0,13^{0,019}$	$0,27^{0,031}$	$0,76^{0,101}$	0,25	$0,30^{0,002}$		0,44
	Variante 2	2*100	0,84	0,68	$0,13^{0,004}$	$0,23^{0,003}$	$0,27^{0,002}$	$0,35^{0,004}$	$0,39^{0,006}$	✓	0,34
		2*1000	0,84	0,68	$0,11^{0,007}$	$0,23^{0,010}$	$0,15^{0,004}$	$0,19^{0,024}$	$0,26^{0,057}$		0,20
	Variante 3	2*100	0,84	0,68	$0,12^{0,003}$	$0,20^{0,015}$	$0,37^{0,009}$	$0,32^{0,002}$	$0,43^{0,006}$		0,38
		2*1000	0,84	0,68	$0,13^{0,012}$	$0,34^{0,006}$	$0,45^{0,035}$	$0,41^{0,071}$	$0,64^{0,092}$		0,50
herbstFig6p34*	Variante 1	2*100	0,90	0,80	$0,15^{0,011}$	$0,35^{0,012}$	$0,35^{0,009}$	$0,52^{0,012}$	$0,97^{0,012}$	✓	0,61
		2*1000	0,90	0,80	$0,12^{0,005}$	$0,30^{0,003}$	$0,71^{0,089}$	$0,83^{0,056}$	$0,63^{0,072}$		0,72
	Variante 2	2*100	0,90	0,80	$0,12^{0,003}$	$0,36^{0,014}$	$0,52^{0,003}$	$0,60^{0,021}$	$0,82^{0,077}$	✓	0,64
		2*1000	0,90	0,80	$0,11^{0,004}$	$0,22^{0,002}$	$0,34^{0,015}$	$0,64^{0,032}$	$0,44^{0,006}$		0,47
	Variante 3	2*100	0,90	0,80	$0,15^{0,005}$	$0,21^{0,007}$	$0,17^{0,002}$	$0,31^{0,035}$	$0,30^{0,008}$		0,26
		2*1000	0,90	0,80	$0,09^{0,002}$	$0,21^{0,006}$	$0,26^{0,012}$	$0,27^{0,018}$	$0,52^{0,100}$	✓	0,35
al2*	Variante 1	2*100	0,97	0,77	$0,15^{0,003}$	$0,22^{0,007}$	0,18	0,18	0,18		0,18
		2*1000	1,00	0,81	$0,16^{0,011}$	$0,15^{0,002}$	$0,25^{0,015}$	$0,13^{0,009}$	$0,45^{0,001}$		0,28
	Variante 2	2*100	0,89	0,61	$0,13^{0,003}$	$0,19^{0,008}$	$0,26^{0,012}$	$0,26^{0,011}$	0,24		0,25
		2*1000	1,00	0,81	$0,09^{0,003}$	$0,31^{0,031}$	$0,33^{0,001}$	$0,28^{0,002}$	$0,36^{0,004}$		0,32
	Variante 3	2*100	0,97	0,77	$0,11^{0,002}$	$0,29^{0,005}$	$0,14^{0,005}$	$0,13^{0,003}$	0,11		0,13
		2*1000	1,00	0,81	$0,10^{0,006}$	0,04	0,04	0,04	0,04		0,04

Erreichte Präzision von APClustering für Änderungstyp (AP5) bei $m_c = 4$.

Netz		Pfade	*rel*	*par*	Präzision ($m_c = 6$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,34	0,03	$0,20^{0,043}$	$0,31^{0,002}$	$0,43^{0,008}$	$0,45^{0,002}$	$0,52^{0,013}$	✓	0,47
		2*1000	0,34	0,03	$0,83^{0,118}$	$0,95^{0,011}$	0,99	0,99	0,99		0,99
	Variante 2	2*100	0,31	0,03	$0,32^{0,066}$	$0,46^{0,005}$	$0,53^{0,024}$	$0,69^{0,101}$	0,98	✓	0,74
		2*1000	0,31	0,03	$0,54^{0,086}$	0,50	$0,60^{0,041}$	$0,99^{0,004}$	1,00		0,86
	Variante 3	2*100	0,34	0,03	$0,22^{0,041}$	$0,46^{0,052}$	0,49	$0,93^{0,022}$	0,98	✓	0,80
		2*1000	0,34	0,03	$0,40^{0,091}$	0,33	0,33	$0,40^{0,007}$	$0,52^{0,008}$		0,42
herbstFig6p41*	Variante 1	2*100	0,57	0,21	$0,14^{0,045}$	$0,33^{0,052}$	$0,72^{0,063}$	$0,92^{0,032}$	1,00	✓	0,88
		2*1000	0,57	0,21	$0,49^{0,160}$	0,50	$0,56^{0,027}$	$0,62^{0,045}$	1,00	✓	0,72
	Variante 2	2*100	0,49	0,18	$0,14^{0,042}$	$0,17^{0,004}$	0,24	$0,30^{0,002}$	$0,41^{0,023}$	✓	0,32
		2*1000	0,49	0,18	$0,39^{0,056}$	0,50	0,50	0,50	$0,51^{0,007}$	✓	0,50
	Variante 3	2*100	0,57	0,21	$0,14^{0,041}$	$0,19^{0,006}$	$0,53^{0,023}$	$0,82^{0,072}$	$0,59^{0,071}$		0,64
		2*1000	0,57	0,21	$0,26^{0,027}$	$0,42^{0,007}$	0,50	$0,50^{0,001}$	$0,52^{0,012}$	✓	0,51
herbstFig6p42	Variante 1	2*100	0,80	0,54	$0,10^{0,007}$	$0,18^{0,005}$	$0,16^{0,014}$	0,06	0,07		0,10
		2*1000	0,80	0,54	$0,05^{0,003}$	0,03	0,04	$0,21^{0,021}$	0,33		0,19
	Variante 2	2*100	0,80	0,54	$0,17^{0,077}$	$0,15^{0,002}$	$0,34^{0,010}$	$0,40^{0,007}$	0,50		0,41
		2*1000	0,80	0,54	$0,22^{0,032}$	0,50	0,50	0,50	$0,73^{0,062}$	✓	0,58
	Variante 3	2*100	0,80	0,54	$0,21^{0,048}$	$0,14^{0,002}$	$0,38^{0,006}$	$0,39^{0,006}$	0,50		0,42
		2*1000	0,80	0,54	$0,12^{0,004}$	$0,25^{0,001}$	$0,38^{0,005}$	$0,38^{0,003}$	$0,36^{0,004}$		0,37
herbstFig6p18*	Variante 1	2*100	0,84	0,68	$0,10^{0,001}$	$0,24^{0,006}$	$0,37^{0,019}$	0,49	$0,47^{0,009}$		0,45
		2*1000	0,84	0,68	$0,15^{0,019}$	$0,30^{0,023}$	$0,76^{0,100}$	0,25	$0,30^{0,002}$		0,44
	Variante 2	2*100	0,84	0,68	$0,11^{0,002}$	$0,22^{0,002}$	$0,26^{0,002}$	$0,31^{0,001}$	$0,39^{0,006}$	✓	0,32
		2*1000	0,84	0,68	$0,09^{0,004}$	$0,29^{0,025}$	$0,17^{0,002}$	$0,23^{0,023}$	$0,30^{0,052}$		0,23
	Variante 3	2*100	0,84	0,68	$0,12^{0,004}$	$0,22^{0,013}$	$0,37^{0,005}$	0,33	$0,40^{0,006}$		0,37
		2*1000	0,84	0,68	$0,12^{0,011}$	$0,34^{0,006}$	$0,44^{0,029}$	$0,41^{0,068}$	$0,66^{0,081}$		0,50
herbstFig6p34*	Variante 1	2*100	0,90	0,80	$0,18^{0,041}$	$0,32^{0,008}$	$0,35^{0,009}$	0,50	$0,97^{0,012}$	✓	0,60
		2*1000	0,90	0,80	$0,11^{0,005}$	$0,30^{0,003}$	$0,71^{0,091}$	$0,77^{0,062}$	$0,58^{0,062}$		0,69
	Variante 2	2*100	0,90	0,80	$0,12^{0,003}$	$0,35^{0,014}$	$0,52^{0,003}$	$0,60^{0,021}$	$0,82^{0,077}$	✓	0,64
		2*1000	0,90	0,80	$0,10^{0,003}$	$0,22^{0,002}$	$0,34^{0,015}$	$0,64^{0,032}$	$0,44^{0,006}$		0,47
	Variante 3	2*100	0,90	0,80	$0,17^{0,038}$	$0,19^{0,003}$	$0,19^{0,002}$	$0,30^{0,025}$	$0,30^{0,008}$		0,26
		2*1000	0,90	0,80	$0,08^{0,002}$	$0,19^{0,003}$	$0,29^{0,014}$	$0,27^{0,018}$	$0,53^{0,099}$		0,36
al2*	Variante 1	2*100	0,97	0,77	$0,12^{0,002}$	$0,20^{0,006}$	0,18	0,18	0,18		0,18
		2*1000	1,00	0,81	$0,15^{0,011}$	$0,15^{0,002}$	$0,25^{0,015}$	$0,13^{0,010}$	$0,45^{0,001}$		0,28
	Variante 2	2*100	0,89	0,61	$0,14^{0,003}$	$0,31^{0,003}$	$0,30^{0,009}$	$0,26^{0,011}$	$0,35^{0,001}$		0,30
		2*1000	1,00	0,81	$0,09^{0,003}$	$0,31^{0,030}$	$0,33^{0,001}$	$0,28^{0,002}$	$0,36^{0,004}$		0,32
	Variante 3	2*100	0,97	0,77	$0,12^{0,002}$	$0,31^{0,004}$	$0,13^{0,002}$	$0,22^{0,009}$	0,11		0,15
		2*1000	1,00	0,81	$0,10^{0,005}$	0,04	0,04	0,04	0,04		0,04

Erreichte Präzision von APClustering für Änderungstyp (AP5) bei $m_c = 6$.

Netz		Pfade	*rel*	*par*	Präzision ($m_c = 10$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,34	0,03	$0,27^{0,066}$	$0,46^{0,007}$	$0,44^{0,007}$	$0,45^{0,004}$	$0,51^{0,013}$		0,47
		2*1000	0,34	0,03	$0,83^{0,113}$	$0,94^{0,011}$	1,00	1,00	1,00		1,00
	Variante 2	2*100	0,31	0,03	$0,33^{0,069}$	$0,45^{0,004}$	$0,47^{0,002}$	0,49	$0,96^{0,012}$	✓	0,64
		2*1000	0,31	0,03	$0,58^{0,101}$	0,50	$0,62^{0,045}$	$1,00^{0,001}$	1,00		0,87
	Variante 3	2*100	0,34	0,03	$0,28^{0,094}$	$0,46^{0,052}$	0,49	$0,93^{0,022}$	0,98	✓	0,80
		2*1000	0,34	0,03	$0,44^{0,083}$	0,50	0,50	$0,69^{0,059}$	1,00	✓	0,73
herbstFig6p41*	Variante 1	2*100	0,57	0,21	$0,25^{0,102}$	$0,33^{0,050}$	$0,89^{0,030}$	0,99	0,99	✓	0,96
		2*1000	0,57	0,21	$0,50^{0,162}$	0,50	$0,56^{0,027}$	$0,62^{0,045}$	1,00		0,72
	Variante 2	2*100	0,49	0,18	$0,25^{0,102}$	$0,19^{0,004}$	$0,31^{0,001}$	$0,42^{0,006}$	$0,51^{0,013}$		0,41
		2*1000	0,49	0,18	$0,64^{0,166}$	1,00	1,00	1,00	1,00	✓	1,00
	Variante 3	2*100	0,57	0,21	$0,29^{0,126}$	$0,20^{0,003}$	$0,60^{0,041}$	$0,82^{0,072}$	$0,61^{0,062}$		0,68
		2*1000	0,57	0,21	$0,30^{0,040}$	$0,76^{0,061}$	0,99	0,99	0,99	✓	0,99
herbstFig6p42	Variante 1	2*100	0,80	0,54	$0,25^{0,100}$	$0,18^{0,002}$	$0,16^{0,013}$	0,06	0,07		0,10
		2*1000	0,80	0,54	$0,06^{0,008}$	0,03	0,04	$0,21^{0,021}$	0,33		0,19
	Variante 2	2*100	0,80	0,54	$0,24^{0,103}$	$0,16^{0,004}$	$0,28^{0,013}$	$0,39^{0,008}$	0,49		0,39
		2*1000	0,80	0,54	$0,23^{0,038}$	0,50	0,50	0,50	$0,73^{0,062}$	✓	0,58
	Variante 3	2*100	0,80	0,54	$0,25^{0,071}$	$0,17^{0,003}$	$0,31^{0,004}$	$0,38^{0,006}$	0,50		0,39
		2*1000	0,80	0,54	$0,11^{0,007}$	$0,23^{0,001}$	$0,38^{0,005}$	$0,37^{0,003}$	$0,36^{0,004}$		0,37
herbstFig6p18*	Variante 1	2*100	0,84	0,68	$0,12^{0,001}$	$0,20^{0,002}$	$0,38^{0,015}$	0,49	$0,47^{0,009}$		0,45
		2*1000	0,84	0,68	$0,16^{0,020}$	$0,30^{0,020}$	$0,75^{0,106}$	0,25	$0,30^{0,002}$		0,43
	Variante 2	2*100	0,84	0,68	$0,13^{0,003}$	$0,22^{0,003}$	0,24	$0,30^{0,002}$	$0,42^{0,006}$	✓	0,32
		2*1000	0,84	0,68	$0,09^{0,003}$	$0,28^{0,023}$	$0,17^{0,002}$	$0,23^{0,014}$	$0,30^{0,052}$		0,23
	Variante 3	2*100	0,84	0,68	$0,14^{0,003}$	$0,21^{0,013}$	$0,36^{0,007}$	0,32	$0,47^{0,004}$		0,38
		2*1000	0,84	0,68	$0,10^{0,006}$	$0,33^{0,006}$	$0,42^{0,027}$	$0,41^{0,066}$	$0,66^{0,077}$		0,50
herbstFig6p34*	Variante 1	2*100	0,90	0,80	$0,25^{0,068}$	$0,28^{0,004}$	$0,35^{0,009}$	$0,45^{0,006}$	$0,96^{0,012}$	✓	0,59
		2*1000	0,90	0,80	$0,11^{0,005}$	$0,30^{0,003}$	$0,71^{0,093}$	$0,68^{0,057}$	$0,54^{0,053}$		0,64
	Variante 2	2*100	0,90	0,80	$0,19^{0,074}$	$0,29^{0,010}$	$0,45^{0,012}$	$0,43^{0,006}$	$0,81^{0,076}$		0,56
		2*1000	0,90	0,80	$0,10^{0,007}$	$0,21^{0,002}$	$0,34^{0,015}$	$0,64^{0,032}$	$0,44^{0,006}$		0,48
	Variante 3	2*100	0,90	0,80	$0,20^{0,073}$	$0,19^{0,008}$	$0,20^{0,003}$	$0,36^{0,024}$	$0,30^{0,008}$		0,29
		2*1000	0,90	0,80	$0,08^{0,007}$	$0,20^{0,006}$	$0,30^{0,012}$	$0,27^{0,017}$	$0,54^{0,094}$		0,37
al2*	Variante 1	2*100	0,97	0,77	$0,17^{0,037}$	$0,23^{0,015}$	0,18	0,18	0,18		0,18
		2*1000	1,00	0,81	$0,14^{0,011}$	$0,15^{0,002}$	$0,25^{0,013}$	$0,16^{0,013}$	$0,45^{0,001}$		0,29
	Variante 2	2*100	0,89	0,61	$0,19^{0,037}$	$0,30^{0,004}$	$0,32^{0,005}$	$0,28^{0,008}$	0,36		0,32
		2*1000	1,00	0,81	$0,13^{0,008}$	$0,38^{0,033}$	$0,32^{0,001}$	$0,28^{0,002}$	$0,36^{0,004}$		0,32
	Variante 3	2*100	0,97	0,77	$0,17^{0,039}$	$0,27^{0,003}$	$0,12^{0,006}$	$0,31^{0,023}$	0,15		0,19
		2*1000	1,00	0,81	$0,09^{0,004}$	0,04	0,04	0,04	0,04		0,04

Erreichte Präzision von APClustering für Änderungstyp (AP5) bei $m_c = 10$.

Netz		Pfade	*rel*	*par*	Präzision ($m_c = 4$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,28	0,03	$0,18^{0,010}$	$0,44^{0,006}$	0,50	0,50	$0,62^{0,046}$	✓	0,54
		2*1000	0,28	0,03	$0,85^{0,112}$	1,00	1,00	1,00	1,00		1,00
	Variante 2	2*100	0,28	0,03	$0,15^{0,006}$	$0,47^{0,004}$	0,49	0,49	$0,81^{0,055}$	✓	0,59
		2*1000	0,28	0,03	$0,48^{0,048}$	0,50	0,50	$0,85^{0,051}$	1,00	✓	0,78
	Variante 3	2*100	0,28	0,03	$0,45^{0,164}$	0,99	0,99	0,99	0,99	✓	0,99
		2*1000	0,28	0,03	$0,93^{0,059}$	1,00	1,00	1,00	1,00		1,00
herbstFig6p41*	Variante 1	2*100	0,43	0,16	$0,28^{0,076}$	0,97	0,97	0,98	0,98		0,98
		2*1000	0,43	0,16	$0,87^{0,100}$	1,00	1,00	1,00	1,00		1,00
	Variante 2	2*100	0,43	0,16	$0,09^{0,001}$	$0,21^{0,003}$	$0,42^{0,007}$	0,50	$0,52^{0,012}$	✓	0,48
		2*1000	0,43	0,16	$0,36^{0,034}$	0,50	0,50	0,50	$0,52^{0,012}$	✓	0,50
	Variante 3	2*100	0,43	0,16	$0,12^{0,004}$	$0,34^{0,009}$	$0,39^{0,006}$	0,33	$0,45^{0,038}$		0,39
		2*1000	0,43	0,16	$0,76^{0,158}$	1,00	1,00	$0,96^{0,017}$	$0,51^{0,007}$		0,82
herbstFig6p42	Variante 1	2*100	0,63	0,42	$0,11^{0,009}$	$0,23^{0,005}$	$0,42^{0,011}$	$0,48^{0,003}$	$0,57^{0,032}$	✓	0,49
		2*1000	0,63	0,42	$0,20^{0,012}$	0,33	0,33	0,33	$0,34^{0,005}$	✓	0,34
	Variante 2	2*100	0,63	0,42	$0,11^{0,006}$	$0,76^{0,105}$	0,99	1,00	1,00	✓	0,99
		2*1000	0,63	0,42	$0,59^{0,111}$	1,00	1,00	1,00	1,00		1,00
	Variante 3	2*100	0,70	0,47	$0,10^{0,003}$	$0,12^{0,007}$	0,08	0,08	0,08		0,08
		2*1000	0,70	0,47	$0,08^{0,006}$	0,02	0,02	0,01	0,00		0,01
herbstFig6p18*	Variante 1	2*100	0,67	0,50	$0,15^{0,004}$	$0,24^{0,001}$	0,16	0,16	$0,21^{0,014}$		0,18
		2*1000	0,70	0,57	$0,09^{0,001}$	$0,18^{0,001}$	$0,30^{0,002}$	0,33	$0,34^{0,003}$	✓	0,32
	Variante 2	2*100	0,70	0,57	$0,17^{0,004}$	$0,41^{0,009}$	$0,40^{0,013}$	0,49	$0,62^{0,046}$		0,50
		2*1000	0,70	0,57	$0,81^{0,130}$	1,00	1,00	1,00	1,00		1,00
	Variante 3	2*100	0,70	0,57	$0,16^{0,003}$	$0,30^{0,003}$	$0,33^{0,005}$	$0,46^{0,004}$	$0,48^{0,001}$	✓	0,42
		2*1000	0,70	0,57	$0,19^{0,025}$	$0,22^{0,006}$	$0,46^{0,060}$	0,50	$0,51^{0,006}$	✓	0,49
herbstFig6p34*	Variante 1	2*100	0,69	0,61	$0,24^{0,039}$	$0,43^{0,017}$	$0,76^{0,049}$	0,91	0,91	✓	0,86
		2*1000	0,69	0,61	$0,20^{0,011}$	$0,45^{0,007}$	$0,66^{0,058}$	1,00	1,00	✓	0,89
	Variante 2	2*100	0,69	0,61	$0,18^{0,020}$	$0,63^{0,107}$	0,99	0,99	0,99	✓	0,99
		2*1000	0,69	0,61	$0,16^{0,010}$	$0,46^{0,007}$	$0,81^{0,059}$	$1,00^{0,002}$	$0,90^{0,039}$		0,90
	Variante 3	2*100	0,69	0,61	$0,65^{0,182}$	1,00	1,00	1,00	0,99		0,99
		2*1000	0,69	0,61	$0,95^{0,041}$	1,00	1,00	1,00	1,00		1,00
al2*	Variante 1	2*100	0,83	0,67	$0,23^{0,018}$	$0,24^{0,001}$	$0,44^{0,006}$	0,49	$0,59^{0,040}$	✓	0,51
		2*1000	0,83	0,67	$0,67^{0,185}$	1,00	1,00	1,00	1,00		1,00
	Variante 2	2*100	0,83	0,67	$0,14^{0,005}$	$0,63^{0,082}$	$0,80^{0,046}$	$0,61^{0,047}$	$0,74^{0,059}$		0,72
		2*1000	0,83	0,67	$0,22^{0,013}$	0,25	$0,26^{0,001}$	0,33	$0,43^{0,053}$	✓	0,34
	Variante 3	2*100	0,83	0,67	$0,11^{0,003}$	$0,22^{0,001}$	$0,31^{0,003}$	0,50	$0,71^{0,061}$	✓	0,51
		2*1000	0,83	0,67	$0,30^{0,020}$	0,50	$0,79^{0,061}$	1,00	1,00	✓	0,93

Erreichte Präzision von APClustering für Änderungstyp (AP6) bei $m_c = 4$.

