**Albert-Ludwigs-Universität Freiburg**
**Institut für Informatik und Gesellschaft**

---

Dissertation zur Erlangung des
Doktorgrades der Technischen Fakultät der
Albert-Ludwigs-Universität Freiburg im Breisgau

# Mechanismen zur Sicherheitszertifizierung formalisierter Geschäftsprozesse

vorgelegt von

**Claus Hendrik Wonnemann**

aus Bielefeld

Juli 2011

---

| | |
|---|---|
| Dekan: | Prof. Dr. Bernd Becker,<br>Albert-Ludwigs-Universität Freiburg |
| Erstreferent: | Prof. Dr. Günter Müller,<br>Albert-Ludwigs-Universität Freiburg |
| Zweitreferent: | Prof. Dr. Bernd Becker,<br>Albert-Ludwigs-Universität Freiburg |
| Datum der Disputation: | 5. Dezember 2011 |

## Zusammenfassung

Die IT-getriebene Automatisierung geschäftlicher Abläufe ist eng mit der Diskussion um die Informationssicherheit verknüpft. Die Abbildung von Geschäftsprozessen in Workflow-Systemen ermöglicht deren dynamische Integration mit externen Partnern und die Auslagerung von Teilaufgaben auf fremde Infrastrukturen, wie z.B. *Service-Clouds*. Die durch diese Entwicklung realisierbaren wirtschaftlichen Vorteile erwachsen aus einem effizienten Informationsaustausch, einer großen Flexibilität bei der Prozess-Gestaltung und geringen Investitionen für den Aufbau eigener Infrastrukturen. Ihre Kehrseite und Achillesferse sind unzureichende Sicherheitsgarantien bzgl. der verarbeiteten Informationen: Es kann nicht zuverlässig garantiert werden, dass sensible Informationen bei der Verarbeitung auf geteilten Plattformen vor unbefugter Einsicht geschützt sind. In Folge des mangelnden Vertrauens in die Sicherheit werden kritische Prozesse nicht auf externe Infrastrukturen ausgelagert und potentielle wirtschaftliche Vorteile bleiben ungenutzt.

Als Lösungsansatz schlägt diese Arbeit mit InDico ein automatisiertes Vorgehensmodell für die Sicherheitszertifizierung von Geschäftsprozess-Konstellationen vor. Ausgangspunkt sind in der Praxis verwendete Prozess-Beschreibungen, die in InDico in die formal fundierte Sprache IFnet übersetzt werden. Sicherheitsanforderungen werden Strategie-geleitet als Einschränkungen an den erlaubten Informationsfluss in einem IFnet-Modell kodiert. Den Kern des Vorgehensmodells bildet ein statisches Prüfungsverfahren, das den Informationsfluss in einem IFnet-Modell von der Quelle bis zur Senke (Ende-zu-Ende) auf mögliche Sicherheitsverletzungen untersucht. Dabei wird sowohl die Informationsübertragung über Datencontainer (Datenfluss) als auch über sogenannte *verdeckte Kanäle* betrachtet. Auf diese Weise kann InDico zusätzliche Verletzungen erkennen und damit umfassendere Sicherheitsgarantien bereitstellen, als es mit bestehenden Verfahren möglich ist. Die mit InDico generierten Zertifikate bescheinigen oder widerlegen die Erfüllung der gestellten Sicherheitsanforderungen und können als Grundlage für entsprechende Sicherheitsgarantien seitens eines Dienstanbieters gegenüber seinen Nutzern dienen.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

Kapitel 1

# Sichere Geschäftsprozesse als wirtschaftliches Erfordernis

Mit dem Übergang vom Industrie- zum Informationszeitalter ist die effiziente Gewinnung, Verarbeitung und Verbreitung von Informationen für praktisch jedes Unternehmen zu einer Voraussetzung für den geschäftlichen Erfolg geworden. War im Industriezeitalter die Verfügbarkeit menschlicher Arbeit als wichtigstem Produktionsfaktor entscheidend, geht deren direkter Anteil an der Wertschöpfung zurück, während die Bedeutung des Informationsmanagements rasant zunimmt (Rifkin, 1997). In vielen Unternehmen ist der Umgang mit Informationen gar zum wesentlichen Gegenstand der Geschäftstätigkeit geworden (Sackmann und Strüker, 2005).

Wegbereiter für diese Entwicklung und ihre treibende Kraft ist die „Computerisierung" geschäftlicher Abläufe, also ihre Unterstützung durch Systeme zur elektronischen Datenverarbeitung. Wurden diese zunächst für die Beschleunigung relativ einfacher Vorgänge eingesetzt (wie z.B. der Rechnungslegung und Buchhaltung), begann in den 1980er Jahren die vollständige Abbildung komplexer Geschäftsprozesse in IT-Systemen. Mittlerweile wird ein Großteil der betrieblichen Prozesse mit IT-gestützten Managementsystemen ausgeführt (Wolf und Harmon, 2010). Die technische Entwicklung dieser Systeme hat eine weitreichende Flexibilisierung der Gestaltung und Ausführung von Prozessen ermöglicht, durch die diese in Echtzeit an unternehmerische Anforderungen angepasst werden können. Das wirtschaftliche Potential, das damit realisiert wird, entsteht aus verbesserter Effizienz und Agilität eines Unternehmens und damit reduzierten Kosten und Risiken (Etro, 2009; Woods und Mattern, 2006).

Durch die Verbreitung des Internets in den letzten fünfzehn Jahren ist eine neue Stufe der Automatisierung von Geschäftsabläufen erreicht worden, die durch die Kopplung von Prozessen mittels organisationsübergreifender Computernetzwerke geprägt ist. Während Prozesse früher vorwiegend innerhalb eines Unternehmens ausgeführt wurden, werden sie

heute eng mit externen Partnern verzahnt (Hammer, 2010). Beispiele einer solchen Verzahnung reichen vom einfachen Austausch aktueller Lagerbestände zwischen den Partnern einer Lieferkette bis hin zur dynamischen Komposition ganzer Prozesse aus den Dienstleistungen externer Anbieter, die automatisch ausgewählt werden. Den vorläufigen Höhepunkt der Entwicklung zu derartigen über Netzwerke abrufbaren Diensten bilden öffentliche *(Service-)Clouds*, die jedem Kunden gegen Bezahlung spezifische Leistungen – angefangen bei bloßer Rechen- oder Speicherkapazität bis hin zur komplexen Auftragsbearbeitung – bereitstellen.

Ein Artikel aus der Wochenzeitung „Die Zeit“ illustriert anschaulich die Potentiale derartig vernetzter Geschäftsabläufe: In dem beschriebenen Szenario tritt ein deutscher Kleinbetrieb als Anbieter von Lebensmitteln auf, die er über eine zentrale Dienst-Plattform vertreibt. Durch die so erreichte Markttransparenz kann er seine Waren zu Höchstpreisen in Asien absetzen. Die für die Abwicklung notwendigen Arbeitsschritte – von der Distribution und Lagerung über die Rechtsberatung bis hin zur Zollabwicklung – werden von externen Anbietern bezogen, deren Leistungen auf der Plattform zu einem Gesamtprozess zusammengesetzt werden (Asendorpf, 2011). Die Vorteile einer solchen Lösung sind evident: Der Verkäufer kann bedarfsgerecht die Leistungen abrufen, die er benötigt, und seine Ware zu günstigeren Konditionen verkaufen, als er dies ohne den Einsatz moderner Informationstechnologien könnte.

Die Kehrseite und Achillesferse derartiger Anwendungen ist die unzureichende Absicherung der verarbeiteten Informationen. Durch die Öffnung von Geschäftsprozessen gegenüber externen Parteien verliert ein Teilnehmer gleichzeitig auch die unmittelbare Kontrolle über die dabei ausgetauschten Daten. Die ehemals klaren Grenzen zwischen einer vertrauenswürdigen „inneren Welt“ und einer potentiell feindlichen „äußeren Welt“ verschwimmen zusehends: Auf einer öffentlichen Dienst-Plattform – oder *(Service-)Cloud* – laufen die Informationen vieler unterschiedlicher (und untereinander nicht bekannter) Teilnehmer zusammen und verschmelzen ggf. zu gemeinsamen Anwendungen. Hier kann Schaden für Unternehmen entstehen, wenn sensible Informationen oder geistiges Eigentum öffentlich gemacht werden. Die Persönlichkeitsrechte von Individuen werden verletzt, wenn personenbezogene Daten von unberechtigten Parteien eingesehen werden können.

Als Konsequenz aus dem mangelnden Vertrauen in die Sicherheit werden kritische Geschäftsabläufe nicht auf geteilte Infrastrukturen ausgelagert: So zitiert der oben genannte „Zeit“-Artikel einen IT-Verantwortlichen des Großunternehmens Continental AG mit den Worten, dass aus Sicherheitsgründen nur 10 bis 15 Prozent der IT-Prozesse prinzipiell für das Cloud-Computing geeignet seien. Die Aussage wird von einer aktuellen Untersuchung gestützt, nach der Sicherheitsbedenken derzeit das größte Hemmnis für den betrieblichen Einsatz solcher Infrastrukturen darstellen (Mather u. a., 2009). Die Frage nach dem wirtschaftlichen Potential, das mit dem Einsatz dieser neuen Technologien verbunden wird, ist somit unmittelbar verknüpft mit der Herausforderung, die Sicherheit von Geschäftsprozessen garantieren zu können.

## 1.1 Wirtschaftliche Aktivität erfordert geprüfte Regeleinhaltung

Anforderungen bzgl. der Sicherheit von Informationen[1] machen einen erheblichen Teil der Vorschriften und Richtlinien aus, die ein Unternehmen zur Ausübung seiner Geschäftstätigkeit einhalten muss. Die Pflicht zur Einhaltung wird üblicherweise mit dem Begriff *Compliance* umschrieben, der dem anglo-amerikanischen Rechtskreis entstammt (Vetter, 2008). Obwohl dieser Sachverhalt keinesfalls neu ist, hat er durch eine Reihe von Betrugsskandalen eines bis dahin ungekannten Ausmaßes – prominent waren z.B. die Fälle *Enron* (Fox, 2003) und *WorldCom* (Malik, 2003) in den USA – in den vergangenen Jahren ein besonderes Augenmerk erfahren. Im Zuge der Aufarbeitung wurden in vielen Ländern neue Gesetze erlassen und bestehende Vorschriften verschärft. Diese sollen sicherstellen, dass Unternehmen Strukturen schaffen, die Compliance fördern und Verantwortlichkeiten im Falle von Verletzungen festlegen. Verstöße gegen entsprechende Gesetze werden mit teils empfindlichen Strafen belegt, die von Geldbußen über Haftstrafen für verantwortliche Führungskräfte bis zur zeitweiligen Einstellung der Unternehmenstätigkeit reichen (Sarbanes-Oxley Act, 2002).

Neben Gesetzen entstammen Sicherheitsanforderungen an geschäftliche Abläufe aus Abmachungen mit Geschäftspartnern oder können vom Unternehmen selbst auferlegt sein. Der Deutsche Corporate Governance-Kodex definiert Compliance als „Einhaltung der gesetzlichen Bestimmungen und der unternehmensinternen Richtlinien“, für die der Vorstand Sorge zu tragen hat (Deutscher Corporate Governance-Kodex, 2010, Ziffer 4.1.3). Bace und Rozwell (2006) differenzieren zwischen drei Formen von Compliance, die sich im Ursprung der einzuhaltenden Anforderungen unterscheiden:

- *Regulatory Compliance* bezeichnet die Einhaltung der ein Unternehmen betreffenden Gesetze.
- *Commercial Compliance* bezeichnet die Einhaltung von Richtlinien, die die Beziehungen mit Geschäftspartnern und Kunden betreffen.
- *Organizational Compliance* bezeichnet die Befolgung interner Richtlinien, die ein Unternehmen sich selbst auferlegt.

Beispiele für Sicherheitsanforderungen, deren Einhaltung zur ersten Kategorie von Compliance gehört, finden sich etwa im Bundesdatenschutzgesetz, das die Verwendung von Kunden- und Mitarbeiterdaten regelt (Bundesdatenschutzgesetz, 2003). Die Befolgung Industrie-spezifischer Richtlinien, wie beispielsweise des für die Kreditkartenindustrie geltenden *Payment Card Industry Data Security Standard* (PCI DSS, 2010) sind der zweiten Kategorie zuzurechnen. Selbstverpflichtungen von Unternehmen, etwa zur Einhaltung gesetzlich nicht zwingend vorgeschriebener Datenschutzanforderungen (Briegleb, 2008), gehören zur dritten Kategorie.

[1] Im Folgenden wird der Begriff „Sicherheitsanforderungen“ ausschließlich in Bezug auf Informationen, und nicht etwa bzgl. der Sicherheit von Menschen oder Gegenständen, verwendet.

Die Einhaltung von Sicherheitsanforderungen bei der Geschäftstätigkeit ist die Voraussetzung, um wirtschaftliche oder persönliche Schäden, die durch den Verlust oder die Manipulation von Informationen erwachsen können, zu vermeiden. Das Instrument zur Vermeidung dieser Schäden ist die regelmäßige, systematische Prüfung der geschäftlichen Abläufe auf Möglichkeiten zur Verletzung der gestellten Anforderungen.

### 1.1.1 Zertifizierung geschäftlicher Abläufe

Die Überprüfung der geschäftlichen Abläufe eines Unternehmens ist hinsichtlich *Regulatory Compliance* ein gesetzlich vorgeschriebener Vorgang, der in regelmäßigen Abständen und durch unabhängige Parteien vorgenommen werden muss. Auch im Hinblick auf *Commercial* und *Organizational Compliance* ist die periodische Prüfung vorgesehen, um Möglichkeiten der Verletzung zu erkennen und die damit verbundenen Schäden zu vermeiden. Die Ergebnisse solcher Prüfungen sind *Zertifikate*, die die Erfüllung der gestellten Anforderungen bescheinigen bzw. widerlegen. Der Prüfungsvorgang mit dem Ziel der Ausstellung eines Zertifikates wird daher im Folgenden als Zertifizierung bezeichnet.

Für die Zertifizierung gibt es zwei grundsätzliche Herangehensweisen, die sich bzgl. des Prüfungszeitpunktes unterscheiden:

1. **Zertifizierung auf der Basis von Modellen (a priori):** Bei dieser Art der Zertifizierung wird ein Modell, das einen geschäftlichen Ablauf beschreibt, auf Möglichkeiten der Verletzung gestellter Anforderungen untersucht. Wird keine solche Möglichkeit entdeckt, kann davon ausgegangen werden, dass bei der Durchführung des modellierten Ablaufes auch keine Verletzungen auftreten.[2] Der Vorteil dieser Herangehensweise ist die Vermeidung von Verletzungen, da diese bereits vor der Durchführung eines Geschäftsvorgangs auf der Basis des Modells erkannt werden können.

2. **Zertifizierung auf der Basis von Aufzeichnungen (a posteriori):** Hier werden im Laufe der Durchführung eines geschäftlichen Vorgangs aufgezeichnete Artefakte – wie erstellte Dokumente, Buchungen, Schriftverkehr – *nachträglich* auf Verletzungen der gestellten Anforderungen geprüft. Ziel der Zertifizierung bei dieser Herangehensweise kann nicht die Vermeidung von Verletzungen sein, sondern ist ihre zuverlässige Aufdeckung und die Identifikation ihrer Verursacher.

Beide Herangehensweisen sind in der Praxis von großer Bedeutung. Das *Statement on Auditing Standards No. 70* des *American Institute of Certified Public Accountants*, welches Zertifizierungsrichtlinien für die Prüfung ausgelagerter Geschäftsprozesse beschreibt, sieht zwei Typen von Zertifikaten vor (AICPA, 1993). Der erste Typ (*Report Type I*) umfasst eine Bewertung des Prüfers hinsichtlich der eingerichteten Maßnahmen zur

[2] Hier besteht die Annahme, dass keine Abweichungen zwischen dem modellierten und dem tatsächlich ausgeführten Ablauf entstehen können.

Erkennung von Anforderungsverletzungen und ist damit das Ergebnis einer a-priori-Zertifizierung. Der zweite Typ (*Report Type II*) umfasst zusätzlich eine (nachträgliche) Bewertung der Einhaltung in einem gegebenen Untersuchungszeitraum.

Diese Arbeit befasst sich mit der a-priori-Zertifizierung von Geschäftsprozessen auf der Basis von Modellen. Voraussetzung dafür ist naturgemäß die Existenz solcher Modelle, die den jeweiligen Prozess in hinreichender Genauigkeit abbilden. Diese sind durch Workflow-Modelle, die Geschäftsprozesse in einer durch einen Computer les- und ausführbaren Form darstellen, gegebenen (siehe Abschnitt 1.2).

### 1.1.2 Notwendigkeit automatisierter Prüfung

Die Entwicklung automatisierter Zertifizierungsverfahren hat mit der Entwicklung automatisierter Geschäftsprozesse bisher nicht Schritt halten können: Gegenwärtig ist die Compliance-Prüfung von betrieblichen IT-Systemen ein überwiegend manueller Vorgang, bei dem Prüfer die IT-Einrichtungen und Prozesse eines Unternehmens inspizieren und deren Konformität mit Compliance-Anforderungen bewerten (Carlin und Gallegos, 2007; Bace u. a., 2006). Diese Praxis ist aus folgenden Gründen nachteilig:

- **Hohe Kosten.** Die Kosten, die ein Unternehmen für die Erfüllung von Compliance-Auflagen tragen muss, können einen merklichen Anteil am Umsatz ausmachen. Die *Small Business Administration* der US-Regierung gibt an, dass die Ausgaben amerikanischer Unternehmen, die für die Einhaltung gesetzlicher Bestimmungen aufgebracht werden müssen, bis zu acht Prozent des Bruttoinlandsprodukts ausmachen (Bace u. a., 2006). Compliance-Kosten entstehen vor allem durch Umstrukturierungen in der Organisation, zusätzliche Personalaufwendungen, Produktivitätseinbußen und insbesondere Gebühren für periodisch stattfindende Prüfungen durch externe Gutachter (Audits) (Hartman, 2006). Im Zeitraum von 2002 bis 2008 hat sich der Anteil der Audit-Gebühren an den Gesamtkosten, den börsennotierte US-amerikanische Unternehmen für Compliance aufwenden müssen, von knapp 50% auf annähernd 80% ausgeweitet (Cheffers und Whalen, 2010). Besonders betroffen sind kleine und mittelständische Unternehmen, bei denen sowohl die Höhe der aufzuwenden Kosten für Compliance als auch der Anteil von Audit-Aufwendungen an diesen Kosten überdurchschnittlich stark angestiegen sind (Hartman, 2006).
- **Lange Prüfungsintervalle.** Manuelle Zertifizierung kostet viel Zeit und kann nicht in beliebig kurzen Abständen wiederholt werden. So beträgt beispielsweise das vorgeschriebene Prüfungsintervall im Fall des *Sarbanes-Oxley Act* ein Jahr. Da IT-gestützte Geschäftsprozesse keine statischen Artefakte sind, sondern einer stetigen Anpassung an betriebswirtschaftliche Anforderungen unterliegen (Reichert und Dadam, 1998), sind derartige Abstände zu lang um Verletzungen zeitnah erkennen zu können. Automatisierte Verfahren sind daher notwendig, die sich transparent in die geschäftlichen Abläufe einbinden lassen und den Prüfungsvorgang an die Prozesse und deren Veränderung koppeln. Die so verkürzten Kontrollzeiträume erhöhen

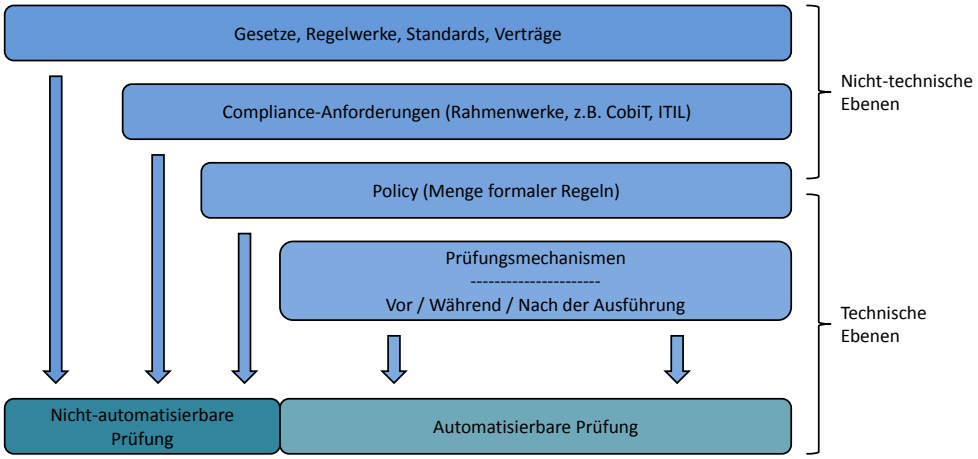



Abbildung 1.1.: Vorgehensmodell zur Formalisierung von Compliance-Anforderungen nach Sackmann und Kähmer (2008).

die Chance auf Compliance, da Verletzungen früher erkannt und diesen unmittelbar entgegengewirkt werden kann (Liebenau und Kärrberg, 2006; Sackmann, 2008).

- **Unvollständige Zertifikate.** Bei der manuellen Zertifizierung werden aufgrund des Umfangs der zu untersuchenden Daten üblicherweise Stichproben genommen (Carlin und Gallegos, 2007). Die ausgestellten Zertifikate sind somit zwangsläufig unvollständig und lassen Verletzungen potentiell unberücksichtigt.

Für die automatisierte Zertifizierung müssen natürlichsprachliche Anforderungen in formale Regeln übersetzt werden, die von Prüfungswerkzeugen durchgesetzt werden können. Abbildung 1.1 beschreibt die dafür notwendigen Arbeiten in einem generischen Vorgehensmodell. Hier werden die maßgeblichen Gesetze und Richtlinien schrittweise zerlegt und zu formal prüfbaren Regeln verfeinert. Bei jedem Schritt ergibt sich üblicherweise ein Teil von Anforderungen, der in der jeweils nächsten Verfeinerungsstufe nicht beschrieben werden kann und damit einer Automatisierung unzugänglich bleibt.

Im ersten Schritt werden Gesetze und Richtlinien für das jeweilige Unternehmen interpretiert und in ein schematisches Modell von Anforderungen übersetzt, für das konkrete Vorgaben zur Umsetzung existieren. *Best Practice*-Rahmenwerke wie COBIT (ISACA, 2007) und ITIL (OGC, 2007) sowie Methodologien wie (Raghupathi, 2007) und (Klempt u. a., 2009) unterstützen diesen Vorgang, er ist jedoch nur in Ansätzen automatisierbar. Beispielsweise schlagen Breaux u. a. (2006) ein Verfahren vor, welches durch Methoden der Mustererkennung Compliance-Anforderungen aus Gesetzestexten extrahiert. Die so gewonnenen Anforderungen werden anschließend in formale *Policies* übersetzt, die von Prüfungsmechanismen direkt umgesetzt werden können. Diese Mechanismen werden entweder vor bzw. während der Laufzeit angewendet oder nach der Ausführung. Hinsichtlich seiner Automatisierbarkeit ist ein Zertifizierungsverfahren umso besser, je höher es in der in Abbildung 1.1 dargestellten Hierarchie ansetzt.

## 1.2 Geschäftsprozesse als Gegenstand der Zertifizierung

Ein Prozess definiert einen standardisierten Arbeitsablauf in einer Unternehmung als Anordnung von Arbeitsschritten, die ausgeführt werden müssen um ein bestimmtes Geschäftsziel zu erreichen. Obwohl Prozesse kein sonderlich neuer Gegenstand sind, sondern sich beispielsweise schon im Hauptwerk Adam Smith' von 1776 finden[3], sind sie spätestens Anfang der 1990er Jahre durch Konzepte wie *Reengineering* und *Business Process Orientation* in den Fokus der Unternehmensorganisation gerückt (Porter, 1985; Davenport und Short, 1990; Davenport, 1992; Hammer und Champy, 1993; Hammer, 1996). Mittlerweile wird die zentrale strategische Stellung, die Geschäftsprozesse und deren Management für ein Unternehmen eingenommen haben, allgemein anerkannt (vom Brocke und Rosemann, 2010; Pfohl u. a., 2001).

Die Definition eines (Geschäfts-)Prozesses in der Literatur ist nicht einheitlich und hat sich mit dem Fortschreiten des computergestützten Geschäftsprozessmanagements weiterentwickelt (Lindsay u. a., 2003). Insbesondere wichtig ist die Abgrenzung zwischen dem sehr allgemein gefassten Begriff des Prozesses zu dem des *Workflows*, der einen Prozess in einer durch ein IT-System ausführbaren Darstellung bezeichnet. Die unten stehenden Definitionen folgen der Darstellung von zur Muehlen (2004), die wiederum auf der Arbeit von Becker und Schütte (1999) aufbaut.

**Definition 1.1 (Prozess).** *Ein Prozess ist eine diskrete Abfolge der Aktivitäten, die notwendig sind, um ein betriebswirtschaftlich relevantes Objekt zu bearbeiten. Dieses Objekt wird als Prozessobjekt bezeichnet und charakterisiert den Prozess. Abhängig von den Eigenschaften des Prozessobjekts wird zwischen materiellen und informationellen Objekten unterschieden.* ⊣

**Definition 1.2 (Geschäftsprozess).** *Die Menge der Geschäftsprozesse ist eine Untermenge der Prozesse. Ein Geschäftsprozess wird durch die übergeordneten Geschäftsziele des Unternehmens bestimmt. Geschäftsprozesse beinhalten Aktivitäten, die mit dritten Parteien (wie z.B. Kunden und Geschäftspartnern) interagieren.* ⊣

**Definition 1.3 (Workflow).** *Ein Workflow ist eine spezielle Darstellung eines Prozesses, die es einem Informationssystem erlaubt, die formalen Koordinationsmechanismen zwischen den einzelnen Aktivitäten, Anwendungen und Prozessteilnehmern zu kontrollieren. Ein solches Informationssystem wird als Geschäftsprozess- bzw. Workflow-Management-System bezeichnet.* ⊣

Der Gegenstand dieser Arbeit ist die Zertifizierung solcher Prozesse, die als Workflows abgebildet sind. Dies trifft mittlerweile auf einen Großteil der wichtigen Unternehmensprozesse zu (Wolf und Harmon, 2010; Sackmann und Strüker, 2005). Im Folgenden werden – sofern nicht anders angegeben – die Begriffe Prozess, Geschäftsprozess und Workflow daher weitgehend synonym verwendet. Die Bedeutung ist in allen Fällen zu verstehen wie in Definition 1.3 beschrieben.

[3] Hier beschreibt Smith beispielhaft einen Prozess zur Herstellung von Stecknadeln (nach Smith (2008)).

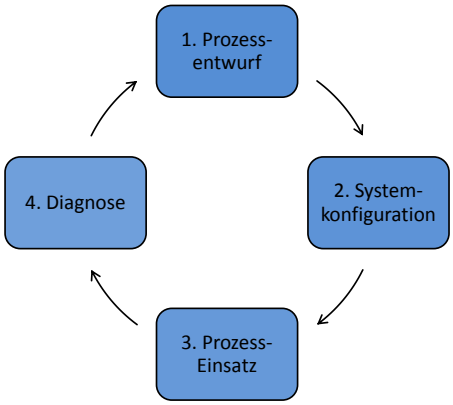



Abbildung 1.2.: Phasenmodell des Geschäftsprozess-Managements nach van der Aalst u. a. (2003b).

### 1.2.1 Geschäftsprozess-Management

Das Geschäftsprozess-Management (*Business Process Management* – BPM) schließt alle Tätigkeiten ein, die im weiteren Sinne mit der Erstellung, Ausführung und Verwaltung von Geschäftsprozessen befasst sind. Obwohl unter Geschäftsprozess-Management oft ausschließlich der Einsatz von entsprechenden Verwaltungsprogrammen – *Business Process Management Systems* (BPMS) – verstanden wird, ist BPM nur am Rande eine technologische Disziplin und in erster Linie mit dem Management und der Transformation organisatorischer Abläufe befasst. Den BPMS kommt dabei eine unterstützende Funktion zu.

Ähnlich wie der Begriff Geschäftsprozess wird auch der des Geschäftsprozess-Managements in der Literatur unscharf gebraucht. Eine verbreitete und anerkannte Definition stammt von van der Aalst u. a. (2003b):[4]

**Definition 1.4 (Geschäftsprozess-Management).** *Das Geschäftsprozess-Management ist eine Tätigkeit, die Geschäftsprozesse fördert, indem es Methoden, Techniken und Software verwendet, um betriebliche Prozesse zu entwerfen, einzusetzen, zu steuern und zu analysieren und dabei Menschen, Organisationen, Anwendungen, Dokumente und weitere Informationsquellen einbezieht.* ⊣

Diese sehr allgemein gefasste Definition wird konkreter durch die Betrachtung eines generischen Phasenmodells (*BPM life cycle*). Van der Aalst u. a. (2003b) beschreiben vier Phasen, die in Abbildung 1.2 schematisch dargestellt sind:

1. Während des **Prozess-Entwurfs** werden Prozesse, die beispielsweise in natürlicher Sprache beschrieben sind, als Workflows in formaler Sprache modelliert bzw. bestehende Workflows umgestaltet.

[4] Alternative Definitionen finden sich beispielsweise in (Ko, 2009; Hammer, 2010; Harmon, 2010).

2. In der **Systemkonfigurationsphase** werden die Workflows in einem BPMS implementiert und die grundlegende Infrastruktur darauf angepasst.
3. Der **Prozess-Einsatz** bezeichnet die Phase, in der die Prozesse produktiv eingesetzt werden.
4. In der **Diagnosephase** werden die Prozesse analysiert, um Probleme und Verbesserungsmöglichkeiten zu identifizieren, die anschließend wieder in die Entwurfsphase einfließen.

Die Bezeichnung BPMS hat sich im letzten Jahrzehnt als Oberbegriff für Softwareanwendungen herausgebildet, die eine oder mehrere Phasen des Geschäftsprozess-Managements unterstützen (Harmon, 2010). BPMS sind in der Regel integrierte Systeme, die mehrere Funktionen vereinen, welche davor – falls überhaupt Softwarelösungen existierten – von Einzelanwendungen umgesetzt wurden. Diese Funktionen fallen zum Beispiel in die Kategorien Workflow-Management, *Business Intelligence*, Regelbasierte Systeme (*Business Rules Engines*) und Unternehmensanwendungsintegration (*Enterprise Application Integration*) (Khan, 2004).

Der Definition 1.4 entsprechend schließt das Geschäftsprozess-Management auch Maßnahmen zur Gewährleistung von Sicherheitsanforderungen in Geschäftsprozessen ein. Funktionen zur Automatisierung dieser Maßnahmen sind ein möglicher Bestandteil eines BPMS und können in allen Phasen des Geschäftsprozess-Managements ansetzen, also entweder vor der Ausführung eines Prozesses (Phasen 1 und 2), während seiner Ausführung (Phase 3) und im Anschluss an die Ausführung (Phase 4). Die a-priori-Zertifizierung, wie sie Gegenstand dieser Arbeit ist, ist ein Bestandteil der ersten beiden Phasen des Modells aus Abbildung 1.2.

### 1.2.2 IT-Unterstützung des Geschäftsprozess-Managements

Die Entwicklungen des BPM und der technischen Systeme zu seiner Unterstützung haben sich stets wechselseitig bedingt und gefördert. Zum Einen sind aus den Erfordernissen des Geschäftsprozess-Managements neue Funktionen in den unterstützenden Softwareanwendungen erwachsen. Dies betrifft vor allem die (teilweise) Automatisierung von zuvor manuell ausgeführten Tätigkeiten, wie z.B. die Verwaltung von Workflow-Beschreibungen (van der Aalst und van Hee, 2004). Zum Anderen haben sich durch das Aufgreifen von Entwicklungen der Computertechnik neue Möglichkeiten zur Organisation und Ausführung von Prozessen ergeben, die ein effizienteres Erreichen der angestrebten Geschäftsziele erlauben. So kann beispielsweise durch dienstbasierte Architekturen eine flexiblere Anpassung von Prozessen an organisatorische Strukturen erreicht werden (Cummins, 2010).

Höhn (2010) identifiziert in der Entwicklung des Geschäftsprozess-Managements die folgenden drei Phasen, die durch technische Entwicklungen auf Seiten der jeweils zugrunde liegenden BPMS gekennzeichnet sind.

- **Angepasste Standardsoftware.** Infolge der hohen Kosten für die Entwicklung kundenspezifischer Anwendungen hat sich seit den 1970er Jahren Standardsoftware als Baustein für Unternehmens-IT etabliert. Die Standardsoftware wird in vielen Unternehmen für ähnliche Prozesse eingesetzt und entsprechend den Rahmenbedingungen eines Unternehmens bzw. einer Branche angepasst (*Customization*).[5] Voraussetzung für den Einsatz solcher Software in einem Unternehmen ist somit die Standardisierung der Geschäftsprozesse (Scheer, 1978, 1980). Die Änderung der Prozesse bedingt den Austausch oder die Umgestaltung großer Teile der Unternehmenssoftware, was sehr kostenintensiv ist und insbesondere von kleineren Unternehmen nicht geleistet werden kann (Dibrell und Miller, 2002; Robey, 1981).
- **Workflow-Systeme.** Die Arbeiten zu *Reengineering* und *Business Process Orientation* haben gezeigt, dass Geschäftsprozesse möglichst entsprechend der Struktur eines Unternehmens gestaltet werden müssen, um effizient wirtschaften zu können. Dies erschwert den Einsatz von einheitlicher Software, die Standardprozesse abbildet, da die Unternehmens-IT an die *individuellen* Prozesse des Unternehmens angepasst werden muss. Workflow-Systeme erlauben die freie Definition der Struktur eines Prozesses als Abfolge von Einzelaktivitäten (van der Aalst und van Hee, 2004). Die Einzelaktivitäten wiederum können ggf. durch Standardanwendungen ausgeführt werden. Mit Workflow-Systemen wird das Geschäftsprozess-Management flexibler als durch den Einsatz reiner Standardsoftware. Allerdings ist ein Workflow noch immer ein weithin starres Artefakt, da die Schnittstellen der Einzelaktivitäten nicht standardisiert sind und bei Prozess-Änderungen manuell angepasst werden müssen.
- **Dienstbasierte Architekturen.** Verkürzte Produktzyklen und steigender Effizienzdruck erfordern eine fortlaufende Anpassung der Prozesse eines Unternehmens an veränderliche betriebswirtschaftliche Rahmenbedingungen (Cummins, 2010). Prozessmodifikationen aufgrund von gewandelten Anforderungen und Störungen müssen nahezu in Echtzeit geschehen (Reichert und Dadam, 1998). BPMS müssen daher die Voraussetzungen schaffen, Prozesse schnell umkonfigurieren und Prozessbestandteile austauschen zu können, ohne dass eine Neuimplementierung erforderlich wird. Dienst-Architekturen (*Service-oriented Architectures – SOA*) stellen dazu einheitliche Schnittstellen für Softwareanwendungen – Dienste (*Services*) – bereit, aus denen Prozesse nach Bedarf zusammengestellt werden können (Leymann u. a., 2002; Woods und Mattern, 2006). Die Modularität dieses Prinzips ermöglicht den schnellen Austausch von Diensten und ihre Wiederverwendung in mehreren Prozessen. Darüber hinaus können Arbeitsschritte einfach ausgelagert werden, indem Dienste eines spezialisierten externen Anbieters über Datennetze in die eigenen Prozesse eingebunden werden.

Dienstbasierte Architekturen bilden die technische Grundlage für die Verknüpfung von Geschäftsprozessen mit externen Dienstanbietern (etwa *Cloud-Services*), wie sie im Bei-

[5] Eine Anwendung aus dieser Kategorie ist beispielsweise das ERP-System (*Enterprise Resource Planning*) R/3 von SAP.

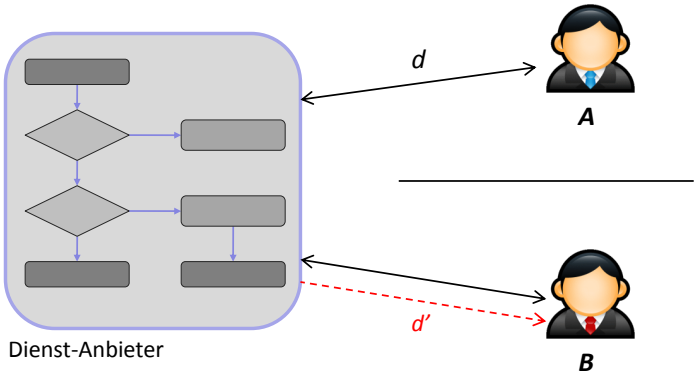



Abbildung 1.3.: Isolation mehrerer Client-Prozesse.

spiel am Anfang dieses Kapitels illustriert wurden (Jin u. a., 2010; Howard, 2010). *Cloud-Computing* im Allgemeinen beschreibt ein Paradigma der über leistungsfähige Rechenzentren verteilten Datenverarbeitung, die externen Kunden in Form verschiedener Dienste als abrufbare Leistungen bereitgestellt wird (Foster u. a., 2008). Cloud-Computing wird derzeit von vielen Beobachtern als eine der IT-Entwicklungen mit der für Unternehmen größten strategischen Bedeutung angesehen, deren Vorteile geringere IT-Investitionen, eine gute Skalierbarkeit und ein niedrigerer Wartungsaufwand sind (Gartner, 2010).

Einhergehend mit den Vorteilen Cloud-gestützter Geschäftsprozesse weist praktisch jeder wissenschaftliche Beitrag zu Cloud-Computing auf die derzeit noch weitgehend ungelöste Sicherheitsproblematik hin, die aus der externen Verarbeitung sensibler Informationen erwächst und ein Hemmnis für deren großflächigen Einsatz darstellt (vgl. etwa Chow u. a. (2009); Ristenpart u. a. (2009); Tchifilionova (2011)). Damit kommt einer zuverlässigen Sicherheits-Zertifizierung entsprechender Geschäftsprozesse eine zentrale Bedeutung für deren praktischen Einsatz zu.

## 1.3 Sicherheit durch Isolationsgarantien

In Geschäftsprozess-Konstellation mit mehreren beteiligten Parteien muss als grundlegende Sicherheitsanforderung gewährleistet werden können, dass jede Partei ausschließlich Zugriff auf Informationen erhält, für die sie eine Berechtigung besitzt. Für das Beispiel eines externen Dienst-Anbieters zeigt Abbildung 1.3 schematisch eine Prozess-Konstellation mit drei beteiligten Parteien. Hier nehmen die beiden Parteien A und B die Leistungen einer dritten Partei, des externen Dienst-Anbieters, in Anspruch und tauschen dabei Informationen aus. Es muss sichergestellt sein, dass keine sensible Information von Partei A zu Partei B gelangen kann (wie durch den gestrichelten Pfeil symbolisiert) – und umgekehrt. Dies bedeutet, dass gekennzeichnete Informationen (in diesem Fall Daten von A) während der Prozessausführung von unberechtigten Parteien (in diesem Fall B) *isoliert* sein müssen.

Die Isolation von Anwendungen auf Dienst-Plattformen ist eine fundamentale Sicherheitsanforderung, die an deren Betreiber gestellt wird. Sie findet sich beispielsweise in den entsprechenden Empfehlungen des deutschen Bundesamtes für Sicherheit in der Informationstechnik (BSI, 2010), der Europäischen Agentur für Netz- und Informationssicherheit (ENISA, 2010) und des US-amerikanischen *National Institute of Standards and Technology* (Jansen und Grance, 2011).

Auf Anwendungsebene wird Isolierung üblicherweise durch den Einsatz virtueller Maschinen oder auf der Ebene der Betriebssysteme durch getrennte Adressräume erreicht, ohne dass die Anwendung selbst betrachtet wird. Diese Sichtweise wird vielen Geschäftsprozess-Szenarien, wie etwa dem eingangs betrachteten, jedoch nicht gerecht, da die Granularität der Isolierung zu grob ist: Hier werden Geschäftsprozesse verschiedener Herkunft auf gemeinsamen Plattformen miteinander verknüpft. Dabei muss gewährleistet sein, dass Prozesse legitimerweise benötigte Daten von ihren Partnern erhalten können, ihnen darüber hinaus aber der Zugriff auf sensible Informationen verwehrt bleibt.

Auch für Dienste, die als sogenannte Multimandanten-fähige Architekturen (*Multi-Tenant Architectures*) implementiert sind, ist die reine Anwendungsisolation zu grob: In einer solchen Architektur stellt *dieselbe* Anwendung gleichzeitig mehreren Klienten einen Dienst bereit. Hier muss gewährleistet sein, dass zwischen den Klienten keine ungewollte Informationsübertragung stattfindet, wie es in Abbildung 1.3 schematisch dargestellt ist (Banks u. a., 2009; Bacon u. a., 2010). Für die Sicherheits-Zertifizierung von verteilt ausgeführten Geschäftsprozessen sind daher feingranulare Prüfungsmechanismen erforderlich, die innerhalb einer Prozess-Konstellation die Verbreitung von Informationen analysieren und hinsichtlich ihrer Isolation bewerten können.

## 1.4 Anforderungen an Zertifzierungsverfahren

Zusammenfassend ergeben sich aus den Ausführungen dieses Kapitels die folgenden Anforderungen an ein Verfahren zur Sicherheitszertifzierung von Geschäftsprozessen:

1. **Workflow-Modelle:** Für die a-priori-Zertifizierung ist die Existenz aussagekräftiger Workflow-Modelle erforderlich, die die Verbreitung von Informationen in einer Prozess-Konstellation in eindeutiger Weise darstellen können.

2. **Anforderungsbeschreibung:** Es muss auf der Ebene einzelner Datenobjekte ausgedrückt werden können, welche an einem Prozess beteiligte Partei Zugriff auf welche Informationen hat. Ein entsprechender Beschreibungsformalismus muss kompatibel zu den betrachteten Workflow-Modellen sein.

3. **Zertifizierungsgarantien:** Das Prüfverfahren muss zuverlässig entscheiden können, ob die Isolation von Informationen bzgl. einer gegebenen Anforderung in einem Workflow-Modell gewährleistet ist. Zuverlässig bedeutet insbesondere, dass alle möglichen Verletzungen (innerhalb der betrachteten Klasse) erkannt werden.

4. **Automatisierung:** Für die praktische Einsetzbarkeit sollten alle im Zusammenhang mit der Zertifizierung auszuführenden Arbeitsschritte automatisierbar sein. Dies betrifft neben dem eigentlichen Prüfvorgang insbesondere die Generierung von geeigneten Workflow-Modellen und eine dazu kompatible Formalisierung der Isolationsanforderungen.

## 1.5 Stärkere Garantien durch Informationsflussmechanismen

Diese Arbeit gründet sich auf die These, dass Mechanismen der Informationsflusskontrolle effektiv als Kern eines Verfahrens für die Sicherheitszertifizierung von Geschäftsprozessen eingesetzt und damit stärkere Isolationsgarantien erreicht werden können, als dies mit bestehenden Verfahren möglich ist.

Mechanismen der Informationsflusskontrolle arbeiten auf der Grundlage einer abstrakten Spezifikation von Sicherheitsanforderungen, die als Einschränkungen an den erlaubten Informationsfluss in einem System beschrieben werden (McLean, 1990; Ryan, 2001). Dabei werden den Trägern von Informationen (beispielsweise Dateien, Dokumenten, Nachrichten) in Abhängigkeit ihres Schutzanspruchs *Sicherheitsstufen* zugeordnet, die hierarchisch angeordnet sind (Denning, 1976). Subjekten des Systems werden *Freigaben* erteilt, die festlegen, Informationen welcher Stufe ein Subjekt erhalten darf. Eine Informationsfluss-Policy definiert die Sicherheit von Informationen damit als eine *Ende-zu-Ende*-Eigenschaft eines Systems (Saltzer u. a., 1984), bei der die Enden die Träger oder Konsumenten von Informationen sind.

Die Aufgabe von Informationsflussmechanismen ist zu prüfen, ob eine Policy erfüllt wird, d.h. ob es eine unerlaubte Informationsübertragung geben kann. Dazu wird ein Systemmodell – wie in diesem Kontext das Modell einer Geschäftsprozess-Konstellation – darauf untersucht, wie sich Informationen verbreiten können und ob diese entsprechend den gestellten Einschränkungen *isoliert* voneinander sind (Goguen und Meseguer, 1982).

Verglichen mit traditionellen Zugriffskontrollen, die das am weitesten verbreitete Instrument zur Beschreibung und Durchsetzung von Sicherheitsanforderungen sind (Sandhu und Samarati, 1994; Schneider, 2000), können Informationsfluss-Mechanismen vollständigere und stärkere Sicherheitsgarantien geben. Dieser Vorteil wird deutlich bei der Gegenüberstellung einer Informationsfluss-Policy mit einer entsprechenden Zugriffskontrollregel. Die folgende (natürlichsprachliche) Informationsfluss-Policy beschreibt eine Vertraulichkeitsanforderung hinsichtlich des Inhalts einer Email:[6]

*„Die in dieser Email enthaltenen Informationen dürfen nur mich und den Empfänger erreichen.“*

Eine entsprechende Formulierung durch Zugriffsregeln könnte wie folgt aussehen:

[6] Das Beispiel ist an eine Darstellung von Zdancewic (2002) angelehnt.

*„Nur Systemprozesse, die durch mich oder den Empfänger autorisiert werden, dürfen die Datei öffnen, die diese Email enthält.“*

Die Informationsfluss-Policy ist stärker, da sie sich unmittelbar auf Informationen bezieht, anstatt lediglich den Zugriff auf Datencontainer zu kontrollieren. So verhindern die Zugriffsregeln nicht, dass der Empfänger den Inhalt der Email (unter Umständen verschlüsselt oder in einer anderen Kodierung) weiterreicht. Außerdem umfasst die Informationsfluss-Policy im Gegensatz zur Zugriffsregel auch Informationsübertragung abseits von direkten Zugriffen auf Datenobjekte. Dies betrifft insbesondere sogenannte *verdeckte Kanäle*, über die Information an unberechtigte Dritte fließen kann. Diese bezeichnen Wege der Informationsübertragung über Kanäle, die nicht für diese Nutzung bestimmt sind, aber dafür zweckentfremdet werden können (Lampson, 1973). Sie werden beispielsweise durch die Beobachtung von Speicherzugriffen, des Kontrollflusses oder des zeitlichen Verhaltens eines Systems ermöglicht (McHugh, 1995).

Der Grund für die Beschränkung von Zugriffskontrollen liegt darin, dass sie ein Schutzziel (wie hier Vertaulichkeit) nicht direkt formulieren, sondern mithilfe eines spezifischen Sicherheitsmechanismus (einem Zugriffskontrollmonitor) *umschreiben*. Eine solche *intensionale* Art der Spezifikation ist oft unvollständig, da jede mögliche Verletzung im Voraus berücksichtigt und entsprechende Regeln aufgestellt werden müssen. So müsste, um im obigen Beispiel die unberechtigte Weiterleitung der Email zu verhindern, für jeden Akteur eine Zugriffsregel bzgl. der Email und davon abgeleiteten Dateien formuliert werden. Verdeckte Informationsübertragung wird nicht erkannt, da sie nicht über Datenzugriffe geschieht und die Mächtigkeit von Zugriffkontrollmonitoren übersteigt (Clarkson und Schneider, 2008). *Extensionale* Spezifikationen – wie Informationsfluss-Policies – erscheinen für die Beschreibung von Sicherheitseigenschaften daher besser geeignet als intensionale (Gollmann, 2001; Roscoe, 1996).

Auf der Grundlage dieser Überlegungen erscheinen Informationsflussmechanismen als ein geeignetes Instrument für die Definition und Prüfung von starken Isolationsanforderungen in Geschäftsprozessen. Die umfassenderen Garantien, die durch eine Zertifizierung auf dieser Basis erreicht werden, können insbesondere die Bereitschaft zur Auslagerung sicherheitskritischer Geschäftsprozesse erhöhen. So wird von einschlägigen Richtlinien, wie etwa den *Trusted Computer System Evaluation Criteria* (TCSEC, 1985) bzw. *Common Criteria* (Common Criteria, 2009) die Betrachtung verdeckter Informationsübertragung in sicherheitskritischen Anwendungen explizit empfohlen. Möglichkeiten zur Ausnutzung von verdeckten Kanälen in Dienst-Infrastrukturen wurden bereits aufgezeigt (Ristenpart u. a., 2009).

## 1.6 Beitrag und Aufbau der Arbeit

Die Arbeit stellt mit InDico ein automatisiertes Vorgehensmodell für die Zertifizierung von Isolationsanforderungen in Geschäftsprozessen vor. In seinem Kern nutzt InDico Informationfluss-Mechanismen zur Analyse von Geschäftsprozess-Modellen, die in einer Petri-Netz-basierten Sprache beschrieben sind. Wesentliche Arbeitsschritte von der Modellerzeugung bis zur Ableitung von Anforderungen können automatisiert werden. Im Einzelnen leistet die Arbeit damit die folgenden Beiträge:

- Sie stellt mit IFnet eine ausdrucksstarke, formal fundierte Sprache zur Beschreibung von Geschäftsprozess-Konstellationen bereit. IFnet basiert auf Petri-Netzen und erweitert diese um Konstrukte zur Prozess-Modellierung, die für die Analyse des Informationsflusses benötigt werden.
- Sie entwickelt ein in IFnet integriertes Sicherheitsmodell, das die feingranulare Klassifikation von Informationsträgern erlaubt. In IFnet können Träger von Informationen sowohl Datenobjekte als auch die Aktivitäten des Prozesses sein. Den am Prozess beteiligten Parteien werden entsprechende Freigaben zugeordnet. Es wird eine spezielle Freigabeklasse für „neutrale“ Parteien (wie z.B. Intermediäre) eingeführt.
- Sie definiert Kriterien für die unerlaubte Informationsübertragung in einem IFnet-Modell. Diese berücksichtigen sowohl den Datenfluss als auch die verdeckte Informationsübertragung durch Abhängigkeiten zwischen Workflow-Aktivitäten. Prüfungsalgorithmen zur Entscheidung der Kriterien werden angegeben.
- Sie gibt eine Abbildung der Kernkonstrukte der *Business Process Modeling Language* (BPMN) auf IFnet an, mit denen die Erzeugung von Modellen für die Zertifizierung (teil-)automatisiert werden kann. Darüber hinaus schlägt sie Strategien für die automatisierte Kodierung von Isolationsanforderungen in einem IFnet-Modell vor.

Die Arbeit gliedert sich wie folgt: Kapitel 2 diskutiert bestehende Formalismen zur Modellierung von Geschäftsprozessen und Verfahren zu deren Sicherheitsprüfung im Hinblick auf die in diesem Kapitel aufgestellten Anforderungen. Kapitel 3 bildet den Kern der Arbeit und beschreibt das InDico-Vorgehensmodell und seine Komponenten. Die Algorithmen zur Zertifzierung von IFnet-Modellen und die Implementierung von InDico werden separat in Kapitel 4 beschrieben. Kapitel 5 demonstriert die Anwendung von InDico in einem Anwendungsszenario und bewertet den Ansatz hinsichtlich der Erfüllung der in diesem Kapitel aufgestellten Anforderungen. Die Arbeit schließt mit einer Zusammenfassung und einem Ausblick in Kapitel 6.

Kapitel 2

# Bestehende Zertifizierungsverfahren

Dieses Kapitel diskutiert bestehende Verfahren zur Prüfung von Sicherheitsanforderungen in Geschäftsprozessen und bewertet sie hinsichtlich ihrer Eignung zur Zertifizierung von Isolationseigenschaften. Dazu wird zunächst eine Klassifikation eingeführt, die die bestehenden Verfahren hinsichtlich der Prüfungsebene und des Prüfungszeitpunktes einordnet.

Die darauf folgende Diskussion ist zweigeteilt: Zunächst werden Formalismen zur Modellierung von Geschäftsprozessen betrachtet, anschließend werden die Methoden zur Prüfung von Sicherheitseigenschaften untersucht, die auf der Grundlage dieser Formalismen arbeiten. Den Abschluss jedes Teils bildet jeweils eine Bewertung der vorgestellten Formalismen und Verfahren im Hinblick auf die in Kapitel 1 aufgestellten Anforderungen.

## 2.1 Klassifikation der Verfahren

Zwei Dimensionen werden betrachtet, um automatisierte Verfahren zur Prüfung von Sicherheitsanforderungen in Geschäftsprozessen zu klassifizieren. Diese sind zum einen die *Abstraktionsebene*, auf der ein Ansatz arbeitet, und zum anderen der *Zeitpunkt der Prüfung* im Phasenmodell des Geschäftsprozessmanagements (siehe Abschnitt 1.2.1).

### 2.1.1 Prüfungsebene

Abbildung 2.1 zeigt eine Einteilung der Abstraktionsebenen bei der Ausführung IT-gestützter Geschäftsprozesse, wobei der Abstraktionsgrad der Ebenen in der Abbildung von unten nach oben zunimmt. Die Ebenen sind in drei Gruppen eingeteilt, wobei die unteren beiden Gruppen aus jeweils zwei Ebenen bestehen und die obere Gruppe eine Ebene enthält.

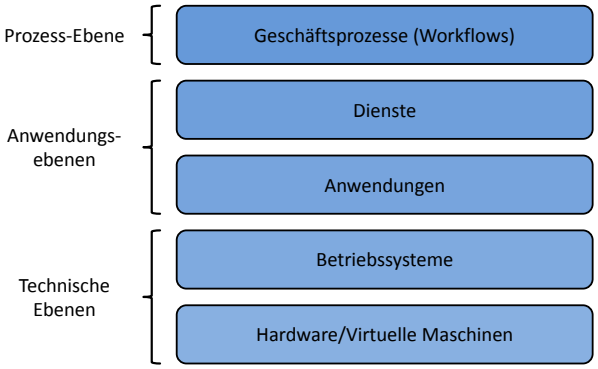



Abbildung 2.1.: Abstraktionsebenen der IT-gestützten Geschäftsprozessausführung.

In der unteren Gruppe befinden sich zum einen die Hardware und ggf. virtuelle Maschinen, die Hardware emulieren, sowie – darauf aufbauend – die Betriebssysteme. Diese beiden Ebenen stellen die generische Funktionalität zur Ausführung der spezifischen Anwendungslogik bereit, die in der mittleren Gruppe eingeordnet sind.

Hier befindet sich die Anwendungsebene, die Funktionalität zur Ausführung einzelner Arbeitsschritte in Anwendungen bereitstellt, wie beispielsweise Buchhaltungssysteme, aber auch Basistechnologien zur Speicherung von Daten (Datenbanken) und zur Kommunikation. Auf die Anwendungen baut die Dienst-Schicht auf, die die Anwendungsfunktionalität über standardisierte Schnittstellen als Dienste bereitstellt. Außerdem ist sie für die Auswahl und Zusammensetzung von Diensten zu komplexen Anwendungen – wie in diesem Fall Geschäftsprozessen – zuständig.

Auf der Prozess-Ebene sind BPMS angesiedelt, die das Zusammenspiel der – möglicherweise verteilt auf verschiedenen Plattformen ausgeführten – Dienste koordinieren, um den Prozess umzusetzen. Der zentrale Gegenstand auf dieser Ebene ist ein formal beschriebenes Prozessmodell (d.h. ein *Workflow* nach Definition 1.3).

Sicherheitsprüfungen können an jeder dieser Ebenen ansetzen und für alle Ebenen gibt es entsprechende Verfahren. In diesem Kapitel werden im Folgenden ausschließlich diejenigen Ansätze betrachtet, die auf der Prozess-Ebene ansetzen, d.h. solche, die ein explizit dargestelltes Prozessmodell analysieren. Der Bezug zu den anderen Ebenen wird im Einzelfall hergestellt, sofern es sinnvoll erscheint.

### 2.1.2 Prüfungszeitpunkt

Orthogonal zur Abstraktionsebene werden die Ansätze hinsichtlich des Prüfungszeitpunktes aufgeteilt in solche, die die Verletzung einer Sicherheits-Policy *von vornherein* verhindern können und jene, die eine Verletzung *im Nachhinein* erkennen.

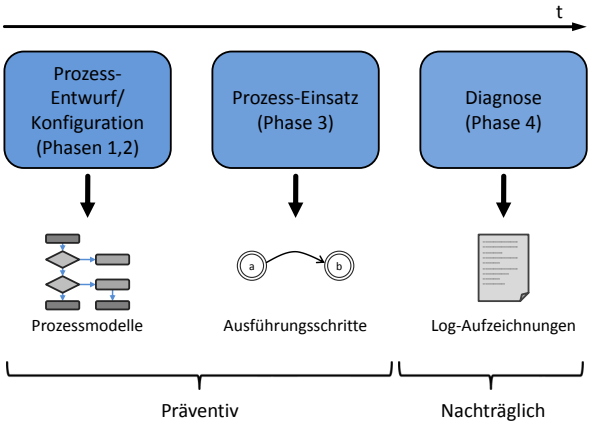



Abbildung 2.2.: Einordnung von Prüfungsverfahren bzgl. des Prüfungszeitpunktes.

Bezogen auf das Phasenmodell des Geschäftsprozessmanagements (siehe Abbildung 1.2) sind Erstere Teil der Entwurfs-, Konfigurations oder Einsatzphase. Diese *präventiven* Verfahren können wiederum unterteilt werden in solche, die ein Prozessmodell vor der Laufzeit betrachten (Entwurfs- bzw. Konfigurationsphase) und solche, die als Laufzeitmonitor während der Ausführung jeden Arbeitsschritte prüfen und ggf. einschreiten, wenn eine Verletzung einzutreten droht (Einsatzphase). Verfahren, die Verletzungen nachträglich erkennen, kommen in der Diagnosephase zum Einsatz und analysieren Log-Daten, die während der Ausführung aufgezeichnet worden sind. Abbildung 2.2 stellt diese Einteilung schematisch dar.

Der Gegenstand dieser Arbeit ist die *a priori*-Zertifizierung von Geschäftsprozessen. Daher werden in diesem Kapitel ausschließlich präventive Ansätze diskutiert. Reine Laufzeitverfahren werden nicht betrachtet, da sie nicht mächtig genug sind, um verdeckte Informationsübertragung zu erkennen und deshalb nicht als Grundlage für die Erzeugung hinreichend aussagekräftiger Zertifikate dienen können (Schneider, 2000; Clarkson und Schneider, 2008).

## 2.2 Formalismen zur Geschäftsprozess-Modellierung

In diesem Abschnitt werden drei Familien von Formalismen diskutiert, die zur Beschreibung von Geschäftsprozessen verwendet werden: Industrielle Modellierungssprachen, Prozesskalküle und Petri-Netze.

### 2.2.1 Industrielle Modellierungssprachen

Den De-facto-Standard zur Modellierung von Dienst-basierten Geschäftsprozessen in industriellen BPMS bilden zwei Sprachen, die oft komplementär verwendet werden (Pfitzner u. a., 2009; Ouyang u. a., 2006): Die XML-basierte Ausführungssprache *Business Process Execution Language* (BPEL) und die grafische *Business Process Modeling Notation* (BPMN).

#### Business Process Execution Language

Die *Business Process Execution Language*[1] (BPEL) ist ein Ergebnis der Aktivitäten zur Standardisierung von *Web Service*-Technologien, die seit ca. 2002 von mehreren Unternehmen und Institutionen vorangetrieben werden (OASIS, 2010). *Web Services* (WS) bezeichnen Anwendungen mit einheitlichen Schnittstellen, auf die über das Internet entfernt zugegriffen werden kann. Die WS-Technologien sind der am weitesten verbreitete Standard, um Dienst-basierte Architekturen zu realisieren. Bezogen auf das Ebenenmodell in Abbildung 2.1 stellen sie Technologien zur Implementierung der oberen beiden Schichten bereit. BPEL ist die WS-Komponente zur Beschreibung von Geschäftsprozessen auf der obersten Ebene in diesem Modell.

BPEL ist in wesentlichen Teilen eine Verknüpfung von zwei Vorgängersprachen (van Breugel und Koshkina, 2006): WSFL von IBM (Leymann, 2001) und XLANG von Microsoft (Thatte, 2001). BPEL-Prozesse werden in einem XML-Dokument beschrieben, das zwei Arten von Aktivitäten auszeichnet[2]: grundlegende (*elementary activities*) und strukturierte Aktivitäten (*structured activities*). Die wichtigsten elementaren Aktivitäten sind `empty` (keine Tätigkeit), `wait` (warten), `assign` (Zuweisung eines Wertes), `receive` (auf eine Nachricht warten), `invoke` (einen Dienst aufrufen), `reply` (auf eine Nachricht mit einer weiteren Nachricht antworten), `throw` (einen Fehler signalisieren) und `terminate` (die Prozess-Instanz beenden).

Strukturierte Aktivitäten definieren den Kontrollfluss über elementaren Aktivitäten und können ineinander verschachtelt sein. Die wichtigsten strukturierten Aktivitäten sind

[1] Die vollständige Bezeichnung für die frühen Versionen (bis 1.1) lautet *Business Process Execution Language for Web Services* (BPEL4WS). Mit Einführung der Version 2.0 wurde die Bezeichnung zu *Web Services Business Process Execution Language* (WS-BPEL) geändert. Hier wird die übliche Kurzform BPEL verwendet, die – soweit nicht anders angegeben – alle Versionen einschließt.

[2] Die hier beschriebene Syntax entspricht BPEL in der verbreiteten Version 1.1 (Andrews u. a., 2003).

```
<sequence name="main">
  <empty name="Enter_Patient_ID"/>
  <switch name="Switch_1">
    <case>
      <sequence>
        <empty name="Open_Patient_Record"/>
        <empty name="Update_Patient_Record"/>
        <empty name="Close_Patient_Record"/>
      </sequence>
    </case>
    <otherwise>
      <empty name="Issue_Warning"/>
    </otherwise>
  </switch>
</sequence>
```

Programmtext 2.1: BPEL-Codefragment eines einfachen Prozesses.

`sequence` (sequentielle Ausführung von Aktivitäten), `flow` (parallele Ausführung), `while` (Schleife mit Anfangsbedingung), `switch` (Auswahl eines Pfades in Abhängigkeit von Daten) und `pick` (Auswahl eines Pfades in Abhängigkeit von zeitlichen Bedingungen oder Nachrichten).

Programmtext 2.1 zeigt einen Ausschnitt aus einem Prozess in BPEL-Notation, der aus fünf grundlegenden `empty`-Aktivitäten besteht. Diese sind durch strukturierte Aktivitäten derart angeordnet, dass die (grundlegende) Aktivität `Enter_Patient_ID` zu Anfang ausgeführt wird und der Prozess anschließend in zwei alternative Pfade verzweigt. Der erste Pfad enthält drei Aktivitäten, die sequentiell ausgeführt werden, der zweite Pfad enthält eine Aktivität.

BPEL erfährt in der Industrie eine breite Unterstützung und wird in zahlreichen Produkten eingesetzt (Andrews u. a., 2003), eignet sich aufgrund der fehlenden formalen Semantik aber nicht für entsprechende (formale) Analysen (Hinz u. a., 2005). Die Semantik von BPEL wird im natürlichsprachlich abgefassten Standard definiert, der in Teilen inkonsistent, mehrdeutig und unvollständig ist (van Breugel und Koshkina, 2006). Für die Umsetzung von BPEL in industriellen Softwarepaketen wird der Standard von den Herstellern interpretiert und mit spezifischen Ausführungssemantiken versehen, die voneinander abweichen können. Für formale Analysen wird BPEL üblicherweise in Darstellungen übersetzt, die Prozesskalküle (Lucchia und Mazzara, 2007; Ferrara, 2004) oder Petri-Netze (Lohmann u. a., 2009) verwenden.

