Westfälische
Wilhelms-Universität
Münster

Diplomarbeit

Ein generischer, beobachtungsbasierter Noninterferenz-Begriff

Thema gestellt von
Prof. Dr. Müller-Olm

vorgelegt von

Ari Rasch
6. September 2013

Arbeitsgruppe Softwareentwicklung und Verifikation
Fachbereich 10 - Mathematik und Informatik
Westfälische Wilhelms-Universität Münster

Inhaltsverzeichnis

KAPITEL 1

Einführung

"Informations- und Kommunikationstechnologie (IKT) ist heute nahezu in allen Bereichen von zentraler Bedeutung. Eingebettete Systeme, Machine-to-Machine Kommunikation, Vernetzung, aber auch on-demand beziehbare Mehrwertdienste aus der Cloud sind zentrale Wachstumstreiber einer immer stärker digitalisierten Wirtschaft. Der IT-Sicherheit kommt hierbei eine Schlüsselrolle zu.
IT-Sicherheit hat die Aufgabe, Unternehmen und deren Werte, hierzu gehören beispielsweise (Know-How, Kundendaten, Personaldaten) zu schützen und wirtschaftliche Schäden, die durch Vertraulichkeitsverletzungen, Manipulation oder auch Störungen der Verfügbarkeit von Diensten des Unternehmens entstehen können, zu verhindern. Da eine vollständige Vermeidung oder Verhinderung von Angriffen in der Praxis nicht möglich ist, umfasst das Gebiet der IT-Sicherheit insbesondere auch Maßnahmen und Konzepte, um das Ausmaß potentieller Schäden, die durch Sicherheitsvorfälle entstehen könne, zu reduzieren und damit die Risiken beim Einsatz von IKT-Systemen zu verringern. Schwachstellen und konkrete Angriffe oder Angriffsversuche auf IKT-Systeme müssen frühzeitig und mit möglichst hoher Präzision erkannt werden können und auf eingetretene Schadensfälle muss mit geeigneten technischen Maßnahmen reagiert werden. Techniken der Angriffserkennung und -Reaktion gehören deshalb ebenso zur IT-Sicherheit, wie methodische Grundlagen, IKT-Systeme so zu entwickeln, dass sie qua Design ein hohes Maß an Sicherheit bieten. Man spricht in diesem Zusammenhang auch oft von "Secure-by-Design". Gleichzeitig dient IT-Sicherheit als Enabling-Technologie. Sie ermöglicht die Entwicklung neuer, Vertrauenswürdiger Anwendungen und Dienstleistungen verbunden mit innovativen Geschäftsmodellen beispielsweise im Gesundheitsbereich, bei Automotive-Anwendungen oder aber auch in zukünftigen, intelligenten Umgebungen, wie Smart Grid, Smart Factory, Smart Health oder auch Smart Cities.
Zudem dienen die Technologien und Verfahren der IT-Sicherheit dazu, die Sicherheit in einem allgemeineren Sinn (u.a. im Sinne öffentlicher Sicherheit) zu erhöhen. Zur Überwachung von beispielsweise kritischen Infrastrukturen, wie Kraftwerken, Produktionsanlagen, Transportleitsysteme etc. wird bereits heute eine Vielzahl von Sensoren eingesetzt, die

kontinuierlich Umgebungsdaten erfassen, Daten austauschen und die das Ablaufverhalten bzw. den operativen Betrieb komplexer Anlagen permanent überwachen. Probleme und auffälliges Verhalten werden an zumeist zentrale Leitstellen gemeldet, so dass auf der Basis dieser Steuerungsdaten kontrollierend in kritische Abläufe (zum Teil vollständig automatisiert) eingegriffen wird. Es ist unmittelbar klar, dass die zur Steuerung und Kontrolle verwendeten Daten vor Manipulationen zu schützen sind. Sie müssen zudem rechtzeitig und vollständig vorliegen. Vielfach ist es auch aus Gründen des Datenschützers zudem notwendig, die Daten vertraulich zu verarbeiten, man denke nur an die aktuelle Diskussion zu den digital erfassten Daten und per Datenkommunikation übermittelten Stromverbrauchsdaten privater Haushalte. IT-Sicherheitskonzepte sind die wichtige Basis, damit die zur Überwachung und Steuerung der Anlagen eingesetzten IKT-Komponenten auch verlässlich für sicherheitskritische Anwendungen eingesetzt werden können."[E11]

Ziele dieser Arbeit

Ziel dieser Arbeit stellt eine allgemeingültige Definition des IT-Sicherheitskonzepts der "Noninterferenz" dar. Noninterferenz ist eine Sicherheit-Spezifikation welche erstmalig von Goguen und Meseguer 1982 in [GM82] definiert wurde und sich über die Jahre großer Beliebtheit erfreuen durfte. Diese beschreibt dabei eine Form von Datenflusskontrolle, welche Sicherheit in Bezug auf Angreifer in der Rolle von Beobachtern garantieren soll. Unter Beobachtern verstehen wir dabei eine passive, das heißt nicht in das System eingreifende Form von Entitäten, welche Programmausführungen über die öffentliche Schnittstelle der ausführenden Maschine beobachten, mit dem Ziel anhand ihrer Beobachtungen Rückschlüsse auf vertrauliche Daten zu treffen. Wie solche Rückschlüsse dabei aussehen können wollen wir anhand einiger Standardbeispiele verschiedener Problemklassen verdeutlichen. Hierfür betrachten wir verschiedene minimal unsicherer Programme (in Pseudocode). Dabei legen wir einen bi-partionierten Zustandsraum zugrunde, aufgeteilt in eine L-Partition, repräsentiert durch eine Variable l, für die Kapselung öffentlich Daten, sowie einer einer H Partition, repräsentiert durch eine Variable h, für die Kapselung vertraulicher Daten, auf welche anhand von l keine Rückschlüsse möglich sein sollen.

Direkt: $[l := h]$
: Anhand von l kann direkt auf h geschlossen werden

Indirekt: $[if(h == 0)\ then\ \{l := 0\}\ else\ \{l := 1\}$
: Anhand von l kann darauf geschlossen werden, ob h den Wert 0 hat oder nicht

Terminationsverhalten: $[if\ (h == 0)\ then\ \{\ while(1 == 1)\ \{skip\}\ \}\ else\ \{skip\}\}$
: Anhand der Terminierung des Programms kann darauf geschlossen werden, ob h den Wert 0 hat oder nicht

Stochastisch: $[l := h \mid l := 0 \mid l := 1]$
wobei "|" für die Abgrenzung konkurrierender Threads steht.
Nehmen wir an h und l sind boolesch und jeder Thread wird mit Wahrscheinlichkeit $\frac{1}{3}$ ausgeführt, dann steht in l final mit einer Wahrscheinlichkeit von $\frac{2}{3}$ der Wert von h

Timing: $[if\ (h == 0)\ then\ \{skip; skip\}\ else\ \{skip\}]$
Anhand der Programmlaufzeit (in Einzelschritten) kann darauf geschlossen werden, ob h den Wert 0 hat oder nicht:
3 Programmschritte $\Rightarrow$ $(h == 0)$,
2 Programmschritte $\Rightarrow$ $(h! = 0)$

Um Informationslecks obiger Formen auszuschließen verlangt Noninterferenz-Sicherheit in der Regel die Ununterscheidbarkeit einer entsprechenden Programmausführung bzgl. Programmzustände, welche hinsichtlich öffentlicher Daten übereinstimmen und somit lediglich in Bezug auf vertrauliche Daten differieren. Zu beachten ist hierbei, dass eine Definition von Ununterscheidbarkeit keineswegs kanonisch ist. Auch dies wollen wir uns anhand eines Beispiels verdeutlichen.
Hierfür betrachten man ein festes Programm p_0 sowie Programmzustände z und z', welche in l übereinstimmen, wir schreiben hierfür auch $z =_L z'$. Seien die entsprechenden Programmausführungen von folgender Form

$$\langle p,z\rangle \rightarrow \langle p_1,z_1\rangle \rightarrow \ldots \langle\langle\rangle,z_n\rangle$$
$$\langle p,z'\rangle \rightarrow \langle p'_1,z'_1\rangle \rightarrow \ldots \langle\langle\rangle,z'_m\rangle$$

Wir werden nun zwei verschiedene Ununterscheidbarkeits-Begriffe formulieren von denen wir anschließend feststellen werden, dass diese unterschiedliche Sicherheiten definieren und somit insbesondere Schutz vor unterschiedlichen Beobachter-Typen spezifizieren.

1. [*Terminale-Ununterscheidbarkeit*]

   Wir bezeichnen die obigen Programmausführungen als *terminal-ununterscheidbar* gdw.

   $$z_n =_L z'_m$$

2. [*Timing-Ununterscheidbarkeit*]

   Wir bezeichnen die obigen Programmausführungen als *timing-ununterscheidbar* gdw.

   $$n = m$$

Die Terminale-Ununterscheidbarkeit identifiziert obige Programmausführungen somit genau dann, wenn die Programmzustände der entsprechenden Endkonfigurationen in den öffentlichen Daten übereinstimmen und bietet somit Schutz vor Beobachtern, welche die Terminierung entsprechender Programme beobachten.

Die Timing-Ununterscheidbarkeit hingegen identifiziert obige Programmausführungen genau dann, wenn diese dieselbe Ausführungszeit benötigen und bietet somit Schutz vor Beobachtern welche die Ausführungszeiten von Programmausführungen messen.
Wir haben somit für zwei verschiedene Formen von Ununterscheidbarkeit feststellen können, dass diese sinnige Sicherheits-Begriffe definieren. Es ist somit offensichtlich, dass ein allgemeingültiger Noninterferenz-Begriff "Ununterscheidbarkeit" nicht fixieren sollte. Betrachte man nun erneut die obigen Beispiele unsicherer Informationsflüsse stellt man weiterhin fest, dass Noninterferenz für verschiedene System-Klassen definierbar ist. Beispielsweise beschreibt $\langle l := h \mid l := 0 \mid l := 1 \rangle$ das Programm eines parallelen und somit nicht-deterministischen Systems, wobei das Programm $\langle l := h \rangle$ evtl. einem sequentiellen System zugeordnet ist. Man erkennt hier zum einen, dass auch die entsprechenden Systeme in einer generischen Noninterferenz-Definition einen variablen Parameter darstellen sollten, zum anderen aber auch, dass Ununterscheidbarkeits-Begriffe für System-Klassen nahezu beliebiger Komplexität und insbesondere auch für nicht-deterministische Systeme formuliert werden müssen.
Unser primäres Ziel stellt nun die formale Erfassung des Konzepts der Noninterferenz dar, sodass eine intuitive, generische Definition von Noninterferenz und einer dadurch induzierten Sicherheit resultiert, in welche sich spezifische Konkretisierungen auf natürliche Weise einbetten lassen.

Weiteres Vorgehen

Die weitere Arbeit gliedert sich in acht weitere Kapitel, wobei Kapitel 2 lediglich eine knappe Wiederholung grundlegender mathematischer Begriffe darstellt, welche wir der Übersicht halber einmalig an zentraler Stelle bewerkstelligen wollen. Unser eigentlichen Überlegungen beginnen erst in Kapitel 3 mit der Fixierung und Analyse eines festen Maschinenmodells, auf welche die kommende Theorie einheitlich aufsetzten wird. Aufbauend werden wir in Kapitel 4 zunächst ermitteln, durch welche Teilstrukturen entsprechender Systemmodelle Programmausführungen formal repräsentiert werden können, um anschließend den Begriff eines Beobachters mathematisch sinnvoll erfassen zu können. In Kapitel 5 werden wir anhand der gewonnenen Erkenntnisse dazu im Stande sein einen generischen, beobachtungsbasierten Begriff von Noninterferenz definieren zu können, welchen wir dann im Laufe der Arbeit hinsichtlich verschiedener Gesichtspunkte weiter betrachten werden. Beispielsweise werden in Kapitel 6 verschiedene Ordnungsstrukturen definieren von denen wir anschließend feststellen werden, dass diese mit der Noninterferenz harmonieren. Uns wird es anschließend möglich sein Beobachter in Bezug auf ihre Angriffsstärke zu klassifizieren mit dem Wissen, dass Noninterferenz-Sicherheit bzgl. stärkerer Beobachter insbesondere auch bzgl. schwächerer Beobachter gilt. An dieser Stelle ist die eigentliche Theorie abgeschlossen und es beginnt die Validierung unseres Vorgehens. Hierfür werden

wir zunächst in Kapitel 7 zeigen, dass wir Beobachter durch typische Semantiken der wissenschaftlichen Literatur beschreiben können. Anschließend werden wir dies in Kapitel 8 nutzen um zwei bestehende Formen von Noninterferenz-Sicherheit mit unserem Ansatz zu formulieren. Wir betrachten hierfür zunächst eine simple, sequentielle Sprache. Anschließend übertragen wir unsere Überlegungen auf eine durch Parallelität verzweigten Sprache, bei welcher wir die intuitiven Vorstellungen der Autoren bzgl. Sicherheit sogar passender umsetzen können.

KAPITEL 2

Mathematische Grundlagen

Ziel dieses Kapitels stellt die Definition einer mathematischen Grundlage sowie die Wiederholung grundlegender mathematischer Begriffe dar, welches uns eine fundierte Basis für den weiteren Verlauf dieser Arbeit bieten soll. Da die angesprochenen Themen reine Wiederholung darstellen sollten, verzichten wir auf detaillierte Beispiele und Erläuterungen.

Logische Grundlagen

Als mathematisches Fundament werden wir die Neumann-Bernays-Gödel Klassentheorie NBG zugrunde legen. Diese kann als die übliche Axiomatisierung ZFC von Ernst Zermelo und Abraham Fraenkel betrachtet werden, erweitert um den Klassenbegriff. Klassen stellen dabei beliebig große Kollektionen von Mengen dar, welche in der üblichen Mengenlehre wie beispielsweise ZFC aufgrund einhergehender Antinomien undefiniert sind. Wir wollen auf die explizite Definition von NBG im Rahmen dieser Arbeit verzichten und verweisen für weitere Details auf [N27].
Auch werden uns Ordinal- und Kardinalzahlen zusammen mit ihrer natürlichen Ordnungsstruktur sowie der entsprechenden transfiniten Arithmetik in dieser Arbeit begegnen. Kardinalzahlen fassen wir dabei als besondere Ordinalzahlen auf. Ordinal- und Kardinalzahlen kleiner ω identifizieren wir mit natürlichen Zahlen. Für weitere Details verweisen wir auf [S09]

Notation 2.0.1:

Wir bezeichnen mit

- ord die echte Klasse aller Ordinalzahlen
- $card$ die echte Klasse aller Kardinalzahlen

sowie mit

- $ord_{\leq\alpha}$, für $\alpha \in ord$, die Menge aller Ordinalzahlen ι mit $\iota \leq \alpha$
- $kard_{<\alpha}$, für $\alpha \in ord$, die Menge aller Ordinalzahlen ι mit $\iota < \alpha$

Oft wird es nötig sein über Wertebereiche zu indizieren. In den meisten Fällen stellen diese Wertebereiche gerade die natürlichen Zahlen bzw. Anfangstücke dieser da. Wir werden diese im Folgenden einheitlich als Indexmengen bezeichnen.

Definition 2.0.2: [Indexmenge]

Als *Indexmenge* bezeichnen wir Elemente aus $ord_{\leq\omega}$.

Wir werden nun zum Abschluss dieses Abschnitts an die Verallgemeinerung der Potenzmenge auf Klassen erinnern. Man beachte dabei, dass Potenzklassen möglicherweise echte Klasse darstellen, Elemente von Potenzklassen jedoch stets Mengen sind. Dies ergibt sich anhand der Klassenbildung mit Hilfe des Komprehensionsschemas.

Definition 2.0.3: [Potenzklasse]

Sei K eine Klasse. Die Potenzklasse $\mathscr{P}(K)$ von K ist definiert als

$$\mathscr{P}(K) := \{M \mid M \subseteq K\}$$

Ist K eine Menge bezeichnen wir $\mathscr{P}(K)$ auch als *Potenzmenge*.

Folgen

In diesem Abschnitt soll die Theorie um den Begriff der “Folge“ wiederholt werden. Um endliche und unendlichen Folgen einheitlich handhaben zu können werden wir hierfür zunächst den Begriff der “Abbildung“ definieren, welche uns anschließend als Basis einer Folge dienen wird.

Definition 2.0.4: [Abbildung]

Seien A und B beliebige Mengen. Als Abbildung $f : A \rightsquigarrow B$ bezeichnen wir eine Struktur der Form

$$(f,A,B)$$

für die f eine Menge geordneter Paare der Form $f \subseteq \{(a,b) \mid a \in A, b \in B\}$ mit $(a,b) \in f \ \wedge \ (a,b') \in f \ \Rightarrow \ b = b'$. Wir schreiben dann auch $f(a)$ für das eindeutig bestimmte $b \in B$ mit $(a,b) \in f$.

Wir bezeichnen

- f als *Graph von* f
- A als *Quellmenge von* f
- B als *Zielmenge von* f

Definition 2.0.5: [Domain/Bild]

Sei $f : A \rightsquigarrow B$ eine Abbildung. Wir definieren

- $dom(f) := \{a \in A \mid \exists b \in B : \ (a,b) \in f\}$ und bezeichnen $dom(f)$ als den *Definitionsbereich* von f.
- $im(f) := \{b \in B \mid \exists a \in A : \ (a,b) \in f\}$ und bezeichnen $im(f)$ als das *Bild* von f.

Definition 2.0.6: [partiell/total]

Sei $f : A \rightsquigarrow B$ eine Abbildung. Wir bezeichnen f als

- *partiell* gdw. $dom(f) \subsetneq A$
- *total* gdw. $dom(f) = A$.

Ist f total so schreiben wir auch $f : A \to B$ für $f : A \rightsquigarrow B$.

Definition 2.0.7: [Folge]

Sei K eine beliebige Klasse. Als K-Folge bezeichnen wir den Graphen einer Abbildung der Form $\pi : \alpha \to K$ für eine Ordinalzahl $\alpha \in Ord_{\leq \omega}$.

- Für $\alpha < \omega$ bezeichnen wir π auch als endliche Folge (der Länge $|\alpha|$)
- Für $\alpha = \omega$ als unendliche Folge (der Länge $|\omega|$)
- Wir schreiben auch π_i für $\pi(i)$

Notation 2.0.8:

- Wir nutzen für eine Folge π wie üblich die Klammer-Notation

$$(\pi_i)_{i \in dom(\pi)} = (\pi_0, \pi_1, \dots)$$

- Mit ϵ bezeichnen wir auch die eindeutig bestimmte Folge π mit $|\pi| = 0$

Bemerkung 2.0.9:

Wir nutzen die natürliche Ordnung der Ordinalzahlen um Längen von Folgen miteinander zu vergleichen.

Definition 2.0.10: [kartesisches Produkt]

Seien $K_1, \dots, K_{n-1}$ beliebige Klassen. Als *kartesischen Produkt* der $K_1, \dots, K_{n-1}$ bezeichnen wir die Klasse aller $(\bigcup_i K_i)$-Folgen π mit

$$\forall i \in \{0, \dots, n-1\}_{\mathbb{N}} : \ \pi_i \in K_i$$

Wir schreiben für diese auch $K_1 \times \dots \times K_n$.

Definition 2.0.11: [Konkatenation]

Sei $\pi : \alpha \to M$ und $\pi' : \beta \to M'$ beliebige Folgen mit $\alpha < \omega$. Als *Konkatenation* von π und π' bezeichnen wir die Folge $\mu : (\alpha + \beta) \to S$ mit

$$\mu_x = \begin{cases} \pi(x) & : \ x < \alpha \\ \pi'(x - \alpha) & : \ x \geq \alpha \end{cases}$$

Wir schreiben für diese auch $\pi.\pi'$.

Bemerkung 2.0.12:

$\alpha + \beta$ sowie $x - \alpha$ ist hier als gewöhnliche Ordinal-Arithmetik aufzufassen.

Relationen & Funktionen

Im Fokus dieses Abschnitts steht die Verallgemeinerung des Funktions- und Relationsbegriffs auf Klassen.

Definition 2.0.13: [relationale Klasse]

Seien $K_1,\ldots,K_{n-1}$ beliebige Klassen. Wir bezeichnen ein $R \subseteq K_1 \times \ldots \times K_n$ als *(n-stellige) relationale Klasse auf* $K_1,\ldots,K_{n-1}$.

Definition 2.0.14: [Relation]

Wir bezeichnen eine (n-stellige) relationale Klasse R auch als *(n-stellige) Relation*, falls R eine Menge ist.

Notation 2.0.15:

Für eine 2-stellige relationale Klasse R schreiben wir auch sRs' für $(s,s') \in R$.

Definition 2.0.16: [Umkehrrelation]

Sei $R \subseteq A \times B$ eine relationale Klasse, dann ist die *Umkehrrelation* R^{-1} von R definiert durch $R^{-1} \subseteq B \times A$ mit

$$R^{-1} = \{(b,a) \mid (a,b) \in R\}$$

Definition 2.0.17: [Domain/Bild]

Sei $R \subseteq A \times B$ eine relationale Klasse. Wir definieren

- $dom(R) := \{a \in A \mid \exists b \in B : \ (a,b) \in R\}$ und bezeichnen $dom(R)$ als den *Definitionsbereich* von R
- $im(R) := \{b \in B \mid \exists a \in A : \ (a,b) \in R\}$ und bezeichnen $im(R)$ als das *Bild* von R

Außerdem sei

- $R(a) := \{b \in B \mid (a,b) \in R\}$
- $R(A') := \{b \in B \mid \exists a \in A' : \ (a,b) \in R\}$ für $A' \subseteq A$
- $R|_{A'} := \{(a,b) \in R \mid a \in A'\}$ für $A' \subseteq A$

Definition 2.0.18: [Attribute]

Wir bezeichnen eine relationale Klasse $R \subseteq A \times B$ als

- *reflexiv* gdw. $\forall a \in A : \ (a,a) \in R$
- *symmetrisch* gdw. $\forall (a,b) \in R : \ (b,a) \in R$
- *anti-symmetrisch* gdw. $\forall (a,b),(b,a) \in R : \ a = b$
- *transitiv* gdw. $\forall (a,b),(b,c) \in R : \ (a,c) \in R$

Wir bezeichnen eine relationale Klasse $f \subseteq X \times Y$ als

- *linkstotal* gdw. $\forall x \in X \exists y \in Y : \ (x,y) \in f$
- *rechtstotal/surjektiv* gdw. $\forall y \in Y \exists x \in X : \ (x,y) \in f$
- *linkseindeutig/injektiv* gdw. $\forall (x,y),(x',y) \in f : \ x = x'$
- *rechtseindeutig* gdw. $\forall (x,y),(x,y') \in f : \ y = y'$
- *bijektiv* gdw. linkstotal, rechtstotal, linkseindeutig, rechtseindeutig

Definition 2.0.19: [relationale Komposition]

Seien R und R' relationale Klassen mit $im(R) \subseteq dom(R')$, dann ist die *relationale Komposition* $R' \circ R$ definiert als

$$R' \circ R := \{ \ (r,r') \mid \exists s \in im(R) : \ (r,s) \in R \wedge (s,r') \in R' \ \}$$

Definition 2.0.20: [Äquivalenzrelation]

Wir bezeichnen eine relationale Klasse $R \subseteq M \times M$ als

- *Äquivalenzrelation*, wenn diese reflexiv, symmetrisch und transitiv ist
- *partielle Äquivalenzrelation*, wenn diese symmetrisch und transitiv ist

Für $m \in M$ definieren wir

$$[m]_R := \{m' \in M \mid (m,m') \in R\}$$

und bezeichnen $[m]_R$ als die *R-Äquivalenzklasse von m*. Wir schreiben auch nur $[m]$ falls R aus dem Kontext ersichtlich ist. Ist R eine Relation schreiben wir auch $M/_R$ für die Menge aller R-Äquivalenzklassen, d.h.

$$M/_R := \{[m]_R \mid m \in M\}$$

Bemerkung 2.0.21:

- Sei $R \subseteq M \times M$ eine Äquivalenzrelation, dann gilt

  $$M = \dot{\bigcup_{m \in M}} [m]_R$$

  d.h. R induziert eine Partitionierung von M
- Sei $R \subseteq M \times M$ eine partielle Äquivalenzrelation, dann gilt

  $$dom(R) = \dot{\bigcup_{m \in M}} [m]_R$$

  d.h. R induziert eine Partitionierung von $dom(R) \subseteq M$

Bemerkung 2.0.22:

Sei M eine Menge sowie $M' \subseteq M$ eine zugehörige Teilmenge.

- Jede M-Partitionierung $(P_i)_{i \in I}$, d.h. $M = \dot{\bigcup}_{i \in I} P_i$, induziert eine Äquivalenzrelation $R \subseteq M \times M$ durch $(x,y) \in R \ :\Leftrightarrow \ \exists i : \ x,y \in P_i$
- Jede M'-Partitionierung $(P_i)_{i \in I}$, d.h. $M' = \dot{\bigcup}_{i \in I} P_i$, induziert eine partielle Äquivalenzrelation $R \subseteq M \times M$ durch $(x,y) \in R \ :\Leftrightarrow \ \exists i : \ x,y \in P_i$

Notation 2.0.23:

Sei K eine Klasse. Wir definieren ...

- $rel(K) := \mathscr{P}(K \times K)$
- $per(K) := \{R \in rel(K) \mid R \text{ ist partielle Äquivalenzrelation}\}$
- $er(K) := \{R \in rel(K) \mid R \text{ ist Äquivalenzrelation}\}$

Definition 2.0.24: [funktionale Klasse]

Wir bezeichnen eine relationale Klasse $f \subseteq X \times Y$ als *funktionale Klasse* gdw. f rechtseindeutig ist. Wir schreiben

- $f : X \to Y$, falls $dom(f) = X$
- $f : X \rightsquigarrow Y$, falls $dom(f) \subseteq X$

Definition 2.0.25: [Funktion]

Wir bezeichnen eine funktionale Klasse f als *Funktion* gdw. f eine Menge ist.

Bemerkung 2.0.26:

- Wir werden funktionale Klassen wie üblich durch Zuordnungsvorschriften, Wertetabellen, etc. angeben.
- Für eine funktionale Klasse $f : X \to Y$ und $x \in X$ ist $f(x)$ stets von der Form $f(x) = y$ für das eindeutig bestimmte $y \in Y$ mit $(x,y) \in f$. Wir identifizieren daher $f(x)$ mit y
- Für eine funktionale Klasse $f : X \to Y$ und $X' \subseteq X$ schreiben wir $f(X') = const$ gdw. $f(X') = \{y\}$ für ein $y \in Y$

Bemerkung 2.0.27:

Jede Funktion f induziert auf natürliche Weise eine Abbildung der Form $(f,dom(f),im(f))$. Man beachte, dass dies für eine funktionale Klasse in der Regel nicht möglich ist, da hier $dom(f)$ und $im(f)$ möglicherweise echte Klassen darstellen und somit $(f,dom(f),im(f))$ nicht definiert wäre.

Bemerkung 2.0.28:

Jede Abbildung der Form (f,A,B) induziert auf natürliche Weise die Funktion f.

Ordnungen

In diesem Abschnitt soll der Begriff der "Ordnung" wiederholt werden. Hierbei möchten wir insbesondere an die Definition der "kanonischen Äquivalenzrelation" von Präordnungen erinnern.

Definition 2.0.29: [relationale Ordnungsklasse]

Wir bezeichnen eine binäre relationale Klasse als *relationale Ordnungsklasse*, falls diese transitiv ist.

Definition 2.0.30: [Ordnung]

Wir bezeichnen eine relationale Ordnungsklasse $\preceq \subseteq M \times M$ als *Ordnung* gdw. $\preceq$ eine Menge ist.

Definition 2.0.31: [Totalordnung]

Sei $\preceq \subseteq K \times K$ eine relationale Ordnungsklasse. Wir bezeichnen $\preceq$ als *Totalordnung* gdw.

$$\forall k,k' \in K : \; k \preceq k' \; \vee \; k' \preceq k$$

Definition 2.0.32: [Ordnungstypen]

Wir bezeichnen eine relationale Ordnungsklasse als

- *Präordnung*, falls diese reflexiv ist

- *Halbordnung*, falls diese reflexiv und anti-symmetrisch ist

Definition 2.0.33: [kanonische Äquivalenzrelation auf Präordnungen]

Sei $\preceq$ eine Präordnung, dann induziert diese eine Äquivalenzrelation $\sim$ mit

$$A \sim B \text{ gdw. } A \preceq B \ \wedge \ B \preceq A$$

Bemerkung 2.0.34:

Sei $\preceq$ eine Quasiordnung sowie $\sim$ die durch $\preceq$ induzierte Äquivalenzrelation, dann ist $\sim$ tatsächlich eine Äquivalenzrelation.

Beweis.
Folgt direkt aus Reflexivität und Transitivität von $\preceq$.

□

Mathematische Konventionen

In diesem Abschnitt sollen lediglich einige notationelle Konventionen eingeführt werden.

Notation 2.0.35:

Sei f eine funktionale Klasse sowie $\bullet \subseteq dom(f) \times dom(f)$ eine 2-stellige relationale Klasse. Wir schreiben dann

$$a \bullet_f b \text{ für } f(x_1) \bullet f(x_2)$$

Wir fassen $\bullet_f \subset im(f) \times im(f)$ dabei als binäre relationale Klasse auf.

Beispiel 2.0.36:

Sei $f : \mathbb{Z} \to \mathbb{Z}$ mit $n \mapsto n^2$, dann gilt

- $-2 =_f 2$ gdw. $4 = 4$
- $2 \leq_f 3$ gdw. $4 \leq 9$

Notation 2.0.37:

Für eine konstante Funktion $f : X \to \{y\}$ identifizieren wir f auch mit y.

Notation 2.0.38:

Wir nutzen für Intervalle wie üblich die Klammer-Notation (),[] annotieren diese jedoch zusätzlich mit dem zugehörigen Wertebereich

Beispiel 2.0.39:

- $[0,10)_{\mathbb{N}} = \{0,2,...,9\}$
- $[0,10)_{\mathbb{R}} = \{x \in \mathbb{R} \mid 0 \leq x < 10\}$

Notation 2.0.40:

Für eine Grammatik G in beliebiger Darstellungsform schreiben wir $Lng(G)$ für die durch G induzierte Sprache.