Netz		Pfade	rel	par	Präzision ($m_c = 6$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,28	0,03	$0,21^{0,041}$	$0,44^{0,006}$	0,50	0,50	$0,67^{0,055}$	✓	0,56
		2*1000	0,28	0,03	$0,85^{0,112}$	1,00	1,00	1,00	1,00		1,00
	Variante 2	2*100	0,28	0,03	$0,17^{0,007}$	$0,47^{0,004}$	0,49	0,49	$0,80^{0,054}$	✓	0,59
		2*1000	0,28	0,03	$0,48^{0,049}$	0,50	0,50	$0,85^{0,051}$	1,00	✓	0,78
	Variante 3	2*100	0,28	0,03	$0,45^{0,163}$	0,99	0,99	0,99	0,99		0,99
		2*1000	0,28	0,03	$0,93^{0,061}$	1,00	1,00	1,00	1,00		1,00
herbstFig6p41*	Variante 1	2*100	0,43	0,16	$0,27^{0,066}$	0,98	0,98	0,98	0,98	✓	0,98
		2*1000	0,43	0,16	$0,87^{0,103}$	1,00	1,00	1,00	1,00		1,00
	Variante 2	2*100	0,43	0,16	$0,17^{0,077}$	$0,18^{0,005}$	$0,42^{0,007}$	0,50	$0,52^{0,012}$	✓	0,48
		2*1000	0,43	0,16	$0,36^{0,036}$	0,50	0,50	0,50	$0,52^{0,012}$	✓	0,50
	Variante 3	2*100	0,43	0,16	$0,13^{0,005}$	$0,53^{0,045}$	$0,52^{0,012}$	0,50	$0,69^{0,058}$		0,57
		2*1000	0,43	0,16	$0,75^{0,161}$	1,00	1,00	$0,96^{0,017}$	$0,51^{0,006}$		0,82
herbstFig6p42	Variante 1	2*100	0,63	0,42	$0,15^{0,042}$	$0,23^{0,003}$	$0,37^{0,016}$	$0,48^{0,004}$	$0,57^{0,032}$	✓	0,47
		2*1000	0,63	0,42	$0,20^{0,012}$	0,33	0,33	0,33	$0,34^{0,005}$	✓	0,34
	Variante 2	2*100	0,63	0,42	$0,12^{0,003}$	$0,62^{0,098}$	0,99	1,00	0,99		0,99
		2*1000	0,63	0,42	$0,55^{0,128}$	1,00	1,00	1,00	1,00		1,00
	Variante 3	2*100	0,70	0,47	$0,15^{0,007}$	$0,22^{0,026}$	0,08	0,08	0,08		0,08
		2*1000	0,70	0,47	$0,08^{0,007}$	0,02	0,02	0,01	0,00		0,01
herbstFig6p18*	Variante 1	2*100	0,67	0,50	$0,13^{0,003}$	$0,25^{0,001}$	$0,42^{0,012}$	0,49	0,49	✓	0,47
		2*1000	0,70	0,57	$0,08^{0,001}$	$0,19^{0,001}$	$0,30^{0,002}$	0,33	$0,34^{0,003}$	✓	0,32
	Variante 2	2*100	0,70	0,57	$0,14^{0,001}$	$0,38^{0,014}$	$0,51^{0,073}$	0,49	$0,66^{0,059}$		0,55
		2*1000	0,70	0,57	$0,80^{0,138}$	0,99	0,99	0,99	0,99		0,99
	Variante 3	2*100	0,70	0,57	$0,14^{0,002}$	$0,30^{0,003}$	$0,34^{0,005}$	$0,31^{0,001}$	$0,35^{0,004}$		0,33
		2*1000	0,70	0,57	$0,17^{0,020}$	$0,27^{0,006}$	$0,63^{0,097}$	1,00	1,00	✓	0,88
herbstFig6p34*	Variante 1	2*100	0,69	0,61	$0,21^{0,019}$	$0,33^{0,016}$	$0,70^{0,064}$	0,91	0,92	✓	0,84
		2*1000	0,69	0,61	$0,18^{0,009}$	$0,43^{0,009}$	$0,62^{0,063}$	1,00	1,00	✓	0,87
	Variante 2	2*100	0,69	0,61	$0,16^{0,013}$	$0,52^{0,084}$	0,99	0,99	0,99		0,99
		2*1000	0,69	0,61	$0,15^{0,010}$	$0,45^{0,007}$	$0,79^{0,062}$	$0,51^{0,007}$	$0,49^{0,015}$		0,60
	Variante 3	2*100	0,69	0,61	$0,57^{0,152}$	0,98	0,98	0,98	0,98	✓	0,98
		2*1000	0,69	0,61	$0,95^{0,045}$	1,00	1,00	1,00	1,00		1,00
al2*	Variante 1	2*100	0,83	0,67	$0,17^{0,010}$	$0,21^{0,001}$	$0,41^{0,012}$	0,49	$0,59^{0,041}$	✓	0,50
		2*1000	0,83	0,67	$0,67^{0,190}$	$0,99^{0,002}$	1,00	1,00	1,00		1,00
	Variante 2	2*100	0,83	0,67	$0,13^{0,005}$	$0,57^{0,081}$	$0,80^{0,047}$	$0,57^{0,035}$	$0,78^{0,053}$		0,72
		2*1000	0,83	0,67	$0,21^{0,013}$	0,25	$0,26^{0,001}$	0,33	$0,43^{0,053}$	✓	0,34
	Variante 3	2*100	0,83	0,67	$0,12^{0,002}$	$0,22^{0,001}$	$0,31^{0,003}$	0,50	$0,74^{0,061}$	✓	0,51
		2*1000	0,83	0,67	$0,30^{0,021}$	0,50	$0,79^{0,061}$	1,00	1,00		0,93

Erreichte Präzision von APClustering für Änderungstyp (AP6) bei $m_c = 6$.

Netz		Pfade	*rel*	*par*	Präzision ($m_c = 10$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,28	0,03	$0,27^{0,064}$	$0,44^{0,007}$	0,50	0,50	$0,76^{0,058}$	✓	0,59
		2*1000	0,28	0,03	$0,85^{0,110}$	1,00	1,00	1,00	1,00		1,00
	Variante 2	2*100	0,28	0,03	$0,29^{0,070}$	$0,48^{0,002}$	0,49	0,49	$0,81^{0,055}$	✓	0,60
		2*1000	0,28	0,03	$0,51^{0,066}$	$0,52^{0,008}$	0,50	$0,85^{0,051}$	1,00		0,78
	Variante 3	2*100	0,28	0,03	$0,45^{0,164}$	0,99	0,99	0,99	0,98		0,99
		2*1000	0,28	0,03	$0,91^{0,074}$	1,00	1,00	1,00	1,00		1,00
herbstFig6p41*	Variante 1	2*100	0,43	0,16	$0,30^{0,062}$	0,96	0,96	0,96	0,97		0,96
		2*1000	0,43	0,16	$0,86^{0,112}$	1,00	1,00	1,00	1,00		1,00
	Variante 2	2*100	0,43	0,16	$0,26^{0,103}$	$0,18^{0,005}$	$0,42^{0,007}$	0,50	$0,52^{0,012}$		0,48
		2*1000	0,43	0,16	$0,36^{0,038}$	0,50	0,50	0,50	$0,52^{0,012}$	✓	0,50
	Variante 3	2*100	0,43	0,16	$0,27^{0,095}$	$0,32^{0,008}$	$0,44^{0,006}$	0,50	$0,81^{0,055}$	✓	0,58
		2*1000	0,43	0,16	$0,41^{0,049}$	0,50	0,50	$0,54^{0,017}$	$0,99^{0,005}$	✓	0,67
herbstFig6p42	Variante 1	2*100	0,63	0,42	$0,23^{0,071}$	$0,22^{0,003}$	$0,29^{0,004}$	0,33	$0,42^{0,052}$		0,35
		2*1000	0,63	0,42	$0,24^{0,020}$	$0,46^{0,005}$	0,50	0,50	$0,50^{0,002}$	✓	0,50
	Variante 2	2*100	0,63	0,42	$0,23^{0,070}$	$0,53^{0,105}$	0,99	0,99	0,99		0,99
		2*1000	0,63	0,42	$0,56^{0,132}$	1,00	1,00	1,00	1,00		1,00
	Variante 3	2*100	0,70	0,47	$0,37^{0,133}$	$0,23^{0,008}$	0,11	0,13	0,13		0,12
		2*1000	0,70	0,47	$0,08^{0,014}$	0,03	0,03	0,01	0,00		0,01
herbstFig6p18*	Variante 1	2*100	0,67	0,50	$0,14^{0,002}$	0,25	$0,31^{0,001}$	0,33	$0,35^{0,002}$	✓	0,33
		2*1000	0,70	0,57	$0,08^{0,002}$	$0,29^{0,006}$	$0,78^{0,065}$	1,00	1,00		0,93
	Variante 2	2*100	0,70	0,57	$0,23^{0,008}$	$0,61^{0,105}$	$0,69^{0,061}$	0,93	$0,95^{0,001}$	✓	0,86
		2*1000	0,70	0,57	$0,79^{0,143}$	0,99	0,99	0,99	0,99		0,99
	Variante 3	2*100	0,70	0,57	$0,13^{0,002}$	$0,29^{0,003}$	$0,34^{0,005}$	0,23	$0,32^{0,007}$		0,30
		2*1000	0,70	0,57	$0,13^{0,011}$	$0,23^{0,003}$	$0,41^{0,069}$	1,00	1,00	✓	0,80
herbstFig6p34*	Variante 1	2*100	0,69	0,61	$0,27^{0,068}$	$0,29^{0,011}$	$0,72^{0,049}$	$0,74^{0,043}$	0,48		0,65
		2*1000	0,69	0,61	$0,17^{0,013}$	$0,42^{0,009}$	$0,57^{0,068}$	1,00	1,00	✓	0,86
	Variante 2	2*100	0,69	0,61	$0,22^{0,068}$	$0,41^{0,067}$	0,99	0,99	0,99		0,99
		2*1000	0,69	0,61	$0,13^{0,013}$	$0,32^{0,008}$	$0,48^{0,083}$	0,25	$0,29^{0,011}$		0,34
	Variante 3	2*100	0,69	0,61	$0,59^{0,147}$	1,00	1,00	1,00	1,00		1,00
		2*1000	0,69	0,61	$0,93^{0,055}$	1,00	1,00	1,00	1,00		1,00
al2*	Variante 1	2*100	0,83	0,67	$0,16^{0,006}$	$0,18^{0,002}$	$0,29^{0,005}$	$0,45^{0,006}$	$0,59^{0,036}$	✓	0,44
		2*1000	0,83	0,67	$0,34^{0,043}$	0,50	0,50	$0,77^{0,062}$	1,00	✓	0,75
	Variante 2	2*100	0,83	0,67	$0,15^{0,005}$	$0,31^{0,011}$	0,45	0,46	$0,87^{0,029}$	✓	0,59
		2*1000	0,83	0,67	$0,20^{0,011}$	0,25	$0,26^{0,001}$	0,33	$0,44^{0,053}$	✓	0,34
	Variante 3	2*100	0,83	0,67	$0,13^{0,002}$	$0,21^{0,001}$	$0,30^{0,007}$	$0,48^{0,003}$	$0,74^{0,062}$	✓	0,50
		2*1000	0,83	0,67	$0,27^{0,025}$	0,50	$0,79^{0,060}$	1,00	1,00	✓	0,93

Erreichte Präzision von APClustering für Änderungstyp (AP6) bei $m_c = 10$.

Netz		Pfade	*rel*	*par*	Präzision ($m_c = 4$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,21	0,02	$0,20^{0,025}$	$0,48^{0,003}$	0,49	0,49	$0,83^{0,050}$	✓	0,60
		2*1000	0,21	0,02	$0,81^{0,131}$	1,00	1,00	1,00	1,00		1,00
	Variante 2	2*100	0,20	0,02	$0,13^{0,005}$	$0,48^{0,002}$	0,50	0,50	$0,49^{0,001}$		0,49
		2*1000	0,20	0,02	$0,82^{0,128}$	1,00	1,00	1,00	1,00		1,00
	Variante 3	2*100	0,21	0,02	$0,10^{0,002}$	$0,29^{0,012}$	$0,42^{0,007}$	0,50	$0,55^{0,023}$	✓	0,49
		2*1000	0,21	0,02	$0,39^{0,028}$	$0,49^{0,001}$	0,50	0,50	0,50	✓	0,50
herbstFig6p41*	Variante 1	2*100	0,33	0,13	$0,08^{0,001}$	$0,40^{0,024}$	0,49	0,49	0,49		0,49
		2*1000	0,33	0,13	$0,33^{0,049}$	0,50	0,50	0,50	0,50		0,50
	Variante 2	2*100	0,33	0,13	$0,09^{0,003}$	$0,16^{0,003}$	$0,23^{0,005}$	$0,31^{0,001}$	0,32	✓	0,29
		2*1000	0,33	0,13	$0,18^{0,016}$	0,33	0,33	0,33	0,33	✓	0,33
	Variante 3	2*100	0,33	0,13	$0,14^{0,007}$	$0,20^{0,001}$	$0,35^{0,015}$	$0,33^{0,002}$	$0,50^{0,012}$		0,39
		2*1000	0,33	0,13	$0,52^{0,105}$	$0,51^{0,006}$	0,50	0,50	0,50		0,50
herbstFig6p42	Variante 1	2*100	0,50	0,33	$0,08^{0,001}$	0,13	0,18	0,20	$0,24^{0,030}$	✓	0,21
		2*1000	0,50	0,33	$0,35^{0,041}$	0,50	0,50	0,50	$0,50^{0,002}$	✓	0,50
	Variante 2	2*100	0,50	0,33	$0,10^{0,001}$	$0,22^{0,002}$	0,24	$0,33^{0,011}$	$0,63^{0,052}$	✓	0,40
		2*1000	0,50	0,33	$0,66^{0,158}$	0,50	0,50	0,50	0,50		0,50
	Variante 3	2*100	0,50	0,33	$0,18^{0,016}$	$0,14^{0,011}$	0,04	$0,20^{0,048}$	0,49		0,24
		2*1000	0,50	0,33	$0,06^{0,006}$	0,02	$0,13^{0,023}$	$0,90^{0,042}$	1,00		0,68
herbstFig6p18*	Variante 1	2*100	0,43	0,35	$0,16^{0,011}$	$0,08^{0,003}$	0,06	0,06	$0,19^{0,039}$		0,10
		2*1000	0,43	0,35	$0,39^{0,033}$	$0,42^{0,007}$	$0,47^{0,004}$	$0,73^{0,062}$	$0,94^{0,024}$	✓	0,71
	Variante 2	2*100	0,41	0,31	$0,11^{0,002}$	$0,20^{0,002}$	0,24	0,25	$0,33^{0,078}$	✓	0,27
		2*1000	0,43	0,35	$0,32^{0,039}$	$0,49^{0,001}$	0,50	0,50	0,50	✓	0,50
	Variante 3	2*100	0,43	0,35	$0,12^{0,002}$	$0,34^{0,011}$	$0,45^{0,003}$	$0,27^{0,001}$	$0,30^{0,003}$		0,34
		2*1000	0,43	0,35	$0,36^{0,035}$	$0,48^{0,003}$	$0,48^{0,003}$	$0,50^{0,001}$	1,00		0,66
herbstFig6p34*	Variante 1	2*100	0,54	0,48	$0,29^{0,052}$	$0,77^{0,078}$	0,98	0,98	0,98		0,98
		2*1000	0,54	0,48	$0,40^{0,028}$	0,50	0,50	0,50	$0,51^{0,006}$	✓	0,50
	Variante 2	2*100	0,54	0,48	$0,26^{0,032}$	$0,17^{0,019}$	$0,27^{0,037}$	0,49	0,49		0,41
		2*1000	0,54	0,48	$0,15^{0,004}$	$0,16^{0,004}$	$0,33^{0,004}$	0,44	$0,54^{0,016}$	✓	0,44
	Variante 3	2*100	0,54	0,48	$0,15^{0,007}$	$0,46^{0,102}$	$0,96^{0,012}$	$0,94^{0,022}$	$0,72^{0,061}$		0,87
		2*1000	0,54	0,48	$0,34^{0,037}$	$0,95^{0,026}$	$0,94^{0,024}$	1,00	$0,95^{0,020}$		0,97
al2*	Variante 1	2*100	0,83	0,67	$0,15^{0,007}$	$0,29^{0,002}$	$0,28^{0,002}$	$0,31^{0,001}$	$0,42^{0,039}$		0,34
		2*1000	0,83	0,67	$0,19^{0,016}$	0,50	0,50	0,50	$0,52^{0,010}$	✓	0,51
	Variante 2	2*100	0,49	0,39	$0,13^{0,002}$	$0,28^{0,009}$	$0,31^{0,002}$	$0,42^{0,007}$	$0,44^{0,006}$	✓	0,39
		2*1000	0,51	0,41	$0,21^{0,023}$	0,50	$0,42^{0,007}$	$0,36^{0,003}$	$0,50^{0,002}$		0,43
	Variante 3	2*100	0,65	0,48	$0,15^{0,010}$	$0,33^{0,009}$	$0,33^{0,013}$	0,30	$0,32^{0,001}$		0,32
		2*1000	0,69	0,56	$0,28^{0,033}$	0,50	0,50	0,50	$0,48^{0,007}$		0,49

Erreichte Präzision von APClustering für Änderungstyp (AP7) bei $m_c = 4$.