### Business Process Modeling Notation

Die *Business Process Modeling Notation* (BPMN) ist eine Modellierungssprache für Geschäftsprozesse, die von der *Object Management Group* (OMG) verwaltet wird (OMG,

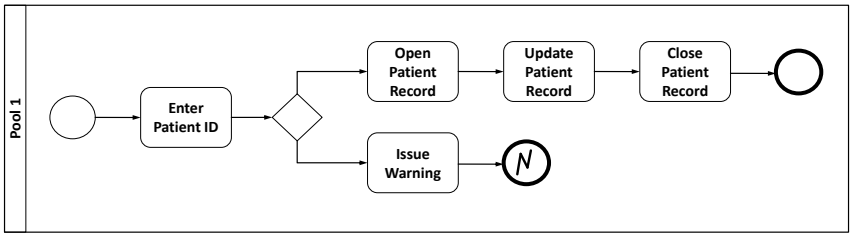



Abbildung 2.3.: Der Prozess aus Programmtext 2.1 in BPMN-Darstellung.

2009). Mit ihrer grafischen Notation, die in der Regel anschaulicher als eine rein textuelle Darstellung ist, spricht sie ausdrücklich alle an einem Geschäftsprozess beteiligten Parteien an, sowohl im betriebswirtschaftlichen als auch im technischen Bereich.

Die Darstellung von BPMN-Prozessen ist an Flowchart-Diagramme angelehnt. Ein Prozess wird aus grafischen Elementen zusammengesetzt, die zu einer der folgenden vier Kategorien gehören:

- *Flow Objects* beschreiben die Funktionseinheiten eines Prozesses und umfassen *Activities* (Aktivitäten), *Gateways* (exklusive und parallele Verzweigungen) und *Events* (Ereignisse).
- *Connecting Objects* verbinden die *Flow Objects* zu einem Graphen. Sie sind unterteilt in *Sequence Flow* (Kontrollfluss), *Message Flow* (Nachrichtenaustausch) und *Association* (Verbindung von *Flow Objects* mit Annotationen und *Artifacts*).
- *Swimlanes* unterteilen ein BPMN-Diagramm in mehrere Subprozesse, die von jeweils anderen Parteien verantwortet werden. Die Eingrenzung eines solches Subprozesses ist ein *Pool*, der wiederum in mehrere *Lanes* unterteilt sein kann.
- *Artifacts* sind Attribute mit denen ein Prozess ausgezeichnet werden kann, die nicht unmittelbar von dem Kontroll- oder Nachrichtenfluss abhängen. *Artifacts* sind entweder vom Typ *Data Object* (Datenobjekte), *Group* (Gruppierung von Aktivitäten ohne Einfluss auf den Ablauf) oder *Annotation* (Zusatzinformationen als Freitext).

Abbildung 2.3 zeigt den in Programmtext 2.1 beschriebenen BPEL-Prozess in BPMN-Darstellung. Der Prozess ist in einem *Pool* („Pool 1“) dargestellt. *Events* werden durch Kreise repräsentiert, *Aktivitäten* durch Rechtecke und *Gateways* durch gekippte Quadrate. Sämtliche in der Darstellung vorkommenden *Connecting Objects* sind vom Typ *Sequence Flow* und durch Pfeile bezeichnet.

Genau wie BPEL ist BPMN in erster Linie eine industriell gebrauchte Sprache, deren Definition keine formale Semantik zugrunde liegt und in Teilen inkonsistent und mehrdeutig ist (Dijkman u. a., 2008). Dies führt dazu, dass auch die Übertragung von BPMN in BPEL und umgekehrt, deren Ermöglichung ein Ziel bei der Entwicklung von BPMN war, nicht eindeutig ist (Weidlich u. a., 2008; Recker und Mendling, 2006). Zur formal

$$
\begin{aligned}
x(pid).\tau_{\mathsf{Enter\_PID}}.\overline{y}\,\langle pid\rangle\,.0 \\
y(pid).\,[pid = ok]\,.\overline{z}\,\langle pid\rangle\,.0 \\
y(pid).\,[pid = nok]\,.\overline{r}\,\langle pid\rangle\,.0 \\
z(pid).\tau_{\mathsf{Open\_PR}}.\tau_{\mathsf{Update\_PR}}.\tau_{\mathsf{Close\_PR}}.0 \\
r(pid).\tau_{\mathsf{Issue\_Warning}}.0
\end{aligned}
$$

Abbildung 2.4.: Der Prozess aus Programmtext 2.1 in der Notation des $\mu$BPMN-Prozesskalküls.

fundierten Analyse von BPMN-Prozessen werden diese üblicherweise in andere Darstellungen übersetzt (Dijkman u. a., 2008; Wong und Gibbons, 2008).

### 2.2.2 Prozesskalküle

Neben Petri-Netzen sind Prozesskalküle die am weitesten genutzten formalen Sprachen zur Modellierung von Geschäftsprozessen (van der Aalst, 2005). Prozesskalküle sind Meta-Modelle für die Beschreibung von nebenläufigen Systemen mit einem relativ geringen Sprachumfang, die von den meisten Details industriell eingesetzter Sprachen (wie z.B. BPEL und BPMN) abstrahieren. Sie konzentrieren sich vielmehr auf die Beschreibung weniger grundlegender Eigenschaften nebenläufiger Systemen, wie Interaktion, Kommunikation und Synchronisation. Ihre Entwicklung beginnt mit der Beschreibung von CSP (Hoare, 1978) und CCS (Milner, 1980) in den 1970er Jahren und setzt sich bis heute fort (Baeten, 2005).

Prozesskalküle stellen Konstrukte bereit, die das Zusammenspiel der Komponenten eines Systems (wie beispielsweise eines Geschäftsprozesses) auf abstraktem Niveau definieren. Die grundlegenden Primitive umfassen typischerweise parallele, sequentielle und bedingte Ausführung sowie Kommunikation. Ein System wird in seinem Startzustand durch einen Ausdruck beschrieben und kann anschließend „ausgeführt“ werden, indem anwendbare Ableitungsregeln des jeweiligen Kalküls auf den Ausdruck angewendet werden.

Das in Abbildung 2.4 dargestellte Beispiel beschreibt den BPMN-Geschäftsprozess aus Abbildung 2.3 durch einen Ausdruck der Sprache $\mu$BPMN (Höhn, 2010), einer vereinfachten Variante des $\pi$-Kalküls (Milner u. a., 1992), die eine Untermenge von BPMN formal fasst. Jede Zeile beschreibt eine Folge von Aktivitäten, die ausgelöst werden, wenn eine bestimmte *Nachricht* über einen bestimmten *Kanal* empfangen wird, die jeweils am Anfang der Zeile definiert sind.

Die erste Zeile drückt aus, dass der Prozess gestartet wird, wenn die Patientennummer (*pid*) über den Kanal $x$ empfangen wird, womit die Aktivität $\tau_{\mathsf{Enter\_PID}}$ ausgelöst wird. Nachdem $\tau_{\mathsf{Enter\_PID}}$ geendet hat, sendet sie *pid* über den Kanal $y$ weiter. Die nächsten

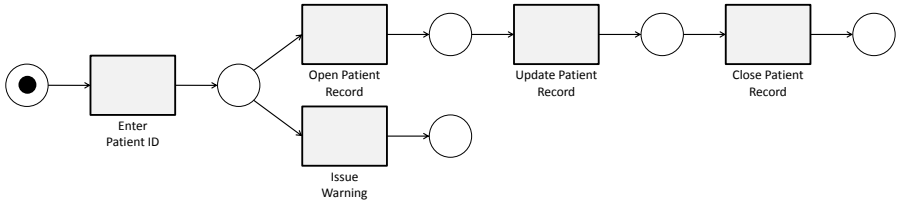



Abbildung 2.5.: Der Prozess aus Programmtext 2.1 in Petri-Netz-Darstellung.

beiden Zeilen verfügen über eine zusätzliche Bedingung in eckigen Klammern und modellieren die Verzweigung des Prozesses. Nur wenn die Bedingung erfüllt ist, wird die Nachricht *pid* über den jeweiligen Kanal ($z$ bzw. $r$) weitergesendet. Die letzten beiden Zeilen modellieren die alternativen Ausführungspfade nach der Verzweigung. Die jeweiligen Aktivitäten werden gestartet, wenn die Nachricht *pid* auf dem entsprechenden Kanal empfangen wird.

Prozesskalküle sind in mehreren Fällen für die formale Analyse von Geschäftsprozessen verwendet worden (vgl. etwa (Höhn, 2010; Lucchia und Mazzara, 2007; Wong und Gibbons, 2008; Ferrara, 2004; Salaün u. a., 2004)). Die modellierten Prozesseigenschaften beschränken sich dabei auf die Darstellung des Kontroll- und Datenflusses. Für die Modellierung weiterer Geschäftsprozess-Eigenschaften sowie für Sicherheitsanalysen, wie sie in dieser Arbeit durchgeführt werden, sind Erweiterungen notwendig.

### 2.2.3 Petri-Netze

Die zweite wichtige Familie formal fundierter Sprachen zur Modellierung von Geschäftsprozessen basiert auf Petri-Netzen. Petri-Netze wurden in den 1960er Jahren zur Beschreibung nebenläufiger Systeme eingeführt (Petri, 1962). Ein Petri-Netz ist ein bipartiter, gerichteter Graph mit zwei verschiedenen Typen von Ecken: *Stellen*, die durch Kreise dargestellt werden und *Transitionen*, die durch Rechtecke beschrieben werden. Stellen und Transitionen sind in einem Petri-Netz durch Kanten verbunden, wobei eine Kante immer eine Stelle mit einer Transition verbindet. Stellen können mit einer oder mehreren *Marken* belegt sein, die durch Punkte dargestellt werden. Die Belegung von Stellen mit Marken charakterisiert den Zustand eines Petri-Netzes und bestimmt die möglichen Zustandsübergänge. Ein Petri-Netz ändert seinen Zustand durch das *Schalten* bzw. *Feuern* einer Transition. Eine Transition kann genau dann feuern, wenn alle ihrer Eingangsstellen (d.h. die Stellen, die durch eine *eingehende* Kante mit der Transition verbunden sind) mit mindestens einer Marke belegt sind. Mit dem Schalten einer Transition wird jeweils eine Marke von allen Eingangsstellen entfernt und jeweils eine Marke zu allen Ausgangsstellen (d.h. den Stellen, die durch eine *ausgehende* Kante mit der Transition verbunden sind) hinzugefügt.

Abbildung 2.5 zeigt den in Programmtext 2.1 beschriebenen BPEL-Prozess in einer Petri-Netz-Darstellung. Die Aktivitäten des Prozesses werden durch Transitionen modelliert.

Im dargestellten Anfangszustand ist lediglich eine Stelle mit einer Marke belegt. Da die Stelle die einzige Eingangsstelle von Enter Patient ID ist, kann diese Transition feuern. In diesem Fall würde die Marke entfernt und eine Marke in der (einzigen) Ausgangstelle von Enter Patient ID platziert. Im Hinblick auf den entsprechenden Geschäftsprozess bedeutet der damit erreichte Zustand, dass die Aktivität Enter Patient ID ausgeführt wurde und der Prozess nun in einen der beiden möglichen Pfade verzweigen kann.

### Workflow Net

Der Einsatz von Petri-Netzen für die Modellierung von Geschäftsprozessen hat in den 1990er Jahren begonnen (Adam u. a., 1998). Hier ist der wohl am weitesten verbreitete Typ das *Workflow Net* (WF-Net) (van der Aalst, 1995, 1996; Lassen und van der Aalst, 2009). Ein WF-Net ist in seiner Essenz ein klassisches Petri-Netz, in dem – wie im Beispiel in Abbildung 2.5 – die Aktivitäten eines Geschäftsprozesses durch Transitionen modelliert werden. Die Definition von WF-Nets schränkt traditionelle Petri-Netze insofern ein, dass ein Netz eine definierte Start- und Endstelle haben und jede Ecke (d.h. Stelle oder Transition) auf einem Pfad von der Start- zur Endstelle liegen muss. Die Ausdrucksfähigkeit von WF-Nets beschränkt sich auf den Kontrollfluss eines Prozesses.

### Modellierung von Daten

Um neben dem Kontrollfluss auch Daten in einem Geschäftsprozesses modellieren zu können, schlägt Knorr (2001) die Aufteilung der Menge der Stellen in zwei Partitionen vor: Kontrollfluss- und Datenflussstellen. Während Erstere den üblichen Stellen in einem Petri-Netz entsprechen, werden Marken, die Datenflussstellen belegen, als Datenobjekte interpretiert, die von angeschlossenen Transitionen (d.h. Aktivitäten eines Geschäftsprozesses) gebraucht bzw. produziert werden. Es ist jedoch nicht möglich, explizite Zugriffsrechte auf Daten zu spezifieren: Statt dessen wird implizit festgelegt, dass auf jede von einer Transition konsumierte Marke *lesend*, und auf jede produzierte Marke *schreibend* zugegriffen wird.

*Workflow Nets with Data* (WFD-Nets) erweitern WF-Nets um Fähigkeiten zur Modellierung von Datenzugriffen (Trčka u. a., 2009). Zu diesem Zweck werden verschiedene Relationen (*Read*, *Write*, *Destroy*) zwischen der Menge der Transitionen und einer abstrakten Menge von Datenobjekten eingeführt. Im Gegensatz zum Modell von Knorr (2001) wird der Datenfluss allerdings nicht explizit durch Marken dargestellt. Des weiteren bieten WFD-Nets die Möglichkeit, an Datenobjekte geknüpfte Verzweigungsbedingungen zu modellieren und somit die Ausführung deterministisch zu machen.

#### Petri-Netze höherer Stufen

Um die Ausdrucksmächtigkeit klassischer Petri-Netze zu erhöhen, sind für bestimmte Zwecke erweiterte Petri-Netz-Typen entwickelt worden, die als Petri-Netze höherer Stufen (*High-Level Petri Nets*) bezeichnet werden. In Stochastischen Petri-Netzen können die Kanten mit einer Übergangswahrscheinlichkeit belegt werden (Marsan, 1990). *Time Petri Nets* ermöglichen die Spezifikation temporaler Einschränkungen bzgl. der Schaltdauer von Transitionen (Cassez und Roux, 2006). In *Petri Nets with Individual Tokens* und gefärbten Petri-Netzen (*Coloured Petri Nets*) können Marken mit Bezeichnern versehen und auf diese Weise unterscheidbar gemacht werden (Reisig, 1985; Jensen, 1996).

Konzepte von Petri-Netzen höherer Stufen sind für die Modellierung von Geschäftsprozessen verwendet worden. van der Aalst und van Dongen (2002) integrieren eine zeitliche Komponente in *Workflow Nets*, um die Geschwindigkeit der Ausführung von Arbeitsschritten darstellen zu können. Pesic und van der Aalst (2005) verwenden gefärbte Petri-Netze, um zwischen Ressourcen, die für die Bearbeitung einer Aktivität eingesetzt werden, differenzieren zu können.

### 2.2.4 Bewertung

Beschreibungssprachen wie BPEL und BPMN sind trotz ihrer weiten Verbreitung für Sicherheitsanalysen wenig geeignet, da sie über keine formale Semantik verfügen und mitunter mehrdeutig sind. Sowohl ein Petri-Netz- als auch ein Prozesskalkül-basierter Formalismus wären in dieser Hinsicht für die Geschäftsprozessbeschreibung geeignet, wobei in beiden Fällen die elementaren Formalismen ergänzt werden müssten.

In dieser Arbeit wird eine Petri-Netz-basierte Darstellung gewählt, weil diese zum Einen durch ihre grafische Notation (die Flussdiagrammen ähnelt) auch für Nichtexperten vergleichsweise leicht zu verstehen sind. Zum Anderen haben Petri-Netze sowohl im industriellen als auch im akademischen Bereich als Modellierungswerkzeug für Geschäftsprozesse weite Verbreitung gefunden (Ko, 2009; van der Aalst, 2005). Transformationen für viele praktische verwendete Beschreibungssprachen – wie etwa BPEL und BPMN – existieren.

Um als Grundlage für die Zertifizierung von Isolationsanforderungen genutzt werden zu können, müssen klassische Petri-Netze erweitert werden. Dies betrifft zum Einen Konstrukte für die Geschäftsprozess-Modellierung und zum Anderen ein ausdrucksstarkes Sicherheitsmodell:

- **Geschäftsprozess-Modellierung:** Für die Sicherheitsanalyse müssen alle wesentlichen Kanäle der Informationsübertragung in einer Prozess-Konstellation darstellbar sein, also insbesondere der Datenfluss. Datenobjekte müssen unterschieden und verschiedene Zugriffsmodi dargestellt werden können. Ebenso müssen Verzweigungsbedingungen abgebildet werden können, da auch auf diesem Wege Information übertragen werden kann (Denning und Denning, 1977). Um eine Prozess-

Konstellation mit verschiedenen Client-Prozessen beschreiben zu können, müssen Transitionen alternative Zustandsübergänge (Schaltregeln) zulassen.

- **Sicherheitsmodell:** Das Sicherheitsmodell muss ausdrücken können, welche Informationen voneinander isoliert sein müssen. Zu diesem Zweck müssen die in einer Prozess-Konstellation auftretenden Informationsträger – also Datenobjekte und Aktivitäten klassifiziert werden können. Ferner muss beschrieben werden können, welche Prozess-Teilnehmer welche Informationen erhalten dürfen.

Einige entsprechende Konstrukte finden sich bereits in den oben diskutierten Petri-Netz-Varianten. So kann etwa das Modell von Trčka u. a. (2009) Zugriffsmodi und Verzweigungsbedingungen darstellen. Die Unterscheidbarkeit von Marken findet sich bei Reisig (1985) und Jensen (1996). Andere Konstrukte wurden bisher nicht bzw. in einem für die Zwecke dieser Arbeit nicht ausreichendem Maße integriert. Das Modell von Knorr (2001) etwa kann Datenfluss darstellen, definiert aber für jede Transition eines fixes Ein-/Ausgabeverhalten (und damit einen festgelegten Zustandsübergang).

In dieser Arbeit wird mit IFnet daher eine spezielle Petri-Netz-Variante eingeführt, die auf WF-Nets aufbaut und die oben beschriebenen Konstrukte integriert. Bestehende Ansätze in dieser Richtung werden dabei, soweit möglich, aufgegriffen und ergänzt.

## 2.3 Verfahren zur Sicherheitsprüfung

Dieser Abschnitt diskutiert die Verfahren, die Geschäftsprozesse auf Sicherheitsverletzungen überprüfen können. Entsprechend der in Abbildung 2.2 dargestellten Einteilung werden nur diejenigen Verfahren betrachtet, die ein Geschäftsprozess-*Modell* untersuchen, um Verletzungen präventiv erkennen zu können.

Die Darstellung ist in zwei Abschnitte gegliedert. Im ersten Abschnitt werden Verfahren diskutiert, die eine *aktivitätsbezogene Autorisierung* in Geschäftsprozessen umsetzen. Diese bilden einen Schwerpunkt in der Sicherheitsforschung zu Geschäftsprozessen und beschäftigen sich mit Methoden um sicherzustellen, dass eine Aktivität nur von einer Partei mit ausreichenden Berechtigungen ausgeführt werden kann. Dementsprechend liegt der Fokus auf der Regulierung des Zugangs zu Aktivitäten, während Datenobjekte allenfalls am Rande betrachtet werden. Der zweite Abschnitt behandelt Verfahren, in denen auch der Datenfluss explizit modelliert und geprüft wird.

Praktisch alle vorgestellten Verfahren spezifizieren Sicherheitsanforderungen durch Datenzugriffspolicies (Sandhu und Samarati, 1994). Die Policies werden entweder systemweit (*Mandatory Access Control* – MAC), durch Nutzer (*Discretionary Access Control* – DAC) oder rollenbasiert (*Role Based Access Control* – RBAC)[3] festgelegt.

### 2.3.1 Aktivitätsbezogene Autorisierung

Der Betrachtungsgegenstand von Verfahren zur aktivitätsbezogenen Autorisierung sind die Aktivitäten eines Prozesses. Diese sollen ausführenden Subjekten nur im Einklang mit anzuwendenden Sicherheitsrichtlinien zugeordnet werden dürfen.[4] Neben der einfachen Zuordnung von Subjekten zu Aktivitäten können mit diesen Verfahren zum Teil weitere Sicherheits- und Compliance-Anforderungen geprüft werden, die als Abhängigkeiten zwischen Subjekt-Aktivität-Zuordnungen spezifiziert werden, wie z.B. *Separation of Duty* (Funktionstrennung) oder *Binding of Duty*.

#### RBAC-WS-BPEL

RBAC-WS-BPEL ist ein Rahmenwerk, das die Modellierungssprache BPEL um Möglichkeiten zur rollenbasierten Zugriffskontrolle erweitert (Paci u. a., 2008a). RBAC-WS-BPEL erlaubt die Spezifikation von Policies, die angeben, welche Rolle oder welcher Benutzer autorisiert ist, eine Aktivität in einem Prozess auszuführen. In einer Policy

[3] In einem RBAC-Modell werden den Nutzern eines Systems in Abhängigkeit ihrer Aufgaben *Rollen* zugewiesen, die die Zugriffsmöglichkeiten der Nutzer bestimmen (Ferraiolo und Kuhn, 1992). RBAC ist ein mächtiges Modell, das sowohl MAC- als auch DAC-Policies simulieren kann.

[4] Subjekte können sowohl reale Personen sein, die manuelle Aufgaben in einem Prozess durchführen, als auch Nutzerkennungen in einem Computersystem, mit deren Privilegien ein entsprechender Dienst ausgeführt wird.

können Rollenhierarchien, Relationen zur Verknüpfung von Rechten (*permissions*) mit Rollen sowie Mengen von Rechten, die die Möglichkeiten zur Ausführung von Aktivitäten modellieren, definiert werden. Darüber hinaus können Einschränkungen (*constraints*) bezüglich der Ausführung von Aktivitäten, wie beispielsweise zur Erzwingung von Funktionstrennung, angegeben werden. RBAC-WS-BPEL dient insbesondere der Autorisierung von Aktivitäten eines Prozesses, die teilweise oder vollständig von Menschen ausgeführt werden. Es wird in einer anderen Arbeit erweitert, um sogenannte Resilienzeinschränkungen (*resiliency constraints*) ausdrücken zu können (Paci u. a., 2008b). Hier ist das Ziel, die Ausführung eines Prozesses garantieren zu können, auch wenn bestimmte Nutzer ausfallen.

### Xiangpeng u. a. (2006)

Xiangpeng u. a. (2006) schlagen ein vergleichsweise einfaches RBAC-Modell zur Autorisierung von BPEL-Prozessen vor, in dem Rechte zur Ausführung grundlegender Aktivitäten (*elementary activities*) spezifiziert werden können. Außerdem können Anforderungen zur Funktionstrennung bestimmt werden. Policies werden durch Formeln einer linearen temporalen Logik beschrieben. Ein Modellprüfer wird benutzt, um zu entscheiden, ob ein BPEL-Prozess die Anforderungen erfüllt. Im Vergleich zu RBAC-WS-BPEL besitzt der Ansatz eine kleinere Ausdrucksmächtigkeit.

### $R^2$BAC

Das *Role-and-Relation-based Access Control*-Modell ($R^2$BAC) ist eng verwandt mit RBAC-WS-BPEL (Wang und Li, 2007). Hier können neben der Beziehung von Nutzern zu Rollen auch Beziehungen zwischen Nutzern spezifiziert werden, um Berechtigungen zum Ausführen von Aktivitäten festzulegen. Damit kann beispielsweise festgelegt werden, dass zwei Nutzer bestimmte Aktivitäten nicht ausführen dürfen, wenn zwischen ihnen ein Interessenkonflikt besteht. In der Arbeit wird untersucht, ob eine Gruppe von Nutzern einen Prozess unter Einhaltung der angegebenen Sicherheitspolicy ausführen kann, und gezeigt, dass dieses Problem für $R^2$BAC NP-vollständig ist. Zudem werden mehrere aufeinander aufbauende Definitionen von Resilienz vorgeschlagen, die angeben, unter welchen Bedingungen ein Prozess ausgeführt werden kann, wenn bestimmte Nutzer ausfallen (siehe den Abschnitt zu RBAC-WS-BPEL).

### Koshutanski und Massacci (2003)

Wie die oben beschriebenen Modelle definiert das von Koshutanski und Massacci (2003) eine rollenbasierte Zugriffskontrolle, mit der die Rechte zur Ausführung der Aktivitäten eines BPEL-Prozesses festgelegt und geprüft werden können. Autorisierungen werden in diesem Modell durch logische Formeln beschrieben, die Rollen und Benutzern zugeordnet werden können. Im Gegensatz zu den vorher beschriebenen Modellen ist es hier möglich,

sogenannte *Release Policies* anzugeben, um zusätzliche Berechtigungsnachweise (*credentials*) zu bestimmen, die ein Nutzer erbringen muss, dem zuvor ein Zugriff verwehrt wurde.

### Wolter und Schaad (2007)

Während die oben vorgestellten Verfahren BPEL erweitern, schlagen Wolter und Schaad (2007) ein Modell zur aktivitätsbezogenen Autorisierung von BPMN-Prozessen vor. Dazu wird eine formale Notation von Autorisierungsregeln (*Authorization Constraints*) eingeführt, mit der sich Sicherheitsanforderungen wie die rollen- und historiebasierte Zuweisung von Aktivitäten sowie *Separation of Duty* und *Binding of Duty* ausdrücken lassen. Um die Anforderungen in einen BPMN-Prozess zu integrieren, wird das Meta-Modell um mehrere Elemente erweitert. Neben einem Annotationselement zur Beschreibung der Autorisierungsregeln umfassen diese einen neuen Aktivitätstyp (manuelle Aktivität) sowie ein Artefakt zur Gruppierung von Regeln und Aktivitäten. Das Modell ist in verschiedenen Richtungen erweitert worden, etwa um die spezifizierten Autorisierungsregeln mit einem Modellprüfer auf Konsistenz zu prüfen (Wolter u. a., 2009a) und um plattformspezifische Zugriffsregeln daraus abzuleiten (Wolter u. a., 2009b).

### Workflow Authorization Model

Neben den Rechten zur Ausführung von Aktivitäten erlaubt das *Workflow Authorization Model* (WAM) zusätzlich eine Kontrolle über die damit verbundenen Objekte (Atluri und kuang Huang, 1997; Atluri und Huang, 1996). So ist es möglich festzulegen, dass Benutzer nur während der Ausführung der jeweiligen Aktivität über Rechte zum Zugriff auf damit verbundene Objekte (wie beispielsweise Dokumente) verfügen und ihnen diese Rechte mit Beendigung der Aktivität wieder entzogen werden sollen. Auf diese Weise wird der Ablauf eines Prozesses mit einer angepassten Rechtevergabe gekoppelt. Zugriffsrechte werden in einer „Schablone“ (*Authorization Template*) angegeben, mit der jede Aktivität in einem Prozessmodell versehen wird. Das WAM sieht ein transaktionsorientiertes Ausführungsmodell vor, in dem die Ausführung einer Aktivität einer Transaktion entspricht. Damit sowohl die Abfolge der Aktivitäten korrekt ist, als auch die Sicherheitsanforderungen erfüllt werden, muss eine Transaktion Abhängigkeiten zu anderen Transaktionen und zu sie betreffenden Objekten genügen. Das WAM geht damit über die rein strukturelle Betrachtung eines Prozesses hinaus, berücksichtigt den Zugriff auf Objekte aber nur im Zusammenhang mit der Ausführung von Aktivitäten. Interessant ist das WAM zudem, weil es – als einziges uns bekanntes Modell – verdeckte Informationsübertragung betrachtet, die sich durch Abhängigkeiten zwischen unterschiedlich klassifizierten Transaktionen bzw. Objekten ausdrückt. Es kann allerdings keine verdeckte Übertragung erkennen, die über derartige Abhängigkeiten hinausgeht (wie z.B. Datenzugriffskonflikte).

Das WAM wird in einer späteren Arbeit erweitert, um *Chinese Wall Policies* ausdrücken und prüfen zu können (Atluri u. a., 2004). Das Chinese-Wall-Sicherheitsmodell wurde ursprünglich entwickelt, um Interessenkonflikte in Beratungsfirmen zu unterbinden (Brewer und Nash, 1989). Es verhindert, dass Berater auf Objekte von mehreren Kunden zugreifen, die miteinander konkurrieren. In der Arbeit von Atluri u. a. (2004) werden zu diesem Zweck die Aktivitäten eines verteilt ausgeführten Geschäftsprozesses in Partitionen eingeteilt. Ein Algorithmus sorgt dafür, dass ausführende Subjekte diesen Partitionen so zugeteilt werden, dass keine Interessenkonflikte entstehen können.

### Bertino u. a. (1999)

In dem Modell von Bertino u. a. (1999) werden Geschäftsprozesse als Zuordnungen von Aktivitäten zu Rollen (*Task-Role-Specifications*) ausgedrückt, wobei allerdings nur die sequentielle Anordnung von Aktivitäten unterstützt wird. Durch die Zuordnungen werden Aktivitäten mit den Rollen verknüpft, die berechtigt sind, sie auszuführen. Darüber hinaus kann eine Grenze für die Anzahl der Ausführungen einer Aktivität in einer Prozess-Instanz bestimmt werden. Abhängigkeiten zwischen Rollen und Benutzern, etwa um eine statische oder dynamische Funktionstrennung zu beschreiben, werden durch logische Formeln ausgedrückt. Darüber hinaus beinhaltet der Ansatz Algorithmen, um alle möglichen Rollen- und Nutzer-Zuordnungen zu Aktivitäten zu berechnen, die die Einhaltung der Anforderungen gewährleisten.

### Casati u. a. (2001)

Im Rahmenwerk von Casati u. a. (2001) werden den Aktivitäten eines Prozesses Rollen, Organisationsstufen und Nutzer zugeordnet. Rollen und Organisationsstufen können hierarchisch angeordnet werden, um die Zuordnung von Aktivitäten zu Nutzern zu vereinfachen. Zuordnungen von Nutzern zu Rollen und Organisationsstufen sowie von Rollen und Organisationsstufen zu Aktivitäten können sowohl generell festgelegt werden als auch in Abhängigkeit von Parametern wie der Ausführungszeit, der Instanz eines Prozesses sowie der Ausführungshistorie. Im letzteren Fall werden die Zuordnungen entsprechend den Vorgaben geändert, wenn eine bestimmte Bedingung (z.B. ein Zeitpunkt oder ein bestimmter Wert eines Datums) eintritt.

### Armando und Ponta (2010)

Im Ansatz von Armando und Ponta (2010) werden Geschäftsprozesse und Autorisierungsregeln getrennt spezifiziert und mit dem SATMC-Modellprüfer gegeneinander verifiziert. Geschäftsprozesse werden hier durch Petri-Netze und Autorisierungsregeln durch Formeln einer linearen temporalen Logik (LTL) beschrieben. Die Grundlage zur Richtlinienspezifikation bilden rollenbasierte Autorisierungsregeln (d.h. eine Zuordnung von Subjekten

zu Rollen sowie von Rollen zu Aktivitäten) sowie Delegationsregeln, mit denen die Bedingungen festgelegt werden können, unter denen die Ausführung einer Aktivität an eine andere Rolle delegiert werden darf. Für die Verifikation werden ein Geschäftsprozess-Modell und die entsprechenden LTL-Formeln in ein sogenanntes *Planning System* überführt (ein formales Modell für die Spezfikation nebenläufiger Systeme), das die Eingabe für den SATMC-Modellprüfer bildet.

### 2.3.2 Datenflussprüfung

Verfahren zur Datenflussprüfung verfügen über ein explizites Modell des Datenaustauschs in einem Geschäftsprozess und prüfen dieses bzgl. der Einhaltung spezifizierbarer Einschränkungen. Einige Ansätze befassen sich mit dem Problem, verbreitete MAC-Policies (insbesondere das Bell-LaPadula-Modell (Bell und LaPadula, 1976)) auf den Prozesskontext zu übertragen. Des weiteren gibt es Verfahren, die in Geschäftsprozessen häufig auftretende Datenfluss-Fehler (wie z.B. das Auftreten inkonsistenter Daten) formalisieren und erkennen können. Letztere überlappen sich stark hinsichtlich der Bandbreite der erkannten Fehler und werden daher vergleichsweise knapp in einem zusammengefassten Abschnitt dargestellt.

#### Mikolajczak und Gami (2008)

Der Fokus des Verfahrens von Mikolajczak und Gami (2008) liegt auf der Kontrolle des Datenaustauschs zwischen interagierenden Geschäftsprozessen von unterschiedlichen Unternehmen. Zu diesem Zweck werden sowohl „positive“ (d.h. erlaubte) als auch „negative“ (d.h. unerwünschte) Kommunikationsmuster durch *Message Sequence Charts* beschrieben. Anschließend werden diese Muster in eine Petri-Netz-Darstellung der interagierenden Prozesse übertragen, in der positives Verhalten möglich ist und negatives Verhalten ausgeschlossen bleibt. Allerdings bleiben die Autoren an dieser Stelle unklar darüber, wie dieser Übertragungsschritt automatisiert werden soll bzw. ob es eine manuelle Tätigkeit des Prozess-Designers bleibt. Als zusätzlicher Beitrag der Arbeit wird ein Algorithmus für die Integration eines stufenbasierten Sicherheitsmodells vorgestellt, bei dem Subjekte und Objekte mit Sicherheitsstufen ausgezeichnet werden, die hierarchisch angeordnet sind. In dem Modell ist ein Zugriff auf ein Datenobjekt durch ein Subjekt nur dann erlaubt, wenn die Sicherheitsstufe des Subjektes größer oder gleich der des Objektes ist. Der Algorithmus wird weitgehend informell beschrieben und es bleibt auch hier unklar, ob er automatisiert werden soll oder als manuelle Tätigkeit vorgesehen ist. Das Sicherheitsmodell wird anschließend in die Petri-Netz-Darstellung der interagierenden Prozesse integriert, wobei dazu kein Verfahren angegeben, sondern das Resultat lediglich anhand eines Beispielnetzes illustriert wird. Eine formale Definition des erweiterten Petri-Netzes und der darin enthaltenden Modellierung von Daten und Zugriffsbeschränkgungen fehlt.

### Barkaoui u. a. (2008)

Mit dem Verfahren von Barkaoui u. a. (2008) können Geschäftsprozesse auf Korrektheitseigenschaften (*Soundness*) und die Einhaltung von Bell-LaPadula-Policies geprüft werden. Ausgangspunkt bilden Prozesse in WF-Net-Darstellung, die allerdings keinen Datenfluss modellieren können. Aus diesem Grund werden diese in ein *Extended Concurrent Algebraic Term Net* (ECATNet) übertragen, das für die Modellierung abstrakter Datentypen geeignet ist (Bettaz und Maouche, 1993), wobei keine automatisierbare Abbildung angegeben wird. Ähnlich wie in dem Systemmodell von Knorr (2001) wird in einem ECATNet die Unterscheidung zwischen Datenobjekten nicht durch unterscheidbare Marken ermöglicht, sondern durch eine Typisierung der Stellen. Zugriffsmodi auf Datenobjekte werden als Bedingungen an die Eingangsstellen von Transitionen beschrieben. Für die Verifikation eines auf diese Weise dargestellten Geschäftsprozesses wird ein ECATNet in eine Menge von *Rewrite*-Regeln übertragen, die als Eingabe für den MAUDE-Beweiser fungieren. Mit diesem kann überprüft werden, ob in einem Geschäftsprozess Datenzugriffe stattfinden können, die nach dem Bell-LaPadula-Modell verboten sind, d.h. ob ein höher eingestuftes Subjekt schreibend auf ein niedriger eingestuftes Datenobjekt bzw. ob ein niedriger eingestuftes Subjekt lesend auf ein höher eingestuftes Datenobjekt zugreifen kann.

### Knorr (2001)

Knorr (2001) untersucht ebenfalls die Durchsetzung von Bell-LaPadula-Policies in Petri-Netz-basierten Geschäftsprozessen. Die Arbeit benutzt als zentrale Datenstruktur den Abdeckungsgraphen (*Coverability Graphs*) eines Petri-Netzes. Der Abdeckungsgraph beschreibt alle möglichen Markierungen, die das Petri-Netz ausgehend von seiner initialen Konfiguration annehmen kann. In der für die Bell-LaPadula-Analyse angepassten Variante des Abdeckungsgraphen werden zusätzlich zu den möglichen Belegungen von Stellen mit Marken auch alle möglichen Sicherheitseinstufungen der Datenobjekte dargestellt. Ein solcher Graph kann genutzt werden um festzustellen, ob es für eine Gruppe von Subjekten (die Sicherheitsfreigaben besitzen) eine Klassifizierung der Datenobjekte gibt, mit der ein Geschäftsprozess in Konformität mit den BLP-Regeln ausgeführt werden kann.

### Erkennung von Datenflussfehlern

Die Grundlage für die meisten Arbeiten zu Datenflussfehlern in Geschäftsprozessen bildet die Arbeit von Sadiq u. a. (2004). Hier werden sieben typische Datenflussprobleme vorgestellt, die das korrekte Funktionieren eines Prozesses behindern können. Diese sind:

1. Redundante Daten (*redundant data*): Produzierte Daten werden nie gebraucht.
2. Verlorene Daten (*lost data*): Das gleiche Datenelement wird von mehreren parallel ablaufenden Aktivitäten produziert.

3. Fehlende Daten (*missing data*): Eine Aktivität benötigt ein Datenelement, das zum Zeitpunkt ihrer Ausführung nicht vorhanden ist.

4. Falsch zugeordnete Daten (*mismatched data*): Die Struktur eines bereitgestellten Datenobjektes ist inkompatibel mit der erwarteten.

5. Inkonsistente Daten (*inconsistent data*): Ein zwischengespeichertes Datenelement wird extern (d.h. außerhalb des Prozesses) geändert.

6. Fehlgeleitete Daten (*misdirected data*): Die Aktivität, die ein Datenobjekt bereitstellt, wird erst nach der Aktivität ausgeführt, die es benötigt.

7. Unzureichende Daten (*insufficient data*): Die spezifizierten Daten sind nicht ausreichend, um eine Aktivität auszuführen.

Die im Folgenden diskutierten Ansätze prüfen Geschäftsprozesse jeweils auf eine Teilmenge dieser möglichen Fehler.

Das Verfahren von Meda u. a. (2010) betrachtet die drei Fehlertypen „Fehlende Daten“, „Verlorene bzw. inkonsistente Daten“ (*inconsistent/lost data*) und „Redundante Daten“. Geschäftsprozesse werden in der Arbeit als Graphen mit verschiedenen Eckentypen dargestellt, die Aktivitäten und Verzweigungen beschreiben. Die Datenperspektive wird durch eine zentrale Datenbank modelliert, auf die alle Aktivitäten zugreifen können, d.h. der Fluss der Daten durch den Prozess wird nicht explizit dargestellt. Für die Prüfung wird ein Graph-Traversierungsalgorithmus angegeben.

Die Methode von Trčka u. a. (2009) benutzt WFD-Nets (siehe Abschnitt 2.2.3) für die Prozess-Darstellung und beschreibt Datenflussfehler durch Formeln einer temporalen Logik. Um diese mit bestehenden Petri-Netz-basierten Modellprüfern verifizieren zu können, werden WFD-Nets in normale Petri-Netze überführt. In diesen werden Transitionen des WFD-Nets durch komplexe Ausdrücke beschrieben, die die Daten- und Kontrollflussbedingungen für die Ausführung der entsprechenden Aktivität darstellen. Es können mit dieser Methode im Wesentlichen die gleichen Datenflussfehler wie im Ansatz von Meda u. a. (2010) geprüft werden. Zusätzlich ermöglicht sie die Erkennung von nicht rechtzeitig gelöschten Daten (*not deleted on time*).

In dem Modell von Sun u. a. (2006) wird die Datenfluss-Perspektive auf einen Geschäftsprozess durch eine zweidimensionale „Datenfluss-Matrix“ (*Data-Flow Matrix*) dargestellt, in der die Datenzugriffsoperationen (lesen/schreiben) für jede Aktivität und jedes Datenobjekt verzeichnet sind. Es werden drei Arten von „Datenfluss-Anomalien“ eingeführt: Fehlende Daten, Redundante Daten und Konfligierende Daten, die im Wesentlichen die ersten sechs Fehlertypen aus der Aufstellung von Sadiq u. a. (2004) abdecken. Die Prüfung erfolgt auf der Basis der Datenfluss-Matrix, wobei die durch den Geschäftsprozess gegebenen Abhängigkeiten zwischen Datenobjekten in Betracht gezogen werden.

### 2.3.3 Bewertung

Tabelle 2.1 fasst die Bewertung der diskutierten Verfahren hinsichtlich der in Kapitel 1 aufgestellten Anforderungen zusammen. Ein Verfahren wird hinsichtlich des zu Grunde liegenden Prozess-Metamodells, den bereitgestellten Isolationsgarantien und dem Grad seiner Automatisierbarkeit bewertet, wobei jedes Kriterium in zwei oder drei Unterpunkte gegliedert ist. In der Tabelle steht ein Haken (✓) für die Erfüllung einer Anforderungen und ein Querstrich (–) für die Nichterfüllung. Ein Haken in Klammern bedeutet, dass die Anforderung teilweise erfüllt wird.

Nur die Hälfte der diskutierten Verfahren arbeitet auf der Basis eines formal fundierten Geschäftsprozess-Modells. Wie im ersten Abschnitt dieses Kapitels ausgeführt, erfüllen insbesondere die Ansätze, die auf BPEL oder BPMN basieren, diese Anforderung nicht. Die benötigte Ausdrucksstärke eines Meta-Modells bemisst sich an der Liste der in Abschnitt 2.2.4 identifizierten Konstrukte, über die ein Meta-Modell für die Zertifizierung von Sicherheitsanforderungen verfügen muss und die über die rein strukturelle Darstellung eines Geschäftsprozesses hinausgehen. Während die industriellen Modellierungssprachen BPEL und BPMN über die erforderliche Mächtigkeit verfügen (aber nicht semantisch fundiert sind), stellt nur eine kleine Menge der übrigen Meta-Modelle – wie in Abschnitt 2.2.4 dargestellt – jeweils einen Teil dieser Konstrukte bereit.

Der weit überwiegende Teil der Verfahren formalisiert Sicherheits- bzw. Isolationsanforderungen als Mengen von Zugriffsregeln, wodurch keine Garantien bzgl. der vollständigen Erfassung aller möglichen Informationspfade gegeben ist (vgl. Abschnitt 1.5). Lediglich die Verfahren von Barkaoui u. a. (2008) und Knorr (2001) formulieren Anforderungen als Ende-zu-Ende-Eigenschaften, indem sie beide eine Form des Bell-LaPadula-Modells in Geschäftsprozessen umsetzen. Hinsichtlich der betrachteten Informationskanäle können die Verfahren zur aktivitätsbezogenen Autorisierung den Austausch von Daten nur indirekt kontrollieren, durch die Regulierung des Zugriffs auf die Aktivitäten und den damit verbundenen Informationen. Die Verfahren zur Erkennung von Datenflussfehlern – Meda u. a. (2010), Trčka u. a. (2009) und Sun u. a. (2006) – beschränken sich auf eine Menge fest definierter Fehlertypen, sind aber nicht in der Lage, die Verbreitung von Datenobjekten wirksam zu kontrollieren. Einzig das *Workflow Authorization Model* verfügt über einen grundlegenden Mechanismus zur Erkennung von Informationsübertragung abseits direkter Datenzugriffe (verdeckte Kanäle). Dieser beschränkt sich allerdings auf einfache Abhängigkeiten zwischen unterschiedliche klassifizierten Aktivitäten. Verdeckte Informationsübertragung, die durch Ressourcenkonflikte induziert wird, wird dagegen nicht betrachtet.

Bei allen diskutierten Verfahren verläuft der eigentliche Prüfungsvorgang automatisch. Für die Anwendung im Kontext des Geschäftsprozess-Management ist es darüber hinaus notwendig, dass auch weitere Arbeitsschritte, die im Rahmen der Zertifizierung anfallen, automatisierbar sind (vgl. Abschnitt 1.1.2). Zum Einen betrifft dies die Erzeugung des Prozess-Modells, auf dessen Grundlage ein Verfahren arbeitet. Dieser Schritt entfällt bei Verfahren, die direkt auf der Grundlage industrieller Modellierungssprachen wie BPEL

oder BPMN arbeiten, da diese in der Regel unmittelbar für den Entwurf, die Dokumentation und die Ausführung von Prozessen verwendet werden. Bei den übrigen Verfahren werden keine Angaben bzgl. der Erzeugung eines Modells gemacht. Zwar existieren zum Teil generische Verfahren, um etwa Petri-Netze aus anderen Prozessbeschreibungen zu erzeugen (vgl. Abschnitt 2.2.1), diese sind aber nicht direkt kompatibel zu den von den jeweiligen Verfahren benutzten Varianten (im Fall der Petri-Netze etwa die Verfahren von Knorr (2001) oder Trčka u. a. (2009)). Zum Anderen ist eine Automatisierung bei der Ableitung der benutzten Policies sinnvoll. Dazu müssen Anforderungen in einer üblicherweise benutzten Darstellung – wie z.B. einer Zugriffskontrollvorschrift – in eine Richtlinie übertragen werden, die von dem entsprechenden Verfahren direkt geprüft werden kann (vgl. das Vorgehensmodell zur Compliance-Automatisierung in Abbildung 1.1 und die Diskussion in Abschnitt 1.4). Zu diesem Punkt werden von keinem Verfahren Ansätze aufgezeigt.

| Verfahren | Prozess-Metamodell | | Isolationsgarantien | | | Automatisierbarkeit | |
|---|---|---|---|---|---|---|---|
| | Formale Semantik | Ausdrucks-stärke | Ende-zu-Ende-Policies | Datenfluss | Verdeckte Kanäle | Modell-erzeugung | Ableitung von Policies |
| RBAC-WS-BPEL | – | ✓ | – | – | – | ✓(BPEL) | – |
| Xiangpeng u. a. | – | ✓ | – | – | – | ✓(BPEL) | – |
| Koshutanski und Massacci | – | ✓ | – | – | – | ✓(BPEL) | – |
| Wolter und Schaad | – | ✓ | – | – | – | ✓(BPMN) | – |
| Workflow Authorization Model | ✓ | (✓) | – | (✓) | (✓) | – | – |
| Bertino u. a. | ✓ | – | – | – | – | – | – |
| Casati u. a. | ✓ | – | – | – | – | – | – |
| Armando und Ponta | ✓ | – | – | – | – | – | – |
| Mikolajczak und Gami | – | – | – | ✓ | – | – | – |
| Barkaoui u. a. | ✓ | – | ✓ | ✓ | – | – | – |
| Knorr | ✓ | (✓) | ✓ | ✓ | – | – | – |
| Meda u. a. | – | – | – | (✓) | – | – | – |
| Trčka u. a. | ✓ | (✓) | – | (✓) | – | – | – |
| Sun u. a. | – | – | – | (✓) | – | – | – |

Tabelle 2.1.: Vergleich der Prüfungsverfahren.

Kapitel 3

# Geschäftsprozess-Zertifizierung mit InDico

Dieses Kapitel stellt den Kern der Arbeit dar und beschreibt InDico, ein automatisiertes Vorgehensmodell für die Zertifizierung von Isolationsanforderungen in Geschäftsprozessen. Im folgenden Abschnitt wird zunächst der Ansatz von InDico beschrieben. Die darauf folgenden Abschnitte stellen anschließend die einzelnen Komponenten im Detail dar.

## 3.1 Der InDico-Ansatz

InDico setzt in der Entwurfsphase des Geschäftsprozessmanagements an (siehe Abbildung 1.2) und zielt darauf ab, mögliche Sicherheitsverletzungen vor der Prozessausführung zu erkennen und damit die Möglichkeit zu ihrer frühzeitigen Beseitigung zu geben. Die Grundlage, auf der InDico arbeitet, bildet IFnet, ein ausdrucksstarkes und formal fundiertes Meta-Modell für die Beschreibung von Geschäftsprozessen. Die Isolation von Informationen wird in InDico durch Informationsfluss-Policies beschrieben, die sowohl im Hinblick auf Datenfluss als auch auf verdeckte Informationsübertragung geprüft werden. Zur Automatisierung des Zertifizierungsvorgangs stellt InDico eine Abbildung von BPMN-Prozessen auf IFnet sowie Strategien zur Ableitung von Informationsfluss-Policies aus Sicherheitsanforderungen bereit.

Abbildung 3.1 gibt einen schematischen Überblick über die Zertifizierung mit InDico und seine Komponenten. Die Eingabe für die Zertifizierung bilden Prozessmodelle in BPMN-Notation, die im ersten Schritt in das Petri-Netz-basierte IFnet, überführt werden. Im zweiten Schritt wird das erzeugte IFnet-Modell mit Sicherheitsstufen ausgezeichnet, die eine Anforderung – wie z.B. die Isolation nebenläufig ausgeführter Prozess-Instanzen – als Einschränkung an den in einer Prozess-Konfiguration erlaubten Informationsfluss ausdrücken. Der dritte Schritt ist die statische Prüfung des mit Sicherheitsstufen ausgezeichneten IFnet-Modells, in der das Modell auf die Möglichkeit einer Sicherheitsverletzung hin untersucht wird. Das Resultat der Prüfung mit InDico ist ein Zertifikat, das den

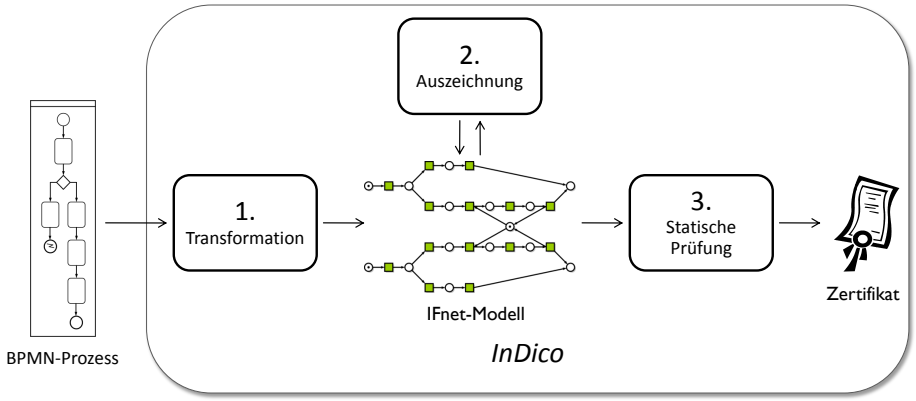



Abbildung 3.1.: Übersicht des InDico-Vorgehensmodells.

untersuchten Prozessen die Einhaltung der überprüften Anforderungen bescheinigt oder aber die Möglichkeit von Verletzungen anhand von Gegenbeispielen zeigt.

InDico eignet sich sowohl für die Prüfung von Geschäftsprozessen die vollständig automatisiert sind, d.h. solche, in denen sämtliche Aktivitäten von Software-Diensten ausgeführt werden, als auch für Prozesse, in denen Teilaufgaben von Menschen erledigt werden (van der Aalst u. a., 2007). Voraussetzung ist lediglich, dass ein explizites Prozessmodell vorliegt, welches die Grundlage für die Zertifizierung bildet. Dieses wird in der Regel in der Entwurfsphase von Prozess-Designern spezifiziert. Alternativ besteht etwa die Möglichkeit, ein Prozessmodell mit forensischen Methoden aus Log-Aufzeichnungen vergangener Ausführungen zu rekonstruieren (van der Aalst und Weijters, 2004).[1]

Anwendungsszenarien für InDico sind Geschäftsprozess-Konstellation, in denen Teilnehmer aus verschiedenen Sicherheitsdomänen kooperieren oder auf geteilten Plattformen zusammenkommen. Eine Sicherheitsdomäne fasst die Teilnehmer zusammen, die über die gleichen Rechte an den in der Prozesskonstellation auftretenden Informationen verfügen. Mögliche Szenarien sind etwa:

- **Dienst-Plattformen:** Ein Dienstleister bietet – wie im Szenario zu Beginn von Kapitel 1 dargestellt – seinen Kunden die Ausführung von (Teil-)Prozessen auf der eigenen Infrastruktur an. Durch die Zertifizierung kann der Dienstleister zeigen, dass die von ihm zu gewährleistenden Sicherheitsanforderungen, wie z.B. die Isolation von Prozessen unterschiedlicher Kunden, erfüllt werden. Die zertifizierte Gewährleistung der Sicherheitsanforderungen auf der technischen Ebene kann als Basis für entsprechende rechtliche Garantien gegenüber den Kunden dienen.

- **Organisationsübergreifend integrierte Prozesse:** Eine Organisation integriert ihre Prozesse mit denen von Geschäftspartnern. Die Zertifizierung stellt sicher, dass

[1] Diese Möglichkeit wird ausführlicher in Kapitel 6 diskutiert.

keine sensiblen Informationen an die externen Partner abfließen bzw. von diesen manipuliert werden können.

- **Innerorganisatorische Prozesse:** Innerhalb einer Organisation sind Parteien mit unterschiedlichen Sicherheitsfreigaben in die Ausführung eines Prozesses involviert und dürfen nur Informationen erhalten, die ihrer Freigabe entsprechen. Die Zertifizierung bescheinigt, dass keine Partei bei der Prozess-Ausführung Informationen erhalten kann, die über ihre Freigabe hinausgehen.

### 3.1.1 Komponenten

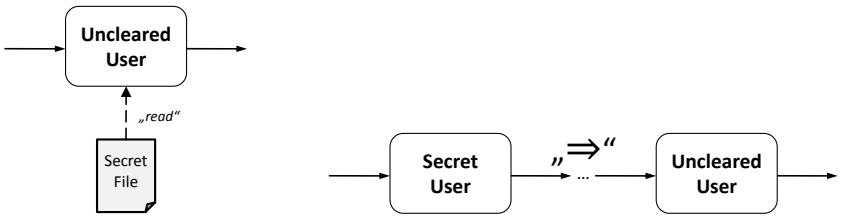
Im Folgenden wird die Aufgabe der Komponenten von InDico, die in dem Schaubild aus Abbildung 3.1 vorkommen, und ihr Zusammenwirken knapp dargestellt.

**IFnet** IFnet ist eine Petri-Netz-basierte Modellierungssprache für Geschäftsprozesse. Das Ziel der Entwicklung von IFnet ist die Möglichkeit zur formal fundierten Beschreibung der Bestandteile eines Geschäftsprozesses, die für Sicherheitsanalysen wesentlich sind (vgl. Abschnitt 2.2.4).

**BPMN-Transformation** Um den InDico-Ansatz unmittelbar für die Zertifizierung bestehender Prozesse anwendbar zu machen, wird eine automatisierbare Abbildung angegeben, die BPMN-Konstrukte auf entsprechende IFnet-Muster übersetzt.

**Auszeichnung (Labeling)** Ein zentrales Merkmal von InDico ist die Nutzung extensionaler Informationsfluss-Policies, die eine Sicherheitsanforderung als Einschränkung an den in einer Prozess-Konstellation erlaubten Informationsfluss ausdrücken. Diese Einschränkungen werden als Ende-zu-Ende-Eigenschaften ausgedrückt, indem die Quellen und Senken möglicher Informationsflüsse mit Sicherheitsstufen bzw. Freigaben annotiert werden. Auszeichnungsstrategien führen diese Annotierung im Hinblick auf eine bestimmte Sicherheitsanforderung – wie z.B. die Isolation von Prozess-Instanzen – automatisch durch. Sie stellen damit ein Bindeglied zwischen abstrakt formulierten Anforderungen und konkreten technischen Policies dar (vgl. dazu das Vorgehensmodell zur in Abbildung 1.1).

**Statische Prüfung** Die statische Prüfung analysiert ein mit Sicherheitsstufen ausgezeichnetes IFnet-Modell auf die Möglichkeit illegaler (d.h. die jeweilige Anforderung verletzender) Informationsübertragung. Die Entscheidungsgrundlage bildet ein Isolationskriterium, das die Möglichkeiten der unautorisierten Informationsübertragung in einem IFnet-Modell formal fasst. Dabei werden sowohl der direkte Datenaustausch als auch Möglichkeiten der verdeckten Informationsübertragung berücksichtigt.




(a) Datenfluss (b) Kausalität

Abbildung 3.2.: Datenfluss und verdeckte Informationsübertragung.

**Zertifikate** InDico generiert als Ergebnis der statischen Prüfung ein Zertifikat bzgl. der Erfüllung der Sicherheitsanforderungen durch ein gegebenes IFnet-Modell. Im Fall einer Verletzung werden Gegenbeispiele generiert, die die Verletzung nachvollziehbar machen.

### 3.1.2 Isolationsgarantien

InDico überprüft, ob Informationen, die in einer Konstellation interagierender Geschäftsprozesse verarbeitet und dargestellt werden, gegebenen Anforderungen entsprechend isoliert sind. Um die Anforderungen auszudrücken, werden die Träger der Informationen mit Sicherheitsstufen ausgezeichnet. Sicherheitsstufen gruppieren Informationen mit dem gleichen Schutzanspruch (Denning, 1976). Eine Relation auf der Menge der Sicherheitsstufen bestimmt, zwischen welchen Stufen Information fließen darf bzw. nicht fließen darf.

Der von InDico verwendete Isolationsbegriff umfasst die Abwesenheit ungewollter Informationsübertragung sowohl durch direkten Datenzugriff als auch über verdeckte Kanäle. Abbildung 3.2 illustriert beispielhaft jede dieser beiden Formen der Informationsübertragung.

Abbildung 3.2(a) beschreibt einen lesenden Zugriff durch eine Aktivität Uncleared User auf ein Datenobjekt Secret File. Sofern die die Aktivität Uncleared User ausführende Partei nicht berechtigt ist, auf die in Secret File enthaltenen Informationen zuzugreifen, liegt hier eine Verletzung der Isolationsanforderung vor, die durch einen Datenfluss verursacht wird.

Abbildung 3.2(b) stellt eine kausale Abhängigkeit zwischen den beiden Prozess-Aktivitäten Secret User und Uncleared User dar. Der Implikationspfeil „⇒“ soll bedeuten, dass die Ausführung der Aktivität Secret User eine Voraussetzung für die Aktivität Uncleared User ist. Sofern die Ausführung von Secret User eine Information darstellt, die die Uncleared User ausführende Partei nicht erhalten darf, liegt hier eine Isolationsverletzung vor, die durch einen verdeckten Kanal zustande kommt: Durch die Abhängigkeit zwischen beiden Aktivitäten weiß die Partei, die Uncleared User ausführt, ob die (geheime) Aktivität Secret User ausgeführt worden ist.

InDico kann verdeckte Informationsübertragung erkennen, die durch Abhängigkeiten bei der Ausführung bzw. Nicht-Ausführung von Prozessaktivitäten induziert werden, d.h. durch den Kontrollfluss eines Geschäftsprozesses. Darüber hinaus existieren weitere Möglichkeiten der verdeckten Informationsübertragung, wie z.B. die statistische Ausführungsdauer von Aktivitäten und die relative Häufigkeit ihres Auftretens (Sabelfeld und Myers, 2003). Diese sind allerdings in der Regel kein Bestandteil statischer Prozessmodelle, wie sie von InDico benutzt werden. Entsprechende Erweiterungen wären denkbar, wenn derartige Daten in das Prozess-Modell (etwa auf der Grundlage einer forensischen Rekonstruktion) integriert werden könnten (vgl. die Diskussion in Kapitel 6).

### 3.1.3 Sicherheitsmodell

In InDico sind sowohl Ressourcen (wie etwa Dokumente, Nachrichten, Dateien) als auch Prozess-Aktivitäten potentielle Träger von Informationen. Dementsprechend werden in einem IFnet-Modell den Transitionen (die für Aktivitäten stehen) und den Marken (die Ressourcen repräsentieren) Sicherheitsstufen zugeordnet. Eine Sicherheitsstufe bezeichnet die relative Schutzwürdigkeit der Information, die erlangt wird, wenn die (Nicht-)Ausführung einer Aktivität beobachtet bzw. wenn auf eine Ressource zugegriffen werden kann.

Die Menge der Sicherheitsstufen wird in InDico mit $\mathcal{SC}$ bezeichnet und umfasst die beiden Stufen `High` und `Low`, wobei `Low` $<$ `High`. Die Verwendung von lediglich zwei Sicherheitsstufen stellt keine Beschränkung der Allgemeinheit dar, weil Spezifikationen mit weiteren Abstufungen sukzessive geprüft werden können, indem in jedem Durchgang ein Stufenpaar auf `High` und `Low` abgebildet wird.

Elemente aus der Menge $\mathcal{U}$ der Subjekte bezeichnen die in einer Geschäftsprozess-Konstellation involvierten Parteien, in deren Namen (Teil-)Prozesse ausgeführt werden. Dementsprechend ist jeder Prozess-Aktivität ein Subjekt zugeordnet. Jedem Subjekt wiederum ist eine Freigabe zugeordnet, die festlegt, Informationen welcher Sicherheitsstufe ein Subjekt erhalten darf. Ein Subjekt mit der Freigabe `High` darf alle Informationen erhalten, während ein Subjekt mit der Freigabe `Low` nur als `Low` klassifizierte Informationen erreichen dürfen.

Die Sicherheitsstufen von Prozessaktivitäten werden zu Beginn festgelegt und ändern sich während der Ausführung nicht. Die Klassifikation von Ressourcen kann ebenfalls zu Beginn festgelegt werden. Alternativ wird diese während der Ausführung dynamisch bestimmt. Dabei erhält eine unklassifizierte Marke eine der Freigabe des Subjektes entsprechende Stufe, das zuerst schreibend auf die Marke zugreift. Das bedeutet, dass die Marke die Einstufung `High` erhält, wenn der erste Schreibzugriff durch eine Aktivität stattfindet, die im Namen eines Subjektes mit der Freigabe `High` ausgeführt wird. Erfolgt der erste Schreibzugriff durch eine Aktivität, die im Namen eines Subjektes mit der Freigabe `Low` ausgeführt wird, erhält die Marke die Einstufung `Low`. Dasselbe gilt für Marken, die erst während der Ausführung erzeugt werden.

Eine Besonderheit von InDico ist die Einführung einer speziellen Freigabe `neutral`. Diese wird Subjekten zugeordnet, die „neutral“ sind und selbst keine Sicherheitsinteressen verfolgen. Eine solche Klassifikation wird als sinnvoll erachtet, um das Verhalten vermittelnder Parteien zu modellieren, die von den übrigen Parteien als vertrauenswürdige Instanz anerkannt werden („*Trusted Third Party*“). Im Kontext der Geschäftsprozess-Ausführung könnte dies etwa der Betreiber einer Dienstplattform sein, der – z.B. in einer mandantenfähigen Architektur – einen Dienst-Prozess betreibt, welcher gleichzeitig mit mehreren Klienten interagiert. Subjekte mit der Freigabe `neutral` dürfen jegliche Informationen erhalten. Im Gegensatz zu Subjekten der Freigabe `High` werden neu erzeugte Ressourcen, bzw. noch nicht eingestufte Ressourcen, auf die schreibend zugegriffen wird, nicht als `High` klassifiziert, sondern verbleiben ohne Einstufung.

Eine Ausnahme besteht hier, wenn gleichzeitig klassifizierte Ressourcen verarbeitet werden. Hier erhält ein neu erzeugtes Objekt die Einstufung, die der höchsten aller Eingabeobjekte entspricht. Liest eine Aktivität, die im Namen eines `neutral`-Subjekts ausgeführt wird, etwa eine `High`-Ressource ein und erzeugt als Ausgabe eine neue Ressource, so erhält auch diese die Einstufung `High`. Der Grund dafür ist, dass die Möglichkeit besteht, dass Informationen aus der `High`-Ressource in die neu erzeugte Ressource übertragen worden sind.

## 3.2 Prozess-Modellierung mit IFnet

Das Ziel der Prozess-Modellierung mit IFnet ist die formal eindeutige Beschreibung der Möglichkeiten der Informationsübertragung in einer Geschäftsprozess-Konfiguration. Die Grundlage für IFnet bilden Petri-Netze und insbesondere *Workflow Nets* (WF-Nets), die sich für die strukturelle Modellierung von Geschäftsprozessen (d.h. für die Darstellung der möglichen Abfolgen ihrer Aktivitäten) bewährt haben (van der Aalst, 1995). Genau wie in WF-Nets wird eine Aktivität in IFnet durch eine einzelne Transition modelliert und die Bedingungen für die Ausführung einer Aktivität werden durch die Markierung von Eingangsstellen beschrieben. Dies ermöglicht eine kompakte Darstellung eines Prozesses, die Flussdiagrammen ähnelt und intuitiv leicht zu erfassen ist.