Notation 2.0.41:

Wir schreiben

- $\exists^{\leq 1} x : \varphi(x)$ abkürzend für

$$\forall x,y : \varphi(x) \wedge \varphi(y) \;\Rightarrow\; x = y$$

und drücken damit aus *"Es gibt höchstens ein x mit $\varphi(x)$"*

- $\exists! x : \varphi(x)$ abkürzend für

$$\exists x : \varphi(x) \wedge \forall y : \varphi(y) \Rightarrow x = y$$

und drücken damit aus *"Es gibt genau ein x mit $\varphi(x)$"*

Notation 2.0.42:

Wahrscheinlichkeitsverteilungen]

- Für $n \in \mathbb{N}$ schreiben $\mathcal{D}(n)$ für die Menge aller Wahrscheinlichkeitsverteilungen der Form $\rho : [0,n]_{\mathbb{N}} \to [0,1]_{\mathbb{R}}$
- Wir bezeichnen mit $\mathcal{D}(*) := \bigcup_{n \in \mathbb{N}} \mathcal{D}(n)$ die Menge aller Wahrscheinlichkeitsverteilungen auf natürlichen Zahlen.

Elementare Fixpunkttheorie

In diesem Abschnitt soll an die Grundlagen der elementaren Fixpunkttheorie erinnert werden. Wir beziehen uns hierbei weitestgehend auf [M10].

Definition 2.0.43: [vollständiger Verband]

Sei R eine beliebige Menge sowie $\subseteq_R \in Rel(R)$ eine Halbordnung. Wir bezeichnen $(R, \subseteq_R)$ als *vollständigen Verband*, falls jede Teilmenge $R' \subseteq R$ eine kleinste obere Schranke besitzt. Wir bezeichnen diese auch mit $\bigcup_R R'$.

Definition 2.0.44: [Kette]

Sei $(R, \subseteq_R)$ ein vollständiger Verband. Wir bezeichnen eine Teilmenge $R' \subseteq R$ als Kette gdw.

$$\forall r,r' \in R: \; r \subseteq_R r' \; \vee \; r' \subseteq_R r$$

Definition 2.0.45: [(Prä/Post-)Fixpunkt]

Sei $(R, \subseteq_R)$ ein vollständiger Verband sowie $f : R \to R$ eine beliebige Funktion. Wir bezeichnen ein Element $x \in R$ als

- *Prä-Fixpunkt von* f gdw. $f(x) \subseteq_R x$
- *Post-Fixpunkt von* f gdw. $x \subseteq_R f(x)$
- *Fixpunkt von* f gdw. $f(x) = x$

Proposition 2.0.46:

- Für einen vollständigen Verband $(R, \subseteq_R)$ besitzt jede Teilmenge $R' \subseteq R$ auch eine größte untere Schranke $\bigcap_R R' := \bigcup\{r \in R \mid \forall r' \in R': \; r \subseteq_R r'\}$
- Kleinste obere Schranken sind aufgrund ihrer Minimalität eindeutig
- Größte untere Schranken sind aufgrund ihrer Maximalität eindeutig
- In jedem vollständigen Verband $(R, \subseteq_R)$ existiert ein eindeutig bestimmtes kleinstes Element, nämlich $\bot_R := \bigcup_R \varnothing = \bigcap_R R$ sowie ein eindeutig bestimmtes größtes Element, nämlich $\top_R := \bigcap_R \varnothing = \bigcup_R R$

Beweis.
Wir verweisen hierfür auf [M10].

□

Definition 2.0.47: [Monotonie]

Sei $(R, \subseteq_R)$ eine Halbordnung sowie $F : R \to R$. Wir bezeichnen F als *monoton* gdw.

$$\forall r,r' \in R : \; r \subseteq_R r' \;\Rightarrow\; F(r) \subseteq_R F(r')$$

Definition 2.0.48: [Stetigkeit]

Sei $(R, \subseteq_R)$ ein vollständiger Verband sowie $F : R \to R$, dann bezeichnen wir F als *stetig* gdw. für jede nichtleere Kette $R' \subseteq R$ gilt

$$F(\bigcap_R R') = \bigcap_R \{F(P) \mid P \in R'\}$$

Theorem 2.0.49: [Kleene]

Sei $(R, \subseteq_R)$ ein vollständiger Verband sowie $F : R \to R$ eine monotone, stetige Funktion, dann besitzt F

- einen kleinsten Fixpunkt $lfp(F)$ in R und es gilt $lfp(F) = \bigcup_R \{F^n(\bot_R) \mid n \in \mathbb{N}\}$
- einen größten Fixpunkt $gfp(F)$ in R und es gilt $gfp(F) = \bigcap_R \{F^n(\top_R) \mid n \in \mathbb{N}\}$

Beweis.
Wir verweisen hierfür auf [M10].

□

Theorem 2.0.50: [Knaster-Tarski]

Sei $(R, \subseteq_R)$ ein vollständiger Verband sowie $F : R \to R$ eine monotone Funktion, dann besitzt F

- einen kleinsten Fixpunkt $lfp(F)$ in R und es gilt $lfp(F) = \bigcap_R \{R \in L \mid F(R) \subseteq R\}$, d.h. insbesondere $lfp(F)$ ist der kleinste Prä-Fixpunkt
- einen größten Fixpunkt $gfp(F)$ in R und es gilt $gfp(F) = \bigcup_R \{R \in L \mid R \subseteq F(R)\}$, d.h. insbesondere $gfp(F)$ ist der größte Post-Fixpunkt

Beweis.
Wir verweisen hierfür auf [M10].

□

Beispiel 2.0.51:

Für jeder beliebige Menge M ist $(\mathscr{P}(M), \subseteq)$ ein vollständiger Verband.

KAPITEL 3

System-Modellierung

Ziel dieses Kapitels stellt die Fixierung einer Modellierungsstruktur zur Erfassung bestehender Systeme und Formalismen dar. Fundament der Modellierungen wird dabei die Struktur der Transitionssysteme darstellen. Diese beschreiben eine graphenartige Struktur zur natürliche Erfassung zustandsbasierter Ansätze. Wir werden diese im Laufe des Kapitels um Möglichkeiten der Ausgabe ergänzen und somit die Kommunikation mit der Außenwelt mit in das Modell integrieren. Abschließend ergänzen wir diese um "Programmfunktionen" und erlangen so auf Modell-Ebene eine formalisierte Darstellung des Programm-Begriffs. Finales Resultat wird ein für unsere Zwecke vollständiges Modell darstellen, auf welche unsere anschließenden Überlegen aufsetzen können.

3.1 Transitionssysteme

In diesem Abschnitt werden wir uns mit der Theorie rund um die Struktur der Transitionssysteme befassen. Transitionssysteme stellen eine fundamentale und zugleich populäre Modellierungsstruktur der theoretischen Informatik dar. Sie weisen einen hohen Abstraktionsgrad auf und definieren dadurch einen mächtigen Modellierungsrahmen, in welche sich zustandsbasierte (berechnungsuniverselle) Maschinen und Formalismen auf natürliche Weise einbetten lassen. Wir wollen den Begriff des Transitionssystems zunächst formal definieren. Anschließend werden wir einen formalen Rahmen zum Umgang und Analyse dieser schaffen.

Definition 3.1.1: [Transitionssystem]

Wir bezeichnen eine Struktur der Form (S,T) als *Transitionssystem*, wobei

- S eine beliebige Menge (von (S,T)-*Systemzuständen*)
- $T \subseteq S \times S$ eine beliebige Menge (von (S,T)-*Transitionen*)

Notation 3.1.2:

Für ein Transitionssystem $\mathcal{T}$ schreiben wir

- $S^{(\mathcal{T})}$ für die Menge der $\mathcal{T}$-Systemzustände
- $T^{(\mathcal{T})}$ für die Menge der $\mathcal{T}$-Transitionen

Transitionssystem kapseln somit eine beliebige Menge von Systemzuständen zusammen mit einer entsprechenden Menge unidirektionaler Transitionen und können somit als graphenartige Struktur aufgefasst werden. Versucht man sich Transitionssysteme anhand eines handelsüblichen Computers zu veranschaulichen, dann entsprechen die Systemzustände gerade konkreten Ausprägungen des Arbeitsspeichers, die Transitionen kann man mit dem Verhalten des Prozessors identifizieren.

Bemerkung 3.1.3:

Wie bei graphenartigen Strukturen üblich bietet es sich an, sich Transitionssysteme graphisch zu veranschaulichen. Wir interpretieren hierfür Systemzustände als Knoten und fassen Transitionen als gerichtete Kanten zwischen diesen auf (vgl. hierzu auch Abbildung 3.1).

Gelegentlich wird es für gesonderte Betrachtungen von Interesse sein die Transitionen eines Transitionssystems bidirektional aufzufassen. Aufgrund dessen wollen wir Möglichkeit haben dies auf formaler Ebene ausdrücken zu können.

Definition 3.1.4: [ungerichtetes Transitionssystem]

Sei $\mathcal{T} = (S,T)$ ein Transitionssystem. Wir bezeichnen das Transitionssystem $\overleftrightarrow{\mathcal{T}}$ als das *ungerichtete Transitionssystem zu* $\mathcal{T}$ wobei

$$\overleftrightarrow{\mathcal{T}} = (S,\ T \cup T^{-1})$$

Bemerkung 3.1.5:

In der Regel sind wir bei ungerichteten Transitionssystemen vorrangig an deren Transitionen interessiert. Man beachte daher für ein Transitionssystem $\mathcal{T}$, dass $T^{(\overleftrightarrow{\mathcal{T}})}$ als Menge von $\mathcal{T}$-Transitionen aufgefasst werden darf.

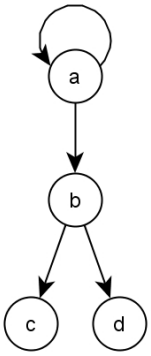


Abbildung 3.1: Graphische Darstellung des Transitionssystems $\mathcal{T}$ = ($\{a,b,c,d\}$, $\{a \to a, a \to b, b \to c, b \to d\}$)

Beispiel 3.1.6:

Bezogen auf Abbildung 3.1 gilt

$$\overleftrightarrow{\mathcal{T}} = (\ \{a,b,c,d\}\ ,\ \{(a,a),(a,b),(b,a),(b,c),(c,b),(b,d),(d,b)\}\ )$$

Im weiteren Verlauf dieses Abschnitts wollen wir uns grundlegende Hilfsmittel zum Umgang und zur Klassifizierung von Transitionssystemen schaffen. Wir beginnen hierbei mit der Definition eines Pfads. Diese beschreiben Zustandsfolgen eines Systems bei gleichzeitiger Respektierung der zugehörigen Transitionsrelation. Diese werden wir zunächst in allgemeiner Form definieren. Anschließend erschaffen wir uns Möglichkeit diese bzgl. Startpunkt und Länge einzuschränken.

Definition 3.1.7: [Pfad]

Sei $\mathcal{T}$ ein Transitionssystem. Wir bezeichnen eine Folge $\pi : \alpha \to S^{(\mathcal{T})}$ als $\mathcal{T}$*-Pfad* gdw.

$$\forall i \in [0,|\pi|)_{\mathbb{N}} :\ (\pi_i,\pi_{i+1}) \in T^{(\mathcal{T})}$$

Notation 3.1.8:

Wir schreiben für einen $\mathcal{T}$-Pfad $\pi = (\pi_0,\pi_1,\dots)$ auch

$$\pi_0 \to \pi_1 \to \dots$$

Bemerkung 3.1.9:

Da wir Transitionen und Pfade strukturell als Folgen definiert haben, dürfen wir Transitionen mit Pfaden (der Länge 2) identifizieren.

Beispiel 3.1.10:

Bezogen auf Abbildung 3.1 gilt

- (a,b,c) ist $\mathcal{T}$-Pfad
- $(a,a,\dots)$ ist $\mathcal{T}$-Pfad

Hierbei beschreibt technisch gesehen

- (a,b,c) den Graph der Abbildung $\pi : \{0,1,2\} \to \{a,b,c\}$ mit $0 \mapsto a$, $1 \mapsto b$, $2 \mapsto c$
- $(a,a,\dots)$ den Graph der Abbildung $\pi : \mathbb{N} \to \{a\}$ mit $n \mapsto a$

Definition 3.1.11: [spezielle Pfade]

Sei $\mathcal{T}$ ein Transitionssystem. Sei weiter $\bullet_1 \in S^{(\mathcal{T})} \cup \{*\}$ sowie $\bullet_2 \in \mathbb{N} \cup \{*,\omega\}$. Wir definieren

$$
{}_{(\bullet_1)}\mathcal{T}^{(\bullet_2)} := \{\ \pi \text{ ist } \mathcal{T}\text{-Pfad} \mid \pi_0 = \begin{cases} s & : \bullet_1 = s \in S^{(\mathcal{T})} \\ s \in S^{(\mathcal{T})} \text{ beliebig} & : \bullet_1 = * \end{cases} \wedge\ |\pi| = \begin{cases} n & : \bullet_2 = n \in \mathbb{N} \\ n \in \mathbb{N} \text{ beliebig} & : \bullet_2 = * \\ \omega & : \bullet_2 = \omega \end{cases} \}
$$

Wir bezeichnen $\pi \in {}_{(\bullet_1)}\mathcal{T}^{(\bullet_2)}$ als

- *$\mathcal{T}$-Pfad in s*, falls $\bullet_1 = s \in S^{(\mathcal{T})}$
- *endlichen $\mathcal{T}$-Pfad (der Länge n)*, falls $\bullet_2 = n \in \mathbb{N}$
- *endlichen $\mathcal{T}$-Pfad*, falls $\bullet_2 = *$
- *unendlichen $\mathcal{T}$-Pfad*, falls $\bullet_2 = \omega$

Beispiel 3.1.12:

Bezogen auf Abbildung 3.1 gilt

- $(a,b,c) \in {}_{(a)}\mathcal{T}^{(3)} \subseteq {}_{(*)}\mathcal{T}^{(3)} \subseteq {}_{(*)}\mathcal{T}^{(*)}$
- $(a,a,\dots) \in {}_{(a)}\mathcal{S}^{(*)} \subseteq {}_{(*)}\mathcal{T}^{(*)}$
- $(a,a).(b,c) = (a,a,b,c) \in {}_{(*)}\mathcal{T}^{(*)}$
- $(a,a,\dots).(a) \notin {}_{(*)}\mathcal{T}^{(*)}$

Bemerkung 3.1.13:

Man beachte hier insbesondere, dass die Definitionen in 3.1.11 sich nicht gegenseitig ausschließen. Betrachtet man beispielsweise in Beispiel 3.1.12 den Pfad (a,b,c), dann ist dieser sowohl ein "endlicher $\mathcal{T}$-Pfad der Länge 3 in a" und somit auch ein "endlicher $\mathcal{T}$-Pfad der Länge 3" und somit auch ein "endlicher Pfad". Formal geht dies aus der Tatsache ${}_{(a)}\mathcal{T}^{(3)} \subseteq {}_{(*)}\mathcal{T}^{(3)} \subseteq {}_{(*)}\mathcal{T}^{(*)}$ hervor.

Wir kommen aufbauend zu dem Begriff der Erreichbarkeit. Dieser ist spezifische für ein Transitionssystem $\mathcal{T}$ sowie einem entsprechenden $\mathcal{T}$-Systemzustand s definiert und beschreibt eine Form in gewissem Sinne den "deduktiven Abschluss" von s bzgl. $T^{(\mathcal{T})}$.

Definition 3.1.14: [Erreichbarkeit]

Sei $\mathcal{T}$ ein Transitionssystem sowie $s \in S^{(\mathcal{T})}$ beliebig. Die Menge der *von s erreichbaren $\mathcal{T}$-Systemzustände* $reach_{\mathcal{T}}(s)$ ist definiert durch

$$reach_{\mathcal{T}}(s) := \{s' \in S \mid \exists \pi \in {}_{(*)}\mathcal{T}^{(*)} : \ \pi = (s,\ldots,s')\}$$

Beispiel 3.1.15:

Bezogen auf Abbildung 3.1 gilt

- $reach_{\mathcal{T}}(a) = \{a,b,c,d\}$
- $reach_{\mathcal{T}}(b) = \{b,c,d\}$

Im Verlauf dieser Arbeit wird es für vielerlei Betrachtungen von Interesse sein Transitionssysteme auf gewisse Teilbereiche einzuschränken zu können. Dies ist beispielsweise dann vonnöten um gewissen Teilstrukturen zu fokussieren oder aber um von gewissen Informationen zu abstrahieren. Wir definieren aufgrund dessen den Begriff des "Teilsystems". Diese sind konkreten Transitionssystem zugeordnet und beschreiben eine Reduktion der Zustandsmengen bei gleichzeitiger Erhaltung der zugehörigen Transitionsrelation.

Definition 3.1.16: [Teilsystem]

Sei $\mathcal{T}$ ein Transitionssystem. Als *$\mathcal{T}$-Teilsystem* bezeichnen wir ein Transitionssystem $\mathcal{T}'$ mit

- $S^{(\mathcal{T}')} \subseteq S^{(\mathcal{T})}$
- $T^{(\mathcal{T}')} = T^{(\mathcal{T})} \cap \left(S^{(\mathcal{T}')} \times S^{(\mathcal{T}')}\right)$

Notation 3.1.17:

- Wir schreiben auch $\mathcal{T}' \sqsubseteq \mathcal{T}$ für ein $\mathcal{T}$-Teilsystem $\mathcal{T}'$
- Wir schreiben auch $\mathcal{T}' \sqsubset \mathcal{T}$, falls $S^{(\mathcal{T}')} \subset S^{(\mathcal{T})}$

Bemerkung 3.1.18:

$\mathcal{T}$-Teilsysteme eines beliebigen Transitionssystems $\mathcal{T}$ sind durch ihre Systemzustandsmenge eindeutig bestimmt.

Definition 3.1.19: [zustands-induziertes Teilsystem]

Sei $\mathcal{T}$ ein Transitionssystem sowie $S \subseteq S^{(\mathcal{T})}$ beliebig. Als S *induzierte* $\mathcal{T}$*-Teilsystem* bezeichnen wir das nach Bemerkung 3.1.18 eindeutig bestimmte $\mathcal{T}$-Teilsystem $\mathcal{T}'$ mit

$$S^{(\mathcal{T}')} = \bigcup_{s \in S} reach_{\mathcal{T}}(s)$$

Notation 3.1.20:

- Wir schreiben auch $\mathcal{T}(S)$ für das S induzierte $\mathcal{T}$-Teilsystem
- Wir schreiben auch als $\mathcal{T}(s)$ für $\mathcal{T}(\{s\})$.

Beispiel 3.1.21:

Bezogen auf Abbildung 3.1 gilt

- $\mathcal{T}' = (\{a,c,d\}\ ,\ \{a \to a\})$ ist $\mathcal{T}$-Teilsystem
- $\mathcal{T}(b) = \{\{b,c,d\},\ \{b \to c\ ,\ b \to d\}\}$ ist $\mathcal{T}$-Teilsystem

Im Folgenden werden wir Attribute zur Klassifizierung von Transitionssystemen vorstellen und erläutern.

Definition 3.1.22: [Attribute]

Für ein Transitionssystem $\mathcal{T}$ definieren wir folgende Attribute.

gewurzelt	$\exists s \in S^{(\mathcal{T})} :\ reach_{\mathcal{T}}(s) = S^{(\mathcal{T})}$ Wir bezeichnen s dann als *Wurzel* von $\mathcal{T}$.
eindeutig gewurzelt	$\exists! s \in S^{(\mathcal{T})} :\ reach_{\mathcal{T}}(s) = S^{(\mathcal{T})}$ Wir bezeichnen s dann als *die Wurzel* von $\mathcal{T}$.
azyklisch	$\forall s,s' \in S^{(\mathcal{T})}\ \exists^{\leq 1} \pi \in {}_{(*)}\mathcal{T}^{(*)} :\ \pi = (s,\ldots,s')$

zyklisch	$\mathcal{T}$ nicht azyklisch
zusammenhängend	$\forall s \in S^{(\mathcal{T})} : \; reach_{\overleftrightarrow{\mathcal{T}}}(s) = S^{(\mathcal{T})}$
deterministisch	$T^{(\mathcal{T})}$ rechtseindeutig
endlich verzweigt	$\forall s \in S^{(\mathcal{T})} : \; \lvert\{s' \in S^{(\mathcal{T})} \mid s \to s' \in T^{(\mathcal{T})}\}\rvert < \aleph_0$
Baum	$\mathcal{T}$ gewurzelt und azyklisch

Beispiel 3.1.23:

Bezogen auf Abbildung 3.1 gelten folgende Attribute für $\mathcal{T}$.

- gewurzelt, mit Wurzel a
- eindeutig gewurzelt, mit Wurzel a
- nicht azyklisch aufgrund der Existenz der Pfade (a) und (a,a)
- zyklisch, da nicht azyklisch
- zusammenhängend
- nicht deterministisch aufgrund der Transitionen (b,c),(b,d)
- endlich verzweigt
- kein Baum, da zyklisch

Ein Transitionssystem ist gewurzelt, falls es von mindestens einem Systemzustand aufgespannt wird. Wird ein Transitionssystem von genau einem Systemzustand aufgespannt ist dieses eindeutig gewurzelt. Azyklische Transitionssysteme kann man sich frei von Konstrukten wie in Abbildung 3.2 vorstellen.

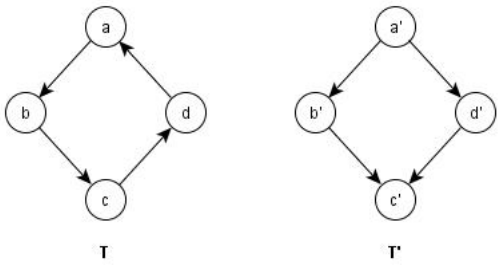


Abbildung 3.2: zyklische Transitionssysteme - Die Existenz der $\mathcal{T}$-Pfade (a) und (a,b,c,d,a) bzw. der $\mathcal{T}'$-Pfade (a',b',c') und (a',d',c') stehen im direkten Widerspruch zur Azyklizität von $\mathcal{T}$ bzw. $\mathcal{T}'$.

Entsprechend beinhalten zyklische Transitionssysteme gerade Konstrukte dieser Form. Zusammenhängende Transitionssysteme bilden ein in sich zusammengeschlossenes Gebilde. Auf anschaulicher Ebene bedeutet dies, dass je zwei Knoten durch eine Abfolge von Kanten, bei Missachtung der Pfeilrichtung, verbunden sind. Deterministische Transitionssysteme

haben einen linearen Charakter. Man kann sich diese als Zusammenfassung unabhängiger Pfade vorstellen. Endlich verzweigte Transitionssysteme haben in jedem Systemzustand lediglich eine endliche Auswahl an Folgezuständen.
Bäume wollen wir detaillierter betrachten, da diese für unsere kommenden Betrachtungen von besonderem Interesse sein werden. Hierzu stellen wir zunächst zwei Propositionen vor, welche sowohl einer Präzisierung unserer Anschauung als auch als Unterstützung in Beweisführungen kommender Theoreme dienen sollen. Erstere Proposition besagt dabei, dass Bäume eindeutig gewurzelt sind. Natürlich hätte man Bäume direkt als eindeutig gewurzelte, azyklische Transitionssysteme definieren können. In der Regel ist es jedoch wesentlich angenehmer die reine Existenz einer Wurzel nachzuweisen, ohne Argumente für dessen Eindeutigkeit erbringen zu müssen. Zweite Proposition besagt, dass gewurzelte Transitionssysteme zusammenhängend sind. Da Bäume per Definition gewurzelt sind, ergibt sich die Aussage für diese direkt als Korollar. Für spätere Zwecke werden wir darüber hinaus in einem Lemma beweisen, dass abgesehen von der Wurzel jeder Systemzustand eines Baums einen eindeutigen Vorgänger besitzt.

Proposition 3.1.24:

Bäume sind eindeutig gewurzelt.

Beweis.
Sei $\mathcal{T}$ ein Baum. Wir zeigen Existenz und Eindeutigkeit der Wurzel.

- [Existenz]

  Folgt direkt aus der Definition, denn Bäume sind unter anderem gewurzelt. ✓

- [Eindeutigkeit]

  Seien w und w' Wurzeln von $\mathcal{T}$. Angenommen $w \neq w'$, dann gilt per Definition einer Wurzel

  - $w' \in reach_{\mathcal{T}}(w)$ und folglich existiert ein $\mathcal{T}$-Pfad der Form $(w,\ldots,w')$
  - $w \in reach_{\mathcal{T}}(w')$ und folglich existiert ein $\mathcal{T}$-Pfad der Form $(w',\ldots,w)$

  Es ergibt sich somit die Existenz eines Pfads $\pi = (w,\ldots,w',\ldots,w)$. Aus der Annahme $w \neq w'$ folgt $|\pi| > 1$ und somit insbesondere $\pi \neq (w)$. Es ergibt sich direkt ein Widerspruch zur Azyklizität von $\mathcal{T}$ und somit auch ein Widerspruch zur Annahme. ↯

□

Proposition 3.1.25:

Gewurzelte Transitionssysteme sind zusammenhängend.

Beweis.
Sei $\mathcal{T}$ ein gewurzeltes Transitionssystem mit Wurzel w. Seien weiter $s,s' \in S^{(\mathcal{T})}$ beliebig $\mathcal{T}$-Systemzustände. Wir müssen die Existenz eines $\overleftrightarrow{\mathcal{T}}$-Pfads der Form $(s,\ldots,s')$ nachweisen. Aufgrund der Wurzel-Eingenschaft von w existiert sowohl ein $\mathcal{T}$-Pfad $(w,\ldots,s)$ als auch ein $\mathcal{T}$-Pfad $(w,\ldots,s')$. Folglich existiert auch ein $\overleftrightarrow{\mathcal{T}}$-Pfad $(s,\ldots,w,\ldots,s')$.

□

Proposition 3.1.26:

Bäume sind zusammenhängend.

Beweis.
Bäume sind per Definition gewurzelt. Die Behauptung folgt somit direkt aufgrund von Proposition 3.1.25.

□

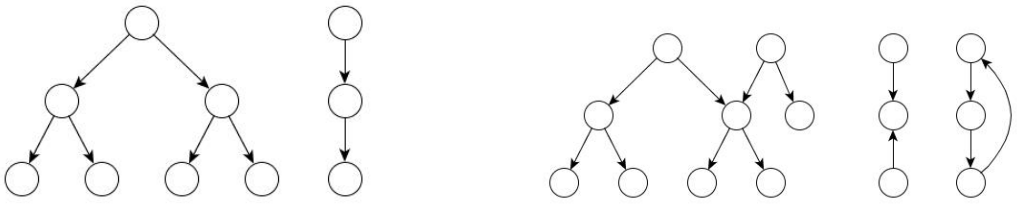
Lemma 3.1.27:

Für einen Baum $\mathcal{T}$ mit Wurzel w existiert für jedes $s \in S^{(\mathcal{T})}$ mit $s \neq w$ ein eindeutig bestimmtes $s' \in S^{(\mathcal{T})}$ mit $(s',s) \in T^{(\mathcal{T})}$

Beweis.
Gelten die selben Bezeichnungen wie oben. Wir zeigen Existenz und Eindeutigkeit von s'.

- [Existenz]

  Per Definition einer Wurzel existiert ein $\mathcal{T}$-Pfad der Form $\pi = (w,\ldots,s)$, mit $|\pi| \geq 2$, da $s \neq w$. Offensichtlich gilt für $s' := \pi_{|\pi|-2}$, dass $s' \to s \in T^{(\mathcal{T})}$. ✓



(a) Bäume **(b)** keine Bäume

Abbildung 3.3: Transitionssystem wie in Abbildung 3.3(b) stellen keine Bäume dar, da diese nicht gewurzelt in den ersten beiden Fällen bzw. zyklisch im dritten Fall

- [Eindeutigkeit]
  Sei $s'' \in S^{(\mathcal{T})}$ mit $s'' \to s \in T^{(\mathcal{T})}$. Per Definition einer Wurzel existiert nun sowohl ein $\mathcal{T}$-Pfade der Form $\pi'' = (w,\ldots,s'',s)$, als auch ein $\mathcal{T}$-Pfade der Form $\pi' = (w,\ldots,s',s)$. Es ergibt sich unmittelbar aus Azyzklität von $\mathcal{T}$, dass $\pi'' = \pi'$ und folglich auch $s'' = s'$. ✓

□

Wir werden abschließend klären, wann wir Transitionssysteme als isomorph auffassen.

Definition 3.1.28: [Isomorphie]

Wir bezeichnen zwei Transitionssystem $\mathcal{T}$ und $\mathcal{T}'$ als *isomorph* gdw. eine Bijektion $f : S^{(\mathcal{T})} \to S^{(\mathcal{T}')}$ existiert mit

$$(s_1 \to s_2) \in T^{(\mathcal{T})} \;\Leftrightarrow\; (f(s_1) \to f(s_2)) \in T^{(\mathcal{T}')}$$

Wir bezeichnen f dann als *Isomorphismus zwischen $\mathcal{T}$ und $\mathcal{T}'$* und schreiben

$$\mathcal{T} \simeq \mathcal{T}'$$

Hierbei interpretieren wir $\simeq$ als relationale Klasse.

Bemerkung 3.1.29:

Ist f ein Isomorphismus zwischen beliebigen Transitionssystemen $\mathcal{T}$ und $\mathcal{T}'$, dann ist f^{-1} ein Isomorphismus zwischen $\mathcal{T}'$ und $\mathcal{T}$. Insbesondere gilt

$$\mathcal{T} \simeq \mathcal{T}' \text{ gdw. } \mathcal{T}' \simeq \mathcal{T}$$

Zwei Transitionssysteme sind somit isomorph, wenn sie bis auf die Benennung ihrer Systemzustände gleich sind. Anschaulich bedeutet dies, dass ihre korrespondierenden Graphen bei Außerachtlassung ihren Knoteninhalten übereinstimmen.