Netz		Pfade	*rel*	*par*	Präzision ($m_c = 6$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,21	0,02	$0,23^{0,049}$	$0,48^{0,003}$	0,49	0,49	$0,54^{0,021}$	✓	0,50
		2*1000	0,21	0,02	$0,81^{0,132}$	1,00	1,00	1,00	1,00		1,00
	Variante 2	2*100	0,20	0,02	$0,12^{0,003}$	$0,44^{0,007}$	0,49	0,49	$0,48^{0,001}$		0,48
		2*1000	0,20	0,02	$0,82^{0,128}$	1,00	1,00	1,00	1,00		1,00
	Variante 3	2*100	0,21	0,02	$0,13^{0,007}$	$0,28^{0,009}$	$0,36^{0,004}$	$0,43^{0,007}$	$0,55^{0,023}$	✓	0,45
		2*1000	0,21	0,02	$0,39^{0,028}$	$0,49^{0,001}$	0,50	0,50	0,50		0,50
herbstFig6p41*	Variante 1	2*100	0,33	0,13	$0,13^{0,042}$	$0,41^{0,019}$	0,48	0,48	$0,59^{0,041}$	✓	0,51
		2*1000	0,33	0,13	$0,60^{0,200}$	1,00	1,00	1,00	0,99		0,99
	Variante 2	2*100	0,33	0,13	$0,14^{0,042}$	$0,16^{0,002}$	$0,24^{0,004}$	$0,31^{0,001}$	0,32	✓	0,29
		2*1000	0,33	0,13	$0,22^{0,029}$	0,50	0,50	0,50	$0,50^{0,001}$	✓	0,50
	Variante 3	2*100	0,33	0,13	$0,13^{0,005}$	$0,20^{0,002}$	$0,35^{0,015}$	$0,33^{0,002}$	$0,55^{0,032}$		0,41
		2*1000	0,33	0,13	$0,52^{0,108}$	$0,51^{0,006}$	0,50	0,50	0,50		0,50
herbstFig6p42	Variante 1	2*100	0,50	0,33	$0,15^{0,045}$	0,12	0,15	0,19	$0,24^{0,030}$		0,19
		2*1000	0,50	0,33	$0,68^{0,194}$	1,00	1,00	1,00	1,00		1,00
	Variante 2	2*100	0,50	0,33	$0,12^{0,006}$	$0,32^{0,022}$	$0,35^{0,002}$	0,33	$0,44^{0,036}$		0,37
		2*1000	0,50	0,33	$0,65^{0,164}$	0,50	0,50	0,50	0,50		0,50
	Variante 3	2*100	0,50	0,33	$0,22^{0,041}$	$0,14^{0,010}$	0,04	$0,20^{0,048}$	0,49		0,24
		2*1000	0,50	0,33	$0,06^{0,006}$	0,02	$0,13^{0,023}$	$0,90^{0,042}$	1,00		0,68
herbstFig6p18*	Variante 1	2*100	0,43	0,35	$0,13^{0,004}$	$0,08^{0,002}$	0,06	0,06	$0,23^{0,044}$		0,12
		2*1000	0,43	0,35	$0,38^{0,033}$	$0,42^{0,007}$	$0,47^{0,004}$	$0,73^{0,062}$	$0,94^{0,026}$	✓	0,71
	Variante 2	2*100	0,41	0,31	$0,10^{0,001}$	$0,19^{0,001}$	0,24	0,25	$0,33^{0,078}$	✓	0,27
		2*1000	0,43	0,35	$0,31^{0,040}$	$0,49^{0,001}$	0,50	0,50	0,50		0,50
	Variante 3	2*100	0,43	0,35	$0,10^{0,001}$	$0,28^{0,009}$	$0,44^{0,007}$	0,25	$0,30^{0,004}$		0,33
		2*1000	0,43	0,35	$0,35^{0,035}$	$0,48^{0,003}$	$0,47^{0,004}$	$0,50^{0,001}$	1,00		0,66
herbstFig6p34*	Variante 1	2*100	0,54	0,48	$0,23^{0,031}$	$0,77^{0,087}$	0,98	0,98	0,98		0,98
		2*1000	0,54	0,48	$0,72^{0,147}$	1,00	1,00	1,00	1,00		1,00
	Variante 2	2*100	0,54	0,48	$0,18^{0,011}$	$0,20^{0,028}$	$0,27^{0,039}$	0,49	0,49	✓	0,41
		2*1000	0,54	0,48	$0,15^{0,004}$	$0,16^{0,003}$	$0,33^{0,004}$	0,44	$0,54^{0,016}$	✓	0,44
	Variante 3	2*100	0,54	0,48	$0,18^{0,011}$	$0,30^{0,014}$	$0,52^{0,012}$	$0,77^{0,061}$	$0,96^{0,012}$	✓	0,75
		2*1000	0,54	0,48	$0,28^{0,029}$	$0,94^{0,029}$	$0,92^{0,033}$	1,00	$0,93^{0,031}$		0,95
al2*	Variante 1	2*100	0,83	0,67	$0,15^{0,007}$	$0,28^{0,002}$	$0,29^{0,002}$	0,25	$0,38^{0,049}$		0,30
		2*1000	0,83	0,67	$0,24^{0,034}$	$0,99^{0,006}$	1,00	1,00	$0,99^{0,005}$		0,99
	Variante 2	2*100	0,49	0,39	$0,11^{0,002}$	$0,24^{0,004}$	$0,28^{0,003}$	$0,42^{0,007}$	$0,43^{0,007}$	✓	0,37
		2*1000	0,51	0,41	$0,22^{0,024}$	0,50	$0,42^{0,007}$	$0,36^{0,003}$	$0,50^{0,002}$		0,43
	Variante 3	2*100	0,65	0,48	$0,13^{0,004}$	$0,28^{0,006}$	$0,31^{0,011}$	0,30	$0,32^{0,001}$		0,31
		2*1000	0,69	0,56	$0,22^{0,015}$	0,50	0,50	0,50	$0,48^{0,007}$		0,49

Erreichte Präzision von APClustering für Änderungstyp (AP7) bei $m_c = 6$.

Netz		Pfade	*rel*	*par*	Präzision ($m_c = 10$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,21	0,02	$0,26^{0,069}$	$0,38^{0,005}$	0,49	0,49	$0,53^{0,018}$	✓	0,50
		2*1000	0,21	0,02	$0,81^{0,134}$	1,00	1,00	1,00	1,00		1,00
	Variante 2	2*100	0,20	0,02	$0,22^{0,071}$	$0,79^{0,089}$	0,96	0,96	$0,94^{0,012}$		0,95
		2*1000	0,20	0,02	$0,79^{0,133}$	1,00	1,00	1,00	1,00		1,00
	Variante 3	2*100	0,21	0,02	$0,23^{0,071}$	$0,31^{0,009}$	$0,36^{0,004}$	$0,43^{0,007}$	$0,55^{0,023}$	✓	0,45
		2*1000	0,21	0,02	$0,73^{0,145}$	$0,98^{0,008}$	1,00	1,00	1,00		1,00
herbstFig6p41*	Variante 1	2*100	0,33	0,13	$0,27^{0,100}$	$0,73^{0,112}$	0,93	0,93	0,92		0,92
		2*1000	0,33	0,13	$0,60^{0,200}$	0,99	0,99	0,99	0,99		0,99
	Variante 2	2*100	0,33	0,13	$0,31^{0,120}$	$0,15^{0,002}$	$0,29^{0,003}$	0,32	0,32		0,31
		2*1000	0,33	0,13	$0,42^{0,149}$	0,99	0,99	0,99	0,99		0,99
	Variante 3	2*100	0,33	0,13	$0,19^{0,042}$	$0,20^{0,001}$	$0,35^{0,008}$	$0,33^{0,003}$	$0,63^{0,054}$		0,43
		2*1000	0,33	0,13	$0,44^{0,120}$	$0,37^{0,013}$	0,33	0,33	0,33		0,33
herbstFig6p42	Variante 1	2*100	0,50	0,33	$0,28^{0,096}$	0,15	$0,16^{0,001}$	$0,21^{0,001}$	$0,27^{0,025}$		0,22
		2*1000	0,50	0,33	$0,62^{0,191}$	1,00	1,00	1,00	1,00		1,00
	Variante 2	2*100	0,50	0,33	$0,24^{0,102}$	$0,29^{0,026}$	0,50	0,50	$0,54^{0,020}$		0,51
		2*1000	0,50	0,33	$0,41^{0,070}$	0,99	0,99	0,99	0,99		0,99
	Variante 3	2*100	0,50	0,33	$0,27^{0,070}$	$0,16^{0,007}$	$0,06^{0,001}$	$0,14^{0,009}$	$0,35^{0,003}$		0,18
		2*1000	0,50	0,33	$0,07^{0,011}$	0,02	$0,20^{0,054}$	$0,91^{0,037}$	1,00		0,70
herbstFig6p18*	Variante 1	2*100	0,43	0,35	$0,12^{0,001}$	$0,31^{0,018}$	$0,09^{0,005}$	0,06	$0,32^{0,044}$		0,16
		2*1000	0,43	0,35	$0,37^{0,036}$	$0,37^{0,016}$	$0,41^{0,010}$	$0,73^{0,062}$	$0,91^{0,036}$	✓	0,68
	Variante 2	2*100	0,41	0,31	$0,11^{0,001}$	0,16	0,19	0,17	$0,33^{0,078}$		0,23
		2*1000	0,43	0,35	$0,19^{0,015}$	$0,31^{0,001}$	0,33	$0,39^{0,006}$	0,50	✓	0,41
	Variante 3	2*100	0,43	0,35	$0,13^{0,002}$	$0,31^{0,017}$	$0,46^{0,003}$	0,32	$0,36^{0,006}$		0,38
		2*1000	0,43	0,35	$0,26^{0,022}$	$0,38^{0,006}$	$0,46^{0,005}$	$0,50^{0,001}$	1,00	✓	0,65
herbstFig6p34*	Variante 1	2*100	0,54	0,48	$0,23^{0,037}$	$0,72^{0,086}$	0,98	0,98	0,98		0,98
		2*1000	0,54	0,48	$0,68^{0,158}$	1,00	1,00	1,00	1,00	✓	1,00
	Variante 2	2*100	0,54	0,48	$0,24^{0,039}$	$0,22^{0,033}$	$0,27^{0,031}$	$0,40^{0,006}$	0,49		0,38
		2*1000	0,54	0,48	$0,16^{0,009}$	$0,15^{0,003}$	$0,33^{0,004}$	0,44	$0,54^{0,016}$		0,44
	Variante 3	2*100	0,54	0,48	$0,24^{0,069}$	$0,30^{0,039}$	$0,72^{0,087}$	$0,73^{0,072}$	0,99	✓	0,81
		2*1000	0,54	0,48	$0,18^{0,012}$	$0,86^{0,058}$	$0,90^{0,037}$	1,00	$0,89^{0,041}$		0,93
al2*	Variante 1	2*100	0,83	0,67	$0,20^{0,040}$	$0,25^{0,001}$	$0,30^{0,001}$	$0,24^{0,001}$	$0,38^{0,051}$		0,31
		2*1000	0,83	0,67	$0,20^{0,028}$	$0,98^{0,008}$	1,00	$1,00^{0,001}$	$0,99^{0,002}$		1,00
	Variante 2	2*100	0,49	0,39	$0,10^{0,001}$	$0,19^{0,002}$	$0,21^{0,001}$	$0,29^{0,002}$	$0,41^{0,006}$	✓	0,30
		2*1000	0,51	0,41	$0,21^{0,025}$	0,50	$0,42^{0,007}$	$0,36^{0,003}$	$0,50^{0,002}$		0,43
	Variante 3	2*100	0,65	0,48	$0,18^{0,039}$	$0,21^{0,001}$	$0,29^{0,001}$	$0,47^{0,001}$	$0,47^{0,001}$	✓	0,41
		2*1000	0,69	0,56	$0,18^{0,009}$	$0,47^{0,004}$	0,50	0,50	$0,48^{0,007}$		0,49

Erreichte Präzision von APClustering für Änderungstyp (AP7) bei $m_c = 10$.

Netz		Pfade	*rel*	*par*	Präzision ($m_c = 4$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,34	0,03	$0,15^{0,008}$	$0,08$	$0,07$	$0,07$	$0,07$		0,07
		2*1000	0,34	0,03	$0,43^{0,108}$	$0,99^{0,005}$	$1,00$	$1,00$	$1,00$	✓	1,00
	Variante 2	2*100	0,34	0,03	$0,10^{0,003}$	$0,17^{0,008}$	$0,42^{0,001}$	$0,71^{0,068}$	$0,93^{0,022}$	✓	0,69
		2*1000	0,34	0,03	$0,18^{0,007}$	$0,51^{0,018}$	$0,67^{0,055}$	$1,00$	$1,00$	✓	0,89
	Variante 3	2*100	0,34	0,03	$0,12^{0,001}$	$0,19^{0,001}$	$0,22$	$0,16^{0,001}$	$0,17$		0,18
		2*1000	0,34	0,03	$0,05^{0,011}$	$0,12^{0,006}$	$0,22^{0,001}$	$0,25$	$0,25$	✓	0,24
herbstFig6p41*	Variante 1	2*100	0,57	0,21	$0,12^{0,005}$	$0,22^{0,009}$	$0,46^{0,001}$	$0,38^{0,012}$	$0,52^{0,020}$		0,45
		2*1000	0,57	0,21	$0,11^{0,005}$	$0,33^{0,022}$	$0,78^{0,076}$	$1,00$	$0,88^{0,075}$		0,89
	Variante 2	2*100	0,57	0,21	$0,10^{0,002}$	$0,20^{0,004}$	$0,26^{0,009}$	$0,47$	$0,47$	✓	0,40
		2*1000	0,57	0,21	$0,58^{0,183}$	$1,00$	$1,00$	$1,00$	$1,00$	✓	1,00
	Variante 3	2*100	0,57	0,21	$0,11^{0,004}$	$0,17^{0,002}$	$0,33^{0,006}$	$0,31$	$0,41^{0,005}$		0,35
		2*1000	0,57	0,21	$0,11^{0,003}$	$0,22^{0,001}$	$0,25$	$0,32^{0,001}$	$0,38^{0,006}$	✓	0,31
herbstFig6p42	Variante 1	2*100	0,80	0,54	$0,10^{0,003}$	$0,30^{0,003}$	$0,19^{0,003}$	$0,13$	$0,13$		0,15
		2*1000	0,80	0,54	$0,17^{0,017}$	$0,18^{0,010}$	$0,02$	$0,02$	$0,19^{0,045}$		0,08
	Variante 2	2*100	0,80	0,54	$0,09^{0,002}$	$0,20^{0,003}$	$0,25^{0,002}$	$0,33^{0,007}$	$0,24$		0,27
		2*1000	0,80	0,54	$0,19^{0,011}$	$0,39^{0,013}$	$0,92^{0,033}$	$0,92^{0,032}$	$0,93^{0,030}$	✓	0,92
	Variante 3	2*100	0,80	0,54	$0,09^{0,001}$	$0,19^{0,005}$	$0,27^{0,002}$	$0,43^{0,007}$	$0,47^{0,009}$	✓	0,39
		2*1000	0,80	0,54	$0,12^{0,004}$	$0,19^{0,002}$	$0,37^{0,011}$	$0,89^{0,042}$	$1,00$	✓	0,76
herbstFig6p18*	Variante 1	2*100	0,84	0,68	$0,13^{0,003}$	$0,20^{0,006}$	$0,09^{0,001}$	$0,28^{0,025}$	$0,29^{0,001}$		0,22
		2*1000	0,84	0,68	$0,08^{0,006}$	$0,02$	$0,02$	$0,02$	$0,02$		0,02
	Variante 2	2*100	0,84	0,68	$0,11^{0,002}$	$0,19^{0,006}$	$0,28^{0,002}$	$0,28^{0,005}$	$0,30^{0,001}$		0,29
		2*1000	0,84	0,68	$0,19^{0,010}$	$0,52^{0,038}$	$0,75^{0,055}$	$0,95$	$0,89^{0,054}$		0,86
	Variante 3	2*100	0,84	0,68	$0,10^{0,001}$	$0,13^{0,001}$	$0,20^{0,001}$	$0,27^{0,007}$	$0,21$		0,23
		2*1000	0,84	0,68	$0,13^{0,005}$	$0,48^{0,002}$	$0,98^{0,008}$	$0,99$	$0,99$	✓	0,99
herbstFig6p34*	Variante 1	2*100	0,90	0,80	$0,24^{0,018}$	$0,18^{0,007}$	$0,22^{0,001}$	$0,40^{0,010}$	$0,47$		0,36
		2*1000	0,90	0,80	$0,12^{0,007}$	$0,21^{0,001}$	$0,25^{0,001}$	$0,33^{0,004}$	$0,54^{0,030}$	✓	0,37
	Variante 2	2*100	0,90	0,80	$0,22^{0,035}$	$0,23^{0,014}$	$0,27^{0,014}$	$0,24^{0,006}$	$0,17$		0,23
		2*1000	0,90	0,80	$0,08^{0,004}$	$0,23^{0,011}$	$0,45^{0,007}$	$0,38^{0,148}$	$0,00$		0,28
	Variante 3	2*100	0,90	0,80	$0,12^{0,004}$	$0,17^{0,002}$	$0,41^{0,056}$	$0,42^{0,006}$	$0,51^{0,012}$	✓	0,44
		2*1000	0,90	0,80	$0,08^{0,006}$	$0,24^{0,007}$	$0,41^{0,014}$	$0,58^{0,025}$	$0,79^{0,060}$	✓	0,59
al2*	Variante 1	2*100	1,00	0,81	$0,12^{0,002}$	$0,18^{0,004}$	$0,21^{0,003}$	$0,15^{0,004}$	$0,17^{0,011}$		0,18
		2*1000	1,00	0,81	$0,16^{0,009}$	$0,32^{0,004}$	$0,28^{0,021}$	$0,18^{0,024}$	$0,06$		0,17
	Variante 2	2*100	1,00	0,81	$0,16^{0,008}$	$0,18^{0,001}$	$0,29^{0,014}$	$0,22^{0,001}$	$0,27^{0,001}$		0,26
		2*1000	1,00	0,81	$0,11^{0,003}$	$0,12^{0,001}$	$0,25^{0,003}$	$0,21^{0,005}$	$0,27^{0,001}$		0,24
	Variante 3	2*100	0,87	0,66	$0,14^{0,011}$	$0,11$	$0,18^{0,006}$	$0,12^{0,008}$	$0,27^{0,029}$		0,19
		2*1000	0,91	0,74	$0,07^{0,006}$	$0,11^{0,009}$	$0,26^{0,001}$	$0,30^{0,005}$	$0,45^{0,001}$	✓	0,34

Erreichte Präzision von APClustering für Änderungstyp (AP8) bei $m_c = 4$.