Darüber hinaus verfügt IFnet über mehrere Konstrukte, die die in Abschnitt 2.2.4 identifizierten notwendigen Erweiterungen realisieren. Diese sind im Einzelnen:

- **Datenfluss.** Datenobjekte werden in IFnet durch unterscheidbare Marken dargestellt, die von Transitionen (d.h. Aktivitäten) eingelesen und ausgegeben werden können. Die Ein- und Ausgabemenge einer Aktivität ist in einem IFnet nicht zwangsläufig fix, sondern kann in Abhängigkeit des Zustandes des Netzes variieren. Indem der Fluss von Datenobjekten in IFnet explizit durch Marken modelliert wird, können Zugriffskonflikte dargestellt werden, die eine Möglichkeit zur verdeckten Informationsübertragung darstellen.

- **Attributierte Transitionen.** Eigenschaften von Geschäftsprozess-Aktivitäten, die für die Modellierung bzw. für die Sicherheitsanalyse wichtig sind, werden in einem IFnet als Attribute von Transitionen modelliert. Diese umfassen Subjektbezeichner, die die Partei kennzeichnen, in deren Namen eine Aktivität ausgeführt wird, Zugriffsmodi bzgl. der verarbeiteten Datenobjekte und Bedingungen, die angeben, von welchen Datenobjekten die Ausführung einer Aktivität abhängt.

- **Sicherheitsstufen.** In IFnet werden sowohl Marken als auch Transitionen als mögliche Informationsträger mit Sicherheitsstufen versehen. Diese kennzeichnen den relativen Schutzanspruch der enthaltenen Informationen. Die Zuordnung einer Freigabe zu einem Subjektbezeichner kennzeichnet analog dazu die Berechtigung von Subjekten, Informationen zu erhalten.

Die Darstellung der Prozess-Modellierung mit IFnet ist dreigeteilt. Zunächst wird ein spezieller Petri-Netz-Typ – das *Distinguishable Token Net* (DTN) – eingeführt, der die Grundlage von IFnet bildet und seine Ausführungssemantik definiert. Anschließend wird die Erweiterung von DTNs zu IFnets gezeigt, die das grundlegende Modell um Konstrukte ergänzt, die für die Geschäftsprozess-Darstellung und Sicherheitsanalysen notwendig sind. Der dritte Teil stellt die Abbildung von BPMN auf IFnet vor, die ein wichtiger Baustein ist, um den InDico-Ansatz zu automatisieren.

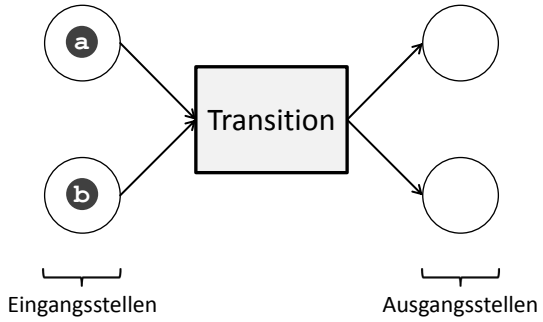



Abbildung 3.3.: Schematische Darstellung einer Transition in einem DTN.

### 3.2.1 Petri-Netze mit unterscheidbaren Marken

Die Grundlage für IFnets bilden spezielle Petri-Netze, die für die Verarbeitung unterscheidbarer Marken generalisiert und als *Distinguishable Token Nets* (DTNs) bezeichnet werden. In einem DTN verfügt jede Marke über eine Kennung, durch die sie identifiziert wird. Der Terminologie in der Literatur zu Gefärbten Petri-Netzen (*Colored Petri Nets* – CPN) folgend (Jensen, 1996), wird diese Kennung auch als die Farbe (*Color*) einer Marke bezeichnet. Die Menge $\mathcal{C}$ ist das Universum aller Marken.

Wie bei Petri-Netzen sind die Kernbestandteile eines DTN Transitionen, Stellen und gerichtete Kanten, die jeweils durch Rechtecke, Kreise und Pfeile dargestellt werden. Die Marken in einem DTN werden durch Punkte innerhalb von Stellen dargestellt, die – im Gegensatz zu einem traditionellen Petri-Netz – zusätzlich mit ihrer Kennung (bzw. Farbe) markiert sind. Eine Zustandsänderung wird in einem DTN durch das „Schalten“ (oder „Feuern“) von Transitionen ausgelöst: In diesem Fall entnimmt (bzw. „konsumiert“) eine Transition Marken von ihren Eingangsstellen und legt (bzw. „produziert“) Marken in ihre Ausgangsstellen.

Abbildung 3.3 zeigt schematisch eine Transition mit zwei Eingangsstellen (linke Seite) und zwei Ausgangsstellen (rechte Seite). Die beiden Eingangsstellen der Transition sind mit jeweils einer Marke belegt, die die Kennungen `a` bzw. `b` haben. Das Schalten einer Transition in einem DTN ist ein atomarer Vorgang, der spontan ausgelöst wird, wenn in den Eingangsstellen einer Transition eine spezifizierte Kombination von Marken vorhanden ist. In diesem Fall wird aus jeder Eingangsstelle die angegebene Menge von Marken entnommen. Übertragen auf einen Geschäftsprozess wird mit dem Schaltvorgang die Ausführung einer Aktivität ausgelöst und werden die für die Ausführung benötigten Ressourcen eingelesen.

Anschließend wird in jeder Ausgangsstelle die angegebene Menge von Marken platziert. Dies entspricht dem Signal, dass die entsprechende Prozess-Aktivität geendet hat und der Ausgabe der bearbeiteten Datenobjekte. Es muss nicht zwangsläufig dieselbe Mengen von Marken ausgegeben werden, die eingelesen worden ist, da während der Ausführung einer Aktivität beispielsweise auch Datenobjekte neu erzeugt oder gelöscht werden können.

Im Folgenden werden DTNs und dazugehörige Konzepte formal definiert. Als Hilfsmittel werden zunächst Notationen für Multimengen und Reihungen definiert, die in den darauf folgenden Teilen verwendet werden.

**Definition 3.1 (Multimenge).** *Eine Multimenge*[2] *ist eine spezielle Menge deren Elemente mehrfach vorkommen können. Für eine Menge $S$ bezeichnet $S^+$ die Menge aller Multimengen mit Elementen aus $S$. Die Summe zweier Multimengen $(A + B)$, ihre Differenz $(A - B)$ und der Begriff einer Unter-Multimenge $(A \leq B)$ werden in natürlicher Weise definiert.* ⊣

Eine Reihung legt für die Elemente einer (Multi-)Menge eine feste Abfolge fest. Auf die Elemente kann über einen Index zugegriffen werden, der bei 0 beginnt.

**Definition 3.2 (Reihung).** *Eine Reihung ist eine festgelegte, endliche Anordnung von Elementen einer (Multi-)Menge. Für eine (Multi-)Menge $S$ bezeichnet $S[]$ die Menge aller Reihungen mit Elementen aus $S$. Die Größe einer Reihung $A$ ist die Anzahl der darin enthaltenen Elemente und wird als $|A|$ notiert. Auf die Elemente einer Reihung $A$ wird über einen Index zugegriffen, der die Werte $\{0, ..., |A| - 1\}$ annehmen kann. Der Ausdruck $A[i]$ liefert das Element mit Index $i$ zurück. Die Verkettung von Reihungen $(A \cup B)$ und ihre Differenz $(A \setminus B)$ werden durch die üblichen Mengen-Operatoren notiert.* ⊣

In der folgenden Definition wird ein DTN formal charakterisiert.

**Definition 3.3 (Distinguishable Token Net).** *Ein DTN ist ein Tupel $N = (P, T, F, \text{In}, \text{Out})$, wobei $P$ eine endliche Menge von Stellen und $T$ eine endliche Menge von Transitionen ist für die $P \cap T = \emptyset$ gilt. $F \subseteq (P \times T) \cup (T \times P)$ ist eine Menge gerichteter Kanten, die „Flussrelation" (Flow Relation) heißt. Für zwei Elemente $x, y \in (P \cup T)$ bedeutet die Notation $xFy$, dass es eine Kante von $x$ nach $y$ gibt. Die Abbildungen* In *und* Out *definieren die von Transitionen eingelesenen und ausgegebenen Mengen von Marken:*

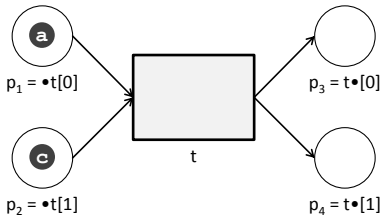
- *Die Funktion* In *definiert für jede Transition $t$ und für jede Stelle $i$, für die $iFt$ gilt (d.h. für jede Eingangsstelle von $t$) die Reihung aus Marken-Kombinationen, die erwartet werden. Es muss von jeder Eingangsstelle mindestens eine Marke eingelesen werden:*

  $$\text{In} : T \times P \to (\mathcal{C}^+ \setminus \emptyset)[].$$

- *Die Funktion* Out *definiert für jede Transition $t$ und für jede Stelle $o$, für die $tFo$ gilt (d.h. für jede Ausgangsstelle von $t$) die Reihung aus Marken-Kombinationen, die ausgegeben werden können. Es muss in jeder Ausgangsstelle mindestens eine Marke erzeugt werden:*

  $$\text{Out} : T \times P \to (\mathcal{C}^+ \setminus \emptyset)[].$$ ⊣

[2] Eine Multimenge wird im Englischen als *Multiset* oder *Bag* bezeichnet.




$\textsc{In}(t, p_1) = [\{\mathtt{a}\}, \{\mathtt{b}\}]$
$\textsc{In}(t, p_2) = [\{\mathtt{c}\}, \{\mathtt{d}\}]$
$\textsc{Out}(t, p_3) = [\{\mathtt{a}\}, \{\mathtt{c}\}]$
$\textsc{Out}(t, p_4) = [\{\mathtt{b}\}, \{\mathtt{d}\}]$

Abbildung 3.4.: Ausschnitt aus einem markierten DTN.

In einem DTN haben – für eine feste Transition $t$ – die durch die Abbildung In und Out definierten Reihungen stets dieselbe Größe, die die Anzahl der Schaltmöglichkeiten von $t$ angibt.

Beispiel: Abbildung 3.4 zeigt beispielhaft eine Transition $t$ mit jeweils zwei Eingangs- und Ausgangsstellen. Durch die Abbildungen In und Out werden zwei Schaltmöglichkeiten definiert:

1. $t$ liest die Marke a aus der Eingangsstelle $\mathtt{p_1}$ und die Marke c aus der Eingangsstelle $\mathtt{p_2}$ aus. Als Ausgabe produziert $t$ in der Stelle $\mathtt{p_3}$ die Marke a und in der Stelle $\mathtt{p_4}$ die Marke b.
2. $t$ liest die Marke b aus der Eingangsstelle $\mathtt{p_1}$ und die Marke d aus der Eingangsstelle $\mathtt{p_2}$ aus. Als Ausgabe produziert $t$ in der Stelle $\mathtt{p_3}$ die Marke c und in der Stelle $\mathtt{p_4}$ die Marke d.

Die Aufzählung der Elemente einer Reihung wird dabei durch eckige Klammern eingeschlossen. In dem gezeigten Zustand hat die $t$ nur die erste Schaltmöglichkeit. Für die zweite Möglichkeit fehlen die entsprechenden Marken in den Eingangsstellen.

In dem vorhergehenden Beispiel wird für die Bezeichnung der Ein- und Ausgangsstellen von $t$ eine abkürzende $\bullet$-Notation verwendet, die in der folgenden Definition festgelegt wird.

**Definition 3.4 (Knoten).** *Für ein DTN $N$ heißen die Elemente der Menge $P \cup T$ die Knoten von $N$. Ein Knoten $x$ ist ein Eingangsknoten eines anderen Knotens $y$, genau dann, wenn es eine gerichtete Kante von $x$ nach $y$ gibt (d.h. $xFy$). Ein Knoten $x$ ist ein Ausgangsknoten eines anderen Knotens $y$, genau dann, wenn es eine gerichtete Kante von $y$ nach $x$ gibt (d.h. $yFx$). Für jeden Knoten $x \in P \cup T$ definieren $\bullet x \in \{y \mid yFx\}\,[]$ sowie $x\bullet \in \{y \mid xFy\}\,[]$ Reihungen der Eingangs- bzw. der Ausgangsknoten.* ⊣

Um das Verhalten eines DTN zu beschreiben, ist die Belegung seiner Stellen mit Marken entscheidend. Eine solche Belegung mit Marken wird als Markierung bezeichnet und charakterisiert einen Zustand des DTN.

**Definition 3.5 (Markierung).** *Die Markierung eines DTN $N$ bezeichnet die Belegung seiner Stellen mit Marken. Eine Markierung $M$ ist eine Abbildung:*

$$M : P \to \mathcal{C}^+$$

*Die Multimenge, die alle im Zustand $M$ in $N$ enthaltenen Marken enthält, wird durch $M^*$ bezeichnet. $(N, M)$ bezeichnet das DTN $N$ im Zustand $M$. $(N, M)$ ist ein markiertes DTN. Die Menge aller Markierungen von $N$ wird mit $\mathcal{M}^N$ bezeichnet.* ⊣

Anzahl und Verteilung der Marken in einem DTN können sich während der Ausführung ändern. Die Transitionen sind die aktiven Bestandteile des DTN und lösen Zustandsübergänge gemäß einer Schaltregel aus. Anders als in einem klassischen Petri-Netz kann eine Transition in einem DTN bei mehreren Schaltvorgängen unterschiedliche Marken-Kombinationen ein- und ausgeben. Die verschiedenen Möglichkeiten einer Transition, Marken zu konsumieren bzw. zu erzeugen, werden durch die Funktionen In bzw. Out charakterisiert.

Die *Eingabemenge* fasst für eine Schaltmöglichkeit alle Marken zusammen, die aus sämtlichen Eingangsstellen zusammengenommen erwartet werden.

**Definition 3.6 (Eingabemenge).** *In einem DTN $N$ liefert die Funktion* InputSet : $T \times \mathbb{N} \to \mathcal{C}^+$ *die Multimenge der Marken zurück, die eine Transition $t$ für die Schaltmöglichkeit mit dem Index $i \in \mathbb{N}$ erwartet:*

$$\textsc{InputSet}(t, i) = \sum_{j \in \{0,..,|\bullet t|-1\}} \textsc{In}(t, \bullet t[j])[i].$$

⊣

Beispiel: Die Transition $t$ aus dem in Abbildung 3.4 gezeigten Beispiel hat die folgenden Eingabemengen:

$$\textsc{InputSet}(t, 0) = \{\mathsf{a}, \mathsf{c}\}$$
$$\textsc{InputSet}(t, 1) = \{\mathsf{b}, \mathsf{d}\}.$$

Analog zur Eingabeseite einer Transition wird auch für ihre Ausgabeseite ein entsprechendes Konstrukt definiert. Die *Ausgabemenge* fasst für eine Schaltmöglichkeit alle Marken zusammen, die in sämtlichen Ausgangsstellen zusammengenommen produziert werden.

**Definition 3.7 (Ausgabemenge).** *In einem DTN $N$ liefert die Funktion* OutputSet : $T \times \mathbb{N} \to \mathcal{C}^+$ *die Multimenge der Marken zurück, die eine Transition $t$ für die Schaltmöglichkeit mit dem Index $i \in \mathbb{N}$ produziert:*

$$\textsc{OutputSet}(t, i) = \sum_{j \in \{0,..,|t\bullet|-1\}} \textsc{Out}(t, t \bullet [j])[i].$$

⊣

Beispiel: Die Transition $t$ aus dem in Abbildung 3.4 gezeigten Beispiel hat die folgenden Ausgabemengen:

$$\textsc{OutputSet}(t, 0) = \{\mathsf{a}, \mathsf{b}\}$$
$$\textsc{OutputSet}(t, 1) = \{\mathsf{c}, \mathsf{d}\}.$$

Damit eine Transition in einem Zustand schalten kann, müssen ihre Eingangsstellen die erforderliche Markenmenge in der richtigen Kombination bereitstellen. Die folgende Definition gibt die Bedingung für das Schalten einer Transition formal an.

**Definition 3.8 (Schaltbereite Transition).** *In einem markierten DTN $(N, M)$ ist eine Transition $t$ schaltbereit, genau dann, wenn gilt:*

$$\exists i \in \mathbb{N} : \forall j \in \{0, .., |\bullet t| - 1\} : \textsc{In}(t, \bullet t[j])[i] \leq M(\bullet t[j]).$$

*Existiert eine solche Schaltmöglichkeit $i$, ist die Transition $t$ „mit Schaltmöglichkeit $i$ schaltbereit".* ⊣

Es kann in einem Zustand für eine Transition mehrere Schaltmöglichkeiten geben, mit denen die Transition schaltbereit ist. Im Folgenden wird für einen Zustand und eine Transition die Menge dieser *aktivierbaren* Schaltmöglichkeiten definiert.

**Definition 3.9 (Menge der aktivierbaren Schaltmöglichkeiten).** *In einem markierten DTN $(N, M)$ bezeichnet die Menge $CV_t^M$ die Menge der Schaltmöglichkeiten, mit denen die Transition $t$ in Zustand $M$ schaltbereit ist:*

$$CV_t^M = \{i \in \mathbb{N} \mid t \text{ ist in } M \text{ mit Schaltmöglichkeit } i \text{ schaltbereit}\}.$$

⊣

Beispiel: In dem Beispiel aus Abbildung 3.4 ist die Transition mit der Schaltmöglichkeit 0 schaltbereit. Für den dargestellten Zustand ist diese die einzige Schaltmöglichkeit, mit der $t$ schaltbereit ist.

Eine schaltbereite Transition kann spontan schalten. Beim Schaltvorgang wird eine aktivierbare Schaltmöglichkeit gewählt. Der neue Zustand wird erreicht, nachdem die durch die Schaltmöglichkeit charakterisierten Markenmengen von den Eingangsstellen genommen und in den Ausgangsstellen platziert worden sind.

**Definition 3.10 (Schaltregel).** *In einem markierten DTN $(N, M)$ sei $t$ eine schaltbereite Transition und $i \in CV_t^M$. Das Schalten von $t$ mit $i$ bewirkt den Übergang in einen Zustand $M'$, der wie folgt definiert ist:*

- $\forall \bullet t[j], j \in \{0, .., |\bullet t| - 1\} : M'(\bullet t[j]) = M(\bullet t[j]) - \textsc{In}(t, \bullet t[j])[i]$.
- $\forall t \bullet [j], j \in \{0, .., |t \bullet| - 1\} : M'(t \bullet [j]) = M(t \bullet [j]) + \textsc{Out}(t, t \bullet [j])[i]$.
- $\forall p \in P \setminus (\bullet t + t\bullet) : M'(p) = M(p)$. ⊣

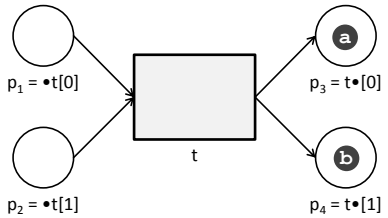



Abbildung 3.5.: Zustand des DTN aus Abbildung 3.4 nach dem Schaltvorgang.

Beispiel: Schaltet die in Abbildung 3.4 dargestellte Transition mit der (einzig aktivierbaren) Schaltmöglichkeit 0, erreicht sie den in Abbildung 3.5 dargestellten Zustand. In diesem Fall ist die Marke `a` aus der Eingangsstelle $p_1$ in die Ausgangsstelle $p_3$ weitergereicht worden, während in Ausgangsstelle $p_4$ die neue Marke `b` produziert worden ist.

### 3.2.2 IFnet

Die Definition von *Information Flow Nets* (IFnets) greift die Definition von DTN auf und erweitert sie um Konstrukte, die für die Modellierung von Geschäftsprozessen und für deren Sicherheitsanalyse benötigt werden. In einem IFnet werden Prozess-Aktivitäten durch Transitionen und Ressourcen (wie beispielsweise Dokumente, Nachrichten und Variable) durch Marken modelliert.

Marken mit der Kennung `black` („schwarze Marken“) haben dabei einen besonderen Status: Sie stehen nicht für Ressourcen im eigentlichen Sinne, sondern signalisieren ausschließlich den Zustand eines Netzes. Sie können eingesetzt werden, um das Auslösen und die Terminierung von Aktivitäten zu modellieren, wenn dabei keine Ressourcen eingelesen bzw. ausgegeben werden.

Schwarze Marken werden grafisch durch schwarze, ausgefüllte Punkte ohne Beschriftung dargestellt. Ihnen gegenüber stehen die „gefärbten Marken“, d.h. alle diejenigen Marken, die *nicht* schwarz sind. Die Menge aller gefärbten Marken wird mit $\mathcal{C}_c$ bezeichnet, wobei $\mathcal{C}_c = \mathcal{C} \setminus \{\texttt{black}\}$ gilt.

**Definition 3.11 (IFnet).** *Ein IFnet ist ein Tupel* $N = ((P, T, F, \text{In}, \text{Out}), \text{S}_\mathcal{U}, \text{A}, \text{G}, \text{Label}_{\mathcal{SC}}, \text{Clear}_{\mathcal{SC}})$, *wobei* $(P, T, F, \text{In}, \text{Out})$ *ein DTN ist und die weiteren Elemente wie folgt definiert sind:*

- *Die Funktion* $\text{S}_\mathcal{U}$ *ordnet Transitionen Subjekte aus der Menge* $\mathcal{U}$ *zu:*

  $$\text{S}_\mathcal{U} : T \to \mathcal{U}.$$

- *Die Funktion* A *definiert den Modus, in dem eine Transition* $t$ *auf ein Ressource zugreift (lesend, schreibend oder beides):*

  $$\text{A} : T \times \mathcal{C}_c \to \mathcal{P}(\{read, write\}).$$

- *Die Funktion* G *ordnet Transitionen Prädikate zu, die aus einem Prädikatsbezeichner aus der Menge $P_C$ und einer Menge gefärbter Marken bestehen. Ein Prädikat modelliert eine Ausführungbedingung, deren Auswertung von den bezeichneten Ressourcen abhängt:*[3]

  $$\mathrm{G} : T \to P_C \times \mathcal{P}(C_c).$$

- *Die Funktion* $\textsc{Label}_{\mathcal{SC}}$ *ordnet Transitionen und gefärbten Marken eine Sicherheitsstufe aus der Menge $\mathcal{SC}$ zu. Eine Marke kann außerdem als unklassifiziert (*`unlabeled`*) eingestuft werden:*

  $$\textsc{Label}_{\mathcal{SC}} : T \cup C_c \to \mathcal{SC} \cup \{\texttt{unlabeled}\}.$$

- *Die Funktion* $\textsc{Clear}_{\mathcal{SC}}$ *ordnet Subjekten Freigaben zu. Eine Freigabe entspricht entweder einer Sicherheitsstufe oder* `neutral`*:*

  $$\textsc{Clear}_{\mathcal{SC}} : \mathcal{U} \to \mathcal{SC} \cup \{\texttt{neutral}\}.$$

*Darüber hinaus muss ein IFnet die folgenden Bedingungen erfüllen:*

- *Anfangsstellen: Es gibt eine nicht-leere Menge $P_I \subset P$ von Stellen mit $\bullet i = \emptyset, i \in P_I$. $P_I$ ist die Menge der Anfangsstellen von $N$.*
- *Verbundenheit: Für jeden Knoten $x \in P \cup T$ existiert eine Anfangsstelle $i \in P_I$, so dass es einen Pfad von $i$ nach $x$ gibt, d.h. es gibt eine Folge $(k_0, k_1, ..., k_n)$ von Knoten, in der $k_0 = i$ und $k_n = x$ ist und $k_r F k_{r+1}$ für alle $r \in \{0, ..., n-1\}$ gilt.*

$\dashv$

Die letzten beiden Bedingungen bewirken, dass ein IFnet definierte Startpunkte hat und dass es keine losgelösten Transitionen gibt, d.h. Aktivitäten, die nicht zum Prozessablauf beitragen. Des weiteren werden die folgenden Bezeichnungen verwendet:

- *Anfangszustand.* Eine Anfangszustand ist eine Markierung, in der jede Anfangsstelle mindestens eine schwarze Marke enthält. Die Menge aller Anfangszustände eines IFnet $N$ wird durch $\mathcal{M}_I^N$ bezeichnet.
- *Markiertes IFnet.* Ein markiertes IFnet ist ein Paar $(N, M) = (((P, T, F, \textsc{In}, \textsc{Out}), \mathrm{A}, \mathrm{G}, \textsc{Label}_{\mathcal{SC}}, \textsc{Clear}_{\mathcal{SC}}), M)$, in dem $((P, T, F, \textsc{In}, \textsc{Out}), M)$ ein markiertes DTN mit Markierung $M$ ist. Die Menge aller markierten IFnets wird durch $\mathcal{N}$ bezeichnet.

In einem markierten IFnet $(N, M)$ werden die folgenden Notationen verwendet:

- $M \xrightarrow{t} M'$: Die Transition $t$ ist im Zustand $M$ schaltbereit; ihr Schalten in diesem Zustand überführt das Netz in den Zustand $M'$.

[3] Die genaue Semantik eines Prädikats ist im Kontext von IFnets unerheblich, da es lediglich dazu dient, eine Abhängigkeit der Ausführung von bestimmten Marken zu erfassen. Ein Prädikat wird beispielsweise durch `p(a,b)` notiert, wobei `p` der Prädikatsbezeichner ist und in Klammern die gefärbten Marken aufgeführt sind (hier `a` und `b`), die für die Auswertung benötigt werden.

- $M \xrightarrow{t} *$: Die Transition $t$ ist im Zustand $M$ schaltbereit.
- $M \xrightarrow{t_i} M'$: Die Transition $t$ ist im Zustand $M$ mit der Schaltmöglichkeit $i \in CV_t^M$ schaltbereit; ihr Schalten in diesem Zustand mit $i$ überführt das Netz in den Zustand $M'$.
- $M \longrightarrow M'$: Es gibt in $M$ eine schaltbereite Transition $t$, so dass $M \xrightarrow{t} M'$ gilt.
- $M \xrightarrow{\sigma} M_n$: Die Schaltfolge $\sigma = t_1 t_2 t_3 ... t_{n-1}$ führt von Zustand $M$ über eine (möglicherweise leere) Menge von Zwischenzuständen $M_2, ..., M_{n-1}$ zu Zustand $M_n$, d.h. $M \xrightarrow{t_1} M_2 \xrightarrow{t_2} ... \xrightarrow{t_{n-1}} M_n$.

**Definition 3.12 (Erreichbarkeit).** *In einem markierten IFnet $(N, M)$ ist ein Zustand $M'$ erreichbar (notiert als $M \xrightarrow{*} M'$), genau dann, wenn es eine Schaltfolge $\sigma$ gibt, so dass $M \xrightarrow{\sigma} M'$ gilt. Dabei ist auch die leere Schaltfolge erlaubt, d.h. $M \xrightarrow{*} M$. Die Menge von Zuständen, die von $M$ aus erreichbar ist, wird als $[M]$ notiert. Die Menge aller von einem Anfangszustand aus erreichbaren Zustände wird als $\left[\mathcal{M}_I^N\right]$ notiert.* ⊣

Die Definition von IFnets berücksichtigt ausschließlich strukturelle Aspekte, stellt aber keine Bedingungen an das dynamische Verhalten. Daher kann ein IFnet Eigenschaften aufweisen, die in einem Geschäftsprozess normalerweise nicht erwünscht sind. Dies betrifft insbesondere die Nicht-Ausführbarkeit von Aktivitäten. Die folgende Korrektheitsbedingung an IFnets schließt daher das Vorhandensein nicht-erreichbarer („toter“) Transitionen aus.

**Definition 3.13 (Korrektheit).** *Ein IFnet $N$ mit Anfangszustand $M$ ist genau dann korrekt, wenn jede Transition von $M$ aus erreichbar ist:*

$$\forall t \in T \; \exists M_1, M_2 \in \mathcal{M}^N : M \xrightarrow{*} M_1 \xrightarrow{t} M_2.$$ ⊣

Durch die Möglichkeit, in einem IFnet alternative Kombinationen von Ein- und Ausgabemarken für eine Transition anzugeben, kann es zu einem Indeterminismus bei der Ausführung von Transitionen kommen, wenn ein Zustand einer Transition mehrere Möglichkeiten zur Ein- oder Ausgabe lässt. Liegt ein solcher Indeterminismus für einen Anfangszustand *nicht* vor, ist das Netz „T-deterministisch“.

**Definition 3.14 (T-Determinismus).** *Ein IFnet $N$ mit Anfangszustand $M$ ist T-deterministisch, genau dann, wenn jede Transition in jedem von $M$ erreichbaren Zustand über maximal eine Schaltmöglichkeit verfügt:*

$$\forall t \in T \; \forall M' \in [M] : \left| CV_t^{M'} \right| \leq 1.$$ ⊣

Abschließend wird die Klassifizierung von gefärbten Marken definiert, die im Startzustand die Auszeichnung `unlabeled` besitzen. Gemäß des Sicherheitsmodells von InDico erhalten diese eine Auszeichnung, die von der Freigabe der Transition abhängt, die die Marke produziert bzw. zum ersten Mal schreibend darauf zugreift (vgl. die Beschreibung in Abschnitt 3.1.3).

**Definition 3.15 (Dynamische Klassifikation von `unlabeled`-Marken).** *Sei* $(N, M)$ *ein markiertes IFnet mit Anfangszustand* $M$*. Bei einem Schaltvorgang* $M_1 \xrightarrow{t_i} M_2$ *mit* $M_1, M_2 \in [M]$ *wird die Auszeichnung jeder gefärbten Marke* $c \in \textsc{OutputSet}(t, i)$ *mit* $\textsc{Label}_{\mathcal{SC}}(c) = \texttt{unlabeled}$ *in den folgenden Fällen aktualisiert:*

- *Falls* $\textsc{Clear}_{\mathcal{SC}}(\mathrm{S}_{\mathcal{U}}(t)) \neq \texttt{neutral}$ *und* $c \notin \textsc{InputSet}(t, i)$ *gilt* $\textsc{Label}_{\mathcal{SC}}(t) = \textsc{Clear}_{\mathcal{SC}}(\mathrm{S}_{\mathcal{U}}(t))$.
- *Falls* $\textsc{Clear}_{\mathcal{SC}}(\mathrm{S}_{\mathcal{U}}(t)) \neq \texttt{neutral}$*,* $c \in \textsc{InputSet}(t, i)$ *und* $\{write\} \subseteq \mathrm{A}(t, c)$ *gilt* $\textsc{Label}_{\mathcal{SC}}(t) = \textsc{Clear}_{\mathcal{SC}}(\mathrm{S}_{\mathcal{U}}(t))$.
- *Falls* $\textsc{Clear}_{\mathcal{SC}}(\mathrm{S}_{\mathcal{U}}(t)) = \texttt{neutral}$*,* $c \notin \textsc{InputSet}(t, i)$ *oder* $\{write\} \subseteq \mathrm{A}(t, c)$*,* $\exists d \in \textsc{InputSet}(t, i) : \textsc{Label}_{\mathcal{SC}}(d) = \texttt{High}$ *und* $\{read\} \subseteq \mathrm{A}(t, d)$ *gilt* $\textsc{Label}_{\mathcal{SC}}(t) = \texttt{High}$.

*Andernfalls bleibt die Klassifikation der Marke bei diesem Schaltvorgang unverändert.* ⊣

## 3.3 Abbildung von BPMN auf IFnet

Damit bestehende Prozessmodelle mit dem InDico-Vorgehensmodell untersucht werden können, wird in diesem Abschnitt eine Abbildung der *Business Process Modeling Notation* (BPMN) auf IFnet angegeben. BPMN ist eine moderne Modellierungssprache für Geschäftsprozesse mit grafischer Notation, deren Schwerpunkt die Spezifikation des Kontrollflusses ist (Dijkman u. a., 2007). Wie in Abschnitt 2.2.1 diskutiert, hat BPMN keine formale Semantik. Durch die Transformation in IFnet wird eine formale Semantik für die abgebildeten Elemente von BPMN angegeben.

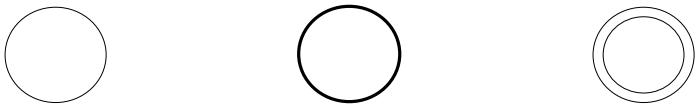
Die folgende Abbildung zeigt, wie der Kontroll- und Datenfluss eines BPMN-Prozesses durch ein IFnet-Modell abgebildet werden kann. Die Übersetzung des Ausnahmeverhaltens (*Exception Flow*) bleibt hier unberücksichtigt. Auf die Transformation von strukturierenden Elementen (wie beispielsweise *Swimlanes* und *Groups*) und Abkürzungen (wie z.B. spezielle Elemente für Schleifen) wird ebenfalls verzichtet, da diese keine Auswirkung auf das Prozessverhalten haben bzw. mit den grundlegenden Elementen dargestellt werden können. Die Grundlage für die Abbildung bildet die BPMN-Version 1.2 (OMG, 2009).

Mit der BPMN-Abbildung soll exemplarisch gezeigt werden, dass IFnet geeignet ist, um in der Praxis eingesetzte Prozesse modellieren zu können. BPMN eignet sich zu diesem Zweck insofern gut, als es im Vergleich zu anderen Prozess-Modellierungssprachen relativ mächtig ist (Russell u. a., 2006). Daher kann davon ausgegangen werden, dass auch die Übersetzung anderer Sprachen in IFnet problemlos ist.

### Übersetzungsvorgang

Die Abbildung von BPMN auf IFnet wird elementweise vorgenommen, d.h. für jedes BPMN-Element wird ein entsprechendes IFnet-Konstrukt eingesetzt.[4] Ein BPMN-Prozess wird übersetzt, indem sukzessive für jedes seiner Elemente das entsprechende IFnet-Konstrukt ausgewählt und – analog zur Struktur des BPMN-Prozesses – mit den bereits abgebildeten IFnet-Komponenten verbunden wird. Bei der Abbildung auf IFnet werden zunächst die *Flow Objects* (d.h. *Events*, *Activities* und *Gateways*) und die sie verbindenden *Sequence Flows* (d.h. der Kontrollfluss) übersetzt. Anschließend werden in dem resultierenden IFnet die Konstrukte zur Modellierung des BPMN-*Message Flow* (d.h. des Nachrichtenaustauschs) ergänzt und die Namen von BPMN-*Pools* auf Subjektbezeichner von IFnet-Transitionen abgebildet. In einem abschließenden Schritt wird das so entstandene IFnet gegebenenfalls manuell ergänzt.

[4] Die Idee zu einer elementweisen Übersetzung sowie der Ansatz zur Übersetzung einzelner BPMN-Konstrukte entstammt der Arbeit von Dijkman u. a. (2007), in der eine Übersetzung von BPMN in klassische Petri-Netze vorgestellt wird.



(a) *Start* (b) *End* (c) *Intermediate*

Abbildung 3.6.: *Event*-Typen in BPMN.

### Abbildung von *Events*

Ein *Event* (Ereignis) ist etwas, das während der Ausführung eines Geschäftsprozesses „passiert". *Events* beeinflussen in der Regel den Ablauf des Prozesses. Sie werden durch nicht-ausgefüllte Kreise dargestellt, die mit Symbolen versehen sein können, die verschiedene Spezial-*Events* kennzeichnen. Es gibt – abhängig von ihrer Position in einem Prozess – drei *Event*-Typen, die in Abbildung 3.6 dargestellt sind:

1. Das ***Start-Event***, das den Beginn eines Prozesses kennzeichnet und durch einen einfachen Kreis dargestellt wird (Abbildung 3.6(a)).
2. Das ***End-Event***, das das Ende eines Prozesses darstellt und durch einen Kreis mit einer dickeren Umrandung dargestellt wird (Abbildung 3.6(b)).
3. Das ***Intermediate-Event***, das ein Ereignis zwischen *Start-* und *End-Event* kennzeichnet und durch einen doppelten Kreis dargestellt wird (Abbildung 3.6(c)).

Ein *Event* wird in ein IFnet-Konstrukt übersetzt, das aus einer Transition besteht (deren Schalten das Auftreten des entsprechenden Ereignisses repräsentiert), die durch eine oder mehrere Stellen mit den angrenzenden Konstrukturen verknüpft ist. Dabei werden verschiedene Spezial-*Events* – es gibt beispielsweise *Message-*, *Timer-* und *Error-Events*, die jeweils durch eigene Symbole innerhalb des Kreises gekennzeichnet sind – in dasselbe IFnet-Konstrukt übersetzt. Tabelle 3.1 zeigt für jeden der drei *Event*-Typen von BPMN die Abbildung in das entsprechende IFnet-Konstrukt.

Knoten (d.h. Stellen oder Transitionen) in IFnet-Konstrukten, die durch gestrichelte Umrisse dargestellt sind, bedeuten, dass der entsprechende Knoten nicht notwendigerweise allein von dem jeweiligen Konstrukt verwendet wird, sondern auch Teil anderer Konstrukte sein kann. Neben der strukturellen Darstellung des IFnet-Konstrukts ist das Ein- und Ausgabeverhalten der jeweiligen Transition separat angegeben, da dieses aus der grafischen Darstellung nicht ersichtlich ist.

### Abbildung von *Activities*

Eine *Activity* (Aktivität) bezeichnet in BPMN als Sammelbegriff alle Formen von unterscheidbaren Arbeitsschritten, die im Rahmen eines Geschäftsprozesses ausgeführt werden.

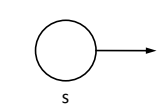
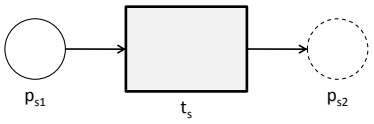
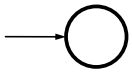
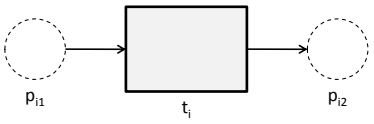
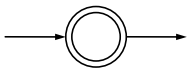
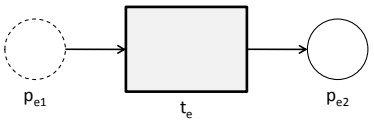
| BPMN-Event | IFnet-Konstrukt |
|---|---|
|  |  $\textsc{In}(t_s, p_{s1}) = [\{\texttt{black}\}]$<br>$\textsc{Out}(t_s, p_{s2}) = [\{\texttt{black}\}]$ |
|  |  $\textsc{In}(t_e, p_{e1}) = [\{\texttt{black}\}]$<br>$\textsc{Out}(t_e, p_{e2}) = [\{\texttt{black}\}]$ |
|  |  $\textsc{In}(t_i, p_{i1}) = [\{\texttt{black}\}]$<br>$\textsc{Out}(t_i, p_{i2}) = [\{\texttt{black}\}]$ |

Tabelle 3.1.: Übersetzung von BPMN-*Events*.

Während BPMN auch komplexe Aktiviäten kennt, die mehrere Arbeitsschritte zusammenfassen (etwa als Sub-Prozess), werden hier nur atomare Aktivitäten (*Tasks*) betrachtet. Dies ist keine Beschränkung der Allgemeinheit, da sich jede komplexe Aktivität durch eine Menge von *Tasks* darstellen lässt.

Eine Aktivität (im Sinne einer *Task*) wird in BPMN durch ein Rechteck mit abgerundeten Ecken dargestellt. Sie wird in IFnet durch eine Transition modelliert. Das entsprechende Konstrukt ist in Tabelle 3.2 dargestellt.

### Abbildung von *Gateways*

*Gateways* dienen in BPMN zur Beschreibung der Bereiche eines Geschäftsprozesses, an denen sich Ausführungspfade verzweigen bzw. vereinigen. In BPMN gibt es verschiedene *Gateway*-Typen, um *exklusive* Verzweigungen (bei denen von mehreren möglichen Pfaden einer ausgeführt wird) sowie *parallele* Verzweigungen (bei denen alle angegebenen Pfade nebenläufig ausgeführt werden) darzustellen. Zu diesen *Gateways* gibt es jeweils entsprechende Gegenstücke, die die Pfade (ob exklusiv oder parallel ausgeführt) wieder zusammenführen.

*Gateways* werden in BPMN durch auf der Spitze stehende Quadrate („Diamanten“) dargestellt, die – je nach Typ – mit unterschiedlichen Symbolen gekennzeichnet sein können. Abbildung 3.7 zeigt die wichtigsten drei Typen:

1. Ein ***Parallel-Gateway*** beschreibt eine Verzweigung, bei der alle angegebenen Pfade nebenläufig ausgeführt werden. Ein solcher *Gateway* ist in Abbildung 3.7(a)

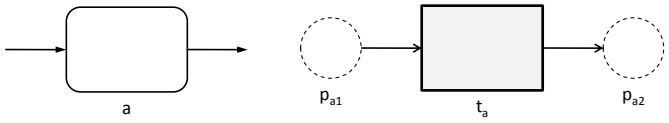
| BPMN-Activity | IFnet-Konstrukt |
|---|---|
|   | $\textsc{In}(t_a, p_{a1}) = [\{\texttt{black}\}]$<br>$\textsc{Out}(t_a, p_{a2}) = [\{\texttt{black}\}]$ |

Tabelle 3.2.: Übersetzung von BPMN-*Activities*.

dargestellt. Die Vereinigung von parallel ausgeführten Pfaden wird durch dasselbe Symbol beschrieben.

2. Ein **datenbasierter *Exclusive-Gateway*** beschreibt eine Verzweigung, bei der nur jeweils einer der angegebenen Pfade ausgeführt wird. Die Entscheidung darüber, welcher Pfad ausgeführt wird, erfolgt durch die Auswertung von Datenobjekten. Beide in Abbildung 3.7(b) dargestellten Symbole können benutzt werden, um einen solchen *Gateway* darzustellen. Dieselben Symbole werden benutzt, um die Vereinigung der alternativen Ausführungspfade darzustellen.

3. Bei einem **ereignisbasierten *Exclusive-Gateway*** wird wie oben jeweils nur einer der möglichen Ausführungspfade ausgewählt. Allerdings wird hier die Entscheidung nicht auf der Grundlage von Datenobjekten getroffen, sondern hängt von dem Eintreten eines Ereignisses ab: Es wird der Pfad gewählt, dessen zugeordnetes Ereignis zuerst eintritt. Das entsprechende Symbol ist in Abbildung 3.7(c) dargestellt.

Neben diesen drei Typen beschreibt die BPMN-Spezifikation zwei weitere *Gateways* (*Inclusive* und *Complex*), die hier nicht berücksichtigt werden, da sie sich durch die oben dargestellten Typen abbilden lassen.

In Tabelle 3.3 ist die Abbildung der oben beschriebenen *Gateways* in die entsprechenden IFnet-Konstrukte dargestellt. Für jeden *Gateway*-Typ ist der einfachste Fall mit jeweils zwei ein- oder ausgehenden Ausführungspfaden dargestellt. In Fällen mit mehr als zwei Pfaden kann die Abbildung in der vorgestellten Systematik verallgemeinert werden.

Ein *Parallel-Gateway* wird auf eine Transition abgebildet, die aus einer Eingangsstelle eine schwarze Marke empfängt, eine weitere erzeugt und diese beiden Marken an die zwei Ausgangsstellen weitergibt, so dass auf diesen Pfaden die Ausführung angestoßen wird. Entsprechend wird für die Vereinigung der Pfade eine Transition eingeführt, die auf dem jeweils ausgeführten Pfad eine Marke empfängt und diese weiterreicht.

Für die Abbildung eines datenbasierten *Exclusive-Gateways* werden zwei Transitionen eingeführt, deren Eintreten die Erfüllung der jeweiligen (datenabhängigen) Bedingung signalisiert.[5] Ist die Bedingung erfüllt, schaltet die entsprechende Transition, indem ei-

[5] Eine Alternative ist die Abbildung auf nur eine Transition, die die empfangene Marke wahlweise an eine der Ausgangsstellen weitergibt. Die hier gewählte Abbildung ist dagegen T-deterministisch und orientiert sich stärker an der Semantik klassischer Petri-Netze.

(a) *Parallel* (b) *Exclusive* (*Data-Based*) (c) *Exclusive* (*Event-Based*)

Abbildung 3.7.: *Gateway*-Typen in BPMN.

ne scharze Marke von der gemeinsamen Eingangsstelle entfernt und in der jeweiligen Ausgangsstelle platziert wird. Die Abbildung der Ausführungsbedingungen kann nicht automatisiert werden, da BPMN das Format der entsprechenden Ausdrücke nicht spezifiziert (OMG, 2009). Diese müssen ggf. in einem manuellen Ergänzungsschritt auf ein IFnet-Prädikat abgebildet werden.

Die Abbildung ereignisbasierter *Exclusive-Gateways* ähnelt der vorhergehenden, wobei hier keine zusätzlichen Transitionen eingeführt, sondern die Ereignisse auf den verzweigenden Pfaden – die ebenfalls durch Transitionen modelliert werden – direkt angeschlossen werden. Auf diese Weise kann die Transition, die das zuerst eintretende Ereignis modelliert, die schwarze Marke direkt aus der Eingangsstelle entnehmen, ohne dass eine weitere Transition zwischengeschaltet wäre.

Die Vereinigung von Ausführungspfaden, die durch *Exclusive-Gateways* verzweigt worden sind, wird wiederum auf zwei Transitionen abgebildet, die auf ihrem jeweiligen Pfad ggf. eine schwarze Marke entgegennehmen und diese zu einer gemeinsamen Ausgangsstelle weiterleiten. Unabhängig davon, ob daten- oder ereignisbasiert verzweigt wurde, wird dasselbe IFnet-Konstrukt für die Vereinigung benutzt.

### Abbildung von *Message Flows*

Während BPMN-*Sequence Flows* (d.h. der Kontrollfluss) im Zuge der Abbildung der *Flow Objects* mit übersetzt wird, werden *Message Flows* (d.h. der Nachrichtenaustausch) in einem separaten Schritt auf IFnet-Konstrukte abgebildet. *Message Flows* werden in BPMN durch gestrichelte Pfeile dargestellt, die mit einer Bezeichnung versehen sind und mit einem Datenobjekt assoziiert sein können. Nachrichtenaustausch wird in BPMN ausschließlich zwischen verschiedenen *Pools* dargestellt.

Nachrichten können in BPMN zwischen Ereignissen und Aktivitäten ausgetauscht werden (die zu jeweils unterschiedlichen *Pools* gehören müssen). Da sowohl Ereignisse als auch Aktivitäten in IFnet durch Transitionen dargestellt werden, ist das entsprechende IFnet-Konstrukt in allen Fällen dasselbe.

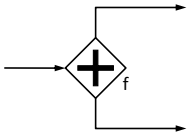
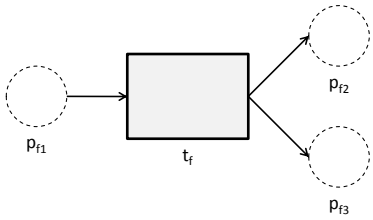
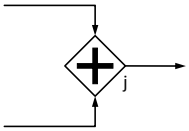
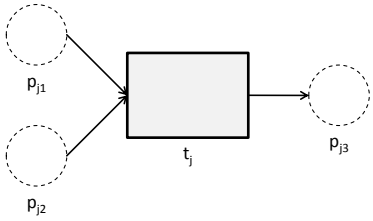
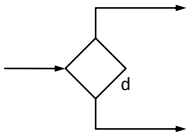
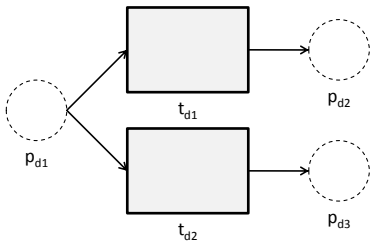
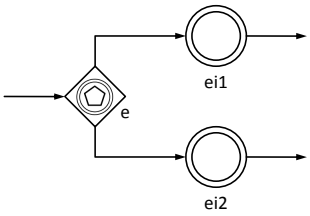
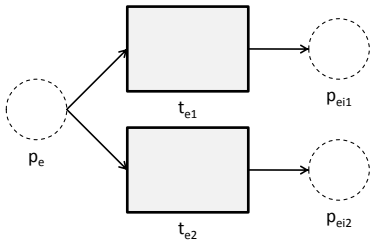
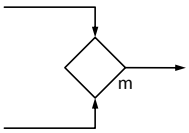
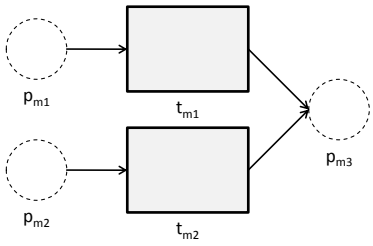
| BPMN-Gateway | IFnet-Konstrukt | |
|---|---|---|
|  |  | $\text{In}(t_f, p_{f1}) = [\{\texttt{black}\}]$<br>$\text{Out}(t_f, p_{f2}) = [\{\texttt{black}\}]$<br>$\text{Out}(t_f, p_{f3}) = [\{\texttt{black}\}]$ |
|  |  | $\text{In}(t_j, p_{j1}) = [\{\texttt{black}\}]$<br>$\text{In}(t_j, p_{j2}) = [\{\texttt{black}\}]$<br>$\text{Out}(t_j, p_{j3}) = [\{\texttt{black}\}]$ |
|  |  | $\text{In}(t_d1, p_{d1}) = [\{\texttt{black}\}]$<br>$\text{Out}(t_d1, p_{d2}) = [\{\texttt{black}\}]$<br>$\text{In}(t_d2, p_{d1}) = [\{\texttt{black}\}]$<br>$\text{Out}(t_d2, p_{d3}) = [\{\texttt{black}\}]$ |
|  |  | $\text{In}(t_e1, p_{e1}) = [\{\texttt{black}\}]$<br>$\text{Out}(t_e1, p_{e2}) = [\{\texttt{black}\}]$<br>$\text{In}(t_e2, p_{e1}) = [\{\texttt{black}\}]$<br>$\text{Out}(t_e2, p_{e3}) = [\{\texttt{black}\}]$ |
|  |  | $\text{In}(t_m1, p_{m1}) = [\{\texttt{black}\}]$<br>$\text{Out}(t_m1, p_{m3}) = [\{\texttt{black}\}]$<br>$\text{In}(t_m2, p_{m2}) = [\{\texttt{black}\}]$<br>$\text{Out}(t_m2, p_{m3}) = [\{\texttt{black}\}]$ |

Tabelle 3.3.: Übersetzung von BPMN-*Gateways*.

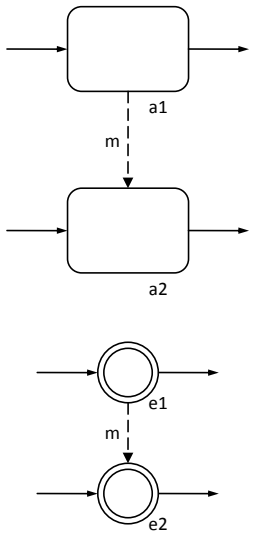
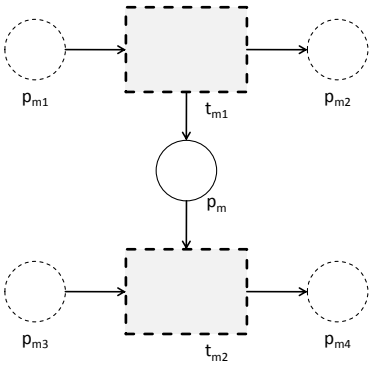
| BPMN-Message Flow | IFnet-Konstrukt |
|---|---|
|  |  $\textsc{Out}(t_m1, p_m) = [\{\mathtt{m}\}]$<br>$\textsc{In}(t_m2, p_m) = [\{\mathtt{m}\}]$ |

Tabelle 3.4.: Übersetzung von BPMN-*Message Flows*.

Tabelle 3.4 zeigt die Übersetzung von BPMN-*Message Flows* für den Nachrichtenaustausch zwischen zwei Aktivitäten sowie zwischen zwei Ereignissen. Die Fälle, in denen eine Nachricht von einem Ereignis zu einer Aktivität oder von einer Aktivität zu einem Ereignis fließt, werden in derselben Weise übersetzt. In Fällen, in denen ein BPMN-Ausführungspfad mit dem Empfangen einer Nachricht beginnt, wird die zweite Eingangsstelle der Transition, die dieses Ereignis modelliert, weggelassen (Stelle $\mathsf{p_{m3}}$ in der Abbildung).

### Abbildung von *Pool*-Namen

Ein *Pool* repräsentiert in BPMN eine an einem Geschäftsprozess teilnehmende Partei. Er wird durch ein mit einem Namen annotiertes Rechteck dargestellt, das die *Flow Objects*, die der entsprechenden Partei zugeordnet sind, einschließt. Ein *Pool* wird in das abgebildete IFnet überführt, indem sein Name als Subjektbezeichner für diejenigen Transitionen übernommen wird, die aus der Übersetzung derjenigen *Flow Objects* hervorgegangen sind, die diesem *Pool* zugeordnet sind.

### Anfangszustand

Der Anfangszustand (d.h. die initiale Belegung mit Marken) des abgebildeten IFnets wird erzeugt, indem jede Anfangsstelle (d.h. jede Stelle die keine Eingangsknoten hat) mit genau einer schwarzen Marke belegt wird.

### Manuelle Ergänzung abgebildeter BPMN-Prozesse

Im Anschluss an die beschriebene Abbildung müssen ggf. weitere Ergänzungen und Anpassungen an einem erzeugten IFnet-Modell vorgenommen werden, die nicht automatisiert werden können, da die entsprechenden Eigenschaften in BPMN nicht bzw. mehrdeutig spezifiziert werden. Diese betreffen im Wesentlichen:

- **Nachrichtenfluss innerhalb eines *Pools*.** BPMN spezifiziert ausschließlich den Nachrichtenaustausch zwischen verschiedenen *Pools* durch *Message Flows*. Die Verbreitung von Datenobjekten innerhalb eines Pools wird nicht formal spezifiziert (OMG, 2009, S. 99), sondern kann lediglich durch informelle Annotationen angegeben werden. Im übersetzten IFnet müssen die entsprechenden Anpassungen daher von Hand vorgenommen werden.
- **Datenzugriff.** BPMN spezifiziert nicht formal, ob auf ein Datum lesend oder schreibend zugegriffen wird. Die entsprechenden Transitionsattribute müssen im übersetzten IFnet-Modell ergänzt werden.
- **Interaktion von Instanzen.** Die Interaktion mehrerer Instanzen kann in BPMN formal nicht eindeutig spezifiziert werden (Decker und Barros, 2008). So kann beispielsweise nicht formal ausgedrückt werden, ob mehrere Instanzen parallel mit derselben Partner-Instanz interagieren können oder ob dies sequentiell geschehen muss. Das Verhalten des IFnet-Modells muss dementsprechend definiert werden.
- **Prädikate.** Verzweigungsbedingungen werden in einem BPMN-Modell ebenfalls informell durch Annotationen spezifiziert (OMG, 2009, S. 73) und müssen manuell in ein entsprechendes IFnet-Prädikat umgewandelt werden, das die Abhängigkeiten einer Transition von Datenobjekten ausdrückt.
- **Markierung.** Im Standardfall ist der Anfangszustand des übersetzten IFnets durch jeweils eine schwarze Marke in jeder Anfangsstelle gegeben. Soll die Verarbeitung mehrerer Durchläufe dargestellt werden, müssen ggf. zusätzliche Marken eingefügt werden. Auch die initiale Belegung mit Datenobjekten (d.h. gefärbten Marken) muss von Hand geschehen, da BPMN-Modelle zu den entsprechenden Datenobjekten keine eindeutigen Vorgaben machen.

## 3.4 Strategien zur Auszeichnung von IFnet-Modellen

Dieser Abschnitt beschreibt die Umsetzung des zweiten Schrittes im InDico-Vorgehensmodell, die automatisierte Übersetzung von Sicherheitsanforderungen in Informationsfluss-Policies. In der Literatur zur mehrstufigen Sicherheit wird das Problem der Auszeichnung der Komponenten eines Systemmodells mit Sicherheitsstufen bislang nicht thematisiert, d.h. eine Auszeichnung wird hier als gegeben angenommen bzw. ist das Ergebnis eines manuell durchgeführten Vorgangs.

InDico strebt eine (Teil-)Automatisierung dieses Vorgangs durch die Definition von *Auszeichnungsstrategien* an. Eine solche Strategie zeichnet – in Abhängigkeit von der jeweiligen Sicherheitsanforderung – die Marken und Transitionen eines IFnet-Modells mit Sicherheitsstufen aus und ordnet den Subjekten Freigaben zu, so dass die Nicht-Übertragung `High`-klassifizierter Informationen zu Subjekten mit `Low`-Freigabe die Erfüllung der entsprechenden Sicherheitsanforderung impliziert. Neben der Auszeichnung mit Sicherheitsstufen und Freigaben kann eine Strategie, sofern erforderlich, auch das IFnet-Modell selbst anpassen.

Im Folgenden werden zwei Auszeichnungsstrategien definiert, die Isolationsanforderungen zum Einen für Datenobjekte und zum Anderen für Prozess-Instanzen umsetzen. Die erste Strategie – genannt ACMLabeler – arbeitet auf der Grundlage einer Zugriffskontrollmatrix, in der Zugriffsrechte auf Datenobjekte notiert sind. Die zweite Strategie – MultiInstanceLabeler – erzeugt auf der Basis eines gegebenen IFnet-Modells ein erweitertes Modell, in dem ein (Teil-)Prozess geklont und an das ursprüngliche Modell angefügt wird. Anschließend zeichnet die Strategie das so entstandene Modell – das die nebenläufige Ausführung zweier Prozess-Instanzen darstellt – mit Sicherheitsstufen aus, die die Isolation der beiden Instanzen ausdrücken.

### 3.4.1 ACMLabeler

Vertraulichkeitsanforderungen an Datenobjekte sind Bestandteil praktisch jeder Sicherheits- oder Compliance-Richtlinie. Das übliche Mittel zu ihrer Formalisierung in einem technischen System sind Zugriffsrichtlinien (Sandhu und Samarati, 1994). In ihrem Kern legen diese fest, welche Zugriffsrechte ein Subjekt bzgl. eines (Daten-)Objektes ausüben darf.

Das Ziel der ACMLabeler-Strategie ist die Auszeichnung eines IFnet-Modells, so dass die in einem Datenobjekt gespeicherten Informationen nicht an Subjekte fließen können, die diese laut Zugriffsrichtlinie nicht erhalten dürfen. Zu diesem Zweck wird die durch die Zugriffsrichtlinie (intensional) umschriebene Sicherheitsanforderung bzgl. eines Datenobjektes in eine (extensionale) Informationsfluss-Policy übersetzt. Durch diese Übersetzung werden die an das Datenobjekt gestellten Sicherheitsanforderungen verstärkt: Neben der Unterbindung von direkten Zugriffen durch unbefugte Parteien wird zusätzlich die Vermeidung von Informationslecks durch indirekte Datenflüsse (über dritte Parteien) und

| | x | y | z |
|---|---|---|---|
| A | *read* | – | *write* |
| B | – | *read* | *write* |
| C | – | – | *read, write* |

Tabelle 3.5.: Beispiel für eine Zugriffskontrollmatrix.

verdeckte Informationsübertragung gefordert (vgl. dazu die Gegenüberstellung von intensionalen und extensionalen Spezifikationen in Abschnitt 1.5). Eine dieser Strategie zugrunde liegende Annahme ist, dass durch die Zugriffsregeln tatsächlich ein übergeordnetes Schutzziel (hier Vertraulichkeit) umschrieben werden soll. Es wird davon ausgegangen, dass *nicht* ausschließlich Datenzugriffe geregelt werden und anderweitige Wege der Informationsübertragung ausdrücklich unberücksichtigt sein sollen.

Eine Zugriffskontrollmatrix spezifiziert die Rechte (wie z.B. „lesen", „schreiben" oder „ausführen"), die Subjekte auf Ressourcen ausüben dürfen. Die folgende Definition charakterisiert eine einfache Zugriffskontrollmatrix, wie sie im Rahmen dieser Arbeit benutzt wird. Sie ist definiert als eine Abbildung $ACM$, die ein Paar aus einem Subjekt (aus der Menge $\mathcal{U}$) und einer Ressource (aus der Menge $\mathcal{C}_c$) auf eine Menge von Rechten abbildet. Die Rechte beschränken sich hier auf *read* (Lesezugriff) und *write* (Schreibzugriff).

**Definition 3.16 (Zugriffskontrollmatrix).** *Eine Zugriffskontrollmatrix $ACM$ ist eine Funktion, die einem Subjekt bzgl. einer gefärbten Marke Zugriffsrechte zuordnet:*

$$ACM : \mathcal{U} \times \mathcal{C}_c \to \mathcal{P}(\{read, write\}). \qquad \dashv$$

Zugriffskontrollmatrizen dieser Art können von praktisch allen modernen Policy-Sprachen, wie z.B. XACML (Moses, 2005), EPAL (Ashley u. a., 2003) und ExPDT (Kähmer, 2009), ausgedrückt werden.

Tabelle 3.5 zeigt ein Beispiel für eine Zugriffskontrollmatrix, anhand derer das Vorgehen der ACMLabeler-Strategie illustriert wird. In dem Beispiel sind die Subjekte in der ersten Spalte und die Datenobjekte in der obersten Zeile aufgetragen. Ein Strich in einer Zelle bedeutet, dass das entsprechende Subjekt bzgl. des entsprechenden Datenobjektes keinerlei Rechte besitzt.

Bezogen auf das Datenobjekt `x` zeichnet ACMLabeler ein IFnet-Modell wie folgt aus: Die Marke, die `x` repräsentiert, wird mit der Sicherheitsstufe `High` ausgezeichnet. Entsprechend erhält das Subjekt A, das laut Zugriffsmatrix lesend auf `x` zugreifen darf, die Freigabe `High`, während die Subjekte B und C, die keinerlei Rechte bzgl. `x` besitzen, die Freigabe `Low` erhalten.

ACMLabeler definiert in Abhängigkeit einer gegebenen Zugriffsmatrix $ACM$ und einer darin enthaltenen Ressource $x$ die Abbildungen $\textsc{Label}_{\mathcal{SC}}^{ACM(x)}$ und $\textsc{Clear}_{\mathcal{SC}}^{ACM(x)}$, die

die Auszeichnung von Marken und Transitionen bzw. die Freigaben der Subjekte festlegen. $\textsc{Label}_{\mathcal{SC}}^{ACM(x)}$ zeichnet die Marke, die die Ressource $x$ repräsentiert, mit `High` aus, während die übrigen gefärbten Marken die Einstufung `unlabeled` erhalten. Besitzt ein Subjekt ein Leserecht auf $x$, wird ihm von $\textsc{Clear}_{\mathcal{SC}}^{ACM(x)}$ die Freigabe `High` erteilt, andernfalls erhält es die Freigabe `Low`. Die folgenden Definitionen fassen diese Abbildung in formaler Weise.

**Definition 3.17** ($\textsc{Label}_{\boldsymbol{\mathcal{SC}}}^{\boldsymbol{ACM(x)}}$). *Für ein IFnet $N$, eine Zugriffskontrollmatrix $ACM$ und eine Markenkennung $x \in \mathcal{C}_c$ ist die Funktion* $\textsc{Label}_{\mathcal{SC}}^{ACM(x)} : \mathcal{C}_c \rightarrow \mathcal{SC} \cup \{\texttt{unlabeled}\}$ *wie folgt definiert:*

$$\textsc{Label}_{\mathcal{SC}}^{ACM(x)}(c) = \begin{cases} \texttt{High} & \textit{falls} \quad x = c \\ \texttt{unlabeled} & \textit{falls} \quad x \neq c \end{cases}$$

⊣

**Definition 3.18** ($\textsc{Clear}_{\boldsymbol{\mathcal{SC}}}^{\boldsymbol{ACM(x)}}$). *Für ein IFnet $N$, eine Zugriffskontrollmatrix $ACM$ und eine Markenkennung $x \in \mathcal{C}_c$ ist die Funktion* $\textsc{Clear}_{\mathcal{SC}}^{ACM(x)} : \mathcal{U} \rightarrow \mathcal{SC}$ *wie folgt definiert:*

$$\textsc{Clear}_{\mathcal{SC}}^{ACM(x)}(s) = \begin{cases} \texttt{High} & \textit{falls} \quad ACM(s,x) \cap \{read\} \neq \emptyset \\ \texttt{Low} & \textit{falls} \quad ACM(s,x) \cap \{read\} = \emptyset \end{cases}$$

⊣

Da sich eine Zugriffsmatrix ausschließlich auf Datenobjekte bezieht, werden Transitionen von dieser Strategie nicht klassifiziert. Sofern ein Prozess-Modell nicht nur bzgl. eines Datenobjektes sondern hinsichtlich einer vollständigen Matrix $ACM$ zertifiziert werden soll, muss der Auszeichnungsvorgang (und die anschließende Zertifizierung) für jede Ressource, für die $ACM$ Zugriffsrechte definiert, wiederholt werden.