3.2 Annotierte Transitionssysteme

Ziel dieses Abschnitts wird es sein die Struktur der Transitionssystem um Annotationen zu erweitern. Transitionssysteme umfassen die Modellierung einer “Rechenvorschrift“ und somit lediglich den Kern bestehender Systeme. Systeme interagieren jedoch in Form von wohldefinierten Ein- und Ausgaben mit ihrer Umwelt. Eingaben dienen der äußeren

Beeinflussung von Ausführungen und implizieren in der Regel ein nicht deterministisches Verhalten des Systems. Ausgaben dienen dazu Außenstehenden eine spezifizierte Sicht auf Systemdaten zu gewähren. Daten die nicht in Form von Ausgaben das System verlassen stellen dabei aus äußerer Sicht unzugängliche Informationen dar. Da wir in dieser Arbeit vorrangig an der Beobachtung von Systemen interessiert sind, werden wir explizite Möglichkeiten zur Systemmanipulation durch Eingaben vernachlässigen, insbesondere können diese als "externe Manipulationen" auch als Ausgaben aufgefasst werden. Wir werden uns daher lediglich der Modellierung von Ausgaben widmen. Ausgaben sollen dabei sowohl beim Erreichen von Zuständen, als auch bei Zustandsübergängen anfallen. Umsetzen werden wir dies durch Zustands- und Übergangsannotationen. Eine Zustands-Annotation soll dabei eine Funktion zur Abbildung von Systemzustände eines spezifischen Transitionssystems auf Elemente einer vollkommen beliebigen Menge sein. Elemente dieser Menge möchten wir als mögliche Ausgaben interpretieren, wobei wir das Symbol τ einheitlich für eine unterdrückte Ausgabe verwenden. Wie solche "Ausgabemengen" dabei auszusehen haben ist nicht weiter spezifiziert. Wir versprechen uns durch diese Freiheit maximale Abstraktion. Entsprechend soll eine Übergangs-Annotation eine Funktion zur Abbildung von Transitionen eines spezifischen Systems auf Elemente einer beliebigen Menge darstellen. Wir interpretieren die Bildmenge wieder als Menge von möglichen Ausgaben und verwenden τ auch in Bezug auf diese einheitlich als unterdrückte Ausgabe. In der Regel wird man dabei für ein konkretes Transitionssystem sowohl an einer entsprechenden Zustandsannotation, als auch an einer entsprechenden Übergangsannotation interessiert sein. Wir werden diese daher in simplen Tupeln zusammenfassen, welche wir als Annotationen bezeichnen werden.

Definition 3.2.1: [Zustands-Annotation]

Sei $\mathcal{T}$ ein Transitionssystem. Als *$\mathcal{T}$-Zustands-Annotation* bezeichnen wir eine Funktion $\mathfrak{a}_S$ mit $dom(\mathfrak{a}_S) = S^{(\mathcal{T})}$.

Bemerkung 3.2.2:

Funktionen $\mathfrak{a}_S$ mit $S^{(\mathcal{T})} \subseteq dom(\mathfrak{a}_S)$ betrachten wir ebenfalls als $\mathcal{T}$-Zustands-Annotationen, indem wir diese implizit als $\mathfrak{a}_S|_{S^{(\mathcal{T})}}$ auffassen.

Definition 3.2.3: [Übergangs-Annotation]

Sei $\mathcal{T}$ ein Transitionssystem. Wir bezeichnen eine Funktion $\mathfrak{a}_T$ als *$\mathcal{T}$-Übergangs-Annotation* gdw. $dom(\mathfrak{a}_T) = T^{(\mathcal{T})}$.

Bemerkung 3.2.4:

Funktionen $\mathfrak{a}_T$ mit $T^{(\mathcal{T})} \subseteq dom(\mathfrak{a}_T)$ betrachten wir ebenfalls als $\mathcal{T}$-Übergangs-Annotationen, indem wir diese implizit als $\mathfrak{a}_T|_{T^{(\mathcal{T})}}$ auffassen.

Definition 3.2.5: [Annotation]

Sei $\mathcal{T}$ ein Transitionssystem. Als $\mathcal{T}$-*Annotation* bezeichnen wir ein Tupel der Form $(\mathfrak{a}_S,\mathfrak{a}_T)$ für welche

- $\mathfrak{a}_S$ eine $\mathcal{T}$-Zustands-Annotation
- $\mathfrak{a}_T$ eine $\mathcal{T}$-Übergangs-Annotation

Notation 3.2.6:

Für eine $\mathcal{T}$-Annotation $\mathfrak{a}$ schreiben wir

- $\mathfrak{a}_S$ für die zu $\mathfrak{a}$ gehörige $\mathcal{T}$-Zustands-Annotation
- $\mathfrak{a}_T$ für die zu $\mathfrak{a}$ gehörige $\mathcal{T}$-Übergangs-Annotation

Wir haben nun alle nötigen Mittel erarbeitet um die Struktur des "annotierten Transitionssystems" zu definieren.

Definition 3.2.7: [annotiertes Transitionssystem]

Sei $\mathcal{T}$ ein Transitionssystem sowie $\mathfrak{a}$ eine $\mathcal{T}$-Annotation. Wir bezeichnen eine Struktur der Form $(\mathcal{T},\mathfrak{a})$ als *annotiertes Transitionssystem*.

Notation 3.2.8:

Wir schreiben auch $\mathcal{T}_{<\mathfrak{a}>}$ für $(\mathcal{T},\mathfrak{a})$.

Annotierte Transitionssysteme definieren simple Paare, bestehend aus einem Transitionssystem zusammen mit einer zugehörigen Annotation. Überträgt man seine Vorstellungen hierzu auf handelsübliche Computer beschreiben Annotationen gerade Ausgaben gegeben durch Bildschirm, Drucker und/oder Lautsprecher.

Beispiel 3.2.9:

Sei $\mathcal{T}$ definiert wie in Abbildung 3.1, dann ist $\mathfrak{a}$ mit

- $\mathfrak{a}_S : \{a,\ldots,z\} \to \mathbb{N}$ die Funktion die Buchstaben ihre natürliche Position im Alphabet zuordnet.
- $\mathfrak{a}_T : \{a,\ldots,z\}^2 \to \mathbb{N}$ mit $diff(\alpha,\beta) = |\#(\alpha) - \#(\beta)|$

eine $\mathcal{T}$-Annotation und $\mathcal{T}_{<\mathfrak{a}>}$ ein annotiertes Transitionssystem. Wir wollen uns dieses graphisch veranschaulichen wie in Abbildung 3.4.

Bemerkung 3.2.10:

Bei der Visualisierung von annotierten Zustandsgraphen werden wir auf das einzeichnen von τ-Symbolen verzichten.

Wir werden nun beispielhaft eine Modellierung pseudo-paraller Systeme vorstellen, welche auch im Fokus späterer Betrachtungen sein wird. Diese kann man sich als gewöhnliche annotierte Transitionssysteme vorstellen, dessen Übergänge mit Wahrscheinlichkeiten für die entsprechenden Zustandswechsel annotiert sind.

Beispiel 3.2.11:

Wir bezeichnen ein annotiertes Transitionssystem $\mathcal{T}_{<\mathfrak{a}>}$ als *probabilistisch* gdw.

1. $im(\mathfrak{a}_T) \subseteq [0,1]_{\mathbb{R}}$
2. $\forall s \in S^{\mathcal{T}} : \sum_{s \to s' \in T^{(\mathcal{T})}} \mathfrak{a}_T(s \to s') = 1$

Notation 3.2.12:

Sei $\mathcal{T}_{<\mathfrak{a}>}$ ein probabilistisches Transitionssystem. Sei weiter $s \in S^{(\mathcal{T})}$ sowie $S \subseteq S^{(\mathcal{T})}$. Wir schreiben

$$s \xrightarrow[\mathcal{T}_{<\mathfrak{a}>}]{\rho} S \quad \text{gdw.} \quad \rho = \sum_{\substack{s \to s' \in T^{(\mathcal{T})}, \\ s' \in S}} \mathfrak{a}_T(s \to s')$$

Falls $S = \{s'\}$ schreiben wir auch $s \xrightarrow[\mathcal{T}_{<\mathfrak{a}>}]{\rho} s'$ für $s \xrightarrow[\mathcal{T}_{<\mathfrak{a}>}]{\rho} \{s'\}$. Ist $\mathcal{T}_{<\mathfrak{a}>}$ aus dem Kontext ersichtlich schreiben wir auch nur " $\xrightarrow{\rho}$ " für " $\xrightarrow[\mathcal{T}_{<\mathfrak{a}>}]{\rho}$ ".

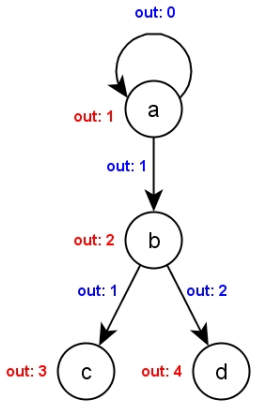


Abbildung 3.4: Graphische Darstellung von $\mathcal{T}_{<\mathfrak{a}>}$ aus Beispiel 3.2.9

Beispiel 3.2.13:

Sei $\mathcal{T}$ definiert wie in Abbildung 3.1, dann ist $\mathfrak{a}$ mit

- $\mathfrak{a}_S : \{a,\ldots,z\} \to \{\tau\}$ die Funktion die $\mathcal{T}$-Systemzustände einheitlich auf τ abbildet.
- $\mathfrak{a}_T : \{a,\ldots,z\}^2 \to [0,1]_{\mathbb{R}}$ mit $\mathfrak{a}_T(s \to s') = \frac{1}{|\{s' \mid s \to s' \in T^{(\mathcal{T})}\}|}$

eine $\mathcal{T}$-Annotation und $\mathcal{T}_{<\mathfrak{a}>}$ probabilistisch. Dabei kann die zugrunde gelegte Wahrscheinlichkeitsverteilung als eine Gleichverteilung betrachtet werden (vgl. hierzu auch Abbildung 3.5).

Wir werden abschließend klären, wann wir annotierte Transitionssysteme als isomorph auffassen.

Definition 3.2.14: [Isomorphie]

Wir bezeichnen zwei annotierte Transitionssysteme $\mathcal{T}_{<\mathfrak{a}>}$ und $\mathcal{T}'_{<\mathfrak{a}'>}$ als *isomorph* gdw. ein Isomorphismus $f : S^{(\mathcal{T})} \to S^{(\mathcal{T}')}$ zwischen $\mathcal{T}$ und $\mathcal{T}'$ existiert mit

(i) $\mathfrak{a}_S(s) = \mathfrak{a}'_S(f(s))$

(ii) $\mathfrak{a}_T(s \to t) = \mathfrak{a}'_T(f(s) \to f(t))$

Wir bezeichnen f dann als *Isomorphismus zwischen* $\mathcal{T}_{<\mathfrak{a}>}$ *und* $\mathcal{T}'_{<\mathfrak{a}'>}$ und schreiben

$$\mathcal{T}_{<\mathfrak{a}>} \simeq \mathcal{T}'_{<\mathfrak{a}'>}$$

Hierbei interpretieren wir $\simeq$ als relationale Klasse.

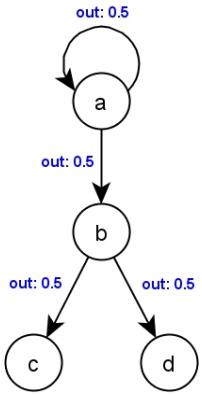


Abbildung 3.5: Graphische Darstellung von $\mathcal{T}_{<\mathfrak{a}>}$ aus Beispiel 3.2.13

Bemerkung 3.2.15:

Ist f ein Isomorphismus zwischen beliebigen annotierten Transitionssystemen $\mathcal{T}_{<\mathfrak{a}>}$ und $\mathcal{T}'_{<\mathfrak{a}'>}$, dann ist f^{-1} ein Isomorphismus zwischen $\mathcal{T}'_{<\mathfrak{a}'>}$ und $\mathcal{T}_{<\mathfrak{a}>}$. Insbesondere gilt

$$\mathcal{T}_{<\mathfrak{a}>} \simeq \mathcal{T}'_{<\mathfrak{a}'>} \text{ gdw. } \mathcal{T}'_{<\mathfrak{a}'>} \simeq \mathcal{T}_{<\mathfrak{a}>}$$

Zwei annotierte Transitionssysteme sind somit isomorph, wenn sie bis auf Benennung ihrer Systemzustände gleich sind. Anschaulich bedeutet dies, dass ihre korrespondierenden Graphen bei Außerachtlassung ihrer Knoteninhalte übereinstimmen.

3.3 Programmbasierte Transitionssysteme

In diesem Abschnitt sind wird an der Integration des Programmbegriffs in unsere bisherigen Modelle interessiert. Da wir vorrangig an der Errichtung eines Sicherheitskonzept für Computerprogramme interessiert sind, stellt die Existenz eines formalen Programmbegriffs eine nötige Voraussetzung dar. Zu diesem Zweck definieren wir zunächst den Begriff der Programmfunktion. Diese stellen ähnlich zu den Zustands-Annotationen Abbildungen von Systemzuständen in beliebige (Programm-)Mengen dar. Programme sind anschließend den einzelnen Systemzuständen zugeordnet und stellen die Bilder dieser unter einer festen Programmfunktion dar. Wie Programme bzw. Programmmengen auszusehen haben, ist dabei vollkommen beliebig und unterliegt keinen Einschränkungen. Durch dieses Vorgehen versprechen wir uns maximale Abstraktion.

Definition 3.3.1: [Programmfunktion]

Sei $\mathcal{T}$ ein Transitionssystem. Als *$\mathcal{T}$-Programmfunktion* bezeichnen wir eine Funktionen $\mathfrak{p}$ mit $dom(\mathfrak{p}) = S^{(\mathcal{T})}$.

Bemerkung 3.3.2:

Funktionen $\mathfrak{p}$ mit $S^{(\mathcal{T})} \subseteq dom(\mathfrak{p})$ betrachten wir ebenfalls als $\mathcal{T}$-Programmfunktionen, indem wir diese implizit als $\mathfrak{p}|_{S^{(\mathcal{T})}}$ auffassen.

Wir werden Programmfunktionen nun in das Modell der annotierten Transitionssysteme und der Vollständigkeit halber auch in die der Transitionssysteme integrieren und erhalten und gelangen so zu der Struktur der “programmbasierten Transitionssysteme“ bzw. zu

dieser der “annotiert-programmbasierte Transitionssysteme“, von lediglich die annotiert-programmbasierten Transitionssysteme von weiterem Interesse sind. Systemzustände annotiert-programmbasierter Transitionssysteme können dabei als Konfigurationen, d.h. als Programme samt der für die Ausführung nötigen Programmdaten, aufgefasst werden. Ein annotiert-programmbasiertes Transitionssystem ist somit eine Modellierungsstruktur welches ein auf dem Zustandsbegriff basierendes System samt Ausgaben und einem definierten Programmbegriff auf natürliche Weise erfassen kann. Wir werden diese auch als Systeme bezeichnen, da ihr Modell für unsere Zwecke vollständig ist, d.h. annotiert-programmbasierte Transitionssysteme kapseln in ihrer Struktur alle für uns relevanten Informationen, sodass wir diese in unserem Kontext ohne Einschränkung mit realen Systemen identifizieren dürfen.

Definition 3.3.3: [(annotiert-)programmbasierte Transitionssysteme]

Sei $\mathcal{T}$ ein Transitionssystem, $\mathfrak{a}$ eine $\mathcal{T}$-Annotation sowie $\mathfrak{p}$ eine $\mathcal{T}$-Programmfunktion. Wir bezeichnen

- $(\mathcal{T},\mathfrak{p})$ als *programmbasiertes Transitionssystem*
- $((\mathcal{T},\mathfrak{a}),\ \mathfrak{p})$ als *annotiert-programmbasiertes Transitionssystem* oder kurz als *System*.

Notation 3.3.4:

Wir schreiben auch

- $\mathcal{T}_{<\mathfrak{p}>}$ für $(\mathcal{T},\mathfrak{p})$
- $\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>}$ für $((\mathcal{T},\mathfrak{a}),\mathfrak{p})$

Zum Ende dieses Kapitels wollen wir die konstruierten Modellierungsstrukturen in entsprechenden Klassen zusammenfassen.

Definition 3.3.5: [Modell-Klassen]

Wir bezeichnen mit

- TS die Klasse der Transitionssysteme
- TS-A die Klasse der annotierte Transitionssysteme
- TS-P die Klasse der programmbasierten Transitionssysteme
- TS-A-P die Klasse der annotiert-programmbasierten Transitionssysteme

KAPITEL 4

Beobachter

Ziel dieses Kapitels stellt die formale Erfassung von Beobachtern dar. Beobachter fassen wir als eine passive, d.h. nicht in das System eingreifende Form von Angreifern auf, welche anhand ihrer Beobachtungen und eventuellen Interpretationen dieser versuchen Rückschlüsse auf geheime Daten zu treffen. Die Beobachtungen setzen dabei auf der spezifizierten Systemsicht auf, d.h. Beobachter beobachten das wohldefinierte äußere Programmverhalten ohne Zugriff auf Daten, welche aus Systemsicht als unzugänglich spezifiziert sind. Wir werden im folgenden Abschnitt zunächst eine Struktur zur Zusammenfügung aller äußerlich ersichtlichen Informationen im Kontext einer Beobachtung definieren. Darauf aufbauend werden wir den Begriff des Beobachters formal ausarbeiten.

4.1 Ausführungs-Kontexte

Primäres Ziel dieses Abschnitts stellt die Zusammenfassung aller äußerlich ersichtlichen Daten im Kontext einer Programmausführung, in eine eigens für diese Zwecke generierte Struktur "Ausführungs-Kontext", dar. Was wir also suchen ist ein theoretisches Maximum an Information, welche während einer Programmausführung von Außenstehenden beobachtet werden kann. Eine konkrete Programmausführung ist dabei nichts anderes als die Anwendung eines Programms auf einen zugehörigen Programmzustand, in Bezug auf eine spezifische Maschine. Auf Modell-Ebene sind Programmausführungen im Wesentlichen mit induzierten Teilsystem gleichzusetzen, wir wollen dies anhand eines Beispiels erläutern. Sei hierfür $\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>}$ ein beliebiges System. Programme können wir uns mit den zugehörigen Programmdaten gekapselt in den $\mathcal{T}$-Systemzuständen vorstellen. Konkrete Programmausführungen können somit in gewissem Sinne mit $\mathcal{T}$-Pfaden identifiziert werden. Genauer bedeutet dies für ein $s \in S^{(\mathcal{T})}$, dass jeder $\mathcal{T}$-Pfad in s eine (nicht

zwingend vollständige) Programmausführung des Programms $\mathfrak{p}(s)$, angewendet auf die in s entsprechend gekapselten Programmdaten, darstellt. Da $\mathcal{T}$ im Allgemeinen nicht als deterministisch vorausgesetzt werden kann, muss von der Existenz diverser solcher Pfade ausgegangen werden. Für eine Betrachtung in voller Allgemeinheit ist es somit erforderlich das komplette durch s induzierten $\mathcal{T}$-Teilsystem $\mathcal{T}(s)$ einzubeziehen. Zu beachten ist hierbei, dass s aufgrund möglicher zyklischer Strukturen in einem entsprechenden Teilsystem nicht zwingend eindeutig bestimmt ist. Was wir also benötigen ist eine Struktur mit Möglichkeiten zur Auszeichnung von Systemzuständen innerhalb von Transitionssystemen, welche sich für die Modellierung von Programmausführungen als geeignet erweist.

Definition 4.1.1: [Zustandsgraph]

Als *Zustandsgraph* bezeichnen wir Tupel der Form $(\mathcal{T},s_0)$ mit

- $\mathcal{T}$ ist beliebiges Transitionssystem
- $s_0 \in S^{(\mathcal{T})}$ ist beliebiger $\mathcal{T}$-Systemzustand

Wir bezeichnen $(\mathcal{T},s_0)$ dann auch konkret als $\mathcal{T}$*-Zustandsgraph (von* s_0*)* sowie s_0 als $(\mathcal{T},s_0)$*-Startzustand*.

Notation 4.1.2:

Wir schreiben auch

- $\mathcal{T}[s_0]$ für den $\mathcal{T}$-Zustandsgraph von s_0
- $\mathcal{T}[S] := \bigcup_{s\in S}\{\mathcal{T}[s]\}$ für die Menge der $\mathcal{T}$-Zustandsgraphen, induziert durch Systemzustände aus $S \subseteq S^{(\mathcal{T})}$
- $\mathcal{T}[*] := \mathcal{T}[S^{(\mathcal{T})}]$ für die Menge aller $\mathcal{T}$-Zustandsgraphen

Definition 4.1.3: [annotierter Zustandsgraph]

Als *annotierten Zustandsgraph* bezeichnen wir Tupel der Form $(\mathcal{T}_{<\mathfrak{a}>},s_0)$ mit

- $\mathcal{T}_{<\mathfrak{a}>}$ ist beliebiges annotiertes Transitionssystem
- $s_0 \in S^{(\mathcal{T})}$ ist beliebiger $\mathcal{T}$-Systemzustand

Wir bezeichnen $(\mathcal{T}_{<\mathfrak{a}>},s_0)$ dann auch konkret als $\mathcal{T}_{<\mathfrak{a}>}$*-Zustandsgraph (von* s_0*)* sowie s_0 als $(\mathcal{T}_{<\mathfrak{a}>},s_0)$*-Startzustand*.

$\mathcal{T}[a]$ $\mathcal{T}_{<\mathfrak{a}>[a]}$

Abbildung 4.1: Graphische Darstellung von $\mathcal{T}[a]$ und $\mathcal{T}_{<\mathfrak{a}>}[a]$ aus Beispiel 4.1.6.
Ein Pfeil ohne Quell-Knoten soll dabei auf einen entsprechenden Startzustand hinweisen.

Notation 4.1.4:

Wir schreiben auch

- $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ für den $\mathcal{T}_{<\mathfrak{a}>}$-Zustandsgraph von s_0.
- $\mathcal{T}_{<\mathfrak{a}>}[S] := \bigcup_{s \in S}\{\mathcal{T}_{<\mathfrak{a}>}[s]\}$ für die Menge der $\mathcal{T}_{<\mathfrak{a}>}$-Zustandsgraphen, induziert durch Systemzustände aus $S \subseteq S^{(\mathcal{T})}$
- $\mathcal{T}_{<\mathfrak{a}>}[*] := \mathcal{T}_{<\mathfrak{a}>}[S^{(\mathcal{T})}]$ für die Menge aller $\mathcal{T}_{<\mathfrak{a}>}$-Zustandsgraphen

Definition 4.1.5: [Modell-Klassen]

Wir bezeichnen mit ...

- $TS[*]$ die Klasse aller Zustandsgraphen
- $TS[*]$-A die Klasse aller annotierten Zustandsgraphen
- $TS\langle * \rangle$ die Klasse aller Zustandsgraphen für die $\mathcal{T}$ ein Baum ist
- $TS\langle * \rangle$-A die Klasse aller annotierten Zustandsgraphen für die $\mathcal{T}$ ein Baum ist

Beispiel 4.1.6:

Sei $\mathcal{T} = (\{a,b\},\{a \to b, b \to a\})$, dann ist $\mathcal{T}[a] = (\mathcal{T},a)$ ein Zustandsgraph. Sei weiter $\mathfrak{a}$ wie in Beispiel 3.2.9, dann ist $\mathcal{T}_{<\mathfrak{a}>}[a] = (\mathcal{T}_{<\mathfrak{a}>},a)$ ein annotierter Zustandsgraph. Wir wollen uns diese veranschaulichen wie in Abbildung 4.1.

Definition 4.1.7: [Programmausführung]

Sei $\mathcal{T}_{<\mathfrak{a}>}$ ein annotiertes Transitionssystem sowie $s_0 \in S^{(\mathcal{T})}$ beliebig. Als *(formalisierte) s_0-Programmausführung in $\mathcal{T}_{<\mathfrak{a}>}$* bezeichnen wir den annotierten $\mathcal{T}(s_0)_{<\mathfrak{a}>}$-Zustandsgraph

$$\mathcal{T}(s_0)_{<\mathfrak{a}>}[s_0]$$

Wir werden uns nun überlegen wie Programmausführungen nach außen hin ersichtlich werden. Hierfür sind zwei wesentliche Punkte zu beachten:

1. Zustandsgraphen werden von Beobachtern im Allgemeinen azyklisch wahrgenommen
2. Systemzustände sind für Beobachtern nicht direkt ersichtlich

Während der zweite Punkt bereits in Abschnitt 3.2 dadurch motiviert wurde, dass Systemzustände nach außen hin durch ihre spezifischen Ausgaben repräsentiert werden, bedarf der erste Punkt noch weiterer Erläuterungen. Hierfür mache man sich klar, dass aus externe Sicht jeder Programmschritt als "neu" erscheint, d.h. ohne Kenntnisse über die interne Systemstruktur ist es im Allgemeinen nicht möglich zyklische Strukturen zu rekonstruieren. Hierfür betrachte man Abbildung 4.2. Programmausführungen welche formal durch Zustandsgraphen der Form $\mathcal{T}(a)_{<\mathfrak{a}>}[a]$ bzw $\mathcal{T}'(a_0)_{<\mathfrak{a}'>}[a_0]$ charakterisiert sind, stellen aus externer Sicht eine unendliche Aneinanderreihung von "#"-Symbolen dar und sollten somit für beliebigen Beobachter ununterscheidbar sein.
Gesucht ist somit ein Konzept zur Auflösung von Zykel innerhalb von Zustandsgraphen. Zu diesem Zweck werden wir den Begriff der "Ausfaltung" definieren. Diese definieren in Abhängigkeit von einem Transitionssystem und einem zugehörigen Systemzustand das entsprechend induzierte Teilsystem in einer azyklisch Darstellung. Zykel werden hierbei durch das Hinzufügen gewisser Metainformationen aufgelöst. Diese befinden sich innerhalb der Systemzustände und beinhalten Informationen zur Historie der durchlaufenen Systemzuständen.

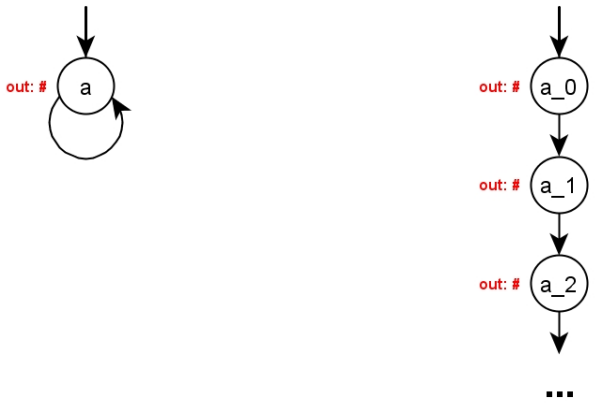
Definition 4.1.8: [Ausfaltung]

Sei $\mathcal{T}$ ein Transitionssystem sowie $s_0 \in S^{(\mathcal{T})}$ beliebig. Als *$\mathcal{T}$-Ausfaltung in s_0* bezeichnen wir das Transitionssystem $\mathcal{T}^{\downarrow s_0}$ mit

- $S^{(\mathcal{T}^{\downarrow s_0})} = \{(V,s) \in {}_{(s_0)}\mathcal{T}^{(*)} \times S^{(\mathcal{T})} \mid V.(s) \in {}_{(s_0)}\mathcal{T}^{(*)}\}$
- $T^{(\mathcal{T}^{\downarrow s_0})} = \{(V,s) \to (V',s') \mid V' = V.(s)\}$

Beispiel 4.1.9:

Sei $\mathcal{T} = (\{a\},\{a \to a\})$ ein Transitionssystem. Die $\mathcal{T}$-Ausfaltung in a ist dann



$\mathcal{T}(a)_{<a>}[a]$ $\mathcal{T}'(a_0)_{<a'>}[a_0]$

Abbildung 4.2: Modelle zweier Programmausführungen, welche für Beobachter ununterscheidbar sind

von folgender Form (vgl. hierzu auch Abbildung 4.3).

$$\mathcal{T}^{\downarrow_a} = (\ \{(\underbrace{(a,\ldots,a)}_{k\text{-mal}},a) \mid k \in \mathbb{N}\},\ \{(\underbrace{(a,\ldots,a)}_{k\text{-mal}},a) \to (\underbrace{(a,\ldots,a)}_{k'\text{-mal}},a) \mid k' = k+1\}\ )$$

Proposition 4.1.10:

Sei $\mathcal{T}$ ein Transitionssystem sowie $s_0 \in S^{(\mathcal{T})}$ beliebig, dann ist $\mathcal{T}^{\downarrow_{s_0}}$ ein Baum.

Beweis.
Hierfür müssen wir für $\mathcal{T}^{\downarrow_{s_0}}$ die Gültigkeit folgender Punkte nachweisen.

- [gewurzelt]
  Wir zeigen (ϵ,s_0) ist eine entsprechende Wurzel. Sei dafür $(V,s) \in S^{(\mathcal{T}^{\downarrow_{s_0}})}$ beliebig. Per Definition gilt $V.(s) \in {}_{(s_0)}\mathcal{T}^{(*)}$. Sei im Folgenden $\pi := V.(s)$ sowie $V_i := (\pi_0,\ldots,\pi_{i-1})$ für $i \in \{0,\ldots,|\pi|\}$. Es gilt

  $$\underbrace{(V_0,\pi_0)}_{=(\epsilon,s_0)} \to (V_1,\pi_1) \to \ldots \to (V_{|\pi|-2},\pi_{|\pi|-2}) \to \underbrace{(V_{|\pi|-1},\pi_{|\pi|-1})}_{=(V,s)} \in {}_{(*)}(\mathcal{T}^{\downarrow_{s_0}})^{(*)}$$

  und folglich auch die Behauptung. ✓

- [azyklisch]
  Seien π,π' zwei Pfade mit $\pi_0 = \pi'_0$ sowie $\pi_{|\pi|-1} = \pi'_{|\pi'|-1}$. Wir zeigen $\pi = \pi'$ woraus

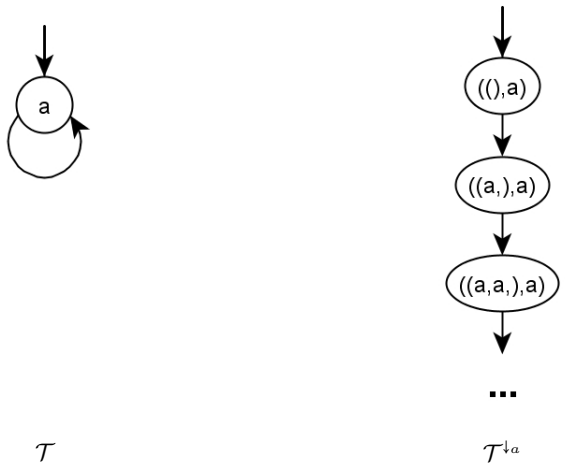


Abbildung 4.3: Graphische Darstellung der Zustandsgraphen aus Beispiel 4.1.9

sich anschließend die Behauptung ergibt. Der Fall $|\pi| = |\pi'| = 1$ ist trivial und wird somit nicht weiter betrachtet.

Wir zeigen die folgenden Punkten aus denen sich anschließend die Behauptung ergibt.