Netz		Pfade	rel	par	Präzision ($m_c = 6$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,34	0,03	$0,19^{0,041}$	$0,08^{0,001}$	0,07	0,07	0,07		0,07
		2*1000	0,34	0,03	$0,41^{0,100}$	$0,97^{0,012}$	1,00	1,00	1,00	✓	1,00
	Variante 2	2*100	0,34	0,03	$0,13^{0,009}$	$0,22^{0,004}$	$0,78^{0,015}$	$0,73^{0,064}$	$0,93^{0,022}$		0,81
		2*1000	0,34	0,03	$0,20^{0,011}$	$0,63^{0,008}$	$0,67^{0,055}$	1,00	1,00	✓	0,89
	Variante 3	2*100	0,34	0,03	$0,14^{0,003}$	0,17	0,20	$0,15^{0,001}$	0,17		0,17
		2*1000	0,34	0,03	$0,06^{0,017}$	$0,27^{0,022}$	$0,21^{0,001}$	0,25	0,25		0,24
herbstFig6p41*	Variante 1	2*100	0,57	0,21	$0,16^{0,043}$	$0,20^{0,007}$	0,47	$0,38^{0,012}$	$0,52^{0,020}$		0,46
		2*1000	0,57	0,21	$0,15^{0,015}$	$0,34^{0,007}$	$0,45^{0,006}$	0,50	$0,47^{0,004}$		0,47
	Variante 2	2*100	0,57	0,21	$0,14^{0,040}$	$0,19^{0,003}$	$0,26^{0,009}$	0,47	0,47	✓	0,40
		2*1000	0,57	0,21	$0,57^{0,185}$	1,00	1,00	1,00	1,00	✓	1,00
	Variante 3	2*100	0,57	0,21	$0,16^{0,045}$	$0,19^{0,004}$	$0,33^{0,006}$	0,31	$0,41^{0,005}$		0,35
		2*1000	0,57	0,21	$0,09^{0,002}$	$0,21^{0,002}$	0,25	$0,32^{0,001}$	$0,38^{0,006}$	✓	0,31
herbstFig6p42	Variante 1	2*100	0,80	0,54	$0,15^{0,046}$	$0,20^{0,003}$	$0,23^{0,001}$	$0,16^{0,002}$	0,13		0,17
		2*1000	0,80	0,54	$0,16^{0,015}$	$0,18^{0,009}$	0,02	0,02	$0,19^{0,045}$		0,08
	Variante 2	2*100	0,80	0,54	$0,14^{0,042}$	$0,20^{0,006}$	$0,19^{0,001}$	$0,25^{0,002}$	0,24		0,23
		2*1000	0,80	0,54	$0,17^{0,010}$	$0,26^{0,007}$	$0,90^{0,039}$	$0,92^{0,032}$	$0,93^{0,030}$	✓	0,91
	Variante 3	2*100	0,80	0,54	$0,15^{0,041}$	$0,18^{0,006}$	$0,30^{0,002}$	$0,42^{0,006}$	0,49	✓	0,41
		2*1000	0,80	0,54	$0,10^{0,003}$	$0,19^{0,002}$	$0,37^{0,011}$	$0,89^{0,042}$	1,00	✓	0,76
herbstFig6p18*	Variante 1	2*100	0,84	0,68	$0,11^{0,001}$	$0,16^{0,003}$	$0,19^{0,008}$	$0,33^{0,014}$	$0,29^{0,002}$		0,27
		2*1000	0,84	0,68	$0,08^{0,005}$	0,02	0,02	0,02	0,02		0,02
	Variante 2	2*100	0,84	0,68	$0,08^{0,001}$	$0,20^{0,006}$	$0,29^{0,002}$	$0,31^{0,011}$	$0,30^{0,001}$		0,30
		2*1000	0,84	0,68	$0,18^{0,011}$	$0,48^{0,030}$	$0,72^{0,057}$	0,95	$0,89^{0,054}$		0,85
	Variante 3	2*100	0,84	0,68	$0,10^{0,001}$	0,13	$0,21^{0,004}$	$0,32^{0,007}$	0,21		0,25
		2*1000	0,84	0,68	$0,11^{0,005}$	$0,48^{0,003}$	$0,98^{0,008}$	0,99	0,99	✓	0,99
herbstFig6p34*	Variante 1	2*100	0,90	0,80	$0,19^{0,038}$	0,20	$0,28^{0,002}$	$0,42^{0,004}$	0,47	✓	0,39
		2*1000	0,90	0,80	$0,11^{0,004}$	$0,23^{0,002}$	$0,26^{0,002}$	$0,31^{0,001}$	$0,37^{0,008}$	✓	0,31
	Variante 2	2*100	0,90	0,80	$0,21^{0,013}$	$0,15^{0,001}$	$0,27^{0,014}$	$0,24^{0,006}$	0,17		0,23
		2*1000	0,90	0,80	$0,07^{0,004}$	$0,20^{0,009}$	$0,41^{0,012}$	$0,38^{0,148}$	0,00		0,26
	Variante 3	2*100	0,90	0,80	$0,13^{0,007}$	$0,17^{0,002}$	$0,34^{0,025}$	$0,42^{0,006}$	$0,56^{0,032}$	✓	0,44
		2*1000	0,90	0,80	$0,07^{0,002}$	$0,23^{0,007}$	$0,41^{0,014}$	$0,58^{0,025}$	$0,79^{0,060}$	✓	0,59
al2*	Variante 1	2*100	1,00	0,81	$0,11^{0,002}$	$0,21^{0,001}$	$0,21^{0,003}$	$0,17^{0,007}$	$0,17^{0,011}$		0,19
		2*1000	1,00	0,81	$0,15^{0,009}$	$0,32^{0,004}$	$0,28^{0,020}$	$0,18^{0,024}$	0,06		0,18
	Variante 2	2*100	1,00	0,81	$0,13^{0,005}$	$0,18^{0,001}$	$0,29^{0,009}$	$0,23^{0,001}$	$0,27^{0,001}$		0,26
		2*1000	1,00	0,81	$0,10^{0,003}$	$0,12^{0,001}$	$0,25^{0,003}$	$0,21^{0,005}$	$0,27^{0,001}$		0,24
	Variante 3	2*100	0,87	0,66	$0,13^{0,007}$	$0,17^{0,002}$	$0,25^{0,002}$	$0,15^{0,010}$	$0,28^{0,033}$		0,23
		2*1000	0,91	0,74	$0,07^{0,007}$	$0,11^{0,009}$	$0,26^{0,001}$	$0,31^{0,005}$	$0,45^{0,001}$	✓	0,34

Erreichte Präzision von APClustering für Änderungstyp (AP8) bei $m_c = 6$.

Netz		Pfade	*rel*	*par*	Präzision ($m_c = 10$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,34	0,03	$0,18^{0,042}$	$0,18^{0,016}$	0,07	0,07	0,07		0,07
		2*1000	0,34	0,03	$0,38^{0,091}$	$0,95^{0,021}$	1,00	1,00	1,00	✓	1,00
	Variante 2	2*100	0,34	0,03	$0,25^{0,100}$	$0,24^{0,003}$	$0,78^{0,015}$	$0,73^{0,064}$	$0,93^{0,022}$		0,81
		2*1000	0,34	0,03	$0,19^{0,017}$	$0,63^{0,008}$	$0,67^{0,055}$	1,00	1,00	✓	0,89
	Variante 3	2*100	0,34	0,03	$0,18^{0,002}$	0,23	0,17	0,20	0,21		0,19
		2*1000	0,34	0,03	$0,05^{0,009}$	$0,18^{0,001}$	$0,21^{0,001}$	0,25	0,25	✓	0,24
herbstFig6p41*	Variante 1	2*100	0,57	0,21	$0,25^{0,104}$	$0,20^{0,007}$	0,47	$0,40^{0,006}$	$0,67^{0,061}$		0,52
		2*1000	0,57	0,21	$0,15^{0,021}$	$0,40^{0,013}$	$0,80^{0,059}$	1,00	$0,92^{0,034}$		0,90
	Variante 2	2*100	0,57	0,21	$0,24^{0,104}$	$0,18^{0,003}$	$0,24^{0,011}$	0,47	0,47		0,39
		2*1000	0,57	0,21	$0,59^{0,184}$	1,00	1,00	1,00	1,00	✓	1,00
	Variante 3	2*100	0,57	0,21	$0,24^{0,103}$	$0,15^{0,001}$	$0,32^{0,010}$	0,30	$0,40^{0,006}$		0,34
		2*1000	0,57	0,21	$0,11^{0,007}$	$0,23^{0,001}$	0,32	0,33	$0,38^{0,006}$	✓	0,34
herbstFig6p42	Variante 1	2*100	0,80	0,54	$0,29^{0,127}$	$0,16^{0,002}$	0,23	$0,16^{0,002}$	0,13		0,17
		2*1000	0,80	0,54	$0,15^{0,018}$	$0,18^{0,009}$	0,02	0,02	$0,19^{0,045}$		0,08
	Variante 2	2*100	0,80	0,54	$0,28^{0,129}$	$0,15^{0,002}$	0,18	$0,24^{0,002}$	0,19		0,20
		2*1000	0,80	0,54	$0,15^{0,012}$	$0,26^{0,007}$	$0,90^{0,039}$	$0,92^{0,032}$	$0,93^{0,030}$	✓	0,91
	Variante 3	2*100	0,80	0,54	$0,27^{0,129}$	$0,15^{0,005}$	$0,29^{0,002}$	$0,42^{0,006}$	0,49		0,40
		2*1000	0,80	0,54	$0,10^{0,007}$	$0,21^{0,002}$	$0,37^{0,011}$	$0,89^{0,042}$	1,00	✓	0,76
herbstFig6p18*	Variante 1	2*100	0,84	0,68	$0,13^{0,004}$	$0,15^{0,002}$	$0,24^{0,009}$	$0,41^{0,002}$	0,31		0,32
		2*1000	0,84	0,68	$0,08^{0,005}$	0,02	0,02	0,02	0,02		0,02
	Variante 2	2*100	0,84	0,68	$0,16^{0,005}$	$0,30^{0,015}$	0,50	$0,39^{0,010}$	$0,45^{0,003}$		0,44
		2*1000	0,84	0,68	$0,15^{0,008}$	$0,42^{0,026}$	$0,69^{0,056}$	0,95	$0,89^{0,054}$		0,84
	Variante 3	2*100	0,84	0,68	$0,11^{0,001}$	$0,18^{0,002}$	$0,32^{0,013}$	$0,32^{0,007}$	0,21		0,28
		2*1000	0,84	0,68	$0,09^{0,003}$	$0,45^{0,007}$	$0,98^{0,008}$	0,99	0,99	✓	0,99
herbstFig6p34*	Variante 1	2*100	0,90	0,80	$0,22^{0,070}$	0,18	$0,27^{0,001}$	0,29	$0,33^{0,003}$		0,30
		2*1000	0,90	0,80	$0,10^{0,007}$	$0,22^{0,001}$	$0,25^{0,002}$	$0,31^{0,001}$	$0,38^{0,015}$	✓	0,31
	Variante 2	2*100	0,90	0,80	$0,27^{0,065}$	$0,13^{0,001}$	$0,29^{0,010}$	$0,41^{0,006}$	$0,36^{0,023}$		0,35
		2*1000	0,90	0,80	$0,07^{0,005}$	$0,20^{0,010}$	$0,41^{0,011}$	$0,38^{0,148}$	0,00		0,26
	Variante 3	2*100	0,90	0,80	$0,19^{0,073}$	$0,18^{0,003}$	$0,30^{0,005}$	$0,41^{0,005}$	$0,64^{0,056}$		0,45
		2*1000	0,90	0,80	$0,07^{0,006}$	$0,22^{0,007}$	$0,41^{0,014}$	$0,58^{0,025}$	$0,79^{0,060}$	✓	0,59
al2*	Variante 1	2*100	1,00	0,81	$0,16^{0,038}$	$0,17^{0,001}$	$0,23^{0,003}$	$0,18^{0,008}$	$0,17^{0,011}$		0,20
		2*1000	1,00	0,81	$0,14^{0,008}$	$0,30^{0,004}$	$0,29^{0,019}$	$0,19^{0,023}$	0,06		0,18
	Variante 2	2*100	1,00	0,81	$0,20^{0,072}$	$0,23^{0,004}$	$0,28^{0,002}$	$0,36^{0,004}$	0,49	✓	0,38
		2*1000	1,00	0,81	$0,09^{0,002}$	$0,11^{0,001}$	$0,25^{0,003}$	$0,21^{0,005}$	$0,27^{0,001}$		0,24
	Variante 3	2*100	0,87	0,66	$0,14^{0,003}$	0,22	0,30	$0,17^{0,015}$	$0,34^{0,031}$		0,27
		2*1000	0,91	0,74	$0,10^{0,005}$	$0,10^{0,008}$	$0,25^{0,001}$	$0,31^{0,005}$	$0,45^{0,001}$	✓	0,34

Erreichte Präzision von APClustering für Änderungstyp (AP8) bei $m_c = 10$.

Netz		Pfade	*rel*	*par*	Präzision ($m_c = 4$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,34	0,03	$0,11^{0,004}$	0,21	$0,24^{0,007}$	$0,46^{0,121}$	0,20		0,30
		2*1000	0,34	0,03	$0,81^{0,134}$	1,00	1,00	1,00	$0,74^{0,190}$		0,91
	Variante 2	2*100	0,34	0,03	$0,12^{0,005}$	$0,44^{0,028}$	$0,74^{0,061}$	0,99	0,99	✓	0,91
		2*1000	0,34	0,03	$0,46^{0,073}$	0,50	0,50	$0,54^{0,020}$	$0,57^{0,032}$	✓	0,54
	Variante 3	2*100	0,34	0,03	$0,14^{0,006}$	$0,39^{0,012}$	$0,17^{0,002}$	$0,31^{0,009}$	0,35		0,28
		2*1000	0,34	0,03	$0,09^{0,018}$	$0,13^{0,114}$	0,00	0,00	$0,54^{0,243}$		0,18
herbstFig6p41*	Variante 1	2*100	0,43	0,16	$0,08^{0,001}$	$0,14^{0,003}$	$0,28^{0,002}$	$0,43^{0,005}$	0,47	✓	0,39
		2*1000	0,43	0,16	$0,14^{0,008}$	$0,40^{0,036}$	$0,32^{0,011}$	$0,32^{0,001}$	0,33		0,32
	Variante 2	2*100	0,57	0,21	$0,08^{0,001}$	$0,20^{0,013}$	$0,45^{0,053}$	$0,79^{0,055}$	$0,97^{0,001}$	✓	0,74
		2*1000	0,57	0,21	$0,30^{0,108}$	$0,87^{0,045}$	0,99	0,99	0,99	✓	0,99
	Variante 3	2*100	0,49	0,18	$0,08^{0,002}$	$0,17^{0,006}$	$0,43^{0,004}$	0,91	0,91		0,75
		2*1000	0,49	0,18	$0,18^{0,031}$	$0,29^{0,060}$	$0,24^{0,033}$	$0,33^{0,002}$	0,33		0,30
herbstFig6p42	Variante 1	2*100	0,63	0,42	$0,10^{0,003}$	$0,26^{0,011}$	$0,34^{0,009}$	$0,33^{0,004}$	$0,39^{0,008}$		0,36
		2*1000	0,63	0,42	$0,28^{0,018}$	0,50	$0,68^{0,058}$	0,99	0,99	✓	0,89
	Variante 2	2*100	0,70	0,47	$0,08^{0,001}$	$0,15^{0,004}$	$0,12^{0,015}$	$0,17^{0,005}$	0,12		0,14
		2*1000	0,70	0,47	$0,19^{0,017}$	$0,35^{0,004}$	0,49	0,49	$0,64^{0,051}$	✓	0,54
	Variante 3	2*100	0,63	0,42	$0,14^{0,010}$	$0,30^{0,013}$	$0,69^{0,089}$	0,98	$0,87^{0,054}$		0,85
		2*1000	0,63	0,42	$0,12^{0,018}$	0,50	$0,49^{0,004}$	0,50	0,50		0,49
herbstFig6p18*	Variante 1	2*100	0,84	0,68	$0,18^{0,017}$	$0,19^{0,021}$	$0,20^{0,013}$	0,07	0,07		0,11
		2*1000	0,84	0,68	$0,16^{0,004}$	$0,10^{0,010}$	0,02	0,02	0,02		0,02
herbstFig6p34*	Variante 1	2*100	0,69	0,61	$0,16^{0,008}$	$0,23^{0,013}$	$0,36^{0,013}$	$0,45^{0,029}$	$0,48^{0,083}$	✓	0,43
		2*1000	0,69	0,61	$0,10^{0,003}$	$0,20^{0,008}$	$0,32^{0,012}$	$0,38^{0,013}$	$0,61^{0,046}$	✓	0,44
	Variante 2	2*100	0,69	0,61	$0,21^{0,020}$	$0,17^{0,002}$	$0,26^{0,001}$	$0,34^{0,005}$	$0,78^{0,058}$		0,46
		2*1000	0,69	0,61	$0,36^{0,065}$	$0,99^{0,005}$	1,00	1,00	1,00	✓	1,00
	Variante 3	2*100	0,90	0,80	$0,10^{0,002}$	$0,25^{0,010}$	$0,64^{0,025}$	$0,76^{0,002}$	$0,78^{0,043}$	✓	0,73
		2*1000	0,90	0,80	$0,09^{0,002}$	$0,21^{0,004}$	$0,40^{0,010}$	$0,46^{0,035}$	$0,64^{0,156}$	✓	0,50
al2*	Variante 1	2*100	0,97	0,77	$0,11^{0,003}$	$0,25^{0,005}$	0,40	$0,30^{0,002}$	$0,31^{0,002}$		0,34
		2*1000	1,00	0,81	$0,10^{0,004}$	0,06	0,06	0,06	0,06		0,06
	Variante 2	2*100	0,97	0,77	$0,36^{0,085}$	$0,26^{0,003}$	0,09	0,13	0,14		0,12
		2*1000	1,00	0,81	$0,06^{0,003}$	$0,06^{0,006}$	$0,28^{0,005}$	$0,36^{0,003}$	$0,45^{0,004}$	✓	0,36

Erreichte Präzision von APClustering für Änderungstyp (AP9) bei $m_c = 4$.

Netz		Pfade	*rel*	*par*	Präzision ($m_c = 6$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,34	0,03	$0,17^{0,040}$	0,21	$0,24^{0,007}$	$0,46^{0,121}$	0,20		0,30
		2*1000	0,34	0,03	$0,81^{0,133}$	1,00	1,00	1,00	$0,74^{0,190}$		0,91
	Variante 2	2*100	0,34	0,03	$0,12^{0,007}$	$0,44^{0,028}$	$0,74^{0,061}$	0,99	0,99	✓	0,91
		2*1000	0,34	0,03	$0,75^{0,154}$	1,00	1,00	1,00	1,00	✓	1,00
	Variante 3	2*100	0,34	0,03	$0,12^{0,003}$	$0,38^{0,015}$	$0,19^{0,002}$	$0,31^{0,009}$	0,35		0,28
		2*1000	0,34	0,03	$0,08^{0,014}$	$0,13^{0,114}$	0,00	0,00	$0,54^{0,243}$		0,18
herbstFig6p41*	Variante 1	2*100	0,43	0,16	$0,13^{0,042}$	$0,13^{0,002}$	$0,29^{0,002}$	$0,43^{0,005}$	0,47	✓	0,40
		2*1000	0,43	0,16	$0,14^{0,008}$	$0,43^{0,030}$	$0,43^{0,023}$	$0,42^{0,010}$	$0,45^{0,006}$		0,43
	Variante 2	2*100	0,57	0,21	$0,08^{0,004}$	$0,20^{0,008}$	$0,42^{0,056}$	$0,79^{0,055}$	$0,98^{0,001}$	✓	0,73
		2*1000	0,57	0,21	$0,29^{0,109}$	$0,87^{0,045}$	0,99	0,99	0,99	✓	0,99
	Variante 3	2*100	0,49	0,18	$0,12^{0,042}$	$0,17^{0,005}$	$0,50^{0,031}$	$0,79^{0,038}$	$0,88^{0,010}$	✓	0,73
		2*1000	0,49	0,18	$0,18^{0,030}$	$0,31^{0,056}$	$0,29^{0,030}$	$0,50^{0,002}$	0,50		0,43
herbstFig6p42	Variante 1	2*100	0,63	0,42	$0,13^{0,041}$	$0,19^{0,010}$	$0,35^{0,006}$	$0,46^{0,002}$	$0,41^{0,008}$		0,41
		2*1000	0,63	0,42	$0,27^{0,020}$	0,50	$0,68^{0,058}$	0,99	0,99		0,89
	Variante 2	2*100	0,70	0,47	$0,15^{0,047}$	$0,18^{0,003}$	$0,21^{0,006}$	$0,17^{0,005}$	0,12		0,17
		2*1000	0,70	0,47	$0,18^{0,017}$	$0,35^{0,004}$	0,49	0,49	$0,64^{0,051}$	✓	0,54
	Variante 3	2*100	0,63	0,42	$0,18^{0,046}$	$0,29^{0,013}$	$0,68^{0,093}$	0,98	$0,87^{0,054}$		0,84
		2*1000	0,63	0,42	$0,29^{0,038}$	0,50	0,50	0,50	0,50	✓	0,50
herbstFig6p18*	Variante 1	2*100	0,84	0,68	$0,17^{0,015}$	$0,19^{0,016}$	$0,21^{0,011}$	0,09	0,09		0,13
		2*1000	0,84	0,68	$0,15^{0,004}$	$0,10^{0,010}$	0,02	0,02	0,02		0,02
herbstFig6p34*	Variante 1	2*100	0,69	0,61	$0,16^{0,007}$	$0,22^{0,014}$	$0,36^{0,010}$	$0,45^{0,029}$	$0,48^{0,083}$	✓	0,43
		2*1000	0,69	0,61	$0,09^{0,002}$	$0,20^{0,008}$	$0,31^{0,013}$	$0,38^{0,013}$	$0,62^{0,045}$	✓	0,44
	Variante 2	2*100	0,69	0,61	$0,19^{0,010}$	$0,16^{0,002}$	$0,26^{0,001}$	$0,34^{0,005}$	$0,78^{0,058}$		0,46
		2*1000	0,69	0,61	$0,34^{0,067}$	$0,94^{0,025}$	1,00	1,00	1,00		1,00
	Variante 3	2*100	0,90	0,80	$0,10^{0,002}$	$0,21^{0,009}$	$0,64^{0,025}$	$0,76^{0,003}$	$0,82^{0,037}$	✓	0,74
		2*1000	0,90	0,80	$0,08^{0,001}$	$0,21^{0,004}$	$0,40^{0,010}$	$0,46^{0,034}$	$0,64^{0,156}$	✓	0,50
al2*	Variante 1	2*100	0,97	0,77	$0,11^{0,003}$	$0,25^{0,005}$	0,41	$0,30^{0,002}$	$0,31^{0,002}$		0,34
		2*1000	1,00	0,81	$0,09^{0,004}$	0,06	0,06	0,06	0,06		0,06
	Variante 2	2*100	0,97	0,77	$0,20^{0,030}$	$0,19^{0,001}$	0,12	$0,20^{0,003}$	0,24		0,19
		2*1000	1,00	0,81	$0,05^{0,003}$	$0,06^{0,005}$	$0,28^{0,006}$	$0,35^{0,003}$	$0,45^{0,004}$	✓	0,36

Erreichte Präzision von APClustering für Änderungstyp (AP9) bei $m_c = 6$.