### 3.4.2 MultiInstanceLabeler

Während die ACMLabeler-Strategie die Isolation der in Datenobjekten gespeicherten Informationen ausdrückt, befasst sich die zweite Strategie – MultiInstanceLabeler– mit der Isolation von (Teil-)Prozessen, die nebenläufig ausgeführt werden. Die Isolation von Prozessen ist eine zentrale Sicherheitsanforderung, die Dienst-Infrastrukturen gewährleisten müssen, auf denen gleichzeitig mehrere Kundenprozesse ausgeführt werden (vgl. Abschnitt 1.4).

Den Ausgangspunkt von MultiInstanceLabeler bilden ein IFnet-Modell und eine darin enthaltende Teilmenge von Transitionen, die einen (Teil-)Prozess definieren. MultiInstanceLabeler erstellt eine Kopie dieses (Teil-)Prozesses mit gleicher Struktur und Semantik und verbindet diese Kopie mit dem ursprünglichen Modell. Auf diese Weise wird ein IFnet-Modell erzeugt, das die parallele Ausführung von zwei Instanzen des (Teil-)

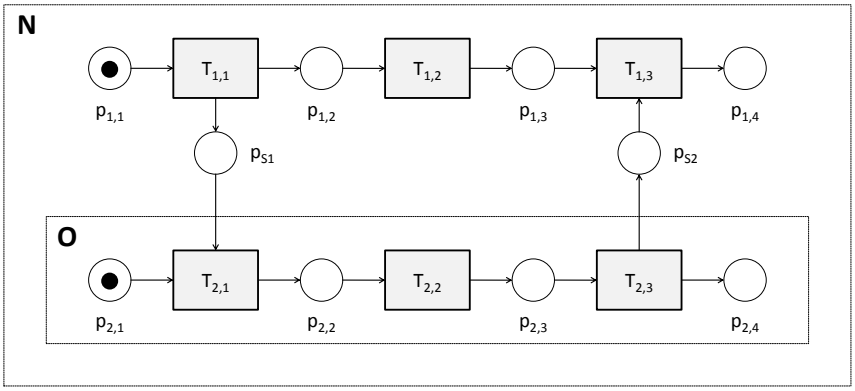



Abbildung 3.8.: Schematische Darstellung eines IFnet-Modells interagierender Teilprozesse.

Prozesses modelliert. Anschließend werden die beiden Instanzen mit unterschiedlichen Sicherheitsstufen ausgezeichnet. Das so erzeugte und ausgezeichnete Modell kann dann darauf untersucht werden, ob es zu Informationsflüssen zwischen beiden Instanzen und damit einer Verletzung der Isolation kommen kann.

MultiInstanceLabeler ist eine hilfreiche Strategie, da in vielen Geschäftsprozessen das Auftreten mehrerer parallel ausgeführter Instanzen zwar vorgesehen ist, in den entsprechenden Modellen aber in der Regel nur eine Instanz modelliert ist (vgl. etwa das Anwendungsszenario in Kapitel 5). Um in diesem Zusammenhang auftretende Isolationsverletzungen erkennen zu können, ist eine explizite Modellierung des vollständigen Prozessverhaltens notwendig, die von MultiInstanceLabeler unterstützt wird.

Im Folgenden wird die Strategie detailliert beschrieben. Zunächst werden die Anforderungen an das Ausgangsmodell definiert und anschließend die Struktur und das Verhalten des erzeugten Modells sowie die Auszeichnung beschrieben.

### Anforderungen an das Ausgangsmodell

Die Eingabe von MultiInstanceLabeler bildet ein markiertes IFnet $N$ mit Anfangszustand $M$, das ein Teilnetz mit mindestens einer Anfangsstelle enthält. Dieses Teilnetz ist über Stellen mit dem übrigen Netz verbunden. Ein solches Netz ist schematisch in Abbildung 3.8 dargestellt: Das Netz $N$ enthält das Teilnetz $O$, welches die Anfangsstelle $p_{2,1}$ besitzt. Die einzigen Verbindungen zum übrigen Netz bilden die Stellen $p_{S1}$ und $p_{S2}$.

IFnets mit der oben beschriebenen Struktur modellieren typischerweise die Interaktion zweier (Teil-)Prozesse, wobei die Verbindungsstellen Kommunikationskanäle repräsentieren. Ein Teilnetz wie $O$, das den obigen Anforderungen entspricht, entsteht beispielsweise durch die Abbildung eines BPMN-*Pools* auf IFnet durch die in Abschnitt 3.3 beschriebene Übersetzung.

**Definition 3.19 (Teilnetz).** *Gegeben sei ein IFnet-Modell* $N = ((P, T, F, \text{In}, \text{Out}), \text{S}_\mathcal{U}, \text{A}, \text{G}, \text{Label}_\mathcal{SC}, \text{Clear}_\mathcal{SC})$ *mit Anfangszustand* $M$*. Für eine Teilmenge* $T_O \subseteq T$ *der Transitionen von* $N$ *ist das Teilnetz* $O = (P_O, T_O, F_O)$ *wie folgt definiert:*

- *Eine Stelle ist in* $P_O$*, wenn jede ihrer Eingangs- und Ausgangsknoten eine Transition aus* $T_O$ *ist:*

  $$P_O = \{p \in P \mid \forall t \in {\bullet p} : t \in T_O \ \wedge \ \forall t \in p{\bullet} : t \in T_O\}.$$

- *Die Flussrelation* $F_O$ *beschreibt alle Verbindungen zwischen Knoten aus* $O$*:*

  $$F_O = \{(x, y) \mid x, y \in P_O \cup T_O \ \wedge \ xFy\}.$$

⊣

Damit ein Teilnetz $O$ von MultiInstanceLabeler geklont werden kann, muss es die folgenden Bedingungen erfüllen:

1. Es gibt eine Stelle $i \in P_O \cap P_I$, d.h. $O$ besitzt mindestens eine Anfangsstelle.
2. Es gibt eine Menge $P^O_{SI}$ von Stellen, über die $O$ mit dem übrigen Netz verbunden ist:

   $$P^O_{SI} = \{p \in P \setminus P_O \mid \exists t \in T_O : tFp \vee pFt\}.$$

3. Es gibt kein Subjekt, in dessen Namen sowohl eine Transition aus $T_O$ als auch aus $T \setminus T_O$ ausgeführt wird.

   $$\forall (t_1, t_2) \in T_O \times (T \setminus T_O) : \text{S}_\mathcal{U}(t_1) \neq \text{S}_\mathcal{U}(t_2).$$

### Struktur des erzeugten Netzes

Im ersten Schritt erzeugt MultiInstanceLabeler eine strukturelle Kopie des Teilnetzes $O$ und verbindet diese mit dem ursprünglichen Netz. Dazu wird ein Tripel $Q = (P_Q, T_Q, F_Q)$ aus einer Menge von Stellen, einer Menge von Transitionen und einer Flussrelation mit den folgenden Eigenschaften erzeugt:

- $P_Q$ enthält die gleiche Anzahl Elemente wie $P_O$ und ist zu $P$ disjunkt:

  $$|P_Q| = |P_O| \ \wedge \ P_Q \cap P = \emptyset.$$

- $T_Q$ enthält die gleiche Anzahl Elemente wie $T_O$ und ist zu $T$ disjunkt:

  $$|T_Q| = |T_O| \ \wedge \ T_Q \cap T = \emptyset.$$

Die bijektive Abbildung $\Pi$ wird definiert, um Stellen aus $P_O$ bzw. Transitionen aus $T_O$ auf ein Gegenstück in $P_Q$ bzw. $T_Q$ abzubilden. Aufbauend auf $\Pi$ wird die Flussrelation $F_Q$ definiert als:

$$F_Q = \{(\Pi(a), \Pi(b)) \in (P_Q \times T_Q) \cup (T_Q \times P_Q) \mid aF_Ob\}.$$

Dieser „Klon“ $Q$ des Teilnetzes $O$ wird anschließend mit dem Ausgangsnetz $N$ vereinigt. Damit wird ein Netz $N^+ = ((P^+, T^+, F^+, \text{IN}^+, \text{OUT}^+), \text{S}_\mathcal{U}{}^+, \text{A}^+, \text{G}^+, \text{LABEL}_\mathcal{SC}{}^+, \text{CLEAR}_\mathcal{SC}{}^+)$ mit Anfangszustand $M^+$ erzeugt, das im Gegensatz zu $N$ zusätzlich das geklonte und verbundene Teilnetz $Q$ enthält. Die Struktur von $N^+$ (d.h. seine Stellen, Transitionen und die Flussrelation) sind wie folgt definiert:

- $P^+ = P \cup P_Q$.
- $T^+ = T \cup T_Q$.
- $F^+ = F \cup F_Q \cup \big\{(\Pi(t), p), (p, \Pi(t)) \mid t \in T_O \wedge p \in P^O_{SI}\big\}$.

Die Menge $P_I^+ \subseteq P^+$ der Anfangsstellen von $N^+$ ist damit:

$$P_I^+ = P_I \cup \{\Pi(p) \mid p \in P_I \cap P_O\}.$$

### Verhalten des erzeugten Netzes

MultiInstanceLabeler definiert das Verhalten des neu erzeugten Teilprozesses $Q$ analog zu dem Verhalten von $O$. Für die Menge der von $O$ verwendeten gefärbten Marken wird eine entsprechend große Menge von Marken erzeugt, die von $Q$ verwendet werden. Es werden folgenden Bezeichnungen verwendet:

- $\mathcal{C}_\mathsf{c}^N \subseteq \mathcal{C}_\mathsf{c}$ ist die Menge der gefärbten Marken, die in $N$ potentiell verarbeitet werden, d.h. die von einer Transition in $N$ potentiell eingelesen, erzeugt oder ausgegeben werden.
- $\mathcal{C}_\mathsf{c}^O \subseteq \mathcal{C}_\mathsf{c}^N$ ist die Menge der gefärbten Marken, die in $O$ potentiell verarbeitet werden.
- $\mathcal{C}_\mathsf{c}^Q \subseteq (\mathcal{C}_\mathsf{c} \backslash \mathcal{C}_\mathsf{c}^N) \cup \mathcal{C}_\mathsf{c}^O$ ist eine Menge von gefärbten Marken mit derselben Mächtigkeit wie $\mathcal{C}_\mathsf{c}^O$.

Die Funktion $\Pi$ fungiert auch hier wieder als bijektive Abbildung, die eine Marke aus $\mathcal{C}_\mathsf{c}^O$ auf ein Gegenstück in $\mathcal{C}_\mathsf{c}^Q$ abbildet. Das Ein- und Ausgabeverhalten einer Transition aus $T_Q$ wird analog zu ihrem jeweiligen Gegenstück aus $T_O$ definiert, wobei die Verarbeitung einer Marke aus $\mathcal{C}_\mathsf{c}^O$ durch die Verarbeitung der entsprechenden Marke aus $\mathcal{C}_\mathsf{c}^Q$ ersetzt wird. Dabei müssen die Mengen $\mathcal{C}_\mathsf{c}^O$ und $\mathcal{C}_\mathsf{c}^Q$ nicht disjunkt sein: Es können auch Fälle modelliert werden, in denen mehrere Instanzen das gleiche Datenobjekt verarbeiten, d.h. die gleiche gefärbte Marke verwenden. Entsprechend kann als Parameter der MultiInstanceLabeler-Strategie die Funktion $\Pi$ ausgestaltet werden.

Zur Konstruktion der Ein- und Ausgabemengen der Transitionen aus $T^+$ wird der Operator $\Phi$ definiert, der für eine Multimenge von Marken aus $\mathcal{C}^+$ eine andere, gleichmächtige Multi-Menge konstruiert, in der Marken aus $\mathcal{C}_c^O$ durch Marken aus $\mathcal{C}_c^Q$ ersetzt sind.

**Definition 3.20 (Φ-Operator für Multimengen).** *Für eine Multimenge $B \in \mathcal{C}^+$ wird die Menge $\Phi(B)$ konstruiert, indem für jedes Element aus $B$ wie folgt ein entsprechendes Element zu $\Phi(B)$ hinzugefügt wird:*

- *Für eine Marke $b \in B$, die nicht Element von $\mathcal{C}_c^O$ ist, wird die Marke $b$ zu $\Phi(B)$ hinzugefügt.*
- *Für eine Marke $b \in B$, die gleichzeitig Element von $\mathcal{C}_c^O$ ist, wird die Marke $\Pi(b)$ zu $\Phi(B)$ hinzugefügt.* ⊣

In entsprechender Weise wird $\Phi$ für die Konstruktion von Reihungen, die als Elemente Multi-Mengen von Marken enthalten, erweitert.

**Definition 3.21 (Φ-Operator für Reihungen).** *Für eine Reihung $A \in \mathcal{C}^+[]$ sind die Elemente der Reihung $\Phi(A)$ definiert durch:*

$\Phi(A)[i] = \Phi(A[i])$ *für alle* $i \in \{0, .., |A| - 1\}$. ⊣

Das Ein- und Ausgabeverhalten der Transitionen aus $T^+$ wird wie folgt festgelegt.

- Das Verhalten einer Transition $t \in T_O$ bleibt unverändert:
  - $\text{In}^+(t, p) = \text{In}(t, p)$ für alle $p \in \bullet t$.
  - $\text{Out}^+(t, q) = \text{Out}(t, q)$ für alle $q \in t\bullet$.
- Das Verhalten einer Transition $\Pi(t) \in T_Q$ wird an die Marken aus $\mathcal{C}_c^Q$ angepasst:
  - $\text{In}^+(\Pi(t), \Pi(p)) = \Phi(\text{In}(t, p))$ für alle $\Pi(p) \in \bullet\Pi(t)$.
  - $\text{Out}^+(\Pi(t), \Pi(q)) = \Phi(\text{Out}(t, q))$ für alle $\Pi(q) \in \Pi(t)\bullet$.
- Das Verhalten einer Transition $t \in T^+ \setminus (T_O \cup T_Q)$ wird dermaßen erweitert, dass die Transition, sofern sie Marken aus $\mathcal{C}_c^O$ verarbeitet (d.h. einliest, erzeugt oder ausgibt), alternativ auch Marken aus $\mathcal{C}_c^Q$ verarbeiten kann:
  - $\text{In}^+(t, p) = \text{In}(t, p) \cup (\Phi(\text{In}(t, p)) \setminus \text{In}(t, p))$ für alle $p \in \bullet t$.
  - $\text{Out}^+(t, q) = \text{Out}(t, q) \cup (\Phi(\text{Out}(t, q)) \setminus \text{Out}(t, q))$ für alle $q \in t\bullet$.

### Transitionsattribute

Die Transitionsattribute legen die Zuordnung von Subjekten zu Transitionen, den Zugriffsmodus auf Ressourcen und die Zuordnung von Prädikaten zu Transitionen fest. Als Hilfskonstrukte werden die folgenden beiden Mengen verwendet:

- $\mathcal{U}_O$ bezeichnet die Menge aller Subjekte, die einer Transition aus $T_O$ zugeordnet sind:

  $$\mathcal{U}_O = \{s \in \mathcal{U} \mid \exists t \in T_O : \mathrm{S}_{\mathcal{U}}(t) = s\}.$$

- $\mathcal{U}_Q$ bezeichnet eine zu $\mathcal{U}_O$ gleichmächtige Menge von Subjekten, die keiner Transition aus $T$ zugeordnet sind.

Die Funktion $\Pi$ fungiert hier wieder als bijektive Abbildung zwischen beiden Mengen. Für jedes Subjekt $s \in \mathcal{U}_O$ gibt es ein Subjekt $\Pi(s) \in \mathcal{U}_Q$.

Die Attribute der Transitionen aus $T^+$ werden wie folgt definiert:

- Für eine Transition $t \in T_O$ bleiben die Attribute unverändert:
  - ${\mathrm{S}_{\mathcal{U}}}^+(t) = \mathrm{S}_{\mathcal{U}}(t)$.
  - $\mathrm{A}^+(t, c) = \mathrm{A}(t, c)$ für alle $c \in \mathcal{C}_{\mathsf{c}}^O$.
  - $\mathrm{G}^+(t) = \mathrm{G}(t)$.
- Für eine Transition $\Pi(t) \in T_Q$ werden die Attribute analog zu denen von $t$ definiert:
  - ${\mathrm{S}_{\mathcal{U}}}^+(\Pi(t)) = \Pi(\mathrm{S}_{\mathcal{U}}(t))$.
  - $\mathrm{A}^+(\Pi(t), \Pi(c)) = \mathrm{A}(t, c)$ für alle $\Pi(c) \in \mathcal{C}_{\mathsf{c}}^Q$.
  - Sei $\mathrm{G}(t) = (\mathsf{p}, C)$. Um das Prädikat von $\Pi(t)$ zu definieren, wird ein neuer Prädikatsbezeichner $\mathsf{q} \in \mathsf{P}_{\mathcal{C}}$ eingeführt, der von keiner Transition aus $T^+$ verwendet wird. Das Prädikat von $\Pi(t)$ ist definiert als:

    $$\mathrm{G}^+(t) = (\mathsf{q}, \Phi(C)).$$

- Die Attribute einer Transition $t \in T^+ \setminus (T_O \cup T_Q)$ werden dermaßen erweitert, dass Marken aus $\mathcal{C}_{\mathsf{c}}^Q$ analog zu ihren Gegenstücken aus $\mathcal{C}_{\mathsf{c}}^O$ behandelt werden. Die Zuordnung der Subjekte bleibt unberührt:
  - ${\mathrm{S}_{\mathcal{U}}}^+(t) = \mathrm{S}_{\mathcal{U}}(t)$.
  - $\mathrm{A}^+(t, c) = \begin{cases} \mathrm{A}(t, d) & \text{falls} \quad \exists d \in \mathcal{C}_{\mathsf{c}}^O : \Pi(c) = d. \\ \mathrm{A}(t, c) & \text{andernfalls.} \end{cases}$
  - Sei $\mathrm{G}(t) = (\mathsf{p}, C)$. Das Prädikat wird erweitert, um auch verarbeitete Marken aus $\mathcal{C}_{\mathsf{c}}^Q$ zu berücksichtigen:

    $$\mathrm{G}^+(t) = (\mathsf{p}, C \cup (\Phi(C) \setminus C)).$$

### Auszeichnung

Die durch MultiInstanceLabeler umgesetzte Sicherheitsanforderung verlangt die Isolation von nebenläufigen Teilprozessen, die im erzeugten Modell $N^+$ durch die Teilnetze $O$ und $Q$ modelliert werden. MultiInstanceLabeler definiert Isolation als die Abwesenheit von Informationsflüssen zwischen den Teilprozessen, d.h. es darf keinen Informationsaustausch zwischen $O$ und $Q$ geben.

Zu diesem Zweck werden die Transitionen *eines* Teilprozesses (in diesem Fall $O$) sowie die im Anfangszustand in diesem Teilprozess vorhandenen Marken als `High` klassifiziert. Die Subjekte, in deren Namen diese Transitionen ausgeführt werden, erhalten entsprechend die Freigabe `High`.

Die Transitionen und Marken des anderen Teilprozesses (hier $Q$) werden mit `Low` ausgezeichnet und die ausführenden Subjekte erhalten die Freigabe `Low`. Da beide Teilprozesse per Konstruktion dasselbe Verhalten aufweisen, ist es ausreichend, den Informationsfluss zwischen den Teilprozessen in nur einer Richtung zu untersuchen.[6]

Alle weiteren Transitionen, die weder Teil $O$ noch von $Q$ sind, erhalten ebenfalls die Klassifikation `Low`. Dasselbe gilt für Marken, die im Anfangszustand eine Stelle aus der Menge $P^+ \setminus (P_O \cup P_Q)$ belegen. Subjekte, in deren Namen Transitionen aus der Menge $T^+ \setminus (T_O \cup T_Q)$ ausgeführt werden, erhalten die Freigabe `neutral`.

Entsprechend werden im Folgenden die Abbildungen $\textsc{Label}_{\mathcal{SC}}$ und $\textsc{Clear}_{\mathcal{SC}}$ definiert.

- Für eine Transition $t \in T^+$ gilt:

$$\textsc{Label}_{\mathcal{SC}}(t) = \begin{cases} \texttt{High} & \text{falls} \quad t \in T_O. \\ \texttt{Low} & \text{falls} \quad t \in T_Q. \\ \texttt{Low} & \text{falls} \quad t \in T^+ \setminus (T_O \cup T_Q). \end{cases}$$

- Für eine gefärbte Marke $c \in \mathcal{C}_{\mathsf{c}}$ gilt:

$$\textsc{Label}_{\mathcal{SC}}(c) = \begin{cases} \texttt{High} & \text{falls} \quad \exists p \in P_O : c \in M^+(p). \\ \texttt{Low} & \text{falls} \quad \exists p \in P_Q : c \in M^+(p). \\ \texttt{Low} & \text{falls} \quad \exists p \in P^+ \setminus (P_O \cup P_Q) : c \in M^+(p). \end{cases}$$

- Für ein Subjekt $s \in \mathcal{U}$ gilt:

$$\textsc{Clear}_{\mathcal{SC}}(s) = \begin{cases} \texttt{High} & \text{falls} \quad \exists t \in T_O : \mathrm{S}_{\mathcal{U}}(t) = s. \\ \texttt{Low} & \text{falls} \quad \exists t \in T_Q : \mathrm{S}_{\mathcal{U}}(t) = s. \\ \texttt{neutral} & \text{falls} \quad \exists t \in T^+ \setminus (T_O \cup T_Q) : \mathrm{S}_{\mathcal{U}}(t) = s. \end{cases}$$

[6] Dieselbe Auszeichnungsstrategie kann verwendet werden, um die Isolation zweier beliebiger Teilprozesse – die nicht notwendigerweise dasselbe Verhalten haben – zu untersuchen. Allerdings müsste in diesem Fall die Möglichkeit von Informationsflüssen in beiden Richtungen, d.h. vom einen Teilprozess zum anderen *und umgekehrt*, betrachtet werden.

### Anfangszustand

Ausgehend vom Anfangszustand $M$ des Ausgangsnetzes $N$ wird für $N^+$ der Anfangszustand $M^+$ definiert. In diesem bleibt die Belegung der Stellen, die bereits Teil des Ausgangsnetzes $N$ sind, unverändert. Stellen aus $P_Q$ (d.h. Stellen, die neu hinzugekommen sind) werden mit Marken belegt, sofern ihr jeweiliges Gegenstück aus $P_O$ mit Marken belegt ist. In diesem Fall wird durch den Operator $\Phi$ eine entsprechende Markenmenge erzeugt, in der jede Marke aus $\mathcal{C}_{\mathsf{c}}^O$ durch ihr Gegenstück aus $\mathcal{C}_{\mathsf{c}}^Q$ ersetzt wird. Zusammengefasst ist die Belegung einer Stelle $p \in P^+$ (und in ihrer Gesamtheit der Anfangszustand $M^+$ von $N^+$) definiert als:

$$M^+(p) = \begin{cases} M(p) & \text{falls} \quad p \in P. \\ \Phi(M(q)) & \text{falls} \quad \exists q \in P : \Pi(q) = p. \end{cases}$$

### Konformität des erzeugten Netzes

Um sicherzustellen, dass das durch MultiInstanceLabeler aus einem IFnet $N$ erzeugte Netz $N^+$ für die Zertifizierung mit InDico verwendet werden kann, wird gezeigt, dass es sich bei $N^+$ ebenfalls um ein IFnet handelt.

**Beweis:** Nach Konstruktion entsprechen die Elemente des Tupels $N^+ = ((P^+, T^+, F^+, \text{In}^+, \text{Out}^+), \text{S}_\mathcal{U}{}^+, \text{A}^+, \text{G}^+, \text{Label}_\mathcal{SC}{}^+, \text{Clear}_\mathcal{SC}{}^+)$ den Mengen und Funktionen, die für die Beschreibung eines IFnets gebraucht werden, d.h. $P^+$ ist eine Menge von Stellen, $T^+$ ist eine Menge von Transitionen usw. Es muss darüber hinaus noch gezeigt werden, dass $N^+$ eine Menge von Anfangsstellen besitzt und dass die Knoten verbunden sind:

- Die Menge $P_I^+ \subset P^+$ der Anfangsstellen von $N^+$ ist definiert als:

  $$P_I \cup \{\Pi(p) \mid p \in P_I \cap P_O\}.$$

  Da die Mengen $P_I$ und $P_O$ nach Voraussetzung nicht leer sind, ist auch die Menge $P_I^+$ nicht leer (sie umfasst mindestens zwei Elemente). Da jede Stelle $p \in P_I$ eine Anfangsstelle ist (d.h. sie besitzt keine Eingangsknoten und kann nur schwarze Marken aufnehmen), ist nach Konstruktion auch jedes Bild $\Pi(p)$ von $p$ eine Anfangsstelle.

- Um die Verbundenheit von $N^+$ nachzuweisen, muss gezeigt werden, dass es für jeden Knoten $x \in P^+ \cup T^+$ eine Anfangsstelle $i \in P_I^+$ gibt, so dass $i$ mit $x$ durch einen Pfad verbunden ist. Da die Verbundenheit der Teilmenge $(P \cup T)$ von Knoten, die aus dem Ursprungsnetz $N$ übernommen worden sind, durch die Erweiterung nicht beeinträchtigt wird $(F \subset F^+)$, muss nur noch die Verbundenheit der neu hinzugekommenen Knoten der Menge $T_Q \cup T_O$ gezeigt werden:

  Für einen Knoten $y \in P_O \cup T_O$ sei ein Pfad von einer Anfangsstelle $i \in P_I$ zu $y$ durch die Knotenfolge $\{k_0, k_1, ..., k_n\}$ gegeben, wobei $k_0 = i$ und $k_n = y$.

Darauf aufbauend lässt sich für einen Knoten $\Pi(y) \in P_Q \cup T_Q$ die Knotenfolge $\{\Pi(k_0), \Pi(k_1), ..., \Pi(k_n)\}$ konstruieren. Da $k_r F^+ k_{r+1}$ für alle $r \in \{0, ..., n-1\}$ gilt, gilt nach Konstruktion auch $\Pi(k_r) F^+ \Pi(k_{r+1})$, d.h. die Knotenfolge ist ein Pfad von $\Pi(k_0) = \Pi(i)$ nach $\Pi(k_n) = \Pi(y)$. Weil $i$ eine Anfangsstelle ist, ist – wie oben gezeigt – auch $\Pi(i)$ eine Anfangsstelle von $N^+$. Da jeder Knoten aus $P_Q \cup T_Q$ das $\Pi$-Abbild eines Knotens aus $P_O \cup T_O$ ist, ist die Verbundenheit von $N^+$ damit gezeigt. $\dashv$

## 3.5 Prüfung von IFnet-Modellen

Die statische Prüfung eines IFnet-Modells dient dazu herauszufinden, ob es während der Laufzeit des Prozesses zu einer Verletzung der durch die Sicherheitsstufen ausgedrückten Isolationsanforderung kommen kann. Eine solche Verletzung liegt vor, wenn ein Subjekt mit niedrigerer Freigabe (`Low`) an höher klassifizierte Informationen (`High`) gelangen kann – entweder durch einen direkten Zugriff auf ein Datenobjekt oder mittels verdeckter Informationsübertragung.

Zu diesem Zweck definiert InDico eine Menge von Informationsflusskriterien an ein IFnet, die die Abwesenheit illegaler Informationsübertragung (von `High` nach `Low`) charakterisieren. Entsprechend den von InDico betrachteten Informationskanälen setzt sich diese Menge aus zwei Teilen zusammen:

1. **Informationsflusskriterien für Datenfluss:** InDico definiert drei Informationsflusskriterien bzgl. des Datenflusses, die unterbinden, dass Subjekte mit `Low`-Freigabe auf `High`-klassifizierte Datenobjekte zugreifen und `High`-klassifizierte Informationen in `Low`-klassifizierte Datenobjekte übertragen werden können. Konzeptionell orientieren sich die Kriterien an den Zugriffsregeln der Modelle von Denning (1976) und Bell und LaPadula (1976), wobei diese erweitert werden, um das Verhalten von Subjekten mit der Freigabe `neutral` zu berücksichtigen.
2. **Informationsflusskriterien für verdeckte Übertragung:** Für die Erkennung verdeckter Informationsübertragung definiert InDico drei Informationsflusskriterien. Diese charakterisieren Ausführungsmöglichkeiten, bei denen der Ablauf eines Prozesses mit der Freigabe `Low`[7] durch `High`-klassifizierte Informationen beeinflusst werden kann. Ausgangspunkt für die Definition dieser Kriterien ist die Arbeit von Busi und Gorrieri (2009), in der ähnliche Kriterien für klassische Petri-Netze entwickelt werden. Die in InDico eingeführten Kriterien unterscheiden sich stark von diesen, da sie die zusätzlichen Möglichkeiten von IFnets berücksichtigen, insbesondere die Existenz unterscheidbarer Marken und Subjekten mit `neutral`-Freigabe sowie der Informationsübertragung durch bedingte Verzweigungen. Darüber hinaus wird für jedes dieser drei Kriterien eine abgeschwächte Variante angegeben, die weniger restriktiv ist und bestimmte Sicherheitsanforderungen präziser erfasst. Die Korrespondenz der Kriterien zu einer formalen Noninterferenz-Eigenschaft wird in Kapitel 5 gezeigt.

Im Folgenden werden die Informationsflusskriterien formal definiert. Die Algorithmen zu ihrer Prüfung, d.h. zur Entscheidung, ob ein IFnet-Modell sie erfüllt, finden sich in Kapitel 4. Die Definition der Informationsflusskriterien ist in allen Fällen zweigeteilt: Zunächst wird für ein Informationsflusskriterium eine Menge definiert, die alle sogenannten „kritischen Bereiche“ (bzgl. des jeweiligen Informationsflusskriteriums) in einem IFnet

[7] Hier ist ein Prozess gemeint, der im Namen von Subjekten mit der Freigabe `Low` ausgeführt wird. Diese sprachliche Vereinfachung wird im Folgenden an verschiedenen Stellen benutzt, wenn die Bedeutung unmissverständlich ist.

enthält. Die Zugehörigkeit zu einer solche Menge kann allein durch die Betrachtung der Struktur des IFnets – also der Transitionen, Stellen und der Flussrelation – entschieden werden, ohne dass die Zustände betrachtet werden müssen. Ein solcher kritischer Bereich beschreibt einen Teil des betrachteten IFnets, in dem eine Verletzung des Informationsflusskriteriums auftreten *könnte*. Diese Mengen werden mit STR* bezeichnet, wobei der Stern durch die Bezeichnung für das jeweilige Informationsflusskriterium ersetzt wird. Anschließend wird das eigentliche Informationsflusskriterium definiert, das auch die Zustände eines Netzes berücksichtigt.

Der Grund für diese Zweiteilung ist ein verminderter Prüfungsaufwand, da die Kosten für die rein strukturelle Untersuchung eines IFnets geringer sind als für die zusätzliche Betrachtung seiner Zustände. Stellt sich im ersten Schritt heraus, dass in einem IFnet keine kritischen Bereiche existieren, kann die Prüfung (bzgl. dieses Informationsflusskriteriums) bereits abgebrochen werden. Andernfalls kann sich die Untersuchung des Zustandsraums auf die kritischen Bereiche beschränken, da nur hier Verletzungen auftreten können.

### 3.5.1 Informationsflusskriterien für Datenfluss

Im Sicherheitsmodell von InDico darf ein Subjekt $s \in \mathcal{U}$ auf alle Datenobjekte zugreifen, deren Klassifikation gleich oder niedriger als die Freigabe von $s$ ist. Infolgedessen werden Lesezugriffe auf höher klassifizierte Ressourcen von der statischen Prüfung als Verletzung gewertet. Darüber hinaus ist es unzulässig, wenn das Subjekt schreibend auf eine Ressource zugreift, deren Einstufung niedriger als $\textsc{Clear}_{\mathcal{SC}}(s)$ ist, da derartige Ressourcen auch für Subjekte mit einer entsprechend niedrigeren Freigabe zugänglich sind. Für Subjekte mit der Freigabe `neutral` wird eine besondere Regel eingeführt, die erkennt, wenn Information aus `High`-Eingabeobjekten möglicherweise in niedriger eingestufte Ausgaberessourcen übertragen werden.

Die folgenden drei Anforderungen fassen diesen Sachverhalt zusammen:

- **Leseregel (NoRead):** In keinem erreichbaren Zustand darf ein Subjekt mit einer `Low`-Freigabe lesend auf ein `High`-klassifiziertes Datenobjekt zugreifen.
- **Schreibregel für Subjekte mit `High`-Freigabe (NoWriteHigh):** In keinem erreichbaren Zustand darf ein Subjekt mit einer `High`-Freigabe schreibend auf ein `Low`-klassifiziertes Datenobjekt zugreifen.
- **Schreibregel für Subjekte mit `neutral`-Freigabe (NoWriteNeutral):** In keinem erreichbaren Zustand darf ein Subjekt mit der Freigabe `neutral` schreibend auf ein `Low`-klassifiziertes Datenobjekt zugreifen und gleichzeitig lesenden Zugriff auf ein (relativ) höher klassifiziertes Datenobjekt haben.

### NoRead

Im Fall der Informationsflusskriterien für Datenfluss ist die Identifikation kritischer Bereiche relativ einfach, weil nur die Transitionen für sich betrachtet werden müssen und nicht ihr Zusammenhang mit weiteren Teilen des IFnets. Die Menge $\mathsf{STR}_{\text{NoRead}}$ enthält die Transitionen, die potentiell die Leseregel verletzen können, und ist wie folgt definiert.

**Definition 3.22** ($\mathsf{STR}_{\text{NoRead}}$)**.** *Die Menge* $\mathsf{STR}_{\text{NoRead}}$ *umfasst alle Transitionen, die die Leseregel verletzen können. Eine Transition* $t \in T$ *ist Element von* $\mathsf{STR}_{\text{NoRead}}$ *genau dann, wenn:*

$$\begin{aligned}&\textsc{Clear}_{\mathcal{SC}}(\mathrm{S}_{\mathcal{U}}(t)) = \texttt{Low} \wedge \\ &\exists i \in \mathbb{N}, \exists c \in \textsc{InputSet}(t,i) : \\ &\quad \{read\} \subseteq \mathrm{A}(t,c).\end{aligned}$$

⊣

Das NoRead-Kriterium greift die Definition von $\mathsf{STR}_{\text{NoRead}}$ auf und prüft für jedes seiner Elemente, ob es einen Zustandsübergang geben kann, in dem die Leseregel verletzt wird.

**Definition 3.23** (NoRead**-Kriterium**)**.** *Ein markiertes IFnet* $(N, M)$ *mit Anfangszustand* $M$ *erfüllt das* NoRead*-Kriterium genau dann, wenn für alle erreichbaren Zustände* $M_1 \in [M]$ *gilt:*

$$\begin{aligned}&\forall t \in \mathsf{STR}_{\text{NoRead}}, \forall M_1, M_2 \in [M] : \\ &\quad M_1 \xrightarrow{t_i} M_2 \Rightarrow \\ &\qquad \forall c \in \textsc{InputSet}(t,i) : \{read\} \subseteq A(t,c) \Rightarrow \textsc{Label}_{\mathcal{SC}}(c) \leq \textsc{Clear}_{\mathcal{SC}}(\mathrm{S}_{\mathcal{U}}(t)).\end{aligned}$$

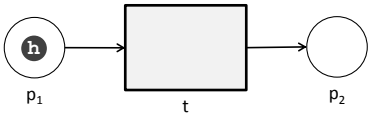
⊣

Abbildung 3.9(a) zeigt beispielhaft einen Zustand in einem IFnet-Fragment, von dem aus das NoRead-Kriterium verletzt werden kann: Die Transition $t$, die von Subjekt LowUser mit niedrigerer Freigabe ausgeführt wird, ist in dem gezeigten Zustand schaltbereit und greift im Fall des Schaltens lesend auf das höher eingestufte Datenobjekt `h` zu.

### NoWriteHigh

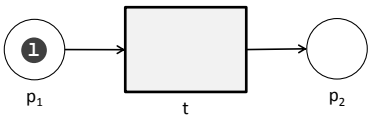
Für die Schreibregel für Subjekte mit `High`-Freigabe wird die Menge $\mathsf{STR}_{\text{NoWriteHigh}}$ definiert, die die Transitionen enthält, durch die eine Verletzung auftreten kann.

**Definition 3.24** ($\mathsf{STR}_{\text{NoWriteHigh}}$)**.** *Die Menge* $\mathsf{STR}_{\text{NoWriteHigh}}$ *umfasst alle Transitionen, die die Schreibregel für Subjekte mit* `High`*-Freigabe verletzen können. Eine Transition* $t \in T$ *ist Element von* $\mathsf{STR}_{\text{NoWriteHigh}}$ *genau dann, wenn:*



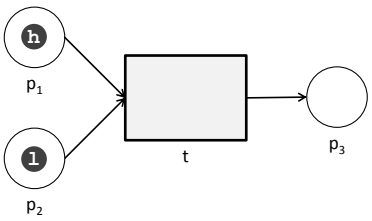

$\mathrm{S}_{\mathcal{U}}(t) = \mathsf{LowUser}$, $\textsc{Clear}_{\mathcal{SC}}(\mathsf{LowUser}) = \mathtt{Low}$
$\textsc{Label}_{\mathcal{SC}}(\mathtt{h}) = \mathtt{High}$
$\textsc{In}(t, p_1) = [\{\mathtt{h}\}]$
$\mathrm{A}(t, \mathtt{h}) = \{read\}$

(a) NoRead




$\mathrm{S}_{\mathcal{U}}(t) = \mathsf{HighUser}$, $\textsc{Label}_{\mathcal{SC}}(\mathsf{HighUser}) = \mathtt{High}$
$\textsc{Label}_{\mathcal{SC}}(\mathtt{l}) = \mathtt{Low}$
$\textsc{In}(t, p_1) = [\{\mathtt{l}\}]$
$\mathrm{A}(t, \mathtt{l}) = \{write\}$

(b) NoWriteHigh




$\mathrm{S}_{\mathcal{U}}(t) = \mathsf{NeutralUser}$, $\textsc{Clear}_{\mathcal{SC}}(\mathsf{NeutralUser}) = \mathtt{neutral}$
$\textsc{Label}_{\mathcal{SC}}(\mathtt{h}) = \mathtt{High}$, $\textsc{Label}_{\mathcal{SC}}(\mathtt{l}) = \mathtt{Low}$
$\textsc{In}(t, p_1) = [\{\mathtt{h}\}]$
$\textsc{In}(t, p_2) = [\{\mathtt{l}\}]$
$\mathrm{A}(t, \mathtt{h}) = \{read\}$, $\mathrm{A}(t, \mathtt{l}) = \{write\}$

(c) NoWriteNeutral

Abbildung 3.9.: Beispiele für mögliche Verletzungen der Datenflusskriterien.

$$\begin{aligned}&\textsc{Clear}_{\mathcal{SC}}(\mathrm{S}_{\mathcal{U}}(t)) = \mathtt{High} \wedge \\ &\exists i \in \mathbb{N}, \exists c \in \textsc{InputSet}(t, i) : \\ &\quad \{write\} \subseteq \mathrm{A}(t, c).\end{aligned}$$ ⊣

Das NoWriteHigh-Kriterium greift die Definition auf und prüft für jedes seiner Elemente, ob eine Verletzung tatsächlich eintreten kann.

**Definition 3.25 (NoWriteHigh-Kriterium).** *Ein markiertes IFnet $(N, M)$ mit Anfangszustand $M$ erfüllt das* NoWriteHigh*-Kriterium genau dann, wenn:*

$$\begin{aligned}&\forall t \in \mathsf{STR}_{\textsc{NoWriteHigh}}, \forall M_1, M_2 \in [M] : \\ &\quad M_1 \xrightarrow{t_i} M_2 \Rightarrow \\ &\qquad \forall c \in \textsc{InputSet}(t, i) : \{write\} \subseteq A(t, c) \Rightarrow \textsc{Clear}_{\mathcal{SC}}(\mathrm{S}_{\mathcal{U}}(t)) \leq \textsc{Label}_{\mathcal{SC}}(c).\end{aligned}$$ ⊣

Abbildung 3.9(b) zeigt beispielhaft einen Zustand in einem IFnet-Fragment, von dem aus das NoWriteHigh-Kriterium verletzt werden kann: Die Transition $t$, die von Subjekt HighUser mit höherer Freigabe ausgeführt wird, ist in dem gezeigten Zustand schaltbereit und greift im Fall des Schaltens schreibend auf das niedriger eingestufte Datenobjekt h zu.

### NoWriteNeutral

Für die Schreibregel für Subjekte mit `neutral`-Freigabe wird die Menge $\mathsf{STR}_{\text{NoWriteNeutral}}$ definiert, die die Transitionen mit `neutral`-Freigabe enthält, die eine Verletzung herbeiführen können.

**Definition 3.26** ($\mathsf{STR}_{\text{NoWriteNeutral}}$)**.** *Die Menge* $\mathsf{STR}_{\text{NoWriteNeutral}}$ *umfasst alle Transitionen, die die Schreibregel für Subjekte mit* `neutral`*-Freigabe verletzen können. Eine Transition* $t \in T$ *ist Element von* $\mathsf{STR}_{\text{NoWriteNeutral}}$ *genau dann, wenn:*

$$
\begin{aligned}
&\textsc{Clear}_{\mathcal{SC}}(\mathrm{S}_{\mathcal{U}}(t)) = \texttt{neutral} \wedge \\
&\exists i \in \mathbb{N}, \exists c \in \textsc{InputSet}(t,i), \exists d \in \textsc{InputSet}(t,i) \cup \textsc{OutputSet}(t,i) : \\
&\quad c \neq d \wedge \\
&\quad \{read\} \subseteq \mathrm{A}(t,c) \wedge \\
&\quad \{write\} \subseteq \mathrm{A}(t,d).
\end{aligned}
$$

⊣

In der Beschreibung des NoWriteNeutral-Kriteriums wird diese Definition aufgegriffen, um zu entscheiden, ob eine Verletzung eintritt.

**Definition 3.27** (NoWriteNeutral**-Kriterium).** *Ein markiertes IFnet* $(N, M)$ *mit Anfangszustand* $M$ *erfüllt das* NoWriteNeutral*-Kriterium genau dann, wenn:*

$$
\begin{aligned}
&\forall t \in \mathsf{STR}_{\text{NoWriteNeutral}}, \forall M_1, M_2 \in [M] : \\
&\quad M1 \xrightarrow{t_i} M2 \Rightarrow \\
&\qquad \forall c_1 \in \textsc{InputSet}(t,i), \forall c_2 \in \textsc{InputSet}(t,i) \cup \textsc{OutputSet}(t,i) : \\
&\qquad\quad \{read\} \subseteq A(t,c_1) \wedge \{write\} \subseteq A(t,c_2) \Rightarrow \\
&\qquad\qquad \textsc{Label}_{\mathcal{SC}}(c_1) \leq \textsc{Label}_{\mathcal{SC}}(c_2).
\end{aligned}
$$

⊣

Abbildung 3.9(c) zeigt beispielhaft einen Zustand in einem IFnet-Modell, von dem aus das NoWriteNeutral-Kriterium verletzt werden kann: Die Transition $t$, die von einem Subjekt NeutralUser mit der Freigabe `neutral` ausgeführt wird, ist in dem gezeigten Zustand mit den Eingangsmarken `l` und `h` schaltbereit. Im Fall des Schaltens wird lesend auf die Marke `h` (die höher eingestuft ist) zugegriffen, und schreibend auf die Marke `l` (die niedriger eingestuft ist). Es kann damit zu einem Informationstransfer von der höher eingestuften zur niedriger eingestuften Ressource kommen.

### 3.5.2 Informationsflusskriterien für verdeckte Übertragung

Neben den Datenobjekten kann auch das Verhalten eines Prozesses Informationen übermitteln, die vor unbefugter Einsichtnahme geschützt werden müssen. InDico definiert eine Menge von Kriterien an ein IFnet-Modell, mit denen entschieden werden kann, ob die Ausführung höher klassifizierter Aktivitäten Einfluss auf die Ausführung von Aktivitäten mit niedrigerer Freigabe hat. Sofern es in einem IFnet-Modell zu keiner derartigen Einflussnahme kommen kann, ist sichergestellt, dass Subjekte mit niedrigerer Freigabe auf diesem Wege keine Informationen bzgl. höher klassifizierter Prozess-Aktivitäten erhalten.

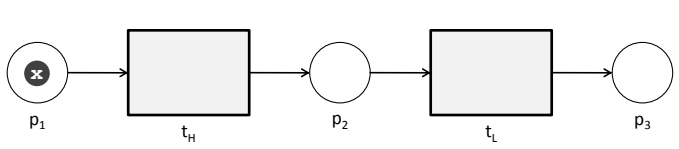
Aktivitäten in einem Geschäftsprozess beeinflussen sich, wenn Kontrollfluss-Abhängigkeiten bestehen, d.h. wenn die Ausführung einer Aktivität Voraussetzung für die Ausführung einer anderen Aktivität ist, oder aber die Ausführung einer anderen Aktivität verhindert. Es liegt eine Isolationsverletzung vor, wenn das Schalten einer höher klassifizierten Transition Einfluss auf das Schalten einer Transition mit niedrigerer Freigabe nehmen kann. Dies kommt in den folgenden Fällen vor:

- **Kausalität** (StrongCausal, WeakCausal)**:** Das Schalten einer höher klassifizierten Transition ist Voraussetzung für das Schalten einer Transition mit niedrigerer Freigabe. Aus der Ausführung der Transition mit niedrigerer Freigabe kann der entsprechende Nutzer auf die vorhergehende Ausführung der höher klassifizierten Transition schließen.
- **Nutzungskonflikt** (StrongUsage, WeakUsage)**:** Eine höher klassifizierte Transition und eine Transition mit niedrigerer Freigabe konkurrieren um dieselbe Marke. Schaltet die höher klassifizierte Transition (und konsumiert damit die entsprechende Marke), kann der Nutzer mit niedrigerer Freigabe diese Tatsache erschließen, da die Ausführung seiner Transition blockiert wird.

Darüber hinaus werden Fälle betrachtet, in denen der Ablauf eines Prozesses durch Datenobjekte beeinflusst wird. Diese treten bei Verzweigungen im Prozessablauf auf, in denen die Wahl eines Pfades möglicherweise von dem Wert eines Datums abhängt. Hier besteht die Möglichkeit einer Isolationsverletzung, wenn die Ausführung einer Transition mit niedrigerer Freigabe von einem höher-klassifizierten Datenobjekt abhängt:

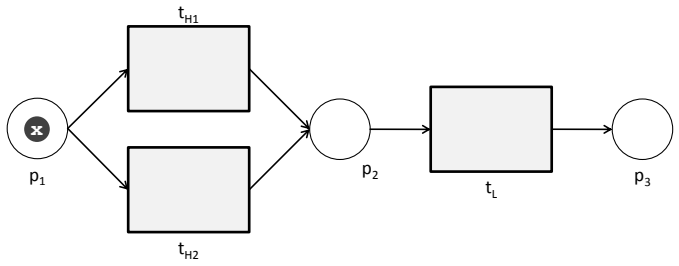
- **Bedingte Ausführung** (StrongCond, WeakCond)**:** Der Ausführung einer Transition mit niedrigerer Freigabe geht die Ausführung einer Transition voraus, der ein Prädikat zugeordnet ist, das ein höher klassifiziertes Datenobjekt benutzt. Aus der Ausführung der Transition mit niedrigerer Freigabe kann der entsprechende Nutzer schließen, dass das Prädikat wahr ist. Er erhält somit eine Information über das höher klassifizierte Datenobjekt.

Im Folgenden wird für jeden dieser Fälle ein Kriterium an ein IFnet definiert, das die Möglichkeit einer ungewollten Informationsübertragung auf dem beschriebenen Weg ausschließt.




$\textsc{Clear}_{\mathcal{SC}}(\mathrm{S}_{\mathcal{U}}(t_L)) <$
$\textsc{Label}_{\mathcal{SC}}(t_H)$
$\textsc{In}(t_H, p_1) = \textsc{Out}(t_H, p_2) = [\{\mathtt{x}\}]$
$\textsc{In}(t_L, p_2) = \textsc{Out}(t_L, p_3) = [\{\mathtt{x}\}]$

(a) StrongCausal




$\textsc{Clear}_{\mathcal{SC}}(\mathrm{S}_{\mathcal{U}}(t_L)) <$
$\textsc{Label}_{\mathcal{SC}}(t_{H1}), \textsc{Label}_{\mathcal{SC}}(t_{H2})$
$\textsc{In}(t_{H1}, p_1) = \textsc{In}(t_{H2}, p_1) = [\{\mathtt{x}\}]$
$\textsc{Out}(t_{H1}, p_2) =$
$\textsc{Out}(t_{H2}, p_2) = [\{\mathtt{x}\}]$
$\textsc{In}(t_L, p_2) = \textsc{Out}(t_L, p_3) = [\{\mathtt{x}\}]$

(b) WeakCausal

Abbildung 3.10.: Informationsfluss durch eine kausale Abhängigkeit.

### Kausalität

Abbildung 3.10 zeigt beispielhaft zwei IFnet-Fragmente, in denen es zu einer illegalen Informationsübertragung durch abhängige Transitionen kommen kann. Das Fragment in Abbildung 3.10(a) enthält zwei Transitionen $t_H$ und $t_L$. $t_H$ ist höher klassifiziert (`High`) und $t_L$ wird von einem Benutzer mit `Low`-Freigabe ausgeführt. Damit $t_L$ schalten kann, muss zuvor $t_H$ ausgeführt worden sein ($t_H$ belegt die Stelle $p_2$ mit einer Marke, die $t_L$ zum Schalten benötigt). Durch die Ausführung erfährt der Nutzer von $t_L$, dass zuvor eine höher klassifizierte Transition ausgeführt wurde.

Um eine derartige Situation in einem IFnet zu erkennen, wird zunächst die Menge $\mathsf{STR}_{\textsc{Causal}}$ der „kritischen Transitionenfolgen“ eingeführt. Eine kritische Transitionenfolge ist eine durch Kanten verbundene Knotenmenge, an deren Anfang eine `High`-klassifizierte Transition steht und an deren Ende eine Transition mit `Low`-Freigabe, während (etwaig) dazwischenliegende Transitionen die Freigabe `neutral` besitzen.

**Definition 3.28** ($\mathsf{STR}_{\textsc{Causal}}$). *In einem IFnet $N$ gehört eine Knotenmenge $t_0, ..., t_n \cup p_0, ..., p_{n-1} \subseteq T \cup P$ zur Menge $\mathsf{STR}_{\textsc{Causal}}$ der kritischen Transitionenfolgen, wenn gilt:*

- $p_i \in t_i \bullet \cap \bullet t_{i+1}$ *für alle* $i \in \{0, ..., n-1\}$,
- $\textsc{Label}_{\mathcal{SC}}(t_0) = \mathtt{High}$,
- $\textsc{Label}_{\mathcal{SC}}(t_i) = \mathtt{Low}$ *für alle* $i \in \{0, ..., n-1\} \setminus \{0\}$,
- $\textsc{Clear}_{\mathcal{SC}}(\mathrm{S}_{\mathcal{U}}(t_n)) = \mathtt{Low}$ *und*
- $\textsc{Clear}_{\mathcal{SC}}(\mathrm{S}_{\mathcal{U}}(t_i)) = \mathtt{neutral}$ *für alle* $i \in \{0, ..., n-1\} \setminus \{0, n-1\}$. ⊣

In einer kritischen Transitionenfolge kann es nur dann zu einer ungewollten Informationsübertragung kommen, wenn die Ausführung der ersten Transition tatsächlich Voraussetzung für die Ausführung der letzten Transition ist, d.h. wenn es in dem betreffenden IFnet eine Schaltsequenz gibt, in der die letzte Transition der kritischen Folge nur dann feuern kann, wenn zuvor deren erste Transition gefeuert hat. Um ein entsprechendes Kriterium für derartige Situation zu fassen, wird zunächst ein Begriff der Abhängigkeit zwischen zwei Transitionen definiert. In der ersten Form wird dieser zwischen zwei unmittelbar benachbarten Transitionen definiert.

**Definition 3.29 (Abhängigkeit benachbarter Transitionen).** *Seien $N$ ein IFnet mit Anfangszustand $M$, $t_1, t_2 \in T$ und $p \in P$ mit $p \in t_1 \bullet \cap \bullet t_2$. Das Schalten von $t_1$ ist Voraussetzung für das Schalten von $t_2$ (über Stelle $p$), genau dann, wenn es Zustände $M_1, M_2 \in [M]$ gibt, so dass:*

- *$M_1 \xrightarrow{t_1} M' \xrightarrow{\sigma} M_2 \xrightarrow{t_2} M''$ für zwei Zustände $M', M''$ und eine Zustandsfolge $\sigma$,*
- *es gibt eine Marke $c \in (M'(p) \setminus M_1(p))$, die $t_2$ im Zustand $M_2$ in Stelle $p$ zum Schalten benötigt und die während der Schaltfolge $M' \xrightarrow{\sigma} M_2$ nicht in $p$ produziert wird, und*
- *$M_1(p) < I$ für alle $I \in \text{In}(t_2, p)$.*

*Eine solche Abhängigkeit wird als $M_1 \overset{t_1 p t_2}{\rightharpoonup} M_2$ notiert.* ⊣

Die Definition besagt, dass Transition $t_1$ Voraussetzung für die über Stelle $p$ verbundene Transition $t_2$ ist, wenn es eine Schaltfolge gibt, in der $t_1$ Marken in $p$ produziert, die $t_2$ zum Schalten benötigt. Sie wird nun für Transitionen verallgemeinert, die nicht notwendigerweise benachbart sein müssen.

**Definition 3.30 (Abhängigkeit von Transitionen).** *Seien $N$ ein IFnet mit Anfangszustand $M$ und $t_0, ..., t_n \cup p_0, ..., p_{n-1} \subseteq T \cup P$ eine Transitionenfolge mit $p_i \in t_i \bullet \cap \bullet t_{i+1}$ für alle $i \in \{0, ..., n-1\}$. $t_0$ ist Voraussetzung für $t_n$ genau dann, wenn es Zustände $M_0, M_1, ..., M_n, M_{n+1} \in [M]$ gibt, so dass:*

- *$M_0 \xrightarrow{t_0\sigma_0} M_1 \xrightarrow{t_1\sigma_1} M_2 \xrightarrow{t_2\sigma_2} ... \xrightarrow{t_{n-1}\sigma_{n-1}} M_n \xrightarrow{t_n} M_{n+1}$ für Schaltfolgen $\sigma_0, .., \sigma_{n-1}$ und*
- *$M_i \overset{t_i p_i t_{i+1}}{\rightharpoonup} M_{i+1}$ für alle $i \in \{0, ..., n-1\}$.*

*Eine solche Abhängigkeit wird als $M_0 \overset{t_0 p_0 .. t_{n-1} p_{n-1} t_n}{\rightharpoonup} M_n$ notiert.* ⊣

Aufbauend auf den obigen Definition wird nun das StrongCausal-Kriterium definiert, das solche IFnets ausschließt, die „ausführbare“ kritische Transitionenfolgen enthalten, d.h. kritische Transitionenfolgen, in denen das Schalten der ersten Transition Voraussetzung für das Schalten der letzten Transition ist.

**Definition 3.31 (StrongCausal-Kriterium).** *Ein markiertes IFnet $(N, M)$ mit Anfangszustand $M$ erfüllt das* StrongCausal*-Kriterium genau dann, wenn für alle kritischen Transitionenfolgen $t_0, ..., t_n \cup p_0, ..., p_{n-1} \in \mathsf{STR}_{\text{Causal}}$ gilt:*

$$\nexists M_1, M_2 \in [M] : M_1 \overset{t_0 p_0 .. t_{n-1} p_{n-1} t_n}{\rightharpoonup} M_2.$$ ⊣

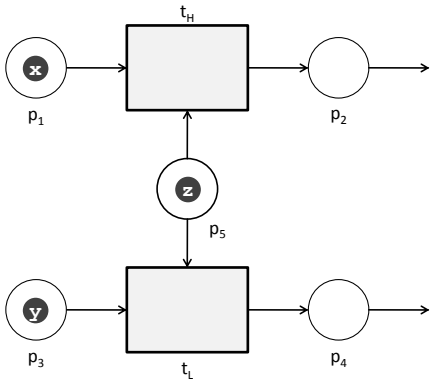
Das StrongCausal-Kriterium ist sehr stark, da es nur von IFnets erfüllt wird, in denen das Schalten einer höher klassifizierten Transition niemals Voraussetzung für das Schalten einer Transition mit niedrigerer Freigabe sein kann.

In vielen Situationen ist es jedoch lediglich notwendig zu verhindern, dass ein Nutzer mit niedrigerer Freigabe ableiten kann, *welche* von mehreren höher klassifizierten Aktivitäten ausgeführt wurde. Abbildung 3.10(b) zeigt ein Beispiel für ein IFnet, welches das StrongCausal-Kriterium klar verletzt, da sowohl $t_{H1}$ als auch $t_{H2}$ Voraussetzung für $t_L$ sind. Allerdings erfährt der Benutzer von $t_L$ mit dem Schalten der Transition nicht, ob $t_{H1}$ oder $t_{H2}$ davor geschaltet hat.

Mit WeakCausal wird ein entsprechend schwächeres Kriterium eingeführt, das lediglich solche IFnets ausschließt, in denen einer Transition mit niedrigerer Freigabe stets *dieselbe* höher klassifizierte Transition vorausgehen muss. Das in Abbildung 3.10(b) gezeigte Netz erfüllt dieses Kriterium (während es StrongCausal verletzt), da es zwei alternative Schaltfolgen gibt, die die Ausführung von $t_L$ beinhalten.

**Definition 3.32 (WeakCausal-Kriterium).** *Ein markiertes IFnet $(N, M)$ mit Anfangszustand $M$ erfüllt das* WeakCausal*-Kriterium genau dann, wenn für alle kritischen Transitionenfolgen $t_0, ..., t_n \cup p_0, ..., p_{n-1} \in \mathsf{STR}_{\text{Causal}}$ gilt:*

$$\exists M_1, M_2 \in [M] : M_1 \overset{t_0 p_0 .. t_{n-1} p_{n-1} t_n}{\rightharpoonup} M_2 \Rightarrow$$
$$\exists M_3 \in [M] : M \xrightarrow{\sigma t_n} M_3 \ \wedge \ t_0 \notin \sigma \text{ für eine Schaltfolge } \sigma.$$ ⊣




$\textsc{Clear}_{\mathcal{SC}}(\mathrm{S}_{\mathcal{U}}(t_L)) < \textsc{Label}_{\mathcal{SC}}(t_H)$
$\textsc{In}(t_H, p_1) = [\{\mathtt{x}\}]$
$\textsc{In}(t_L, p_2) = [\{\mathtt{y}\}]$
$\textsc{In}(t_L, p_5) = \textsc{In}(t_H, p_5) = [\{\mathtt{z}\}]$
$\textsc{Out}(t_H, p_2) = \textsc{Out}(t_L, p_4) = [\{\mathtt{z}\}]$

Abbildung 3.11.: Informationsfluss durch einen Nutzungskonflikt.

### Nutzungskonflikt

Eine andere Art der Beeinflussung (und damit Informationsübertragung) ist möglich, wenn eine höher klassifizierte Transition mit einer Transition, die eine niedrigere Freigabe besitzt, um eine Marke konkurriert, die beide zum Schalten benötigen. Ein solcher Fall tritt in dem in Abbildung 3.11 dargestellten IFnet auf: Sowohl $t_H$ als auch $t_L$ benötigen zum Schalten die Marke $z$ aus der Stelle $p_5$, die für beide eine Eingangsstelle ist. Wenn $t_H$ schaltet, ist $t_L$ blockiert, so dass der Nutzer von $t_L$ die Ausführung von $t_H$ ableiten kann.

Um derartige Situationen in einem IFnet zu erkennen, wird zunächst die Menge $\mathsf{STR}_{\textsc{Usage}}$ der „kritischen Nutzungsstellen“ eingeführt. Eine kritische Nutzungsstelle bezeichnet einen Bereich in einem IFnet, in dem durch das Schalten einer `High`-klassifizierten Transition das Schalten einer Transition mit `Low`-Freigabe möglicherweise verhindert wird, weil beide dieselbe Marke benötigen.

Ein solcher Nutzungskonflikt kann auch „indirekt“ sein. In diesem Fall haben beide Transitionen keine gemeinsame Eingangsstelle. Allerdings ist das Schalten der höherklassifizierten Transition Voraussetzung für das Schalten einer `neutral`-Transition $t_1$, die wiederum eine gemeinsame Eingangsstelle mit einer anderen `neutral`-Transition $t_2$ besitzt und deren Schalten durch einen Nutzungskonflikt verhindern kann. Das Schalten von $t_2$ wiederum ist Voraussetzung für das Schalten der Transition mit niedrigerer Freigabe. Genau wie bei einem „direkten“ Nutzungskonflikt (in dem die beiden Transitionen eine gemeinsame Eingangsstelle besitzen), wird durch das Schalten der `High`-Transition das Schalten der `Low`-Transition verhindert, weil (zwischengeschaltete) Transitionen um dieselbe Marke konkurrieren.

**Definition 3.33** ($\mathsf{STR}_{\text{USAGE}}$). *In einem IFnet $N$ gehört eine Knotenmenge $t_0, ..., t_m, t_{m+1}, .., t_n \cup p_0, ..., p_{m-1}, p_m, p_{m+1}, .., p_{n-1} \subseteq T \cup P$ zur Menge $\mathsf{STR}_{\text{USAGE}}$ der kritischen Nutzungsstellen, wenn gilt:*

- $t_m, t_{m+1} \in p_m\bullet$,
- $p_i \in t_i \bullet \cap \bullet t_{i+1}$ *für alle* $i \in \{0, ..., m-1, m+1, ..., n-1\}$,
- $\text{LABEL}_{\mathcal{SC}}(t_0) = \texttt{High}$,
- $\text{LABEL}_{\mathcal{SC}}(t_i) = \texttt{Low}$ *für alle* $i \in \{0, .., n\} \setminus \{0\}$,
- $\text{CLEAR}_{\mathcal{SC}}(\text{S}_{\mathcal{U}}(t_n)) = \texttt{Low}$ *und*
- $\text{CLEAR}_{\mathcal{SC}}(\text{S}_{\mathcal{U}}(t_i)) = \texttt{neutral}$ *für alle* $i \in \{0, ..., n\} \setminus \{0, n\}$. ⊣

Durch eine kritische Nutzungsstelle kann kann es zu ungewollter Informationsübertragung kommen, wenn – wie in Abbildung 3.11 gezeigt – eine höher klassifizierte Transition die Ausführung einer Transition mit niedrigerer Freigabe blockiert. Um ein entsprechendes Kriterium anzugeben, wird zunächst für benachbarte Transitionen, die eine gemeinsame Eingangsstelle haben, ein Nutzungskonflikt definiert.