- [$|\pi| = |\pi'|$]
  Es gilt $\pi = ((V_0,s_0),\ldots,(V_k,s_k))$ für passende $(V_0,s_0),\ldots,(V_k,s_k) \in S^{(\mathcal{T}^{\downarrow_{s_0}})}$ und somit $|\pi| = |V_k| - |V_0| + 1$, denn $V_k = V_0.(s_0).(s_1).\ldots.(s_{k-1})$. Ebenso gilt aber $\pi' = ((V_0,s_0),\ldots,(V_k,s_k))$ und mit analoger Begründung $|\pi'| = |V_n| - |V_k| + 1$. ✓
- [$\pi_i = \pi'_i$ für alle $i \in \{0,\ldots,|\pi|-1\}$]
  Es reicht hierfür zu zeigen, dass $T^{(\mathcal{T}^{\downarrow_{s_0}})}$ linkseindeutig ist. Die Behauptung ergibt sich anschließend aufgrund von $\pi_{|\pi|-1} = \pi'_{|\pi'|-1}$.

  Seien also $(V_1,s_1) \to (V,s)$, $(V_2,s_2) \to (V,s) \in T^{(\mathcal{T}^{\downarrow_{s_0}})}$. Wir müssen zeigen $(V_1,s_1) = (V_2,s_2)$.

  Da $(V_1,s_1) \to (V,s) \in T^{(\mathcal{T}^{\downarrow_{s_0}})}$ gilt $V_1.(s_1) = V$. Ebenso folgt aber aufgrund von $(V_2,s_2) \to (V,s) \in T^{(\mathcal{T}^{\downarrow_{s_0}})}$, dass $V_2.(s_2) = V$. ✓

□

Proposition 4.1.11:

Sei $\mathcal{T}$ ein Transitionssystem sowie $\mathcal{T}^{\downarrow_{s_0}}$ die $\mathcal{T}$-Ausfaltung in einem beliebigen $\mathcal{T}$-Systemzustand s_0, dann ist (ϵ,s_0) die zugehörige Wurzel

Beweis.
Ergibt sich direkt aus dem Beweis von 4.1.10. □

Mit Hilfe der obigen Mittel wollen wir zunächst in einer Art Zwischenschritt "(annotierte) Zustandsbäume" definieren, welche wir als formalisierte Programmausführungen in einer azyklische Darstellung auffassen wollen.

Definition 4.1.12: [$\mathcal{T}$-Zustandsbaum]

Sei $\mathcal{T}$ ein Transitionssystem sowie $s_0 \in S^{(\mathcal{T})}$ beliebig. Als *$\mathcal{T}$-Zustandsbaum von s_0* bezeichnen wir den Zustandsgraphen $\mathcal{T}^{\downarrow s_0}[(\epsilon,s_0)]$

Notation 4.1.13:

Wir schreiben

- $\mathcal{T}\langle s_0\rangle$ für den $\mathcal{T}$-Zustandsbaum von s_0.
- $\mathcal{T}\langle S\rangle := \bigcup_{s\in S}\{\mathcal{T}\langle s\rangle\}$ für die Menge der $\mathcal{T}$-Zustandsbäume, induziert durch Systemzustände aus $S \subseteq S^{(\mathcal{T})}$
- $\mathcal{T}\langle *\rangle := \mathcal{T}\langle S^{(\mathcal{T})}\rangle$ für die Menge aller $\mathcal{T}$-Zustandsbäume

Definition 4.1.14: [Ausfaltungs-Annotation]

Sei $\mathcal{T}$ ein Transitionssystem sowie $\mathfrak{a}$ eine beliebige $\mathcal{T}$-Annotation. Die durch $\mathfrak{a}$ *induzierte Ausfaltungs-Annotation* $\mathfrak{a}^{\downarrow}$ ist definiert durch

- $\mathfrak{a}^{\downarrow}_S : ({}_{(*)}\mathcal{T}^{(*)} \times S^{(\mathcal{T})}) \to im(\mathfrak{a}_S)$ mit $(V,s) \mapsto \mathfrak{a}_S(s)$
- $\mathfrak{a}^{\downarrow}_T : ({}_{(*)}\mathcal{T}^{(*)} \times S^{(\mathcal{T})})^2 \rightsquigarrow im(\mathfrak{a}_T)$ mit $((V,s) \to (V',s')) \mapsto \mathfrak{a}_T(s \to s')$

Bemerkung 4.1.15:

Sei $\mathcal{T}_{<\mathfrak{a}>}$ ein beliebiges annotiertes Transitionssystem sowie $s_0 \in S^{(\mathcal{T})}$ beliebig, dann ist $\mathfrak{a}^{\downarrow}_T((V,s) \to (V',s'))$ definiert gdw. $s \to s' \in T^{(\mathcal{T})}$. Insbesondere ist $\mathfrak{a}^{\downarrow}((V,s) \to (V',s'))$ definiert für jedes $(V,s) \to (V',s') \in T^{(\mathcal{T}^{\downarrow s_0})}$.

Definition 4.1.16: [annotierter $\mathcal{T}$-Zustandsbaum]

Sei $\mathcal{T}_{<\mathfrak{a}>}$ ein annotiertes Transitionssystem sowie $s_0 \in S^{(\mathcal{T})}$ beliebig. Als *annotierten $\mathcal{T}$-Zustandsbaum in s_0* bezeichnen wir den annotierten Zustandsgraphen $\mathcal{T}^{\downarrow s_0}_{<\mathfrak{a}^{\downarrow}>}[(\epsilon,s_0)]$

Notation 4.1.17:

Wir schreiben

- $\mathcal{T}_{<\mathfrak{a}>}\langle s_0\rangle$ für den $\mathcal{T}_{<\mathfrak{a}>}$-Zustandsbaum von $s_0 \in S^{(\mathcal{T})}$.

- $\mathcal{T}_{<\mathfrak{a}>}\langle S\rangle := \bigcup_{s\in S}\{\mathcal{T}\langle s\rangle]\}$ für die Menge der $\mathcal{T}_{<\mathfrak{a}>}$-Zustandsbäume der Systemzustände in $S \subseteq S^{(\mathcal{T})}$
- $\mathcal{T}_{<\mathfrak{a}>}\langle *\rangle := \mathcal{T}_{<\mathfrak{a}>}[S^{(\mathcal{T})}]$ für die Menge aller $\mathcal{T}_{<\mathfrak{a}>}$-Zustandsbäume

Wir werden nun abschließend zeigen, dass formalisierte Programmausführungen und Zustandsbäume wechselseitig ineinander überführbar sind. Dies garantiert uns eine gewisse Form von Äquivalenz in Bezug auf den jeweiligen Informationsgehalt und soll zudem auf formaler Ebene rechtfertigen, dass wir Zustandsbäume als azyklische Darstellungsform formalisierter Programmausführungen betrachten werden.

Proposition 4.1.18:

Sei $\mathcal{T}_{<\mathfrak{a}>}$ ein annotiertes Transitionssystem, dann ist $\mathcal{T}_{<\mathfrak{a}>}\langle s_0\rangle$ aus $\mathcal{T}(s_0)_{<\mathfrak{a}>}[s_0]$ rekonstruierbar.

Beweis.
Per Definition gilt $\mathcal{T}(s_0)_{<\mathfrak{a}>}[s_0] = (\mathcal{T}(s)_{<\mathfrak{a}>},s)$ sowie $\mathcal{T}_{<\mathfrak{a}>}\langle s_0\rangle = (\mathcal{T}^{\downarrow s_0}_{<\mathfrak{a}\downarrow>},(\epsilon,s_0))$. Wir zeigen $S^{(\mathcal{T}^{\downarrow s_0})} = S^{(\mathcal{T}(s_0)^{\downarrow s_0})}$, anschließend ist die Behauptung offensichtlich.

Per Definition gilt $S^{(\mathcal{T}^{\downarrow s_0})} = \{(V,s_0) \mid V.(s_0) \in {}_{(s_0)}\mathcal{T}^{(*)}\}$ sowie $S^{(\mathcal{T}(s_0)^{\downarrow s_0})} = \{(V,s_0) \mid V.(s_0) \in {}_{(s_0)}\mathcal{T}(s_0)^{(*)}\}$. Der Beweis reduziert sich somit auf den Nachweis von ${}_{(s_0)}\mathcal{T}^{(*)} = {}_{(s_0)}\mathcal{T}(s_0)^{(*)}$. Wir zeigen " $\subseteq$ " und " $\supseteq$ " getrennt.

- [" $\subseteq$ "]
  Für jedes $\pi \in {}_{(s_0)}\mathcal{T}^{(*)}$ gilt $\pi_i \in reach_{\mathcal{T}}(s_0)$ und somit insbesondere $\pi \in {}_{(s_0)}\mathcal{T}(s_0)^{(*)}$. ✓
- [" $\supseteq$ "]
  Da $\mathcal{T}(s_0) \subseteq \mathcal{T}$ ist die Aussagen trivial, denn Pfade eines Teilsystems sind insbesondere Pfade des eigentlichen Systems. ✓

□

Proposition 4.1.19:

Sei $\mathcal{T}_{<\mathfrak{a}>}$ ein Transitionssystem, dann ist $\mathcal{T}(s_0)_{<\mathfrak{a}>}[s_0]$ aus $\mathcal{T}_{<\mathfrak{a}>}\langle s_0\rangle$ rekonstruierbar

Beweis.
Per Definition gilt $\mathcal{T}_{<\mathfrak{a}>}\langle s_0\rangle = (\mathcal{T}^{\downarrow s_0}_{<\mathfrak{a}^\downarrow>},(\epsilon,s_0))$ sowie $\mathcal{T}(s_0)_{<\mathfrak{a}>}[s_0] = (\mathcal{T}(s_0)_{<\mathfrak{a}>},s_0)$. Aus den folgenden drei Punkten ergibt sich die Behauptung.

- $[S^{(\mathcal{T}(s_0))} = \{s \mid \exists V : (V,s) \in S^{(\mathcal{T}^{\downarrow s_0})}\}]$

  Wir zeigen " $\subseteq$ ". Für " $\supseteq$ " gehe man die Schritte rückwärts durch.

  Per Definition gilt $S^{(\mathcal{T}(s_0))} = reach_{\mathcal{T}}(s_0)$. Sei also $s \in reach_{\mathcal{T}}(s_0)$ beliebig. Es existiert ein $\mathcal{T}$-Pfad der Form $\pi = (s_0,\ldots,s)$ und es gilt $((\underbrace{\pi_0}_{=s_0},\ldots,\pi_{|\pi|-2}),\underbrace{\pi_{|\pi|-1}}_{=s}) \in S^{(\mathcal{T}^{\downarrow s_0})}$. ✓

- $[T^{(\mathcal{T}(s_0))} = \{s \to s' \mid \exists V,V' : (V,s) \to (V',s') \in T^{(\mathcal{T}^{\downarrow s})}\}]$

  Wir zeigen " $\subseteq$ ". Für " $\supseteq$ " gehe man die Schritte rückwärts durch.

  Per Definition gilt $T^{(\mathcal{T}(s_0))} = \{s \to s' \in T^{(\mathcal{T})} \mid s,s' \in reach_{\mathcal{T}}(s_0)\}$. Sei also $s \to s' \in T^{(\mathcal{T}(s_0))}$ beliebig. Es existiert ein $\mathcal{T}$-Pfad der Form $\pi = (s_0,\ldots,s,s')$ und es gilt $((\underbrace{\pi_0}_{=s_0},\ldots,\pi_{|\pi|-3}),\underbrace{\pi_{|\pi|-2}}_{s}) \to ((\underbrace{\pi_0}_{=s_0},\ldots,\underbrace{\pi_{|\pi|-2}}_{=s}),\underbrace{\pi_{|\pi|-1}}_{s'}) \in T^{(\mathcal{T}^{\downarrow s})}$. ✓

- $[\mathfrak{a}_S(s) = \mathfrak{a}^\downarrow((\epsilon,s)),\ \mathfrak{a}_T(s \to s') = \mathfrak{a}^\downarrow((\epsilon,s) \to (\epsilon,s'))]$

  Die Behauptung ist mit Bemerkung 4.1.15 trivial. ✓

□

Bemerkung 4.1.20:

> Durch den Beweis der beiden obigen Propositionen könne die Vermutung aufkommen, die Betrachtung von Zustandsbäumen wäre unnötig, da diese sich bereits vollständig aus den formalisierten Programmausführungen ergeben. Man beachte jedoch, dass wir an dieser Stelle noch nicht von den Systemzuständen abstrahiert haben, d.h. obigen "Punkt 2" noch nicht umgesetzt haben. Erst im Anschluss erweist sich obiges Vorgehen als effektiv.

Wir werden nun abschließend in einem letzten Schritt annotierte Zustandsbäume an obigen "Punkt 2" adaptieren um final eine Modellierung von Programmausführungen aus Beobachter-Perspektive zu gewinnen.
Wie bereits angedeutet haben Beobachter lediglich die Möglichkeit über entsprechende Annotationen eine Sicht auf Systemzustände zu erlangen. Was wir somit benötigen ist eine Abstraktion annotierter Zustandsbäume, sodass Informationen durch zugehörige Systemzustände ausschließlich durch ihre entsprechenden Annotationen gegeben sind. Folgender Ansatz soll diesen Überlegungen gerecht werden. Sei dafür $\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>}$ ein System. Wir müssen für annotierten $\mathcal{T}$-Zustandsbäumen $\mathcal{T}_{<\mathfrak{a}>}\langle s_0\rangle$ und $\mathcal{T}_{<\mathfrak{a}>}\langle s_0'\rangle$ mit $s_0 =_{\mathfrak{p}} s_0'$ sowie

$\mathcal{T}_{<\mathfrak{a}>}\langle s_0\rangle \simeq \mathcal{T}_{<\mathfrak{a}>}\langle s_0'\rangle$ davon ausgehen, dass korrespondierende Programmausführungen aus Beobachter-Sicht stets ununterscheidbar sind, den beiden $\mathcal{T}$-Zustandsbäume repräsentieren dasselbe Programm und sind aufgrund der Isomorphie in ihrer äußeren Struktur identisch.
Dies motiviert uns $\mathcal{T}_{<\mathfrak{a}>}\langle s_0\rangle$ (genau wie $\mathcal{T}_{<\mathfrak{a}>}\langle s_0\rangle$) durch die Menge

$$[\mathcal{T}_{<\mathfrak{a}>}\langle s_0\rangle] \in \mathcal{T}_{<\mathfrak{a}>}\langle\{s' \in S^{(\mathcal{T})} \mid \mathfrak{p}(s') = \mathfrak{p}(s)\}\rangle/_{\simeq}$$

zu abstrahieren. Wir werden diesen Ansatz für die Modellierung von Programmausführungen aus Beobachter-Perspektive nutzen.

Definition 4.1.21: [externe Programmausführung]

Sei $\mathcal{T}_{<\mathfrak{a}>}$ ein annotiertes Transitionssystem sowie $s_0 \in S^{(\mathcal{T})}$ beliebig. Als *externe s_0-Programmausführung in $\mathcal{T}_{<\mathfrak{a}>}$* bezeichnen wir die Äquivalenzklasse

$$[\mathcal{T}_{<\mathfrak{a}>}\langle s_0\rangle] \in \mathcal{T}_{<\mathfrak{a}>}\langle\{s' \in S^{(\mathcal{T})} \mid \mathfrak{p}(s') = \mathfrak{p}(s)\}\rangle/_{\simeq}$$

Wir haben durch die externen Programmausführungen nun eine für uns vollständige Modellierung von Programmausführungen aus Beobachter-Sicht konstruiert, weshalb wir diese ohne Einschränkung mit diesen identifizieren dürfen. Da unser Interesse im Folgenden ausschließlich Programmausführungen aus Beobachter-Sicht gilt und uns formalisierte Programmausführungen in dieser Arbeit nicht wieder begegnen werden, werden wir externe Programmausführungen auch einfach als "Programmausführungen" bezeichnen.
Es ist uns nun möglich die Struktur der "Ausführungs-Kontext" sinnvoll zu definieren. Elemente dieser spezifizieren wir als theoretisches Maximum äußerlich ersichtlicher Informationen im Kontext einer Programmausführung.

Definition 4.1.22: [Ausführungs-Kontext]

Als *Ausführungs-Kontext* bezeichnen wir Elemente der Klasse

$$\mathfrak{AK} := \{\ (\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\ p,\ [\mathfrak{b}]) \mid \mathcal{T}_{<\mathfrak{a},\mathfrak{p}>} \in TS\text{-}A\text{-}P \wedge p \in im(\mathfrak{p}) \wedge [\mathfrak{b}] \in \mathcal{T}_{<\mathfrak{a}>}\langle\{s' \in S^{(\mathcal{T})} \mid \mathfrak{p}(s') = p\}\rangle/_{\simeq}\ \}$$

Bemerkung 4.1.23:

Schreibt man Ausführungs-Kontexte in der Form $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\ p,\ [\mathcal{G}_{<\mathfrak{a}^{\downarrow}>}[s_0]])$ ist klar, dass $s_0 = (\epsilon, s)$ für passendes $s \in S^{\mathcal{T}}$ sowie $\mathcal{G} = \mathcal{T}^{\downarrow s_0}$. Durch diese Form der Notation kann man geschickt von der, für gewisse Zwecke "umständlichen", Struktur von $\mathcal{T}^{\downarrow s_0}$ abstrahieren.

Abschließend werden wir für spätere Zwecke einige Grundbegriffe aus Kapitel 3 auf Zustandsgraphen übertragen.

Definition 4.1.24: [Pfad]

Sei $\mathcal{T}[s_0]$ ein Zustandsgraph. Als $\mathcal{T}[s_0]$-*Pfad* bezeichnen wir $\mathcal{T}$-Pfade der Form

$$(s_0, \dots) \in {}_{(s_0)}\mathcal{T}^{(*)} \cup {}_{(s_0)}\mathcal{T}^{(\omega)}$$

Definition 4.1.25: [spezielle Pfade]

Sei $\mathcal{T}[s_0]$ ein Zustandsgraph. Sei weiter $\bullet \in \mathbb{N} \cup \{*, \omega\}$. Wir definieren

$$(\mathcal{T}[s_0])^{(\bullet)} := \{\pi \text{ ist } \mathcal{T}[s_0]\text{-Pfad} \mid |\pi| = \begin{cases} n & : \bullet = n \in \mathbb{N} \\ n \in \mathbb{N} \text{ beliebig} & : \bullet = * \\ \omega & : \bullet = \omega \end{cases} \}$$

Wir bezeichnen $\pi \in (\mathcal{T}[s_0])^{(\bullet)}$ als

- *endlichen $\mathcal{T}[s_0]$-Pfad (der Länge n)*, falls $\bullet = n \in \mathbb{N}$
- *endlichen $\mathcal{T}[s_0]$-Pfad*, falls $\bullet = *$
- *unendlichen $\mathcal{T}[s_0]$-Pfad*, falls $\bullet = \omega$

Definition 4.1.26: [Isomorphie]

Wir bezeichnen zwei Zustandsgraphen $\mathcal{T}[s_0]$ und $\mathcal{T}'[s_0']$ als *isomorph* gdw. ein Isomorphismus f zwischen $\mathcal{T}$ und $\mathcal{T}'$ existiert mit

$$f(s_0) = f(s_0')$$

Wir bezeichnen f dann als *Isomorphismus zwischen $\mathcal{T}[s_0]$ und $\mathcal{T}'[s_0']$* und schreiben

$$\mathcal{T}[s_0] \simeq \mathcal{T}'[s_0']$$

Hierbei interpretieren wir $\simeq$ als relationale Klasse.

Bemerkung 4.1.27:

Ist f ein Isomorphismus zwischen beliebigen Zustandsgraphen $\mathcal{T}[s_0]$ und $\mathcal{T}'[s_0']$, dann ist f^{-1} ein Isomorphismus zwischen $\mathcal{T}'[s_0']$ und $\mathcal{T}[s_0]$. Insbesondere gilt

$$\mathcal{T}[s_0] \simeq \mathcal{T}'[s_0'] \text{ gdw. } \mathcal{T}'[s_0'] \simeq \mathcal{T}[s_0]$$

Bemerkung 4.1.28:

Für isomorphe Zustandsgraphen $\mathcal{T}[s_0]$ und $\mathcal{T}'[s'_0]$ ist klar, dass auch $\mathcal{T}[s] \simeq \mathcal{T}'[f(s)]$ für beliebiges $s \in S^{(\mathcal{T})}$.

Definition 4.1.29: [Isomorphie]

Wir bezeichnen zwei annotierte Zustandsgraphen $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ als *isomorph* gdw. ein Isomorphismus f zwischen $\mathcal{T}_{<\mathfrak{a}>}$ und $\mathcal{T}'_{<\mathfrak{a}'>}$ existiert mit

$$f(s_0) = f(s'_0)$$

Wir bezeichnen f dann als *Isomorphismus zwischen* $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ *und* $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ und schreiben

$$\mathcal{T}_{<\mathfrak{a}>}[s_0] \simeq \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$$

Bemerkung 4.1.30:

Ist f ein Isomorphismus zwischen beliebigen Zustandsgraphen $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$, dann ist f^{-1} ein Isomorphismus zwischen $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ und $\mathcal{T}_{<\mathfrak{a}>}[s_0]$. Insbesondere gilt

$$\mathcal{T}[s_0]_{<\mathfrak{a}>} \simeq \mathcal{T}'_{<\mathfrak{a}'>}[s'_0] \text{ gdw. } \mathcal{T}'_{<\mathfrak{a}'>}[s'_0] \simeq \mathcal{T}_{<\mathfrak{a}>}[s_0]$$

Bemerkung 4.1.31:

Für isomorphe annotierte Zustandsgraphen $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ ist klar, dass auch $\mathcal{T}_{<\mathfrak{a}>}[s] \simeq \mathcal{T}'_{<\mathfrak{a}'>}[f(s)]$ für beliebiges $s \in S^{(\mathcal{T})}$.

4.2 Formale Beobachter

In diesem Abschnitt werden wir aufbauend auf der Struktur der Ausführungs-Kontexte Beobachter formal definieren. Anschließend werden wir diese anhand von Beispielen anschaulich erläutern.

Definition 4.2.1: [Beobachter]

Als *Beobachter* bezeichnen wir eine funktionale Klasse $\mathcal{B}$ mit

$$dom(\mathcal{B}) = \mathfrak{AK}$$

Wir wollen uns Beobachter als Entitäten vorstellen, welche uns zu einem gegeben Ausführungs-Kontext ihre spezifische Beobachtung mitteilen. Hierbei beschreibt das Bild $im(\mathcal{B})$ eines Beobachters $\mathcal{B}$ gerade eine Kollektion möglicher Beobachtungen von $\mathcal{B}$. In welcher Darstellungsform Beobachtungen dabei vorliegen ist vollkommen willkürlich. Wir versprechen uns hierdurch maximale Abstraktion, welche uns erlaubt Beobachtungen stets in einer natürlichen Form zu erfassen.
Im Folgenden werden wir beispielhaft einige Beobachter vorstellen. In Kapitel 7 werden wir anschließend einen beispielhaften Formalismus zur strukturierten Konstruktion von Beobachtern kennenlernen.

Bemerkung 4.2.2:

Natürlich hätten wir Beobachter auch als funktionale Klassen definieren können mit $dom(\mathcal{B}) = \mathfrak{AK}'$, mit

$$\mathfrak{AK}' = \{(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\ p,\ \mathcal{T}(s_0)_{<\mathfrak{a}>}[s_0]) \mid s_0 \in S^{(\mathcal{T})}\}$$

d.h. die Elemente aus $\mathfrak{AK}'$ sind bzgl. formalisierte Programmausführungen definiert. Dies hätte jedoch zum Nachteil, dass bei der Beobachter-Definition die externe Programmausführung (bzw. eine äquivalente Struktur) in der Regel explizit konstruiert werden müssten, da Beobachtungen auf dieser Sicht aufsetzten. Andernfalls müssten Argumente erbracht werden, warum die definierten Beobachter trotz dessen intuitiv operieren. Durch unser Vorgehen haben wir den Vorteil dies an zentrale Stelle bewerkstelligt zu haben, um so das weitere Vorgehen zu erleichtern. Nichts desto trotz ist die kommende Theorie, falls gewünscht, mit Leichtigkeit auf Ausführungs-Kontexte mit Kapselung interner Programmausführungen verallgemeinerbar.

Beispiel 4.2.3:

Wir definieren den Beobachter ...

- $Z\text{-}pTrace$ durch
  $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\ p,\ [\mathcal{G}_{<\mathfrak{a}^{\downarrow}>}[s_0]]) \mapsto \{\ (a_0, a_1, \dots) \mid \exists \pi \in (\mathcal{G}[s_0])^{(*)}\ :\ a_i = \mathfrak{a}_S^{\downarrow}(\pi_i)\}$
- Ü-$pTrace$ durch
  $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\ p,\ [\mathcal{G}_{<\mathfrak{a}^{\downarrow}>}[s_0]]) \mapsto \{\ (b_0, b_1, \dots) \mid \exists \pi \in (\mathcal{G}[s_0])^{(*)}\ :\ b_i = \mathfrak{a}_T^{\downarrow}(\pi_i \to \pi_{i+1})\}$
- $pTrace$ durch
  $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\ p,\ [\mathcal{G}_{<\mathfrak{a}^{\downarrow}>}[s_0]]) \mapsto \{\ (a_0, b_0, a_1, b_1, \dots) \mid \exists \pi \in (\mathcal{G}[s_0])^{(*)}\ :\ a_i = \mathfrak{a}_S^{\downarrow}(\pi_i),\ b_i = \mathfrak{a}_T^{\downarrow}(\pi_i \to \pi_{i+1})\}$

Bemerkung 4.2.4:

Die Beobachter aus Definition 4.2.3 sind wohldefiniert.

Beweis.
Wir beschränken uns auf den Beweis der Wohldefiniertheit des Beobachters $pTrace$. Für $Z\text{-}pTrace$ und Ü-$pTrace$ verläuft der Beweis analog.
Hierfür zeigen wir für beliebige Zustandsgraphen $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ mit $\mathcal{T}_{<\mathfrak{a}>}[s_0] \simeq \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$, dass folgende Mengen identisch sind.

- $M := \{\ (a_0, b_0, a_1, b_1, \dots) \mid \exists \pi \in (\mathcal{T}[s_0])^{(*)}\ :\ a_i = \mathfrak{a}_S(\pi_i),\ b_i = \mathfrak{a}_T(\pi_i \to \pi_{i+1})\}$
- $M' := \{\ (a_0, b_0, a_1, b_1, \dots) \mid \exists \pi \in (\mathcal{T}'[s'_0])^{(*)}\ :\ a_i = \mathfrak{a}'_S(\pi_i),\ b_i = \mathfrak{a}'_T(\pi_i \to \pi_{i+1})\}$

Anschließend ergibt sich die Behauptung, denn für beliebiges $(\mathcal{S},\ p,\ [\mathfrak{b}]) \in \mathfrak{AR}$ folgt für $\mathfrak{b}', \mathfrak{b}'' \in [\mathfrak{b}]$ per Definition, dass $\mathfrak{b}' \simeq \mathfrak{b}''$.
Wir zeigen die Inklusionen $\subseteq$ und $\supseteq$ getrennt.

- " $\subseteq$ "

  Zunächst einmal folgt aus der Isomorphie der Zustandsgraphen die Existenz eines entsprechenden Isomorphismus $f : S^{(\mathcal{T})} \to S^{(\mathcal{T}')}$. Sei nun $(a_0, b_0, a_1, b_1, \dots) \in M$. Es folgt die Existenz eines $(\pi_i)_{i\in I} \in (\mathcal{T}[s_0])^{(*)}$ mit $a_i = \mathfrak{a}_S(\pi_i)$ sowie $b_i = \mathfrak{a}_T(\pi_i \to \pi_{i+1})$. Hieraus ergibt sich aber für $(f(\pi_i))_{i\in I} \in (\mathcal{T}'[s'_0])^{(*)}$, dass $a_i = \mathfrak{a}'_S(f(\pi_i))$ sowie $b_i = \mathfrak{a}'_T(f(\pi_i) \to f(\pi_{i+1}))$ und es folgt $(a_0, b_0, a_1, b_1, \dots) \in M'$. ✓

- " $\supseteq$ "

  Folgt analog. ✓

□

Beispiel 4.2.5:

Sei $\mathcal{T}_{<(id,\tau),id>}$ ein System mit $\mathcal{T} = (\mathbb{N},\ \{n \to n+1\}_{n\in\mathbb{N}}$. Sei weiter $\mathfrak{k} := (\mathcal{T}_{<(id,\tau),id>}, 1, \mathcal{T}_{<(id,\tau)>}\langle 1\rangle) \in \mathfrak{AR}$, dann gilt

- $Z\text{-}pTrace(\mathfrak{k}) = \{(1,\dots,n)\}_{n\in\mathbb{N}} = \{(),(1),(1,2),\dots\}$
- $Z\text{-}pTrace(\mathfrak{k}) = \{\underbrace{(\tau,\dots,\tau)}_{n\text{-mal}}\}_{n\in\mathbb{N}} = \{(),(\tau),(\tau,\tau),\dots\}$
- $pTrace(\mathfrak{k}) = \{(1,\tau,2,\tau,\dots,\tau,n)\}_{n\in\mathbb{N}} = \{(),(1),(1,\tau,2),\dots\}$

Wir werden uns abschließend kurze Angreifer-Szenarien für die jeweiligen Beobachter überlegen.

Z-pTrace Der *Z-pTrace*-Beobachter ist dadurch charakterisiert, dass er Abfolgen von Zustands-Ausgaben einer Maschine sequentiell wahrnimmt. Als Beispiel stelle man sich einen handelsüblichen Computer mit zwei zugehörigen Bildschirmen M_1 und M_2 vor. Bildschirm M_1 soll dabei die Zustands-Ausgaben repräsentieren, Bildschirm M_2 entsprechend die Übergangs-Ausgaben. Den *Z-pTrace* Beobachter kann man sich nun als Angreifer vorstellen, der während der gesamten Programmausführung die sequentielle Folge von Ausgaben auf Bildschirm M_1 wahrnimmt.

Ü-*pTrace* Der Ü-*pTrace*-Beobachter entspricht im Wesentlichen dem *Z-pTrace*-Beobachter, dieser beobachtet jedoch lediglich Übergangs-Ausgaben. Übertragen auf obiges Beispiel baut dieser seine Beobachtungen anhand der sequentiellen Folge von Ausgaben auf Bildschirm M_2 auf.

pTrace Der *pTrace*-Beobachter stellt eine Zusammenfassung der beiden obigen Beobachter *Z-pTrace* und Ü-*pTrace* dar und beobachtet simultan die sequentielle Abfolgen sowohl von Zustands- als auch Übergangs-Ausgaben. Übertragen auf obiges Beispiel baut dieser seine Beobachtungen parallel anhand der alternierenden Folgen von Ausgaben, gegeben durch die Bildschirme M_1 und M_2, auf.