Netz		Pfade	*rel*	*par*	Präzision ($m_c = 10$)					↗	ϑ
					1	2	3	4	5		
a12*	Variante 1	2*100	0,34	0,03	$0,25^{0,066}$	$0,17^{0,001}$	0,20	$0,46^{0,121}$	0,20		0,28
		2*1000	0,34	0,03	$0,82^{0,129}$	1,00	1,00	1,00	$0,74^{0,190}$		0,91
	Variante 2	2*100	0,34	0,03	$0,25^{0,070}$	$0,46^{0,028}$	$0,74^{0,061}$	0,99	0,99	✓	0,91
		2*1000	0,34	0,03	$0,76^{0,154}$	1,00	1,00	1,00	1,00	✓	1,00
	Variante 3	2*100	0,34	0,03	$0,22^{0,071}$	$0,36^{0,010}$	$0,19^{0,003}$	$0,31^{0,009}$	0,35		0,28
		2*1000	0,34	0,03	$0,08^{0,013}$	$0,14^{0,114}$	0,00	0,00	$0,54^{0,243}$		0,18
herbstFig6p41*	Variante 1	2*100	0,43	0,16	$0,28^{0,129}$	$0,15^{0,002}$	$0,29^{0,002}$	$0,43^{0,005}$	0,47		0,39
		2*1000	0,43	0,16	$0,14^{0,015}$	$0,50^{0,026}$	$0,53^{0,034}$	$0,70^{0,080}$	$0,81^{0,057}$	✓	0,68
	Variante 2	2*100	0,57	0,21	$0,27^{0,105}$	$0,17^{0,006}$	$0,28^{0,035}$	$0,69^{0,055}$	$0,97^{0,002}$		0,65
		2*1000	0,57	0,21	$0,29^{0,112}$	$0,87^{0,045}$	0,99	0,99	0,99	✓	0,99
	Variante 3	2*100	0,49	0,18	$0,34^{0,147}$	$0,16^{0,003}$	$0,29^{0,001}$	0,30	$0,41^{0,005}$		0,33
		2*1000	0,49	0,18	$0,17^{0,023}$	$0,29^{0,023}$	$0,28^{0,006}$	0,50	0,50		0,43
herbstFig6p42	Variante 1	2*100	0,63	0,42	$0,23^{0,077}$	$0,20^{0,005}$	$0,47^{0,004}$	0,49	$0,43^{0,006}$		0,46
		2*1000	0,63	0,42	$0,33^{0,038}$	0,50	$0,68^{0,058}$	0,99	0,99		0,89
	Variante 2	2*100	0,70	0,47	$0,29^{0,128}$	$0,17^{0,003}$	$0,23^{0,005}$	$0,22^{0,002}$	0,19		0,21
		2*1000	0,70	0,47	$0,22^{0,019}$	$0,36^{0,004}$	0,49	0,50	$0,64^{0,052}$	✓	0,54
	Variante 3	2*100	0,63	0,42	$0,22^{0,073}$	$0,22^{0,018}$	$0,40^{0,013}$	0,49	$0,79^{0,067}$		0,56
		2*1000	0,63	0,42	$0,29^{0,042}$	0,50	0,50	0,50	0,50	✓	0,50
herbstFig6p18*	Variante 1	2*100	0,84	0,68	$0,14^{0,005}$	$0,23^{0,012}$	$0,20^{0,010}$	0,09	0,09		0,13
		2*1000	0,84	0,68	$0,13^{0,004}$	$0,10^{0,010}$	0,02	0,02	0,02		0,02
herbstFig6p34*	Variante 1	2*100	0,69	0,61	$0,20^{0,071}$	$0,19^{0,008}$	$0,36^{0,010}$	$0,45^{0,029}$	$0,49^{0,079}$		0,43
		2*1000	0,69	0,61	$0,08^{0,007}$	$0,19^{0,007}$	$0,30^{0,014}$	$0,39^{0,014}$	$0,62^{0,045}$	✓	0,44
	Variante 2	2*100	0,69	0,61	$0,20^{0,041}$	$0,18^{0,002}$	$0,26^{0,001}$	$0,34^{0,005}$	$0,79^{0,061}$		0,46
		2*1000	0,69	0,61	$0,26^{0,060}$	$0,91^{0,036}$	1,00	1,00	1,00		1,00
	Variante 3	2*100	0,90	0,80	$0,21^{0,069}$	$0,22^{0,007}$	$0,64^{0,025}$	$0,75^{0,003}$	$0,91^{0,012}$	✓	0,77
		2*1000	0,90	0,80	$0,08^{0,006}$	$0,21^{0,004}$	$0,40^{0,010}$	$0,45^{0,031}$	$0,64^{0,156}$	✓	0,50
al2*	Variante 1	2*100	0,97	0,77	$0,13^{0,003}$	$0,23^{0,006}$	0,39	$0,31^{0,001}$	$0,32^{0,001}$		0,34
		2*1000	1,00	0,81	$0,09^{0,004}$	0,06	0,06	0,06	0,06		0,06
	Variante 2	2*100	0,97	0,77	$0,18^{0,029}$	$0,20^{0,006}$	0,19	$0,18^{0,001}$	0,21		0,19
		2*1000	1,00	0,81	$0,05^{0,003}$	$0,05^{0,004}$	$0,27^{0,006}$	$0,34^{0,004}$	$0,45^{0,003}$		0,36

Erreichte Präzision von APClustering für Änderungstyp (AP9) bei $m_c = 10$.

Netz	$rel_\varnothing$	$par_\varnothing$	$m_c = 4$		$m_c = 6$		$m_c = 10$		Aggregiert	
			$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$
a12*	0,29	0,03	0,52	0,66+	0,35	0,66+	0,35	0,50+		
herbstFig6p41*	0,45	0,17	0,37	0,82+	0,47	0,82+	0,49	0,82+	0,38	0,63
herbstFig6p42	0,63	0,42	0,26	0,45+	0,30	0,45+	0,30	0,45+		
herbstFig6p18*	0,58	0,47	0,16+	0,16	0,18+	0,16	0,19+	0,16		
herbstFig6p34*	0,69	0,61	0,26	0,49+	0,27	0,53+	0,37	0,53+	0,21	0,27
al2*	0,82	0,66	0,11	0,12+	0,17+	0,12	0,19+	0,12		

Exaktheit der Erkennung von Änderungen des Typs (AP1b).

Netz	$rel_\varnothing$	$par_\varnothing$	$m_c = 4$		$m_c = 6$		$m_c = 10$		Aggregiert	
			$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$
a12*	0,28	0,03	0,75	0,83+	0,65	0,77+	0,61	0,77+		
herbstFig6p41*	0,43	0,16	0,55	0,77+	0,62	0,83+	0,74	0,81+	0,56	0,76
herbstFig6p42	0,63	0,42	0,40	0,64+	0,40	0,72+	0,36	0,69+		
herbstFig6p18*	0,52	0,41	0,17	0,20+	0,23+	0,20	0,24+	0,20		
herbstFig6p34*	0,69	0,61	0,50+	0,46	0,50+	0,45	0,50+	0,46	0,32	0,28
al2*	0,79	0,63	0,26+	0,20	0,23+	0,19	0,24+	0,18		

Exaktheit der Erkennung von Änderungen des Typs (AP1c).

Netz	$rel_\varnothing$	$par_\varnothing$	$m_c = 4$		$m_c = 6$		$m_c = 10$		Aggregiert	
			$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$
a12*	0,34	0,03	0,71+	0,60	0,86	0,92+	0,85	0,92+		
herbstFig6p41*	0,57	0,21	0,66	0,74+	0,66	0,74+	0,76+	0,71	0,74	0,77
herbstFig6p42	0,80	0,54	0,71	0,75+	0,74	0,77+	0,71	0,77+		
herbstFig6p18*	0,84	0,68	0,78+	0,52	0,77+	0,51	0,79+	0,54		
herbstFig6p34*	0,90	0,80	0,52	0,55+	0,52	0,56+	0,64+	0,61	0,57	0,60
al2*	1,00	0,81	0,37	0,72+	0,36	0,72+	0,42	0,71+		

Exaktheit der Erkennung von Änderungen des Typs (AP2).

Netz	$rel_\varnothing$	$par_\varnothing$	$m_c = 4$		$m_c = 6$		$m_c = 10$		Aggregiert	
			$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$
a12*	0,34	0,03	0,70	0,74+	0,77	0,90+	0,85	0,94+		
herbstFig6p41*	0,54	0,20	0,55	0,70+	0,53	0,73+	0,53	0,92+	0,53	0,75
herbstFig6p42	0,80	0,53	0,27	0,63+	0,28	0,63+	0,31	0,62+		
herbstFig6p18*	0,70	0,57	0,33	0,79+	0,34	0,79+	0,38	0,79+		
herbstFig6p34*	0,90	0,80	0,37	0,57+	0,38	0,56+	0,41	0,56+	0,32	0,54
al2*	0,96	0,77	0,22	0,23+	0,22	0,28+	0,22	0,28+		

Exaktheit der Erkennung von Änderungen des Typs (AP3a).

Netz	$rel_{\varnothing}$	$par_{\varnothing}$	$m_c = 4$		$m_c = 6$		$m_c = 10$		Aggregiert	
			$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$
a12*	0,34	0,03	0,39	0,92+	0,40	0,92+	0,45	0,92+		
herbstFig6p41*	0,57	0,21	0,61+	0,57	0,68+	0,56	0,71+	0,70	0,45	0,63
herbstFig6p42	0,80	0,53	0,26	0,30+	0,27	0,38+	0,30	0,41+		
herbstFig6p18*	0,84	0,68	0,42+	0,31	0,42+	0,39	0,35	0,51+		
herbstFig6p34*	0,90	0,80	0,38	0,56+	0,41	0,56+	0,42	0,55+	0,36	0,46
al2*	0,99	0,80	0,28	0,36+	0,27	0,48+	0,28	0,41+		

Exaktheit der Erkennung von Änderungen des Typs (AP3b).

Netz	$rel_{\varnothing}$	$par_{\varnothing}$	$m_c = 4$		$m_c = 6$		$m_c = 10$		Aggregiert	
			$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$
a12*	0,33	0,03	0,64+	0,59	0,60+	0,59	0,60+	0,59		
herbstFig6p41*	0,57	0,21	0,47	0,55+	0,49	0,68+	0,49	0,68+	0,46	0,55
herbstFig6p42	0,80	0,54	0,30	0,38+	0,28	0,45+	0,29	0,44+		
herbstFig6p18*	0,67	0,54	0,20	0,29+	0,20	0,29+	0,26	0,32+		
herbstFig6p34*	0,90	0,80	0,37	0,51+	0,36	0,50+	0,37	0,51+	0,27	0,33
al2*	0,90	0,73	0,20+	0,16	0,21+	0,18	0,21+	0,19		

Exaktheit der Erkennung von Änderungen des Typs (AP3c).

Netz	$rel_{\varnothing}$	$par_{\varnothing}$	$m_c = 4$		$m_c = 6$		$m_c = 10$		Aggregiert	
			$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$
a12*	0,28	0,03	0,85	1,00+	0,85	1,00+	0,84	1,00+		
herbstFig6p41*	0,45	0,17	0,73	0,78+	0,86	1,00+	0,86+	0,83	0,64	0,80
herbstFig6p42	0,63	0,42	0,29	0,66+	0,25	0,50+	0,27	0,46+		
herbstFig6p18*	0,58	0,46	0,40	0,84+	0,40	0,83+	0,45	0,61+		
herbstFig6p34*	0,69	0,61	0,52	0,60+	0,52	0,59+	0,53	0,58+	0,43	0,65
al2*	0,82	0,66	0,34	0,59+	0,35	0,60+	0,35	0,60+		

Exaktheit der Erkennung von Änderungen des Typs (AP4).

Netz	$rel_{\varnothing}$	$par_{\varnothing}$	$m_c = 4$		$m_c = 6$		$m_c = 10$		Aggregiert	
			$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$
a12*	0,33	0,03	0,57	0,75+	0,67	0,76+	0,64	0,87+		
herbstFig6p41*	0,54	0,20	0,55	0,58+	0,61+	0,58	0,68	0,90+	0,51	0,61
herbstFig6p42	0,80	0,54	0,29	0,31+	0,31	0,38+	0,29	0,38+		
herbstFig6p18*	0,84	0,68	0,39+	0,38	0,38	0,39+	0,38	0,39+		
herbstFig6p34*	0,90	0,80	0,51	0,52+	0,50	0,51+	0,48	0,50+	0,36	0,37
al2*	0,97	0,76	0,19	0,21+	0,21	0,21+	0,23+	0,22		

Exaktheit der Erkennung von Änderungen des Typs (AP5).

Netz	$rel_\varnothing$	$par_\varnothing$	$m_c = 4$		$m_c = 6$		$m_c = 10$		Aggregiert	
			$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$
a12*	0,28	0,03	0,71	0,93+	0,71	0,93+	0,72	0,93+		
herbstFig6p41*	0,43	0,16	0,62	0,78+	0,68	0,78+	0,68	0,73+	0,63	0,72
herbstFig6p42	0,65	0,43	0,52+	0,45	0,52+	0,45	0,49	0,50+		
herbstFig6p18*	0,69	0,56	0,37	0,60+	0,45	0,73+	0,49	0,91+		
herbstFig6p34*	0,69	0,61	0,95+	0,93	0,94+	0,82	0,88+	0,73	0,64	0,77
al2*	0,83	0,67	0,58	0,76+	0,58	0,76+	0,51	0,68+		

Exaktheit der Erkennung von Änderungen des Typs (AP6).

Netz	$rel_\varnothing$	$par_\varnothing$	$m_c = 4$		$m_c = 6$		$m_c = 10$		Aggregiert	
			$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$
a12*	0,20	0,02	0,53	0,83+	0,48	0,83+	0,63	1,00+		
herbstFig6p41*	0,33	0,13	0,39	0,44+	0,40	0,67+	0,56	0,77+	0,43	0,75
herbstFig6p42	0,50	0,33	0,28	0,56+	0,27	0,73+	0,30	0,90+		
herbstFig6p18*	0,43	0,34	0,24	0,62+	0,24	0,62+	0,26	0,58+		
herbstFig6p34*	0,54	0,48	0,75+	0,64	0,71	0,79+	0,73	0,79+	0,44	0,64
al2*	0,67	0,53	0,35	0,48+	0,33	0,64+	0,34	0,64+		

Exaktheit der Erkennung von Änderungen des Typs (AP7).

Netz	$rel_\varnothing$	$par_\varnothing$	$m_c = 4$		$m_c = 6$		$m_c = 10$		Aggregiert	
			$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$
a12*	0,34	0,03	0,31	0,71+	0,35	0,71+	0,36	0,71+		
herbstFig6p41*	0,57	0,21	0,40	0,73+	0,40	0,59+	0,42	0,75+	0,34	0,66
herbstFig6p42	0,80	0,54	0,27	0,59+	0,27	0,58+	0,26	0,58+		
herbstFig6p18*	0,84	0,68	0,24	0,62+	0,27	0,62+	0,35	0,62+		
herbstFig6p34*	0,90	0,80	0,34	0,42+	0,35	0,39+	0,37	0,39+	0,29	0,42
al2*	0,96	0,77	0,21	0,25+	0,23	0,25+	0,28+	0,25		

Exaktheit der Erkennung von Änderungen des Typs (AP8).

Netz	$rel_\varnothing$	$par_\varnothing$	$m_c = 4$		$m_c = 6$		$m_c = 10$		Aggregiert	
			$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$	$\vartheta_{\varnothing_{100}}$	$\vartheta_{\varnothing_{1000}}$
a12*	0,34	0,03	0,49	0,54+	0,50	0,70+	0,49	0,70+		
herbstFig6p41*	0,50	0,19	0,63+	0,54	0,62	0,62+	0,46	0,70+	0,50	0,64
herbstFig6p42	0,65	0,43	0,45	0,64+	0,47	0,64+	0,41	0,64+		
herbstFig6p18*	0,84	0,68	0,11+	0,02	0,13+	0,02	0,13+	0,02		
herbstFig6p34*	0,76	0,67	0,54	0,65+	0,54	0,64+	0,55	0,64+	0,31	0,29
al2*	0,99	0,79	0,23+	0,21	0,27+	0,21	0,27+	0,21		

Exaktheit der Erkennung von Änderungen des Typs (AP9).

Literatur

Aalst, Wil M.P. van der (1996). "Three Good reasons for Using a Petri-net-based Workflow Management System". In: *Proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPIC'96)*. Hrsg. von S. Navathe und T. Wakayama, S. 179–201.

— (1998). "The Application of Petri Nets to Workflow Management". In: *Journal of Circuits, Systems, and Computers* 8.1, S. 21–66.

— (2003). "Business Process Management Demystified: A Tutorial on Models, Systems and Standards for Workflow Management". In: *Lectures on Concurrency and Petri Nets*. Hrsg. von Jörg Desel, Wolfgang Reisig und Grzegorz Rozenberg. Bd. 3098. Lecture Notes in Computer Science. Springer, S. 1–65. ISBN: 3-540-22261-8.

— (2005). "Business Alignment: Using Process Mining as a Tool for Delta Analysis and Conformance Testing". In: *Requirements Engineering* 10.3, S. 198–211.

— (2011). *Process Mining - Discovery, Conformance and Enhancement of Business Processes*. Springer, S. I–XVI, 1–352. ISBN: 978-3-642-19344-6.

Aalst, Wil M.P. van der, Arya Adriansyah und Boudewijn F. van Dongen (2011). "Causal Nets: A Modeling Language Tailored towards Process Discovery". In: *CONCUR*. Hrsg. von Joost-Pieter Katoen und Barbara König. Bd. 6901. Lecture Notes in Computer Science. Springer, S. 28–42. ISBN: 978-3-642-23216-9.