**Definition 3.34 (Nutzungskonflikt benachbarter Transitionen).** *Seien $N$ ein IFnet mit Anfangszustand $M$, $p$ eine Stelle von $N$ und $t_1, t_2$ Transitionen mit $p \in \bullet t_1 \cap \bullet t_2$. Zwischen $t_1$ und $t_2$ besteht ein Nutzungskonflikt, genau dann, wenn es Zustände $M_1, M_2 \in [M]$ gibt, so dass:*

- $M_1 \xrightarrow{t_1} M_2$,
- $M_1 \xrightarrow{\sigma} M' \xrightarrow{t_2} M_3$ *für Zustände $M', M_3$ und eine Schaltfolge $\sigma$ und*
- *es gibt eine Marke $c \in (M_1(p) \setminus M_2(p))$, die $t_2$ in Zustand $M'$ in Stelle $p$ zum Schalten benötigt und die während der Schaltfolge $M_1 \xrightarrow{\sigma} M'$ nicht in $p$ produziert wird.*

*Ein solcher Nutzungskonflikt wird als $M_1 \overset{t_1 p t_2}{\times_{uc}} M_2$ notiert.* ⊣

Die Definition besagt, dass zwischen $t_1$ und $t_2$ ein Nutzungskonflikt besteht, wenn es eine Schaltfolge gibt, in der $t_1$ eine Marke aus $p$ entnimmt, die $t_2$ zum Schalten benötigt. Sie wird nun erweitert, um indirekte Nutzungskonflikte einzuschließen, in denen das Schalten einer Transition das Schalten einer anderen Transition verhindert, wobei die beiden Transitionen nicht notwendigerweise benachbart sein müssen.

**Definition 3.35 (Nutzungskonflikt zwischen Transitionen).** *Seien $N$ ein IFnet mit Anfangszustand $M$ und $t_0, ..., t_m, t_{m+1}, .., t_n \cup p_0, ..., p_{m-1}, p_m, p_{m+1}, .., p_{n-1} \subseteq T \cup P$ eine Knotenmenge mit $p_i \in t_i \bullet \cap \bullet t_{i+1}$ für alle $i \in \{0, ..., m-1, m+1, ..., n-1\}$ sowie $t_m, t_{m+1} \in p_m\bullet$. Zwischen $t_1$ und $t_2$ besteht ein Nutzungskonflikt genau dann, wenn es Zustände $M_1, M_3 \in [M]$ gibt, so dass:*

- $M_1 \xrightarrow{t_0\sigma_0} M_2 \xrightarrow{t_m} M_3$ *für einen Zustand* $M_2$ *und eine Schaltfolge* $\sigma_0$,
- $M_1 \overset{t_0 p_0 .. p_{m-1} t_m}{\rightharpoonup} M_3$,
- $M_2 \xrightarrow{\sigma_2} M_4 \xrightarrow{t_{m+1}\sigma_3 t_n} M_5$ *für Zustände* $M_4, M_5$ *und Schaltfolgen* $\sigma_2, \sigma_3$,
- $M_4 \overset{t_{m+1} p_{m+1} .. p_{n-1} t_n}{\rightharpoonup} M_5$ *und*
- $M_2 \overset{t_m p_m t_{m+1}}{\times_{uc}} M_3$.

*Ein solcher Nutzungskonflikt wird als* $M_1 \overset{t_0 p_0 .. t_m p_m t_{m+1} .. p_{n-1} t_n}{\times_{uc}} M_3$ *notiert.* ⊣

Aufbauend auf den obigen Definition wird das StrongUsage-Kriterium definiert, das Netze mit „ausführbaren“ kritischen Nutzungsstellen ausschließt, also solche kritische Nutzungsstellen, in denen eine Ausführung zu einem Nutzungskonflikt zwischen einer höher-klassifizierten Transition und einer Transition mit niedrigerer Freigabe führen kann.

**Definition 3.36 (StrongUsage-Kriterium).** *Ein markiertes IFnet* $(N, M)$ *mit Anfangszustand* $M$ *erfüllt das* StrongUsage*-Kriterium genau dann, wenn für alle kritischen Nutzungsstellen* $t_0, ..., t_m, t_{m+1}, .., t_n \cup p_0, ..., p_{m-1}, p_m, p_{m+1}, .., p_{n-1} \in \mathsf{STR}_{\text{Usage}}$ *gilt:*

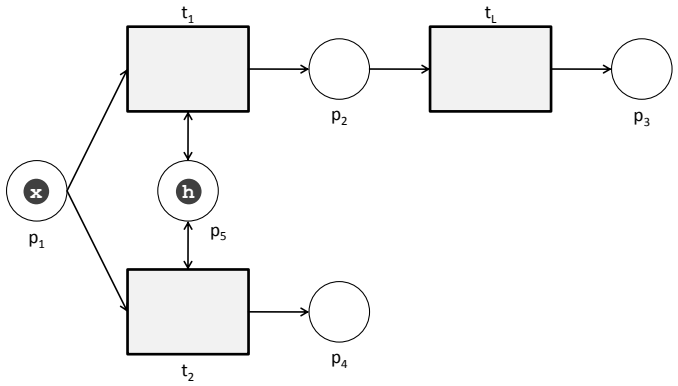
$$\nexists M_1, M_2 \in [M] : M_1 \overset{t_0 p_0 .. t_m p_m t_{m+1} .. p_{n-1} t_n}{\times_{uc}} M_2.$$ ⊣

Es wird mit WeakUsage wiederum eine abgeschwächte Variante des StrongUsage-Kriteriums definiert, die nur solche Netze ausschließt, bei denen der Nutzungskonflikt stets von derselben höher-klassifizierten Transition verursacht wird.

**Definition 3.37 (WeakUsage-Kriterium).** *Ein markiertes IFnet* $(N, M)$ *mit Anfangszustand* $M$ *erfüllt das* WeakUsage*-Kriterium genau dann, wenn für alle kritischen Nutzungsstellen* $t_0, ..., t_m, t_{m+1}, .., t_n \cup p_0, ..., p_{m-1}, p_m, p_{m+1}, .., p_{n-1} \in \mathsf{STR}_{\text{Usage}}$ *gilt:*

$$\begin{aligned}
&\exists M_1, M_2 \in [M] : M_1 \overset{t_0 p_0 .. t_m p_m t_{m+1} .. p_{n-1} t_n}{\times_{uc}} M_2 \Rightarrow \\
&\quad \exists M_3, M_4 \in [M], \exists t' \in p_m\bullet, t' \neq t_0 : \\
&\qquad M \xrightarrow{\sigma} M_3 \xrightarrow{t'} M_4 \ \wedge \\
&\qquad t_0 \notin \sigma \ \wedge \\
&\qquad M_3 \overset{t' p_m t_{m+1} .. p_{n-1} t_n}{\times_{uc}} M_4.
\end{aligned}$$

⊣




$\mathrm{G}(t_1) = (\mathtt{p}, \{\mathtt{h}\})$
$\mathrm{G}(t_2) = (\mathtt{not\_p}, \{\mathtt{h}\})$
$\textsc{Clear}_{\mathcal{SC}}(\mathrm{S}_{\mathcal{U}}(t_L)) < \textsc{Label}_{\mathcal{SC}}(\mathtt{h})$
$\textsc{In}(t_1, p_1) = \textsc{In}(t_2, p_1) = [\{\mathtt{x}\}]$
$\textsc{In}(t_1, p_5) = \textsc{In}(t_2, p_5) = [\{\mathtt{h}\}]$
$\textsc{Out}(t_1, p_5) = \textsc{Out}(t_2, p_5) = [\{\mathtt{h}\}]$
$\textsc{Out}(t_1, p_2) = \textsc{Out}(t_2, p_4) = [\{\mathtt{x}\}]$
$\textsc{In}(t_L, p_2) = [\{\mathtt{x}\}]$
$\textsc{Out}(t_L, p_3) = [\{\mathtt{x}\}]$

Abbildung 3.12.: Informationsfluss durch eine bedingte Ausführung.

### Bedingte Ausführung

Das im Folgenden definierte Informationsflusskriterium betrachtet Fälle, in denen ein Datenobjekt Einfluss auf die Ausführung eines Prozesses nehmen und auf diese Weise Information übertragen könnte. Ein solcher Fall ist exemplarisch in dem in Abbildung 3.12 gezeigten IFnet dargestellt. Hier verzweigt die Ausführung an der Stelle $p_1$ und es wird entweder die Transition $t_1$ oder $t_2$ ausgeführt. Beiden Transitionen ist ein Prädikat zugeordnet, das das (höher klassifizierte) Datenobjekt `h` benutzt. Auf die Ausführung von $t_1$ folgt wiederum die Ausführung der Transition $t_L$, deren Benutzer eine niedrigere Freigabe besitzt. Dieser erhält somit durch die Ausführung von $t_L$ eine Information über das höher klassifizierte Datenobjekt `h`, nämlich, dass das der Transition zugeordnete Prädikat $p$ wahr ist.

Um eine derartige Situation in einem IFnet zu erkennen, wird zunächst die Menge $\mathsf{STR}_{\textsc{Cond}}$ der „kritischen bedingten Transitionenfolgen“ eingeführt. Ein kritische bedingte Transitionenfolge ist eine durch Kanten verbundene Knotenmenge, an deren Anfang eine Transition steht, die mit einem Prädikat ausgezeichnet ist, das eine `High`-klassifizierte Marke benutzt. An deren Ende steht eine Transition mit `Low`-Freigabe.

**Definition 3.38** ($\mathsf{STR}_{\textsc{Cond}}$). *In einem IFnet $N$ gehört eine Knotenmenge $t_0, ..., t_n \cup p_0, ..., p_{n-1} \subseteq T \cup P$ zur Menge $\mathsf{STR}_{\textsc{Cond}}$ der kritischen bedingten Transitionenfolgen, wenn gilt:*

- $p_i \in t_i \bullet \cap \bullet t_{i+1}$ *für alle* $i \in \{0, ..., n-1\}$,
- $\mathrm{G}(t_0) = (p, C)$ *mit* $C \neq \emptyset$ *und*
- $\textsc{Clear}_{\mathcal{SC}}(\mathrm{S}_{\mathcal{U}}(t_n)) = \mathtt{Low}$. ⊣

Durch eine kritische bedingte Transitionenfolge kann es zu ungewollter Informationsübertragung kommen, wenn – wie in Abbildung 3.12 gezeigt – der Ausführung einer Transition mit niedrigerer Freigabe die Ausführung einer Transition vorausgeht, deren Ausführung von einer höher-klassifzierten Marke (Datenobjekt) abhängt. Um ein entsprechendes Kriterium anzugeben, wird zunächst definiert, wann die Ausführung einer Transition von einem Datenobjekte abhängig ist.

**Definition 3.39 (Datenabhängigkeit einer Transition).** *Seien $N$ ein IFnet mit Anfangszustand $M$, $t_0, ..., t_n \cup p_0, ..., p_{n-1} \subseteq T \cup P$ eine Knotenmenge mit $p_i \in t_i \bullet \cap \bullet t_{i+1}$ für alle $i \in \{0, ..., n-1\}$ und $c \in \mathcal{C}_\mathsf{c}$ eine gefärbte Marke. Die Ausführung $t_n$ hängt von dem Datum $c$ ab genau dann, wenn es Zustände $M_1, M_2 \in [M]$ gibt, so dass:*

- $M_1 \xrightarrow{t_0 p_0 .. p_{n-1} t_n} M_2$ *und*
- *für* $\mathrm{G}(t_0) = (\mathsf{p}, C)$ *ist* $c \in C$.

*Eine solche Datenabhängigkeit wird als* $M_1 \xrightarrow{t_0 p_0 .. p_{n-1} t_n}_c M_2$ *notiert.* ⊣

Das StrongCond-Kriterium schließt alle IFnets aus, die über eine „ausführbare“ kritische bedingte Transitionenfolge verfügen, d.h. für die es eine Ausführung gibt, in der das Schalten der ersten Transition Voraussetzung für das Schalten der letzten Transition in der Folge ist.

**Definition 3.40 (StrongCond-Kriterium).** *Ein markiertes IFnet $(N, M)$ mit Anfangszustand $M$ erfüllt das* StrongCond*-Kriterium genau dann, wenn für alle kritischen bedingten Transitionenfolgen $t_0, ..., t_n \cup p_0, ..., p_{n-1} \in \mathsf{STR}_{\mathrm{Cond}}$ gilt:*

$$\nexists M_1, M_2 \in [M], c \in \mathcal{C}_\mathsf{c} : M_1 \xrightarrow{t_0 p_0 .. p_{n-1} t_n}_c M_2.$$

Mit dem WeakCond-Kriterium wird auch hier eine abgeschwächte Variante definiert. Diese schließt nur solche Netze als unsicher aus, bei denen der Ausführung der letzten Transition einer kritischen bedingten Transitionenfolge *stets* die Ausführung dessen erster Transition vorausgeht.

**Definition 3.41 (WeakCond-Kriterium).** *Ein markiertes IFnet $(N, M)$ mit Anfangszustand $M$ erfüllt das* WeakCond*-Kriterium genau dann, wenn für alle kritischen bedingten Transitionenfolgen $t_0, ..., t_n \cup p_0, ..., p_{n-1} \in \mathsf{STR}_{\mathrm{Cond}}$ gilt:*

$$\exists M_1, M_2 \in [M], c \in \mathcal{C}_\mathsf{c} : M_1 \xrightarrow{t_0 p_0 .. p_{n-1} t_n}_c M_2 \Rightarrow$$
$$\exists M_3 \in [M] : M \xrightarrow{\sigma t_n} M_3 \ \wedge \ t_0 \notin \sigma \text{ für eine Schaltfolge } \sigma.$$

## 3.6 Zertifikate

Ein InDico-Zertifikat fasst die Ergebnisse der statischen Analyse zusammen, indem es die bereitgestellten Isolationsgarantien beschreibt und deren mögliche Verletzungen charakterisiert. Ein Zertifikat ist ein Tripel

$((N, M)$, ListOfCertifiedCriteria, SetOfViolations$)$

dessen Komponenten die folgenden Informationen enthalten:

- $(N, M)$ beschreibt das ausgezeichnete IFnet-Modell (einschließlich seiner Anfangsmarkierung) auf das sich das Zertifikat bezieht.
- ListOfCertifiedCriteria ist eine Liste der Isolationskriterien bzgl. deren Einhaltung das Netz geprüft wurde.
- SetOfViolations $= (\mathsf{IF}_{\text{NoRead}}, \mathsf{IF}_{\text{NoWriteHigh}}, \mathsf{IF}_{\text{NoWriteNeutral}}, ...)$ enthält für jedes Kriterium eine (ggf. leere) Ergebnismenge, die die erkannten Verletzungen des jeweiligen Kriteriums beschreibt. Eine Verletzung wird charakterisiert durch ein Tripel (Source, Destination, SetOfExecPaths), wobei Source die Quelle des unerlaubten Informationsflusses (eine gefärbte Marke oder eine Transition) bezeichnet, Destination sein Ziel (wiederum eine gefärbte Marke oder eine Transition) und SetOfExecPaths die Menge aller Schaltfolgen, die – ausgehend von der Anfangsmarkierung – die Verletzung herbeiführen. Die Algorithmen zur Berechnung der Ergebnismengen, die den wesentlichen Teil eines Zertifikates ausmachen, werden im folgenden Kapitel 4 beschrieben.

Ein Zertifikat dient als Grundlage für die Beurteilung der Sicherheit des gegebenen Prozesses im Hinblick auf die gestellten Anforderungen. Werden keine Verletzungen angezeigt, ist der Prozess bzgl. der geprüften Kriterien sicher, d.h. es wird für die betrachteten Informationskanäle garantiert, dass keine höher klassifizierte Information an Subjekte mit niedrigerer Freigabe fließen kann. Angezeigte Verletzungen weisen auf die Möglichkeit eines solchen ungewollten Informationsflusses hin. Die Implikationen einer Verletzung für die Sicherheit des konkreten Prozesses können daraufhin im Einzelfall beurteilt werden.

Kapitel 4

# Algorithmen und Implementierung

Die prototypische Implementierung von InDico dient zur Demonstration der praktischen Umsetzbarkeit der Zertifizierung im Sinne eines *Proof-of-Concept* (Génova, 2010) und zu ihrer Bewertung im Hinblick auf die gestellten Anforderungen.

Dieses Kapitel beschreibt die Implementierung des IFnet-Formalismus und die Algorithmen zur Prüfung der Informationsflusskriterien. Die Implementierung setzt auf der *Prozess-Werkstatt* auf, einer Rahmenanwendung für die Analyse von Sicherheits- und Compliance-Anforderungen in Geschäftsprozessen, die am Institut für Informatik und Gesellschaft der Universität Freiburg entwickelt wurde. Abbildung 4.1 stellt die Architektur der Prozess-Werkstatt schematisch dar. Diese umfasst eine Kernapplikation mit grafischer Benutzerschnittstelle und ist in der Programmiersprache *Java* unter Benutzung der quelloffenen *Eclipse Rich Appliaction Platform* geschrieben. Die Funktionalität zur Modellierung, Ausführung und Analyse von Geschäftsprozessen ist in separaten Softwaremodulen (*Plugins*) implementiert.

Die Komponenten von InDico sind in drei Plugins implementiert und in die Prozess-Werkstatt integriert worden. Das erste Plugin stellt die Funktionalität zum Einlesen von IFnet-Modellen aus externen Dateien und zu ihrer internen Repräsentation bereit. Das zweite Plugin implementiert die Schaltregel von IFnet und macht ein Modell damit ausführbar. Das dritte Plugin implementiert die Analysealgorithmen zur Prüfung von IFnet-Modellen und zur Generierung der Zertifikate.

## 4.1 Spezifikation von IFnet-Modellen

Zur Spezifikation und Speicherung von IFnet-Modellen ist eine XML-basierte Sprache definiert worden, die eine Teilmenge der *Petri Net Markup Language* (PNML) ist. PNML ist ein ISO-standardisiertes Austauschformat für Petri-Netz-Beschreibungen, das von einer Vielzahl von Softwareanwendungen unterstützt wird. IFnet nutzt die standardisierten

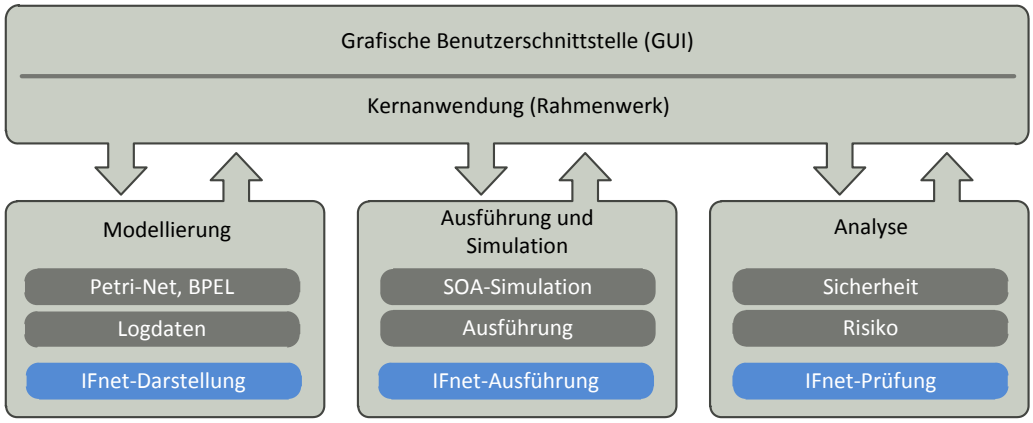



Abbildung 4.1.: Architektur der Prozesswerkstatt mit InDico-Komponenten.

PNML-Konstrukte für die Beschreibung von Elementen, die es mit klassischen Petri-Netzen gemeinsam hat, nämlich Transitionen, Stellen und gerichtete Kanten. Spezifische Erweiterungen von IFnet, wie z.B. die Attribute von Transitionen, werden innerhalb spezieller PNML-Markierungen (`<toolspecific>`-*Tags*) definiert, deren Inhalt vom PNML-Standard nicht interpretiert wird. Auf diese Weise ist jedes IFnet gleichzeitig ein PNML-konformes Petri-Netz, das mit entsprechenden Werkzeugen verarbeitet werden kann (wobei von diesen nur die Struktur des Netzes ohne IFnet-spezifische Besonderheiten berücksichtigt wird).

Abbildung 4.2 zeigt die oberen Ebenen der Struktur des XML-Schemas, das die Syntax der Sprache beschreibt. Auf der obersten Ebene des Netzes (markiert durch den Knoten `<net>`) befinden sich die Elemente, die die Transitionen (Knoten `<transition>`), Stellen (Knoten `<place>`) und Kanten (Knoten `<arc>`) definieren. Die IFnet-spezifischen Konstrukte von Transitionen und Stellen sind unterhalb von `<toolspecific>`-Knoten beschrieben. Des weiteren ist auf der obersten Ebene des Netzes ein `<toolspecific>`-Knoten, unterhalb dessen der Anfangszustand (`<initialMarking>`) und die Subjekt-Bezeichner mit ihren jeweiligen Freigaben (`<subjectIds>`) definiert sind. Das vollständige XML-Schema ist in Abschnitt B wiedergegeben.

IFnet-Modelle werden in XML-Dateien gespeichert, die gültig (*valid*) bzgl. dieses Schemas sind. Um ein Modell zu analysieren, wird seine Beschreibung in die Prozess-Werkstatt eingelesen und in eine interne Darstellung übersetzt. Diese Darstellung besteht im Wesentlichen aus zwei Listen, in denen die Objekte gespeichert sind, die Stellen bzw. Transitionen repräsentieren. Ein Objekt aus einer der beiden Listen verfügt über Referenzen auf Objekte in der jeweils anderen Liste. Auf diese Weise wird unmittelbar der bipartite Graph modelliert, der ein Petri-Netz bzw. ein IFnet ausmacht. Um einfacher in diesem Graph navigieren zu können, verfügt jedes Objekt neben den Referenzen auf seine Nachfolger (d.h. seine Ausgangsknoten) auch über Referenzen auf die Vorgänger (Eingangsknoten).

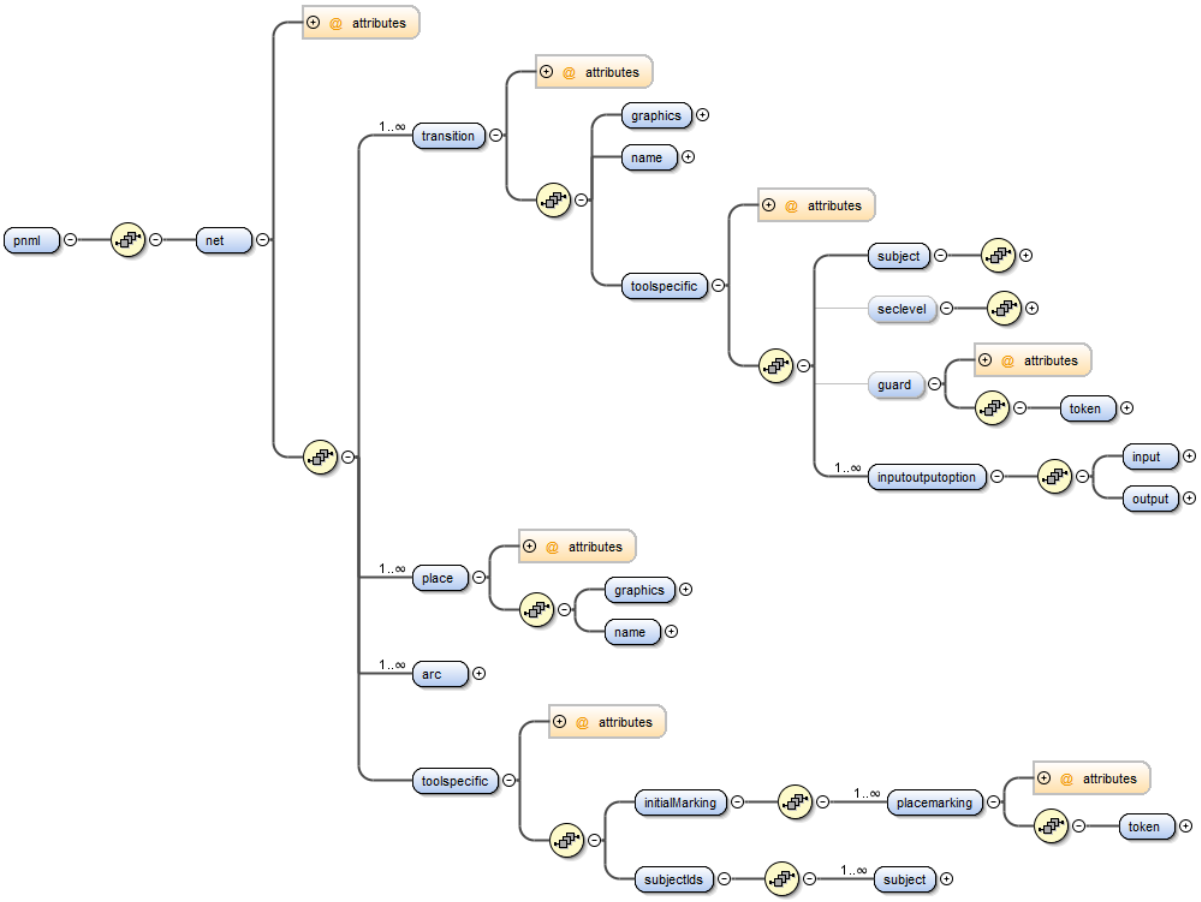



Abbildung 4.2.: Struktur des XML-Schemas für IFnet-Modelle.

| Informationsflusskriterium | Kritische Bereiche |
|---|---|
| NoRead | Elemente der Menge $\mathsf{STR}_{\text{NoRead}}$. |
| NoWriteHigh | Elemente der Menge $\mathsf{STR}_{\text{NoWriteHigh}}$. |
| NoWriteNeutral | Elemente der Menge $\mathsf{STR}_{\text{NoWriteNeutral}}$. |
| StrongCausal, WeakCausal | Elemente der Menge $\mathsf{STR}_{\text{Causal}}$. |
| StrongUsage, WeakUsage | Elemente der Menge $\mathsf{STR}_{\text{Usage}}$. |
| StrongCond, WeakCond | Elemente der Menge $\mathsf{STR}_{\text{Cond}}$. |

Tabelle 4.1.: Informationsflusskriterien und kritische Bereiche.

## 4.2 Prüfungsalgorithmen

Die Aufgabe der Prüfungsalgorithmen von InDico ist zu entscheiden, ob ein IFnet-Modell die in Abschnitt 3.5 beschriebenen Informationsflusskriterien erfüllt. Die Algorithmen sind in einem Plugin der Prozess-Werkstatt implementiert und erhalten als Eingabe ein IFnet-Modell, das aus einer konformen XML-Datei eingelesen wird.

Für jedes Informationsflusskriterium wird ein eigener Algorithmus verwendet, die allerdings auf gemeinsame Subroutinen zurückgreifen. Als Ausgaben erzeugt jeder der Algorithmen eine Ergebnismenge, die die erkannten Verletzungen charakterisiert. Aus diesen Ergebnismengen wird nach der Prüfung ein Zertifikat zusammengestellt. Die Prüfung gliedert sich in die folgenden drei Phasen:

1. **Strukturelle Analyse.** Die strukturelle Analyse dient der Identifikation der „kritischen Bereiche“ in einem IFnet, d.h. der Teilnetze in der es zu einer Verletzung eines Informationsflusskriteriums kommen kann. Als Ergebnis generiert sie für jedes Informationsflusskriterium eine Menge STR* (wobei der Stern durch den Namen des jeweiligen Kriteriums ersetzt wird), die die entsprechenden Teilnetze enthält (vgl. die Definition dieser Mengen in Abschnitt 3.5). Tabelle 4.1 gibt die Zuordnung der Informationsflusskriterien zu diesen Mengen wieder.

2. **Erzeugung des Zustandsgraphen.** Um zu überprüfen, ob es in einem kritischen Bereich tatsächlich zu einem illegalen Informationsfluss kommen kann, müssen die Zustände, die ein IFnet annehmen kann, betrachtet werden. Der Zustandsgraph ist die dazugehörige Datenstruktur, die sämtliche (vom Ausgangszustand aus erreichbaren) Zustände und die zwischen diesen möglichen Übergänge darstellt. Er repräsentiert damit sämtliche möglichen Ausführungspfade, die der modellierte Prozess einnehmen kann.

3. **Analyse des Zustandsgraphen.** Der Zustandsgraph wird in der dritten Phase der Prüfung verwendet, um zu entscheiden, ob es in einem IFnet Ausführungen gibt, die eine Isolationsverletzung verursachen. Die in der ersten Phase identifizierten kritischen Bereiche dienen hier dazu, die Analyse auf die Regionen des Zustandsgraphen einzuschränken, in denen Verletzungen möglich sind.

Die folgenden Abschnitte beschreiben die Algorithmen, die die drei Prüfungsschritte implementieren. Sofern es für die Beschreibung hilfreich ist, werden die wichtigsten Algorithmen in einer Pseudocode-Darstellung wiedergegeben.

### 4.2.1 Strukturelle Analyse

Dieser Abschnitt beschreibt die Algorithmen zur Berechnung der Mengen STR*, die die kritischen Bereiche eines IFnet-Modells enthalten. Bei der Beschreibung der Algorithmen wird auf die Bezeichnungen zurückgegriffen, die in Kapitel 3 eingeführt worden sind. So bezeichnet etwa $T$ die Menge der Transitionen des untersuchten IFnet-Modells, $P$ die Menge seiner Stellen usw.

#### Berechnung von $\mathsf{STR}_{\text{NoRead}}$, $\mathsf{STR}_{\text{NoWriteHigh}}$ und $\mathsf{STR}_{\text{NoWriteNeutral}}$

Die Mengen $\mathsf{STR}_{\text{NoRead}}$, $\mathsf{STR}_{\text{NoWriteHigh}}$ und $\mathsf{STR}_{\text{NoWriteNeutral}}$ enthalten die Transitionen, die einen illegalen Datenfluss verursachen können. Ihre Berechnung ist algorithmisch einfach, da die Zugehörigkeit zu einer dieser Mengen durch die Betrachtung einzelner Transitionen entschieden werden kann. Dazu wird die Liste $T$ der Transitionen vollständig durchlaufen und für jede Transition untersucht, ob sie die entsprechenden, in Abschnitt 3.5.1 beschriebenen Bedingungen erfüllt. Ist dies der Fall, wird die entsprechende Transition als ein Element der jeweiligen Menge zugeordnet.

#### Berechnung von $\mathsf{STR}_{\text{Causal}}$

Die Menge $\mathsf{STR}_{\text{Causal}}$ speichert die kritischen Transitionenfolgen eines IFnet-Modells. Ein Element ist eine durch Stellen verbundene Folge $t_0, ..., t_n \cup p_0, ..., p_{n-1} \subseteq T \cup P$, die den in Definition 3.28 festgelegten Bedingungen entsprechend mit einer `High`-klassifzierten Transition beginnt und mit einer Transition mit `Low`-Freigabe endet. Für die Berechnung von $\mathsf{STR}_{\text{Causal}}$ wird das IFnet-Modell mit den folgenden Routinen traversiert.

Algorithmus 4.1 beschreibt in Pseudocode-Notation die Prozedur $\mathsf{CreateSTR}_{\text{Causal}}$, die unter Benutzung der in Algorithmus 4.2 dargestellten Subroutine $\mathsf{VisitSTR}_{\text{Causal}}$ die Menge aller kritischen Transitionenfolgen in einem gegebenen IFnet berechnet. Neben den Komponenten des untersuchten IFnets benutzen die Prozeduren eine Datenstruktur zur Speicherung von Pfaden durch ein IFnet-Modell (Variable *modelPath*), die eine Liste von Knoten (Stellen und Transitionen) ist.

**Algorithmus 4.1** $\mathsf{CreateSTR}_{\text{CAUSAL}}$-Prozedur.

**Eingabe:** IFnet $N = ((P, T, F, \text{IN}, \text{OUT}), \text{S}_{\mathcal{U}}, \text{A}, \text{G}, \text{LABEL}_{\mathcal{SC}}, \text{CLEAR}_{\mathcal{SC}})$.

```
1: STR_CAUSAL ← ∅
2: for all t ∈ T do
3:    if LABEL_SC (t) = High then
4:       modelPath ← ∅
5:       VisitSTR_CAUSAL(t, modelPath)
6:    end if
7: end for
```

**Algorithmus 4.2** $\mathsf{VisitSTR}_{\text{CAUSAL}}$-Prozedur.

**Eingabe:** Knoten $n \in P \cup T$, Pfad *modelPath*.

```
1: if n ∈ modelPath then
2:    return
3: end if
4: modelPath ← modelPath.n
5: if n ∈ T then
6:    if LABEL_SC(t) = High and |modelPath| > 1 then
7:       return
8:    end if
9:    if CLEAR_SC(S_U(n)) = Low then
10:      STR_CAUSAL ← STR_CAUSAL ∪ modelPath
11:      return
12:   end if
13: end if
14: for all n' ∈ n• do
15:    VisitSTR_CAUSAL(n', CLONE(modelPath))
16: end for
```

Die Hauptroutine $\mathsf{CreateSTR}_{\text{CAUSAL}}$ initialisiert $\mathsf{STR}_{\text{CAUSAL}}$ und wendet auf alle als `High` klassifizierten Transitionen des gegebenen IFnets die $\mathsf{VisitSTR}_{\text{CAUSAL}}$-Prozedur an, die eine Tiefensuche durchführt. Die Argumente von $\mathsf{VisitSTR}_{\text{CAUSAL}}$ sind zum einen der Knoten, der als nächstes untersucht werden soll, und der Pfad, der bereits von der als `High` klassifizierten Transition zurückgelegt worden ist. Zunächst prüft die Prozedur, ob der aktuell untersuchte Knoten bereits in dem zurückgelegten Pfad vorkommt, d.h. ob ein Zyklus vorliegt, und bricht in diesem Fall ab. Andernfalls wird der aktuelle Knoten zum Pfad hinzugefügt. Falls er die Klassifikation `High` besitzt und nicht die Anfangstransition ist, bricht die Routine hier ab. Eine kritische Transitionenfolge ist gefunden worden, wenn die untersuchte Transition die Freigabe `Low` hat. In diesem Fall wird der Pfad zur Menge $\mathsf{STR}_{\text{CAUSAL}}$ hinzugefügt und die Routine beendet die Suche. Ist dies nicht der Fall, wird $\mathsf{VisitSTR}_{\text{CAUSAL}}$ auf allen Ausgangsknoten aufgerufen, wobei jeweils eine Kopie von *modelPath* übergeben wird.

**Algorithmus 4.3** CreateSTR$_{\text{USAGE}}$-Prozedur.

**Eingabe:** IFnet $N = ((P, T, F, \text{IN}, \text{OUT}), \text{S}_{\mathcal{U}}, \text{A}, \text{G}, \text{LABEL}_{\mathcal{SC}}, \text{CLEAR}_{\mathcal{SC}})$.

```
 1: STR_USAGE ← ∅
 2: for all p ∈ P do
 3:   if |p•| ≥ 2 then
 4:     resultH ← ∅
 5:     resultL ← ∅
 6:     for all t ∈ p• do
 7:       modelPath ← ∅
 8:       VisitHighSTR_USAGE(t, modelPath)
 9:       VisitLowSTR_USAGE(t, modelPath)
10:     end for
11:     for all H ∈ resultH do
12:       for all L ∈ resultL do
13:         if H ∩ L = ∅ then
14:           STR_USAGE ← STR_USAGE ∪ H.p.L
15:         end if
16:       end for
17:     end for
18:   end if
19: end for
```

### Berechnung von STR$_{\text{USAGE}}$

Die Menge STR$_{\text{USAGE}}$ enthält die in einem IFnet enthaltenen kritischen Nutzungsstellen, d.h. die Bereiche, in denen es zu einem durch einen Nutzungskonflikt verursachten illegalen Informationsfluss kommen kann. Die folgenden Routinen werden zu ihrer Berechnung eingesetzt.

Algorithmus 4.3 beschreibt in Pseudocode-Notation die Prozedur CreateSTR$_{\text{USAGE}}$, die unter Benutzung der in Algorithmus 4.4 und Algorithmus 4.5 dargestellten Subroutinen die Menge aller kritischen Nutzungsstellen in einem IFnet angibt. Eine kritische Nutzungstelle ist eine Stelle, in die sowohl ein Pfad von einer `High`-klassifizierten Transition mündet, als auch ein Pfad zu einer Transition mit `Low`-Freigabe ausgeht.

Die in Frage kommenden Stellen (Kandidatenstellen) sind demnach jene, die mindestens zwei Ausgangsknoten besitzen. Die Prozedur CreateSTR$_{\text{USAGE}}$ sucht die Liste $P$ der Stellen nach diesen ab und wendet darauf die Tiefensuchen VisitHighSTR$_{\text{USAGE}}$ und VisitLowSTR$_{\text{USAGE}}$ an. Die erste (beschrieben in Algorithmus 4.4) sucht nach allen `High`-klassifizierten Transitionen, von denen es einen ausgehenden Pfad zur jeweiligen Kandidatenstelle gibt. Dazu wird das IFnet-Modell entgegen der Flussrelation durchsucht.

**Algorithmus 4.4** VisitHighSTR$_{\text{USAGE}}$-Prozedur.

**Eingabe:** Knoten $n \in P \cup T$, Pfad *modelPath*.

```
1: if n ∈ modelPath then
2:    return
3: end if
4: modelPath ← n.modelPath
5: if n ∈ T then
6:    if CLEAR_SC(S_U(n)) = Low then
7:       return
8:    end if
9:    if LABEL_SC(n) = High then
10:      resultH ← resultH ∪ modelPath
11:      return
12:   end if
13: end if
14: for all n' ∈ •n do
15:    VisitHighSTR_USAGE(n', CLONE(modelPath))
16: end for
```

**Algorithmus 4.5** VisitLowSTR$_{\text{USAGE}}$-Prozedur.

**Eingabe:** Knoten $n \in P \cup T$, Pfad *modelPath*.

```
1: if n ∈ modelPath then
2:    return
3: end if
4: modelPath ← modelPath.n
5: if n ∈ T then
6:    if LABEL_SC(n) = High then
7:       return
8:    end if
9:    if CLEAR_SC(S_U(n)) = Low then
10:      resultL ← resultL ∪ modelPath
11:      return
12:   end if
13: end if
14: for all n' ∈ n• do
15:    VisitLowSTR_USAGE(n', CLONE(modelPath))
16: end for
```

Die zweite Tiefensuche (beschrieben in Algorithmus 4.5) sucht ausgehend von der Kandidatenstelle nach einer darauf folgenden Transition mit `Low`-Freigabe. Die Suchroutinen speichern die gefundenen Pfade in den Variablen *resultH* bzw. *resultL*. Im Anschluss an die Suche kombiniert $\mathsf{CreateSTR}_{\text{USAGE}}$ jeden Pfad aus *resultH* mit jedem Pfad aus *resultL* (sofern diese unterschiedliche Transitionen enthalten) sowie der Kandidaten-Stelle und speichert das Ergebnis in der Menge $\mathsf{STR}_{\text{USAGE}}$.

#### Berechnung von $\mathsf{STR}_{\text{COND}}$

Die Menge $\mathsf{STR}_{\text{COND}}$ speichert alle kritischen bedingten Transitionenfolgen eines IFnet-Modells. Da kritische bedingte Transitionenfolgen eine sehr ähnliche Struktur wie kritische Transitionenfolgen (d.h. Elemente der Menge $\mathsf{STR}_{\text{CAUSAL}}$) haben, sind die verwendeten Routinen annähernd die gleichen wie $\mathsf{CreateSTR}_{\text{CAUSAL}}$ (Algorithmus 4.1) bzw. $\mathsf{VisitSTR}_{\text{CAUSAL}}$ (Algorithmus 4.2).

Der Unterschied zwischen beiden Arten von Transitionenfolgen ist, dass die Anfangstransition einer Folge aus $\mathsf{STR}_{\text{CAUSAL}}$ eine `High`-klassifizierte Transition ist, während der Anfangstransition einer Folge aus $\mathsf{STR}_{\text{COND}}$ ein Prädikat zugeordnet ist, das eine `High`-klassifizierte Marke benutzt. Im Wesentlichen unterscheiden sich die Routinen zur Berechnung von $\mathsf{STR}_{\text{COND}}$ von den Algorithmen $\mathsf{CreateSTR}_{\text{CAUSAL}}$ und $\mathsf{VisitSTR}_{\text{CAUSAL}}$ in der Prüfung dieser Bedingung. Entsprechend ist die Verzweigung in Zeile 3 von Algorithmus 4.1 sowie in Zeile 6 von Algorithmus 4.2 angepasst. Abgesehen von der unterschiedlichen Ergebnismenge ($\mathsf{STR}_{\text{COND}}$ statt $\mathsf{STR}_{\text{CAUSAL}}$) ist das Vorgehen darüber hinaus identisch.

### 4.2.2 Erzeugung des Zustandsgraphen

Der Zustand eines IFnets wird durch die Belegung der Stellen mit Marken charakterisiert. Der Zustandsgraph repräsentiert alle erreichbaren Zustände durch einen gerichteten Graphen, in dem ein Knoten einen Zustand repräsentiert und dessen Kinder die von diesem Zustand aus erreichbaren Folgezustände darstellen, die durch das Schalten von Transitionen erreicht werden können.

Zur Illustration zeigt Abbildung 4.3 einen Ausschnitt aus einem Zustandsgraphen.[1] Der oberste Knoten repräsentiert den Anfangszustand: Hier sind lediglich die Stellen $\mathsf{p}_0$ und $\mathsf{p}_{15}$ mit jeweils einer schwarzen Marke (abgekürzt als $\mathsf{B}$), sowie die Stelle $\mathsf{p}_9$ mit einer gefärbten Marke $\mathsf{H}$ belegt. Die Klassifikation von gefärbten Marken wird durch einen tiefgestellten Buchstaben ($\mathsf{H}$ bzw. $\mathsf{L}$) angezeigt. Eine Kante von einem Knoten $A$ zu einem Knoten $B$ ist mit der Bezeichnung der Transition annotiert, deren Schalten den Übergang von $A$ zu $B$ bewirkt.

[1] Die Abbildung zeigt einen Ausschnitt aus dem Zustandsgraphen des IFnet-Modells, das im Rahmen des in Abschnitt 5.1 gezeigten Anwendungsszenarios verwendet wird. Das zugehörige Modell ist in Abbildung 5.2 dargestellt.

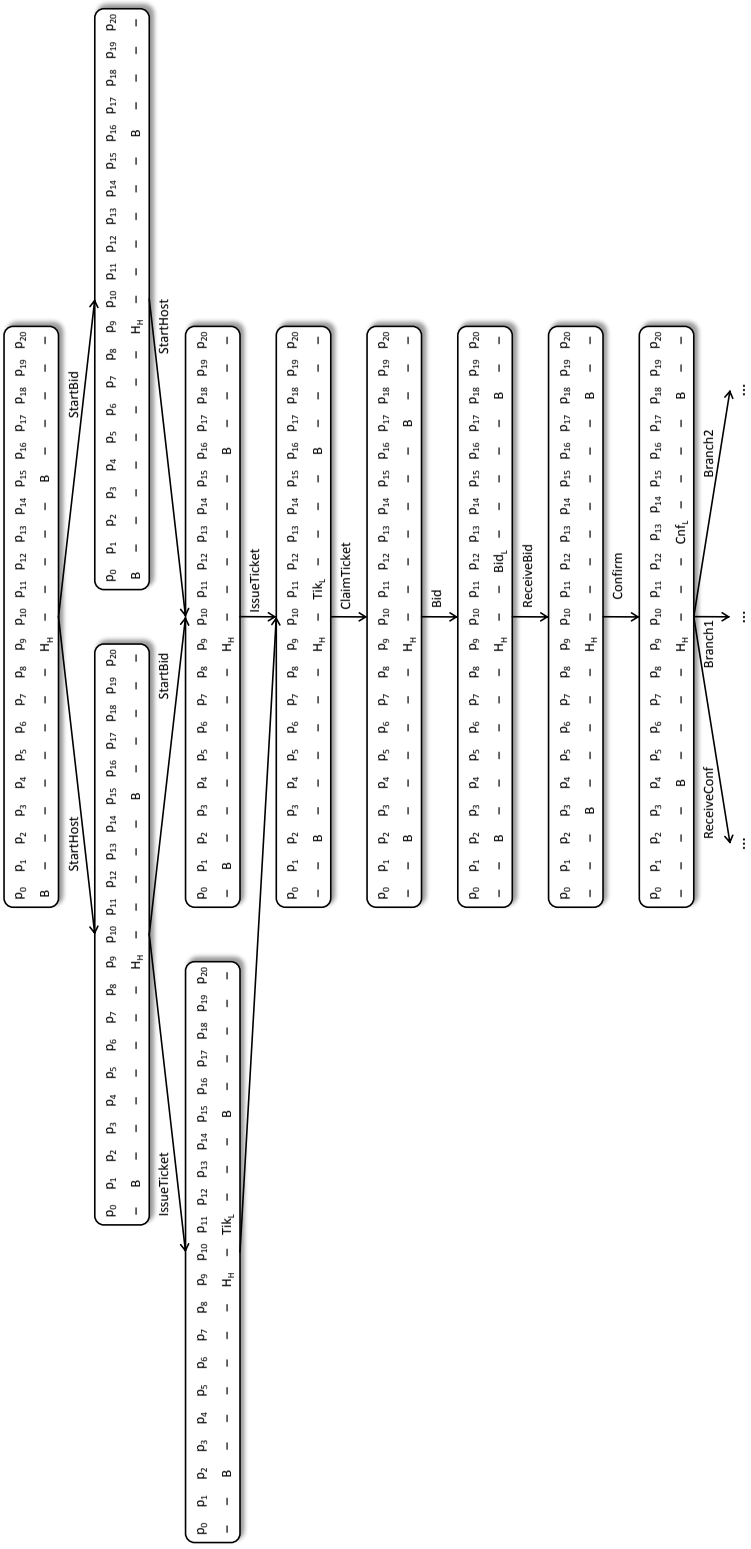



Abbildung 4.3.: Ausschnitt aus einem Zustandsgraphen.

**Algorithmus 4.6** CreateMarkingGraph-Prozedur.

**Eingabe:** Zustand $M$.

```
1:  for all t ∈ T do
2:    for all i ∈ CV_t^M do
3:      Erzeuge Zustand M_1, so dass M --t_i--> M_1
4:      if ∃M_2 ∈ markingGraph : M_1 ≅ M_2 then
5:        Erzeuge Referenz von M nach M_2 mit Annotation t_i
6:      else
7:        markingGraph ← markingGraph ∪ M_1
8:        Erzeuge Referenz von M nach M_1 mit Annotation t_i
9:        CreateMarkingGraph(M_1)
10:     end if
11:   end for
12: end for
```

Die rekursive Prozedur CreateMarkingGraph erzeugt für ein markiertes IFnet den zugehörigen Zustandsgraphen. Der erzeugte Zustandsgraph wird in der Liste *markingGraph* gespeichert, die Knoten-Objekte als Repräsentanten von Zuständen enthält. Ein solches Objekt charakterisiert einen Zustand, indem es für jede Stelle des IFnet-Modells die zugehörige Markierung (eine Multimenge von Marken) speichert. Darüber hinaus enthält es eine Liste von Referenzen auf weitere Knoten-Objekte, die die Folgezustände repräsentieren. Eine solche Referenz ist mit einem Bezeichner versehen, der die Transition (und ggf. die Schaltmöglichkeit) kennzeichnet, die den entsprechenden Übergang bewirkt. Zusätzlich verfügt ein Knoten über Referenzen auf seine Vorgängerknoten, die die Navigation im Zustandsgraphen erleichtern.

Vor dem Aufruf von CreateMarkingGraph ist die Liste *markingGraph* leer. Zur Konstruktion des Zustandsgraphen wird CreateMarkingGraph mit dem Startzustand aufgerufen und füllt *markingGraph* sukzessive mit den vom Startzustand aus erreichbaren Zuständen. Algorithmus 4.6 beschreibt CreateMarkingGraph in Pseudocode-Notation.[2]

CreateMarkingGraph geht zunächst die Liste der Transitionen durch und prüft für jede, ob diese im aktuell untersuchten Zustand schaltbereit ist. Die für diese Prüfung notwendige Funktionalität liefert das InDico-Plugin der Prozess-Werkstatt, das die Schaltregel von IFnet implementiert (siehe Abbildung 4.1). Falls es eine schaltbereite Transition gibt, wird ein neues Objekt erzeugt, das den durch diesen Übergang erreichten Zustand repräsentiert. Falls es in der Liste *markingGraph* bereits einen solchen Zustand gibt, wird ein Verweis von dem aktuellen Zustand zu diesem (bereits in der Liste vorhandenen) Zustand eingefügt und die Prozedur endet.[3] Ist dies nicht der Fall, wird das neu erzeugte

[2] Um die Darstellung zu vereinfachen, wird im Pseudocode und im Text nicht zwischen Zuständen und den Objekten, die diese Zustände repräsentieren, unterschieden.

[3] Im Pseudocode wird der Vergleich zweier Zustände durch den Operator $\cong$ dargestellt. In der Implementierung wird dafür eine Routine verwendet, die zwei Objekte hinsichtlich des dargestellten Zustandes vergleicht.

**Algorithmus 4.7** CheckNoRead-Prozedur.

**Eingabe:** *markingGraph*, $\mathsf{STR}_{\text{NoRead}}$.

1: $\mathsf{IF}_{\text{NoRead}} \leftarrow \emptyset$
2: **for all** $M \in$ *markingGraph* **do**
3:  **for all** $M_1 : M \xrightarrow{t_i} M_1$ **do**
4:  **if** $t \in \mathsf{STR}_{\text{NoRead}}$ **and** $\exists c \in \text{InputSet}(t, i)$ : $\{read\} \in \text{A}(t, c) \wedge \text{Label}_{\mathcal{SC}}(c) = \texttt{High}$ **then**
5:  $\mathsf{IF}_{\text{NoRead}} \leftarrow \mathsf{IF}_{\text{NoRead}} \cup (c, t, \mathsf{TraceBack}(M.M_1))$
6:  **end if**
7:  **end for**
8: **end for**

Zustandsobjekt in *markingGraph* eingefügt und ein Verweis vom aktuellen Zustand auf dieses erzeugt. Anschließend wird CreateMarkingGraph auf dem neu erzeugten Zustandsobjekt aufgerufen.

Eine Voraussetzung für die Terminierung von CreateMarkingGraph ist, dass die Zustandsmenge des untersuchten IFnet-Modells endlich ist. Dies ist nicht zwangsläufig der Fall, etwa wenn Schleifen vorliegen, die mit jedem Durchlauf in einer Stelle eine zusätzliche Marke produzieren. In der Implementierung von InDico wird in solchen Fällen die Verarbeitung nach Überschreitung eines Zeitlimits abgebrochen.

### 4.2.3 Analyse des Zustandsgraphen

Der Zustandsgraph ist die Grundlage für die Prüfung der Informationsflusskriterien. Er wird benutzt, um für einen kritischen Bereich zu prüfen, ob dieser „ausführbar“ ist, d.h. ob es eine Schaltfolge gibt, die einen ungewollten Informationsfluss auslösen kann.

Die folgenden Abschnitte stellen dazu die entsprechenden Algorithmen vor. Als Ausgabe erzeugen diese für jedes Informationsflusskriterium eine Menge $\mathsf{IF}_*$ (wobei der Stern durch den Namen des jeweiligen Kriteriums ersetzt wird), die entdeckte Verletzungen als Tripel aus der Quelle des illegalen Informationsflusses, seiner Senke sowie den Ausführungen, die die Verletzung auslösen, speichern (vgl. die Beschreibung der Zertifikate in Abschnitt 3.6).

#### Prüfung von NoRead, NoWriteHigh und NoWriteNeutral

Die Datenflusskriterien NoRead, NoWriteHigh und NoWriteNeutral sind verhältnismäßig einfach zu prüfen, da hier nur einzelne Zustandsübergänge betrachtet werden müssen. Das Vorgehen wird exemplarisch für das NoRead-Kriterium gezeigt und lässt sich direkt auf die beiden übrigen Kriterien übertragen.

**Algorithmus 4.8** TraceBack-Prozedur.

**Eingabe:** Pfad durch den Zustandsgraphen *statePath*.

```
1: statePathSet ← ∅
2: TraceBackVisit (statePath)
3: return statePathSet
```

**Algorithmus 4.9** TraceBackVisit-Prozedur.

**Eingabe:** Pfad *statePath*.

```
1: M ← statePath[0]
2: for all M1 : M1 ⟶ M do
3:    if M1 ∉ statePath then
4:       TraceBackVisit(M1.CLONE(statePath))
5:    end if
6: end for
7: if ∄M1 : M1 ⟶ M then
8:    statePathSet ← statePathSet ∪ statePath
9: end if
```

Algorithmus 4.7 zeigt in Pseudocode-Notation die CheckNoRead-Prozedur, die in einem Zustandsgraphen *markingGraph* prüft, ob das NoRead-Kriterium eingehalten wird. In der Menge $\mathsf{IF}_{\text{NoRead}}$ werden gefundene Verletzungen gespeichert, wobei eine Verletzung durch ein Tripel charakterisiert wird, das die Quelle des illegalen Informationsflusses (die gelesene Marke), sein Ziel (die lesende Transition) sowie die Menge der Pfade vom Anfangszustand zu dem Zustand, bei dem die Verletzung auftritt, umfasst.

Zu diesem Zweck geht CheckNoRead die Menge der Zustände durch und untersucht, ob es von dort einen Übergang gibt, bei dem die schaltende Transition zur Menge $\mathsf{STR}_{\text{NoRead}}$ gehört und lesend auf eine höher klassifizierte Marke zugreift. In diesem Fall wird die Verletzung in $\mathsf{IF}_{\text{NoRead}}$ gespeichert.

Die dabei aufgerufene Prozedur TraceBack liefert für einen übergebenen Pfad durch den Zustandsgraphen alle Pfade zurück, die beim Anfangszustand beginnen und mit dem übergebenen Pfad enden. TraceBack und die von ihr aufgerufene Routine TraceBackVisit sind in Algorithmus 4.8 bzw. Algorithmus 4.9 dargestellt. Die Aufgabe von TraceBack beschränkt sich auf die Initialisierung der globalen Variable *statePathSet*, in der sämtliche gefundenen Pfade gespeichert werden, und den Aufruf von TraceBackVisit. TraceBackVisit steigt, ausgehend vom übergebenen Zustand, im Markierungsgraphen auf allen möglichen Pfaden auf bis der initiale Zustand gefunden worden ist und speichert den dabei durchlaufenen Pfad anschließend in *statePathSet*.

Die Kriterien NoWriteHigh und NoWriteNeutral werden mit annähernd denselben Prozeduren geprüft, bei denen lediglich die Bedingungen und Ergebnismengen in den Zeile 4 und 5 von Algorithmus 4.7 entsprechend modifiziert werden.

**Algorithmus 4.10** CheckStrongCausal-Prozedur.

**Eingabe:** *markingGraph*, $\mathsf{STR}_{\text{Causal}}$.

```
1:  IF_StrongCausal ← ∅
2:  for all modelPath ∈ STR_Causal do
3:     statePathSet ← ∅
4:     CheckCausality (modelPath)
5:     resultStatePathSet ← ∅
6:     for all statePath ∈ statePathSet do
7:        resultStatePathSet ← resultStatePathSet ∪ TraceBack(statePath)
8:     end for
9:     IF_StrongCausal ← IF_StrongCausal ∪
           (modelPath[0], modelPath[|modelPath| − 1], resultStatePathSet)
10: end for
```

### Prüfung von StrongCausal und WeakCausal

Das StrongCausal-Kriterium schließt alle IFnets aus, in denen das Schalten einer höher klassifizierten Transition Voraussetzung für das Schalten einer Transition mit niedrigerer Freigabe ist. Es ist erfüllt, wenn keine der kritischen Transitionenfolgen aus $\mathsf{STR}_{\text{Causal}}$ „ausführbar“ ist, d.h. wenn es keine Pfade durch den Zustandsgraphen gibt, in denen alle Transitionen einer Folge ausgeführt werden und die Ausführung einer Transition Voraussetzung für die Ausführung der jeweils nächsten Transition aus der Folge ist.

Die Prozedur CheckStrongCausal überprüft mit Hilfe der Subroutinen CheckCausality und CheckAdjacent ob für eine Transitionenfolge ein entsprechender Ausführungspfad im Zustandsgraphen vorliegt. Die Prozeduren sind in Pseudocode-Notation in Algorithmus 4.10, Algorithmus 4.11 und Algorithmus 4.12 beschrieben.

CheckStrongCausal initialisiert die Ergebnismenge $\mathsf{IF}_{\text{StrongCausal}}$ und wendet auf jeder kritischen Transitionenfolge aus der Menge $\mathsf{STR}_{\text{Causal}}$ die Prozedur CheckCausality an, die in der globalen Variable *statePathSet* sämtliche Pfade durch den Zustandsgraphen zurückliefert, die die übergebene Transitionenfolge ausführen. CheckCausality baut ausgehend von den Zuständen, in denen die erste Transition der gegebenen Folge schaltbereit ist, sukzessive (Transition für Transition) die Menge dieser Pfade auf. Zu Anfang von CheckCausality wird die Menge *statePathSet* mit den Zustandspaaren initialisiert, zwischen denen die erste Transition der untersuchten Folge einen Übergang herstellt.

Die folgende Schleife stellt den Kern der Prozedur dar: Mit jedem Durchlauf aktualisiert und erweitert sie die Pfade in der Menge *statePathSet*, so dass diese anschließend sämtliche Pfade bis zur Ausführung der jeweils nächsten Transition der untersuchten Folge umfasst. Zu Anfang der Schleife werden die als nächstes gesuchte Transition in der globalen Variablen $t_{End}$ und die davor liegende Eingangstelle in der globalen Variablen *connectingPlace* gespeichert. Die Menge *tempPathSet* speichert in jedem Durchlauf Zwischenergebnisse und wird zurückgesetzt. Die anschließende innere Schleife arbeitet

**Algorithmus 4.11** CheckCausality-Prozedur.

**Eingabe:** *modelPath* $\in \mathsf{STR}_{\text{CAUSAL}}$.

```
1:  for all M ∈ markingGraph do
2:    for all M_1 : M --modelPath[0]--> M_1 do
3:      statePathSet ← statePathSet ∪ M.M_1
4:    end for
5:  end for
6:  for all i : 0 ≤ i ≤ |modelPath| − 2 and i gerade do
7:    connectingPlace ← modelPath[i + 1]
8:    t_End ← modelPath[i + 2]
9:    tempPathSet ← ∅
10:   for all σ.M_{n−1}.M_n ∈ statePathSet do
11:     tokenBag ← M_n(p) − M_{n−1}(p)
12:     adjacentPathSet ← ∅
13:     CheckAdjacent(σ.M_{n−1}.M_n)
14:     tempPathSet ← tempPathSet ∪ adjacentPathSet
15:   end for
16:   statePathSet ← tempPathSet
17: end for
```

nacheinander jeden in *statePathSet* vorhandenen Pfad ab, indem versucht wird, diesen zu einer Ausführung von $t_{End}$ zu „verlängern“, wobei die Ausführung von $t_{End}$ abhängig von der Ausführung der vorhergehenden Transition sein muss.

Entscheidend für die Abhängigkeit der über *connectingPlace* verbundenen Transitionen ist, dass $t_{End}$ Marken konsumiert, die von der vorhergehenden Transition in *connectingPlace* produziert worden sind. Die in *connectingPlace* produzierte Markenmenge wird in der Variablen *tokenBag* zwischengespeichert. Die eigentliche Suche nach einem Pfad zu $t_{End}$ wird von der rekursiven Prozedur CheckAdjacent durchgeführt. Diese speichert das Ergebnis des Suchlaufs in der globalen Variablen *adjacentPathSet*, die vor dem Aufruf zurückgesetzt wird. Nach der Rückkehr von CheckAdjacent enthält *adjacentPathSet* alle Pfade, die mit dem aktuellen Teilpfad $(\sigma.M_{n-1}.M_n)$ beginnen und mit einem Zustand nach der Ausführung von $t_{End}$ enden. Am Ende eines Durchlaufs der inneren Schleife wird die Menge *adjacentPathSet* zu *tempPathSet* hinzugefügt. Nachdem die innere Schleife geendet hat, enthält *tempPathSet* sämtliche Pfade, die mit einem Zustand nach der Ausführung von $t_{End}$ enden. Damit die Suche nach der nächsten Transition bei den richtigen Zuständen beginnt, wird *statePathSet* vor dem nächsten Durchlauf der äußeren Schleife auf *tempPathSet* gesetzt.

Die rekursive CheckAdjacent-Prozedur, die in Algorithmus 4.12 dargestellt ist, prüft, ob die Transition $t_{End}$ (die vor jedem Aufruf von der CheckCausality-Prozedur gesetzt wird) im letzten Zustand $M$ des übergebenen Pfades $\tau.M$ schaltbereit ist und durch das Schalten mindestens eine Marke aus der Menge *tokenBag* konsumiert. In diesem Fall ist ein Pfad zu einer Ausführung von $t_{End}$ gefunden worden, der in der Menge *adjacentPathSet*

**Algorithmus 4.12** CheckAdjacent-Prozedur.

**Eingabe:** Zustandsfolge $\tau.M$.

```
1: for all M_1 : M --s_i--> M_1 do
2:   if s = t_End then
3:     if In(t_End, connectingPlace)[i] ∩ tokenBag ≠ ∅ then
4:       adjacentPathSet ← adjacentPathSet ∪ τ.M.M_1
5:     end if
6:   else
7:     if M_1 ∉ τ.M then
8:       CheckAdjacent(CLONE(τ.M.M_1))
9:     end if
10:  end if
11: end for
```

gespeichert wird. Jeder durch das Schalten einer anderen Transition als $t_{End}$ erreichbare Zustand wird – sofern dieser noch nicht im Pfad enthalten ist – zu einer Kopie des Pfades hinzugefügt und CheckAdjacent auf dieser aufgerufen.

Die Pfade durch den Zustandsgraphen, die die gegebene kritische Transitionenfolge ausführen, sind im Anschluss an die Ausführung von CheckCausality in der Menge *statePathSet* gespeichert. Aus dieser konstruiert die CheckStrongCausal-Prozedur die Menge $\mathsf{IF}_{\text{StrongCausal}}$ der Verletzungen des StrongCausal-Kriteriums: Diese enthalten für jede ausführbare kritische Transitionenfolge ein Tripel, das die Quelle des illegalen Informationsflusses (die erste Transition der kritischen Folge), sein Ziel (die letzte Transition) sowie sämtliche Pfade vom Anfangszustand zu den Zuständen, bei denen Verletzungen auftreten, umfasst.

Die Prüfung des schächeren WeakCausal-Kriteriums baut auf der durch die CheckStrongCausal-Prozedur erzeugten Ergebnismenge $\mathsf{IF}_{\text{StrongCausal}}$ auf. Für jedes Element $(t_H, t_L, \textit{resultStatePathSet}) \in \mathsf{IF}_{\text{StrongCausal}}$ wird untersucht, ob es einen Pfad im Zustandsgraphen gibt, der vom Anfangszustand bis zu einer Ausführung von $t_L$ führt, ohne dass eine Ausführung von $t_H$ vorkommt. Dies wird überprüft, indem von allen Knoten im Zustandsgraphen, die auf eine Ausführung von $t_L$ folgen, mit Hilfe der TraceBack-Prozedur alle Pfade vom Ausgangsknoten zu $t_L$ generiert werden. Ist die Ausführung von $t_H$ Teil jedes dieser Pfade, so ist WeakCausal verletzt und das Tripel $(t_H, t_L, \textit{resultStatePathSet})$ wird in die Menge $\mathsf{IF}_{\text{WeakCausal}}$ aufgenommen, die alle Verletzungen dieses Kriteriums speichert. Da WeakCausal ein schwächeres Kriterium als StrongCausal ist, gilt stets $\mathsf{IF}_{\text{WeakCausal}} \subseteq \mathsf{IF}_{\text{StrongCausal}}$.