KAPITEL 5

Noninterferenz

Ziel dieses Kapitels wird die Definition eines allgemeinen Noninterferenz Begriffs darstellen. Hierzu bauen wir auf den bisher vorgestellten Grundlagen auf. Wir beginnen zunächst mit einer formalen Definition von Ununterscheidbarkeit. Aufbauend definieren wir sowohl einen Noninterferenz-Begriff für Programme als auch für Systeme.

5.1 Ununterscheidbarkeit

Ziel dieses Abschnitts stellt die formale Erfassung des Ununterscheidbarkeit-Begriffs dar. Dieser wird in Abhängigkeit von einem konkreten Beobachter definiert und beschreibt die Kollektion aller Mengen von Ausführungs-Kontexten, welche aus Sicht des entsprechenden Beobachters zu identischer Beobachtung führen.

Definition 5.1.1: [Ununterscheidbarkeit]

Sei $\mathcal{B}$ ein Beobachter sowie $M \subseteq \mathfrak{K}$ eine Menge von Ausführungs-Kontexten.
Wir bezeichnen M als $\mathcal{B}$-*Ununterscheidbar* gdw.

$$\mathcal{B}(M) = const$$

Notation 5.1.2:

Wir schreiben auch

- $\mathfrak{U}_{<\mathcal{B}>}$ für die Klasse aller $\mathcal{B}$-ununterscheidbaren Mengen

- $\mathfrak{U}_{<\mathcal{B},\mathcal{K}>}$, mit $\mathcal{K} \subseteq \mathfrak{M}$, für die Klasse aller $\mathcal{B}$-ununterscheidbaren Mengen $M \subseteq \mathcal{K}$

Bemerkung 5.1.3:

Die Notationen aus 5.1.2 dienen der besseren formalen Handhabbarkeit des Ununterscheidbarkeit-Begriffs. Erstere ist dabei direkt an die Definition angelehnt. Die zweite dient in erster Linie späteren Zwecken und schränkt Ununterscheidbarkeit auf eine Teilklasse der Ausführungs-Kontexte ein.

Bemerkung 5.1.4:

- Für jedes $\mathcal{K} \subseteq \mathfrak{M}$ gilt $\mathfrak{U}_{<\mathcal{B},\mathcal{K}>} \subseteq \mathfrak{U}_{<\mathcal{B}>}$, gleichheit gilt für $\mathcal{K} = \mathfrak{M}$.
- Für jedes $\mathcal{K}' \subseteq \mathcal{K}$ gilt $\mathfrak{U}_{<\mathcal{B},\mathcal{K}'>} \subseteq \mathfrak{U}_{<\mathcal{B},\mathcal{K}>}$

Proposition 5.1.5:

Es gilt

(i) $M \in \mathfrak{U}_{<\mathcal{B},\mathcal{K}>}$ gdw. $\forall \mathfrak{k},\mathfrak{k}' \in M : \ \mathfrak{k} =_{\mathcal{B}|_{\mathcal{K}}} \mathfrak{k}'$

(ii) $M \in \mathfrak{U}_{<\mathcal{B}>}$ gdw. $\forall \mathfrak{k},\mathfrak{k}' \in M : \ \mathfrak{k} =_{\mathcal{B}} \mathfrak{k}'$

Beweis.
Wir zeigen nur (i), anschließend ergibt sich (ii) für $\mathcal{K} := \mathfrak{M}$ aufgrund von Bemerkung 5.1.4 zusammen mit $\mathcal{B}|_{\mathfrak{M}} = \mathcal{B}$.
Wir zeigen die Implikationen " $\Rightarrow$ " und " $\Leftarrow$ " getrennt.

- [" $\Rightarrow$ "]

  Seien $\mathfrak{k},\mathfrak{k}' \in M$. Laut Voraussetzung gilt $M \in \mathfrak{U}_{<\mathcal{B},\mathcal{K}>}$, d.h. $\mathcal{B}|_{\mathcal{K}}(M) = c$ für eine passende Konstante c. Dies wiederum bedeutet, dass sowohl $\mathcal{B}|_{\mathcal{K}}(s) = c$ als auch $\mathcal{B}|_{\mathcal{K}}(s') = c$ und somit insbesondere $\mathfrak{k} =_{\mathcal{B}|_{\mathcal{K}}} \mathfrak{k}'$. ✓

- [" $\Leftarrow$ "]

  Angenommen $M \notin \mathfrak{U}_{<\mathcal{B},\mathcal{K}>}$, dann gibt es $\mathfrak{k},\mathfrak{k}' \in M$ mit $\mathfrak{k} \neq_{\mathcal{B}|_{\mathcal{K}}} \mathfrak{k}'$ im Widerspruch zur Voraussetzung, dass $\forall \mathfrak{k},\mathfrak{k}' \in M : \ \mathfrak{k} =_{\mathcal{B}|_{\mathcal{K}}} \mathfrak{k}'$. ↯

□

5.2 Programm-Noninterferenz

Ziel dieses Abschnitts stellt die Definition eines Noninterferenz-Begriffs für Programm dar. Wir werden diesen als eine Form von Ununterscheidbarkeit definieren. In Anlehnung an unsere bisherigen Erkenntnisse bietet es sich hierbei an Noninterferenz generisch bzgl. System und Beobachter zu erfassen. Zusätzlich wollen wir die allgemein übliche Bedingung auflockern, dass Noninterferenz eine Ununterscheidbarkeit von Programmenausführungen bzgl. beliebiger äußerlich ununterscheidbarer Programmzustände zusichern sollte. Wir werden hierfür den Begriff der Programm-Startzustandsmenge definieren und diesen als zusätzlichen variablen Parameter mit in eine entsprechende Definition von Programm-Noninterferenz einbringen.

Definition 5.2.1: [Startzustand]

Sei $\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>}$ ein System sowie $p \in im(p)$ ein Programm. Sei weiter

$$S_p := \{s \in S^{(\mathcal{T})} \mid \mathfrak{p}(s) = p\}$$

Wir bezeichnen die Elemente aus S_p als *p-Startzustände in* $\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>}$. Teilmengen von S_p bezeichnen wir als *p-Startzustandsmengen in* $\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>}$.

Bemerkung 5.2.2:

Man verwechsele Startzustände aus Definition 5.2.1 nicht mit Startzuständen aus Definition 4.1.1. Diese sind in Definition 5.2.1 bzgl. Programm definiert, in Definition 4.1.1 jedoch bzgl. Zustandsgraphen.

Im Folgenden wollen wir Programm-Noninterfernz durch Noninterferenz-Spezifikationen ausdrücken. Eine Noninterferenz-Spezifikation beschreibt dabei für ein festes Programm eine konkrete Form von Sicherheit in einem konkreten Umfeld. Technisch kann man sich Noninterferenz-Spezifikationen als Zusammenfassung aller variablen Größen für eine konkrete Noninterferenz-Aussage vorstellen.

Definition 5.2.3: [Noninterferenz-Spezifikation]

Als *p-Noninterferenz-Spezifikation* bezeichnen wir Tripel der Form $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B})$ mit

- $\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>}$ ist ein System
- $p \in im(\mathfrak{p})$ ein Programm
- $\mathcal{C} = \{\Delta_i\}_{i \in I}$ eine Menge von p-Startzustandsmengen
- $\mathcal{B}$ ein Beobachter

Definition 5.2.4: [Erfüllbarkeit]

Sei $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B})$ eine p-Noninterferenz-Spezifikation. Wir bezeichnen $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B})$ als *erfüllt* gdw.

$$\forall \Delta \in \mathcal{C}: \ \{(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\ p,\ [\mathcal{T}_{<\mathfrak{a}>}\langle s\rangle]) \in \mathfrak{RK} \mid s \in \Delta\ \} \in \mathfrak{U}_{<\mathcal{B}>}$$

Definition 5.2.5: [Programm-Noninterferenz]

Sei $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B})$ eine erfüllte p-Noninterferenz-Spezifikation. Wir bezeichnen p dann als $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B})$-*noninterferent* und schreiben

$$p \in \mathfrak{N}_{<(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B})>}$$

Bemerkung 5.2.6:

Wir interpretieren $\mathfrak{N}_{<(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B})>}$ als 1-elementige (Indikatormenge) für die

$$p \in \mathfrak{N}_{<(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B})>} \text{ gdw. } p \text{ ist } (\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B})\text{-noninterferent}$$

Wie bereits erwähnt wird Ununterscheidbarkeit in der Regel für alle Startzustände mit übereinstimmender äußerer Erscheinung gefordert. Wir wollen dies im Folgenden ebenfalls berücksichtigen.

Definition 5.2.7: [kanonische Noninterferenz-Spezifikation]

Wir bezeichnen eine p-Noninterferenz-Spezifikation $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B})$ als *kanonisch* gdw.

$$\mathcal{C} = \{s \in S^{(\mathcal{T})} \mid \mathfrak{p}(s) = p\}/_{(=_{\mathfrak{a}_S})}$$

Definition 5.2.8: [kanonische Programm-Noninterferenz]

Sei $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B})$ eine erfüllte kanonische p-Noninterferenz-Spezifikation. Wir bezeichnen p dann als *(kanonisch)* $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{B})$-*noninterferent* und schreiben

$$p \in \mathfrak{N}_{<(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{B})>}$$

Bemerkung 5.2.9:

Wir interpretieren $\mathfrak{N}_{<(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{B})>}$ als Menge aller (kanonisch) $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{B})$-noninterferenten Programme.

5.3 System-Noninterferenz

Ziel dieses Abschnitts stellt die Definition von Noninterferenz auf System-Ebene dar. Diese spielt für unsere Betrachtungen eine eher untergeordnete Rolle, von daher ist dieser Abschnitt eher beiläufig zu betrachten und dient mehr der Vollständigkeit. System-Noninterferenz ist dabei vollkommen unabhängig von einem Programmbegriff definierbar. Aufgrund dessen basieren unsere folgenden Betrachtung auf dem Modell annotierter Transitionssysteme. Da jedes System eine annotiertes Transitionssystem als Teilstruktur kapselt lassen sich die kommenden Überlegungen auf natürliche Weise auch auf diese übertragen. Ein annotiertes Transitionssystem soll dabei in Abhängigkeit von einem Beobachter und einer Menge von Mengen von entsprechenden Systemzuständen noninterferent sein, falls entsprechende Ausführungs-Kontexte ununterscheidbar sind.

Definition 5.3.1: [System-Noninterferenz]

Sei $\mathcal{T}_{<\mathfrak{a}>}$ ein annotiertes Transitionssystem sowie $\mathcal{B}$ eine Beobachter. Sei weiter $\mathcal{C} \subseteq \mathscr{P}(S^{(\mathcal{T})})$ eine Menge von Mengen von $\mathcal{T}$-Systemzuständen. Wir bezeichnen $\mathcal{T}_{<\mathfrak{a}>}$ als $(\mathcal{B},\mathcal{C})$-*noninterferent* gdw.

$$\forall \Delta \in \mathcal{C} : \ \{(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>}, p, [\mathcal{T}_{<\mathfrak{a}>}\langle s\rangle]) \in \mathfrak{IR} \mid s \in \Delta\} \in \mathfrak{U}_{<\mathcal{B}>}$$

Analog zur kanonischen Programm-Noninterferenz wollen wir kanonische System-Noninterferenz definieren. Ein annotiertes Transitionssystem soll dabei kanonisch noninterferent sein falls entsprechende Systemzustände mit selber äußerer Erscheinung ununterscheidbare Ausführungs-Kontexte induzieren. Es ergibt sich anschließend direkt aus der Definition, dass Programme kanonisch noninterferenter Systeme kanonisch noninterferent sind.

Definition 5.3.2: [kanonische System-Noninterferenz]

Sei $\mathcal{T}_{<\mathfrak{a}>}$ ein annotiertes Transitionssystem sowie $\mathcal{B}$ eine Beobachter. Wir bezeichnen $\mathcal{T}_{<\mathfrak{a}>}$ als *(kanonisch)* $\mathcal{B}$*-noninterferent* gdw.

$$\mathcal{T}_{<\mathfrak{a}>} \text{ ist } (\mathcal{B}, S^{(\mathcal{T})}/_{(=_{\mathfrak{a}_S})}) - \text{noninterferent}$$

Proposition 5.3.3:

Sei $\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>}$ ein System für das $\mathcal{T}_{<\mathfrak{a}>}$ kanonisch $\mathcal{B}$-noninterferent ist, dann ist jedes $p \in im(\mathfrak{p})$ kanonisch $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{B})$-noninterferent.

Beweis.
Sei $p \in im(\mathfrak{p})$ beliebig. Wir müssen zeigen, dass die kanonische p-Noninterferenz-Spezifikation $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C}_{kan},\mathcal{B})$ erfüllt ist.

Sei hierfür $\Delta \in \mathcal{C}_{kan}$ beliebig. Per Definition gilt $\mathfrak{a}_S(\Delta) = const$ und folglich existiert ein $\Delta' \in S^{(\mathcal{T})}/_{(=_{\mathfrak{a}_S})}$ mit $\Delta \subseteq \Delta'$. Laut Voraussetzung gilt nun

$$\{(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>}, p, [\mathcal{T}_{<\mathfrak{a}>}\langle s\rangle]) \in \mathfrak{W} \mid s \in \Delta'\} \in \mathfrak{U}_{<\mathcal{B}>}$$

und somit insbesondere

$$\{(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>}, p, [\mathcal{T}_{<\mathfrak{a}>}\langle s\rangle]) \in \mathfrak{W} \mid s \in \Delta\} \in \mathfrak{U}_{<\mathcal{B}>}$$

woraus sich direkt die Behauptung ergibt.

□

Wir möchten an dieser Stelle die Überlegungen zur System-Noninterferenz beenden. Im Folgenden sei, wenn von Noninterferenz die Rede ist, stets die Programm-Noninterferenz gemeint.

KAPITEL 6

Hierarchisierungen

Ziel dieses Kapitels stellt die Definition natürlicher Ordnungsstrukturen auf Beobachtern und Annotationen dar, welche sich in natürlicher Form auf unseren Noninterferenz-Begriff übertragen lassen. Wir möchten beispielsweise eine Vorstellung davon bekommen was es für einen Beobachter heißt stärker zu sein als ein anderer und erwartet parallel dazu die Verträglichkeit mit der Noninterferenz, d.h. eine erfüllte Noninterferenz-Spezifikation soll bei Übergang zu einem schwächen Beobachter weiterhin erfüllt bleiben. Andererseits sind wir an selbigem für Annotationen interessiert. Da wir Annotationen mit Benutzersichten identifizieren, wären wir so in der Lage nachgewiesene Sicherheiten bzgl. einer höheren Sicht auf tiefere Ebenen zu übertragen.
Wir werden uns im folgenden Abschnitt zunächst mit Beobachtern und entsprechenden Ordnungsstrukturen beschäftigen. Anschließend betrachten wir Ordnungsstrukturen auf Annotationen. In beiden Fällen steht die Erarbeitung der Zusammenhänge zur Noninterferenz-Sicherheit im Fokus unserer Betrachtungen.
Bevor wir beginnen werden wir eine allgemeine Ordnung auf Noninterferenz-Spezifikationen definieren, welche als Basis für Aussagen über Ordnungserhaltung bzgl. Beobachter- und Annotationsordnungen dienen soll.

Definition 6.0.4: [Ordnung - Noninterferenz-Spezifikationen]

Wir definieren eine relationale Ordnungsklasse $\preceq$ auf der Klasse der Noninterferenz-Spezifikationen durch

$$\mathfrak{S} \preceq \mathfrak{S}' \text{ gdw. } \mathfrak{N}_{<\mathfrak{S}'>} \subseteq \mathfrak{N}_{<\mathfrak{S}>}$$

Bemerkung 6.0.5:

Für zwei Noninterferenz-Spezifikationen $\mathfrak{S}$ und $\mathfrak{S}'$ mit $\mathfrak{S} \preceq \mathfrak{S}'$ stellt die von $\mathfrak{S}'$ implizierte Sicherheit eine “stärkere“ dar als die diese von $\mathfrak{S}$.

Proposition 6.0.6:

$\preceq$ ist eine Präordnung

Beweis.
Folgende Eigenschaften müssen $\preceq$ nachgewiesen werden.

- [Reflexivität]

  Sei $\mathfrak{S}$ eine beliebige Noninterferenz-Spezifikation. Wir müssen zeigen $\mathfrak{N}_{<\mathfrak{S}>} \subseteq \mathfrak{N}_{<\mathfrak{S}>}$. Dies ist offensichtlich. ✓

- [Transitivität]

  Seien $\mathfrak{S}$,$\mathfrak{S}'$,$\mathfrak{S}''$ beliebige Noninterferenz-Spezifikation mit $\mathfrak{S} \preceq \mathfrak{S}'$ sowie $\mathfrak{S}' \preceq \mathfrak{S}''$. Wir müssen zeigen, dass $\mathfrak{S} \preceq \mathfrak{S}''$.
  Aus $\mathfrak{S} \preceq \mathfrak{S}'$ und $\mathfrak{S}' \preceq \mathfrak{S}''$ folgt $\mathfrak{N}_{<\mathfrak{S}'>} \subseteq \mathfrak{N}_{<\mathfrak{S}>}$ genau wie $\mathfrak{N}_{<\mathfrak{S}''>} \subseteq \mathfrak{N}_{<\mathfrak{S}'>}$. Die Behauptung ergibt sich direkt aufgrund der Transitivität von "$\subseteq$". ✓

□

Bemerkung 6.0.7:

$\preceq$ ist nicht anti-symmetrisch. Dies ergibt sich später unmittelbar anhand von Beispiel 6.1.7 zusammen mit dem 1. Ordnungstheorem 6.1.21.

Definition 6.0.8: [kanonische Äquivalenzrelation]

Wir definieren $\equiv$ als die zu $\preceq$ kanonische Äquivalenzrelation.

Für zwei p-Noninterferenz-Spezifikationen $\mathfrak{S}$ und $\mathfrak{S}'$ ist $\mathfrak{S} \preceq \mathfrak{S}'$ gerade so zu verstehen, dass sofern $\mathfrak{S}$ erfüllt ist auch $\mathfrak{S}'$ erfüllt ist, d.h. $\mathfrak{S}'$ beschreibt eine stärkere Form von Noninterferenz-Sicherheit als $\mathfrak{S}$. Entsprechend bedeutet $\mathfrak{S} \equiv \mathfrak{S}'$ dass beide Noninterferenz-Spezifikationen im Grunde dieselbe Form von Sicherheit spezifizieren, d.h. $\mathfrak{S}$ ist erfüllt genau dann, wenn $\mathfrak{S}'$ erfüllt ist.

6.1 Beobachter-Hierarchien

Ziel dieses Abschnitts stellt die Definition einer natürlichen Ordnungsstruktur auf Beobachtern dar, welche mit der Ordnung der Noninterferenz-Spezifikationen harmoniert.

Wir orientieren uns hierfür an der Vorstellung, welche für zwei Beobachter $\mathcal{B}$ und $\mathcal{B}'$, den Beobachter $\mathcal{B}'$ als stärker erachtet, falls Beobachtungen von $\mathcal{B}'$ so einschränkt bzw. adaptiert werden können, sodass diese mit denen von Beobachter $\mathcal{B}$ übereinstimmen. Zudem möchten wir Möglichkeit haben Beobachter lediglich im Kontext spezifischer Programmausführungen in Relation zu setzten. Beispielsweise wird sich später herausstellen, dass die volle Angriffsstärke einer Vielzahl bedeutender Beobachter im Kontext deterministischer Systeme nicht zur Geltung kommt.
Wir werden diese Ansätze formal durch relationale Ordnungsklassen umzusetzen. Hierzu definieren sowohl eine allgemeine relationale Ordnungsklasse $\preceq$, als auch eine Konkretisierung $\preceq_{\mathcal{K}}$, parametrisiert mit einem variablen $\mathcal{K} \subseteq \mathfrak{2}^{\mathfrak{K}}$.

Definition 6.1.1: [Ordnungen]

Wir definieren folgende relationale Ordnungsklassen auf der Klasse der Beobachter

- $\mathcal{B} \preceq \mathcal{B}' \quad :\Leftrightarrow \; \exists f : \mathcal{B} = f \circ \mathcal{B}'$
- $\mathcal{B} \preceq_{\mathcal{K}} \mathcal{B}' \; :\Leftrightarrow \; \exists f : \; \mathcal{B}|_{\mathcal{K}} = f \circ \mathcal{B}'|_{\mathcal{K}}$ für $\mathcal{K} \subseteq \mathfrak{2}^{\mathfrak{K}}$

Wir bezeichnen f im beiden Fällen als *Reduktion*.

Bemerkung 6.1.2:

.

- Für zwei Beobachter $\mathcal{B}$ und $\mathcal{B}'$ mit $\mathcal{B} \preceq \mathcal{B}'$ darf man sich $\mathcal{B}'$ "stärker" vorstellen als als $\mathcal{B}$
- Für zwei Beobachter $\mathcal{B}$ und $\mathcal{B}'$ mit $\mathcal{B} \preceq_{\mathcal{K}} \mathcal{B}'$ darf man sich $\mathcal{B}'$ "$\mathcal{K}$-stärker" vorstellen als $\mathcal{B}$

Proposition 6.1.3:

Für beliebige Beobachter $\mathcal{B}$ und $\mathcal{B}'$ sind folgende Aussagen äquivalent

(i) $\mathcal{B} \preceq \mathcal{B}'$

(ii) $\mathcal{B} \preceq_{\mathfrak{2}^{\mathfrak{K}}} \mathcal{B}'$

(iii) $\forall \mathcal{K} \subseteq \mathfrak{2}^{\mathfrak{K}} : \; \mathcal{B} \preceq_{\mathcal{K}} \mathcal{B}'$

Beweis.
Wir beweisen die Aussage, indem wir zeigen $(i) \Leftrightarrow (ii)$ sowie $(ii) \Rightarrow (iii)$ und $(iii) \Rightarrow (ii)$.

- $[(i) \Leftrightarrow (ii)]$

  Es gilt $\mathcal{B} \preceq \mathcal{B}'$ per Definition gdw. eine Reduktion f existiert mit $\mathcal{B} = f \circ \mathcal{B}'$. Da sowohl $dom(\mathcal{B}) = \mathfrak{2}^{\mathfrak{K}}$ als auch $dom(\mathcal{B}') = \mathfrak{2}^{\mathfrak{K}}$ gilt dies wiederum gdw. $\mathcal{B}|_{\mathfrak{2}^{\mathfrak{K}}} = f \circ \mathcal{B}'|_{\mathfrak{2}^{\mathfrak{K}}}$ was per Definition gleichbedeutend ist mit $\mathcal{B} \preceq_{\mathfrak{2}^{\mathfrak{K}}} \mathcal{B}'$. ✓

- $[(ii) \Rightarrow (iii)]$

  Sei $\mathcal{K} \subseteq \mathfrak{AK}$ beliebig. Laut Voraussetzung gilt $\mathcal{B} \preceq_{\mathfrak{AK}} \mathcal{B}'$, d.h. es existiert eine Reduktion f mit $\mathcal{B}|_{\mathfrak{AK}} = f \circ \mathcal{B}'|_{\mathfrak{AK}}$. Da $\mathcal{K} \subseteq \mathfrak{AK}$ gilt aber auch $\mathcal{B}|_{\mathcal{K}} = f \circ \mathcal{B}'|_{\mathcal{K}}$. Es ergibt sich $\mathcal{B} \preceq_{\mathcal{K}} \mathcal{B}'$ für f als Reduktion und somit ebenfalls die Behauptung. ✓

- $[(iii) \Rightarrow (ii)]$

  Ergibt sich direkt für $\mathcal{K} = \mathfrak{AK}$. ✓

□

Proposition 6.1.4:

(i) $\preceq$ ist Präordnung

(ii) $\preceq_{\mathcal{K}}$ ist Präordnung, für jedes $\mathcal{K} \subseteq \mathfrak{AK}$

Beweis.
Wir zeigen nur (ii), anschließend ergibt sich (i) mit Proposition 6.1.3 für $\mathcal{K} := \mathfrak{AK}$.
Sei also $\mathcal{K} \subseteq \mathfrak{AK}$ beliebig. Es muss für $\preceq_{\mathcal{K}}$ die Gültigkeit folgender Eigenschaften nachgewiesen werden.

- [Reflexivität]

  Sei $\mathcal{B}$ ein beliebiger Beobachter. Wir müssen zeigen $\mathcal{B} \preceq_{\mathcal{K}} \mathcal{B}$.
  Offensichtlich gilt $\mathcal{B}|_{\mathcal{K}} = id \circ \mathcal{B}|_{\mathcal{K}}$ und somit insbesondere die Behauptung für id als Reduktion. ✓

- [Transitivität]

  Seien $\mathcal{B}$,$\mathcal{B}'$,$\mathcal{B}''$ beliebige Beobachter mit $\mathcal{B} \preceq_{\mathcal{K}} \mathcal{B}'$ sowie $\mathcal{B}' \preceq_{\mathcal{K}} \mathcal{B}''$. Wir müssen zeigen, dass $\mathcal{B} \preceq_{\mathcal{K}} \mathcal{B}''$.
  Laut Voraussetzung existieren Reduktionen f und g mit $\mathcal{B}|_{\mathcal{K}} = f \circ \mathcal{B}'|_{\mathcal{K}}$ sowie $\mathcal{B}'|_{\mathcal{K}} = g \circ \mathcal{B}''|_{\mathcal{K}}$. Zusammenfassend ergibt sich $\mathcal{B}|_{\mathcal{K}} = f \circ (g \circ \mathcal{B}''|_{\mathcal{K}})$ und aufgrund der Assoziativität der Komposition ebenfalls $\mathcal{B}|_{\mathcal{K}} = (f \circ g) \circ \mathcal{B}''|_{\mathcal{K}}$. Es folgt die Behauptung für $(f \circ g)$ als Reduktion. ✓

□

Proposition 6.1.5:

$\preceq$ ist keine Totalordnungen

Beweis.
Wir zeigen hierfür für die Beobachter Ü-$pTrace$ und Z-$pTrace$ aus Beispiel 4.2.3, dass

- Ü-$pTrace \not\leq Z$-$pTrace$
- Z-$pTrace \not\leq$ Ü-$pTrace$

Angenommen Ü-$pTrace \leq Z$-$pTrace$, dann existiert eine Reduktion f mit Ü-$pTrace = f \circ Z$-$pTrace$. Betrachte nun das Transitionssystem $\mathcal{T} = (\{a,b\},\{a \to b, b \to a\})$ sowie die $\mathcal{T}$-Annotation $\mathfrak{a}$ mit $\mathfrak{a}_S = \tau$ sowie $\mathfrak{a}_T(a \to b) = a$ und $\mathfrak{a}_T(b \to a) = b$. Es gilt

- $Z\text{-}Trace((\mathcal{T}_{<\mathfrak{a},\tau>},\tau,\mathcal{T}_{<\mathfrak{a}>}\langle a\rangle)) = \{(),(\tau),(\tau,\tau),\dots\}$
- $Z\text{-}Trace((\mathcal{T}_{<\mathfrak{a},\tau>},\tau,\mathcal{T}_{<\mathfrak{a}>}\langle b\rangle)) = \{(),(\tau),(\tau,\tau),\dots\}$

und somit insbesondere $(\mathcal{T}_{<\mathfrak{a},\tau>},\tau,\mathcal{T}_{<\mathfrak{a}>}\langle a\rangle) =_{(f \circ Z\text{-}Trace)} (\mathcal{T}_{<\mathfrak{a},\tau>},\tau,\mathcal{T}_{<\mathfrak{a}>}\langle b\rangle)$.
Nun gilt aber

- Ü-$Trace((\mathcal{T}_{<\mathfrak{a},\tau>},\tau,\mathcal{T}_{<\mathfrak{a}>}\langle a\rangle)) = \{(),(a),(a,b),\dots\}$
- Ü-$Trace((\mathcal{T}_{<\mathfrak{a},\tau>},\tau,\mathcal{T}_{<\mathfrak{a}>}\langle b\rangle)) = \{(),(b),(b,a),\dots\}$

und folglich $(\mathcal{T}_{<\mathfrak{a},\tau>},\tau,\mathcal{T}_{<\mathfrak{a}>}\langle a\rangle) \neq_{Z\text{-}Trace} (\mathcal{T}_{<\mathfrak{a},\tau>},\tau,\mathcal{T}_{<\mathfrak{a}>}\langle b\rangle)$. Dies steht in direktem Widerspruch zur Existenz von f. $\lightning$

□

Bemerkung 6.1.6:

Aufgrund der Propositionen 6.1.5 und 6.1.3 ist $\leq_{\mathcal{K}}$ im Allgemeinen ebenfalls nicht total.

Beispiel 6.1.7:

Seien $\mathcal{B}$ und $\mathcal{B}'$ Beobachter definiert durch

- $(\mathcal{S}, p, [\mathfrak{b}]) \overset{\mathcal{B}}{\mapsto} p$
- $(\mathcal{S}, p, [\mathfrak{b}]) \overset{\mathcal{B}'}{\mapsto} cod(p)$

wobei $cod(p)$ eine beliebige nicht-triviale Codierung von p darstellt. Offensichtlich gilt sowohl $\mathcal{B} \leq \mathcal{B}'$ für $decod$ als Reduktion, wobei $decod$ die zu cod entsprechende Decodierung darstellt, als auch $\mathcal{B}' \leq \mathcal{B}$ für cod als Reduktion. Außerdem gilt $\mathcal{B} \neq \mathcal{B}'$ da cod als nicht-trivial vorausgesetzt. Es ergibt sich direkt ein Widerspruch zur Anti-Symmetrie von $\leq$.

Bemerkung 6.1.8:

Aufgrund von Beispiel 6.1.7 zusammen mit Proposition 6.1.3 ist $\preceq_{\mathcal{K}}$ im Allgemeinen ebenfalls nicht anti-symmetrisch.

Wir möchten nun weiterhin eine Vorstellung davon bekommen, wann zwei Beobachter als gleichstark aufzufassen sind. Betrachtet man hierzu beispielsweise Beispiel 6.1.7, dann stellt man fest, dass die jeweiligen Beobachtungen der entsprechenden Beobachter im Wesentlichen den selben Informationsgehalt widerspiegeln. Dies ergibt sich nämlich aus der Tatsachen, dass diese wechselseitig ineinander überführbar sind. Die wechselseitige Überführbarkeit ist jedoch gerade damit gleichzusetzen, dass die betreffenden Beobachter wechselseitig stärker bzw. schwächer sind. Wir wollen diesen Ansatz in folgender Definition ausnutzen um Beobachter als gleichstark zu charakterisieren. Anschließend werden wir beweisen, dass gleichstarke Beobachter in geordneten Beobachter-Ketten austauschbar sind.