— (2012). "Replaying History on Process Models for Conformance Checking and Performance Analysis". In: *Wiley Interdisciplinary Reviews.: Data Mining and Knowledge Discovery* 2.2, S. 182–192.

Aalst, Wil M.P. van der, Ana Karla Alves de Medeiros und A.J.M.M. (Ton) Weijters (2005). "Genetic Process Mining". In: *ICATPN*. Hrsg. von Gianfranco Ciardo und Philippe Darondeau. Bd. 3536. Lecture Notes in Computer Science. Springer, S. 48–69. ISBN: 3-540-26301-2.

Aalst, Wil M.P. van der, H.T. de Beer und Boudewijn F. van Dongen (2005). "Process Mining and Verification of Properties: An Approach Based on Temporal Logic". In: *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE, OTM Confederated International Conferences CoopIS, DOA, and ODBASE 2005, Agia*

Napa, Cyprus, October 31 - November 4, 2005, Proceedings, Part I. Hrsg. von Robert Meersman, Zahir Tari, Mohand-Said Hacid, John Mylopoulos, Barbara Pernici, Özalp Babaoglu, Hans-Arno Jacobsen, Joseph P. Loyall, Michael Kifer und Stefano Spaccapietra. Bd. 3760. Lecture Notes in Computer Science. Springer, S. 130–147. ISBN: 3-540-29736-7.

Aalst, Wil M.P. van der, Boudewijn F. van Dongen, Christian W. Günther, Anne Rozinat, H. M. W. (Eric) Verbeek und A.J.M.M. (Ton) Weijters (2009). "ProM: The Process Mining Toolkit". In: *Proceedings of the Business Process Management Demonstration Track (BPMDemos 2009), Ulm, Germany, September 8, 2009*. Hrsg. von Ana Karla Alves de Medeiros und Barbara Weber. Bd. 489. CEUR Workshop Proceedings. CEUR-WS.org.

Aalst, Wil M.P. van der, Kees M. van Hee, Jan Martijn van der Werf und Marc Verdonk (2010). "Auditing 2.0: Using Process Mining to Support Tomorrow's Auditor". In: *IEEE Computer* 43.3, S. 90–93.

Aalst, Wil M.P. van der, Arthur H.M. ter Hofstede und Mathias Weske, Hrsg. (2003). *Business Process Management, International Conference, BPM 2003, Eindhoven, The Netherlands, June 26-27, 2003, Proceedings*. Bd. 2678. Lecture Notes in Computer Science. Springer. ISBN: 3-540-40318-3.

Aalst, Wil M.P. van der, Hajo A. Reijers und Minseok Song (2005). "Discovering Social Networks from Event Logs". In: *Computer Supported Cooperative Work* 14.6, S. 549–593.

Aalst, Wil M.P. van der, Vladimir Rubin, H.M.W. Verbeek, Boudewijn F. van Dongen, Ekkart Kindler und Christian W. Günther (2010). "Process Mining: A Two-Step Approach to Balance Between Underfitting and Overfitting". In: *Software and System Modeling* 9.1, S. 87–111.

Aalst, Wil M.P. van der und Minseok Song (2004). "Mining Social Networks: Uncovering Interaction Patterns in Business Processes". In: *Business Process Management*. Hrsg. von Jörg Desel, Barbara Pernici und Mathias Weske. Bd. 3080. Lecture Notes in Computer Science. Springer, S. 244–260. ISBN: 3-540-22235-9.

Aalst, Wil M.P. van der, A.J.M.M. (Ton) Weijters und Laura Maruster (2004). "Workflow Mining: Discovering Process Models from Event Logs". In: *IEEE Transactions on Knowledge and Data Engineering* 16.9, S. 1128–1142.

Abramowicz, Witold, Dalia Kriksciuniene und Virgilijus Sakalauskas, Hrsg. (2012). *Business Information Systems - 15th International Conference, BIS 2012, Vilnius, Lithuania, May 21-23, 2012. Proceedings*. Bd. 117. Lecture Notes in Business Information Processing. Springer. ISBN: 978-3-642-30358-6.

Accorsi, Rafael, Julius Holderer, Thomas Stocker und Richard M. Zahoransky (2014). "Security Workflow Analysis Toolkit". In: *Sicherheit*. Hrsg. von Stefan Katzenbeisser,

Volkmar Lotz und Edgar R. Weippl. Bd. 228. LNI. GI, S. 433–442. ISBN: 978-3-88579-622-0.

Accorsi, Rafael und Andreas Lehmann (2012). "Automatic Information Flow Analysis of Business Process Models". In: *Business Process Management - 10th International Conference, BPM 2012, Tallinn, Estonia, September 3-6, 2012. Proceedings*. Hrsg. von Alistair P. Barros, Avigdor Gal und Ekkart Kindler. Bd. 7481. Lecture Notes in Computer Science. Springer, S. 172–187. ISBN: 978-3-642-32884-8.

Accorsi, Rafael, Yoshinori Sato und Satoshi Kai (2008). "Compliance-Monitor zur Frühwarnung vor Risiken". In: *Wirtschaftsinformatik* 50.5, S. 375–382.

Accorsi, Rafael und Claus Wonnemann (2011). "Strong Non-leak Guarantees for Workflow Models". In: *Proceedings of the 2011 ACM Symposium on Applied Computing (SAC), TaiChung, Taiwan, March 21 - 24, 2011*. Hrsg. von William C. Chu, W. Eric Wong, Mathew J. Palakal und Chih-Cheng Hung. ACM, S. 308–314. ISBN: 978-1-4503-0113-8.

Accorsi, Rafael, Claus Wonnemann und Thomas Stocker (2011). "Towards Forensic Data Flow Analysis of Business Process Logs". In: *IMF*. Hrsg. von Holger Morgenstern, Ralf Ehlert, Sandra Frings, Oliver Göbel, Detlef Günther, Stefan Kiltz, Jens Nedon und Dirk Schadt. IEEE Computer Society, S. 3–20. ISBN: 978-0-7695-4403-8.

ACL Services Limited (2014). *ACL GRC*. http://www.acl.com/solutions/products/acl-grc/. [Online; Stand: 22.09.2014].

Adam, Nabil R., Vijayalakshmi Atluri und WeiKuang Huang (1998). "Modeling and Analysis of Workflows Using Petri Nets". In: *Journal of Intelligent Information Systems* 10.2, S. 131–158.

Adriansyah, Arya, Boudewijn F. van Dongen und Wil M.P. van der Aalst (2010). "Towards Robust Conformance Checking". In: *Business Process Management Workshops - BPM 2010 International Workshops and Education Track, Hoboken, NJ, USA, September 13-15, 2010, Revised Selected Papers*. Hrsg. von Michael zur Muehlen und Jianwen Su. Bd. 66. Lecture Notes in Business Information Processing. Springer, S. 122–133. ISBN: 978-3-642-20510-1.

— (2011). "Conformance Checking Using Cost-Based Fitness Analysis". In: *EDOC*. IEEE Computer Society, S. 55–64. ISBN: 978-1-4577-0362-1.

Adriansyah, Arya, Jorge Munoz-Gama, Josep Carmona, Boudewijn F. van Dongen und Wil M.P. van der Aalst (2012). "Alignment Based Precision Checking". In: *Business Process Management Workshops*. Hrsg. von Marcello La Rosa und Pnina Soffer. Bd. 132. Lecture Notes in Business Information Processing. Springer, S. 137–149. ISBN: 978-3-642-36284-2.

Agrawal, Rakesh, Dimitrios Gunopulos und Frank Leymann (1998). "Mining Process Models from Workflow Logs". In: *EDBT*. Hrsg. von Hans-Jörg Schek, Fèlix Saltor,

Isidro Ramos und Gustavo Alonso. Bd. 1377. Lecture Notes in Computer Science. Springer, S. 469–483. ISBN: 3-540-64264-1.

AICPA SAS70. *Statement on Auditing Standards (SAS) No. 70*. American Institute of Certified Public Accountants (AICPA).

AICPA SSAE16. *Statement on Standards for Attestation Engagements - Reporting on Controls at a Service Organization*. American Institute of Certified Public Accountants (AICPA).

Alonso, Gustavo, Peter Dadam und Michael Rosemann, Hrsg. (2007). *Business Process Management, 5th International Conference, BPM 2007, Brisbane, Australia, September 24-28, 2007, Proceedings*. Bd. 4714. Lecture Notes in Computer Science. Springer. ISBN: 978-3-540-75182-3.

Alturi, Vijayalakshmi und David F. Ferraiolo (2011). "Role-Based Access Control". In: *Encyclopedia of Cryptography and Security, 2nd Ed.* Hrsg. von Henk C.A. van Tilborg und Sushil Jajodia. Springer, S. 1053–1055. ISBN: 978-1-4419-5905-8.

Alves de Medeiros, Ana Karla (2006). "Genetic Process Mining". Diss. Eindhoven University of Technology.

Alves de Medeiros, Ana Karla, Wil M.P. van der Aalst und A.J.M.M. (Ton) Weijters (2008). "Quantifying process equivalence based on observed behavior". In: *Data and Knowledge Engineering* 64.1, S. 55–74.

Alves de Medeiros, Ana Karla, Boudewijn F. van Dongen, Wil M.P. van der Aalst und A.J.M.M. (Ton) Weijters (2004a). *Process Mining: Extending the α-algorithm to Mine Short Loops*. BETA Working Paper Series WP 113. Eindhoven University of Technology.

— (2004b). "Process Mining for Ubiquitous Mobile Systems: An Overview and a Concrete Algorithm". In: *UMICS*. Hrsg. von Luciano Baresi, Schahram Dustdar, Harald Gall und Maristella Matera. Bd. 3272. Lecture Notes in Computer Science. Springer, S. 151–165. ISBN: 3-540-24100-0.

Alves de Medeiros, Ana Karla, Antonella Guzzo, Gianluigi Greco, Wil M.P. van der Aalst, A.J.M.M. (Ton) Weijters, Boudewijn F. van Dongen und Domenico Saccà (2007). "Process Mining Based on Clustering: A Quest for Precision". In: *Business Process Management Workshops*. Hrsg. von Arthur H.M. ter Hofstede, Boualem Benatallah und Hye-Young Paik. Bd. 4928. Lecture Notes in Computer Science. Springer, S. 17–29. ISBN: 978-3-540-78237-7.

Alves de Medeiros, Ana Karla, A.J.M.M. (Ton) Weijters und Wil M.P. van der Aalst (2004c). *Using Genetic Algorithms to Mine Process Models Representation, Operators and Results*. BETA Working Paper Series WP 124. Eindhoven University of Technology.

— (2007). "Genetic Process Mining: An Experimental Evaluation". In: *Data Mining and Knowledge Discovery* 14.2, S. 245–304.

Atluri, Vijayalakshmi und Janice Warner (2008). "Security for Workflow Systems". In: *Handbook of Database Security - Applications and Trends*. Hrsg. von Michael Gertz und Sushil Jajodia. Springer, S. 213–230. ISBN: 978-0-387-48532-4.

BA 54-FR 2210-2012/0002. *Mindestanforderungen an das Risikomanagement - MaRisk*. Bundesanstalt für Finanzdienstleistungsaufsicht (BaFin).

Badouel, Eric, Luca Bernardinello und Philippe Darondeau (1995). "Polynomial Algorithms for the Synthesis of Bounded Nets". In: *TAPSOFT*. Hrsg. von Peter D. Mosses, Mogens Nielsen und Michael I. Schwartzbach. Bd. 915. Lecture Notes in Computer Science. Springer, S. 364–378. ISBN: 3-540-59293-8.

— (1997). "The Synthesis Problem for Elementary Net Systems is NP-Complete". In: *Theoretical Computer Science* 186.1-2, S. 107–134.

Badouel, Eric und Philippe Darondeau (1996). "Theory of Regions". In: *Petri Nets*. Hrsg. von Wolfgang Reisig und Grzegorz Rozenberg. Bd. 1491. Lecture Notes in Computer Science. Springer, S. 529–586. ISBN: 3-540-65306-6.

Barizo, Dee (2014). *The 10 worst corporate accounting scandals of all time*. http://www.accounting-degree.org/scandals/. [Online; Stand: 03.09.2014].

Baumgrass, Anne, Thomas Baier, Jan Mendling und Mark Strembeck (2011). "Conformance Checking of RBAC Policies in Process-Aware Information Systems". In: *Business Process Management Workshops - BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part II*. Hrsg. von Florian Daniel, Kamel Barkaoui und Schahram Dustdar. Bd. 100. Lecture Notes in Business Information Processing. Springer, S. 435–446. ISBN: 978-3-642-28114-3.

Baumgrass, Anne und Mark Strembeck (2012). "An Approach to Bridge the Gap between Role Mining and Role Engineering via Migration Guides". In: *Seventh International Conference on Availability, Reliability and Security, Prague, ARES 2012, Czech Republic, August 20-24, 2012*. IEEE Computer Society, S. 113–122. ISBN: 978-1-4673-2244-7.

Baumgrass, Anne, Mark Strembeck und Stefanie Rinderle-Ma (2011). "Deriving Role Engineering Artifacts from Business Processes and Scenario Models". In: *SACMAT 2011, 16th ACM Symposium on Access Control Models and Technologies, Innsbruck, Austria, June 15-17, 2011, Proceedings*. Hrsg. von Ruth Breu, Jason Crampton und Jorge Lobo. ACM, S. 11–20. ISBN: 978-1-4503-0688-1.

Becker, Michael und Ralf Laue (2012). "A Comparative Survey of Business Process Similarity Measures". In: *Computers in Industry* 63.2, S. 148–167.

Beer, H.T. de (2004). *The LTL Checker Plugins: A Reference Manual*. Eindhoven University of Technology. Eindhoven.

Bergenthum, Robin, Jörg Desel, Robert Lorenz und Sebastian Mauser (2007). "Process Mining Based on Regions of Languages". In: *BPM*. Hrsg. von Gustavo Alonso, Peter Dadam und Michael Rosemann. Bd. 4714. Lecture Notes in Computer Science. Springer, S. 375–383. ISBN: 978-3-540-75182-3.

— (2008). "Synthesis of Petri Nets from Finite Partial Languages". In: *Fundamenta Informaticae* 88.4, S. 437–468.

Bernardinello, Luca, G. De Michelis und K. Petruni (1993). *Synchronic Distances as Generalized Regions*. Techn. Ber. 107-93. Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano.

Best, Eva und Martin Weth (2010). *Process Excellence*. Gabler Verlag / Springer Fachmedien Wiesbaden GmbH, Wiesbaden. ISBN: 978-3-8349-221-3.

Biermann, A. W. und J. A. Feldman (1972). "On the Synthesis of Finite-State Machines from Samples of Their Behavior". In: *IEEE Transactions on Computers* 21.6, S. 592–597. ISSN: 0018-9340.

Böhr, Frank, Linh Thao Ly und Günter Müller (2013). "Business Process Security Analysis - Design Time, Run Time, Audit Time". In: *it - Information Technology* 55.6, S. 217–224.

Bose, R. P. Jagadeesh Chandra und Wil M.P. van der Aalst (2009a). "Context Aware Trace Clustering: Towards Improving Process Mining Results". In: *SDM*. SIAM, S. 401–412. ISBN: 978-0-89871-682-5, 978-1-61197-279-5.

— (2009b). "Trace Clustering Based on Conserved Patterns: Towards Achieving Better Process Models". In: *Business Process Management Workshops*. Hrsg. von Stefanie Rinderle-Ma, Shazia Wasim Sadiq und Frank Leymann. Bd. 43. Lecture Notes in Business Information Processing. Springer, S. 170–181. ISBN: 978-3-642-12185-2.

— (2010). "Trace Alignment in Process Mining: Opportunities for Process Diagnostics". In: *BPM*. Hrsg. von Richard Hull, Jan Mendling und Stefan Tai. Bd. 6336. Lecture Notes in Computer Science. Springer, S. 227–242. ISBN: 978-3-642-15617-5.

Bose, R. P. Jagadeesh Chandra, Wil M.P. van der Aalst, Indre Zliobaite und Mykola Pechenizkiy (2011). "Handling Concept Drift in Process Mining". In: *CAiSE*. Hrsg. von Haralambos Mouratidis und Colette Rolland. Bd. 6741. Lecture Notes in Computer Science. Springer, S. 391–405. ISBN: 978-3-642-21639-8.

Botha, Reinhardt und Jan Eloff (2001). "Separation of Duties for Access Control Enforcement in Workflow Environments". In: *IBM Systems Journal* 40.3, S. 666–682.

Bratosin, Carmen, Natalia Sidorova und Wil M.P. van der Aalst (2010). "Discovering Process Models with Genetic Algorithms Using Sampling". In: *Knowledge-Based and Intelligent Information and Engineering Systems - 14th International Conference, KES 2010, Cardiff, UK, September 8-10, 2010, Proceedings, Part I*. Hrsg. von

Rossitza Setchi, Ivan Jordanov, Robert J. Howlett und Lakhmi C. Jain. Bd. 6276. Lecture Notes in Computer Science. Springer, S. 41–50. ISBN: 978-3-642-15386-0.

Brewer, David F.C. und Michael J. Nash (1989). "The Chinese Wall Security Policy". In: *Proceedings of the 1989 IEEE Symposium on Security and Privacy, Oakland, California, USA, May 1-3, 1989*. IEEE Computer Society, S. 206–214. ISBN: 0-8186-1939-2.

Broucke, Seppe K. L. M van den, Jochen De Weerdt, Bart Baesens und Jan Vanthienen (2012). "Improved Artificial Negative Event Generation to Enhance Process Event Logs". In: *Proceedings of the 24th International Conference on Advanced Information Systems Engineering*. CAiSE'12. Gdańsk, Poland: Springer-Verlag, S. 254–269. ISBN: 978-3-642-31094-2.

Brucker, Achim D. und Helmut Petritsch (2009). "Extending access control models with break-glass". In: *SACMAT 2009, 14th ACM Symposium on Access Control Models and Technologies, Proceedings*. Hrsg. von Barbara Carminati und James Joshi. Stresa, Italy: ACM, S. 197–206. ISBN: 978-1-60558-537-6.

Buijs, Joos C. A. M., Boudewijn F. van Dongen und Wil M.P. van der Aalst (2012). "On the Role of Fitness, Precision, Generalization and Simplicity in Process Discovery". In: *OTM Conferences (1)*. Hrsg. von Robert Meersman, Hervé Panetto, Tharam S. Dillon, Stefanie Rinderle-Ma, Peter Dadam, Xiaofang Zhou, Siani Pearson, Alois Ferscha, Sonia Bergamaschi und Isabel F. Cruz. Bd. 7565. Lecture Notes in Computer Science. Springer, S. 305–322. ISBN: 978-3-642-33605-8, 978-3-642-33606-5.

Bundeskriminalamt (2013). *Wirtschaftskriminalität - Bundeslagebild 2013*.

Busi, Nadia und Roberto Gorrieri (2009). "Structural Non-interference in Elementary and Trace Nets". In: *Mathematical Structures in Computer Science* 19, S. 1065–1090.

Caillaud, B. (2002). *Synet: A Synthesizer of distributable bounded Petri-nets from finite automata*. http://www.irisa.fr/s4/tools/synet. [Online; Stand: 18.02.2014].

Carlin, Anna und Frederick Gallegos (2007). "IT Audit: A Critical Business Process". In: *IEEE Computer* 40.7, S. 87–89.

Carmona, Josep (2012). "The Label Splitting Problem". In: *Transactions on Petri Nets and Other Models of Concurrency VI*. Hrsg. von Kurt Jensen, Wil M.P. van der Aalst, Marco Ajmone Marsan, Giuliana Franceschinis, Jetty Kleijn und Lars Michael Kristensen. Bd. 6. Lecture Notes in Computer Science. Springer, S. 1–23. ISBN: 978-3-642-35178-5.