**Algorithmus 4.13** CheckStrongUsage-Prozedur.

**Eingabe:** *markingGraph*, $\mathsf{STR}_{\text{Usage}}$.

```
1: IF_StrongUsage ← ∅
2: for all ((t_0, ..., t_m), p, (t_{m+1}, ..., t_n)) ∈ STR_Usage do
3:    setOfUsageConflicts ← CheckUsageConflict(((t_0, ..., t_m), p, (t_{m+1}, ..., t_n)))
4:    resultStatePathSet ← ∅
5:    for all (σ.M_1.M_2, τ.M_3.M_4.v) ∈ setOfUsageConflicts do
6:       resultStatePathSet ← resultStatePathSet ∪ TraceBack(σ.M_1.M_2)
7:    end for
8:    IF_StrongUsage ← IF_StrongUsage ∪ (t_0, t_n, resultStatePathSet)
9: end for
```

### Prüfung von StrongUsage und WeakUsage

Das StrongUsage-Kriterium schließt alle IFnets aus, in denen es zu einem Nutzungskonflikt kommen kann. Es ist erfüllt, wenn keine der in der Menge $\mathsf{STR}_{\text{Usage}}$ enthaltenen kritischen Nutzungsstellen ausführbar ist, d.h. wenn eine `High`-klassifizierte Transition die Ausführung einer Transition mit `Low`-Freigabe blockieren kann, da beide – möglicherweise über zwischengeschaltete Transitionen – in einer Stelle $p$ um dieselbe Marke konkurrieren.

In der Menge $\mathsf{STR}_{\text{Usage}}$ sind diese kritischen Nutzungsstellen als Tripel $((t_0, ..., t_m), p, (t_{m+1}, ..., t_n))$ gespeichert. In dieser Darstellung ist $t_0$ die höher klassifizierte Transition, die den Beginn einer über Stellen verbundenen Transitionenfolge bis hin zu $t_m$ markiert. $t_n$ ist die Transition mit `Low`-Freigabe, die durch das Schalten von $t_H$ möglicherweise blockiert wird und am Ende einer durch Stellen verbundenen Transitionenfolge steht, die bei einer Transition $t_{m+1}$ beginnt. Für $t_m$ und $t_{m+1}$ ist $p$ eine Eingangsstelle.

Die CheckStrongUsage-Prozedur, die in Pseudocode-Notation in Algorithmus 4.13 wiedergegeben ist, prüft für eine kritische Nutzungsstelle, ob ein Nutzungskonflikt vorliegt, wobei sie die zuvor definierten Routinen CheckCausality und CheckAdjacent verwendet. Darüber hinaus benutzt sie die Routine CheckUsageConflict, die in Algorithmus 4.14 beschrieben wird.

CheckStrongUsage ruft zunächst für jede kritische Nutzungsstelle aus der Menge $\mathsf{STR}_{\text{Usage}}$ die Routine CheckUsageConflict auf. Diese generiert mit der CheckCausality-Routine die Mengen *startStatePathSet* und *endStatePathSet*, die für die Transitionenfolgen $t_0, ..., t_m$ bzw. $t_{m+1}, ..., t_n$ alle möglichen Ausführungspfade enthalten. In der darauf folgenden Schleife wird für jeden Ausführungspfad aus *startStatePathSet* untersucht, ob es im Zustand vor der Ausführung von $t_m$ (Zustand $M_1$) einen Ausführungspfad zu $t_{m+1}$ gibt, bei dem $t_{m+1}$ Marken konsumiert, die auch $t_m$ im Falle der Ausführung konsumieren würde. Da es sich hier um die gleiche Situation wie im Fall abhängiger Transitionen handelt, kann die CheckAdjacent-Prozedur eingesetzt werden. Als Parameter wird die globale Variable *tokenBag* auf die Menge der Marken gesetzt, die $t_m$ im Falle des Schaltens

**Algorithmus 4.14** CheckUsageConflict-Prozedur.

**Eingabe:** Nutzungsstelle $((t_0, ..., t_m), p, (t_{m+1}, ..., t_n))$.

```
1:  setOfUsageConflicts ← ∅
2:  statePathSet ← ∅
3:  CheckCausality(t_0, ..., t_m)
4:  startStatePathSet ← statePathSet
5:  statePathSet ← ∅
6:  CheckCausality(t_{m+1}, ..., t_n)
7:  endStatePathSet ← statePathSet
8:  for all σ.M_1.M_2 ∈ startStatePathSet do
9:     tokenBag ← M_1(p) − M_2(p)
10:    connectingPlace ← p
11:    t_End ← t_{m+1}
12:    adjacentPathSet ← ∅
13:    CheckAdjacent(M_1)
14:    for all τ.M_3.M_4 ∈ adjacentPathSet do
15:       for all M_5.M_6.υ ∈ endStatePathSet do
16:          if M_3.M_4 = M_5.M_6 then
17:             setOfUsageConflicts ← setOfUsageConflicts ∪ (σ.M_1.M_2, τ.M_3.M_4.υ)
18:          end if
19:       end for
20:    end for
21: end for
22: return setOfUsageConflicts
```

konsumiert. CheckAdjacent liefert – ausgehend vom Zustand $M_1$ (dem Zustand vor dem Schalten von $t_m$) – alle Ausführungspfade zu $t_{m+1}$, in denen diese Transition Marken aus *tokenBag* von der Stelle $p$ entnimmt. Wird ein solcher Pfad gefunden, liegt ein Nutzungskonflikt vor, da $t_{m+1}$ zum Schalten auf Marken in $p$ angewiesen ist, die auch $t_m$ beim Schalten entnimmt.

Die für den aktuell untersuchten *startStatePathSet*-Pfad gefundenen Verletzungen werden als Paare von Ausführungspfaden in der Menge *setOfUsageConflicts* gespeichert. Der erste Pfad in einem dieser Paare beschreibt die Ausführung vom Zustand vor dem Schalten von $t_0$ bis zum Zustand nach dem Schalten von $t_m$. Der zweite Pfad beschreibt die alternative Ausführung vom Zustand vor dem Schalten von $t_{m+1}$ bis nach dem Schalten von $t_n$, die blockiert wird, wenn $t_m$ vor $t_{m+1}$ schaltet.

Nach der Rückkehr von CheckUsageConflict konstruiert CheckStrongUsage aus der Rückgabemenge die Tripel, die Verletzungen eines Kriteriums beschreiben und in diesem Fall in $\mathsf{IF}_{\text{StrongUsage}}$ gespeichert werden. Ein solches Tripel beschreibt hier die Quelle des illegalen Informationsflusses ($t_0$), sein Ziel ($t_n$) sowie sämtliche Pfade vom Anfangszustand bis zur Ausführung von $t_m$ (die die Blockade und damit den Informationsfluss verursacht).

**Algorithmus 4.15** CheckWeakUsage-Prozedur.

**Eingabe:** *markingGraph*, $\mathsf{IF}_{\text{StrongUsage}}$.

1: $\mathsf{IF}_{\text{WeakUsage}} \leftarrow \mathsf{IF}_{\text{StrongUsage}}$
2: **for all** $((t_0, ..., t_m), p, (t_{m+1}, ..., t_n)) \in \mathsf{STR}_{\text{Usage}}$ **do**
3: **for all** $t_c \neq t_{m+1} \in p\bullet$ **do**
4: *setOfUsageConflicts* $\leftarrow$ CheckUsageConflict$(((t_c), p, (t_{m+1}, ..., t_n)))$
5: **for all** $(\sigma.M_1.M_2, \tau.M_3.M_4.\upsilon) \in$ *setOfUsageConflicts* **do**
6: *setOfConflictPaths* $\leftarrow$ TraceBack$(\sigma.M_1.M_2)$
7: **for all** *conflictPath* $\in$ *setOfConflictPaths* **do**
8: **if** $\nexists M_1, M_2 \in$ *conflictPath* : $M_1 \xrightarrow{t_0} M_2$ **then**
9: Entferne alle für die kritische Nutzungsstelle $((t_0, ..., t_m), p, (t_{m+1}, ..., t_n))$ gespeicherten Verletzungen aus $\mathsf{IF}_{\text{WeakUsage}}$
10: **end if**
11: **end for**
12: **end for**
13: **end for**
14: **end for**

Wenn das StrongUsage-Kriterium verletzt ist, d.h. wenn es für eine kritische Nutzungsstelle $((t_0, ..., t_m), p, (t_{m+1}, ..., t_n))$ einen Nutzungskonflikt gibt, kann das schwächere WeakUsage-Kriterium noch immer erfüllt sein. Dies ist der Fall, wenn bei dem Nutzungskonflikt das Schalten einer Ausgangstransition $t_c \in p\bullet$ die Ausführung von $t_{m+1}$ verhindern kann, ohne dass $t_0$ Voraussetzung für $t_c$ wäre.

Das WeakUsage-Kriterium wird von der CheckWeakUsage-Prozedur geprüft, die in Algorithmus 4.15 dargestellt ist und ebenfalls auf die CheckUsageConflict-Prozedur zugreift. CheckWeakUsage prüft für jede Ausgangstransition $t_c$ von $p$, ob es für die durch $((t_c), p, (t_{m+1}, ..., t_n))$ beschriebene Nutzungsstelle einen Nutzungskonflikt gibt. Ist dies der Fall, wird untersucht, ob in den Pfaden durch den Zustandsgraphen, die zu einem Konflikt führen, stets die Ausführung der Transition $t_0$ vorkommt. Das WeakUsage-Kriterium ist (bzgl. der untersuchten kritischen Nutzungsstelle) erfüllt, wenn es mindestens eine Ausführung gibt, in der $t_n$ blockiert wird, ohne dass $t_0$ beteiligt ist.

### Prüfung von StrongCond und WeakCond

Die Prüfung von StrongCond und WeakCond benutzt dasselbe Vorgehen, das zur Prüfung der StrongCausal- und WeakCausal-Kriterien verwendet wird. Der Grund ist, dass es sich sowohl bei kritischen Transitionenfolgen (Menge $\mathsf{STR}_{\text{Causal}}$) als auch bei kritischen bedingten Transitionenfolgen (Menge $\mathsf{STR}_{\text{Cond}}$) um die gleichen syntaktischen Konstrukte handelt, nämlich Pfade durch ein IFnet-Modell.

In beiden Fällen muss geprüft werden, ob es einen kausalen Zusammenhang zwischen der Ausführung der ersten und der letzten Transition eines Pfades geben kann. Der einzige Unterschied besteht darin, dass der ersten Transition im Fall der StrongCond- und WeakCond-Kriterien ein Prädikat zugeordnet ist, das eine `High`-klassifizierte Marke benutzt, während im Fall der StrongCausal- und WeakCausal-Kriterien die Transition selbst `High`-klassifiziert ist. Dieser Unterschied ist lediglich bei der strukturellen Analyse von Bedeutung, macht bei der darauf aufbauenden Analyse des Zustandsgraphen aber keinen Unterschied.

Kapitel 5

# Evaluation

Dieses Kapitel bewertet InDico hinsichtlich seiner Eignung für die Zertifizierung von Isolationsanforderungen in Geschäftsprozessen. Es gliedert sich in drei Teile: Den ersten Teil bildet die Demonstration von InDico in einem Anwendungsszenario. Hier wird eine Geschäftsprozess-Konstellation beispielhaft auf Möglichkeiten zur Verletzung gegebener Sicherheitsanforderungen untersucht. Dazu wird die in BPMN beschriebene Prozess-Konstellation in ein IFnet-Modell übersetzt, mit Sicherheitsstufen ausgezeichnet und anschließend ein Zertifikat generiert. Den zweiten Teil bildet eine qualitative Bewertung, in der die Erfüllung der in Abschnitt 1.4 hergeleiteten Anforderungen an die Zertifizierung untersucht wird. Im letzten Teil werden die Ergebnisse einer Untersuchung der Laufzeit, die für die InDico-Zertifizierung benötigt wird, dargestellt.

## 5.1 Anwendungsszenario: Zertifizierung von E-Auktionen

Um die Anwendung von InDico zu demonstrieren, wird in diesem Beispiel eine Prozess-Konfiguration bzgl. der Einhaltung von Isolationsanforderungen zertifiziert. Die betrachtete Konstellation besteht aus zwei interagierenden Prozessen, die einen Teil einer *Sealed-Bid*-Auktion implementieren. Die Besonderheit einer solchen Auktion ist, dass die Gebote bis zum Ende der Auktion gegenüber den Bietern verschlossen bleiben (Boyd und Mao, 2000).

Im elektronischen Geschäftsverkehr (*E-Commerce*) sind Auktionen ein weit verbreitetes Instrument, um den Preis von Waren und Dienstleistungen zu bestimmen, wie z.B. von Konsumgütern und Werbeanzeigen (Varian, 2007), aber auch von Großhandelswaren wie Elektrizität und Rechenkapazitäten (Buyya u. a., 2002). Um sichere und faire Auktionen zu gewährleisten, ist es entscheidend, die Prozessmodelle bzgl. der Einhaltung entsprechender Richtlinien zu zertifizieren.

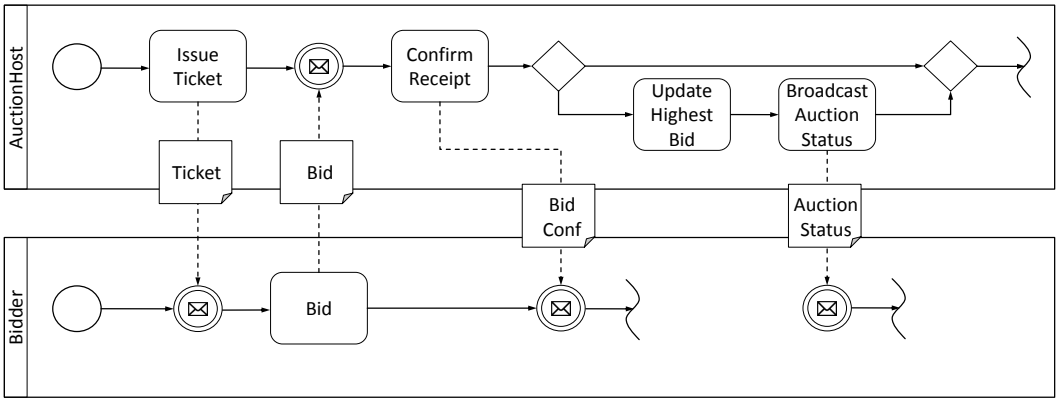



Abbildung 5.1.: BPMN-Fragment eines Auktionsprozesses.

Abbildung 5.1 zeigt eine Prozess-Konfiguration in BPMN-Notation, die an eine Beschreibung aus der Arbeit von Pascalau u. a. (2009) angelehnt ist. Sie modelliert einen Ausschnitt einer Auktion, der zwei Akteure umfasst: den Auktionsbetreiber (AuctionHost) in der oberen Hälfte und einen Bieter (Bidder) in der unteren Hälfte.

Der abgebildete Ausschnitt zeigt die Abgabe und Verarbeitung von Geboten. Dabei signalisiert der Auktionsbetreiber zunächst durch die Ausgabe eines *Tickets* seine Bereitschaft ein Gebot entgegenzunehmen und fordert Bieter zur Abgabe auf. Wenn ein Bieter ein Gebot abgibt, wird ihm dessen Eingang durch den Betreiber bestätigt und von diesem geprüft, ob es das derzeitige Höchstgebot übertrifft. In diesem Fall wird das Höchstgebot erneuert und eine Aktualisierung des Auktionsstatus ausgelöst, über den die Bieter benachrichtigt werden (z.B. durch eine Email oder ein Website-Posting).

InDico wird im Folgenden angewendet, um diese Prozess-Konfiguration bzgl. der Einhaltung der folgenden beiden Sicherheitsanforderungen zu zertifizieren:

1. **Vertraulichkeit des Höchstgebotes.** Da es sich um eine *Sealed-Bid*-Auktion handelt, dürfen Bieter vor dem Ablauf einer Auktion keinerlei Informationen hinsichtlich des aktuellen Höchstgebotes erhalten können.
2. **Isolation von Bieter-Instanzen.** Um eine faire und geheime Auktion zu gewährleisten, müssen mehrere Bieter-Instanzen voneinander isoliert sein, d.h. ein Bieter-Prozess darf keinerlei Einfluss auf andere, nebenläufig ausgeführte, Bieter-Prozesse ausüben können.

### 5.1.1 Übersetzung des BPMN-Modells nach IFnet

Für die Übersetzung des BPMN-Modells nach IFnet wird die in Abschnitt 3.3 entwickelte Abbildungsvorschrift verwendet, die den Kontrollfluss und Teile des Datenflusses (den Nachrichtenaustausch zwischen *Pools*) eines BPMN-Modells abdeckt. Die Subjekt-Attribute der Transitionen werden in Entsprechung zu den Namen der jeweiligen *Pools* gesetzt (AuctionHost im oberen Teil und Bidder im unteren). Abbildung 5.2 zeigt das aus der Übersetzung hervorgegangene IFnet-Modell. Die vollständige Spezifikation des Modells findet sich in Anhang C.

Manuelle Ergänzungen sind an den folgenden Punkten vorgenommen worden, die von dem zugrunde liegenden BPMN-Modell nicht formal beschrieben werden:

- **Gefärbte Marken.** Neben den gefärbten Marken `T`, `B`, `C`, und `S`[1], die durch die Abbildung von BPMN-*Message-Flows* eingeführt worden sind, ist die Marke `H` definiert worden, die ein Datenobjekt zur Speicherung des aktuellen Höchstgebotes repräsentiert. Für ihre Speicherung ist eine zusätzliche Stelle ($p_9$) eingeführt worden.
- **Datenfluss.** Das Ein- und Ausgabeverhalten von Transitionen ist angepasst worden, um den Austausch der Datenobjekte zwischen Transitionen mit gleichem Subjekt-Bezeichner (also innerhalb eines BPMN-*Pools*) zu modellieren.
- **Datenzugriffsmodi.** Für jede Transition, die eine gefärbte Marke einliest oder erzeugt, ist der entsprechende Zugriffsmodus spezifiziert worden.
- **Prädikate.** Die Transitionen Branch1 und Branch2 sind mit Prädikaten versehen worden, die die jeweilige Verzweigungsbedingung modellieren und eine Abhängigkeit von Datenobjekt `H` ausdrücken.

[1] Diese Marken stehen für Tickets, Gebote, Bestätigungen und Statusnachrichten (in dieser Reihenfolge).

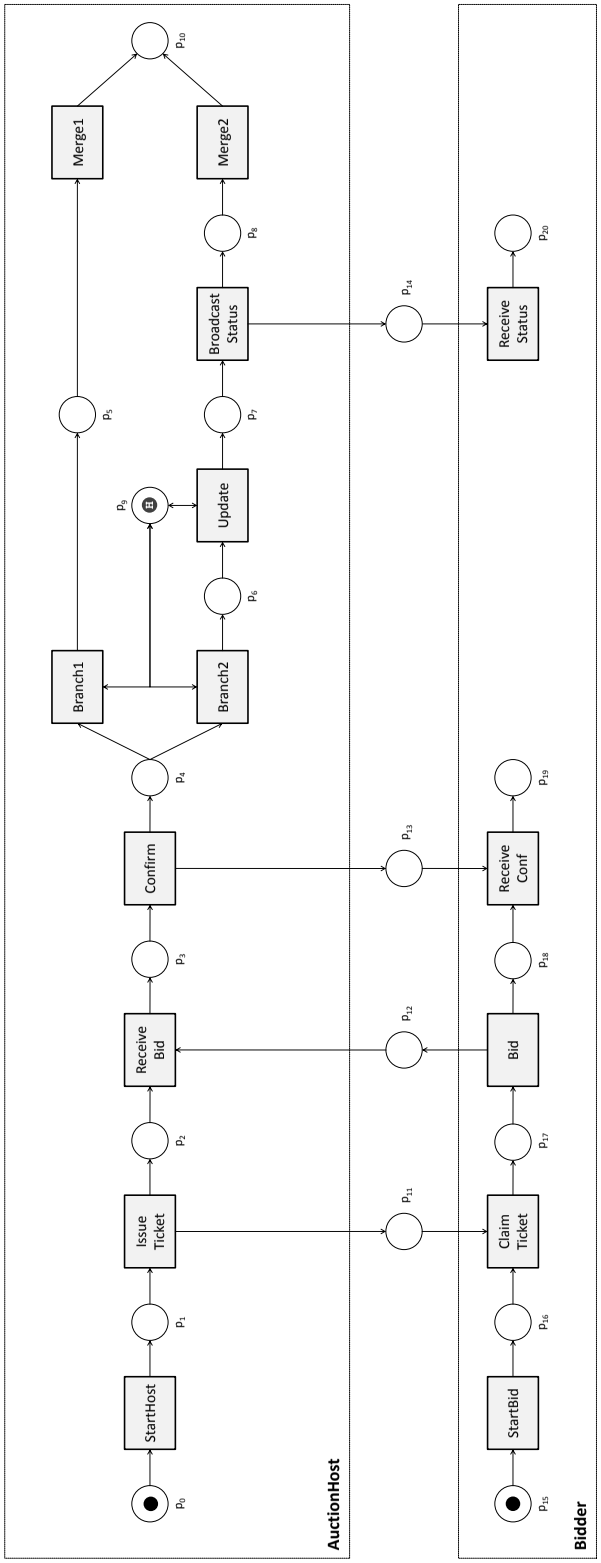



Abbildung 5.2.: Nach IFnet übersetztes Modell des BPMN-Prozesses aus Abbildung 5.1.

| | H |
|---|---|
| AuctionHost | *read, write* |
| Bidder | – |

Tabelle 5.1.: Darstellung der Isolationsanforderung durch eine Zugriffskontrollmatrix.

### 5.1.2 Zertifizierung: Vertraulichkeit des Höchstgebotes

Die erste Isolationsanforderung drückt aus, dass Bieter vor dem Auktionsende keine Informationen bzgl. des aktuellen Höchstgebotes erhalten dürfen. Dieser Umstand wird durch die in Tabelle 5.1 dargestellte Zugriffskontrollmatrix beschrieben, die als Grundlage für die Verwendung der ACMLabeler-Strategie genutzt wird.

Die Anwendung von ACMLabeler mit dieser Zugriffskontrollmatrix auf das in Abbildung 5.2 dargestellte IFnet ergibt die folgende Auszeichnung:

- Für eine gefärbte Marke $c \in \mathcal{C}_c$ ist:

$$\textsc{Label}_{\mathcal{SC}}(c) = \begin{cases} \texttt{High} & \text{falls} \quad c = \texttt{H}. \\ \texttt{unlabeled} & \text{falls} \quad c \neq \texttt{H}. \end{cases}$$

- Für die Subjekte gilt:

$$\textsc{Clear}_{\mathcal{SC}}(\mathsf{AuctionHost}) = \texttt{High},$$
$$\textsc{Clear}_{\mathcal{SC}}(\mathsf{Bidder}) = \texttt{Low}.$$

In dem so ausgezeichneten IFnet ist die einzige höher klassifizierte Entität die Marke H, die das aktuelle Höchstgebot speichert. Information aus dieser Marke darf nicht an das Subjekt Bidder fließen, das eine niedrigere Freigabe besitzt.

#### Statische Prüfung

Die statische Prüfung zeigt als Ergebnis eine Verletzung des StrongCond-Kriteriums an. Die folgende kritische bedingte Transitionenfolge ist – ausgehend vom Anfangszustand – ausführbar und führt zu einer Informationsübertragung von H zu Bidder:

$$\mathsf{Update} \rightarrow \mathsf{p}_6 \rightarrow \mathsf{Broadcast} \rightarrow \mathsf{p}_{12} \rightarrow \mathsf{ReceiveStatus}.$$

Da es in diesem Fall keine weitere Schaltfolge gibt, in der ReceiveStatus, aber nicht Update ausgeführt wird, wird auch das abgeschwächte WeakCond-Kriterium verletzt.

| Zertifikat für IFnet $N$ im Anfangszustand $M$. | | | |
|---|---|---|---|
| Geprüfte Kriterien: | NoRead, NoWriteHigh, NoWriteNeutral, StrongCausal, WeakCausal, StrongUsage, WeakUsage, StrongCond, WeakCond. | | |
| Verletzungen: | | | |
| | Quelle | Ziel | Schaltfolgen |
| StrongCond | H | ReceiveStatus | StartHost → StartBid → Bid → ReceiveBid → Confirm → Branch2 → Update → Broadcast → ReceiveStatus; ... |
| WeakCond | H | ReceiveStatus | StartHost → StartBid → Bid → ReceiveBid → Confirm → Branch2 → Update → Broadcast → ReceiveStatus; ... |

Tabelle 5.2.: Zertifikat zur ersten Isolationsanforderung.

### Zertifikat und Bewertung

Als Ergebnis der statischen Prüfung generiert InDico ein Zertifikat, das in Tabelle 5.2 dargestellt ist.

Es enthält zu Anfang einen Verweis auf das geprüfte IFnet und den Anfangszustand sowie eine Liste der Kritieren bzgl. derer das Modell untersucht wurde. Im Anschluss werden die gefundenen Verletzungen charakterisiert. Hier werden die Quelle des illegalen Informationsflusses (in diesem Fall die gefärbte Marke H), sein Ziel (die Transition ReceiveStatus) sowie sämtliche Schaltfolgen, die zu der Verletzung führen, aufgelistet (aus Platzgründen wird in der Abbildung jeweils nur eine Schaltfolge aufgeführt). In diesem Fall sind die Einträge für Verletzungen des StrongCond-Kriteriums und des WeakCond-Kriteriums identisch.

Das Zertifikat dient als Grundlage für die Sicherheitsbewertung der gefunden Informationsflüsse. In diesem Fall ist die identifizierte Informationsübertragung in hohem Maße sicherheitsrelevant und zeigt einen Design-Fehler im Auktionsprozess auf: Die Aktualisierung des Auktionsstatus, der von den Bietern aus sichtbar ist, setzt die vorhergehende Ersetzung des aktuellen Höchstgebotes voraus, d.h. der Status wird nur dann aktualisiert, wenn ein neues Höchstgebot verbucht worden ist. Diese Schwachstelle kann ein Bieter ausnutzen, um den Wert des aktuellen Höchstgebotes beliebig genau zu approximieren, indem er – ausgehend von einem niedrigen Startgebot – seine Gebote sukzessive bis zu dem Punkt erhöht, an dem eine Aktualisierung des Auktionsstatus ausgelöst wird. Die Vertraulichkeit des aktuellen Höchstgebotes kann auf diese Weise gebrochen werden.

### 5.1.3 Zertifizierung: Isolation von Bieter-Instanzen

Die zweite Isolationsanforderung verlangt, dass mehrere Bieter-Instanzen isoliert voneinander sein müssen, d.h. dass es keinen Informationsfluss von einem Bieter-Prozess zu einem anderen Bieter-Prozess geben darf. Für die Kodierung dieser Anforderung wird die MultiInstanceLabeler-Strategie angewendet.

Für die Anwendung von MultiInstanceLabeler ist die Festlegung mehrerer Parameter notwendig. Zunächst wird das Teilnetz $O$ definiert, das den Bieter-Prozess modelliert und durch die Transitionen $T_O = \{\mathsf{StartBid}, \mathsf{Bid}, \mathsf{ReceiveConf}, \mathsf{ReceiveUpdate}\}$ gegeben ist. Die Abbildung $\Pi$ legt fest, welche Datenobjekte von den Bieter-Prozessen gemeinsam genutzt bzw. für welche Datenobjekte in jeder Prozess-Instanz eigene Marken verwendet werden. Im Falle des Auktionsprozesses wird ausschließlich die Marke `T`, die das vom Auktionsbetreiber herausgegebene *Ticket* repräsentiert, von mehreren Bieter-Prozessen verwendet, während für die übrigen Datenobjekte (z.B. Gebote) jeweils unterschiedliche Marken verwendet werden. Der Grund ist, dass über ein *Ticket* der Zugang zur Gebotsabgabe geregelt wird, d.h. nur der Bieter, der ein *Ticket* entgegennimmt, kann anschließend ein Gebot abgeben. Insofern gilt $\Pi(\mathtt{T}) = \mathtt{T}$, während die Marken `B`, `C` und `S` von $\Pi$ auf jeweils andere, bisher nicht verwendete Marken abgebildet werden.

Anschließend dupliziert MultiInstanceLabeler das Teilnetz $O$ und verbindet das Duplikat mit dem ursprünglichen Netz. Für das Subjekt, das den zweiten (duplizierten) Bieter-Prozess ausführt, wird die Bezeichnung Bidder2 verwendet. Abbildung 5.3 zeigt die Struktur des IFnets $(N^+, M^+)$, das aus dem ursprünglichen Modell durch die Duplizierung der Bieter-Instanz hervorgegangen ist. Die vollständige Spezifikation dieses Netzes findet sich in Anhang C.

Nach der Erzeugung von $(N^+, M^+)$ klassifiziert MultiInstanceLabeler die Transitionen und Marken und ordnet den Subjekten Freigaben zu:

- Für eine Transition $t \in T^+$ gilt:

$$\textsc{Label}_{\mathcal{SC}}(t) = \begin{cases} \mathtt{High} & \text{falls} \quad \mathrm{S}_{\mathcal{U}}(t) = \mathsf{Bidder}. \\ \mathtt{Low} & \text{falls} \quad \mathrm{S}_{\mathcal{U}}(t) = \mathsf{Bidder2}. \\ \mathtt{Low} & \text{falls} \quad \mathrm{S}_{\mathcal{U}}(t) = \mathsf{AuctionHost}. \end{cases}$$

- Die einzige im Anfangszustand $M^+$ vorhandene gefärbte Marke ist `H`. Da diese zu keinem Bieter-Prozess gehört, erhält sie die Einstufung `Low`:

  $\textsc{Label}_{\mathcal{SC}}(\mathtt{H}) = \mathtt{Low}$.

- Die Freigaben der Subjekte sind:

  $\textsc{Clear}_{\mathcal{SC}}(\mathsf{Bidder}) = \mathtt{High}$.
  $\textsc{Clear}_{\mathcal{SC}}(\mathsf{Bidder2}) = \mathtt{Low}$.
  $\textsc{Clear}_{\mathcal{SC}}(\mathsf{AuctionHost}) = \mathtt{neutral}$.

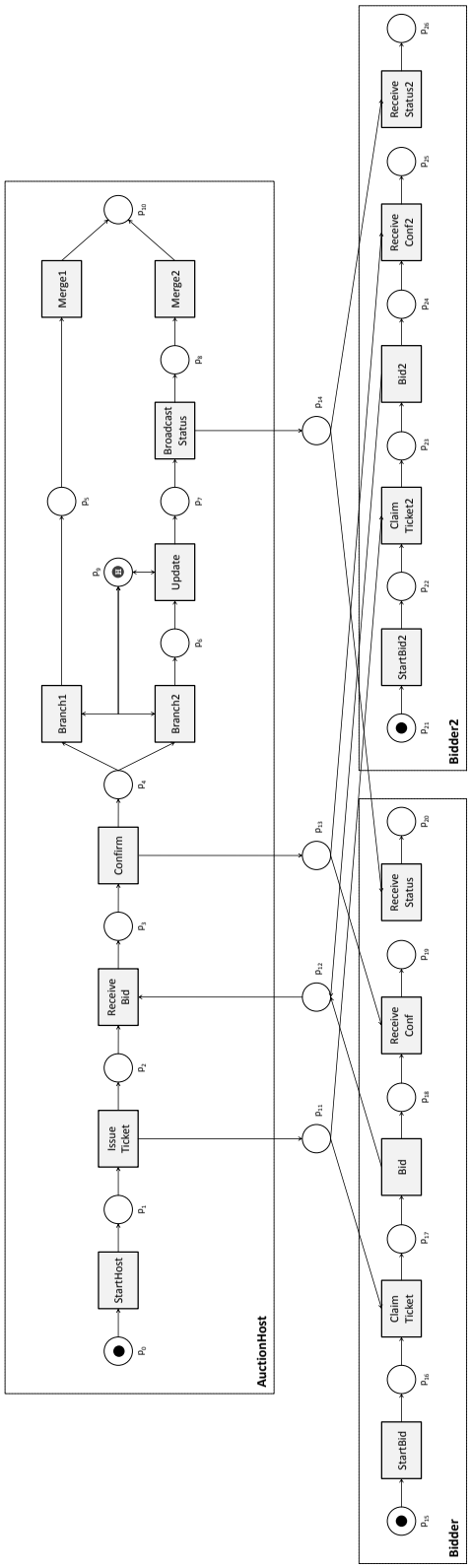



Abbildung 5.3.: IFnet-Modell des Auktionsprozesses mit zwei Bieter-Instanzen.

| Zertifikat für IFnet $N^+$ im Anfangszustand $M^+$. | | | |
|---|---|---|---|
| Geprüfte Kriterien: | NoRead, NoWriteHigh, NoWriteNeutral, StrongCausal, WeakCausal, StrongUsage, WeakUsage, StrongCond, WeakCond. | | |
| Verletzungen: | | | |
| | Quelle | Ziel | Schaltfolgen |
| StrongUsage | ClaimTicket | ClaimTicket2 | StartHost → IssueTicket → StartBid → ClaimTicket → StartBid2 → ClaimTicket2; ... |
| WeakUsage | ClaimTicket | ClaimTicket2 | StartHost → IssueTicket → StartBid → ClaimTicket → StartBid2 → ClaimTicket2; ... |

Tabelle 5.3.: Zertifikat zur zweiten Sicherheitsanforderung.

### Statische Prüfung

Die einzigen höher klassifizierten Entitäten in dem in Abbildung 5.3 dargestellten IFnet sind die Transitionen, die die Aktionen des ersten Bieters `Bidder` modellieren. Der zweite Bieter `Bidder2` verfügt über eine niedrigere Freigabe und darf daher keine Informationen bzgl. `Bidder` erhalten.

Die statische Prüfung identifiziert eine illegale Informationsübertragung an der folgenden kritischen Nutzungsstelle:

$$\mathsf{ClaimTicket} \leftarrow \mathsf{p}_{11} \rightarrow \mathsf{ClaimTicket2}.$$

Schaltet die Transition ClaimTicket, entnimmt sie die Marke `T`, die das *Ticket* repräsentiert, aus der Stelle $\mathsf{p}_{11}$, die vom Auktionsbetreiber dort platziert worden ist. Da in jedem Durchlauf immer nur ein *Ticket* ausgestellt wird, ist die Transition ClaimTicket2 entsprechend lange blockiert. Das Netz verletzt somit das StrongUsage-Kriterium und – weil die Blockade immer von derselben Transition ClaimTicket ausgeht – auch das abgeschwächte WeakUsage-Kriterium.

### Zertifikat und Bewertung

Tabelle 5.3 zeigt das Zertifikat, das als Ergebnis der Prüfung generiert wird. Es zeigt wie im vorherigen Fall die Art der gefundenen Verletzungen und charakterisiert diese durch

das Tripel aus der Quelle des illegalen Informationsflusses, seiner Senke und der Menge der Ausführungspfade, bei denen die Verletzung auftritt.

Die gefundene Verletzung kann in zweierlei Hinsicht sicherheitsrelevant sein. Zum Einen erfährt Bidder2 wann (und ggf. wie häufig) Bidder Gebote abgibt und kann sein eigenes Bietverhalten entsprechend darauf abstimmen. Zum Anderen kann ein Bieter den anderen Bieter gezielt an der Abgabe von Geboten hindern (etwa unmittelbar vor dem Ende einer Auktion) und somit Einfluss auf den Ablauf der Auktion nehmen. Die Bieter-Instanzen sind in diesem Sinne nicht voneinander isoliert.

## 5.2 Bewertung von IFnet

In diesem Abschnitt wird IFnet hinsichtlich seiner Eignung für die Modellierung von Geschäftsprozessen bewertet. Das Beispielszenario zeigt bereits, dass IFnet ausreichend mächtig ist, um die gegebene BPMN-Prozesskonstellation darzustellen. Um die Bewertung auf eine breitere Grundlage zu stellen, wird IFnet im Folgenden mit *Workflow Patterns* verglichen, einer Sammlung von in der Praxis verwendeten Mustern zur Geschäftsprozess-Beschreibung. Darüber hinaus werden die Darstellung der Datenperspektive und das Sicherheitsmodell diskutiert und zu bestehenden Ansätzen in Beziehung gesetzt.

### 5.2.1 Vergleich mit *Workflow Patterns*

Ein Prozess-Modell beschreibt zunächst die Struktur eines geschäftlichen Ablaufes, d.h. die Anordnung der Aktivitäten und ihre Abhängigkeiten bei der Ausführung. Diese *strukturelle* Betrachtung ist die weitaus geläufigste Perspektive auf einen Prozess, auf die sich die meisten Modellierungssprachen beschränken (Sun u. a., 2006).

Für die diesbezügliche Bewertung von IFnet werden die sogenannten *Workflow Patterns* von van der Aalst u. a. (2003a) herangezogen, die eine sprachenunabhängige Kollektion von Mustern definieren, aus denen sich Geschäftsprozesse zusammensetzen lassen. Seit ihrer Einführung haben sie sich als ein Instrument zur Bewertung und zum Vergleich von Prozess-Modellierungssprachen etabliert.

Die *Workflow Patterns* haben den Anspruch, eine möglichst vollständige Sammlung der für die Prozessmodellierung benötigten Konstrukte bereitzustellen, und umfassen auch weniger geläufige Muster, die nur selten gebraucht werden. So kann keine der in der Arbeit von van der Aalst u. a. (2003a) untersuchten industriellen Modellierungssprachen alle *Patterns* darstellen, die meisten sind sogar nur auf eine relativ geringe Anzahl beschränkt. Für die Bewertung von IFnet werden daher nur die Muster herangezogen, die von mindestens der Hälfte der untersuchten Sprachen dargestellt werden können.[2] Indem gezeigt wird, dass diese auch mit IFnet modelliert werden können, wird die Annahme begründet, dass sich IFnet auch für die Modellierung der meisten in der Praxis verwendeten Prozesse verwenden lässt. Im Folgenden wird die Darstellung dieser *Patterns* in IFnet gezeigt.

***Sequence*** Das *Sequence*-Muster beschreibt eine Abfolge mehrerer Aktivitäten, in der eine Aktivität erst dann ausgeführt werden kann, nachdem die Ausführung der jeweils davor auftretenden Aktivität geendet hat. Ein Beispiel für dieses Muster findet sich in

[2] Die Auswahl bezieht sich auf den Vergleich industrieller Modellierungssprachen im Anhang der Arbeit zu *Workflow Patterns* (van der Aalst u. a., 2003a). Es werden nur die *Patterns* herangezogen, die von mindestens 8 der 15 untersuchten Sprachen unterstützt werden.

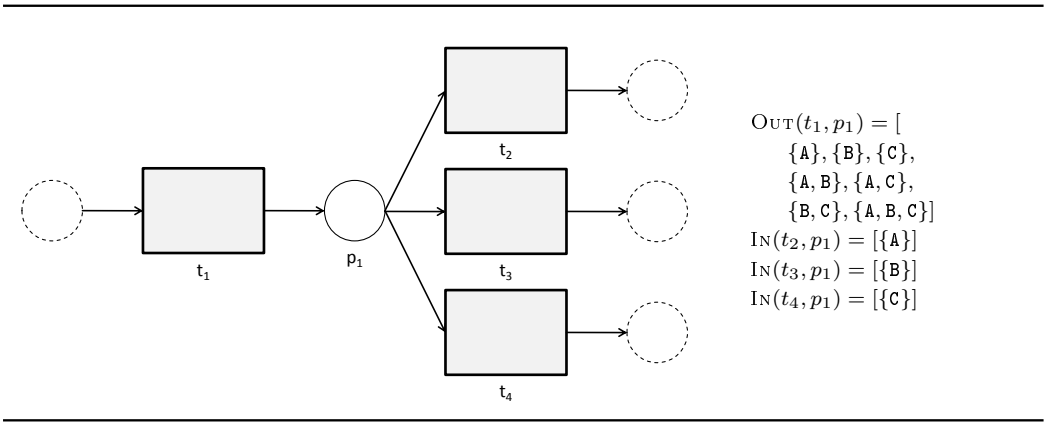



Tabelle 5.4.: Beispiel für ein *Multi-Choice*-Muster in IFnet.

dem in Abbildung 5.2 dargestellten IFnet-Modell: Die Ausführung von IssueTicket kann erst dann beginnen, wenn StartHost geendet hat.

***Parallel Split*** Das *Parallel Split*-Muster bezeichnet ein Geschäftsprozess-Konstrukt, das einen einzelnen Kontrollfaden in mehrere parallele Konstrollfäden auffächert, die gleichzeitig oder in einer beliebigen Abfolge ausgeführt werden können. Seine Modellierung in IFnet wird in der Abbildung des entsprechenden BPMN-Konstruktes (*Parallel-Gateway*) in Tabelle 3.3 gezeigt.

***Synchronization*** Das *Synchronization*-Muster bezeichnet ein Konstrukt, in dem mehrere parallel ausgeführte Kontrollfäden synchronisiert werden und in einen einzelnen Kontrollfaden münden. Es wird oft als Gegenstück zu dem *Parallel Split*-Muster verwendet. Seine Modellierung in IFnet wird in der Abbildung des entsprechenden BPMN-Konstruktes (*Join*) in Tabelle 3.3 gezeigt.

***Exclusive Choice*** Das *Exclusive Choice*-Muster bezeichnet ein Konstrukt, in dem einer von mehreren alternativen Ausführungspfaden gewählt wird. Es wird in dem in Abbildung 5.2 gezeigten IFnet-Modell verwendet, um die Verzweigung zwischen dem Ausführungspfad, der mit der Transition Branch1 beginnt, und dem Ausführungspfad, der mit der Transition Branch2 beginnt, zu modellieren.

***Simple Merge*** Das *Simple Merge*-Muster ist das Gegenstück zu *Exclusive Choice* und bezeichnet ein Konstrukt, in dem zwei oder mehrere alternativ ausgeführte Pfade zusammengeführt werden. Es findet sich in dem IFnet aus Abbildung 5.2 bei der Vereinigung der beiden Ausführungspfade, die mit Merge1 bzw. mit Merge2 enden.

***Multi-Choice*** Das *Multi-Choice*-Muster bezeichnet ein Konstrukt, in dem von mehreren Ausführungspfaden einige gewählt werden. Dieses kann in IFnet einerseits analog zum *Exclusive Choice*-Muster umgesetzt werden. Im Fall des *Multi-Choice* muss die letzte Transition vor der Verweigung anstatt einer (schwarzen) Marke so viele schwarze Marken produzieren, wie Ausführungspfade angestoßen werden sollen. Um die Pfade, die angestoßen werden sollen, exakt spezifizieren zu können, ist alternativ eine Umsetzung mit gefärbten Marken möglich. Ein Beispiel ist in Tabelle 5.4 abgebildet: Jeder der Ausführungspfade beginnt mit einer Transition, die eine andere Marke aus der Stelle $\mathsf{p}_1$ benötigt, um ausgeführt werden zu können. Abhängig davon, welche Kombination von Marken die Transition $\mathsf{t}_1$ in $\mathsf{p}_1$ produziert, ist eine Teilmenge der Transitionen $\{ \mathsf{t}_1, \mathsf{t}_2, \mathsf{t}_3 \}$ schaltbereit.

***Multiple Instances with a Priori Design-Time Knowledge*** Das *Multiple Instances with a Priori Design-Time Knowledge*-Muster bezeichnet eine Situation in einem Geschäftsprozess, in der dieselbe Aktivität mehrfach wiederholt wird und die Anzahl der Wiederholungen bereits in der Entwurfsphase des Prozesses bekannt ist. Eine einfache Methode um dieses Muster zu realisieren – die auch in IFnet unmittelbar umgesetzt werden kann – ist die Replikation der entsprechenden Aktivität im Prozess-Modell und die parallele Ausführung und anschließende Synchronisation der Replikate mit Hilfe der *Parallel Split*- und *Synchronization*-Muster (van der Aalst u. a., 2003a).

### 5.2.2 Datenperspektive

Die Datenperspektive beschreibt den Fluss von Datenobjekten während der Ausführung eines Geschäftsprozesses. Für die mit InDico zertifizierbaren Sicherheitsanforderungen ist ihre Darstellung notwendig um zu entscheiden, ob es in einem Prozess zu illegalen Informationsflüssen durch den Austausch von Datenobjekten kommen kann.

Im Hinblick auf die Datenperspektive ist in BPMN ausschließlich die Darstellung des Nachrichtenaustausches zwischen verschiedenen Prozessen (*pools*) möglich (OMG, 2009). Dieser kann in IFnet durch ein entsprechendes Konstrukt, wie in Tabelle 3.4 gezeigt, direkt abgebildet werden. Ein Beispiel dafür findet sich an mehreren Stellen in dem IFnet-Modell aus Abbildung 5.2, etwa bei der Kommunikation zwischen den Transitionen Bid und ReceiveBid.

Um auch den Datenaustausch zwischen Aktivitäten *innerhalb* eines Prozesses darzustellen, wird in IFnet dasselbe Konstrukt verwendet: Ein gefärbte Marke, die das Datenobjekt repräsentiert, wird von der sendenden Transition in einer Stelle produziert, aus der es von der empfangenden Transition konsumiert wird. Damit wird der Fluss eines Datenobjektes direkt durch den Austausch der entsprechenden gefärbten Marke repräsentiert. In dem IFnet-Modell aus dem Anwendungsszenario wird dieses Konzept etwa bei der Weitergabe der Marke, die ein Gebot repräsentiert (von Transition ReceiveBid zu Transition Confirm), umgesetzt.

Diese Darstellung unterscheidet sich von den meisten anderen Modellierungsansätzen, die einer Aktivität lediglich die benutzten Datenobjekte als Annotation zuordnen, ohne deren Weitergabe explizit zu modellieren (siehe Abschnitt 2.3.2). Hier wird implizit die Existenz eines gemeinsamen Speichers angenommen, auf den alle Aktivitäten Zugriff haben.

Dieser Ansatz ist für IFnet unzureichend, da die exklusive Nutzung (Blockierung) eines Datenobjektes nicht dargestellt werden kann. Im Gegensatz zu anderen Verfahren betrachtet InDico Informationsübertragungen, die durch verdeckte Kanäle hervorgerufen werden. Diese entstehen beispielsweise durch die wechselseitige Blockade von Ressourcen, wie im Anwendungsszenario gezeigt: Ein Bieter kann Informationen über das Verhalten eines anderen Bieters erhalten, weil sie sich gegenseitig bei der Konkurrenz um ein *Ticket* behindern. Derartige Situationen können mit den in Abschnitt 2.3.2 diskutierten Verfahren nicht modelliert werden.

Die explizite Modellierung des Datenflusses wie in IFnet stellt keine Beschränkung im Vergleich zu den in Abschnitt 2.3.2 beschriebenen Verfahren dar. Es kann auch auf diese Weise ein gemeinsamer Speicher, der von einer Gruppe von Aktivitäten genutzt wird, modelliert werden. So sind in dem Modell in Abbildung 5.2 die drei Transitionen Branch1, Branch2 und Update an die Stelle $p_9$ angeschlossen, aus der sie beim Schalten die Marke `H` konsumieren und anschließend wieder dort platzieren. IFnet erweitert somit die Modellierbarkeit des Datenflusses in einem Prozess.

### 5.2.3 Sicherheitsmodell

Das Sicherheitsmodell von InDico baut auf den Arbeiten von Denning (1976) und Bell und LaPadula (1976) auf. Hier werden Datenobjekten Sicherheitsstufen zugeordnet, die hierarchisch angeordnet sind. Subjekte erhalten den Sicherheitsstufen entsprechende Freigaben, die festlegen, auf welche Datenobjekte zugegriffen werden darf bzw. welcher Sicherheitsstufe neu erzeugte Datenobjekte zugeordnet werden. Die Umsetzung in dem Modell von Bell und LaPadula (1976) garantiert, dass es keine Informationsübertragung durch Datenobjekte von einer höher klassifizierten Entität (Datenobjekt oder Subjekt) zu einer Entität mit niedrigerer Freigabe geben kann.

IFnet benutzt ein Sicherheitsmodell, das das Modell von Bell und LaPadula im Wesentlichen in zwei Punkten erweitert:

1. Aktivitäten eines Geschäftsprozesses (Transitionen) werden als potentielle Träger von Informationen behandelt und können ebenfalls klassifiziert werden.

2. Es gibt eine neutrale Freigabe (`neutral`), die Subjekten zugeordnet wird, die keine eigenen Sicherheitsinteressen verfolgen. Ein neutrales Subjekt darf sämtliche Informationen erhalten (es hätte in diesem Sinne die Freigabe `High`), neu erzeugte Datenobjekte werden aber mit der Klassifikation `Low` versehen.

Während die erste Erweiterung durch die Betrachtung verdeckter Kanäle begründet ist, ist die Einführung eines neutralen Nutzers im Hinblick auf viele Geschäftsprozess-Konstellationen sinnvoll: Hier entspricht ein neutrales Subjekt etwa einem Dienstanbieter, der mit mehreren Kunden interagiert, die jeweils unterschiedliche Sicherheitsinteressen haben.

Ein Beispiel für einen solchen Dienstanbieter ist der Auktionsbetreiber AuctionHost in dem in Abbildung 5.3 dargestellten IFnet-Modell. AuctionHost interagiert hier mit zwei Bietern, die unterschiedliche Freigaben besitzen. So muss AuctionHost `High`-Datenobjekte (wie z.B. ein Gebot) von Bidder1 entgegennehmen können. Gleichzeitig sollen die von AuctionHost selbst erzeugten Datenobjekte (wie z.B. der Auktionsstatus) nicht als `High` klassifiziert werden, da sie legal an Bidder2 (mit `Low`-Freigabe) fließen dürfen.

Beide Erweiterungen stellen – verglichen mit dem traditionellen Bell-LaPadula-Modell – echte Ergänzungen dar ohne die Ausdrucksmächtigkeit des ursprünglichen Modells einzuschränken, da ihre Verwendung optional und nicht obligatorisch ist. Sie ermöglichen damit eine Beschreibung von Sicherheitsanforderungen, die mit dem ursprünglichen Modell – wie im Fall des Anwendungsszenarios – nicht ausgedrückt werden können.

## 5.3 Automatisierbarkeit der Zertifizierung

Die Automatisierbarkeit des Zertifizierungsvorgangs ist eine wesentliche Voraussetzung für die praktische Anwendbarkeit eines Verfahrens. InDico schlägt für zwei mit der Analyse des Prozessmodells einhergehende Arbeitsschritte – die Übersetzung aus einer Prozessbeschreibung und die Kodierung von Sicherheitsanforderungen – Ansätze für deren teilweise Automatisierung vor, die in diesem Abschnitt diskutiert werden.

Die Forschung zu Sicherheitsanalysen konzentriert sich gegenwärtig vor allem auf die Automatisierung der Analyseverfahren an sich. So betrachtet keiner der in Abschnitt 2.3 diskutierten Ansätze die Einbettung der Verfahren in praktische Anwendungsszenarien. Zwar gibt es separate Arbeiten zur Übersetzung von industriellen Geschäftsprozess-Modellierungssprachen wie BPMN in formale Modelle wie Petri-Netze oder Prozesskalküle (siehe Abschnitt 2.2.1), jedoch sind diese nicht auf die Analyseverfahren abgestimmt, die in der Regel auf modifizierten Varianten aufsetzen.

Das Vorgehensmodell von InDico – mitsamt den für die Umsetzung benötigten Komponenten – erweitert den Stand der Forschung somit nicht nur hinsichtlich der eigentlichen Analyse (siehe die Bewertung in Abschnitt 5.4), sondern auch bzgl. der Anwendungsunterstützung. Im Folgenden werden die Möglichkeiten und Grenzen der Automatisierung, wie sie von InDico vorgeschlagen werden, untersucht.

**Übersetzung von BPMN-Prozessmodellen** BPMN ist in den letzten Jahren zu einem de-facto-Standard für die Modellierung von Geschäftsprozessen geworden (Pfitzner u. a., 2009). Wie durch die Abbildungsvorschrift aus Abschnitt 3.3 gezeigt wird, kann die Übersetzung der wesentlichen strukturellen Elemente nach IFnet vollständig automatisiert werden. Weitere Konstrukte – wie z.B. der Datenfluss und Verzweigungsbedingungen – müssen dagegen von Hand modelliert werden (vgl. Abschnitt 3.3).

Der Grund dafür liegt in den unterschiedlichen Abstraktionsebenen, die von BPMN und IFnet für die Modellierung verwendet werden. Während BPMN als eher abstrakte Entwurfssprache konzipiert ist, die sich an alle bei der Prozessgestaltung beteiligten Parteien richtet (Recker und Mendling, 2006), benötigt InDico für die Sicherheitsanalyse zusätzliche Informationen, die in einem BPMN-Modell nicht bzw. allenfalls als informelle Annotationen vorhanden sind und deshalb von Hand ergänzt werden müssen. Mit der Bereitstellung dieser Informationen in formaler Form wäre eine weitergehende Automatisierung des Übersetzungsvorgangs denkbar.

**Kodierung von Anforderungen** InDico schlägt die automatisierte Kodierung von Sicherheitsanforderungen mit Hilfe spezialisierter Strategien dar. Diese übersetzen jeweils eine bestimmte Klasse von Anforderungen aus einer Eingangsdarstellung in das von InDico verwendete mehrstufige Sicherheitsmodell. Die beiden im Rahmen dieser Arbeit vorgeschlagenen Strategien – ACMLabeler und MultiInstanceLabeler – demonstrieren das

Konzept anhand zweier häufig auftretender Klassen von Anforderungen, die auch im Anwendungsbeispiel in Abschnitt 5.1 verwendet werden.

Die ACMLabeler-Strategie übersetzt Vertraulichkeitsanforderungen an Datenobjekte, die in einer Zugriffskontrollmatrix gespeichert sind, als Einschränkungen an den Informationsfluss in einem IFnet-Modell. Sie deckt damit eine vergleichsweise umfangreiche Klasse von Sicherheitsanforderungen ab, da der Schutz von Datenobjekten in IT-Systemen Bestandteil von praktisch jeder Sicherheitsrichtlinie ist (Breaux und Antón, 2008). Zugriffsrichtlinien sind das zentrale Instrument für ihre Spezifikation und finden sich in verschiedenen Formen auf allen Ebenen eines IT-Systems (Benantar, 2010).

Die MultiInstanceLabeler-Strategie ist komplexer, da sie zusätzlich zur Auszeichnung eine Transformation des IFnet-Modells durchführt um die nebenläufige Ausführung mehrerer Teilprozesse darzustellen. Die Isolation parallel ausgeführter Anwendungen, wie sie durch MultiInstanceLabeler für Geschäftsprozesse kodiert wird, ist eine zentrale Sicherheitsanforderung in gemeinsam genutzten Infrastrukturen (BSI, 2010; ENISA, 2010; Jansen und Grance, 2011).

Die Verwendung von MultiInstanceLabeler im Anwendungsszenario zeigt, dass die Automatisierung der Auszeichnung ebenfalls von der Bereitstellung der notwendigen Informationen durch das BPMN-Ausgangsmodell abhängt. Weil der BPMN-Prozess nicht definiert, wie sich mehrere Teilprozesse beim Austausch von Daten verhalten, muss diese Information manuell durch die Anpassung der Abbildung $\Pi$ eingebracht werden. Die Ausgestaltung der Abbildung definiert, ob der Host-Prozess mit mehreren Client-Prozessen gleichzeitig, oder – wie hier im Fall der *Ticket*-Ausgabe – sequentiell interagiert.

Durch die Vielzahl denkbarer Sicherheitsanforderungen ist eine umfassende Automatisierung des Auszeichnungsvorgangs – d.h. die Angabe einer begrenzten Anzahl von Strategien die sämtliche Anforderungen abdeckt – kaum möglich. Vielmehr erscheint die kontextabhängige Definition spezialisierter Strategien sinnvoll, deren Ausgestaltung von dem Einsatzszenario eines Prozesses abhängig gemacht wird.

## 5.4 Isolationsgarantien

Die Informationsflusskriterien von InDico identifizieren illegale Informationsflüsse von `High`-klassifizierten Informationen zu Subjekten mit `Low`-Freigabe. In diesem Abschnitt werden die mit diesen Kriterien erreichten Isolationsgarantien bewertet, zunächst hinsichtlich des Datenflusses und anschließend in Bezug auf verdeckte Informationsübertragung.

### 5.4.1 Datenfluss

Das Ziel der Verwendung der drei Datenflusskriterien NoRead, NoWriteHigh und NoWriteNeutral ist es, genau die IFnet-Modelle zu identifizieren, bei denen keine Information aus einem `High`-klassifizierten Datenobjekt durch direkten Zugriff an ein Subjekt mit `Low`-Freigabe oder in ein anderes `Low`-klassifiziertes Datenobjekt fließen kann.

Weil ein illegaler Datenfluss stets von einzelnen Transitionen ausgelöst wird, kann die Erreichung dieses Ziels durch eine Fallunterscheidung gezeigt werden: Hier werden beispielhaft für eine Transition $t$ die Freigabe des ausführenden Subjektes, die Klassifikation der Eingabemarken und die Zugriffsmodi variiert und gezeigt, dass entweder kein illegaler Datenfluss auftreten kann oder aber dieser durch eines der Datenflusskriterien erkannt wird. Zur Illustration zeigt Tabelle 5.5 eine solche Transition $t$, die beim Schalten drei Marken einliest und – zusammen mit einer zusätzlichen vierten Marke – wieder ausgibt.

In der folgenden Aufzählung werden zunächst die Freigabe des ausführenden Subjektes und davon abhängig die Klassifikation der Eingabemarken und die Zugriffsmodi verändert. Für jede Kombination wird untersucht, ob ein illegaler Datenfluss auftreten kann und welches Datenflusskriterium anwendbar ist.

1. $t$ hat Freigabe `High` ($\textsc{Clear}_{\mathcal{SC}}(\mathrm{S}_{\mathcal{U}}(t)) = \texttt{High}$).
   a) Für `High`-klassifizierte Eingabemarken gilt:

      Der Zugriff (schreibend und lesend) ist zulässig ($\texttt{High} \rightsquigarrow \texttt{High}$).

   b) Für `Low`-klassifizierte Eingabemarken gilt:

      – Der Lesezugriff ist zulässig ($\texttt{Low} \rightsquigarrow \texttt{High}$).

      – Der Schreibzugriff ist **unzulässig** ($\texttt{High} \rightsquigarrow \texttt{Low}$).
        $\rightarrow$ Wird vom NoWriteHigh-Kriterium erkannt.

   c) Für neu erzeugte und nicht-klassifizierte (`unlabeled`) Marken gilt:

      Die Marken erhalten die Klassifikation `High`.

2. $t$ hat Freigabe `Low` $\textsc{Clear}_{\mathcal{SC}}(\mathrm{S}_{\mathcal{U}}(t)) = \texttt{Low}$.
   a) Für `High`-klassifizierte Eingabemarken gilt:

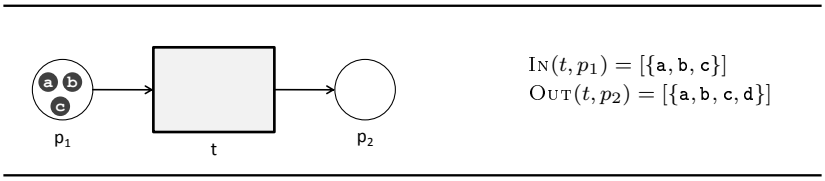



Tabelle 5.5.: Verarbeitung mehrerer Datenobjekte durch Transition $t$.

- Der Lesezugriff ist **unzulässig** (`High` $\rightsquigarrow$ `Low`).
  $\rightarrow$ Wird vom NoRead-Kriterium erkannt.

- Der Schreibzugriff ist zulässig (`Low` $\rightsquigarrow$ `High`).

b) Für `Low`-klassifizierte Eingabemarken gilt:

Der Zugriff (schreibend und lesend) ist zulässig (`Low` $\rightsquigarrow$ `Low`).

c) Für neu erzeugte und nicht-klassifizierte (`unlabeled`) Marken gilt:

Die Marken erhalten die Klassifikation `Low`.

3. $t$ hat Freigabe `neutral` $\textsc{Clear}_{\mathcal{SC}}(S_{\mathcal{U}}(t)) =$ `neutral`.

a) Für `High`-klassifizierte Eingabemarken gilt:

Der Zugriff (schreibend und lesend) ist zulässig (`neutral`/`High` $\rightsquigarrow$ `neutral`/`High`).

b) Für `Low`-klassifizierte Eingabemarken gilt:

- Der Lesezugriff ist zulässig (`Low` $\rightsquigarrow$ `neutral`).

- Der Schreibzugriff ist zulässig, wenn keine `High`-klassifizierten Marken gelesen werden.

- Der Schreibzugriff ist **unzulässig**, wenn `High`-klassifizierte Marken gelesen werden (`High` $\rightsquigarrow$ `Low`).
  $\rightarrow$ Wird vom NoWriteNeutral-Kriterium erkannt.

c) Für neu erzeugte und nicht-klassifizierte (`unlabeled`) Marken gilt:

Die Marken erhalten die Klassifikation der am höchsten klassifizierten Eingabemarke, die gelesen wird.

### 5.4.2 Verdeckte Informationsübertragung

Durch die Informationsflusskriterien StrongCausal, StrongUsage und StrongCond (sowie ihre jeweils abgeschwächten Varianten) sollen genau die IFnets identifiziert werden, bei denen ein Subjekt mit `Low`-Freigabe keine Informationen über die Ausführung von `High`-klassifizierten Aktivitäten erhalten kann. Durch die Fähigkeit, derartige Informationsübertragung zu erkennen, kann InDico umfassendere Isolationsgarantien für Geschäftsprozess-Modelle bereitstellen, als dies bislang möglich ist. So können etwa die Verletzungen der Sicherheitsanforderungen im gezeigten Anwendungsszenario mit bisherigen Ansätzen weder spezifiziert noch erkannt werden.

Um die von InDico bereitgestellten Garantien bzgl. der verdeckten Übertragung von Information bewerten zu können, wird zunächst gezeigt, dass ein IFnet, welches die entsprechenden Informationsflusskriterien erfüllt, auch eine formal definierte Noninterferenz-Eigenschaft besitzt. Noninterferenz-Eigenschaften sind der übliche Weg, um die Abwesenheit verdeckter Kanäle in einem System zu beschreiben. Vereinfacht besagt die Idee der Noninterferenz, dass zwei (Teil-)Prozesse sich nicht beeinflussen (und somit auch keine Informationen übertragen), wenn sie sich – jeweils alleine ausgeführt – in exakt der gleichen Weise verhalten, wie wenn sie gemeinsam ausgeführt werden (Goguen und Meseguer, 1982). Seit ihrer Einführung sind Noninterferenz-Eigenschaften in zahlreichen Varianten für verschiedene Systemmodelle definiert worden (Zakinthinos und Lee, 1997; Mantel, 2005). Eine Übertragung auf Geschäftsprozess-Modelle fehlte allerdings bislang.

#### SBNDC für IFnets

Die im Folgenden verwendete Noninterferenz-Eigenschaft orientiert sich an der *Strong Bisimulation Non Deducibility on Composition*-Eigenschaft (SBNDC), die zunächst auf Prozess-Algebren (Focardi und Gorrieri, 2001) und anschließend auf Petri-Netzen definiert wurde (Busi und Gorrieri, 2004). In der letzteren Variante schließt sie Abhängigkeiten zwischen hoch- und niedrig-klassifizierten Transitionen eines Petri-Netzes aus. Die hier verwendete SBNDC-Eigenschaft ist an die besonderen Eigenschaften von IFnets angepasst, insbesondere die Existenz der zusätzlichen Freigabe `neutral`. Die folgenden Definitionen und die Beweisführung orientieren sich an der Darstellung von Busi und Gorrieri (2009), in der eine ähnliche Vorgehensweise für klassische Petri-Netze gezeigt wird.

SBNDC ist eine vergleichsweise starke Noninterferenz-Eigenschaft, die ausschließt, dass eine Partei, die ausschließlich die Ausführung von Aktivitäten eines `Low`-Subjektes beobachten kann, daraus Informationen bzgl. der Ausführung von `High`-klassifizierten Aktivitäten ableiten kann. Um sie für IFnets formal beschreiben zu können, werden zunächst die „`Low`-Sicht“ (`Low` *view*) auf ein IFnet und eine entsprechende Bisimulation definiert.