Definition 6.1.9: [kanonische Äquivalenzrelation]

Wir definieren folgende relationale Klassen auf Beobachtern.

- $\equiv$ ist die zu $\preceq$ kanonische Äquivalenzrelation
- $\equiv_{\mathcal{K}}$ ist die zu $\preceq_{\mathcal{K}}$ kanonische Äquivalenzrelation für $\mathcal{K} \subseteq \mathfrak{AK}$

Bemerkung 6.1.10:

- Wir fassen zwei Beobachter $\mathcal{B}$ und $\mathcal{B}'$ als gleichstark auf gdw. $\mathcal{B} \equiv \mathcal{B}'$
- Wir fassen zwei Beobachter $\mathcal{B}$ und $\mathcal{B}'$ als $\mathcal{K}$-gleichstark auf gdw. $\mathcal{B} \preceq_{\mathcal{K}} \mathcal{B}'$

Proposition 6.1.11:

Für beliebige Beobachter $\mathcal{B}$ und $\mathcal{B}'$ sind folgende Aussagen äquivalent

(i) $\mathcal{B} \equiv \mathcal{B}'$

(ii) $\mathcal{B} \equiv_{\mathfrak{AK}} \mathcal{B}'$

(iii) $\forall \mathcal{K} \in \mathfrak{AK} : \ \mathcal{B} \equiv_{\mathcal{K}} \mathcal{B}'$

Beweis.
Wir zeigen die Aussagen, indem wir zeigen $(i) \Leftrightarrow (ii)$ sowie $(i) \Leftrightarrow (iii)$.

- $[(i) \Leftrightarrow (ii)]$

  Es gilt $\mathcal{B} \equiv \mathcal{B}'$ per Definition gdw. sowohl $\mathcal{B} \preceq \mathcal{B}'$ als auch $\mathcal{B}' \preceq \mathcal{B}$. Laut Proposition 6.1.3 gilt dies gdw. sowohl $\mathcal{B} \preceq_{\mathfrak{AK}} \mathcal{B}'$ als auch $\mathcal{B}' \preceq_{\mathfrak{AK}} \mathcal{B}$. Dies ist aber wiederum per Definition gleichbedeutend mit $\mathcal{B} \equiv_{\mathfrak{AK}} \mathcal{B}'$. ✓

- $[(i) \Leftrightarrow (iii)]$

  Es gilt $\mathcal{B} \equiv \mathcal{B}'$ per Definition gdw. sowohl $\mathcal{B} \preceq \mathcal{B}'$ als auch $\mathcal{B}' \preceq \mathcal{B}$. Laut Proposition 6.1.3 gilt dies gdw. für beliebiges $\mathcal{K} \subseteq \mathfrak{AK}$ sowohl $\mathcal{B} \preceq_{\mathcal{K}} \mathcal{B}'$ als auch $\mathcal{B}' \preceq_{\mathcal{K}} \mathcal{B}$. Dies ist aber wiederum per Definition gleichbedeutend mit $\mathcal{B} \equiv_{\mathcal{K}} \mathcal{B}'$. ✓

□

Proposition 6.1.12:

Seien $\mathcal{B}$ und $\mathcal{B}'$ beliebige Beobachter mit $\mathcal{B} \equiv_{\mathcal{K}} \mathcal{B}'$. Seien $\mathcal{B}_1$ und $\mathcal{B}_2$ weitere Beobachter mit $\mathcal{B}_1 \preceq_{\mathcal{K}} \mathcal{B} \preceq_{\mathcal{K}} \mathcal{B}_2$, dann gilt auch

$$\mathcal{B}_1 \preceq_{\mathcal{K}} \mathcal{B}' \preceq_{\mathcal{K}} \mathcal{B}_2$$

Beweis.
Wir müssen zeigen

- $[\mathcal{B}_1 \preceq \mathcal{B}']$

  Laut Voraussetzung gilt $\mathcal{B} \equiv \mathcal{B}'$ und somit insbesondere $\mathcal{B} \preceq \mathcal{B}'$. Da $\mathcal{B}_1 \preceq \mathcal{B}$ ergibt sich die Behauptung unmittelbar anhand der in Proposition 6.1.4 gezeigten Transitivität von $\preceq$. ✓

- $[\mathcal{B}' \preceq \mathcal{B}_2]$

  Folgt analog. ✓

□

Proposition 6.1.13:

Seien $\mathcal{B}$ und $\mathcal{B}'$ beliebige Beobachter mit $\mathcal{B} \equiv \mathcal{B}'$. Seien $\mathcal{B}_1$ und $\mathcal{B}_2$ weitere Beobachter mit $\mathcal{B}_1 \preceq \mathcal{B} \preceq \mathcal{B}_2$, dann gilt auch

$$\mathcal{B}_1 \preceq \mathcal{B}' \preceq \mathcal{B}_2$$

Beweis.
Ergibt sich unmittelbar aus den Proposition 6.1.12 und 6.1.3.

□

Wir wollen unsere Betrachtungen nun auf stärkste und schwächste Beobachter lenken. Hierzu definieren wir zwei Beobachter denen wir anschließend diese Eigenschaften nachweisen werden. Dies machen wir indem wir zeigen, dass beliebige Beobachter sich hierarchisch stets zwischen diesen einordnen lassen.

Definition 6.1.14: [*weak*]

Wir definieren den Beobachter *weak* durch

$$(\mathcal{S},\ p,\ [\mathfrak{b}]) \ \mapsto\ \blacksquare$$

Hierbei ist "■" als Konstante zu interpretieren.

Definition 6.1.15: [*strong*]

Wir definieren den Beobachter *strong* durch

$$strong = id_{\mathfrak{B}_{\mathbb{R}}}$$

Theorem 6.1.16: [Hierarchiesatz]

Für beliebige Beobachter $\mathcal{B}$ gilt

$$weak \ \preceq\ \mathcal{B} \ \preceq\ strong$$

Beweis.
Wir zeigen

- $[weak \ \preceq\ \mathcal{B}]$

  Wir müssen die Existenz einer Reduktion f zeigen mit $weak = f \circ \mathcal{B}$.

  Da *weak* eine konstante funktionale Klasse darstellt mit $im(weak) = \{\blacksquare\}$ ergibt sich die Behauptung für die eindeutig bestimmte funktionale Klasse $f : im(\mathcal{B}) \to \{\blacksquare\}$ als Reduktion. ✓

- $[\mathcal{B} \ \preceq\ strong]$

  Wir müssen die Existenz einer Reduktion f zeigen mit $\mathcal{B} = f \circ strong$.
  Da $strong \equiv id_{\mathfrak{B}_{\mathbb{R}}}$ ergibt sich die Behauptung für $\mathcal{B}$ selbst als Reduktion. ✓

□

Bemerkung 6.1.17:

Man beachte, dass Theorem 6.1.16 die Existenz von Beobachtern $weak'$ bzw $strong'$ mit $weak' \ \preceq\ weak$ bzw. $strong \ \preceq\ strong'$ nicht ausschließt. Beispielsweise stellt laut Proposition 6.1.12 jeder Beobachter aus $[weak]_{\equiv}$ ebenfalls einen maximal schwachen Beobachter dar (genau genommen sogar genau die Beobachter aus $[weak]_{\equiv}$). Analog stellt jeder Beobachter aus $[strong]_{\equiv}$ einen maximal starken Beobachter dar (genau genommen sogar genau die Beobachter aus $[strong]_{\equiv}$).

Als Beispiele betrachte man $weak' \in [weak]_{\equiv}$ mit

$$(\mathcal{S}, p, [\mathfrak{b}]) \overset{weak'}{\mapsto} c$$

für eine beliebige aber feste Konstante $c \neq \blacksquare$, sowie $strong' \in [strong]_{\equiv}$ mit

$$(\mathcal{S}, p, [\mathfrak{b}]) \overset{strong'}{\mapsto} cod((\mathcal{S}, p, [\mathfrak{b}]))$$

für eine nicht-triviale Codierungsfunktion cod.

Es bleibt dem Leser überlassen sich davon zu überzeugen, dass tatsächlich $weak' \equiv weak$ sowie $strong \equiv strong$. Insbesondere folgt hieraus die Uneindeutigkeit maximal bzw. minimal starker Beobachter, d.h. es existiert kein schwächster bzw. stärkster Beobachter.

Nachdem wir uns ausgiebig mit Beobachter-Ordnungen beschäftigt haben wollen wir die Zusammenhänge zur Spezifikations-Ordnung herausarbeiten. Hierfür betrachten wir zunächst das Verhalten der Ununterscheidbarkeit bzgl. geordneter Beobachter. Wie erhofft werden wir dabei feststellen, dass Ununterscheidbarkeit stärkerer Beobachter diese schwächerer impliziert. Wir werden sogar die Äquivalenz dieser Aussagen beweisen, was Definition 6.1.1 zusätzlich motiviert. Unter anderem ergibt sich hieraus die Identifizierung von Ununterscheidbarkeit bzgl. gleichstarker Beobachter.

Proposition 6.1.18:

Es gilt

(i) $\mathfrak{U}_{<\mathcal{B}'>} \subseteq \mathfrak{U}_{<\mathcal{B}>}$ gdw. $\mathcal{B} \preceq \mathcal{B}'$

(ii) $\mathfrak{U}_{<\mathcal{B}',\mathcal{K}>} \subseteq \mathfrak{U}_{<\mathcal{B},\mathcal{K}>}$ gdw. $\mathcal{B} \preceq_{\mathcal{K}} \mathcal{B}'$, für jedes $\mathcal{K} \subseteq \mathfrak{AR}$

Beweis.
Wir zeigen nur (ii), anschließend ergibt sich (i) aufgrund von Bemerkung 5.1.4 zusammen mit Proposition 6.1.3 für $\mathcal{K} := \mathfrak{AR}$.
Wir zeigen die Implikationen " $\Rightarrow$ " und " $\Leftarrow$ " getrennt.

- $[\Rightarrow]$

  Wir zeigen $f : im(\mathcal{B}'|_{\mathcal{K}}) \to im(\mathcal{B}|_{\mathcal{K}})$ mit $f(\mathcal{B}'|_{\mathcal{K}}(\mathfrak{k})) = \mathcal{B}|_{\mathcal{K}}(\mathfrak{k})$ ist eine wohldefinierte funktionale Klasse. Anschließend ergibt sich direkt anhand der Definition von f, dass $\mathcal{B}|_{\mathcal{K}} = f \circ \mathcal{B}'|_{\mathcal{K}}$.

  Die Wohldefiniertheit von f ergibt sich dabei wie folgt. Seien $\mathfrak{k},\mathfrak{k}' \in \mathcal{K}$ mit $\mathfrak{k} =_{\mathcal{B}'|_{\mathcal{K}}} \mathfrak{k}'$. Es folgt $\{\mathfrak{k},\mathfrak{k}'\} \in \mathfrak{U}_{<\mathcal{B}',\mathcal{K}>}$ und laut Voraussetzung ebenfalls $\{\mathfrak{k},\mathfrak{k}'\} \in \mathfrak{U}_{<\mathcal{B},\mathcal{K}>}$. Dies ist aber laut Proposition 5.1.5 gleichbedeutend mit $\mathfrak{k} =_{\mathcal{B}} \mathfrak{k}'$ und es ergibt sich die Behauptung. ✓

- $[\Leftarrow]$

  Sei $U \in \mathfrak{U}_{<\mathcal{B}',\mathcal{K}>}$ beliebig. Per Definition gilt $\mathcal{B}'(U) = const$. Laut Voraussetzung existiert nun eine Reduktion f mit $\mathcal{B} = f \circ \mathcal{B}'$. Folglich gilt $(f \circ \mathcal{B}')(U) = const$ und somit insbesondere $\mathcal{B}(U) = const$. Dies ist aber gerade gleichbedeutend mit $U \in \mathfrak{U}_{<\mathcal{B},\mathcal{K}>}$. ✓

□

Proposition 6.1.19:

Es gilt

(i) $\mathfrak{U}_{<\mathcal{B}'>} = \mathfrak{U}_{<\mathcal{B}>}$ gdw. $\mathcal{B} \equiv \mathcal{B}'$

(ii) $\mathfrak{U}_{<\mathcal{B},\mathcal{K}>} = \mathfrak{U}_{<\mathcal{B}',\mathcal{K}>}$ gdw. $\mathcal{B} \equiv_{\mathcal{K}} \mathcal{B}'$, für jedes $\mathcal{K} \subseteq \mathfrak{AK}$

Beweis.
Wir zeigen nur (ii), anschließend ergibt sich (i) analog.
Sei $\mathcal{K} \subseteq \mathfrak{AK}$ beliebig. Es gilt $\mathfrak{U}_{<\mathcal{B},\mathcal{K}>} = \mathfrak{U}_{<\mathcal{B}',\mathcal{K}>}$ gdw. $\mathfrak{U}_{<\mathcal{B},\mathcal{K}>} \subseteq \mathfrak{U}_{<\mathcal{B}',\mathcal{K}>}$ sowie $\mathfrak{U}_{<\mathcal{B}',\mathcal{K}>} \subseteq \mathfrak{U}_{<\mathcal{B},\mathcal{K}>}$. Laut Proposition 6.1.18 gilt dies wiederum gdw. $\mathcal{B}' \preceq_{\mathcal{K}} \mathcal{B}$ als auch $\mathcal{B} \preceq_{\mathcal{K}} \mathcal{B}'$. Dies ist aber per Definition gleichbedeutend mit $\mathcal{B}' \equiv_{\mathcal{K}} \mathcal{B}$. ✓

□

Proposition 6.1.20:

Seien $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B})$ und $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B}')$ beliebige p-Noninterferenz-Spezifikationen mit

(i) $\mathcal{B} \preceq_{\mathcal{K}} \mathcal{B}'$

(ii) $\bigcup_{\Delta \in \mathcal{C}} \{(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\ p,\ [\mathcal{T}_{<\mathfrak{a}>}\langle s\rangle]) \mid s \in \Delta\} \subseteq \mathcal{K}$

dann gilt

$$(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B}) \preceq (\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B}')$$

Beweis.
Wir müssen zeigen $\mathfrak{N}_{<(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B}')>} \subseteq \mathfrak{N}_{<(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B})>}$.

Gelte also $p \in \mathfrak{N}_{<(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B}')>}$. Es gilt für beliebiges $\Delta \in \mathcal{C}$, dass

$$\{(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\ p,\ [\mathcal{T}_{<\mathfrak{a}>}\langle s\rangle]) \mid s \in \Delta\ \} \in \mathfrak{U}_{<\mathcal{B}'>}$$

Laut Voraussetzung gilt aber auch $\{(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\ p,\ [\mathcal{T}_{<\mathfrak{a}>}\langle s\rangle]) \mid s \in \Delta\ \} \subseteq \mathcal{K}$ sowie $\mathcal{B} \preceq_{\mathcal{K}} \mathcal{B}'$. Es folgt

$$\{(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\ p,\ [\mathcal{T}_{<\mathfrak{a}>}\langle s\rangle]) \mid s \in \Delta\ \} \in \mathfrak{U}_{<\mathcal{B}',\mathcal{K}>}$$

und mit Proposition 6.1.18 auch

$$\{(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\ p,\ [\mathcal{T}_{<\mathfrak{a}>}\langle s\rangle]) \mid s \in \Delta\ \} \in \mathfrak{U}_{<\mathcal{B},\mathcal{K}>}$$

Nach Anwendung von Bemerkung 5.1.4 ergibt sich

$$\{(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\ p,\ [\mathcal{T}_{<\mathfrak{a}>}\langle s\rangle]) \mid s \in \Delta\ \} \in \mathfrak{U}_{<\mathcal{B}>}$$

Da Δ beliebig gewählt folgt die Behauptung.

□

Theorem 6.1.21: [1. Ordnungstheorem]

Seien $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B})$ und $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B}')$ beliebige p-Noninterferenz-Spezifikationen mit $\mathcal{B} \preceq \mathcal{B}'$, dann gilt

$$(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B}) \preceq (\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B}')$$

Beweis.
Laut Proposition 6.1.3 gilt $\mathcal{B} \preceq \mathcal{B}'$ gdw. $\mathcal{B} \preceq_{\mathfrak{IR}} \mathcal{B}'$. Außerdem gilt stets $\bigcup_{\Delta\in\mathcal{C}}\{(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\ p,\ [\mathcal{T}_{<\mathfrak{a}>}\langle s\rangle]) \mid s \in \Delta\} \subseteq \mathfrak{IR}$. Die Behauptung ergibt sich somit aufgrund von Proposition 6.1.20 für $\mathcal{K} := \mathfrak{IR}$.

□

Proposition 6.1.22:

Seien $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B})$ und $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B}')$ beliebige p-Noninterferenz-Spezifikationen mit

(i) $\mathcal{B} \equiv_{\mathcal{K}} \mathcal{B}'$

(ii) $\bigcup_{\Delta\in\mathcal{C}}\{(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\ p,\ [\mathcal{T}_{<\mathfrak{a}>}\langle s\rangle]) \mid s \in \Delta\} \subseteq \mathcal{K}$

dann gilt

$$(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B}) \equiv (\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B}')$$

Beweis.
Laut Voraussetzung gilt $\mathcal{B} \equiv_{\mathcal{K}} \mathcal{B}'$ und per Definition somit auch $\mathcal{B} \preceq_{\mathcal{K}} \mathcal{B}'$ sowie $\mathcal{B}' \preceq_{\mathcal{K}} \mathcal{B}$. Es ergibt sich aufgrund von Proposition 6.1.20 zusammen mit (ii), dass $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B}) \preceq (\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B}')$ genau wie $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B}') \preceq (\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B})$ und folglich auch die Behauptung.

□

Theorem 6.1.23: [2. Ordnungstheorem]

Seien $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B})$ und $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B}')$ beliebige p-Noninterferenz-Spezifikationen mit $\mathcal{B} \equiv \mathcal{B}'$, dann gilt

$$(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B}) \equiv (\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B}')$$

Beweis.
Laut Voraussetzung gilt $\mathcal{B} \equiv \mathcal{B}'$. Per Definition gilt dann sowohl $\mathcal{B} \preceq \mathcal{B}'$ als auch $\mathcal{B}' \preceq \mathcal{B}$. Es folgt mit mit dem 1. Ordnungstheorem 6.1.21, dass $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B}) \preceq (\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B}')$ genau wie $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B}') \preceq (\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B})$ und folglich auch die Behauptung.

□

6.2 Annotations-Hierarchien

Ziel dieses Abschnitts stellt die Definition einer natürlichen Ordnungsstruktur auf Annotationen dar. Diese soll mit der Ordnungsstruktur der Noninterferenz-Spezifikation harmonieren und so zum Nachweis von Noninterferenz beitragen. Annotationen können wir dabei als Formalisierung von Benutzersichten auffassen. Benutzersichten sind in der Regel Hierarchiestufen zugeordnet, dabei unterscheiden sich sich zwei Sichten verschiedener Hierarchiestufen in der Regel dadurch, dass die der tieferen Ebene eine Reduktion dieser der höheren Ebene darstellt. Dies motiviert uns erwähntes Konzept in Form einer relationalen Ordnungsklasse auf Annotationen zu übertragen. Hierfür definieren wir zunächst Ordnungsstrukturen für Zustands- und Übergangs-Annotationen. Anschließend werde wir diese auf die Struktur der Annotationen übertragen.

Definition 6.2.1: [Zustands-Reannotation]

Wir definieren eine Ordnung $\preceq$ auf Zustands-Annotationen durch

$$\mathfrak{a}_S \preceq \mathfrak{a}'_S \text{ gdw. } \exists f : \ \mathfrak{a}_S = f \circ \mathfrak{a}'_S$$

Wir bezeichnen f dann auch als *Zustands-Reannotation.*

Definition 6.2.2: [Übergangs-Reannotation]

Wir definieren eine Ordnung $\preceq$ auf Übergangs-Annotationen durch

$$\mathfrak{a}_T \preceq \mathfrak{a}'_T \text{ gdw. } \exists f : \ \mathfrak{a}_T = f \circ \mathfrak{a}'_T$$

Wir bezeichnen f dann auch als *Übergangs-Reannotation*.

Definition 6.2.3: [Annotation]

Wir definieren eine Ordnung $\preceq$ auf Annotationen durch

$$\mathfrak{a} \preceq \mathfrak{a}' \text{ gdw. } \mathfrak{a}_S \preceq \mathfrak{a}'_S \ \wedge \ \mathfrak{a}_T \preceq \mathfrak{a}'_T$$

Proposition 6.2.4:

Die folgende Ordnungen beschreiben Präordnungen

(i) $\preceq$ auf Zustands-Annotationen

(ii) $\preceq$ auf Übergangs-Annotationen

(iii) $\preceq$ auf Annotationen

Beweis.
Die Behauptungen (i) und (ii) ergeben sich analog zu 6.1.4. Es bleibt (iii) zu zeigen. Sei dafür im Folgenden $\mathcal{T}$ ein beliebiges Transitionssystem. Es muss für $\preceq$ die Gültigkeit folgender Eigenschaften nachgewiesen werden.

- [Reflexivität]

  Sei $\mathfrak{a}$ eine beliebige $\mathcal{T}$-Annotation. Wir müssen zeigen $\mathfrak{a}_S \preceq \mathfrak{a}_S$ und $\mathfrak{a}_T \preceq \mathfrak{a}_T$.
  Dies folgt direkt aufgrund der Reflexivität von $\preceq$ auf Zustands-Annotationen sowie $\preceq$ auf Übergangs-Annotationen. ✓

- [Transitivität]

  Seien $\mathfrak{a}$,$\mathfrak{a}'$,$\mathfrak{a}''$ beliebige $\mathcal{T}$-Annotationen mit $\mathfrak{a} \preceq \mathfrak{a}'$ sowie $\mathfrak{a}' \preceq \mathfrak{a}''$. Wir müssen zeigen, dass $\mathfrak{a} \preceq \mathfrak{a}''$.
  Aus $\mathfrak{a} \preceq \mathfrak{a}'$ und $\mathfrak{a}' \preceq \mathfrak{a}''$ folgt $\mathfrak{a}_S \preceq \mathfrak{a}'_S$ sowie $\mathfrak{a}'_S \preceq \mathfrak{a}''_S$ genau wie $\mathfrak{a}_T \preceq \mathfrak{a}'_T$ und $\mathfrak{a}'_T \preceq \mathfrak{a}''_T$. Die Behauptung folgt nun direkt aufgrund der Transitivität von $\preceq$ auf Zustands-Annotationen sowie $\preceq$ auf Übergangs-Annotationen. ✓

□

Wir wollen nun beginnen die Zusammenhänge zur Spezifikations-Ordnung herauszuarbeiten. Hierbei wird sich die Ordnung der Annotationen nicht ohne weiteres auf die der Noninterferenz-Spezifikationen übertragen lassen, was Beispiel 6.2.5 verdeutlichen soll.

Beispiel 6.2.5:

Wir betrachten den Beobachter $\mathcal{B}$ definiert durch

$$(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\ p,\ [\mathcal{G}_{<\mathfrak{a}^{\downarrow}>}[s_0]]) \mapsto \{\ (a,\ldots,a) \mid \exists \pi \in (\mathcal{G}[s_0])^{(*)} \ :\ |\pi| = |(a,\ldots,a)| \wedge \mathfrak{a}^{\downarrow}_S(\pi_i) = a\}$$

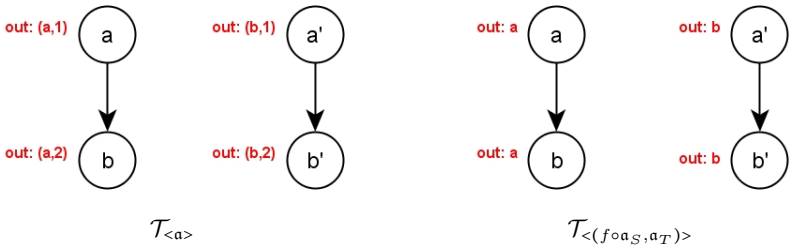


Abbildung 6.1: Graphische Darstellung von $\mathcal{T}_{<\mathfrak{a}>}$ und $\mathcal{T}_{<(f\circ\mathfrak{a}_S,\mathfrak{a}_T)>}$ aus Beispiel 6.2.5

der lediglich dazu in der Lage ist homogen annotierte Pfade einer Programmausführung zu beobachten.

Weiterhin betrachten wir das annotierte Transitionssystem $\mathcal{T}_{<\mathfrak{a}>}$, dargestellt in Abbildung 6.1, mit

- $S^{(\mathcal{T})} = \{a,b,a',b'\}$
- $S^{(\mathcal{T})} = \{a \to b, a' \to b'\}$
- $\mathfrak{a}_S(a) = (a,1)$, $\mathfrak{a}_S(b) = (a,2)$, $\mathfrak{a}_S(a') = (b,1)$, $\mathfrak{a}_S(b') = (b,2)$
- $\mathfrak{a}_T = \tau$

Offensichtlich gilt

$$(\mathcal{T}_{<\mathfrak{a},\tau>},\tau,\mathcal{T}_{<\mathfrak{a}>}[1]) =_{\mathcal{B}} (\mathcal{T}_{<\mathfrak{a},\tau>},\tau,\mathcal{T}_{<\mathfrak{a}>}[1']$$

In beiden Fällen wäre die Beobachtung von $\mathcal{B}$ die leere Menge und somit identisch.

Betrachtet man nun die Zustands-Reannotation $f : im(\mathfrak{a}_S) \to \{a,b\}$ mit $(v,n) \mapsto v$, welche entsprechende Annotationen auf ihre erste Komponente projiziert. Offensichtlich gilt

$$(\mathcal{T}_{<(f\circ\mathfrak{a}_S,\mathfrak{a}_T),\tau>},\tau,\mathcal{T}_{<(f\circ\mathfrak{a}_S,\mathfrak{a}_T)>}[1]) \neq_{\mathcal{B}} (\mathcal{T}_{<(f\circ\mathfrak{a}_S,\mathfrak{a}_T),\tau>},\tau,\mathcal{T}_{<(f\circ\mathfrak{a}_S,\mathfrak{a}_T)>}[1'])$$

denn

- $\mathcal{B}((\mathcal{T}_{<(f\circ\mathfrak{a}_S,\mathfrak{a}_T),\tau>},\tau,\mathcal{T}_{<(f\circ\mathfrak{a}_S,\mathfrak{a}_T)>}[1])) = \{(a,a)\}$
- $\mathcal{B}((\mathcal{T}_{<(f\circ\mathfrak{a}_S,\mathfrak{a}_T),\tau>},\tau,\mathcal{T}_{<(f\circ\mathfrak{a}_S,\mathfrak{a}_T)>}[1'])) = \{(b,b)\}$

Um dem entgegenzuwirken führen wir im Folgenden den Begriff der Reannotations-Invarianz ein. Dieser stellt ein Beobachter-Attribut dar, welches uns garantiert dass die Erfüllbarkeit von Noninterferenz-Spezifikationen bei einheitlichen Annotationsänderungen erhalten bleibt. Wir definieren hierfür zunächst Zustands- und Übergangs-Reannotations-Invarianz und übertragen diese Anschließend auf die Struktur der Annotationen. Die Definitionen stellen dabei eine direkte Umsetzung der Forderungen dar.

Definition 6.2.6: [*z.r.a.i.*]

Wir bezeichnen einen Beobachter $\mathcal{B}$ als *zustands-reannotations-invariant (z.r.a.i.)* gdw.

$$\forall M \in \mathfrak{U}_{<\mathcal{B}>} : \{(\mathcal{T}_{<(f\circ\mathfrak{a}_S,\mathfrak{a}_T),\mathfrak{p}>}, p, [\mathcal{T}_{<(f\circ\mathfrak{a}_S,\mathfrak{a}_T)>}\langle s\rangle]) \mid (\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>}, p, [\mathcal{T}_{<\mathfrak{a}>}\langle s\rangle]) \in M\} \in \mathfrak{U}_{<\mathcal{B}>}$$

für beliebiges, passendes f.

Definition 6.2.7: [ü*.r.a.i.*]

Wir bezeichnen einen Beobachter $\mathcal{B}$ als *übergangs-reannotations-invariant (z.r.a.i.)* gdw.

$$\forall M \in \mathfrak{U}_{<\mathcal{B}>} : \{(\mathcal{T}_{<(\mathfrak{a}_S,f\circ\mathfrak{a}_T),\mathfrak{p}>}, p, [\mathcal{T}_{<(\mathfrak{a}_S,f\circ\mathfrak{a}_T)>}\langle s\rangle]) \mid (\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>}, p, [\mathcal{T}_{<\mathfrak{a}>}\langle s\rangle]) \in M\} \in \mathfrak{U}_{<\mathcal{B}>}$$

für beliebiges, passendes f.

Definition 6.2.8: [*r.a.i.*]

Wir bezeichnen einen Beobachter $\mathcal{B}$ als *reannotations-invariant (r.a.i.)* gdw. $\mathcal{B}$ sowohl *z.r.a.i.* als auch ü*.r.a.i.* ist.

Theorem 6.2.9: [Ordnungstheorem]

Seien $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B})$ und $(\mathcal{T}_{<\mathfrak{a}',\mathfrak{p}>},\mathcal{C},\mathcal{B})$ beliebige p-Noninterferenz-Spezifikationen
Ist eine der folgenden Bedingungen erfüllt

(i) $\mathfrak{a} \preceq_z \mathfrak{a}'$ und $\mathcal{B}$ ist *z.r.a.i.*, wobei $\preceq_z = (\preceq) \otimes (=)$

(ii) $\mathfrak{a} \preceq_{\text{ü}} \mathfrak{a}'$ und $\mathcal{B}$ ist ü*.r.a.i.*, wobei $\preceq_{\text{ü}} = (=) \otimes (\preceq)$

(iii) $\mathfrak{a} \preceq \mathfrak{a}'$ und $\mathcal{B}$ ist *r.a.i.*

dann gilt

$$(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B}) \preceq (\mathcal{T}_{<\mathfrak{a}',\mathfrak{p}>},\mathcal{C},\mathcal{B})$$

Beweis.
Wir beschränken uns auf den Beweis von (*iii*). Anschließend folgen (*i*) und (*ii*) ähnlich. Wir müssen zeigen $\mathfrak{N}_{<(\mathcal{T}_{<\mathfrak{a}',\mathfrak{p}>},\mathcal{C},\mathcal{B})>} \subseteq \mathfrak{N}_{<(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B})>}$.