Carmona, Josep und Jordi Cortadella (2010). "Process Mining Meets Abstract Interpretation". In: *Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part I*. ECML PKDD'10. Barcelona, Spain: Springer-Verlag, S. 184–199. ISBN: 3-642-15879-X, 978-3-642-15879-7.

Carmona, Josep und Jordi Cortadella (2013). "Process Discovery Algorithms using Numerical Abstract Domains". In: *IEEE Transactions on Knowledge and Data Engineering* 99.PrePrints, S. 1. ISSN: 1041-4347.

Carmona, Josep, Jordi Cortadella und Michael Kishinevsky (2008). "A Region-Based Algorithm for Discovering Petri Nets from Event Logs". In: *BPM*. Hrsg. von Marlon Dumas, Manfred Reichert und MingChien Shan. Bd. 5240. Lecture Notes in Computer Science. Springer, S. 358–373. ISBN: 978-3-540-85757-0.

— (2009). "Genet: A Tool for the Synthesis and Mining of Petri Nets". In: *ACSD*. IEEE Computer Society, S. 181–185.

— (2010). "New Region-Based Algorithms for Deriving Bounded Petri Nets". In: *IEEE Transactions on Computers* 59.3, S. 371–384.

Carmona, Josep, Jordi Cortadella, Michael Kishinevsky, Alex Kondratyev, Luciano Lavagno und Alexandre Yakovlev (2008). "A Symbolic Algorithm for the Synthesis of Bounded Petri Nets". In: *Petri Nets*. Hrsg. von Kees M. van Hee und Rüdiger Valk. Bd. 5062. Lecture Notes in Computer Science. Springer, S. 92–111. ISBN: 978-3-540-68745-0.

Caseware Analytics (2014). *IDEA®*. http://www.casewareanalytics.com/products/idea-data-analysis. [Online; Stand: 22.09.2014].

Cook, Jonathan E., Zhidian Du, Chongbing Liu und Alexander L. Wolf (2004). "Discovering Models of Behavior for Concurrent Workflows". In: *Computers in Industry* 53.3, S. 297–319.

Cook, Jonathan E. und Alexander L. Wolf (1995). "Automating Process Discovery Through Event-data Analysis". In: *Proceedings of the 17th International Conference on Software Engineering*. ICSE '95. Seattle, Washington, USA: ACM, S. 73–82. ISBN: 0-89791-708-1.

— (1998a). "Discovering Models of Software Processes from Event-Based Data". In: *ACM Transactions on Software Engineering and Methodology* 7.3, S. 215–249.

— (1998b). "Event-Based Detection of Concurrency". In: *SIGSOFT FSE*. ACM, S. 35–45.

— (1999). "Software Process Validation: Quantitatively Measuring the Correspondence of a Process to a Model". In: *ACM Transactions on Software Engineering and Methodology* 8.2, S. 147–176. ISSN: 1049-331X.

Cortadella, Jordi, Michael Kishinevsky, A.Kondratyev, L. Lavagno und A. Yakovlev (1997). "Petrify: A Tool for Manipulating Concurrent Specifications and Synthesis of Asynchronous Controllers". In: *IEICE Transactions on Information and Systems* E80-D.3, S. 315–325.

Cortadella, Jordi, Michael Kishinevsky, Luciano Lavagno und Alexandre Yakovlev (1995). "Synthesizing Petri Nets from State-Based models". In: *ICCAD*. Hrsg. von Richard L. Rudell. IEEE Computer Society, S. 164–171. ISBN: 0-8186-7213-7.

— (1998). "Deriving Petri Nets from Finite Transition Systems". In: *IEEE Transactions on Computers* 47.8, S. 859–882.

Curtis, Bill, Marc I. Kellner und Jim Over (1992). "Process Modeling". In: *Communications of the ACM* 35.9, S. 75–90.

Daniel, Florian, Jianmin Wang und Barbara Weber, Hrsg. (2013). *Business Process Management - 11th International Conference, BPM 2013, Beijing, China, August 26-30, 2013. Proceedings*. Bd. 8094. Lecture Notes in Computer Science. Springer. ISBN: 978-3-642-40175-6.

Datta, Anindya (1998). "Automating the Discovery of AS-IS Business Process Models: Probabilistic and Algorithmic Approaches". In: *Information Systems Research* 9.3, S. 275–301.

Denning, Dorothy E. und Peter J. Denning (1977). "Certification of Programs for Secure Information Flow". In: *Communications of the ACM* 20.7, S. 504–513.

Desel, Jörg und Wolfgang Reisig (1996). "The Synthesis Problem of Petri Nets". In: *Acta Informatica* 33.4, S. 297–315.

Deutsches Institut für Interne Revision e.V., DIIR (2014). *Grundsätze des Internen Kontrollsystems (IKS)*. http://www.diir.de/arbeitskreise/ak09/pruefungshandbuch/iks/grundsaetze-des-internen-kontrollsystems-iks/. [Online; Stand: 22.09.2014].

DoD 5200.28-STD. *Department of Defence Trusted Computer System Evaluation Criteria*. US Department of Defence.

Dongen, Boudewijn F. van und Wil M.P. van der Aalst (2004). "Multi-phase Process Mining: Building Instance Graphs". In: *ER*. Hrsg. von Paolo Atzeni, Wesley W. Chu, Hongjun Lu, Shuigeng Zhou und Tok Wang Ling. Bd. 3288. Lecture Notes in Computer Science. Springer, S. 362–376. ISBN: 3-540-23723-2.

— (2005a). "A Meta Model for Process Mining Data". In: *EMOI-INTEROP*. Hrsg. von Michele Missikoff und Antonio De Nicola. Bd. 160. CEUR Workshop Proceedings. CEUR-WS.org.

— (2005b). "Multi-Phase Mining: Aggregating Instances Graphs into EPC's and Petri Nets". In: *Application of Concurrency to System Design (ACSD), 2012 12th International Conference on*, S. 35–58.

Dongen, Boudewijn F. van, Ana Karla Alves de Medeiros und L. Wen (2009). "Process Mining: Overview and Outlook of Petri Net Discovery Algorithms". In: *Transactions on Petri Nets and Other Models of Concurrency*. Lecture Notes in Computer Science 2. Hrsg. von Kurt Jensen und Wil M.P. van der Aalst, S. 225–242.

Dongen, Boudewijn F. van, Nadia Busi, Giovanni Michele Pinna und Wil M.P. van der Aalst (2007). "An Iterative Algorithm for Applying the Theory of Regions in Process Mining". In: *Workshop on Formal Approaches to Business Processes and Web Services*, S. 36–55.

Dumas, Marlon, Wil M.P. van der Aalst und Arthur H.M. ter Hofstede (2005). *Process-Aware Information Systems: Bridging People and Software Through Process Technology*. Wiley. ISBN: 978-0-471-66306-5.

Dumas, Marlon, Marcello La Rosa, Jan Mendling und Hajo A. Reijers (2013). *Fundamentals of Business Process Management*. Springer, S. I–XXVII, 1–399. ISBN: 978-3-642-33142-8.

Eckert, Claudia (2003). *IT-Sicherheit - Konzepte, Verfahren, Protokolle (2. Aufl.)* Oldenbourg. ISBN: 978-3-486-27205-5.

Ehrenfeucht, Andrzej und Grzegorz Rozenberg (1989a). "Partial (Set) 2-Structures. Part I: Basic Notions and the Representation Problem". In: *Acta Informatica* 27.4, S. 315–342.

— (1989b). "Partial (Set) 2-Structures. Part II: State Spaces of Concurrent Systems". In: *Acta Informatica* 27.4, S. 343–368.

— (1990). "Theory of 2-Structures, Part I: Clans, Basic Subclasses, and Morphisms". In: *Theoretical Computer Science* 70.3, S. 277–303.

Ellis, Clarence A. und Gary J. Nutt (1993). "Modeling and Enactment of Workflow Systems". In: *Application and Theory of Petri Nets*. Hrsg. von Marco Ajmone Marsan. Bd. 691. Lecture Notes in Computer Science. Springer, S. 1–16. ISBN: 3-540-56863-8.

Ellis, Clarence, Karim Keddara und Grzegorz Rozenberg (1995). "Dynamic Change within Workflow Systems". In: *Proceedings of conference on Organizational computing systems*. COCS '95. New York, NY, USA: ACM, S. 10–21.

Fdhila, Walid, Stefanie Rinderle-Ma und Manfred Reichert (2012). "Change Propagation in Collaborative Processes Scenarios". In: *8th International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom 2012, Pittsburgh, PA, USA, October 14-17, 2012*. IEEE, S. 452–461. ISBN: 978-1-4673-2740-4.

Ferraiolo, David F. und D. Richard Kuhn (2009). "Role-Based Access Controls". In: *CoRR* abs/0903.2171.

Ferreira, Diogo R. und Daniel Gillblad (2009). "Discovering Process Models from Unlabelled Event Logs". In: *BPM*. Hrsg. von Umeshwar Dayal, Johann Eder, Jana Koehler und Hajo A. Reijers. Bd. 5701. Lecture Notes in Computer Science. Springer, S. 143–158. ISBN: 978-3-642-03847-1.

Folino, Francesco, Gianluigi Greco, Antonella Guzzo und Luigi Pontieri (2009). "Discovering Expressive Process Models from Noised Log Data". In: *IDEAS*. Hrsg. von Bipin C. Desai, Domenico Saccà und Sergio Greco. ACM International Conference Proceeding Series. ACM, S. 162–172. ISBN: 978-1-60558-402-7.

Frankfurt School of Finance & Management (2011). *Audit Challenge (2011): Fachkonferenz für Revisionsmethodik und Strategische Weiterentwicklung*. https://www.audit-challenge.com/.

Gabler Wirtschaftslexikon, Springer Gabler Verlag (Herausgeber) (2014). *Stichwort: Internes Kontrollsystem (IKS)*. http://wirtschaftslexikon.gabler.de/Archiv/88947/internes-kontrollsystem-iks-v7.html. [Online; Stand: 03.09.2014].

Goedertier, Stijn, David Martens, Bart Baesens, Raf Haesen und Jan Vanthienen (2007). "Process Mining as First-Order Classification Learning on Logs with Negative Events". In: *Business Process Management Workshops*. Hrsg. von Arthur H.M. ter Hofstede, Boualem Benatallah und Hye-Young Paik. Bd. 4928. Lecture Notes in Computer Science. Springer, S. 42–53. ISBN: 978-3-540-78237-7.

Goedertier, Stijn, David Martens, Jan Vanthienen und Bart Baesens (2009). "Robust Process Discovery with Artificial Negative Events". In: *Journal of Machine Learning Research* 10, S. 1305–1340.

Golani, Mati und Shlomit S. Pinter (2003). "Generating a Process Model from a Process Audit Log". In: *Business Process Management*. Hrsg. von Wil M.P. van der Aalst, Arthur H.M. ter Hofstede und Mathias Weske. Bd. 2678. Lecture Notes in Computer Science. Springer, S. 136–151. ISBN: 3-540-40318-3.

Greco, Gianluigi, Antonella Guzzo, Giuseppe Manco und Domenico Saccà (2005). "Mining and Reasoning on Workflows". In: *IEEE Transactions on Knowledge and Data Engineering* 17.4, S. 519–534.

Greco, Gianluigi, Antonella Guzzo, Luigi Pontieri und Domenico Saccà (2004). "Mining Expressive Process Models by Clustering Workflow Traces". In: *PAKDD*. Hrsg. von Honghua Dai, Ramakrishnan Srikant und Chengqi Zhang. Bd. 3056. Lecture Notes in Computer Science. Springer, S. 52–62. ISBN: 3-540-22064-X.

— (2006). "Discovering Expressive Process Models by Clustering Log Traces". In: *IEEE Transactions on Knowledge and Data Engineering* 18.8, S. 1010–1027.

Günther, Christian W. und Wil M.P. van der Aalst (2007). "Fuzzy Mining - Adaptive Process Simplification Based on Multi-perspective Metrics". In: *Business Process Management, 5th International Conference, BPM 2007, Brisbane, Australia, September 24-28, 2007, Proceedings*. Hrsg. von Gustavo Alonso, Peter Dadam und Michael Rosemann. Bd. 4714. Lecture Notes in Computer Science. Springer, S. 328–343. ISBN: 978-3-540-75182-3.

Günther, Christian W., Stefanie Rinderle-Ma, Manfred Reichert, Wil M.P. van der Aalst und Jan Recker (2007). "Using Process Mining to Learn from Process Changes in Evolutionary Systems". In: *Business Process Integration and Management* 1, S. 111–111.

Günther, Christian W. und Anne Rozinat (2012). "Disco: Discover Your Processes". In: *Proceedings of the Demonstration Track of the 10th International Conference on Business Process Management (BPM 2012), Tallinn, Estonia, September 4, 2012*. Hrsg. von Niels Lohmann und Simon Moser. Bd. 940. CEUR Workshop Proceedings. CEUR-WS.org, S. 40–44.

Hand, David J., Padhraic Smyth und Heikki Mannila (2001). *Principles of Data Mining*. Cambridge, MA, USA: MIT Press. ISBN: 0-262-08290-X, 9780262082907.

Herbst, Joachim (1999a). "An Inductive Approach to the Acquisition and Adaptation of Workflow Models". In: *Proceedings of the IJCAI'99 Workshop on Intelligent Workflow and Process Management: The New Frontier for AI in Business*, S. 52–57.

— (1999b). "Inducing Workflow Models from Workflow Instances". In: *Society for Computer Simulation (SCS*, S. 175–182.

— (2000a). "A Machine Learning Approach to Workflow Management". In: *ECML*. Hrsg. von Ramon López de Mántaras und Enric Plaza. Bd. 1810. Lecture Notes in Computer Science. Springer, S. 183–194. ISBN: 3-540-67602-3.

— (2000b). "Dealing with Concurrency in Workflow Induction". In: *European Concurrent Engineering Conference. SCS Europe*.

Herbst, Joachim und Dimitris Karagiannis (1998). "Integrating Machine Learning and Workflow Management to Support Acquisition and Adaption of Workflow Models". In: *DEXA Workshop*, S. 745–752.

— (2004). "Workflow mining with InWoLvE". In: *Computers in Industry* 53.3, S. 245–264.

Herrmann, Gady und Günther Pernul (1999). "Viewing Business-process Security from Different Perspectives". In: *International Journal of Electronic Commerce* 3.3, S. 89–103. ISSN: 1086-4415.

Hill, J. B., M. Pezzini und Y. V. Natis (2008). *Findings: Confusion Remains Regarding BPM Terminologies*. Techn. Ber. ID G00155817. Gartner Research.

Hiraishi, Kunihiko (1994). "Some Complexity Results on Transition Systems and Elementary Net Systems". In: *Theoretical Computer Science* 135.2, S. 361–376.

Hofstede, Arthur H.M. ter, Wil M.P. van der Aalst, Michael Adams und Nick Russell, Hrsg. (2010). *Modern Business Process Automation - YAWL and its Support Environment*. Springer. ISBN: 978-3-642-03120-5. URL: http://www.yawlbook.com/home/.

Hofstede, Arthur H.M. ter, Boualem Benatallah und Hye-Young Paik, Hrsg. (2008). *Business Process Management Workshops, BPM 2007 International Workshops, BPI, BPD, CBP, ProHealth, RefMod, semantics4ws, Brisbane, Australia, September 24, 2007, Revised Selected Papers*. Bd. 4928. Lecture Notes in Computer Science. Springer. ISBN: 978-3-540-78237-7.

Huang, Hejiao und Hélène Kirchner (2009). "Component-Based Security Policy Design with Colored Petri Nets". In: *Semantics and Algebraic Specification*. Hrsg. von Jens Palsberg. Bd. 5700. Lecture Notes in Computer Science. Springer, S. 21–42. ISBN: 978-3-642-04163-1.

— (2011). "Formal Specification and Verification of Modular Security Policy Based on Colored Petri Nets". In: *IEEE Transactions on Dependable and Secure Computing* 8.6, S. 852–865.

— (2013). "Secure Interoperation Design in Multi-Domains Environments Based on Colored Petri Nets". In: *Information Sciences* 221, S. 591–606.

Hull, Richard, Jan Mendling und Stefan Tai, Hrsg. (2010). *Business Process Management - 8th International Conference, BPM 2010, Hoboken, NJ, USA, September 13-16, 2010. Proceedings*. Bd. 6336. Lecture Notes in Computer Science. Springer. ISBN: 978-3-642-15617-5.

Humm, Bernhard G. und Janina Fengel (2012). "Semantics-Based Business Process Model Similarity". In: *Business Information Systems - 15th International Conference, BIS 2012, Vilnius, Lithuania, May 21-23, 2012. Proceedings*. Hrsg. von Witold Abramowicz, Dalia Kriksciuniene und Virgilijus Sakalauskas. Bd. 117. Lecture Notes in Business Information Processing. Springer, S. 36–47. ISBN: 978-3-642-30358-6.

Hwang, San-Yih und Wan-Shiou Yang (2002). "On the Discovery of Process Models from their Instances". In: *Decision Support Systems* 34.1, S. 41–57.

ISO/IEC 19505-1:2012. *Information technology – Object Management Group Unified Modeling Language (OMG UML) – Part 1: Infrastructure*. International Organization for Standardization. Geneva, Switzerland.

ISO/IEC 19505-2:2012. *Information technology – Object Management Group Unified Modeling Language (OMG UML) – Part 2: Superstructure*. International Organization for Standardization. Geneva, Switzerland.

ISO/IEC 19510:2013. *Information technology – Object Management Group Business Process Model and Notation*. International Organization for Standardization. Geneva, Switzerland.

ISO/IEC 27001:2013. *Information technology – Security techniques – Information security management systems – Requirements*. International Organization for Standardization. Geneva, Switzerland.

ITAF third Edition. *ITAF - A Professional Practices Framework for IS Audit/Assurance.* International Systems audit and Control Association (ISACA).

Jablonski, Stefan und Christoph Bussler (1996). *Workflow Management - Modeling Concepts, Architecture and Implementation.* International Thomson, S. I–X, 1–351. ISBN: 978-1-85032-222-1.

Jager, Tibor, Sebastian Schinzel und Juraj Somorovsky (2012). "Bleichenbacher's Attack Strikes again: Breaking PKCS#1 v1.5 in XML Encryption". In: *Computer Security - ESORICS 2012 - 17th European Symposium on Research in Computer Security, Pisa, Italy, September 10-12, 2012. Proceedings.* Hrsg. von Sara Foresti, Moti Yung und Fabio Martinelli. Bd. 7459. Lecture Notes in Computer Science. Springer, S. 752–769. ISBN: 978-3-642-33166-4.

Jans, Mieke und Michael Alles (2010). "Process Mining of Event Logs in Auditing: Opportunities and Challenges". In: Rutgers Business School. Newark, NY, USA.

Jans, Mieke, Benoît Depaire und Koen Vanhoof (2011). "Does Process Mining Add to Internal Auditing? An Experience Report". In: *Enterprise, Business-Process and Information Systems Modeling - 12th International Conference, BPMDS 2011, and 16th International Conference, EMMSAD 2011, held at CAiSE 2011, London, UK, June 20-21, 2011. Proceedings.* Hrsg. von Terry A. Halpin, Selmin Nurcan, John Krogstie, Pnina Soffer, Erik Proper, Rainer Schmidt und Ilia Bider. Bd. 81. Lecture Notes in Business Information Processing. Springer, S. 31–45. ISBN: 978-3-642-21758-6.

Jans, Mieke, Nadine Lybaert, Koen Vanhoof und Jan Martijn van der Werf (2009). "A framework for Internal Fraud Risk Reduction at IT Integrating Business Processes". In: *International Journal of Digital Accounting Research.* Bd. 9, S. 1–29.

Jiang, Yixin, Chuang Lin, Hao Yin und Zhangxi Tan (2004). "Security Analysis of Mandatory Access Control Model". In: *Systems, Man and Cybernetics, 2004 IEEE International Conference on.* Bd. 6, S. 5013–5018.