**Definition 5.1 (Low-Sicht).** *Sei $(N, M)$ ein markiertes IFnet mit Anfangszustand $M$. Für eine Schaltfolge $\sigma$ ist die* Low*-Sicht $\Lambda_N$ wie folgt definiert:*

$$\Lambda_N(\epsilon) = \epsilon.$$

$$\Lambda_N(\sigma t) = \begin{cases} \Lambda_N(\sigma)t & falls \quad \textsc{Clear}_{\mathcal{SC}}(\mathrm{S}_{\mathcal{U}}(t)) = \texttt{Low} \\ \Lambda_N(\sigma) & andernfalls \end{cases}$$

*Die Definition von $\Lambda_N$ wird in natürlicher Weise auf Mengen von Schaltfolgen $\Sigma$ erweitert: $\Lambda_N(\Sigma) = \{\Lambda_N(\sigma) \mid \sigma \in \Sigma\}$.* ⊣

Um zwei IFnets bzgl. ihrer Low-Sicht vergleichen zu können, wird eine Low-Sicht-Bisimulation definiert.

**Definition 5.2 (Low-Sicht-Bisimulation).** *Sei $(N, M)$ ein markiertes IFnet mit Anfangszustand $M$. Eine Relation $R$ auf $[M] \times [M]$ ist eine* Low*-Sicht-Bisimulation auf $N$, so dass falls $(M_1, M_2) \in R$ für alle $l \in L$ gilt:*

- *Falls $M_1 \xrightarrow{\sigma l} M_1'$, dann existiert ein Zustand $M_2' \in [M]$ und eine Schaltfolge $\sigma'$, so dass $M_2 \xrightarrow{\sigma' l} M_2'$ und $\Lambda_N(\sigma) = \Lambda_N(\sigma')$.*
- *Falls $M_2 \xrightarrow{\sigma l} M_2'$, dann existiert ein Zustand $M_1' \in [M]$ und eine Schaltfolge $\sigma'$, so dass $M_1 \xrightarrow{\sigma' l} M_1'$ und $\Lambda_N(\sigma) = \Lambda_N(\sigma')$.* ⊣

Darauf aufbauend wird nun die SBNDC-Eigenschaft für IFnet definiert.

**Definition 5.3 (SBNDC für IFnet).** *Sei $(N, M)$ ein markiertes IFnet mit Anfangszustand $M$ und $H \subseteq T$ die Menge der* High*-klassifizierten Transitionen. $(N, M)$ erfüllt die SBNDC-Eigenschaft genau dann, wenn für alle Zustände $M_1 \in [M]$ und für alle $h \in H$ gilt:*

*Wenn $M_1 \xrightarrow{h} M_1'$, dann gibt es eine* Low*-Sicht-Bisimulation $R$ auf $N \setminus H$, so dass $(M_1, M_1') \in R$.*[3] ⊣

Die Erfüllung der SBNDC-Eigenschaft bewirkt, dass die Ausführung einer Transition aus der Menge $H$ für ein Low-Subjekt „unsichtbar“ bleibt: Durch die Existenz der Low-Sicht-Bisimulation auf $N \setminus H$ ist sichergestellt, dass die Schaltmöglichkeiten der Transitionen von Subjekten mit Low-Freigabe durch die Ausführung von High-Transitionen unbeeinträchtigt bleiben.

Abbildung 5.4 zeigt zur Illustration der SBNDC-Eigenschaft ein markiertes IFnet $(N, M)$ mit Anfangszustand $M$, das genau eine High-klassifizierte Transition („High“) enthält. Darüber hinaus ist zweimal das IFnet $N \setminus H$ abgebildet, das aus $N$ entsteht, wenn die

[3] Mit $N \setminus H$ wird das IFnet bezeichnet, das aus $N$ entsteht, wenn die Transitionen aus $H$ und die damit verbundenen Kanten entfernt werden.

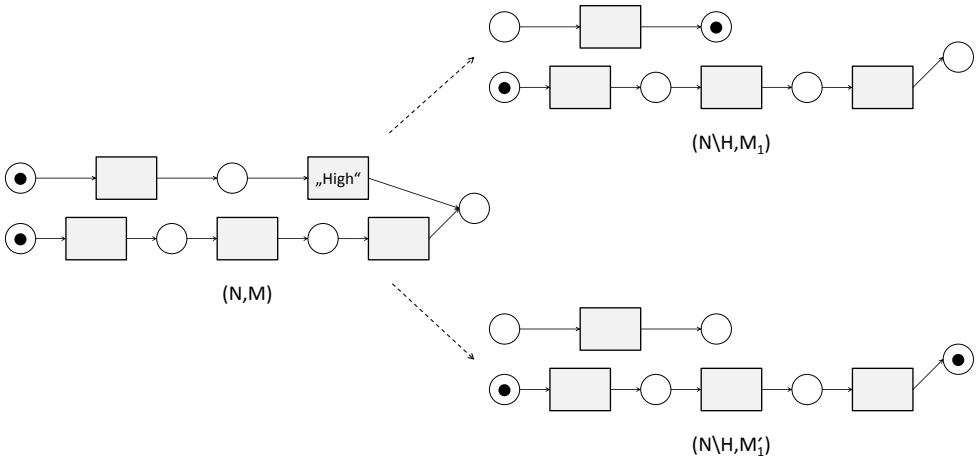



Abbildung 5.4.: Illustration der SBNDC-Eigenschaft für IFnet.

`High`-klassifizierte Transition und die damit verbundenen Kanten entfernt werden. Die obere der beiden Darstellungen zeigt $N \setminus H$ im Zustand $M_1$, in dem die `High`-klassifizierte Transition in $N$ schaltbereit ist.[4] Die untere Darstellung zeigt $N \setminus H$ im Zustand $M_1'$, den $N$ nach dem Schalten der `High`-klassifizierten Transition im Zustand $M_1$ annimmt. Wenn $(N, M)$ die SBNDC-Eigenschaft erfüllt, muss es eine `Low`-Sicht-Bisimulation auf $N \setminus H$ geben, die $M_1$ und $M_1'$ enthält (was hier offensichtlich der Fall ist).

Es wird nun gezeigt, dass ein IFnet $(N, M)$ mit Anfangszustand $M$, das die Kriterien StrongCausal und StrongUsage erfüllt, auch die SBNDC-Eigenschaft besitzt. Für einen Zustand $M_1 \in [M]$ und eine Transition $h \in H$ mit $M_1 \xrightarrow{h} M_1'$ wird dazu gezeigt, dass es eine `Low`-Sicht-Bisimulation auf $N \setminus H$ gibt, die $(M_1, M_1')$ enthält.

**Beweis:** Entsprechend der Definition der `Low`-Sicht-Bisimulation wird zuerst gezeigt, dass, sofern eine Transition mit `Low`-Freigabe im Zustand $M_1$ im IFnet $N \setminus H$ ausgeführt werden kann, diese Transition auch im Zustand $M_1'$ ausgeführt werden kann, wobei die `Low`-Sicht identisch bleibt. Anschließend wird der umgekehrte Fall gezeigt.

- Für eine Schaltfolge $\sigma$ und eine Transition $l \in L$ gelte $M_1 \xrightarrow{\sigma l} M_2$. Es wird gezeigt, dass es eine Schaltfolge $\sigma' \subseteq \sigma$ gibt, so dass $M_1' \xrightarrow{\sigma' l} M_2'$.

  Angenommen, es gäbe eine solche Schaltfolge $\sigma'$ nicht. Dann muss eine Schaltfolge $t_1 t_2 .. t_n \subseteq \sigma l$ existieren, in der $t_n = l$, $t_1$ beim Schalten eine Marke $c$ aus der Menge $M_1 \setminus M_1'$ konsumiert und $m_i \overset{t_i p_i t_{i+1}}{\rightsquigarrow} m_{i+1}, i \in \{1, .., n-1\}$ für Zustände $m_i, m_{i+1}$ sowie Stellen $p \in t_i \bullet \cap \bullet t_{i+1}$.

[4] Für die dargestellten IFnets wird die übliche Petri-Netz-Schaltsemantik angenommen, in der eine Transition beim Schalten aus jeder Eingangsstelle genau eine schwarze Marke entnimmt und in jeder Ausgangsstelle genau eine schwarze Marke produziert.

Da für das IFnet $(N, M)$ $M_1 \xrightarrow{h} M_1'$ gilt, muss die von $t_1$ konsumierte Marke $c$ im Zustand $M_1$ in einer Eingangsstelle $p_h \in {\bullet}h$ der Transition $h$ sein. Das bedeutet, dass $h$ und $t_1$ im Zustand $M_1$ um die in $p_h$ liegende Marke $c$ konkurrieren. Es gilt also $M_1 \overset{hp_ht_1}{\times_{\mathsf{uc}}} M_1'$.

Hinsichtlich der Struktur der Schaltfolge $t_1t_2..t_n$ sind die folgenden Fälle möglich:

1. Die Schaltfolge hat die Länge 1 und besteht demnach nur aus der Transition $l$. In diesem Fall bestünde in dem IFnet $(N, M)$ ein Nutzungskonflikt zwischen der `High`-klassifizierten Transition $h$ und der Transition $l$. Da $(N, M)$ nach Voraussetzung das StrongUsage-Kriterium erfüllt, besteht ein Widerspruch.

2. Die Transitionen $t_1t_2..t_{n-1}$ haben die Freigabe `neutral`. Folglich bildet die Struktur $hp_1t_1p_2t_2..p_{n-1}t_{n-1}p_nl$, wobei $p_i$ für die jeweils verbindende Stelle steht, in IFnet $(N, M)$ eine kritische Nutzungsstelle, die – wie gezeigt – auch ausgeführt werden kann. Damit wäre das StrongUsage-Kriterium verletzt, was ein Widerspruch zur Voraussetzung wäre.

3. Unter den Transitionen $t_1t_2..t_{n-1}$ gibt es mindestens eine Transition mit der Freigabe `Low`. Sei $t_l$ diejenige dieser Transitionen mit dem kleinsten Index. Folglich bildet die Struktur $hp_1t_1p_2t_2..p_lt_l$, wobei $p_i$ für die jeweils verbindende Stelle steht, in IFnet $(N, M)$ eine kritische Nutzungsstelle, die – wie gezeigt – auch ausgeführt werden kann. Damit wäre das StrongCausal-Kriterium verletzt, was ein Widerspruch zur Voraussetzung wäre.

Daraus folgt, dass eine Schaltfolge $\sigma'$ existieren muss. Weiterhin muss es eine solche Folge $\sigma'$ geben, bei der sämtliche Transitionen aus der Menge $L$, die in der Schaltfolge $\sigma$ vorkommen, auch in derselben Abfolge in $\sigma'$ enthalten sind. Wäre dies nicht der Fall, ergäbe sich ein Widerspruch zur vorausgesetzten Erfüllung des StrongCausal-Kriteriums wie oben, weil die Ausführung von $h$ damit die Ausführung einer Transition aus $L$ verhindern würde. Entsprechend gilt für eine solche Schaltfolge $\sigma'$ $\Lambda_N(\sigma) = \Lambda_N(\sigma')$.

- Für eine Schaltfolge $\sigma$ und eine Transition $l \in L$ gelte $M_1' \xrightarrow{\sigma l} M_2'$. Es wird gezeigt, dass es eine Schaltfolge $\sigma' \subseteq \sigma$ gibt, so dass $M_1 \xrightarrow{\sigma' l} M_2$.

  Angenommen, es gäbe eine solche Schaltfolge $\sigma'$ nicht. Dann muss eine Schaltfolge $t_1t_2..t_n \subseteq \sigma l$ existieren, in der $t_n = l, t_1$ beim Schalten eine Marke $c$ aus der Menge $M_1' \setminus M_1$ konsumiert und $m_i \overset{t_ip_it_{i+1}}{\rightharpoonup} m_{i+1}, i \in \{1, .., n-1\}$ für Zustände $m_i, m_{i+1}$ sowie Stellen $p \in t_i \bullet \cap \bullet t_{i+1}$.

Da für das IFnet $(N, M)$ $M_1 \xrightarrow{h} M_1'$ gilt, muss die von $t_1$ konsumierte Marke $c$ im Zustand $M_1'$ in einer Ausgangsstelle $p_h \in h\bullet$ der Transition $h$ sein. Folglich gilt $M_1 \xrightarrow{hp_h t_1} M_1'$.

Hinsichtlich der Struktur der Schaltfolge $t_1 t_2..t_n$ sind die folgenden Fälle möglich:

1. Die Schaltfolge hat die Länge 1 und besteht demnach nur aus der Transition $l$. In diesem Fall bestünde in dem IFnet $(N, M)$ eine direkte Abhängigkeit zwischen der `High`-klassifizierten Transition $h$ und der Transition $l$. Da $(N, M)$ nach Voraussetzung das StrongCausal-Kriterium erfüllt, besteht ein Widerspruch.

2. Die Transitionen $t_1 t_2..t_{n-1}$ haben die Freigabe `neutral`. Folglich bildet die Struktur $hp_1 t_1 p_2 t_2..p_{n-1} t_{n-1} p_n l$, wobei $p_i$ für die jeweils verbindende Stelle steht, in IFnet $(N, M)$ eine kritische Transitionenfolge, die – wie gezeigt – auch ausgeführt werden kann. Damit wäre das StrongCausal-Kriterium verletzt, was ein Widerspruch zur Voraussetzung wäre.

3. Unter den Transitionen $t_1 t_2..t_{n-1}$ gibt es mindestens eine Transition mit der Freigabe `Low`. Sei $t_l$ diejenige dieser Transitionen mit dem kleinsten Index. Folglich bildet die Struktur $hp_1 t_1 p_2 t_2..p_l t_l$, wobei $p_i$ für die jeweils verbindende Stelle steht, in IFnet $(N, M)$ eine kritische Transitionenfolge, die – wie gezeigt – auch ausgeführt werden kann. Damit wäre das StrongCausal-Kriterium verletzt, was ein Widerspruch zur Voraussetzung wäre.

Daraus folgt, dass eine Schaltfolge $\sigma'$ existieren muss. Weiterhin muss es eine solche Folge $\sigma'$ geben, bei der sämtliche Transitionen aus der Menge $L$, die in der Schaltfolge $\sigma$ vorkommen, auch in derselben Abfolge in $\sigma'$ enthalten sind. Wäre dies nicht der Fall, ergäbe sich ein Widerspruch zur vorausgesetzten Erfüllung des StrongCausal-Kriteriums wie oben, weil die Nichtausführung von $h$ damit die Ausführung einer Transition aus $L$ verhindern würde. Entsprechend gilt für eine solche Schaltfolge $\sigma'$ $\Lambda_N(\sigma) = \Lambda_N(\sigma')$.

⊣

### Bedingte Ausführung

Die oben gezeigte Korrespondenz zwischen den Informationsflusskriterien von InDico und der SBNDC-Eigenschaft berücksichtigt nur die Kriterien StrongCausal und StrongUsage, weil SBNDC ausschließlich Abhängigkeiten zwischen unterschiedlich klassifizierten Aktivitäten betrachtet. Das StrongCond-Kriterium hingegen charakterisiert

Abhängigkeiten (und damit Informationsfluss) zwischen einem `High`-klassifizierten Datenobjekt und der Ausführung einer Aktivität mit `Low`-Freigabe. Es wird in InDico eingeführt, da IFnet den Datenfluss explizit modellieren kann und die Wechselwirkung zwischen Daten- und Kontrollfluss zu unerlaubten Informationsflüssen führen kann, wie im Anwendungsszenario gezeigt wird. Ein vergleichbares Kriterium existiert in Programmiersprachen mit Fähigkeiten zur Informationsflusskontrolle (Sabelfeld und Myers, 2003).

Konzeptionell unterscheidet sich StrongCond nur wenig vom StrongCausal-Kriterium: In beiden Fällen werden kausale Abhängigkeiten zwischen der Ausführung von Transitionen, die mit `High`-klassifizierten Informationen verknüpft sind, und Transitionen mit `Low`-Freigabe betrachtet. Im Fall des StrongCausal-Kriteriums ist die Transition selbst `High`-klassifiziert, während es bei StrongCond die Marken des zugehörigen Prädikates sind. Insofern können beide Kriterien auch als alternative Spezifikationsformen desselben Informationsflusses betrachtet werden. So könnte im Anwendungsbeispiel die Sicherheitsanforderung aus Abschnitt 5.1.2 auch kodiert werden, indem die beiden Transitionen Branch1 und Branch2 (denen ein Prädikat mit `High`-Datenobjekt zugeordnet ist) mit der Sicherheitsstufe `High` klassifiziert werden.

### Abgeschwächte Kriterien

Noninterferenz-Eigenschaften wie SBNDC streben die vollständige Isolation `High`-klassifizierter Informationen von Nutzern mit niedrigerer Freigabe an. Eine solche Policy ist für viele Anwendungen zu restriktiv und führt dazu, dass Noninterferenz-basierte Sicherheitsmechanismen in entsprechenden Systemen nicht eingesetzt werden (Zdancewic, 2004; Ryan u. a., 2001).

Die abgeschwächten Varianten der Informationsflusskriterien StrongCausal, StrongUsage und StrongCond sind der Versuch, generische Informationsflusskriterien anzugeben, die weniger restriktiv und für manche Anwendungen besser geeignet sind als ihre stärkeren Gegenstücke. Während StrongCausal, StrongUsage und StrongCond ein IFnet als unsicher ausschließen, sofern eine `High`-klassifizierte Transition (bzw. eine Transition deren zugeordnetes Prädikat ein `High`-klassifiziertes Datenobjekt benutzt) bei mindestens einem Ausführungspfad die Ausführung einer Transition mit `Low`-Freigabe bedingt oder verhindert, schließen die abgeschwächten Varianten das IFnet nur aus, wenn eine solche Abhängigkeit bei *jedem* Ausführungspfad vorliegt, in dem die Transition mit `Low`-Freigabe vorkommt. In letzterem Fall erhält das Subjekt mit `Low`-Freigabe *mehr* Informationen über das Verhalten von `High`-klassifizierten Transitionen, weil es durch die Ausführung bzw. Blockade einer eigenen Transition erfährt, *welche* `High`-klassifizierte Transition ausgeführt wurde.

Der Unterschied zwischen der stärkeren und der schwächeren Variante eines Kriterium kann anhand des Anwendungsbeispiels deutlich gemacht werden. Abbildung 5.5 zeigt das Fragment eines IFnet-Modells, das durch eine leichte Modifikation des in Abbildung 5.2

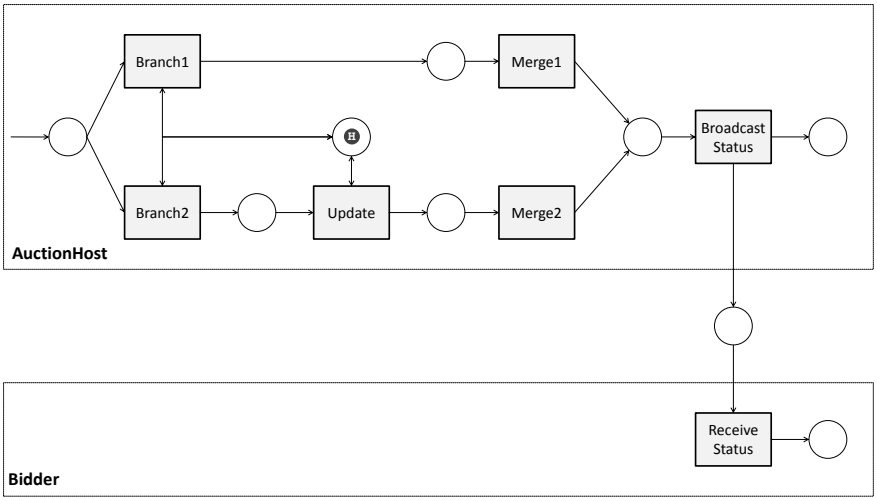



Abbildung 5.5.: Modifiziertes IFnet-Fragment.

gezeigten Modells entsteht. Der einzige Unterschied zum Ursprungsmodell besteht darin, dass die Benachrichtigung über den Auktionsstatus erst nach dem Verschmelzen der beiden Ausführungspfade ausgeführt wird. Durch diese Modifikation ist das Modell hinsichtlich der ersten Sicherheitsanforderung anschaulich „sicher“, da die Benachrichtigung nicht mehr an das Eintreffen eines neuen Höchstgebotes gekoppelt ist, sondern in jedem Fall ausgeführt wird.

Trotzdem verletzt auch dieses Modell das StrongCond-Kriterium, da es eine Ausführung gibt, in der etwa die Transition Branch2 (der ein Prädikat mit einer `High`-klassifizierten Marke zugeordnet ist) Voraussetzung für die Ausführung der Transition ReceiveStatus ist, die eine `Low`-Freigabe hat. Ein Bieter kann allerdings nur Informationen über das Höchstgebot erlangen, wenn er zwischen der Ausführung der beiden Pfade, die mit Branch1 bzw. Branch2 beginnen, unterscheiden kann. Insofern erfasst das WeakCond-Kriterium – das von dem Modell erfüllt wird – die Sicherheitsanforderung in diesem Fall besser als das stärkere StrongCond-Kriterium.

## 5.5 Laufzeit

Die Bewertung der Laufzeit, die für die Zertifizierung von IFnet-Modellen benötigt wird, erfolgt auf der Grundlage einer Reihe von Test-Modellen unterschiedlicher Größe. In der Praxis sind Geschäftsprozess-Modelle – gemessen an der Anzahl der Aktivitäten – relativ klein. *Best-Practice*-Richtlinien schreiben Prozess-Modelle mit nicht mehr als zehn Aktivitäten vor (Wolf und Harmon, 2010). Die Betrachtung öffentlich zugänglicher Prozess-Sammlungen zeigt, dass die abgebildeten Prozesse zwanzig Aktivitäten selten überschreiten (Ben-Eliahu, 2008).

Die für die Laufzeitmessung benutzten IFnet-Modelle sind durch ein Hilfsprogramm automatisch generiert worden und haben eine Größe von bis zu 25 Transitionen. Bei der Generierung der Modelle ist der Verzweigungsgrad – der einen wesentlichen Einfluss auf die Anzahl der Zustände und damit die Laufzeit hat – zufallsgesteuert variiert worden: In durchschnittlich 80% der Fälle folgt auf eine Transition genau eine Transition, in durchschnittlich 15% folgen zwei Transitionen und in durchschnittlich 5% drei Transitionen. Zur Vereinfachung der Generierung benutzen die Test-Modelle ausschließlich schwarze Marken, von denen jede Transition genau eine in ihren Eingangsstellen erwartet und an ihre Ausgangsstellen weitergibt.

Die Möglichkeit einer Anforderungsverletzung ist ebenfalls zufallsgesteuert integriert worden, indem genau eine Transition mit der Klassifikation `High` ausgezeichnet worden ist und einer anderen Transition ein Subjekt mit der Freigabe `Low` zugeordnet wurde. Dies bedeutet, dass ausschließlich Möglichkeiten verdeckter Informationsübertragung bestehen. Diese Fälle sind im Hinblick auf die Laufzeit aussagekräftiger als Verletzungen von Datenflusskriterien, deren Prüfung weniger aufwändig ist (vgl. Abschnitt 4.2.3).

Für die Laufzeitbewertung ist die Größe der Testmodelle sukzessive in Schritten von fünf Transitionen erhöht worden. Für jede Größe sind zehn Testmodelle generiert und die Laufzeit für jeden der drei Analyseschritte – strukturelle Prüfung, Erzeugung des Zustandsgraphen und Analyse des Zustandsgraphen – gemessen worden. Anschließend ist aus den Messwerten für jede Modellgröße das arithmetische Mittel gebildet worden. Abbildung 5.6 zeigt die auf diese Weise erhaltenen Ergebnisse in einer logarithmischen Darstellung.

Die Darstellung zeigt, dass insbesondere die Laufzeit für den zweiten Analyseschritt – die Erzeugung des Zustandsgraphen – im Vergleich zu dem ersten (strukturellen) Schritt überproportional steigt und für jede Testreihe den größten Anteil an der Gesamtlaufzeit hat. Dieses Ergebnis ist nicht überraschend, da bei Petri-Netzen – wie auch bei anderen Systemmodellen – die Größe des Zustandsraums mit der Netzgröße exponentiell ansteigt (Küngas, 2005). Für den ersten Analyseschritt – die strukturelle Prüfung – steigt die Laufzeit dagegen nur moderat. Dies verdeutlicht den Sinn dieser ersten Phase der Prüfung, die unter Umständen bereits die Erfüllung der gestellten Anforderungen zeigen kann, ohne dass die weiteren – aufwändigeren – Schritte durchgeführt werden müssen (vgl. Abschnitt 3.5).

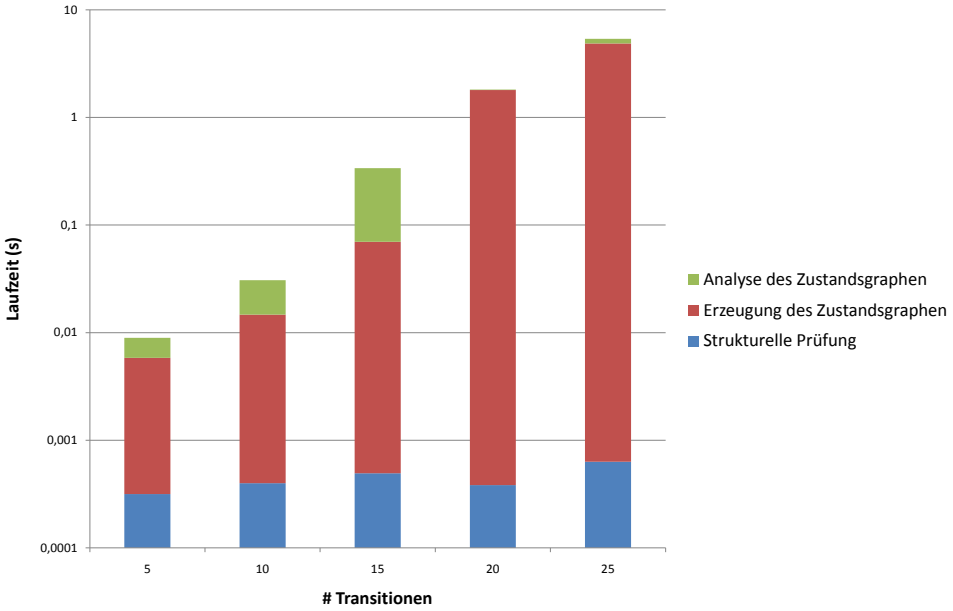



Abbildung 5.6.: Laufzeit des Zertifizierungsvorgangs.

Die Variation der Zunahme bei den einzelnen Schritte zeigt, dass die Laufzeit nicht ausschließlich von der Transitionenzahl, sondern in erheblichem Maße auch vom Verzweigungsgrad abhängt. Ein niedrigerer Verzweigungsgrad (insbesondere in der Nähe der Anfangstransition) ergibt bei gleichbleibender Netzgröße einen kleineren Zustandsraum und eine geringere Laufzeit. Ein weiterer Einflussfaktor ist die Platzierung der ausgezeichneten Transitionen, die über die Anzahl und Art der Sicherheitsverletzungen und damit auch über den Suchaufwand bestimmt. So ergeben sich etwa bei den Modellen der vorletzten Testreihe (mit 20 Transitionen) im Durchschnitt weniger Verletzungen als bei den übrigen Reihen, wodurch die Laufzeit des dritten Analyseschrittes verkürzt wird.

Die Ergebnisse der Laufzeitmessungen zeigen, dass die Zertifizierung mit InDico für Geschäftsprozesse einer in der Praxis üblichen Größenordnung praktikabel ist. Durch die exponentiell wachsende Laufzeit ist InDico mit der gegebenen Implementierung dagegen für große Prozesse (und insbesondere für große Prozess-Konstellationen) unter Umständen nicht mehr praktikabel einsetzbar. Diesem Problem kann ggf. durch verbesserte Prüfungsalgorithmen begegnet werden. Für klassische Petri-Netze existieren etwa Verfahren, um Erreichbarkeitsprüfungen (d.h. Entscheidungsprozeduren für die Frage, ob ein Zustand ausgehend vom Anfangszustand erreicht werden kann) unter bestimmten Bedingungen effizient zu machen (Küngas, 2005). Andere Ansätze befassen sich mit der systematischen Reduktion eines Petri-Netzes, um Bereiche auszuschließen, die für spätere Analysen nicht relevant sind (Llorens u. a., 2008). Diese Ansätze sind ggf. auf die InDico-Prüfung von IFnet-Modellen übertragbar, um auf diese Weise die Zertifizierung für größere Prozesse praktikabel zu machen. Ihre Anwendbarkeit wird im Rahmen dieser Arbeit allerdings nicht untersucht und bleibt ein möglicher Gegenstand weiterer Forschung.

# Kapitel 6

# Zusammenfassung und Ausblick

Diese Arbeit präsentiert mit InDico ein Verfahren zur Zertifizierung von Isolationsanforderungen in Geschäftsprozessen. Mit InDico kann vor der Ausführung eines Prozesses auf der Grundlage eines Modells geprüft werden, ob zur Laufzeit Sicherheitsverletzungen auftreten können, d.h. ob ein Prozess-Teilnehmer unbefugt an sensible Informationen gelangen kann. Der zentrale Beitrag der Arbeit besteht in der Anwendung von Techniken der Informationsflussanalyse für die Geschäftsprozess-Zertifizierung, um auf diese Weise stärkere Sicherheitsgarantien bereitstellen zu können, als dies mit vorhandenen Verfahren möglich ist. Die InDico-Garantien sind umfangreicher, da sie zum Einen Informationsflüsse durch das gesamte Prozess-Modell von der Quelle bis zur Senke prüfen (Ende-zu-Ende-Garantien) und zum Anderen neben dem Datenfluss verdeckte Informationsübertragung berücksichtigen. Darüber hinaus macht InDico die Zertifizierung zu einem hohen Grad automatisierbar, indem eine Transformation von Prozess-Modellen aus praxisüblichen Beschreibungen (BPMN) und ein Ansatz zur Strategie-gesteuerten Kodierung von Sicherheitsanforderungen bereitgestellt werden. Vergleichbare Komponenten existieren bisher nicht bzw. nicht in Verbindung mit Verfahren zur Zertifizierung von Geschäftsprozessen.

Die Arbeit wird motiviert durch die Idee, dass unzureichende Sicherheitsgarantien kein Hindernis für die Integration von Prozessen mit externen Partnern und für ihre Ausführung auf geteilten Infrastrukturen sein sollen. Der Trend zur verteilten Ausführung geschäftlicher Vorgänge wird getrieben von der fortschreitenden Abbildung der Prozesse in flexiblen IT-Architekturen – etwa dienstbasierten Architekturen (*service-oriented architectures*) – durch die sich Prozess-Bestandteile einfach austauschen und über Netzwerke mit externen Parteien koppeln lassen. Insbesondere sogenannten *Cloud*-Diensten, die spezialisierte Leistungen bedarfsgerecht über ein Netzwerk zur Verfügung stellen, wird in dieser Hinsicht zukünftig eine bedeutende Rolle zugeschrieben. Die wirtschaftlichen Vorteile ihres Einsatzes liegen in einer großen Flexibilität bei der Auswahl von Dienstleistern und oft sehr viel geringeren Kosten als für den Betrieb und die Wartung eigener Infrastrukturen aufgewendet werden müssten. Die Kehrseite dieser Entwicklung sind die

potentiellen Sicherheitsprobleme, die durch eine Weitergabe sensibler Informationen an dritte Parteien entstehen können. Der mit InDico vorgeschlagene Lösungsansatz sieht die a-priori-Prüfung von Geschäftsprozess-Konstellationen – etwa durch Betreiber von Dienst-Plattformen – vor, deren Ergebnis als Grundlage für die Vergabe entsprechender Garantien gegenüber den Prozess-Beteiligten verwendet werden kann.

Um das InDico-Verfahren zu realisieren, leistet die Arbeit mehrere Beiträge zum aktuellen Stand der Wissenschaft:

- Die Petri-Netz-basierte Beschreibungssprache IFnet ist ausdrucksstärker als andere formal fundierte Geschäftsprozess-Metamodelle, da sie u.a. die explizite Modellierung des Datenflusses mit unterscheidbaren Datenobjekten, die Charakterisierung alternativen Ein- und Ausgabeverhaltens sowie die Kodierung von Sicherheitsanforderungen ermöglicht.

- Im mehrstufigen Sicherheitsmodell von IFnet können im Gegensatz zu anderen Modellen *alle* Informationsträger (d.h. Datenobjekte/Subjekte *und* Aktivitäten) mit Sicherheitsstufen versehen werden. Das Sicherheitsmodell unterscheidet klar zwischen der Klassifikation von Informationen und der Freigabe von Subjekten und ermöglicht auf diese Weise die Spezifikation feingranularer Sicherheitsanforderungen. Die spezielle Freigabe `neutral` erlaubt die Modellierung von neutralen Intermediären (wie z.B. vertrauenswürdigen Dienstbetreibern), wie sie in vielen betrieblichen Einsatzszenarien auftreten.

- Die Zertifizierungsroutinen von InDico realisieren zum ersten Mal eine formal fundierte Methode zur Erkennung verdeckter Informationsübertragung in Geschäftsprozess-Modellen und erweitern damit die Aussagekraft der bereitgestellten Sicherheitsgarantien im Vergleich zu anderen Verfahren.

- Mit der Abbildung von BPMN-Prozessen nach IFnet und der Definition von Strategien zur Kodierung von Sicherheitsanforderungen schlägt InDico ein vollständiges Vorgehensmodell zur Sicherheitszertifizierung vor, das sich zu einem hohen Grad automatisieren lässt.

InDico wird sowohl experimentell auf der Grundlage eines Anwendungsszenarios als auch durch eine qualitative Bewertung evaluiert. Die experimentelle Evaluation zeigt, dass die automatisierte Zertifizierung mit InDico in einem realitätsnahen Szenario praktisch durchführbar ist und Sicherheitsverletzungen erkannt werden, die von bestehenden Verfahren unberücksichtigt bleiben. Die qualitative Bewertung zeigt durch den Abgleich mit einer Sammlung von Geschäftsprozess-Fragmenten (*Workflow Patterns*), dass praktisch eingesetzte Prozesse mit IFnet dargestellt werden können. Der Umfang der von InDico bereitgestellten Sicherheitsgarantien wird abgegrenzt, indem ihre Korrespondenz zu in der Literatur definierten Sicherheitseigenschaften gezeigt wird.

## 6.1 Möglichkeiten der Erweiterung von InDico

Es bestehen verschiedene Möglichkeiten, InDico zu erweitern und auszubauen. Im Folgenden werden drei dieser Möglichkeiten knapp skizziert, die die Erzeugung von IFnet-Modellen, die Präzision der erzeugten Zertifikate sowie ihren Umfang betreffen.

### 6.1.1 Erzeugung von IFnet-Modellen aus Log-Aufzeichnungen

InDico sieht die Erzeugung von IFnet-Modellen durch die Transformation aus bestehenden Geschäftsprozess-Beschreibungen vor. Diese Herangehensweise ist auf die a-priori-Zertifizierung in der Entwurfsphase des Geschäftsprozess-Managements (vgl. das Phasenmodell in Abbildung 1.2) zugeschnitten und ermöglicht die Erkennung von Design-Fehlern, bevor der Prozess ausgeführt wird.

Van der Aalst und Weijters (2004) weisen allerdings darauf hin, dass die tatsächlich ausgeführten Prozesse in der Praxis oft von den modellierten abweichen, etwa wenn ein Prozess häufigen Anpassungen unterliegt. Aus diesem Grund schlagen sie vor, Prozess-Modelle aus Log-Protokollen, die während der Ausführung aufgezeichnet werden, mittels *Data-Mining*-Techniken zu rekonstruieren. Ein so rekonstruiertes Prozess-Modell könnte anschließend in der gleichen Weise wie ein transformiertes Modell mit InDico auf Sicherheitsverletzungen hin untersucht werden. Dieser Ansatz hat den Vorteil, dass – sofern sichergestellt ist, dass vollständige und authentische Log-Aufzeichnungen vorliegen – stets das aktuell ausgeführte Prozess-Modell geprüft wird. Weil Sicherheitsverletzungen erst *nach* ihrem Auftreten erkannt werden können, fällt eine solche a-posteriori-Zertifizierung in die Diagnosephase des Geschäftsprozess-Managements.

Derzeit bestehen zahlreiche erprobte Verfahren um die Struktur eines Geschäftsprozesses – d.h. den Kontrollfluss – aus Logdaten zu erzeugen und in eine Petri-Netz-Darstellung zu überführen, so etwa der $\alpha$-Algorithmus (van der Aalst u. a., 2004). Ein solches Petri-Netz ließe sich durch manuelle Ergänzungen einfach in ein IFnet transformieren und anschließend prüfen. Forschungsbedarf besteht hier noch bei der Rekonstruktion von Prozess-Eigenschaften, die über die Struktur hinausgehen, so insbesondere des Datenflusses. Außerdem müsste untersucht werden, inwieweit die von InDico gegebenen Sicherheitsgarantien auch für rekonstruierte Prozess-Modelle gelten, die nicht notwendigerweise vollständig sind (weil nicht jede mögliche Ausführung auch in den Log-Aufzeichnungen vorgekommen sein muss).

### 6.1.2 Bewertung von Informationsflüssen

Ein notorisches Problem bei der Verwendung von extensionalen Policies – wie den von InDico verwendeten Informationsfluss-Policies – ist die Spezifikation von Ausnahmen. Das Sicherheitsmodell von InDico schließt jegliche Übertragung von `High`-klassifizierter

Information zu Subjekten mit der Freigabe `Low` aus, d.h. es fordert eine strikte Isolation von Informationen mit unterschiedlichen Sicherheitsstufen. Diese Anforderung ist zu restriktiv für viele Anwendungen, die einen gewissen Informationsaustausch zwischen verschiedenen Stufen zulassen müssen, um ordnungsgemäß funktionieren zu können.

In der Forschung zu Informationsflussanalysen gibt es verschiedene Ansätze, um spezifizieren zu können, welche Informationflüsse erlaubt sein müssen und welche ein Sicherheitsrisiko darstellen können. Diese Ansätze, die unter dem Stichwort „Deklassifikation" zusammengefasst werden, berücksichtigen beispielsweise die (informationstheoretisch gemessene) Informationsmenge, die übertragen werden darf, oder den Zeitpunkt, zu dem die Übertragung stattfindet (Sabelfeld und Sands, 2009).

Neben diesen generischen Ansätzen erscheint für die Anwendung in InDico auch ein kontextspezifisches Vorgehen aussichtsreich, bei dem die Besonderheiten von Geschäftsprozessen berücksichtigt werden. So könnten in Analogie zu den *Workflow Patterns* (van der Aalst u. a., 2003a) entsprechende „Sicherheitspatterns" ermittelt werden, die typischerweise in Geschäftsprozessen auftretende Sicherheitsverletzungen als Prozess-Fragmente kapseln und die für die Prüfung und Konstruktion von Prozessen verwendet werden können.

### 6.1.3 Zusätzliche Sicherheitskriterien

Neben dem Datenfluss betrachtet InDico verdeckte Informationsübertragung, die sich in der Ausführung bzw. Nicht-Ausführung von Aktivitäten manifestiert. Diese beiden Wege bilden nur einen Teil der Kanäle, durch die Information bei der Ausführung eines Geschäftsprozesses übertragen werden kann. So kann sensible Information beispielsweise auch durch die Beobachtung des Zeitverhaltens von Aktivitäten oder der Wahrscheinlichkeitsverteilung von Ausführungspfaden an unberechtigte Prozess-Teilnehmer übertragen werden (Brumley und Boneh, 2005; Sabelfeld und Myers, 2003).

Die Laufzeit-Informationen, die zur Erkennung solcher Informationskanäle benötigt werden, liegen im Fall der a-priori-Zertifizierung, wie sie in dieser Arbeit betrachtet wird, in der Regel nicht vor. In Analogie zu der in Abschnitt 6.1.1 angedachten Erweiterung zur a-posteriori-Zertifizierung auf der Basis von rekonstruierten Prozess-Modellen ließen sich diese Informationen jedoch problemlos aus entsprechenden Log-Aufzeichnungen extrahieren. Für ihre Darstellung müsste das IFnet-Metamodell um entsprechende Konstrukte erweitert werden. Ebenso müssten neue Sicherheitskriterien und Prüfungsroutinen entwickelt werden, die die unerlaubte Informationsübertragung über diese Kanäle charakterisieren bzw. erkennen können.

Neben zusätzlichen Kriterien für die Erkennung von reiner Informationsübertragung ist auch die Erweiterung um komplexere Sicherheits- und Compliancekriterien denkbar, die auf diesen grundlegenden Kriterien aufbauen. So sind in der Praxis Anforderungen zur

Funktionstrennung oder zur Vermeidung von Interessenkonflikten (vgl. etwa das *Chinese-Wall*-Modell (Brewer und Nash, 1989)) von großer Relevanz. In diesen wird der Informationsfluss zwischen Prozess-Beteiligten an bestimmte Bedingungen geknüpft (wie z.B. die Zugehörigkeit zu demselben Projekt-Team), die sich durch entsprechende Kriterien an ein IFnet-Modell auch in InDico ausdrücken ließen.

# Anhang A

# Veröffentlichungen

Teile dieser Arbeit sind in den folgenden Beiträgen veröffentlicht worden.

- Strong non-leak guarantees for workflow models. Mit Rafael Accorsi. In: *Symposium on Applied Computing*, ACM, 2011, S. 308–314
- InDico: Information flow analysis of business processes for confidentiality requirements. Mit Rafael Accorsi. In: *STM 2010*, LNCS, Bd. 6710, Springer-Verlag, 2011, S. 194–209
- Forensic leak detection analysis for business processes models. Mit Rafael Accorsi. In: *Advances in Digital Forensics VII*. Springer-Verlag, 2011, S. 101–113
- Informationsfluss-Mechanismen zur Zertifizierung von Cloud-basierten Geschäftsprozessen. Mit Rafael Accorsi. In: *Sicher in die digitale Welt von morgen – 12. Deutscher IT-Sicherheitskongress des BSI 2011*. SecuMedia, 2011
- Towards forensic data flow analysis of business process logs. Mit Rafael Accorsi und Thomas Stocker. In: *IT Security Incident Management & IT Forensics*, IEEE, 2011, S. 3–20
- SWAT: A security analysis toolkit for reliably process-aware information systems. Mit Rafael Accorsi und Sebastian Dochow. In: *International Conference on Availability, Reliability and Security*, IEEE, 2011, S. 692–697
- Auditing Workflow Executions against Dataflow Policies. Mit Rafael Accorsi. In: *International Conference on Business Information Systems*, LNBIP, Bd. 47, Springer-Verlag, 2010, S. 207–217
- Towards Information Flow Auditing in Workflows. In: *Software Engineering (Workshops)*, LNI, Bd. 160, Bonner Köllen Verlag, 2010, S. 549–554

- Static Information Flow Analysis of Workflow Models. Mit Rafael Accorsi. In: *Business Process and Service Science*, LNI, Bd. 177, Bonner Köllen Verlag, 2010, S. 194–205
- On Information Flow Forensics in Business Application Scenarios. Mit Rafael Accorsi und Günter Müller. In: *International Computer Software and Applications Conference*, IEEE, 2009, S. 324–328
- Detective Information Flow Analysis for Business Processes. Mit Rafael Accorsi. In: *Business Processes, Services Computing and Intelligent Service Management*, LNI, Bd. 147, Bonner Köllen Verlag, 2009, S. 223–224

# Anhang B

# XML-Schema der IFnet-Implementierung

```
<?xml version="1.0" encoding="utf-16"?>
<xsd:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
    version="1.0" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="pnml">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="net">
          <xsd:complexType>
            <xsd:sequence>

<!-- ################################################ -->
<!--       Transitionen                               -->
<!-- ################################################ -->

              <xsd:element name="transition" maxOccurs="unbounded">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="graphics">
                      <xsd:complexType>
                        <xsd:sequence>
                          <xsd:element name="position">
                            <xsd:complexType>
                              <xsd:attribute name="x" type="xsd:int" />
                              <xsd:attribute name="y" type="xsd:int" />
                            </xsd:complexType>
                          </xsd:element>
                          <xsd:element name="dimension">
                            <xsd:complexType>
                              <xsd:attribute name="x" type="xsd:int" />
                              <xsd:attribute name="y" type="xsd:int" />
                            </xsd:complexType>
                          </xsd:element>
                        </xsd:sequence>
                      </xsd:complexType>
                    </xsd:element>
                    <xsd:element name="name">
                      <xsd:complexType>
                        <xsd:sequence>
                          <xsd:element name="text" type="xsd:string" />
                        </xsd:sequence>
```

```
  </xsd:complexType>
</xsd:element>
<xsd:element name="toolspecific">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="subject">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="text" type="xsd:string" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="seclevel" minOccurs="0">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="text">
              <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                  <xsd:enumeration value="High"/>
                  <xsd:enumeration value="Low"/>
                  <xsd:enumeration value="unlabeled"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="guard" minOccurs="0">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="token">
              <xsd:complexType>
                <xsd:attribute name="tokenid"
                    type="xsd:string" />
              </xsd:complexType>
            </xsd:element>
          </xsd:sequence>
          <xsd:attribute name="id" type="xsd:string" />
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="inputoutputoption"
          maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="input">
              <xsd:complexType>
                <xsd:sequence>
                  <xsd:element name="sourceplace"
                      maxOccurs="unbounded">
                    <xsd:complexType>
                      <xsd:sequence>
                        <xsd:element name="token"
                            maxOccurs="unbounded">
                          <xsd:complexType>
                            <xsd:attribute name="tokenid"
                                type="xsd:string" />
                            <xsd:attribute name="accessmode">
                              <xsd:simpleType>
                                <xsd:restriction
                                    base="xsd:string">
                                  <xsd:enumeration
                                      value="read"/>
```

```
                                                    <xsd:enumeration
                                                        value="write"/>
                                                    <xsd:enumeration
                                                        value="read,write"/>
                                                  </xsd:restriction>
                                                </xsd:simpleType>
                                              </xsd:attribute>
                                            </xsd:complexType>
                                          </xsd:element>
                                        </xsd:sequence>
                                        <xsd:attribute name="id"
                                            type="xsd:string" />
                                      </xsd:complexType>
                                    </xsd:element>
                                  </xsd:sequence>
                                </xsd:complexType>
                              </xsd:element>
                              <xsd:element name="output">
                                <xsd:complexType>
                                  <xsd:sequence>
                                    <xsd:element name="targetplace"
                                        maxOccurs="unbounded">
                                      <xsd:complexType>
                                        <xsd:sequence>
                                          <xsd:element name="token"
                                              maxOccurs="unbounded">
                                            <xsd:complexType>
                                              <xsd:attribute name="tokenid"
                                                  type="xsd:string" />
                                            </xsd:complexType>
                                          </xsd:element>
                                        </xsd:sequence>
                                        <xsd:attribute name="id"
                                            type="xsd:string" />
                                      </xsd:complexType>
                                    </xsd:element>
                                  </xsd:sequence>
                                </xsd:complexType>
                              </xsd:element>
                            </xsd:sequence>
                          </xsd:complexType>
                        </xsd:element>
                      </xsd:sequence>
                      <xsd:attribute name="tool" type="xsd:string" />
                    </xsd:complexType>
                  </xsd:element>
                </xsd:sequence>
                <xsd:attribute name="id" type="xsd:string" />
              </xsd:complexType>
            </xsd:element>

<!-- ################################################ -->
<!--      Stellen                                     -->
<!-- ################################################ -->

            <xsd:element maxOccurs="unbounded" name="place">
              <xsd:complexType>
                <xsd:sequence>
                  <xsd:element name="graphics">
                    <xsd:complexType>
                      <xsd:sequence>
                        <xsd:element name="position">
```

```
                        <xsd:complexType>
                          <xsd:attribute name="x" type="xsd:int" />
                          <xsd:attribute name="y" type="xsd:int" />
                        </xsd:complexType>
                      </xsd:element>
                      <xsd:element name="dimension">
                        <xsd:complexType>
                          <xsd:attribute name="x" type="xsd:int" />
                          <xsd:attribute name="y" type="xsd:int" />
                        </xsd:complexType>
                      </xsd:element>
                    </xsd:sequence>
                  </xsd:complexType>
                </xsd:element>
                <xsd:element name="name">
                  <xsd:complexType>
                    <xsd:sequence>
                      <xsd:element name="text" type="xsd:string" />
                    </xsd:sequence>
                  </xsd:complexType>
                </xsd:element>
              </xsd:sequence>
              <xsd:attribute name="id" type="xsd:string" />
            </xsd:complexType>
          </xsd:element>

<!-- ################################################ -->
<!--       Kanten                                     -->
<!-- ################################################ -->

          <xsd:element maxOccurs="unbounded" name="arc">
            <xsd:complexType>
              <xsd:attribute name="id" type="xsd:string" />
              <xsd:attribute name="source" type="xsd:string" />
              <xsd:attribute name="target" type="xsd:string" />
            </xsd:complexType>
          </xsd:element>

<!-- ################################################ -->
<!--       Anfangszustand und Subjektbezeichner       -->
<!-- ################################################ -->

          <xsd:element name="toolspecific">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="initialMarking">
                  <xsd:complexType>
                    <xsd:sequence>
                      <xsd:element name="placemarking" maxOccurs="unbounded">
                        <xsd:complexType>
                          <xsd:sequence>
                            <xsd:element name="token">
                              <xsd:complexType>
                                <xsd:attribute name="tokenid"
                                    type="xsd:string" />
                                <xsd:attribute name="seclevel">
                                  <xsd:simpleType>
                                    <xsd:restriction base="xsd:string">
                                      <xsd:enumeration value="High"/>
                                      <xsd:enumeration value="Low"/>
                                      <xsd:enumeration value="unlabeled"/>
                                    </xsd:restriction>
```

```
                                    </xsd:simpleType>
                                  </xsd:attribute>
                                </xsd:complexType>
                              </xsd:element>
                            </xsd:sequence>
                            <xsd:attribute name="id" type="xsd:string" />
                          </xsd:complexType>
                        </xsd:element>
                      </xsd:sequence>
                    </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="subjectIds">
                    <xsd:complexType>
                      <xsd:sequence>
                        <xsd:element name="subject" maxOccurs="unbounded">
                          <xsd:complexType>
                            <xsd:attribute name="id" type="xsd:string"/>
                            <xsd:attribute name="clearance">
                              <xsd:simpleType>
                                <xsd:restriction base="xsd:string">
                                  <xsd:enumeration value="High"/>
                                  <xsd:enumeration value="Low"/>
                                  <xsd:enumeration value="unlabeled"/>
                                </xsd:restriction>
                              </xsd:simpleType>
                            </xsd:attribute>
                          </xsd:complexType>
                        </xsd:element>
                      </xsd:sequence>
                    </xsd:complexType>
                  </xsd:element>
                </xsd:sequence>
                <xsd:attribute name="tool" type="xsd:string" />
              </xsd:complexType>
            </xsd:element>
          </xsd:sequence>
          <xsd:attribute name="id" type="xsd:string" />
          <xsd:attribute name="type" type="xsd:string" />
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
</xsd:schema>
```

# Anhang C

# Spezifikation der verwendeten IFnet-Modelle

Zur Verbesserung der Lesbarkeit sind in den folgenden Quelltexten die `<graphics>`-Tags, die die Position eines Elementes in der grafischen Darstellung beschreiben, weggelassen worden.

## C.1 Spezifikation des IFnet-Modells aus Abbildung 5.2

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<pnml>
  <net id="ifnet" type="de.unifreiburg.iig.ifnet">

    <transition id="StartHost">
      <name>
        <text>StartHost</text>
      </name>
      <toolspecific tool="ifnet">
        <subject>
          <text>AuctionHost</text>
        </subject>
        <inputoutputoption>
          <input>
            <sourceplace id="place_0">
              <token tokenid="BLACK"/>
            </sourceplace>
          </input>
          <output>
            <targetplace id="place_1">
              <token tokenid="BLACK"/>
            </targetplace>
          </output>
        </inputoutputoption>
      </toolspecific>
    </transition>

    <transition id="IssueTicket">
      <name>
        <text>IssueTicket</text>
```

```
    </name>
    <toolspecific tool="ifnet">
      <subject>
        <text>AuctionHost</text>
      </subject>
      <inputoutputoption>
        <input>
          <sourceplace id="place_1">
            <token tokenid="BLACK"/>
          </sourceplace>
        </input>
        <output>
          <targetplace id="place_2">
            <token tokenid="BLACK"/>
          </targetplace>
          <targetplace id="place_11">
            <token tokenid="Ticket"/>
          </targetplace>
        </output>
      </inputoutputoption>
    </toolspecific>
  </transition>

  <transition id="ReceiveBid">
    <name>
      <text>Msg1</text>
    </name>
    <toolspecific tool="ifnet">
      <subject>
        <text>AuctionHost</text>
      </subject>
      <inputoutputoption>
        <input>
          <sourceplace id="place_2">
            <token tokenid="BLACK"/>
          </sourceplace>
          <sourceplace id="place_12">
            <token tokenid="Bid" accessmode="read"/>
          </sourceplace>
        </input>
        <output>
          <targetplace id="place_3">
            <token tokenid="Bid"/>
          </targetplace>
        </output>
      </inputoutputoption>
    </toolspecific>
  </transition>

  <transition id="Confirm">
    <name>
      <text>Confirm</text>
    </name>
    <toolspecific tool="ifnet">
      <subject>
        <text>AuctionHost</text>
      </subject>
      <inputoutputoption>
        <input>
          <sourceplace id="place_3">
            <token tokenid="Bid" accessmode="read"/>
          </sourceplace>
```

```
      </input>
      <output>
        <targetplace id="place_4">
          <token tokenid="Bid"/>
        </targetplace>
        <targetplace id="place_13">
          <token tokenid="Confirmation"/>
        </targetplace>
      </output>
    </inputoutputoption>
  </toolspecific>
</transition>

<transition id="Branch1">
  <name>
    <text>Branch1</text>
  </name>
  <toolspecific tool="ifnet">
    <subject>
      <text>AuctionHost</text>
    </subject>
    <guard id="notHigher">
      <token tokenid="H"/>
    </guard>
    <inputoutputoption>
      <input>
        <sourceplace id="place_4">
          <token tokenid="Bid" accessmode="read"/>
        </sourceplace>
        <sourceplace id="place_9">
          <token tokenid="H" accessmode="read"/>
        </sourceplace>
      </input>
      <output>
        <targetplace id="place_5">
          <token tokenid="BLACK"/>
        </targetplace>
        <targetplace id="place_9">
          <token tokenid="H"/>
        </targetplace>
      </output>
    </inputoutputoption>
  </toolspecific>
</transition>

<transition id="Branch2">
  <name>
    <text>Branch2</text>
  </name>
  <toolspecific tool="ifnet">
    <subject>
      <text>AuctionHost</text>
    </subject>
    <guard id="Higher">
      <token tokenid="H"/>
    </guard>
    <inputoutputoption>
      <input>
        <sourceplace id="place_4">
          <token tokenid="Bid" accessmode="read"/>
        </sourceplace>
        <sourceplace id="place_9">
```

```
          <token tokenid="H" accessmode="read"/>
        </sourceplace>
      </input>
      <output>
        <targetplace id="place_6">
          <token tokenid="Bid"/>
        </targetplace>
        <targetplace id="place_9">
          <token tokenid="H"/>
        </targetplace>
      </output>
    </inputoutputoption>
  </toolspecific>
</transition>

<transition id="Merge1">
  <name>
    <text>Merge1</text>
  </name>
  <toolspecific tool="ifnet">
    <subject>
      <text>AuctionHost</text>
    </subject>
    <inputoutputoption>
      <input>
        <sourceplace id="place_5">
          <token tokenid="BLACK"/>
        </sourceplace>
      </input>
      <output>
        <targetplace id="place_10">
          <token tokenid="BLACK"/>
        </targetplace>
      </output>
    </inputoutputoption>
  </toolspecific>
</transition>

<transition id="Update">
  <name>
    <text>Update</text>
  </name>
  <toolspecific tool="ifnet">
    <subject>
      <text>AuctionHost</text>
    </subject>
    <inputoutputoption>
      <input>
        <sourceplace id="place_6">
          <token tokenid="Bid" accessmode="read"/>
        </sourceplace>
        <sourceplace id="place_9">
          <token tokenid="H" accessmode="read,write"/>
        </sourceplace>
      </input>
      <output>
        <targetplace id="place_7">
          <token tokenid="Bid"/>
        </targetplace>
        <targetplace id="place_9">
          <token tokenid="H"/>
        </targetplace>
```

```
      </output>
    </inputoutputoption>
  </toolspecific>
</transition>

<transition id="Broadcast">
  <name>
    <text>Broadcast</text>
  </name>
  <toolspecific tool="ifnet">
    <subject>
      <text>AuctionHost</text>
    </subject>
    <inputoutputoption>
      <input>
        <sourceplace id="place_7">
          <token tokenid="Bid" accessmode="read"/>
        </sourceplace>
      </input>
      <output>
        <targetplace id="place_8">
          <token tokenid="BLACK"/>
        </targetplace>
        <targetplace id="place_14">
          <token tokenid="Status"/>
        </targetplace>
      </output>
    </inputoutputoption>
  </toolspecific>
</transition>

<transition id="Merge2">
  <name>
    <text>Merge2</text>
  </name>
  <toolspecific tool="ifnet">
    <subject>
      <text>AuctionHost</text>
    </subject>
    <inputoutputoption>
      <input>
        <sourceplace id="place_8">
          <token tokenid="BLACK"/>
        </sourceplace>
      </input>
      <output>
        <targetplace id="place_10">
          <token tokenid="BLACK"/>
        </targetplace>
      </output>
    </inputoutputoption>
  </toolspecific>
</transition>


<transition id="StartBid">
  <name>
    <text>StartBid</text>
  </name>
  <toolspecific tool="ifnet">
    <subject>
      <text>Bidder</text>
```

```
      </subject>
      <inputoutputoption>
        <input>
          <sourceplace id="place_15">
            <token tokenid="BLACK"/>
          </sourceplace>
        </input>
        <output>
          <targetplace id="place_16">
            <token tokenid="BLACK"/>
          </targetplace>
        </output>
      </inputoutputoption>
    </toolspecific>
  </transition>

  <transition id="ClaimTicket">
    <name>
      <text>ClaimTicket</text>
    </name>
    <toolspecific tool="ifnet">
      <subject>
        <text>AuctionHost</text>
      </subject>
      <inputoutputoption>
        <input>
          <sourceplace id="place_16">
            <token tokenid="BLACK"/>
          </sourceplace>
          <sourceplace id="place_11">
            <token tokenid="Ticket" accessmode="read"/>
          </sourceplace>
        </input>
        <output>
          <targetplace id="place_17">
            <token tokenid="Ticket"/>
          </targetplace>
        </output>
      </inputoutputoption>
    </toolspecific>
  </transition>

  <transition id="Bid">
    <name>
      <text>Bid</text>
    </name>
    <toolspecific tool="ifnet">
      <subject>
        <text>Bidder</text>
      </subject>
      <inputoutputoption>
        <input>
          <sourceplace id="place_17">
            <token tokenid="Ticket" accessmode="read"/>
          </sourceplace>
        </input>
        <output>
          <targetplace id="place_12">
            <token tokenid="Bid"/>
          </targetplace>
          <targetplace id="place_18">
            <token tokenid="BLACK"/>
```

```
        </targetplace>
      </output>
    </inputoutputoption>
  </toolspecific>
</transition>

<transition id="ReceiveConf">
  <name>
    <text>ReceiveConf</text>
  </name>
  <toolspecific tool="ifnet">
    <subject>
      <text>Bidder</text>
    </subject>
    <inputoutputoption>
      <input>
        <sourceplace id="place_18">
          <token tokenid="BLACK"/>
        </sourceplace>
        <sourceplace id="place_13">
          <token tokenid="Confirmation" accessmode="read"/>
        </sourceplace>
      </input>
      <output>
        <targetplace id="place_19">
          <token tokenid="BLACK"/>
        </targetplace>
      </output>
    </inputoutputoption>
  </toolspecific>
</transition>

<transition id="ReceiveStatus">
  <name>
    <text>ReceiveStatus</text>
  </name>
  <toolspecific tool="ifnet">
    <subject>
      <text>Bidder</text>
    </subject>
    <inputoutputoption>
      <input>
        <sourceplace id="place_14">
          <token tokenid="Status" accessmode="read"/>
        </sourceplace>
      </input>
      <output>
        <targetplace id="place_20">
          <token tokenid="BLACK"/>
        </targetplace>
      </output>
    </inputoutputoption>
  </toolspecific>
</transition>


<place id="place_0"/>
<place id="place_1"/>
<place id="place_2"/>
<place id="place_3"/>
<place id="place_4"/>
<place id="place_5"/>
```

```
<place id="place_6"/>
<place id="place_7"/>
<place id="place_8"/>
<place id="place_9"/>
<place id="place_10"/>

<place id="place_11"/>
<place id="place_12"/>
<place id="place_13"/>
<place id="place_14"/>

<place id="place_15"/>
<place id="place_16"/>
<place id="place_17"/>
<place id="place_18"/>
<place id="place_19"/>
<place id="place_20"/>


<arc id="arc_0" source="place_0" target="StartHost"/>
<arc id="arc_1" source="StartHost" target="place_1"/>
<arc id="arc_2" source="place_1" target="IssueTicket"/>
<arc id="arc_3" source="IssueTicket" target="place_2"/>
<arc id="arc_4" source="place_2" target="ReceiveBid"/>
<arc id="arc_5" source="ReceiveBid" target="place_3"/>
<arc id="arc_6" source="place_3" target="Confirm"/>
<arc id="arc_7" source="Confirm" target="place_4"/>
<arc id="arc_8" source="place_4" target="Branch1"/>
<arc id="arc_9" source="Branch1" target="place_5"/>
<arc id="arc_10" source="place_5" target="Merge1"/>
<arc id="arc_11" source="Merge1" target="place_10"/>
<arc id="arc_12" source="place_4" target="Branch2"/>
<arc id="arc_13" source="Branch2" target="place_6"/>
<arc id="arc_14" source="place_6" target="Update"/>
<arc id="arc_15" source="Update" target="place_7"/>
<arc id="arc_16" source="place_7" target="Broadcast"/>
<arc id="arc_17" source="Broadcast" target="place_8"/>
<arc id="arc_18" source="place_8" target="Merge2"/>
<arc id="arc_19" source="Merge2" target="place_10"/>

<arc id="arc_20" source="Branch1" target="place_9"/>
<arc id="arc_21" source="place_9" target="Branch1"/>
<arc id="arc_22" source="Branch2" target="place_9"/>
<arc id="arc_23" source="place_9" target="Branch2"/>
<arc id="arc_24" source="Update" target="place_9"/>
<arc id="arc_25" source="place_9" target="Update"/>

<arc id="arc_26" source="place_15" target="StartBid"/>
<arc id="arc_27" source="StartBid" target="place_16"/>
<arc id="arc_28" source="place_16" target="ClaimTicket"/>
<arc id="arc_29" source="ClaimTicket" target="place_17"/>
<arc id="arc_30" source="place_17" target="Bid"/>
<arc id="arc_31" source="Bid" target="place_18"/>
<arc id="arc_32" source="place_18" target="ReceiveConf"/>
<arc id="arc_33" source="ReceiveConf" target="place_19"/>
<arc id="arc_34" source="ReceiveStatus" target="place_20"/>

<arc id="arc_35" source="IssueTicket" target="place_11"/>
<arc id="arc_36" source="place_11" target="ClaimTicket"/>
<arc id="arc_37" source="Bid" target="place_12"/>
<arc id="arc_38" source="place_12" target="ReceiveBid"/>
<arc id="arc_39" source="Confirm" target="place_13"/>
```

```
    <arc id="arc_40" source="place_13" target="ReceiveConf"/>
    <arc id="arc_41" source="Broadcast" target="place_14"/>
    <arc id="arc_42" source="place_14" target="ReceiveStatus"/>


    <toolspecific tool="ifnet">
      <initialMarking>
        <placemarking id="place_0">
          <token tokenid="BLACK"/>
        </placemarking>
        <placemarking id="place_15">
          <token tokenid="BLACK"/>
        </placemarking>
        <placemarking id="place_9">
          <token tokenid="H"/>
        </placemarking>
      </initialMarking>
      <subjectIds>
        <subject id="AuctionHost"/>
        <subject id="Bidder"/>
      </subjectIds>
    </toolspecific>