Sei $p \in \mathfrak{N}_{<(\mathcal{T}_{<\mathfrak{a}',\mathfrak{p}>},\mathcal{C},\mathcal{B})>}$. Es gilt für beliebiges $\Delta \in \mathcal{C}$, dass

$$\{(\mathcal{T}_{<\mathfrak{a}',\mathfrak{p}>}, p, \mathcal{T}_{<\mathfrak{a}'>}\langle s\rangle \mid s \in \Delta\} \in \mathfrak{U}_{<\mathcal{B}>}$$

Aufgrund von $\mathfrak{a} \preceq \mathfrak{a}'$ folgt nun die Existenz einer Zustands-Reannotation f mit $\mathfrak{a}_S = f \circ \mathfrak{a}'_S$ sowie einer Übergangs-Reannotation g mit $\mathfrak{a}_T = g \circ \mathfrak{a}'_T$. Da $\mathcal{B}$ *z.r.a.i.* ist folgt zunächst

$$\{(\mathcal{T}_{<(f\circ\mathfrak{a}'_S,\mathfrak{a}'_T),\mathfrak{p}>},p,\mathcal{T}_{<(f\circ\mathfrak{a}'_S,\mathfrak{a}'_T)>}\langle s\rangle \mid s \in \Delta\} \in \mathfrak{U}_{<\mathcal{B}>}$$

Hieraus ergibt sich aber, da $\mathcal{B}$ auch ü.*r.a.i.* ist, dass

$$\{(\mathcal{T}_{<(f\circ\mathfrak{a}'_S,g\circ\mathfrak{a}'_T),\mathfrak{p}>},p,\mathcal{T}_{<(f\circ\mathfrak{a}'_S,g\circ\mathfrak{a}'_T)>}\langle s\rangle \mid s \in \Delta\} \in \mathfrak{U}_{<\mathcal{B}>}$$

Es ergibt sich $\{(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},p,\mathcal{T}_{<\mathfrak{a}>}\langle s\rangle \mid s \in \Delta\} \in \mathfrak{U}_{<\mathcal{B}>}$ und da $\Delta \in \mathcal{C}$ beliebig gewählt ebenfalls $p \in \mathfrak{N}_{<(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},\mathcal{C},\mathcal{B})>}$.

□

Bemerkung 6.2.10:

Man beachte, dass Reannotions-Invarianz gerade so definiert ist es, dass gewünschte Zusammenhänge nahezu direkt aus der Definition folgen. Die eigentliche Problematik wurde somit auf die entsprechenden Nachweise von Reannotions-Invarianz verschoben. Dieser Abschnitt bietet hierfür jedoch strukturierte Mittel und eine feste Begriffsapparatur um entsprechende Nachweise zielgerichtet erbringen zu können.

Beispiel 6.2.11:

Der *pTrace*-Beobachter aus Beispiel 4.2.3 ist *r.a.i.*

Beweis.
Wir müssen zeigen, dass *pTrace* sowohl *z.r.a.i* als auch ü.*r.a.i.* Wir beschränken uns hier auf den Beweis der *z.r.a.i.*-Eigenschaft. Der Nachweis der ü.*r.a.i.*-Eigenschaft folgt analog.
Laut Proposition 5.1.5 reicht es hierfür zu zeigen, dass für beliebige annotierte Zustandsgraphen $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ mit

$$\begin{aligned}&\{\ (a_0,b_0,a_1,b_1,\dots) \mid \exists \pi \in (\mathcal{T}[s_0])^{(*)} :\ a_i = \mathfrak{a}_S(\pi_i),\ b_i = \mathfrak{a}_T(\pi_i \to \pi_{i+1})\}\\ =&\{\ (a_0,b_0,a_1,b_1,\dots) \mid \exists \pi \in (\mathcal{T}'[s'_0])^{(*)} :\ a_i = \mathfrak{a}'_S(\pi_i),\ b_i = \mathfrak{a}'_T(\pi_i \to \pi_{i+1})\}\end{aligned}$$

auch

$$\begin{aligned}&\{\ (a_0,b_0,a_1,b_1,\dots) \mid \exists \pi \in (\mathcal{T}[s_0])^{(*)} :\ a_i = f\circ\mathfrak{a}_S(\pi_i),\ b_i = \mathfrak{a}_T(\pi_i \to \pi_{i+1})\}\\ =&\{\ (a_0,b_0,a_1,b_1,\dots) \mid \exists \pi \in (\mathcal{T}'[s'_0])^{(*)} :\ a_i = f\circ\mathfrak{a}'_S(\pi_i),\ b_i = \mathfrak{a}'_T(\pi_i \to \pi_{i+1})\}\end{aligned}$$

für beliebiges f mit passendem Definitionsbereich. Dies ist jedoch offensichtlich.

□

KAPITEL 7

Naives Beobachter-Framework

Ziel dieses Kapitels stellt die Errichtung eines formalen Rahmens zur strukturierten Konstruktion von Beobachtern dar, basierend auf Semantiken annotierter Zustandsgraphen. Motiviert wird dies durch die Popularität und vorhandenen Erkenntnisse über diese innerhalb der theoretischen Informatik. Unsere Überlegungen stützen sich dabei auf die Arbeiten von van Glabbeck in [G01]. Wir errichten hierfür ein simples zweistufiges Framework, welches im Wesentlichen Beobachter konstruiert, die ihr Beobachtungen anhand der Abbildung von Programmausführungen bzgl. der Hintereinanderausführung funktionaler Klassen der Form

$$adap : TS\text{-}A\langle * \rangle \rightsquigarrow TS\text{-}A[*]$$

$$sem : TS\text{-}A[*] \to \mathscr{P}(Ł^{\mu})$$

kreieren. Funktionale Klassen der Form *sem* bezeichnen wir dabei als μ-Semantiken und betrachten diese als das Herzstück eines entsprechenden Beobachters. Die Klasse $Ł^{\mu}$ kann man sich dabei zunächst als Menge von "Zustandsgraph-Charakteristika" vorstellen. Die funktionale Klasse *adap* ist ein optionales Konstrukt um Zustandsgraphen an besondere Umstände anzupassen. Wir werden diese auch als Adaptionen bezeichnen.
Im Folgenden werden wir ausführlich auf Adaptionen und μ-Semantiken eingehen. Anschließend werden den Begriff der "Framwork-Instanz" definieren, welches gerade einem durch das Framework konstruierten Beobachter entspricht. Abschließend werden wir Ordnungen auf μ-Semantiken definieren, von denen wir feststellen werden, dass diese sich auf natürliche Weise auf die entsprechenden Framework-Instanzen übertragen.

Achtung:

> Im Folgenden schränken wir unsere Betrachtungen auf Systeme $\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>}$ mit endliche verzweigten $\mathcal{T}$ ein. Dies verschafft uns den Vorteil die kommende Theorie technisch übersichtlicher aufziehen zu können. Wir werden dies an

entsprechender Stelle explizit vermerken und Ansätze zur Verallgemeinerung vorschlagen. Man beachte hierbei dass existierende Systeme stets auf endlichen Größen basieren und deren Modellierungen in der Regel nicht betroffen sind.

7.1 Adaptionen

Ziel dieses Abschnitts wird die Definition und Erläuterung der Klasse der Adaptionen darstellen. Diese sollen eine optionale Vorstufe vor Anwendung der im Fokus stehenden Semantiken darstellen und beschreiben aus intuitiver Sicht Anpassungen von Zustandsgraphen an spezifische Besonderheiten der konkreten Umgebung. Adaptionen stellen dabei beliebige Isomorphie-Erhaltende Abbildung dar, welche von Zustandsgraphen in Baumausprägung auf allgemeine Zustandsgraphen abbilden. Die Forderung nach Isomorphie-Erhaltung garantiert uns, die Unabhängigkeit der Adaption von Systeminterna, was in Bezug auf die Essenz der Adaptionen eine sinnvolle Forderung ist.

Definition 7.1.1: [Adaption]

Als *Adaption* bezeichnen wir eine funktionale Klasse $adap : TS\langle * \rangle\text{-}A \rightsquigarrow TS[*]\text{-}A$ mit

$$\forall \mathcal{T}_{<\mathfrak{a}>}[s_0], \mathcal{T}'_{<\mathfrak{a}'>}[s'_0] \in TS\langle * \rangle\text{-}A : \mathcal{T}_{<\mathfrak{a}>}[s_0] \simeq \mathcal{T}'_{<\mathfrak{a}'>}[s'_0] \Rightarrow \mathcal{T}_{<\mathfrak{a}>}[s_0] \simeq_{adap} \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$$

Bemerkung 7.1.2:

Für eine Adaption *adap* und Zustandsgraphen $\mathcal{T}_{<\mathfrak{a}>}[s_0],\ \mathcal{T}'_{<\mathfrak{a}'}[s'_0] \in TS\langle * \rangle\text{-}A$ mit $\mathcal{T}_{<\mathfrak{a}}[s_0] \simeq \mathcal{T}'_{<\mathfrak{a}'}[s'_0]$ gilt

$$\mathcal{T}_{<\mathfrak{a}}[s_0] \in dom(adap) \text{ gdw. } \mathcal{T}'_{<\mathfrak{a}'}[s'_0] \in dom(adap)$$

Im Folgenden soll das Konzept der Adaptionen anhand diverser Beispiele veranschaulicht werden. Wir beginnen dabei mit der trivialen Adaption der Identität, welche trotz ihrer Einfachheit eine essentielle Rolle einnimmt. Die Identität als Adaption beschreibt dabei den häufig auftretenden Fall, dass gegebene annotierte Zustandsgraphen in ihrer bestehenden Form den äußeren Umständen entsprechen und somit keiner Adaption bedürfen.

Beispiel 7.1.3: [*id*-Adaption]

Die Identität $id_{TS\langle * \rangle\text{-}A} : TS\langle * \rangle\text{-}A \to TS[*]\text{-}A$ ist eine Adaption.

Bemerkung 7.1.4:

Die *id*-Adaption ist eine Adaption.

Beweis.
Zunächst einmal ist klar, dass da $id_{TS\langle * \rangle\text{-}A} : TS\langle * \rangle\text{-}A \to TS[*]\text{-}A$ insbesondere auch $id_{TS\langle * \rangle\text{-}A} : TS\langle * \rangle\text{-}A \rightsquigarrow TS[*]\text{-}A$.
Es bleibt für $\mathcal{B}$,$\mathcal{B}' \in TS\langle * \rangle\text{-}A$ mit $\mathcal{B} \simeq \mathcal{B}'$ zu zeigen, dass $\mathcal{B} \simeq_{id} \mathcal{B}'$. Dies ist jedoch offensichtlich.

□

Als nächstes wollen wir eine Adaption zur Beseitigung stiller Übergänge annotierter Zustandsgraphen vorstellen, wobei wir in diesem Kontext den Zustands-Annotationen keine Beachtung schenken. Einen Übergang bezeichnen wir dabei als still, falls dieser mit τ annotierten ist.

Beispiel 7.1.5: [*silent*-Adaption]

Die funktionale Klasse $silent : TS\langle * \rangle\text{-}A \to TS[*]\text{-}A$ mit

$$\mathcal{T}_{<(\mathfrak{a}_S,\mathfrak{a}_T)>}[s_0] \mapsto \mathcal{T}'_{<(\mathfrak{a}'_S,\mathfrak{a}'_T)>}[s_0]$$

wobei

- $S^{(\mathcal{T}')} := S^{(\mathcal{T})}$
- $T^{(\mathcal{T}')} := \{s \to s' \mid \exists \pi \in {}_{(*)}\mathcal{T}^{(n\geq 2)} : \begin{array}{l}(i)\ \pi_0 = s,\\ (ii)\ \pi_{|\pi|-1} = s',\\ (iii)\ \mathfrak{a}_T(\pi_{|\pi|-2} \to \pi_{|\pi|-1}) \neq \tau,\\ (iv)\ \forall i \in \{0,\ldots,|\pi|-3\} : \mathfrak{a}_T(\pi_i \to \pi_{i+1}) = \tau\end{array}$
- $\mathfrak{a}'_S(s) := \tau$
- $\mathfrak{a}'_T(s \to s') := \mathfrak{a}_T(s'^{\uparrow} \to s')$, wobei $s^{\uparrow}$ der eindeutig bestimmte $\mathcal{T}$-Systemzustand ist mit $s^{\uparrow} \to s \in T^{(\mathcal{T})}$ ist (existiert laut Lemma 3.1.27)

ist eine Adaption.

Bemerkung 7.1.6:

Wir nutzen in Definition 7.1.5 die Voraussetzung, dass die abzubildenden Zustandsgraphen in Baumausprägung vorliegen. Beachte hierfür die Definition von "$\mathfrak{a}'_T$".

Bemerkung 7.1.7:

Die *silent*-Adaption ist eine Adaption.

Beweis.
Zunächst einmal ist klar, dass da $silent : TS\langle * \rangle\text{-}A \to TS[*]\text{-}A$ insbesondere auch $silent : TS\langle * \rangle\text{-}A \rightsquigarrow TS[*]\text{-}A$.

Es bleibt für beliebige Zustandsgraphen $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ mit

$$\mathcal{T}_{<\mathfrak{a}>}[s_0] \simeq \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$$

zu zeigen, dass auch

$$\mathcal{T}_{<\mathfrak{a}>}[s_0] \simeq_{silent} \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$$

Sei dafür

- $\mathcal{T}^{\tau}{}_{<\mathfrak{a}^{\tau}>}[s_0] := silent(\mathcal{T}_{<\mathfrak{a}>}[s_0])$
- $\mathcal{T}'^{\tau}{}_{<\mathfrak{a}'^{\tau}>}[s'_0] := silent(\mathcal{T}'_{<\mathfrak{a}'>}[s'_0])$

Aufgrund von $\mathcal{T}_{<\mathfrak{a}>}[s_0] \simeq \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ folgt zunächst die Existenz eines Isomorphismus $f : S^{(\mathcal{T})} \to S^{(\mathcal{T}')}$ zwischen $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$.
Wir zeigen, dass dies ebenfalls ein Isomorphismus zwischen $\mathcal{T}^{\tau}{}_{<\mathfrak{a}^{\tau}>}[s_0]$ und $\mathcal{T}'^{\tau}{}_{<\mathfrak{a}'^{\tau}>}[s'_0]$ darstellt, anschließend folgt die Behauptung.

Hierfür müssen wir für f die Gültigkeit folgender Punkte nachweisen.

- $[(s \to s') \in T^{(\mathcal{T}^{\tau})} \Leftrightarrow (f(s) \to f(s')) \in T^{(\mathcal{T}'^{\tau})}]$

  Wir beschränken uns auf den Beweis der Implikation "⇒". Der Fall "⇐" folgt analog.

  Sei dafür $(s \to s') \in T^{(\mathcal{T}^{\tau})}$. Per Definition existiert ein Pfad $(\pi_i)_{i\in I} \in {}_{(*)}\mathcal{T}^{(*)}$ mit $(i) - (iv)$. Aufgrund der Isomorphie-Eigenschaft von f folgt dann aber auch für $(f(\pi_i))_{i\in I} \in {}_{(*)}\mathcal{T}'^{(*)}$, die Gültigkeit von $(i) - (iv)$ und es ergibt sich $(f(s) \to f(s')) \in T^{(\mathcal{T}'^{\tau})}$. ✓

- $[\mathfrak{a}^{\tau}_S(s) = \mathfrak{a}'^{\tau}_S(s)]$

  Trivial da per Definition sowohl $\mathfrak{a}^{\tau}_S(s) = \tau$ als auch $\mathfrak{a}'^{\tau}_S = \tau$. ✓

- $[\mathfrak{a}^{\tau}_T(s \to s') = \mathfrak{a}'^{\tau}_T(f(s) \to f(s'))]$

  Es gilt $\mathfrak{a}^{\tau}_T(s \to s') = \mathfrak{a}_T(s'^{\uparrow} \to s')$ wobei $s'^{\uparrow}$ der eindeutig bestimmten Vorgänger von s' in $\mathcal{T}$ ist. Aufgrund der Isomorphie-Eigenschaft von f folgt $\mathfrak{a}_T(s'^{\uparrow} \to s') = \mathfrak{a}'_T(f(s'^{\uparrow}) \to f(s'))$. Nun ist aber offensichtlich $f(s'^{\uparrow})$ der Vorgänger von $f(s')$ und folglich gilt $\mathfrak{a}'^{\tau}_T(f(s) \to f(s')) = \mathfrak{a}'_T(f(s'^{\uparrow}) \to f(s'))$. Insbesondere ergibt sich $\mathfrak{a}^{\tau}_T(s \to s') = \mathfrak{a}'^{\tau}_T(f(s) \to f(s'))$. ✓

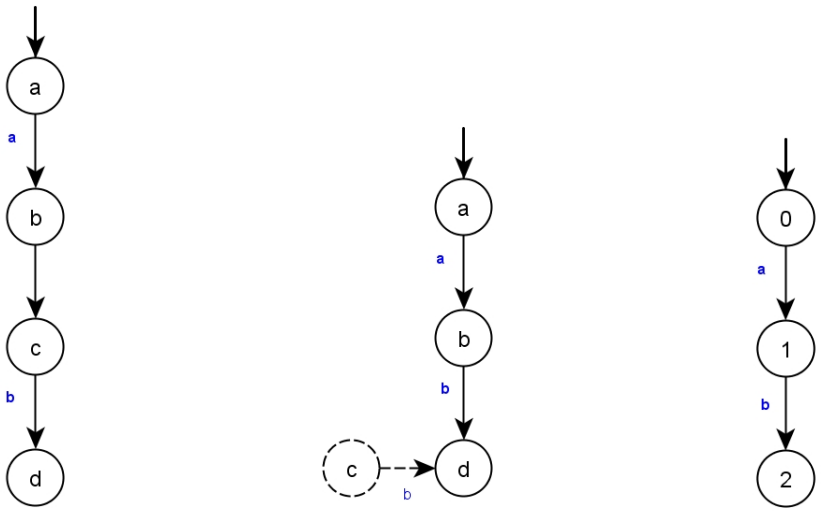


Abbildung 7.1: Graphische Darstellung der Graphen aus *silent*-Motivation

□

Wir wollen an dieser Stelle die Intention hinter den Adaptionen anhand der *silent*-Adaption noch einmal verdeutlichen. Hierfür betrachte man die Zustandsgraphen aus Abbildung 7.1. Erinnern wir uns nun an den Ü-$pTrace$ Beobachter aus Beispiel 4.2.3, dann würde dieser bzgl. eines Ausführungs-Kontextes mit gekapselter Programmausführung $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ zu folgender Beobachtung kommen $\{(),(a),(a,\tau),(a,\tau,b)\}$.
Betrachte man nun weiterhin die Beoachtung eines Ausführungs-Kontextes mit gekapselter Programmausführung $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$, dann käme Ü-$pTrace$ zu der Beobachtung $\{(),(a),(a,b)\}$. Lassen wir weitere Größen wie beispielsweise die Ausführungszeit außen vor, wäre dies nicht das Verhalten, welches wir erwarten würden. Da τ als unterdrücke Ausgabe von Ü-$pTrace$ nicht wahrgenommen werden sollte (wir können uns dies so vorstellen, dass der Bildschirm der entsprechenden Maschine bei einer kommenden Ausgabe von τ auf der gegenwärtigen Ausgabe verweilt), hätte man auch in dem Fall $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ eine Beobachtung der Form $\{(),(a),(a,b)\}$ erwartet, was die vorherige Anwendung der *silent*-Adaption motiviert (vgl. hierzu auch Abbildung 7.3).
Natürlich könnte man auch Ü-$pTrace$ so umdefinieren, dass gewünschtes Verhalten erreicht wird, aber allein aus intuitiver Sicht wäre dies kein vernünftiges Vorgehen, denn die besondere Eigenschaft von τ gilt in voller Allgemeinheit und betrifft somit insbesondere auch andere Beobachter. Es bietet sich daher an das Problem an der Wurzel zu packen und entsprechend die Änderungen am Modell der Programmausführung vorzunehmen. Dies hat unter anderem den Vorteil, dass nach einer Anpassung dieser die Beobachtungen sämtlich betroffener Beobachter wieder den Erwartungen entsprechen ohne diese künstlich anpassen zu müssen.
Würde man sich hingegen doch dazu entscheiden Ü-$pTrace$ so umzudefinieren, das

τ-Symbole entsprechend nicht in den Beobachtungen erscheinen, wäre dies damit gleichzusetzen, dass man Ü-*pTrace* "τ-blind" macht, d.h. lediglich für diesen Beobachter die Wahrnehmung von τ's unterbindet. Beide Ansätze würden zum selben Ergebnis führen, beschreiben aber aus intuitiver Sicht vollkommen verschiedene Ansätze.
Unser nächstes Ziel wird es sein eine Adaption für probabilistische Systeme zu entwickeln(vgl. hierzu Beispiel 3.2.11). Wir wollen dies zunächst anhand eines Beispiels motivieren.
Bedenkt man die besondere Intention hinter dieser der probabilistischen Systeme, nämlich diese als Modelle pseudo-parallele Systeme aufzufassen, dann stellt man schnell fest, dass ein Zustandsgraph, wie in Abbildung 7.2 nach außen in Form des Zustandsgraphen aus Abbildung 7.3 ersichtlich wird.
Dies ergibt sich nämlich wie folgt. Zunächst einmal ist klar, dass ein Angreifer keinen der Systemzustände annotiert mit "d" bzw. "e" unterscheiden kann, da diese terminale Systemzustände mit gleicher Annotation darstellen. Fasst man diese nun jeweils zusammen erkennt man anschließend, dass sich die mit "b" bzw. "c" annotierten Systemzustände nach außen hin ebenfalls nicht unterscheiden, denn jedes dieser geht mit Wahrscheinlichkeit 1 in einen Knoten, annotiert mit "d" bzw. "e". Weitere Zusammenfassung sind allein schon aufgrund der verschiedenen Annotationen der Knoten nicht möglich und es ergibt sich die obige Struktur.
Im Folgenden soll eine Adaption eine für zur Anpassung probabilistischer Systeme definiert werden, dies bedarf jedoch einiges an Vorbereitung. In obiger Motivation haben wir festgestellt, dass Angreifer probabilistischer Systeme nicht dazu in der Lage sind bestimmte Systemzustände zu unterscheiden. Genauer gilt folgendes.

> *"Gleich annotierte Systemzustände, welche mit gleicher Wahrscheinlichkeit in ebenfalls gleich annotierte Systemzustände mit selbigem Verhalten übergehen sind für Beobachter ununterscheidbar"*

Wir werden hierfür eine Äquivalenzrelation $R \in er(S^{(\mathcal{T})})$ definieren, welche uns anschließend als Basis für eine Partitionierung des $\mathcal{T}$-Systemzustandsraums in Partitionen maximaler Größe mit Beinhaltung probabilistisch ununterscheidbarer $\mathcal{T}$-Systemzustände dient. Für R muss also gelten, dass es bzgl. der Mengeninklusion die größte Äquivalenzrelation ist mit $(s,s') \in R$ gdw.

$$s =_{\mathfrak{a}} s'$$

$$\forall S \in S^{(\mathcal{T})}/_R : \; x \xrightarrow{p} S \Leftrightarrow y \xrightarrow{p} S$$

Die Maximalität von R garantiert uns, dass wir anschließend keine verschiedenen Äquivalenzklassen mit gleicher Erscheinung erhalten. Außerdem wird uns dies, wie wir später sehen werden, Eindeutigkeit garantieren. Wir werden nun die Existenz einer solchen Äquivalenzrelation R beweisen, dabei wollen wir uns die in Kapitel 2 aufgezogene Fixpunkttheorie zunutze machen.

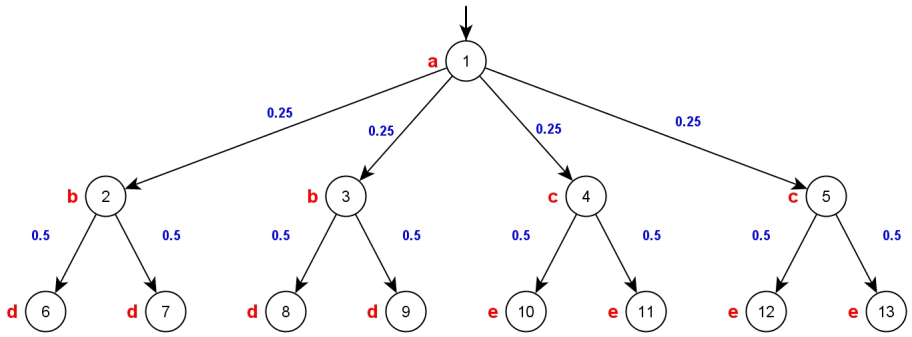


Abbildung 7.2: Zustandsgraph eines probabilistischen Systems

Definition 7.1.8: [probabilistischer Funktor]

Sei $\mathcal{T}_{<\mathfrak{a}>}$ ein probabilistisches System. Als *probabilistischen $\mathcal{T}_{<\mathfrak{a}>}$-Funktor* bezeichnen wir die Funktion $F : er(S^{\mathcal{T}}) \to er(S^{\mathcal{T}})$ mit $(s,s') \in F(R)$ gdw.

(1) $s =_{\mathfrak{a}} s'$

(2) $\forall S \in S^{(\mathcal{T})}/_R : \; s \xrightarrow{p} S \Leftrightarrow s' \xrightarrow{p} S$

Definition 7.1.9: [probabilistische Bisimulation]

Sei $\mathcal{T}_{<\mathfrak{a}>}$ ein probabilistisches System sowie F der zugehörige probabilistische $\mathcal{T}_{<\mathfrak{a}>}$-Funktor. Als *probabilistische $\mathcal{T}_{<\mathfrak{a}>}$-Bisimulation* bezeichnen wir Präfixpunkte von F.

Wir werden nun zeigen, dass probabilistischer Funktoren monoton sind. Anschließend ist es aufgrund des Satzes von Knaster-Tarski 2.0.50 offensichtlich, dass sofern R existiert, dieser den größten Fixpunkt des zugehörigen probabilistischen $\mathcal{T}_{<\mathfrak{a}>}$-Funktors definiert. Zudem werden wir die Stetigkeit von F beweisen. Der Fixpunktsatz von Kleene 2.0.49

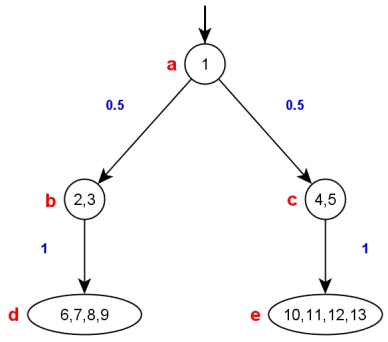


Abbildung 7.3: Zustandsgraph aus Abbildung 7.2 in “probabilistisch angepasster“ Form

bietet anschließend ein konstruktives Verfahren zur Erzeugung größter probabilistischer Bisimulationen.

Proposition 7.1.10:

Probabilistische Funktoren sind monoton

Beweis.
Sei F der probabilistische $\mathcal{T}_{<\mathfrak{a}>}$-Funktor eines beliebigen probabilistischen Transitionssystems $\mathcal{T}_{<\mathfrak{a}>}$. Wir müssen für beliebige $R,R' \in er(S^{(\mathcal{T})})$ mit $R \subseteq R'$ zeigen, dass auch $F(R) \subseteq F(R')$.

Sei dafür $(x,y) \in F(R)$ beliebig. Um zu zeigen, dass $(x,y) \in F(R')$ ist die Gültigkeit folgender Punkte nachzuweisen.

- $[\mathfrak{a}(s) = \mathfrak{a}(s')]$
  Ergibt sich direkt aus $(s,s') \in F(R)$. ✓
- $[\forall S \in S^{\mathcal{T}}/_{R'} : \; s \xrightarrow{p} S \;\Leftrightarrow\; s' \xrightarrow{p} S]$
  Sei $S \in S^{\mathcal{T}}/_{R'}$ beliebig. Laut Voraussetzung reicht es zu zeigen, dass S die disjunkte Vereinigung von R-Äquivalenzklassen darstellt, d.h. dass eine Menge $\{S_i \in S^{\mathcal{T}}/_{R}\}_{i\in I}$ existiert mit $S = \dot{\bigcup}_{i\in I} S_i$. Betrachte dafür die Menge $M := \{S' \in S^{\mathcal{T}}/_{R} \mid S \cap S' \neq \varnothing\}$. Da Äquivalenzklassen stets eine Partition der Grundmenge darstellen gilt $S \subseteq \dot{\bigcup} M$. Wir zeigen nun, dass auch $\dot{\bigcup} M \subseteq S$ und somit $S = \dot{\bigcup} M$ woraus anschließend die Behauptung ergibt. Sei hierfür $S' \in M$ beliebig, wir zeigen $S' \subseteq S$. Per Definition von M gilt $S' \cap S \neq \varnothing$, d.h. es existiert ein $a \in S' \cap S$. Sei nun $a' \in S'$ beliebig. Da $a,a' \in S'$ gilt $(a,a') \in R$. Laut Voraussetzung gilt $R \subseteq R'$ und somit auch $(a,a') \in R'$. Da $a \in S$ gilt somit ebenfalls $a' \in S$. ✓

□

Proposition 7.1.11:

Probabilistische Funktoren sind stetig.

Beweis.
Sei F der probabilistische $\mathcal{T}_{<\mathfrak{a}>}$-Funktor eines beliebigen probabilistischen Transitionssystems $\mathcal{T}_{<\mathfrak{a}>}$. Sei weiter $(R_i \in er(S^{\mathcal{T}}))_{i\in I}$ eine nichtleere Kette in $er(S^{\mathcal{T}})$ mit o.B.d.A "$R_0 \supseteq R_1 \supseteq \ldots$".

Wir müssen zeigen

$$F(\bigcap_{i\in I} R_i) = \bigcap_{i\in I} F(R_i)$$

Wir zeigen die Inklusionen " $\subseteq$ " und " $\supseteq$ " getrennt.

- [" $\subseteq$ "]

  Es gilt $\bigcap_{i\in I} R_i \subseteq R_i$ für alle $i \in I$. Aufgrund der in 7.1.10 gezeigten Monotonie von F folgt $F(\bigcap_{i\in I} R_i) \subseteq F(R_i)$ für alle i und somit insbesondere $F(\bigcap_{i\in I} R_i) \subseteq \bigcap_{i\in I} F(R_i)$. ✓

- [" $\supseteq$ "]

  Sei $(s,t) \in \bigcap_{i\in I} F(R_i)$, d.h. $(s,t) \in F(R_i)$ für alle $i \in I$.

  Wir müssen zeigen $(s,t) \in F(\bigcap_{i\in I} R_i)$, d.h. wir müssen die Gültigkeit folgender Punkte nachweisen.