Katt, Basel, Michael Hafner und Xinwen Zhang (2009). "A Usage Control Policy Specification with Petri Nets". In: *CollaborateCom.* IEEE, S. 1–8. ISBN: 978-963-9799-76-9.

Keller, Gerhard, Markus Nüttgens und August-Wilhelm Scheer (1992). "Semantische Prozeßmodellierung auf der Grundlage Ereignisgesteuerter Prozeßketten (EPK)". In: *Veröffentlichungen des Instituts für Wirtschaftsinformatik.* Heft 89. Saarbrücken.

Kindler, Ekkart, Vladimir Rubin und Wilhelm Schäfer (2006). "Process Mining and Petri Net Synthesis". In: *Business Process Management Workshops.* Hrsg. von Johann Eder und Schahram Dustdar. Bd. 4103. Lecture Notes in Computer Science. Springer, S. 105–116. ISBN: 3-540-38444-8.

Knolmayer, Gerhard, Rainer Endl und Marcel Pfahrer (2000). "Modeling Processes and Workflows by Business Rules". In: *Business Process Management, Models, Techniques, and Empirical Studies*. Springer-Verlag, S. 16–29.

Leoni, Massimiliano de und Wil M.P. van der Aalst (2013a). "Aligning Event Logs and Process Models for Multi-perspective Conformance Checking: An Approach Based on Integer Linear Programming". In: *Business Process Management - 11th International Conference, BPM 2013, Beijing, China, August 26-30, 2013. Proceedings*. Hrsg. von Florian Daniel, Jianmin Wang und Barbara Weber. Bd. 8094. Lecture Notes in Computer Science. Springer, S. 113–129. ISBN: 978-3-642-40175-6.

— (2013b). "Data-aware Process Mining: Discovering Decisions in Processes Using Alignments". In: *Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, Coimbra, Portugal, March 18-22, 2013*. Hrsg. von Sung Y. Shin und José Carlos Maldonado. ACM, S. 1454–1461. ISBN: 978-1-4503-1656-9.

Leoni, Massimiliano de, Wil M.P. van der Aalst und Boudewijn F. van Dongen (2012). "Data- and Resource-Aware Conformance Checking of Business Processes". In: *BIS*. Hrsg. von Witold Abramowicz, Dalia Kriksciuniene und Virgilijus Sakalauskas. Bd. 117. Lecture Notes in Business Information Processing. Springer, S. 48–59. ISBN: 978-3-642-30358-6.

Li, Jiafei, Dayou Liu und Bo Yang (2007). "Process Mining: Extending α-Algorithm to Mine Duplicate Tasks in Process Logs". In: *APWeb/WAIM Workshops*. Hrsg. von Kevin Chen-Chuan Chang, Wei Wang, Lei Chen, Clarence A. Ellis, Ching-Hsien Hsu, Ah Chung Tsoi und Haixun Wang. Bd. 4537. Lecture Notes in Computer Science. Springer, S. 396–407. ISBN: 978-3-540-72908-2.

Lima Bezerra, Fábio de, Jacques Wainer und Wil M.P. van der Aalst (2009). "Anomaly Detection Using Process Mining". In: *BMMDS/EMMSAD*. Hrsg. von Terry A. Halpin, John Krogstie, Selmin Nurcan, Erik Proper, Rainer Schmidt, Pnina Soffer und Roland Ukor. Bd. 29. Lecture Notes in Business Information Processing. Springer, S. 149–161. ISBN: 978-3-642-01861-9.

Livshits, Benjamin, Aditya Nori, Sriram Rajamani und Anindya Banerjee (2009). "Merlin: Specification inference for explicit information flow problems". In: *Proceedings of the 2009 ACM SIGPLAN Conference on Programming Language Design and Implementation*. Hrsg. von Michael Hind und Amer Diwan. ACM, S. 75–86.

Lorenz, Robert, Sebastian Mauser und Gabriel Juhás (2007). "How to Synthesize Nets From Languages: A Survey". In: *Winter Simulation Conference*. Hrsg. von Shane G. Henderson, Bahar Biller, Ming-Hua Hsieh, John Shortle, Jeffrey D. Tew und Russell R. Barton. WSC, S. 637–647. ISBN: 1-4244-1306-0.

Lu, Ruopeng und Shazia Wasim Sadiq (2007). "A Survey of Comparative Business Process Modeling Approaches". In: *BIS*. Hrsg. von Witold Abramowicz. Bd. 4439. Lecture Notes in Computer Science. Springer, S. 82–94. ISBN: 978-3-540-72034-8.

Luengo, Daniela und Marcos Sepúlveda (2011a). "Applying Clustering in Process Mining to Find Different Versions of a Business Process That Changes over Time". In: *Business Process Management Workshops (1)*. Hrsg. von Florian Daniel, Kamel Barkaoui und Schahram Dustdar. Bd. 99. Lecture Notes in Business Information Processing. Springer, S. 153–158. ISBN: 978-3-642-28107-5.

— (2011b). *Detection of Temporal Changes in Business Processes using Clustering Techniques*. Techn. Ber. Pontificia Universidad Católica de Chile, School of Engineering, Computer Science Department.

Man, Henk de (2009). *Case Management: A Review of Modeling Approaches*. BPTrends Report. Available at http://www.bptrends.com/.

Mannila, Heikki und Christopher Meek (2000). "Global Partial Orders From Sequential Data". In: *KDD*. Hrsg. von Raghu Ramakrishnan, Salvatore J. Stolfo, Roberto J. Bayardo und Ismail Parsa. ACM, S. 161–168. ISBN: 1-58113-233-6.

Milner, Robin (1989). *Communication and Concurrency*. PHI Series in computer science. Prentice Hall, S. I–XI, 1–260. ISBN: 978-0-13-115007-2.

Mitchell, Tom M. (1997). *Machine Learning*. McGraw Hill series in computer science. McGraw-Hill. ISBN: 978-0-07-042807-2.

Mukund, Madhavan (1992). "Petri Nets and Step Transition Systems". In: *International Journal of Foundations of Computer Science* 3.4, S. 443–478.

Müller, Günter und Rafael Accorsi (2013). "Why Are Business Processes Not Secure?" In: *Number Theory and Cryptography - Papers in Honor of Johannes Buchmann on the Occasion of His 60th Birthday*. Hrsg. von Marc Fischlin und Stefan Katzenbeisser. Bd. 8260. Lecture Notes in Computer Science. Springer, S. 240–254. ISBN: 978-3-642-42000-9.

Müller, Jens, Jutta A. Mülle, Silvia von Stackelberg und Klemens Böhm (2010). "Secure Business Processes in Service-Oriented Architectures - A Requirements Analysis". In: *8th IEEE European Conference on Web Services (ECOWS 2010), 1-3 December 2010, Ayia Napa, Cyprus*. Hrsg. von Antonio Brogi, Cesare Pautasso und George Angelos Papadopoulos. IEEE Computer Society, S. 35–42. ISBN: 978-1-4244-9397-5.

Munoz-Gama, Jorge und Josep Carmona (2010). "A Fresh Look at Precision in Process Conformance". In: *BPM*. Hrsg. von Richard Hull, Jan Mendling und Stefan Tai. Bd. 6336. Lecture Notes in Computer Science. Springer, S. 211–226. ISBN: 978-3-642-15617-5.

Needleman, Saul B. und Christian D. Wunsch (1970). "A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of two Proteins". In: *Journal of Molecular Biology* 48.3, S. 443–453. ISSN: 0022-2836.

Nerode, Anil (1958). "Linear Automaton Transformations". In: *Proceedings of the American Mathematical Society* 9.4, pages. ISSN: 00029939.

Ni, Qun, Elisa Bertino und Jorge Lobo (2010). "Risk-based Access Control Systems Built on Fuzzy Inferences". In: *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2010*. Hrsg. von Dengguo Feng, David A. Basin und Peng Liu. Beijing, China: ACM, S. 250–260. ISBN: 978-1-60558-936-7.

Nielsen, Mogens, Grzegorz Rozenberg und P. S. Thiagarajan (1992). "Elementary Transition Systems". In: *Theoretical Computer Science* 96.1, S. 3–33.

OASIS (2007). *Web Services Business Process Execution Language Version 2.0*. OASIS. Organization for the Advancement of Structured Information Standards.

Ozip-Philippsen, Sergej (2013). "Compliance-Management und Risikomanagement". In: *ZRFC - Risk, Fraud & Compliance* 8.05, S. 204–209.

Park, Jaehong und Ravi Sandhu (2002). "Towards Usage Control Models: Beyond Traditional Access Control". In: *ACM symposium on Access control models and technologies*. SACMAT '02. ACM, S. 57–64.

Pinter, Shlomit S. und Mati Golani (2004). "Discovering Workflow Models From Activities' Lifespans". In: *Computers in Industry* 53.3, S. 283–296.

Pretschner, Alexander, Manuel Hilty und David Basin (2006). "Distributed Usage Control". In: *Communications of the ACM* 49.9, S. 39–44.

PwC (2014). *Global Economic Crime Survey 2014*. Techn. Ber. PriceWaterhouseCoopers International Limited.

Reichert, Manfred und Peter Dadam (1998). "ADEPT$_{\text{flex}}$-Supporting Dynamic Changes of Workflows Without Losing Control". In: *Intelligent Information Systems* 10.2, S. 93–129.

Rinderle-Ma, Stefanie, Shazia Wasim Sadiq und Frank Leymann, Hrsg. (2010). *Business Process Management Workshops, BPM 2009 International Workshops, Ulm, Germany, September 7, 2009. Revised Papers*. Bd. 43. Lecture Notes in Business Information Processing. Springer. ISBN: 978-3-642-12185-2.

Rittgen, Peter (2011). "Business Process Model Similarity as a Proxy for Group Consensus". In: *The Practice of Enterprise Modeling - 4th IFIP WG 8.1 Working Conference, PoEM 2011 Oslo, Norway, November 2-3, 2011 Proceedings*. Hrsg. von Paul Johannesson, John Krogstie und Andreas L. Opdahl. Bd. 92. Lecture Notes in Business Information Processing. Springer, S. 12–24. ISBN: 978-3-642-24848-1.

Rozinat, Anne und Wil M.P. van der Aalst (2006). *Decision Mining in Business Processes*. BETA Working Paper Series NL-5600. Eindhoven University of Technology.

— (2008). "Conformance Checking Of Processes Based On Monitoring Real Behavior". In: *Information Systems Journal* 33.1, S. 64–95.

Sandhu, Ravi und Pierangela Samarati (1994). "Access Control: Principles and Practice". In: *IEEE Communications Magazine* 32.9, S. 40–48.

Sayana, S. Anantha (2003). "Using CAATs to Support IS Audit". In: *Information Systems Control Journal* 1.

Schimm, Guido (2003). "Mining Most Specific Workflow Models from Event-Based Data". In: *Business Process Management*. Hrsg. von Wil M.P. van der Aalst, Arthur H.M. ter Hofstede und Mathias Weske. Bd. 2678. Lecture Notes in Computer Science. Springer, S. 25–40. ISBN: 3-540-40318-3.

— (2004). "Mining Exact Models of Concurrent Workflows". In: *Computers in Industry* 53.3, S. 265–281.

Schonenberg, Helen, Ronny Mans, Nick Russell, Nataliya Mulyar und Wil M.P. van der Aalst (2008). "Process Flexibility: A Survey of Contemporary Approaches". In: *CIAO! / EOMAS*. Hrsg. von Jan L. G. Dietz, Antonia Albani und Joseph Barjis. Bd. 10. Lecture Notes in Business Information Processing. Springer, S. 16–30. ISBN: 978-3-540-68643-9.

Silva, Ricardo, Jiji Zhang und James G. Shanahan (2005). "Probabilistic Workflow Mining". In: *KDD*. Hrsg. von Robert Grossman, Roberto J. Bayardo und Kristin P. Bennett. ACM, S. 275–284.

Sinur, Jim (2002). *The BPA Market Catches Another Major Updraft*. Techn. Ber. Gartner's Application Development & Maintenance Research Note M-16-8153. Gartner Research.

Société Générale - General Inspection Services (2008). *Mission Green: Summary Report*. Techn. Ber. Société Générale.

Solé, Marc und Josep Carmona (2010a). "Process Mining from a Basis of State Regions". In: *Petri Nets*. Hrsg. von Johan Lilius und Wojciech Penczek. Bd. 6128. Lecture Notes in Computer Science. Springer, S. 226–245. ISBN: 978-3-642-13674-0.

— (2010b). "Rbminer: A Tool for Discovering Petri Nets from Transition Systems". In: *Proceedings of the 8th International Conference on Automated Technology for Verification and Analysis*. ATVA'10. Singapore: Springer-Verlag, S. 396–402. ISBN: 3-642-15642-8, 978-3-642-15642-7.

— (2012a). "A High-Level Strategy for C-net Discovery". In: *Application of Concurrency to System Design (ACSD), 2012 12th International Conference on*, S. 102–111.

— (2012b). "An SMT-Based Discovery Algorithm for C-Nets". In: *Application and Theory of Petri Nets*. Hrsg. von Serge Haddad und Lucia Pomello. Bd. 7347. Lecture Notes in Computer Science. Springer Berlin Heidelberg, S. 51–71. ISBN: 978-3-642-31130-7.

— (2013). "Region-Based Foldings in Process Discovery". In: *IEEE Transactions on Knowledge and Data Engineering* 25.1, S. 192–205.

Song, Minseok, Christian W. Günther und Wil M.P. van der Aalst (2008). "Trace Clustering in Process Mining". In: *Business Process Management Workshops*. Hrsg. von Danilo Ardagna, Massimo Mecella und Jian Yang. Bd. 17. Lecture Notes in Business Information Processing. Springer, S. 109–120. ISBN: 978-3-642-00327-1.

Stocker, Thomas und Frank Böhr (2013). "IF-Net: A Meta-Model for Security-Oriented Process Specification". In: *STM*. Hrsg. von Rafael Accorsi und Silvio Ranise. Bd. 8203. Lecture Notes in Computer Science. Springer, S. 191–206. ISBN: 978-3-642-41097-0.

Tiwari, Ashutosh, Chris J. Turner und Basim Majeed (2008). "A Review of Business Process Mining: State-of-the-Art and Future Trends". In: *Business Process Management Journal* 14.1, S. 5–22.

Veiga, Gabriel M. und Diogo R. Ferreira (2009). "Understanding Spaghetti Models with Sequence Clustering for ProM". In: *Business Process Management Workshops*. Hrsg. von Stefanie Rinderle-Ma, Shazia Wasim Sadiq und Frank Leymann. Bd. 43. Lecture Notes in Business Information Processing. Springer, S. 92–103. ISBN: 978-3-642-12185-2.

Warner, Janice und Vijayalakshmi Atluri (2006). "Inter-Instance Authorization Constraints for Secure Workflow Management". In: *SACMAT 2006,11th ACM Symposium on Access Control Models and Technologies, Lake Tahoe, California, USA, June 7-9, 2006, Proceedings*. Hrsg. von David F. Ferraiolo und Indrakshi Ray. ACM, S. 190–199. ISBN: 1-59593-353-0.

Weber, Barbara, Stefanie Rinderle und Manfred Reichert (2013). "Change Patterns and Change Support Features in Process-Aware Information Systems". In: *Seminal Contributions to Information Systems Engineering*. Hrsg. von Janis A. Bubenko Jr., John Krogstie, Oscar Pastor, Barbara Pernici, Colette Rolland und Arne Sølvberg. Springer, S. 381–395. ISBN: 978-3-642-36925-4, 978-3-642-36926-1.

Weber, Barbara, Werner Wild und Ruth Breu (2004). "CBRFlow: Enabling Adaptive Workflow Management Through Conversational Case-Based Reasoning". In: *ECCBR*. Hrsg. von Peter Funk und Pedro A. González-Calero. Bd. 3155. Lecture Notes in Computer Science. Springer, S. 434–448. ISBN: 3-540-22882-9.

Weerdt, Jochen De, Manu De Backer, Jan Vanthienen und Bart Baesens (2012). "A Multi-Dimensional Quality Assessment of State-of-the-Art Process Discovery Algorithms using Real-life Event Logs". In: *Information Systems Journal* 37.7, S. 654–676.

Weidlich, Matthias, Remco M. Dijkman und Mathias Weske (2012). "Behaviour Equivalence and Compatibility of Business Process Models with Complex Correspondences". In: *The Computer Journal* 55.11, S. 1398–1418.

Weijters, A.J.M.M. (Ton) und Wil M.P. van der Aalst (2003). "Rediscovering Workflow Models from Event-Based Data using Little Thumb". In: *Integrated Computer-Aided Engineering* 10.2, S. 151–162.

Weijters, A.J.M.M. (Ton), Wil M.P. van der Aalst und Ana Karla Alves de Medeiros (2006). *Process Mining with the Heuristics Miner Algorithm*. BETA Working Paper Series WP 166. Eindhoven University of Technology.

Weijters, A.J.M.M. (Ton) und Joel T.S. Ribeiro (2011). "Flexible Heuristics Miner (FHM)". In: *CIDM*. IEEE, S. 310–317. ISBN: 978-1-4244-9925-0.

Wen, Lijie, Wil M.P. van der Aalst, Jianmin Wang und Jiaguang Sun (2007). "Mining Process Models with Non-Free-Choice Constructs". In: *Data Mining and Knowledge Discovery* 15.2, S. 145–180.

Wen, Lijie, Jianmin Wang, Wil M.P. van der Aalst, Biqing Huang und Jiaguang Sun (2009). "A Novel Approach for Process Mining Based on Event Types". In: *Journal of Intelligent Information Systems* 32.2, S. 163–190.

Wen, Lijie, Jianmin Wang und Jiaguang Sun (2007). "Mining Invisible Tasks from Event Logs". In: *APWeb/WAIM*. Hrsg. von Guozhu Dong, Xuemin Lin, Wei Wang, Yun Yang und Jeffrey Xu Yu. Bd. 4505. Lecture Notes in Computer Science. Springer, S. 358–365. ISBN: 978-3-540-72483-4.

Werf, Jan Martijn E. M. van der, Boudewijn F. van Dongen, Cor A. J. Hurkens und Alexander Serebrenik (2008). "Process Discovery Using Integer Linear Programming". In: *Petri Nets*. Hrsg. von Kees M. van Hee und Rüdiger Valk. Bd. 5062. Lecture Notes in Computer Science. Springer, S. 368–387. ISBN: 978-3-540-68745-0.

Weske, Mathias (2012). *Business Process Management - Concepts, Languages, Architectures, 2nd Edition*. Springer, S. I–XV, 1–403. ISBN: 978-3-642-28615-5.

Wil M.P. van der Aalst, S. Jablonski (2000). "Dealing with Workflow Change: Identification of Issues and Solutions". In: *International Journal of Computer Systems Science and Engineering* 15.5, S. 267–276.

Winskel, Glynn und Morgens Nielsen (1995). "Handbook of Logic in Computer Science (Vol. 4)". In: Hrsg. von S. Abramsky, Dov M. Gabbay und T. S. E. Maibaum. Oxford, UK: Oxford University Press. Kap. Models for Concurrency, S. 1–148. ISBN: 0-19-853780-8.

Witten, Ian H. und Eibe Frank (1999). *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann. ISBN: 1-55860-552-5.

Wolf, Celia und Paul Harmon (2011). *Business Process Modeling Survey*. BPTrends Report. Available at http://www.bptrends.com/.

— (2012). *The State of Business Process Management 2012*. BPTrends Report. Available at http://www.bptrends.com/.

Zhang, ZhaoLi, Fan Hong und Junguo Liao (2006). "Modeling Chinese Wall Policy Using Colored Petri Nets". In: *CIT*. IEEE Computer Society, S. 162. ISBN: 0-7695-2687-X.

Zhang, ZhaoLi, Fan Hong und Hai-Jun Xiao (2006). "Verification of Strict Integrity Policy via Petri Nets". In: *ICSNC*. IEEE Computer Society, S. 23.