  </net>
</pnml>
```

## C.2 Spezifikation des IFnet-Modells aus Abbildung 5.3

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<pnml>
  <net id="ifnet" type="de.unifreiburg.iig.ifnet">

    <transition id="StartHost">
      <name>
        <text>StartHost</text>
      </name>
      <toolspecific tool="ifnet">
        <subject>
          <text>AuctionHost</text>
        </subject>
        <seclevel>
          <text>Low</text>
        </seclevel>
        <inputoutputoption>
          <input>
            <sourceplace id="place_0">
              <token tokenid="BLACK"/>
            </sourceplace>
          </input>
          <output>
            <targetplace id="place_1">
              <token tokenid="BLACK"/>
            </targetplace>
          </output>
        </inputoutputoption>
      </toolspecific>
    </transition>

    <transition id="IssueTicket">
      <name>
        <text>IssueTicket</text>
      </name>
      <toolspecific tool="ifnet">
        <subject>
          <text>AuctionHost</text>
        </subject>
        <seclevel>
          <text>Low</text>
        </seclevel>
        <inputoutputoption>
          <input>
            <sourceplace id="place_1">
              <token tokenid="BLACK"/>
            </sourceplace>
          </input>
          <output>
            <targetplace id="place_2">
              <token tokenid="BLACK"/>
            </targetplace>
            <targetplace id="place_11">
              <token tokenid="Ticket"/>
            </targetplace>
          </output>
        </inputoutputoption>
      </toolspecific>
    </transition>
```

```
<transition id="ReceiveBid">
  <name>
    <text>Msg1</text>
  </name>
  <toolspecific tool="ifnet">
    <subject>
      <text>AuctionHost</text>
    </subject>
    <seclevel>
      <text>Low</text>
    </seclevel>
    <inputoutputoption>
      <input>
        <sourceplace id="place_2">
          <token tokenid="BLACK"/>
        </sourceplace>
        <sourceplace id="place_12">
          <token tokenid="Bid" accessmode="read"/>
        </sourceplace>
      </input>
      <output>
        <targetplace id="place_3">
          <token tokenid="Bid"/>
        </targetplace>
      </output>
    </inputoutputoption>
    <inputoutputoption>
      <input>
        <sourceplace id="place_2">
          <token tokenid="BLACK"/>
        </sourceplace>
        <sourceplace id="place_12">
          <token tokenid="Bid2" accessmode="read"/>
        </sourceplace>
      </input>
      <output>
        <targetplace id="place_3">
          <token tokenid="Bid2"/>
        </targetplace>
      </output>
    </inputoutputoption>
  </toolspecific>
</transition>

<transition id="Confirm">
  <name>
    <text>Confirm</text>
  </name>
  <toolspecific tool="ifnet">
    <subject>
      <text>AuctionHost</text>
    </subject>
    <seclevel>
      <text>Low</text>
    </seclevel>
    <inputoutputoption>
      <input>
        <sourceplace id="place_3">
          <token tokenid="Bid" accessmode="read"/>
        </sourceplace>
      </input>
      <output>
```

```
          <targetplace id="place_4">
            <token tokenid="Bid"/>
          </targetplace>
          <targetplace id="place_13">
            <token tokenid="Confirmation"/>
          </targetplace>
        </output>
      </inputoutputoption>
      <inputoutputoption>
        <input>
          <sourceplace id="place_3">
            <token tokenid="Bid2" accessmode="read"/>
          </sourceplace>
        </input>
        <output>
          <targetplace id="place_4">
            <token tokenid="Bid2"/>
          </targetplace>
          <targetplace id="place_13">
            <token tokenid="Confirmation2"/>
          </targetplace>
        </output>
      </inputoutputoption>
    </toolspecific>
  </transition>

  <transition id="Branch1">
    <name>
      <text>Branch1</text>
    </name>
    <toolspecific tool="ifnet">
      <subject>
        <text>AuctionHost</text>
      </subject>
      <seclevel>
        <text>Low</text>
      </seclevel>
      <guard id="notHigher">
        <token tokenid="H"/>
      </guard>
      <inputoutputoption>
        <input>
          <sourceplace id="place_4">
            <token tokenid="Bid" accessmode="read"/>
          </sourceplace>
          <sourceplace id="place_9">
            <token tokenid="H" accessmode="read"/>
          </sourceplace>
        </input>
        <output>
          <targetplace id="place_5">
            <token tokenid="BLACK"/>
          </targetplace>
          <targetplace id="place_9">
            <token tokenid="H"/>
          </targetplace>
        </output>
      </inputoutputoption>
      <inputoutputoption>
        <input>
          <sourceplace id="place_4">
            <token tokenid="Bid2" accessmode="read"/>
```

```
        </sourceplace>
        <sourceplace id="place_9">
          <token tokenid="H" accessmode="read"/>
        </sourceplace>
      </input>
      <output>
        <targetplace id="place_5">
          <token tokenid="BLACK"/>
        </targetplace>
        <targetplace id="place_9">
          <token tokenid="H"/>
        </targetplace>
      </output>
    </inputoutputoption>
  </toolspecific>
</transition>

<transition id="Branch2">
  <name>
    <text>Branch2</text>
  </name>
  <toolspecific tool="ifnet">
    <subject>
      <text>AuctionHost</text>
    </subject>
    <seclevel>
      <text>Low</text>
    </seclevel>
    <guard id="Higher">
      <token tokenid="H"/>
    </guard>
    <inputoutputoption>
      <input>
        <sourceplace id="place_4">
          <token tokenid="Bid" accessmode="read"/>
        </sourceplace>
        <sourceplace id="place_9">
          <token tokenid="H" accessmode="read"/>
        </sourceplace>
      </input>
      <output>
        <targetplace id="place_6">
          <token tokenid="Bid"/>
        </targetplace>
        <targetplace id="place_9">
          <token tokenid="H"/>
        </targetplace>
      </output>
    </inputoutputoption>
    <inputoutputoption>
      <input>
        <sourceplace id="place_4">
          <token tokenid="Bid2" accessmode="read"/>
        </sourceplace>
        <sourceplace id="place_9">
          <token tokenid="H" accessmode="read"/>
        </sourceplace>
      </input>
      <output>
        <targetplace id="place_6">
          <token tokenid="Bid2"/>
        </targetplace>
```

```
          <targetplace id="place_9">
            <token tokenid="H"/>
          </targetplace>
        </output>
      </inputoutputoption>
    </toolspecific>
  </transition>

  <transition id="Merge1">
    <name>
      <text>Merge1</text>
    </name>
    <toolspecific tool="ifnet">
      <subject>
        <text>AuctionHost</text>
      </subject>
      <seclevel>
        <text>Low</text>
      </seclevel>
      <inputoutputoption>
        <input>
          <sourceplace id="place_5">
            <token tokenid="BLACK"/>
          </sourceplace>
        </input>
        <output>
          <targetplace id="place_10">
            <token tokenid="BLACK"/>
          </targetplace>
        </output>
      </inputoutputoption>
    </toolspecific>
  </transition>

  <transition id="Update">
    <name>
      <text>Update</text>
    </name>
    <toolspecific tool="ifnet">
      <subject>
        <text>AuctionHost</text>
      </subject>
      <seclevel>
        <text>Low</text>
      </seclevel>
      <inputoutputoption>
        <input>
          <sourceplace id="place_6">
            <token tokenid="Bid" accessmode="read"/>
          </sourceplace>
          <sourceplace id="place_9">
            <token tokenid="H" accessmode="read,write"/>
          </sourceplace>
        </input>
        <output>
          <targetplace id="place_7">
            <token tokenid="Bid"/>
          </targetplace>
          <targetplace id="place_9">
            <token tokenid="H"/>
          </targetplace>
        </output>
```

```
    </inputoutputoption>
    <inputoutputoption>
      <input>
        <sourceplace id="place_6">
          <token tokenid="Bid2" accessmode="read"/>
        </sourceplace>
        <sourceplace id="place_9">
          <token tokenid="H" accessmode="read,write"/>
        </sourceplace>
      </input>
      <output>
        <targetplace id="place_7">
          <token tokenid="Bid2"/>
        </targetplace>
        <targetplace id="place_9">
          <token tokenid="H"/>
        </targetplace>
      </output>
    </inputoutputoption>
  </toolspecific>
</transition>

<transition id="Broadcast">
  <name>
    <text>Broadcast</text>
  </name>
  <toolspecific tool="ifnet">
    <subject>
      <text>AuctionHost</text>
    </subject>
    <seclevel>
      <text>Low</text>
    </seclevel>
    <inputoutputoption>
      <input>
        <sourceplace id="place_7">
          <token tokenid="Bid" accessmode="read"/>
        </sourceplace>
      </input>
      <output>
        <targetplace id="place_8">
          <token tokenid="BLACK"/>
        </targetplace>
        <targetplace id="place_14">
          <token tokenid="Status"/>
        </targetplace>
      </output>
    </inputoutputoption>
    <inputoutputoption>
      <input>
        <sourceplace id="place_7">
          <token tokenid="Bid2" accessmode="read"/>
        </sourceplace>
      </input>
      <output>
        <targetplace id="place_8">
          <token tokenid="BLACK"/>
        </targetplace>
        <targetplace id="place_14">
          <token tokenid="Status2"/>
        </targetplace>
      </output>
```

```
      </inputoutputoption>
    </toolspecific>
  </transition>

  <transition id="Merge2">
    <name>
      <text>Merge2</text>
    </name>
    <toolspecific tool="ifnet">
      <subject>
        <text>AuctionHost</text>
      </subject>
      <seclevel>
        <text>Low</text>
      </seclevel>
      <inputoutputoption>
        <input>
          <sourceplace id="place_8">
            <token tokenid="BLACK"/>
          </sourceplace>
        </input>
        <output>
          <targetplace id="place_10">
            <token tokenid="BLACK"/>
          </targetplace>
        </output>
      </inputoutputoption>
    </toolspecific>
  </transition>

  <transition id="StartBid">
    <name>
      <text>StartBid</text>
    </name>
    <toolspecific tool="ifnet">
      <subject>
        <text>Bidder</text>
      </subject>
      <seclevel>
        <text>High</text>
      </seclevel>
      <inputoutputoption>
        <input>
          <sourceplace id="place_15">
            <token tokenid="BLACK"/>
          </sourceplace>
        </input>
        <output>
          <targetplace id="place_16">
            <token tokenid="BLACK"/>
          </targetplace>
        </output>
      </inputoutputoption>
    </toolspecific>
  </transition>

  <transition id="ClaimTicket">
    <name>
      <text>ClaimTicket</text>
    </name>
    <toolspecific tool="ifnet">
      <subject>
```

```
      <text>AuctionHost</text>
    </subject>
    <seclevel>
      <text>High</text>
    </seclevel>
    <inputoutputoption>
      <input>
        <sourceplace id="place_16">
          <token tokenid="BLACK"/>
        </sourceplace>
        <sourceplace id="place_11">
          <token tokenid="Ticket" accessmode="read"/>
        </sourceplace>
      </input>
      <output>
        <targetplace id="place_17">
          <token tokenid="Ticket"/>
        </targetplace>
      </output>
    </inputoutputoption>
  </toolspecific>
</transition>

<transition id="Bid">> <name><text>Bid</text></name>
  <toolspecific tool="ifnet">
    <subject>
      <text>Bidder</text>
    </subject>
    <seclevel>
      <text>High</text>
    </seclevel>
    <inputoutputoption>
      <input>
        <sourceplace id="place_17">
          <token tokenid="Ticket" accessmode="read"/>
        </sourceplace>
      </input>
      <output>
        <targetplace id="place_12">
          <token tokenid="Bid"/>
        </targetplace>
        <targetplace id="place_18">
          <token tokenid="BLACK"/>
        </targetplace>
      </output>
    </inputoutputoption>
  </toolspecific>
</transition>

<transition id="ReceiveConf">
  <name>
    <text>ReceiveConf</text>
  </name>
  <toolspecific tool="ifnet">
    <subject>
      <text>Bidder</text>
    </subject>
    <seclevel>
      <text>High</text>
    </seclevel>
    <inputoutputoption>
      <input>
```

```
          <sourceplace id="place_18">
            <token tokenid="BLACK"/>
          </sourceplace>
          <sourceplace id="place_13">
            <token tokenid="Confirmation" accessmode="read"/>
          </sourceplace>
        </input>
        <output>
          <targetplace id="place_19">
            <token tokenid="BLACK"/>
          </targetplace>
        </output>
      </inputoutputoption>
    </toolspecific>
  </transition>

  <transition id="ReceiveStatus">
    <name>
      <text>ReceiveStatus</text>
    </name>
    <toolspecific tool="ifnet">
      <subject>
        <text>Bidder</text>
      </subject>
      <seclevel>
        <text>High</text>
      </seclevel>
      <inputoutputoption>
        <input>
          <sourceplace id="place_14">
            <token tokenid="Status" accessmode="read"/>
          </sourceplace>
        </input>
        <output>
          <targetplace id="place_20">
            <token tokenid="BLACK"/>
          </targetplace>
        </output>
      </inputoutputoption>
    </toolspecific>
  </transition>

  <transition id="StartBid2">
    <name>
      <text>StartBid2</text>
    </name>
    <toolspecific tool="ifnet">
      <subject>
        <text>Bidder2</text>
      </subject>
      <seclevel>
        <text>Low</text>
      </seclevel>
      <inputoutputoption>
        <input>
          <sourceplace id="place_21">
            <token tokenid="BLACK"/>
          </sourceplace>
        </input>
        <output>
          <targetplace id="place_22">
            <token tokenid="BLACK"/>
```

```
        </targetplace>
      </output>
    </inputoutputoption>
  </toolspecific>
</transition>

<transition id="ClaimTicket2">
  <name>
    <text>ClaimTicket2</text>
  </name>
  <toolspecific tool="ifnet">
    <subject>
      <text>Bidder2</text>
    </subject>
    <seclevel>
      <text>Low</text>
    </seclevel>
    <inputoutputoption>
      <input>
        <sourceplace id="place_22">
          <token tokenid="BLACK"/>
        </sourceplace>
        <sourceplace id="place_11">
          <token tokenid="Ticket" accessmode="read"/>
        </sourceplace>
      </input>
      <output>
        <targetplace id="place_23">
          <token tokenid="Ticket"/>
        </targetplace>
      </output>
    </inputoutputoption>
  </toolspecific>
</transition>

<transition id="Bid2">
  <name>
    <text>Bid2</text>
  </name>
  <toolspecific tool="ifnet">
    <subject>
      <text>Bidder2</text>
    </subject>
    <seclevel>
      <text>Low</text>
    </seclevel>
    <inputoutputoption>
      <input>
        <sourceplace id="place_23">
          <token tokenid="Ticket" accessmode="read"/>
        </sourceplace>
      </input>
      <output>
        <targetplace id="place_12">
          <token tokenid="Bid2"/>
        </targetplace>
        <targetplace id="place_24">
          <token tokenid="BLACK"/>
        </targetplace>
      </output>
    </inputoutputoption>
  </toolspecific>
```

```
</transition>

<transition id="ReceiveConf2">
  <name>
    <text>ReceiveConf2</text>
  </name>
  <toolspecific tool="ifnet">
    <subject>
      <text>Bidder2</text>
    </subject>
    <seclevel>
      <text>Low</text>
    </seclevel>
    <inputoutputoption>
      <input>
        <sourceplace id="place_24">
          <token tokenid="BLACK"/>
        </sourceplace>
        <sourceplace id="place_13">
          <token tokenid="Confirmation2" accessmode="read"/>
        </sourceplace>
      </input>
      <output>
        <targetplace id="place_25">
          <token tokenid="BLACK"/>
        </targetplace>
      </output>
    </inputoutputoption>
  </toolspecific>
</transition>

<transition id="ReceiveStatus2">
  <name>
    <text>ReceiveStatus2</text>
  </name>
  <toolspecific tool="ifnet">
    <subject>
      <text>Bidder2</text>
    </subject>
    <seclevel>
      <text>Low</text>
    </seclevel>
    <inputoutputoption>
      <input>
        <sourceplace id="place_14">
          <token tokenid="Status2" accessmode="read"/>
        </sourceplace>
      </input>
      <output>
        <targetplace id="place_26">
          <token tokenid="BLACK"/>
        </targetplace>
      </output>
    </inputoutputoption>
  </toolspecific>
</transition>


<place id="place_0"/>
<place id="place_1"/>
<place id="place_2"/>
<place id="place_3"/>
```

```
<place id="place_4"/>
<place id="place_5"/>
<place id="place_6"/>
<place id="place_7"/>
<place id="place_8"/>
<place id="place_9"/>
<place id="place_10"/>

<place id="place_11"/>
<place id="place_12"/>
<place id="place_13"/>
<place id="place_14"/>

<place id="place_15"/>
<place id="place_16"/>
<place id="place_17"/>
<place id="place_18"/>
<place id="place_19"/>
<place id="place_20"/>

<place id="place_21"/>
<place id="place_22"/>
<place id="place_23"/>
<place id="place_24"/>
<place id="place_25"/>
<place id="place_26"/>


<arc id="arc_0" source="place_0" target="StartHost"/>
<arc id="arc_1" source="StartHost" target="place_1"/>
<arc id="arc_2" source="place_1" target="IssueTicket"/>
<arc id="arc_3" source="IssueTicket" target="place_2"/>
<arc id="arc_4" source="place_2" target="ReceiveBid"/>
<arc id="arc_5" source="ReceiveBid" target="place_3"/>
<arc id="arc_6" source="place_3" target="Confirm"/>
<arc id="arc_7" source="Confirm" target="place_4"/>
<arc id="arc_8" source="place_4" target="Branch1"/>
<arc id="arc_9" source="Branch1" target="place_5"/>
<arc id="arc_10" source="place_5" target="Merge1"/>
<arc id="arc_11" source="Merge1" target="place_10"/>
<arc id="arc_12" source="place_4" target="Branch2"/>
<arc id="arc_13" source="Branch2" target="place_6"/>
<arc id="arc_14" source="place_6" target="Update"/>
<arc id="arc_15" source="Update" target="place_7"/>
<arc id="arc_16" source="place_7" target="Broadcast"/>
<arc id="arc_17" source="Broadcast" target="place_8"/>
<arc id="arc_18" source="place_8" target="Merge2"/>
<arc id="arc_19" source="Merge2" target="place_10"/>

<arc id="arc_20" source="Branch1" target="place_9"/>
<arc id="arc_21" source="place_9" target="Branch1"/>
<arc id="arc_22" source="Branch2" target="place_9"/>
<arc id="arc_23" source="place_9" target="Branch2"/>
<arc id="arc_24" source="Update" target="place_9"/>
<arc id="arc_25" source="place_9" target="Update"/>

<arc id="arc_26" source="place_15" target="StartBid"/>
<arc id="arc_27" source="StartBid" target="place_16"/>
<arc id="arc_28" source="place_16" target="ClaimTicket"/>
<arc id="arc_29" source="ClaimTicket" target="place_17"/>
<arc id="arc_30" source="place_17" target="Bid"/>
<arc id="arc_31" source="Bid" target="place_18"/>
```

```
    <arc id="arc_32" source="place_18" target="ReceiveConf"/>
    <arc id="arc_33" source="ReceiveConf" target="place_19"/>
    <arc id="arc_34" source="ReceiveStatus" target="place_20"/>

    <arc id="arc_35" source="IssueTicket" target="place_11"/>
    <arc id="arc_36" source="place_11" target="ClaimTicket"/>
    <arc id="arc_37" source="Bid" target="place_12"/>
    <arc id="arc_38" source="place_12" target="ReceiveBid"/>
    <arc id="arc_39" source="Confirm" target="place_13"/>
    <arc id="arc_40" source="place_13" target="ReceiveConf"/>
    <arc id="arc_41" source="Broadcast" target="place_14"/>
    <arc id="arc_42" source="place_14" target="ReceiveStatus"/>

    <arc id="arc_43" source="place_21" target="StartBid2"/>
    <arc id="arc_44" source="StartBid2" target="place_22"/>
    <arc id="arc_45" source="place_22" target="ClaimTicket2"/>
    <arc id="arc_46" source="ClaimTicket2" target="place_23"/>
    <arc id="arc_47" source="place_23" target="Bid2"/>
    <arc id="arc_48" source="Bid2" target="place_24"/>
    <arc id="arc_49" source="place_24" target="ReceiveConf2"/>
    <arc id="arc_50" source="ReceiveConf2" target="place_25"/>
    <arc id="arc_51" source="ReceiveStatus2" target="place_26"/>

    <arc id="arc_52" source="place_11" target="ClaimTicket2"/>
    <arc id="arc_53" source="Bid2" target="place_12"/>
    <arc id="arc_54" source="place_13" target="ReceiveConf2"/>
    <arc id="arc_55" source="place_14" target="ReceiveStatus2"/>


    <toolspecific tool="ifnet">
      <initialMarking>
        <placemarking id="place_0">
          <token tokenid="BLACK"/>
        </placemarking>
        <placemarking id="place_15">
          <token tokenid="BLACK"/>
        </placemarking>
        <placemarking id="place_21">
          <token tokenid="BLACK"/>
        </placemarking>
        <placemarking id="place_9">
          <token tokenid="H"/>
        </placemarking>
      </initialMarking>
      <subjectIds>
        <subject id="AuctionHost" clearance="unlabeled"/>
        <subject id="Bidder" clearance="High"/>
        <subject id="Bidder2" clearance="Low"/>
      </subjectIds>
    </toolspecific>

  </net>
</pnml>
```

# Literaturverzeichnis


[van der Aalst 1995] Aalst, Willibrordus Martinus P. van der: A class of Petri net for modeling and analyzing business processes / Eindhoven University of Technology. 1995 (95/32). – Forschungsbericht

[van der Aalst 1996] Aalst, Willibrordus Martinus P. van der: Structural Characterizations of Sound Workflow Nets / Eindhoven University of Technology. 1996 (96/23). – Forschungsbericht

[van der Aalst 2005] Aalst, Willibrordus Martinus P. van der: Pi calculus versus Petri nets: Let us eat "humble pie" rather than further inflate the "Pi hype". In: *BPTrends* 3 (2005), Nr. 5, S. 1–11

[van der Aalst u. a. 2007] Aalst, Willibrordus Martinus P. van der ; Benatallah, Boualem ; Casati, Fabio ; Curbera, Francisco ; Verbe, Eric: "Business process management: Where business processes and web services meet". In: *"Data & Knowledge Engineering"* 61 (2007), Nr. 1, S. 1–5

[van der Aalst und van Dongen 2002] Aalst, Willibrordus Martinus P. van der ; Dongen, Boudewijn F. van: Discovering Workflow Performance Models from Timed Logs. In: Han, Yanbo (Hrsg.) ; Tai, Stefan (Hrsg.) ; Wikarski, Dietmar (Hrsg.): *Engineering and Deployment of Cooperative Information Systems* Bd. 2480, Springer-Verlag, 2002, S. 45–63

[van der Aalst und van Hee 2004] Aalst, Willibrordus Martinus P. van der ; Hee, Kees van: *Workflow Management: Models, Methods, and Systems.* MIT Press, 2004 (Cooperative Information Systems)

[van der Aalst u. a. 2003a] Aalst, Willibrordus Martinus P. van der ; Hofstede, Arthur H. M. ter ; Kiepuszewski, Bartek ; Barros, Alistair P.: Workflow Patterns. In: *Distributed and Parallel Databases* 14 (2003), Nr. 3, S. 5–51

[van der Aalst u. a. 2003b] Aalst, Willibrordus Martinus P. van der ; Hofstede, Arthur H. M. ter ; Weske, Mathias: Business Process Management: A Survey. In: *International Conference on Business Process Management* Bd. 2678, Springer-Verlag, 2003, S. 1–12

[van der Aalst und Weijters 2004] Aalst, Willibrordus Martinus P. van der ; Weijters, A. J. M. M.: Process mining: A research agenda. In: *Computers in Industry* 53 (2004), Nr. 3, S. 231–244

[van der Aalst u. a. 2004] Aalst, Willibrordus Martinus P. van der ; Weijters, Ton ; Maruster, Laura: Workflow Mining: Discovering Process Models from Event Logs. In: *Transactions on Knowledge and Data Engineering* 16 (2004), Nr. 9, S. 1128–1142

[Adam u. a. 1998] Adam, Nabil R. ; Atluri, Vijayalakshmi ; Huang, Wei-Kuang: Modeling and Analysis of Workflows Using Petri Nets. In: *Journal of Intelligent Information Systems* 10 (1998), Nr. 2, S. 131–158

[AICPA 1993] American Institute of Certified Public Accountants (AICPA): *Statement on Auditing Standards No. 70: Service Organizations.* 1993. – Abrufbar unter `http://www.aicpa.org` (Februar 2011)

[Andrews u. a. 2003] Andrews, Tony ; Curbera, Francisco ; Dholakia, Hitesh ; Goland, Yaron ; Klein, Johannes ; Leymann, Frank ; Liu, Kevin ; Roller, Dieter ; Smith, Doug ; Thatte, Satish ; Trickovic, Ivana ; Weerawarana, Sanjiva: *Business Process Execution Language for Web Services Version 1.1.* 2003. – Abrufbar unter `http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel/ws-bpel.pdf` (November 2010)

[Armando und Ponta 2010] Armando, Alessandro ; Ponta, Serena E.: Model Checking of Security-Sensitive Business Processes. In: Degano, Pierpaolo (Hrsg.) ; Guttman, Joshua D. (Hrsg.): *6th International Workshop on Formal Aspects in Security and Trust* Bd. 5983, Springer-Verlag, 2010, S. 66–80

[Asendorpf 2011] Asendorpf, Dirk: Ab in die Wolken. In: *Die Zeit* (2011), 17. Februar, Nr. 08, S. 39–40

[Ashley u. a. 2003] Ashley, Paul ; Hada, Satoshi ; Karjoth, Günter ; Powers, Calvin ; Schunter, Matthias: *Enterprise Privacy Authorization Language (EPAL 1.2) – W3C Member Submission.* 2003. – Abrufbar unter `http://www.w3.org/Submission/2003/SUBM-EPAL-20031110/` (Januar 2011)

[Atluri u. a. 2004] Atluri, Vijayalakshmi ; Chun, Soon A. ; Mazzoleni, Pietro: Chinese wall security for decentralized workflow management systems. In: *Journal of Computer Security* 12 (2004), Nr. 6, S. 799–840

[Atluri und Huang 1996] Atluri, Vijayalakshmi ; Huang, Wei-Kuang: An authorization model for workflows. In: Bertino, Elisa (Hrsg.) ; Kurth, Helmut (Hrsg.) ; Martella, Giancarlo (Hrsg.) ; Montolivo, Emilio (Hrsg.): *4th European Symposium on Research in Computer Security* Bd. 1146, Springer-Verlag, 1996, S. 44–64

[Atluri und kuang Huang 1997] Atluri, Vijayalakshmi ; Huang, Wei kuang: Enforcing Mandatory and Discretionary Security in Workflow Management Systems. In: *Journal of Computer Security* 5 (1997), Nr. 4, S. 303–340

[Bace und Rozwell 2006] Bace, John ; Rozwell, Carol: Understanding the Components of Compliance / Gartner Research. July 2006. – Forschungsbericht

[Bace u. a. 2006] Bace, John ; Rozwell, Carol ; Feiman, Joseph ; Kirwin, Bill: Understanding the Costs of Compliance / Gartner Research. July 2006. – Forschungsbericht

[Bacon u. a. 2010] Bacon, Jean ; Evans, David ; Eyers, David M. ; Migliavacca, Matteo ; Pietzuch, Peter R. ; Shand, Brian: Enforcing End-to-End Application Security in the Cloud. In: Gupta, Indranil (Hrsg.) ; Mascolo, Cecilia (Hrsg.): *International Middleware Conference* Bd. 6452, Springer-Verlag, 2010, S. 293–312

[Baeten 2005] Baeten, Jos C. M.: A brief history of process algebra. In: *Theoretical Computer Science* 335 (2005), Nr. 2-3, S. 131–146

[Banks u. a. 2009] Banks, David ; Erickson, John ; Rhodes, Michael: Multi-Tenancy in Cloud-Based Collaboration Services / Hewlett-Packard. 2009 (HPL-2009-17). – Forschungsbericht

[Barkaoui u. a. 2008] Barkaoui, Kamel ; Ayed, Rahma B. ; Boucheneb, Hanifa ; Hicheur, Awatef: Verification of Workflow processes under multilevel security considerations. In: Jmaiel, Mohamed (Hrsg.) ; Mosbah, Mohamed (Hrsg.): *Third International Conference on Risks and Security of Internet and Systems*, IEEE, 2008, S. 77–84

[Becker und Schütte 1999] Becker, Jörg ; Schütte, Reinhard: *Handelsinformationssysteme*. Moderne Industrie, 1999

[Bell und LaPadula 1976] Bell, David E. ; LaPadula, Leonard J.: Computer Security Model: Unified Exposition and Multics Interpretation / MITRE Corporation. 1976 (MTR-2997). – Forschungsbericht

[Ben-Eliahu 2008] Ben-Eliahu, Ziv: *Business Processes of the world unite*. 2008. – Abrufbar unter `http://www.cs.bgu.ac.il/~bpmn/BPDs` (Mai 2011)

[Benantar 2010] Benantar, Messaoud (Hrsg.): *Access Control Systems: Security, Identity Management and Trust Models* . Springer-Verlag, 2010

[Bertino u. a. 1999] Bertino, Elisa ; Ferrari, Elena ; Atluri, Vijayalakshmi: The specification and enforcement of authorization constraints in workflow management systems. In: *ACM Transactions on Information and System Security* 2 (1999), Nr. 1, S. 65–104

[Bettaz und Maouche 1993] Bettaz, Mohamed ; Maouche, Mourad: How to specify non determinism and true concurrency with algebraic term nets. In: Bidoit, Michel (Hrsg.) ; Choppy, Christine (Hrsg.): *Recent Trends in Data Type Specification* Bd. 655, Springer-Verlag, 1993, S. 164–180

[Boyd und Mao 2000] Boyd, Colin ; Mao, Wenbo: Security Issues for Electronic Auctions / Hewlett-Packard Company. 2000 (HPL-2000-90). – Forschungsbericht

[Breaux und Antón 2008] Breaux, Travis D. ; Antón, Annie I.: Analyzing Regulatory Rules for Privacy and Security Requirements. In: *IEEE Transactions on Software Engineering* 34 (2008), Nr. 1, S. 5–20

[Breaux u. a. 2006] Breaux, Travis D. ; Anton, Annie I. ; Karat, Clare-Marie ; Karat, John: Enforceability vs. Accountability in Electronic Policies. In: *Proceedings of the Seventh IEEE International Workshop on Policies for Distributed Systems and Networks*, IEEE Computer Society, June 2006, S. 227–230

[van Breugel und Koshkina 2006] Breugel, Franck van ; Koshkina, Maria: *Models and Verification of BPEL*. 2006. – Abrufbar unter `http://www.cse.yorku.ca/ franck/research/drafts/tutorial.pdf` (November 2010)

[Brewer und Nash 1989] Brewer, David F. ; Nash, Michael J.: The Chinese Wall Security Policy. In: *IEEE Symposium on Security and Privacy*, IEEE, 1989, S. 206–214

[Briegleb 2008] Briegleb, Volker: Datenschutz: Schäuble drängt auf Selbstverpflichtung der Telecom-Branche. In: *heise online* (2008), 30. Mai

[vom Brocke und Rosemann 2010] Brocke, Jan vom (Hrsg.) ; Rosemann, Michael (Hrsg.): *Handbook on Business Process Management 1 – Introduction, Methods, and Information Systems*. Springer-Verlag, 2010

[Brumley und Boneh 2005] Brumley, David ; Boneh, Dan: Remote timing attacks are practical. In: *Computer Networks* 48 (2005), Nr. 5, S. 701–716

[BSI 2010] Bundesamt für Sicherheit in der Informationstechnik (Veranst.): *BSI-Mindestsicherheitsanforderungen an Cloud-Computing-Anbieter*. 2010. – Entwurf vom 27.09.2010

[Bundesdatenschutzgesetz 2003] *Bundesdatenschutzgesetz (BDSG), Fassung der Bekanntmachung vom 14. Januar 2003*. 2003. – Abrufbar unter `http://bundesrecht.juris.de/bdsg_1990/BJNR029550990.html` (Februar 2011)

[Busi und Gorrieri 2004] Busi, Nadia ; Gorrieri, Roberto: A Survey on Non-interference with Petri Nets. In: Desel, Jörg (Hrsg.) ; Reisig, Wolfgang (Hrsg.) ; Rozenberg, Grzegorz (Hrsg.): *Lectures on Concurrency and Petri Nets* Bd. 3098, Springer-Verlag, 2004, S. 328–344

[Busi und Gorrieri 2009] Busi, Nadia ; Gorrieri, Roberto: Structural non-interference in elementary and trace nets. In: *Mathematical Structures in Computer Science* 19 (2009), Nr. 6, S. 1065–1090

[Buyya u. a. 2002] Buyya, Rajkumar ; Abramson, David ; Giddy, Jonathan ; Stockinger, Heinz: Economic models for resource management and scheduling in Grid computing. In: *Concurrency and Computation: Practice and Experience* 14 (2002), Nr. 13-15, S. 1507–1542

[Carlin und Gallegos 2007] Carlin, Anna ; Gallegos, Frederick: IT Audit: A Critical Business Process. In: *IEEE Computer* 40 (2007), S. 87–89

[Casati u. a. 2001] Casati, Fabio ; Castano, Silvana ; Fugini, MariaGrazia: Managing Workflow Authorization Constraints through Active Database Technology. In: *Information Systems Frontiers* 3 (2001), Nr. 3, S. 319–338

[Cassez und Roux 2006] Cassez, Franck ; Roux, Olivier H.: Structural translation from Time Petri Nets to Timed Automata. In: *Journal of Systems and Software* 79 (2006), Nr. 10, S. 1456–1468

[Cheffers und Whalen 2010] Cheffers, Mark ; Whalen, Don: *Audit Fees and Non-Audit Fees – A Seven-Year Trend.* 2010. – Abrufbar unter `http://www.complianceweek.com/s/documents/AAFeesDrft.pdf` (September 2010)

[Chow u. a. 2009] Chow, Richard ; Golle, Philippe ; Jakobsson, Markus ; Shi, Elaine ; Staddon, Jessica ; Masuoka, Ryusuke ; Molina, Jesus: Controlling Data in the Cloud: Outsourcing Computation without outsourcing Control. In: *Workshop on Cloud Computing Security*, ACM, 2009, S. 85–90

[Clarkson und Schneider 2008] Clarkson, Michael R. ; Schneider, Fred B.: Hyperproperties. In: *Proceedings of the 21st Computer Security Foundations Symposium*, IEEE Computer Society, 2008, S. 51–65

[Common Criteria 2009] *Common Criteria for Information Technology Security Evaluation Version 3.1.* 2009. – Abrufbar unter `http://www.commoncriteriaportal.org/cc/` (Februar 2011)

[Cummins 2010] Cummins, Fred A.: BPM Meets SOA. In: Brocke, Jan v. (Hrsg.) ; Rosemann, Michael (Hrsg.): *Handbook on Business Process Management 1: Introduction, Methods, and Information Systems.* Springer-Verlag, 2010 (International Handbooks on Information Systems)

[Davenport 1992] Davenport, Thomas H.: *Process Innovation: Reengineering Work Through Information Technology.* Harvard Business Press, 1992

[Davenport und Short 1990] Davenport, Thomas H. ; Short, James E.: The New Industrial Engineering: Information Technology and Business Process Redesign. In: *MIT Sloan Management Review* 31 (1990)

[Decker und Barros 2008] Decker, Gero ; Barros, Alistair: Interaction Modeling Using BPMN . In: Hofstede, Arthur ter (Hrsg.) ; Benatallah, Boualem (Hrsg.) ; Paik, Hye-Young (Hrsg.): *Business Process Management Workshops* Bd. 4928, Springer-Verlag, 2008, S. 208–219

[Denning 1976] Denning, Dorothy E.: A lattice model of secure information flow. In: *Communications of the ACM* 19 (1976), S. 236–243

[Denning und Denning 1977] Denning, Dorothy E. ; Denning, Peter J.: Certification of Programs for Secure Information Flow. In: *Communications of the ACM* 20 (1977), Nr. 7, S. 504–513

[Deutscher Corporate Governance-Kodex 2010] *Deutscher Corporate Governance-Kodex (in der Fassung vom 26. Mai 2010).* 2010. – Abrufbar unter `http://www.corporate-governance-code.de/ger/kodex/index.html` (Juli 2010)

[Dibrell und Miller 2002] Dibrell, C. C. ; Miller, Thomas R.: Organization design: The continuing influence of information technology. In: *Management Decision* 40 (2002), S. 620–627

[Dijkman u. a. 2007] Dijkman, Remco M. ; Dumas, Marlon ; Ouyang, Chun: Formal Semantics and Analysis of BPMN Process Models / Queensland University of Technology. 2007 (7115). – Forschungsbericht

[Dijkman u. a. 2008] Dijkman, Remco M. ; Dumas, Marlon ; Ouyang, Chun: Semantics and analysis of business process models in BPMN. In: *Information and Software Technology* 50 (2008), Nr. 12, S. 1281–1294

[ENISA 2010] Security & Resilience in Governmental Clouds / European Network and Information Security Agency. 2010. – Forschungsbericht. Abrufbar unter `http://www.enisa.europa.eu/act/rm/emerging-and-future-risk/deliverables/` (Mai 2011)

[Etro 2009] Etro, Federico: The Economic Impact of Cloud Computing on Business Creation, Employment and Output in Europe. In: *Review of Business and Economics* 54 (2009), Nr. 2, S. 179–208

[Ferraiolo und Kuhn 1992] Ferraiolo, David F. ; Kuhn, D. R.: Role-Based Access Controls. In: *15th National Computer Security Conference*, 1992

[Ferrara 2004] Ferrara, Andrea: Web services: A process algebra approach. In: Aiello, Marco (Hrsg.) ; Aoyama, Mikio (Hrsg.) ; Curbera, Francisco (Hrsg.) ; Papazoglou, Mike P. (Hrsg.): *International Conference on Service-Oriented Computing*, ACM, 2004, S. 242–251

[Focardi und Gorrieri 2001] Focardi, Riccardo ; Gorrieri, Roberto: Classification of Security Properties (Part I: Information Flow). In: Focardi, Riccardo (Hrsg.) ; Gorrieri, Roberto (Hrsg.): *Foundations of Security Analysis and Design* Bd. 2171. Springer-Verlag, 2001, S. 331–396

[Foster u. a. 2008] Foster, I. ; Zhao, Yong ; Raicu, I. ; Lu, S.: Cloud Computing and Grid Computing 360-Degree Compared. In: *Grid Computing Environments Workshop*, IEEE, 2008, S. 1 –10

[Fox 2003] Fox, Loren: *Enron: The Rise and Fall.* Wiley, 2003

[Gartner 2010] Gartner, Inc.: *Gartner Identifies the Top 10 Strategic Technologies for 2011.* Oktober 2010. – Abrufbar unter `http://www.gartner.com/it/page.jsp?id=1454221` (Feburar 2011)

[Génova 2010] Génova, Gonzalo: Is computer science truly scientific? In: *Communications of the ACM* 53 (2010), Nr. 7, S. 37–39

[Goguen und Meseguer 1982] Goguen, Joseph A. ; Meseguer, José: Security Policies and Security Models. In: *Symposium on Security and Privacy*, IEEE, 1982, S. 11–20

[Gollmann 2001] Gollmann, Dieter: Authentication – Myths and Misconceptions. In: Lam, Kwok-Yan (Hrsg.) ; Shparlinski, Igor (Hrsg.) ; Wang, Huaxiong (Hrsg.) ; Xing, Chaoping (Hrsg.): *Cryptography and Computational Number Theory* Bd. 20. Birkhäuser, 2001, S. 203–225

[Hammer 1996] Hammer, Michael: *Beyond Reengineering: How the Process-Centered Organization Is Changing Our Work and Our Lives.* HarperBusiness, 1996

[Hammer 2010] Hammer, Michael: What is Business Process Management? In: Brocke, Jan v. (Hrsg.) ; Rosemann, Michael (Hrsg.): *Handbook on Business Process Management 1: Introduction, Methods, and Information Systems.* Springer-Verlag, 2010 (International Handbooks on Information Systems), S. 3–16

[Hammer und Champy 1993] Hammer, Michael ; Champy, James: *Reengineering the Corporation: A Manifesto for Business Revolution.* 3rd edition. HarperBusiness, 1993

[Harmon 2010] Harmon, Paul: The Scope and Evolution of Business Process Management. In: Brocke, Jan v. (Hrsg.) ; Rosemann, Michael (Hrsg.): *Handbook on Business Process Management 1: Introduction, Methods, and Information Systems.* Springer-Verlag, 2010 (International Handbooks on Information Systems), S. 37–81

[Hartman 2006] Hartman, Thomas E.: The Cost of Being Public in the Era of Sarbanes-Oxley / Foley & Lardner LLP. 2006. – Forschungsbericht

[Höhn 2010] Höhn, Sebastian: *Sicherheit und Gesellschaft. Freiburger Studien des Centre for Security and Society.* Bd. 3: *Prozess-Rewriting zur Durchsetzung von Compliance in flexiblen Geschäftsprozessen: Ein Framework zur automatisierten Durchsetzung von Compliance-Regeln in Geschäftsprozessen.* Nomos, 2010

[Hinz u. a. 2005] Hinz, Sebastian ; Schmidt, Karsten ; Stahl, Christian: Transforming BPEL to Petri Nets. In: *Business Process Management* Bd. 3649, Springer-Verlag, 2005, S. 220–235

[Hoare 1978] Hoare, Charles Antony R.: Communicating sequential processes. In: *Communications of the ACM* 21 (1978), Nr. 8, S. 666–677

[Howard 2010] Howard, Chris: The Lazarus Effect: SOA returns / Burton Group. November 2010. – Forschungsbericht

[ISACA 2007] Information Systems Audit and Control Association (ISACA) (Veranst.): *Control Objectives for Information and related Technology (COBIT) – Framework for IT Governance and Control, Version 4.1.* 2007. – Abrufbar unter `http://www.isaca.org/Knowledge-Center/COBIT/` (Oktober 2010)

[Jansen und Grance 2011] Jansen, Wayne ; Grance, Timothy: Guidelines on Security and Privacy in Public Cloud Computing / National Institute of Standards and Technology (NIST). 2011 (800-144). – Forschungsbericht

[Jensen 1996] Jensen, Kurt: *Coloured Petri Nets.* 2nd edition. Springer-Verlag, 1996 (Monographs in Theoretical Computer Science)

[Jin u. a. 2010] Jin, Hai ; Ibrahim, Shadi ; Bell, Tim ; Gao, Wei ; Huang, Dachuan ; Wu, Song: Cloud Types and Services. In: Furht, Borko (Hrsg.) ; Escalante, Armando (Hrsg.): *Handbook of Cloud Computing.* Springer-Verlag, 2010, S. 335–355

[Khan 2004] Khan, Rashid N.: *Business Process Management: A Practical Guide.* Meghan-Kiffer Press, 2004

[Kähmer 2009] Kähmer, Martin: *ExPDT – Vergleichbarkeit von Richtlinien für Selbstregulierung und Selbstdatenschutz.* Vieweg+Teubner, 2009

[Klempt u. a. 2009] Klempt, Philipp ; Schmidpeter, Hannes ; Sowa, Sebastian ; Tsinas, Lampros: Business Oriented Information Security Management – A Layered Approach. In: Meersman, Robert (Hrsg.) ; Tari, Zahir (Hrsg.): *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS* Bd. 4804. Springer-Verlag, 2009, S. 1835–1852

[Küngas 2005] Küngas, Peep: Petri Net Reachability Checking Is Polynomial with Optimal Abstraction Hierarchies. In: Zucker, Jean-Daniel (Hrsg.) ; Saitta, Lorenza (Hrsg.): *Abstraction, Reformulation and Approximation* Bd. 3607, Springer-Verlag, 2005, S. 149–164

[Knorr 2001] Knorr, Konstantin: Multilevel Security and Information Flow in Petri Net Workflows. In: *International Conference on Telecommunication Systems - Modeling and Analysis*, 2001, S. 9–20

[Ko 2009] Ko, Ryan K. L.: A computer scientist's introductory guide to business process management (BPM). In: *ACM Crossroads* 15 (2009), Nr. 4, S. 11–18

[Koshutanski und Massacci 2003] Koshutanski, Hristo ; Massacci, Fabio: An access control framework for business processes for web services. In: Jajodia, Sushil (Hrsg.) ; Kudo, Michiharu (Hrsg.): *ACM Workshop on XML Security*, 2003, S. 15–24

[Lampson 1973] Lampson, Butler W.: A Note on the Confinement Problem. In: *Communications of the ACM* 16 (1973), Nr. 10, S. 613–615

[Lassen und van der Aalst 2009] Lassen, Kristian B. ; Aalst, Willibrordus Martinus P. van der: Complexity metrics for Workflow nets. In: *Information and Software Technology* 51 (2009), Nr. 3, S. 610–626

[Leymann 2001] Leymann, Frank: *WSFL – Web Services Flow Language*. IBM Software Group, Whitepaper. 2001

[Leymann u. a. 2002] Leymann, Frank ; Roller, Dieter ; Schmidt, Marc-Thomas: Web Services and Business Process Management. In: *IBM Systems Journal* 41 (2002), Nr. 2, S. 198–211

[Liebenau und Kärrberg 2006] Liebenau, Jonathan ; Kärrberg, Patrik: International Perspectives on Information Security Practices / The London School of Economics and Political Science. 2006. – Forschungsbericht

[Lindsay u. a. 2003] Lindsay, Ann ; Downs, Denise ; Lunn, Ken: Business Processes – Attempts to find a Definition . In: *Information and Software Technology* 45 (2003), Nr. 15

[Llorens u. a. 2008] Llorens, Marisa ; Oliver, Javier ; Silva, Josep ; Tamarit, Salvador ; Vidal, Germán: Dynamic Slicing Techniques for Petri Nets. In: *Electronic Notes in Theoretical Computer Science* 223 (2008), S. 153–165

[Lohmann u. a. 2009] Lohmann, Niels ; Verbeek, Eric ; Ouyang, Chun ; Stahl, Christian: Comparing and evaluating Petri net semantics for BPEL. In: *International Journal of Business Process Integration and Management* 4 (2009), Nr. 1, S. 60–73

[Lucchia und Mazzara 2007] Lucchia, Roberto ; Mazzara, Manuel: A pi-calculus based semantics for WS-BPEL. In: *Journal of Logic and Algebraic Programming* 70 (2007), Nr. 1, S. 96–118

[Malik 2003] Malik, Om: *Broadbandits: Inside the $750 Billion Telecom Heist*. Wiley, 2003

[Mantel 2005] Mantel, Heiko: The framework of selective interleaving functions and the modular assembly kit. In: Atluri, Vijayalakshmi (Hrsg.) ; Samarati, Pierangela (Hrsg.) ; Küsters, Ralf (Hrsg.) ; Mitchell, John C. (Hrsg.): *ACM workshop on formal methods in security engineering*, ACM, 2005, S. 53–62

[Marsan 1990] Marsan, M.: Stochastic Petri nets: An elementary introduction. In: Rozenberg, Grzegorz (Hrsg.): *Advances in Petri Nets* Bd. 424, Springer-Verlag, 1990, S. 1–29

[Mather u. a. 2009] Mather, Tim ; Kumaraswamy, Subra ; Latif, Shahed: *Cloud Security and Privacy – An Enterprise Perspective on Risks and Compliance*. O'Reilly Media, 2009

[McHugh 1995] McHugh, John: *Handbook for the Computer Security Certification of Trusted Systems*. Kap. 8, Naval Research Laboratory, 1995

[McLean 1990] McLean, John: Security Models and Information Flow. In: *Symposium on Security and Privacy*, IEEE, 1990, S. 180–187

[Meda u. a. 2010] Meda, Hema S. ; Sen, Anup K. ; Bagchi, Amitava: On Detecting Data Flow Errors in Workflows. In: *Journal of Data and Information Quality* 2 (2010), Nr. 1

[Mikolajczak und Gami 2008] Mikolajczak, Boleslaw ; Gami, Nirmal: Design and Verification of Loosely Coupled Inter-Organizational Workflows with Multi-Level Security. In: *Journal of Computers* 3 (2008), Nr. 1, S. 63–78

[Milner 1980] Milner, Robin: *Lecture Notes in Computer Science*. Bd. 92: *A Calculus of Communicating Systems*. Springer-Verlag, 1980

[Milner u. a. 1992] Milner, Robin ; Parrow, Joachim ; Walker, David: A calculus of mobile processes. In: *Information and Computation* 100 (1992), Nr. 1, S. 1–40

[Moses 2005] Moses, Tim: *eXtensible Access Control Markup Language (XACML) Version 2.0*. 2005. – Abrufbar unter `http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf` (Januar 2011)

[zur Muehlen 2004] Muehlen, Michael zur: *Workflow-based Process Controlling – Foundation, Design, and Application of Workflow-driven Process Information Systems*. Logos Verlag, 2004

[OASIS 2010] *OASIS Web Services Business Process Execution Language Technical Committee Website*. 2010. – `http://www.oasis-open.org/committees/wsbpel` (November 2010)

[OGC 2007] Office of Government Commerce (OGC) (Veranst.): *Information Technology Infrastructure Library (ITIL), Version 3*. 2007. – Abrufbar unter `http://www.itil-officialsite.com` (Oktober 2010)

[OMG 2009] *Business Process Model and Notation (BPMN) Version 1.2*. 2009. – Abrufbar unter `http://www.omg.org/spec/BPMN/1.2` (November 2010)

[Ouyang u. a. 2006] Ouyang, Chun ; Dumas, Marlon ; Hofstede, Arthur H. M. ter ; Aalst, Wil M. P. van der: From BPMN Process Models to BPEL Web Services. In: *International Conference on Web Services*, IEEE, 2006, S. 285–292

[Paci u. a. 2008a] Paci, Federica ; Bertino, Elisa ; Crampton, Jason: An Access-Control Framework for WS-BPEL. In: *International Journal of Web Services Research* 5 (2008), Nr. 3, S. 20–43

[Paci u. a. 2008b] Paci, Federica ; Ferrini, Rodolfo ; Sun, Yuqing ; Bertino, Elisa: Authorization and User Failure Resiliency for WS-BPEL Business Processes. In: Bouguettaya, Athman (Hrsg.) ; Krueger, Ingolf (Hrsg.) ; Margaria, Tiziana (Hrsg.): *International Conference on Service-Oriented Computing* Bd. 5364, Springer-Verlag, 2008, S. 116–131

[Pascalau u. a. 2009] Pascalau, Emilian ; Giurca, Adrian ; Wagner, Gerd: Validating Auction Business Processes using Agent-based Simulations. In: Abramowicz, Witold (Hrsg.) ; Maciaszek, Leszek A. (Hrsg.) ; Kowalczyk, Ryszard (Hrsg.) ; Speck, Andreas (Hrsg.): *Business Process, Services Computing and Intelligent Service Management* Bd. 147, Köllen-Verlag, 2009, S. 95–109

[PCI DSS 2010] *Payment Card Industry (PCI) Datensicherheitsstandard – Anforderungen und Sicherheitsbeurteilungsverfahren Version 2.0.* 2010. – Abrufbar unter `https://www.pcisecuritystandards.org` (Februar 2011)

[Pesic und van der Aalst 2005] Pesic, Maja ; Aalst, Willibrordus Martinus P. van der: Modeling Work Distribution Mechanisms Using Colored Petri Nets / Beta Research School for Operations Management and Logistics. 2005 (146). – Forschungsbericht

[Petri 1962] Petri, Carl A.: *Kommunikation mit Automaten*, Institut für Instrumentelle Mathematik, Universität Bonn, Dissertation, 1962

[Pfitzner u. a. 2009] Pfitzner, Kerstin ; Decker, Gero ; Kopp, Oliver ; Leymann, Frank: Web Service Choreography Configurations for BPMN . In: Nitto, Elisabetta D. (Hrsg.) ; Ripeanu, Matei (Hrsg.): *Service-Oriented Computing – ICSOC 2007 Workshops* Bd. 4907, Springer-Verlag, 2009

[Pfohl u. a. 2001] Pfohl, Hans-Christian ; Elbert, Ralf ; Hofmann, Erik: Strategische Bedeutung von Geschäftsprozessen. In: *Zeitschrift für Unternehmensentwicklung und Industrial Engineering* 50 (2001), Nr. 5, S. 196–200

[Porter 1985] Porter, Michael E.: *Competitive Advantage: Creating and Sustaining Superior Performance.* Free Press, 1985

[Raghupathi 2007] Raghupathi, Wullianallur: Corporate Governance of IT: A Framework for Development. In: *Communications of the ACM* 50 (2007), Nr. 8, S. 94–99

[Recker und Mendling 2006] Recker, Jan ; Mendling, Jan: On the Translation between BPMN and BPEL: Conceptual Mismatch between Process Modeling Languages. In: Latour, Thibaud (Hrsg.) ; Petit, Michael (Hrsg.): *International Workshop on Exploring Modeling Methods in Systems Analysis and Design*, 2006, S. 521–532

[Reichert und Dadam 1998] Reichert, Manfred ; Dadam, Peter: Adept$_{flex}$ – Supporting Dynamic Changes of Workflows Without Losing Control. In: *Journal of Intelligent Information Systems* 10 (1998), Nr. 2, S. 93–129

[Reisig 1985] Reisig, Wolfgang: Petri nets with individual tokens. In: *Theoretical Computer Science* 41 (1985), S. 185–213

[Rifkin 1997] Rifkin, Jeremy: Die dritte Säule der neuen Gesellschaft. In: *Die Zeit* (1997), 2. Mai, Nr. 19, S. 32

[Ristenpart u. a. 2009] Ristenpart, Thomas ; Tromer, Eran ; Shacham, Hovav ; Savage, Stefan: Hey, you, get off of my Cloud: Exploring Information Leakage in Third-Party Compute Clouds. In: *16th ACM Conference on Computer and Communications Security*, ACM, 2009, S. 199–212

[Robey 1981] Robey, Daniel: Computer Information Systems and Organization Structure. In: *Communications of the ACM* 24 (1981), Nr. 10, S. 679–687

[Roscoe 1996] Roscoe, Andrew W.: Intensional Specifications of Security Protocols. In: *Workshop on Computer Security Foundations*, IEEE, 1996, S. 28–38

[Russell u. a. 2006] Russell, Nick ; Hofstede, Arthur H. ter ; Aalst, Willibrordus Martinus P. van der ; Mulyar, Nataliya: Workflow Control-Flow Patterns: A Revised View / BPM Center. 2006 (BPM-06-22). – Forschungsbericht. Abrufbar unter `http://wwwis.win.tue.nl/ wvdaalst/BPMcenter/reports/2006/BPM-06-22.pdf` (Januar 2011)

[Ryan 2001] Ryan, Peter: Mathematical Models of Computer Security. In: Focardi, Riccardo (Hrsg.) ; Gorrieri, Roberto (Hrsg.): *Foundations of Security Analysis and Design* Bd. 2171. Springer-Verlag, 2001, S. 1–62

[Ryan u. a. 2001] Ryan, Peter ; McLean, John ; Millen, Jon ; Gligor, Virgil: Noninterference, who needs it? In: *IEEE Computer Security Foundations Workshop*, 2001, S. 237–238

[Sabelfeld und Myers 2003] Sabelfeld, Andrei ; Myers, Andrew C.: Language-based information-flow security. In: *IEEE Journal on Selected Areas in Communications* 21 (2003), Nr. 1, S. 5–19

[Sabelfeld und Sands 2009] Sabelfeld, Andrei ; Sands, David: Declassification: Dimensions and principles. In: *Journal of Computer Security* 17 (2009), Nr. 5, S. 517–548

[Sackmann 2008] Sackmann, Stefan: Assessing the effects of IT changes on IT risk – A business process-oriented view. In: Bichler, Martin (Hrsg.) ; Hess, Thomas (Hrsg.) ; Krcmar, Helmut (Hrsg.) ; Lechner, Ulrike (Hrsg.) ; Matthes, Florian (Hrsg.) ; Picot, Arnold (Hrsg.) ; Speitkamp, Benjamin (Hrsg.) ; Wolf, Petra (Hrsg.): *Multikonferenz Wirtschaftsinformatik*, GITO-Verlag, Berlin, 2008

[Sackmann und Kähmer 2008] Sackmann, Stefan ; Kähmer, Martin: ExPDT: Ein Policy-basierter Ansatz zur Automatisierung von Compliance. In: *WIRTSCHAFTSINFORMATIK* 50 (2008), S. 366–374

[Sackmann und Strüker 2005] Sackmann, Stefan ; Strüker, Jens: *Electronic Commerce Enquête 2005: 10 Jahre Electronic Commerce – Eine stille Revolution in deutschen Unternehmen.* Konradin IT-Verlag, 2005

[Sadiq u. a. 2004] Sadiq, Shazia W. ; Orlowska, Maria E. ; Sadiq, Wasim ; Foulger, Cameron: Data Flow and Validation in Workflow Modelling. In: Schewe, Klaus-Dieter (Hrsg.) ; Williams, Hugh E. (Hrsg.): *Fifteenth Australasian Database Conference*, Australian Computer Society, 2004, S. 207–214

[Salaün u. a. 2004] Salaün, Gwen ; Bordeaux, Lucas ; Schaerf, Marco: Describing and reasoning on Web services using process algebra. In: *International Conference on Web Services*, IEEE, 2004, S. 43–50

[Saltzer u. a. 1984] Saltzer, J. H. ; Reed, D. P. ; Clark, D. D.: End-to-end arguments in system design. In: *ACM Transactions on Computer Systems* 2 (1984), S. 277–288

[Sandhu und Samarati 1994] Sandhu, Ravi S. ; Samarati, Pierangela: Access Control: Principle and Practice. In: *IEEE Communications Magazine* 32 (1994), Nr. 9, S. 40–48

[Sarbanes-Oxley Act 2002] *Sarbanes-Oxley Act of 2002.* 2002. – Abrufbar unter `http://www.gpo.gov/fdsys/pkg/PLAW-107publ204` (Juli 2010)

[Scheer 1978] Scheer, August-Wilhelm: *Wirtschafts- und Betriebsinformatik.* Verlag Moderne Industrie, 1978 (mi-Studienbibliothek Betriebswirtschaft)

[Scheer 1980] Scheer, August-Wilhelm: Elektronische Datenverarbeitung und Operations Research im Produktionsbereich – Zum gegenwärtigen Stand von Forschung und Anwendung. In: *OR Spectrum* 2 (1980), S. 1–22

[Schneider 2000] Schneider, Fred B.: Enforceable Security Policies. In: *Transactions on Information and System Security* 3 (2000), Nr. 1, S. 30–50

[Smith 2008] Smith, Adam: *An Inquiry into the Nature and Causes of the Wealth of Nations.* Oxford University Press, 2008

[Sun u. a. 2006] Sun, Sherry X. ; Zhao, J. L. ; Nunamaker, Jay F. ; Sheng, Olivia R. L.: Formulating the Data-Flow Perspective for Business Process Management. In: *Information Systems Research* 17 (2006), Nr. 4, S. 374–391

[Tchifilionova 2011] Tchifilionova, Vassilka: Security and Privacy Implications of Cloud Computing – Lost in the Cloud. In: Camenisch, Jan (Hrsg.) ; Kisimov, Valentin (Hrsg.) ; Dubovitskaya, Maria (Hrsg.): *Open Research Problems in Network Security* Bd. 6555, Springer-Verlag, 2011, S. 149–158

[TCSEC 1985] United States Department of Defense: *Trusted Computer System Evaluation Criteria.* 1985. – Abrufbar unter `http://csrc.nist.gov/publications/history/dod85.pdf` (Februar 2011)

[Thatte 2001] Thatte, Satish: *XLANG – Web Services for Business Process Design.* Microsoft Corporation, Initial Public Draft. 2001

[Trčka u. a. 2009] Trčka, Nikola ; Aalst, Willibrordus Martinus P. van der ; Sidorova, Natalia: Data-Flow Anti-patterns: Discovering Data-Flow Errors in Workflows. In: Eck, Pascal van (Hrsg.) ; Gordijn, Jaap (Hrsg.) ; Wieringa, Roel (Hrsg.): *International Conference on Advanced Information Systems* Bd. 5565, 2009, S. 425–439

[Varian 2007] Varian, Hal R.: Position auctions. In: *International Journal of Industrial Organization* 25 (2007), Nr. 6, S. 1163–1178

[Vetter 2008] Vetter, Eberhard: Compliance in der Unternehmerpraxis. In: *Compliance in der Unternehmerpraxis*, Gabler, 2008, S. 29–42

[Wang und Li 2007] Wang, Qihua ; Li, Ninghui: Satisfiability and Resiliency in Workflow Systems. In: Biskup, Joachim (Hrsg.) ; López, Javier (Hrsg.): *12th European Symposium On Research In Computer Security* Bd. 4734, Springer-Verlag, 2007

[Weidlich u. a. 2008] Weidlich, Matthias ; Decker, Gero ; Grosskopf, Alexander ; Weske, Mathias: BPEL to BPMN: The Myth of a Straight-Forward Mapping . In: Meersman, Robert (Hrsg.) ; Tari, Zahir (Hrsg.): *On the Move to meaningful Internet Systems* Bd. 5331, Springer-Verlag, 2008, S. 265–282

[Wolf und Harmon 2010] Wolf, Celia ; Harmon, Paul: *The State of Business Process Management.* 2010. – Abrufbar unter `www.bptrends.com` (Oktober 2010)

[Wolter u. a. 2009a] Wolter, Christian ; Miseldine, Philip ; Meinel, Christoph: Verification of Business Process Entailment Constraints Using SPIN. In: *First International Symposium on Engineering Secure Software and Systems* Bd. 5429, Springer-Verlag, 2009, S. 1–15

[Wolter und Schaad 2007] Wolter, Christian ; Schaad, Andreas: Modeling of Task-Based Authorization Constraints in BPMN. In: Alonso, Gustavo (Hrsg.) ; Dadam, Peter (Hrsg.) ; Rosemann, Michael (Hrsg.): *5th International Conference on Business Process Management* Bd. 4714, Springer-Verlag, 2007

[Wolter u. a. 2009b] Wolter, Christian ; Weiss, Christian ; Meinel, Christoph: An XACML Extension for Business Process-Centric Access Control Policies. In: *IEEE International Symposium on Policies for Distributed Systems and Networks*, IEEE Computer Society, 2009, S. 166–169

[Wong und Gibbons 2008] Wong, Peter Y. H. ; Gibbons, Jeremy: A Process Semantics for BPMN. In: Shaoying Liu, Tom M. (Hrsg.) ; Araki, Keijiro (Hrsg.): *Formal Methods and Software Engineering* Bd. 5256, Springer-Verlag, 2008, S. 355–374

[Woods und Mattern 2006] Woods, Dan ; Mattern, Thomas: *Enterprise SOA: Designing IT for Business Innovation.* O'Reilly Media, 2006

[Xiangpeng u. a. 2006] Xiangpeng, Zhao ; Cerone, Antonio ; Krishnan, Padmanabhan: Verifying BPEL Workflows Under Authorisation Constraints. In: Dustdar, Schahram (Hrsg.) ; Fiadeiro, José L. (Hrsg.) ; Sheth, Amit (Hrsg.): *4th International Conference on Business Process Management* Bd. 4102, Springer-Verlag, 2006, S. 439–444

[Zakinthinos und Lee 1997] Zakinthinos, Aris ; Lee, E. S.: A General Theory of Security Properties. In: *IEEE Symposium on Security and Privacy*, 1997, S. 94–102

[Zdancewic 2002] Zdancewic, Stephan A.: *Programming Languages for Information Security*, Cornell University, Dissertation, 2002

[Zdancewic 2004] Zdancewic, Stephan A.: Challenges for Information-Flow Security. In: *First International Workshop on Programming Language Interference and Dependence*, 2004. – Abrufbar unter `http://profs.sci.univr.it/~mastroen/noninterference.html` (Mai 2011)