  (i) $[\mathfrak{a}(s) = \mathfrak{a}(t)]$

  Klar, denn für beliebiges $i \in I$ gilt $(s,t) \in F(R_i)$ und somit insbesondere die Behauptung. ✓

  (ii) $[\forall S \in S^{\mathcal{T}}/_{(\bigcap_{i\in I} R_i)} : \; s \xrightarrow{p} S \;\Leftrightarrow\; t \xrightarrow{p} S]$

  Sei also $S \in S^{\mathcal{T}}/_{(\bigcap_{i\in I} R_i)}$ beliebig. Da $(R_i)_{i\in I}$ eine absteigende Kette darstellt gibt es eine ebenfalls absteigende Kette $(S_i \in S^{\mathcal{T}}/_{R_i})_{i\in I}$ mit $S = \bigcap_{i\in I} S_i$. Laut Voraussetzung gilt nun $s \xrightarrow{p_i} S_i \;\Leftrightarrow\; t \xrightarrow{p_i} S_i$. Es folgt aufgrund der Ketteneigenschaft von $(S_i)_{i\in I}$ für $p = \inf\{p_i \mid i \in I\}$, dass $s \xrightarrow{p} \bigcap_{i\in I} S_i \;\Leftrightarrow\; t \xrightarrow{p} \bigcap_{i\in I} S_i$. ✓

□

Bevor wir zur Definition der probabilistischen Adaption kommen, wollen wir zwei Beispiele zur Konstruktion größter probabilistischer Bisimulationen anhand des Fixpunkt Satzes von Klenne demonstrieren. Wir beginnen mit der Verifizierung des Beispiel in obiger Motivation.

Beispiel 7.1.12:

Wir verzichten auf Formalitäten und betrachten den Zustandsgraph aus Abbildung 7.2. Äquivalenzrelation werden wer in diesem Beispiel gemäß Bemerkung 2.0.22 durch eine entsprechende Partitionierung repräsentieren. Sei nun $\top := \{1,\ldots,13\}$ sowie F der zugehörige probabilistische Funktor, dann gilt

1. $F^0(\top) = \{\{1,\ldots,13\}\}$
2. $F^1(\top) = \{\{1\},\{2,3\},\{4,5\},\{6,7,8,9\},\{10,11,12,13\}\}$
3. $F^2(\top) = \{\{1\},\{2,3\},\{4,5\},\{6,7,8,9\},\{10,11,12,13\}\}$

und somit $\bigcap\{F^n(\top) \mid n \in \mathbb{N}\} = \{\{1\},\{2,3\},\{4,5\},\{6,7,8,9\},\{10,11,12,13\}\}$

Das Beispiel verdeutlicht, dass unsere in der Motivation beschriebene Intuition mit der theoretischen Umsetzung übereinstimmt. Das nächste Beispiel soll versuchen einen anschaulicheren Zugang zur Funktionsweise des Satzes von Kleene zu demonstrieren.

Beispiel 7.1.13:

Wir verzichten auf Formalitäten und betrachten den Zustandsgraph aus Abbildung 7.4. Äquivalenzrelation werden wer in diesem Beispiel gemäß Bemerkung 2.0.22 durch eine entsprechende Partitionierung repräsentieren. Sei nun $\top := \{1,\ldots,13\}$, dann gilt

1. $F^0(\top) = \{\{1,\ldots,13\}\}$
2. $F^1(\top) = \{\{1\},\{2,\ldots,7\},\{8,9\},\{10,\ldots,15\},\{16,17\}\}$
3. $F^2(\top) = \{\{1\},\{2,\ldots,5\},\{6,7\},\{8,9\},\{10,\ldots,13\},\{14,15\},\{16,17\}\}$
4. $F^3(\top) = \{\{1\},\{2,3\},\{4,5\},\{6,7\},\{8,9\},\{10,11\},\{12,13\},\{14,15\},\{16,17\}\}$
5. $F^4(\top) = \{\{1\},\{2,3\},\{4,5\},\{6,7\},\{8,9\},\{10,11\},\{12,13\},\{14,15\},\{16,17\}\}$

und somit

$\bigcap\{F^n(\top) \mid n \in \mathbb{N}\} = \{\{1\},\{2,3\},\{4,5\},\{6,7\},\{8,9\},\{10,11\},\{12,13\},\{14,15\},\{16,17\}\}$

Man erkennt mit fortschreitender Iteration, dass die Partitionierung gegen eine Aufteilung der Systemzustandsmenge strebt, welche Systemzustände mit gleicher Annotation und gleicher Anzahl an Schritt bis zum Terminierung zusammenfasst.

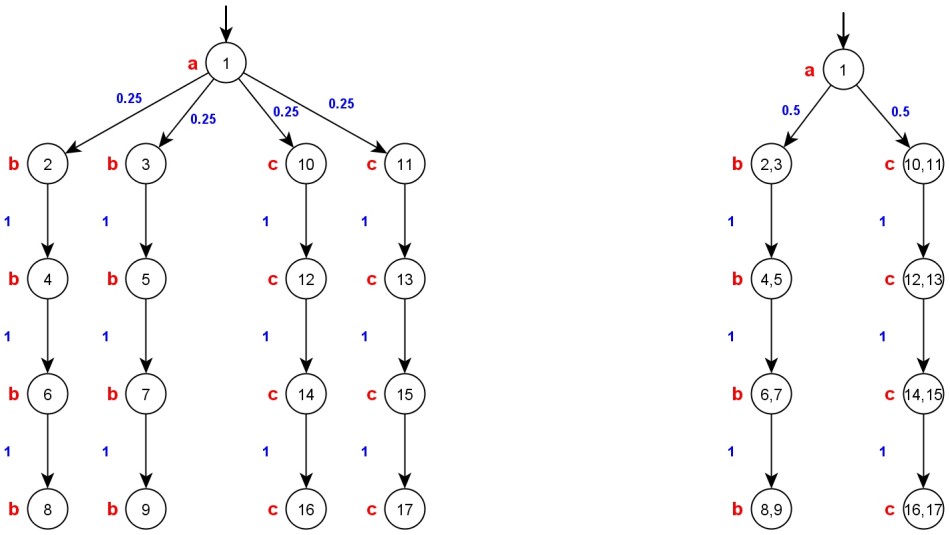


Abbildung 7.4: Graphische Darstellung der Zustandsgraphen aus Beispiel 7.1.13 - Links in ursprünglicher Version, rechts in "probabilistisch angepasster" Version

Bevor wir nun zur der Definition probabilistische Adaption kommen benötigen wir noch eine simple funktionale Klasse zur Reduktion von Systemzustandsmengen von Zustandsgraphen probabilistischer Systeme auf die "probabilistisch erreichbaren Systemzustände", damit meinen wir die Systemzustände, welche mit echt positiver Wahrscheinlichkeit vom Startzustand des entsprechenden Graphen erreicht werden können.

Definition 7.1.14: [*prbRed*]

Sei $pTS[*]\text{-}A$ die Klasse der Zustandsgraphen probabilistischer Transitionssystem. Wir definieren die funktionale Klasse $prbRed : pTS[*]\text{-}A \to pTS[*]\text{-}A$ durch

$$\mathcal{T}_{<\mathfrak{a}>}[s_0] \mapsto \mathcal{T}(S)_{<\mathfrak{a}>}[s_0]$$

wobei

$$S = \{s \in S^{(\mathcal{T})} \mid \exists \pi \in (\mathcal{T}[s_0])^{(*)} : \pi = (s_0, \ldots, s) \wedge \forall i \in [0,|\pi|-1)_{\mathbb{N}}\ \exists \rho > 0 : \pi_i \xrightarrow{\rho} \pi_{i+1}\}$$

Wir bezeichnen $prbRed(\mathcal{T}_{<\mathfrak{a}>})$ dann auch als den *probabilistisch reduzierten Zustandsgraph von* $\mathcal{T}_{<\mathfrak{a}>}[s_0]$.

Lemma 7.1.15:

Seien $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ beliebige Zustandsgraphen mit $\mathcal{T}_{<\mathfrak{a}>}[s_0] \simeq \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$, dann gilt auch

$$\mathcal{T}_{<\mathfrak{a}>}[s_0] \simeq_{prbRed} \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$$

Beweis.
Laut Voraussetzung gilt $\mathcal{T}_{<\mathfrak{a}>}[s_0] \simeq \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$, d.h. es existiert Isomorphismus $f : S^{(\mathcal{T})} \to S^{(\mathcal{T})}$ zwischen $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$.

Wir überlassen es dem Leser sich davon zu überzeugen, dass $f|_S$ ein Isomorphismus zwischen $prbRed(\mathcal{T}_{<\mathfrak{a}>}[s_0])$ und $prbRed(\mathcal{T}'_{<\mathfrak{a}'>}[s'_0])$ darstellt.

□

Beispiel 7.1.16: [*prob*-Adaption]

Sei $pTS\langle * \rangle\text{-}A$ die Klasse der Zustandsgraphen probabilistischer Transitionssystem (in Baumausprägung). Wir definieren die Adaption $prob : pTS\langle * \rangle\text{-}A \to TS[*]\text{-}A$ durch

$$\mathcal{T}_{<\mathfrak{a}>}[s_0] \mapsto \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$$

wobei

- $S^{(\mathcal{T}')} := S^{(\mathcal{T})}/_R$, für die größte probabilistische $prbRed(\mathcal{T}_{<\mathfrak{a}>})$-Bisimulation R

- $T^{(\mathcal{T}')} := \{\ [s]_R \to [t]_R \mid \exists s' \in [s]_R, t' \in [t]_R :\ s' \to t' \in T^{(\mathcal{T})}\}$
- $\mathfrak{a}'_S([s]_R) := \mathfrak{a}_S(s)$
- $\mathfrak{a}'_T([s]_R \to [t]_R) := \rho$, für $\rho \in [0,1]_{\mathbb{R}}$ mit $s \xrightarrow{\rho} [t]_R$
- $s'_0 := [s_0]_R$

Bemerkung 7.1.17:

Man beachte in Definition 7.1.16 dass R eine partielle Äquivalenzrelation darstellt.

Bemerkung 7.1.18:

Die *prob*-Adaption ist eine wohldefinierte Adaption.

Beweis.
Wir weisen zunächst die Wohldefiniertheit und somit die Repräsentantenunabhängigkeit in folgenden Punkten nach. Hierfür seien die Bezeichnungen wie in Definition 7.1.16.

- $[\mathfrak{a}'_S([s]_R) := \mathfrak{a}_S(s)]$
  Sei $s' \in S$ mit $(s,s') \in R$. Per Definition von R gilt $\mathfrak{a}_S(s) = \mathfrak{a}_S(s')$ und somit auch die Behauptung. ✓
- $[\mathfrak{a}'_T([s]_R \to [t]_R) := p$, für $\rho \in [0,1]_{\mathbb{R}}$ mit $s \xrightarrow{\rho} [t]_R]$
  Seien $s',t' \in S^{(\mathcal{T})}$ mit $(s,s'),(t,t') \in R$. Per Definition von R existiert für jedes $S \in S/_R$ ein $p \in [0,1]_{\mathbb{R}}$, sodass $s \xrightarrow{p} S \ \Leftrightarrow\ s' \xrightarrow{p} S$. Da $[t]_R,[t']_R \in S/_R$ und $[t]_R = [t']_R$ folgt die Behauptung. ✓

Wir werden nun zeigen, dass die *prob*-Adaption tatsächlich eine Adaption darstellt.

Zunächst einmal ist klar, dass da $prob : pTS\langle * \rangle\text{-}A \to TS[*]\text{-}A$ insbesondere auch $prob : TS\langle * \rangle\text{-}A \rightsquigarrow TS[*]\text{-}A$.

Es bleibt also für beliebige Zustandsgraphen (in Baumausprägung) $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ mit

$$\mathcal{T}_{<\mathfrak{a}>}[s_0] \simeq \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$$

zu zeigen, dass auch

$$\mathcal{T}_{<\mathfrak{a}>}[s_0] \simeq_{prob} \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$$

Hierfür zeigen wir für beliebige isomorphe Zustandsgraphen $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ mit entsprechendem Isomorphismus $f : S^{(\mathcal{T})} \to S^{(\mathcal{T}')}$, dass für eine beliebige probabilistische $\mathcal{T}_{<\mathfrak{a}>}[s_0]$-Bisimulation R die Relation

$$R' := \{(f(s),f(s')) \in S^{(\mathcal{T}')} \times S^{(\mathcal{T}')} \mid (s,s') \in R\}$$

eine probabilistische $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$-Bisimulation darstellt.
Anschließend ergibt für die größte probabilistische $\mathcal{T}_{<\mathfrak{a}>}[s_0]$-Bisimulation R_{max}, dass

$$R'_{max} \coloneqq \{(f(s),f(s')) \in S^{(\mathcal{T}')} \times S^{(\mathcal{T}')} \mid (s,s') \in R_{max}\}$$

eine probabilistische $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$-Bisimulation darstellt welche aufgrund der Maximalität von R_{max} die größte probabilistische $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$-Bisimulation darstellt.
Hieraus ergibt sich für $\mathcal{T}_{<\mathfrak{a}>}[s_0] \simeq \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$, da laut Lemma 7.1.15 auch $\mathcal{T}_{<\mathfrak{a}>}[s_0] \simeq_{prbRed} \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$, dass

$$R'_{redMax} = \{(f(s),f(s')) \in S^{(\mathcal{T}')} \times S^{(\mathcal{T}')} \mid (s,s') \in R_{redMax}\}$$

für

- R_{redMax} größte probabilistische $prbRed(\mathcal{T}_{<\mathfrak{a}>}[s_0])$-Bisimulation
- R'_{redMax} größte probabilistische $prbRed(\mathcal{T}'_{<\mathfrak{a}'>}[s'_0])$-Bisimulation

Anschließend ergibt sich, dass $F : S^{(\mathcal{T})}/_{R_{prbMax}} \to S^{(\mathcal{T}')}/_{R'_{prbMax}}$ mit

$$F(S) = \{f(s)\}_{s \in S}$$

ein Isomorphismus zwischen $prob(\mathcal{T}_{<\mathfrak{a}>}[s_0])$ und $prob(\mathcal{T}'_{<\mathfrak{a}'>}[s'_0])$ darstellt.

Wir müssen also für $(f(s),f(s')) \in R'$ die Gültigkeit folgender Punkte nachweisen.

- $[(f(s_0),f(s'_0)) \in R']$
  Folgt direkt aus $(s_0,s'_0) \in R$. ✓
- $[f(s) =_{\mathfrak{a}'} f(s')]$
  Da $(s,s') \in R$ folgt $s =_{\mathfrak{a}} s'$ und somit insbesondere $f(s) =_{\mathfrak{a}'} f(s')$. ✓
- $[\forall S' \in S^{(\mathcal{T}')}/_{R'} : f(s) \xrightarrow{\rho} S' \Leftrightarrow f(s') \xrightarrow{\rho} S']$
  Sei $S' \in S^{(\mathcal{T}')}/_{R'}$ beliebig. Sei weiter $S \coloneqq \{f^{-1}(s') \mid s' \in S'\}$. Per Definition von R' existiert ein $\rho \in [0,1]_{\mathbb{R}}$ mit $s \xrightarrow{\rho} S \Leftrightarrow s' \xrightarrow{\rho} S$, d.h $\sum_{s \xrightarrow{\rho_t} (t \in S)} \rho_t = \sum_{s' \xrightarrow{\rho_t} (t \in S)} \rho_t$. Aufgrund der Isomorphie-Eigenschaft von f folgt dann aber auch $\sum_{f(s) \xrightarrow{\rho_t} (t \in S')} \rho_t = \sum_{f(s') \xrightarrow{\rho} (t \in S')} \rho_t$ und somit auch $f(s) \xrightarrow{\rho} S' \Leftrightarrow f(s') \xrightarrow{\rho} S'$. ✓

□

7.2 $TS[*]$-A Semantiken

Wir kommen nun zu dem eigentlichen Kern dieses Frameworks. Ziel wird es sein Semantiken zu definieren, welche die adaptierten Zustandsgraphen auf spezifische Formel-Mengen des modalen μ-Kalküls abbilden. Formeln des μ-Kalküls kann man sich dabei als Eigenschaften von Zustandsgraphen vorstellen. Bevor wir uns den eigentlichen Semantik widmen, werden wir den modalen μ-Kalkül in allgemeiner Form definieren. Abschließend werden wir beispielhaft auf zwei populäre Semantiken eingehen.

Modaler μ-Kalkül

In diesem Abschnitt werden wir den modalen μ-Kalkül für unsere Zwecke spezifisch definieren und haben so insbesondere die Möglichkeit diesen zu wiederholen. Da dieser in ähnlicher Form als bekannt vorausgesetzt wird verzichten wir auf detaillierte Beispiele und Erläuterungen.
Im Folgenden sei APs ein Alias für die Klasse aller Mengen. Elemente aus APs wollen wir als Mengen atomarer Propositionen auffassen. Atomare Propositionen fassen dabei wir als zusammenhängende bzw. unteilbare Informationseinheiten auf. In unserem Kontext können wir uns diese stets als mögliche Ausgaben, bzw. auf Modell-Ebene, als mögliche Annotationen vorstellen.

Definition 7.2.1: [μ-Formeln]

Seien $A, B \in APs$ beliebige Mengen atomarer Propositionen. Sei weiter $Var := \{v_i\}_{i\in\mathbb{N}}$ eine Menge von (Variablen-)Symbolen. Die Menge $Ł^{\mu}_{<A,B>}$ der *μ-Formeln über* (A,B) ist durch folgende kontextfreie Grammatik bestimmt

$$\varphi \;:=\; a \mid \overset{b}{\rightarrow}(\varphi,\varphi) \mid v_i \mid \mu Z.\varphi \mid \neg\varphi \mid \wedge(\varphi,\varphi)$$

wobei $a \in A, b \in B$ und $v_i \in \{v_0, v_1, \dots\}$

Wir definieren weiterhin die Klasse aller μ-Formeln $Ł^{\mu}$ durch $Ł^{\mu} := \bigcup_{A,B\in APs} Ł^{\mu}_{<A,B>}$.

Bemerkung 7.2.2:

Die Grammatik aus Definition 7.2.1 ist aufgrund der einheitlichen Präfixnotation eindeutig.

Notation 7.2.3:

Wir schreiben auch

- $(\varphi_1) \overset{b}{\rightarrow} (\varphi_2)$ für $\overset{b}{\rightarrow}(\varphi_1,\varphi_2)$
- $(\varphi_1) \wedge (\varphi_2)$ für $\wedge(\varphi_1,\varphi_2)$

Bei Rechtsklammerung verzichten wir der Übersicht halber auch auf die Verwendung von Klammersymbolen.

Notation 7.2.4:

Für eine beliebige $\mathcal{T}$-Annotation $\mathfrak{a}$ schreiben wir auch $Ł^{\mu}_{<\mathfrak{a}>}$ für $L^{\mu}_{<im(\mathfrak{a}_S),im(\mathfrak{a}_T)>}$.

Im Folgenden wird unser Interesse μ-Formeln ohne Beinhaltung freier Variablen gelten. Wir fassen dabei eine Variable als frei innerhalb einer μ-Formel auf, falls sie nicht an den μ-Quantor gebunden ist. Wir werden später feststellen, dass μ-Formeln ohne Beinhaltung freier Variablen in gewissem Sinne unbeeinflusst von äußeren Größen sind. Wir wollen den Begriff der freien Variablen als nächstes formal definieren und darauf aufbauend den der variablen-freien μ-Formel.

Definition 7.2.5: [freie Variablen]

Seien $A, B \in APs$ beliebige Mengen atomarer Propositionen. Wir definieren die Funktion $FV_{<A,B>} : Ł^{\mu}_{<A,B>} \rightarrow \{v_i\}_{i\in\mathbb{N}}$ induktiv durch

- $[\psi = a,\ a \in A]$
  $FV_{<A,B>}(a) := \varnothing$
- $[\psi = \varphi_1 \overset{b}{\rightarrow} \varphi_2,\ b \in B]$
  $FV_{<A,B>}(\varphi_1 \overset{b}{\rightarrow} \varphi_2) := FV_{<A,B>}(\varphi_1) \cup FV_{<A,B>}(\varphi_2)$
- $[\psi = v_i]$
  $FV_{<A,B>}(v_i) := \{v_i\}$
- $[\psi = \mu v_i.\varphi]$
  $FV_{<A,B>}(\mu v_i.\varphi) := FV_{<A,B>}(\varphi) \setminus \{v_i\}$
- $[\psi = \neg\varphi]$
  $FV_{<A,B>}(\neg\varphi) := FV_{<A,B>}(\varphi)$
- $[\psi = \varphi_1 \wedge \varphi_2]$
  $FV_{<A,B>}(\varphi_1 \wedge \varphi_2) := FV_{<A,B>}(\varphi_1) \wedge FV_{<A,B>}(\varphi_2)$

Wir bezeichnen für eine μ-Formel φ die Elemente aus $FV(\varphi)$ als *freie Variablen von* φ. Variablen verschachtelt in φ, welche nicht frei sind, bezeichnen wir auch als *gebundene Variablen von* φ.

Definition 7.2.6: [variablen-freie μ-Formel]

Seien $A, B \in APs$ beliebige Mengen atomarer Propositionen. Wir definieren die Menge $Ł^{\mu_{BV}}_{<A,B>}$ der *variablen-freien μ-Formeln über* (A,B) durch

$$Ł^{\mu_{BV}}_{<A,B>} := \{\varphi \in Ł^{\mu}_{<A,B>} \mid FV_{<A,B>}(\varphi) = \varnothing\}$$

Wir definieren weiterhin die Klasse aller variablen-freien μ-Formeln $Ł^{\mu_{BV}}$ durch $Ł^{\mu_{BV}} := \bigcup_{A,B\in APs} Ł^{\mu_{BV}}_{<A,B>}$.

Notation 7.2.7:

Für eine beliebige $\mathcal{T}$-Annotation $\mathfrak{a}$ schreiben wir auch $Ł^{\mu_{BV}}_{<\mathfrak{a}>}$ für $Ł^{\mu_{BV}}_{<im(\mathfrak{a}_S),im(\mathfrak{a}_T)>}$.

Nachdem wir die syntaktischen Betrachtungen des μ-Kalküls abgeschlossen haben, wollen wir als nächstes ein entsprechendes Semantik-Funktional vorstellen. Dieses stellt eine mit einer Variablenbelegung β, sowie einem annotierten Transitionssystem $\mathcal{T}_{<\mathfrak{a}>}$ parametrisierte Funktion von der Menge der μ-Formeln in die Menge der $\mathcal{T}$-Systemzustände dar. Die Intuition hierbei ist, dass μ-Formeln auf passende, d.h. die Formel erfüllende, Systemzustände abgebildet werden.

Definition 7.2.8: [Semantik-Funktional]

Sei $\mathcal{T}_{<\mathfrak{a}>}$ eine annotiertes Transitionssystem sowie $\beta : \{v_i\}_{i\in\mathbb{N}} \to \mathscr{P}(S^{(\mathcal{T})})$ eine *Variablenbelegung*. Wir definieren induktiv die Funktion $\| \cdot \|^{\mathcal{T}_{<\mathfrak{a}>}}_{\beta}$: $Ł^{\mu}_{<\mathfrak{a}>} \to \mathscr{P}(S^{(\mathcal{T})})$ durch

- $\| a \|^{\mathcal{T}_{<\mathfrak{a}>}}_{\beta} := \mathfrak{a}_S^{-1}(a)$
- $\| \varphi_1 \xrightarrow{b} \varphi_2 \|^{\mathcal{T}_{<\mathfrak{a}>}}_{\beta} := \{ s \in \| \varphi_1 \|^{\mathcal{T}_{<\mathfrak{a}>}}_{\beta} \mid \exists s' \in \| \varphi_2 \|^{\mathcal{T}_{<\mathfrak{a}>}}_{\beta} : s \to s' \in T^{(\mathcal{T})} \wedge \mathfrak{a}_T(s \to s') = b \}$
- $\| v_i \|^{\mathcal{T}_{<\mathfrak{a}>}}_{\beta} := \beta(v_i)$
- $\| \mu v_i.\varphi \|^{\mathcal{T}_{<\mathfrak{a}>}}_{\beta} := \bigcap\{S \subseteq S^{(\mathcal{T})} \mid \| \varphi \|^{\mathcal{T}_{<\mathfrak{a}>}}_{\beta[v_i:=S]} \subseteq S\}$
- $\| \neg\varphi \|^{\mathcal{T}_{<\mathfrak{a}>}}_{\beta} := S^{(\mathcal{T})} \setminus \| \varphi \|^{\mathcal{T}_{<\mathfrak{a}>}}_{\beta}$
- $\| \bigwedge_{i\in I} \varphi_i \|^{\mathcal{T}_{<\mathfrak{a}>}}_{\beta} := \bigcap_{i\in I} \| \varphi_i \|^{\mathcal{T}_{<\mathfrak{a}>}}_{\beta}$

Wir werden abschließend eine Erfüllbarkeitsrelation in Abhängigkeit von μ-Formeln und Zustandsgraphen definieren. Dies machen wir zunächst in allgemeiner Form. Da wir aber im Allgemeinen unabhängig von Variablenbelegungen arbeiten wollen, konkretisieren wir unsere Überlegung für μ-Formeln aus $Ł^{\mu_{BV}}$. Für diese werden wir eine Unabhängigkeit von Variablenbelegung feststellen können.

Definition 7.2.9: [Erfüllbarkeitsrelation]

Wir sagen $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ *impliziert ein* $\varphi \in Ł^{\mu}_{<\mathfrak{a}>}$ *bzgl. einer* $\mathcal{T}$*-Variablenbelegung* β, in Zeichen $\mathcal{T}_{<\mathfrak{a}>}[s_0] \models \varphi[\beta]$, gdw. $s_0 \in \| \varphi \|_{\beta}^{\mathcal{T}_{<\mathfrak{a}}}$.

Proposition 7.2.10: .

Sei $\mathcal{T}_{<\mathfrak{a}>}$ ein annotiertes Transitionssystem sowie $\varphi \in Ł^{\mu}_{<\mathfrak{a}>}$ beliebig. Seien weiter β und β' beliebige μ-Variablenbelegungen mit $\beta|_{FV(\varphi)} = \beta'|_{FV(\varphi)}$, dann gilt

$$\| \varphi \|_{\beta}^{\mathcal{T}_{<\mathfrak{a}>}} = \| \varphi \|_{\beta'}^{\mathcal{T}_{<\mathfrak{a}>}}$$

Beweis.
Wir zeigen die Aussage per struktureller Induktion über den Formelaufbau von φ. Gelte die Behauptung also für alle echten Teilformeln von φ.

- $[\varphi = a,\ a \in A]$

  Es gilt
  $\| a \|_{\beta}^{\mathcal{T}_{<\mathfrak{a}>}} = \mathfrak{a}_S^{-1}(a) = \| a \|_{\beta'}^{\mathcal{T}_{<\mathfrak{a}>}}$. ✓

- $[\varphi = \psi_1 \overset{b}{\to} \psi_2,\ b \in B]$

  Es gilt $FV(\psi_1 \overset{b}{\to} \psi_2) = FV(\psi_1) \cup FV(\psi_2)$ und somit
  $\| \psi_1 \overset{b}{\to} \psi_2 \|_{\beta}^{\mathcal{T}_{<\mathfrak{a}>}} = \{\ s \in \| \psi_1 \|_{\beta}^{\mathcal{T}_{<\mathfrak{a}>}} \mid \exists s' \in \| \psi_2 \|_{\beta}^{\mathcal{T}_{<\mathfrak{a}>}}\colon\ s \to s' \in T^{(\mathcal{T})} \wedge \mathfrak{a}_T(s \to s') = b\ \} \overset{[IV]}{=} \{\ s \in \| \psi_1 \|_{\beta'}^{\mathcal{T}_{<\mathfrak{a}>}} \mid \exists s' \in \| \psi_2 \|_{\beta'}^{\mathcal{T}_{<\mathfrak{a}>}}\colon\ s \to s' \in T^{(\mathcal{T})} \wedge \mathfrak{a}_T(s \to s') = b\ \} = \| \psi_1 \overset{b}{\to} \psi_2 \|_{\beta'}^{\mathcal{T}_{<\mathfrak{a}>}}$. ✓

- $[\varphi = v_i]$

  Es gilt $FV(v_i) = \{v_i\}$ und somit
  $\| v_i \|_{\beta}^{\mathcal{T}_{<\mathfrak{a}>}} = \beta(v_i) \overset{[Vor]}{=} \beta'(v_i) = \| v_i \|_{\beta'}^{\mathcal{T}_{<\mathfrak{a}>}}$. ✓

- $[\varphi = \mu v_i.\psi]$

  Es gilt $FV(\mu v_i.\psi) = FV(\psi) \setminus \{v_i\}$ und somit
  $\| \mu v_i.\varphi \|_{\beta}^{\mathcal{T}_{<\mathfrak{a}>}} = \bigcap\{S \subseteq S^{(\mathcal{T})} \mid \| \psi \|_{\beta[v_i:=S]}^{\mathcal{T}_{<\mathfrak{a}>}} \subseteq S\} \overset{[IV]}{=} \bigcap\{S \subseteq S^{(\mathcal{T})} \mid \| \psi \|_{\beta'[v_i:=S]}^{\mathcal{T}_{<\mathfrak{a}>}} \subseteq S\} = \| \mu v_i.\varphi \|_{\beta'}^{\mathcal{T}_{<\mathfrak{a}>}}$. ✓

- $[\varphi = \neg\psi]$

  Es gilt $FV(\neg\psi) = FV(\psi)$ und somit
  $\| \neg\psi \|_{\beta}^{\mathcal{T}_{<\mathfrak{a}>}} = S^{(\mathcal{T})} \setminus \| \psi \|_{\beta}^{\mathcal{T}_{<\mathfrak{a}>}} \overset{[IV]}{=} S^{(\mathcal{T})} \setminus \| \psi \|_{\beta'}^{\mathcal{T}_{<\mathfrak{a}>}} = \| \neg\psi \|_{\beta'}^{\mathcal{T}_{<\mathfrak{a}>}}$. ✓

- $[\varphi = \psi_1 \wedge \psi_2]$
  Es gilt $FV(\psi_1 \wedge \psi_2) = FV(\psi_1) \cup FV(\psi_2)$ und somit
  $\| \psi_1 \wedge \psi_2 \|_{\beta}^{\mathcal{T}_{<\mathfrak{a}>}} = \| \psi_1 \|_{\beta}^{\mathcal{T}_{<\mathfrak{a}>}} \cup \| \psi_2 \|_{\beta}^{\mathcal{T}_{<\mathfrak{a}>}} \overset{[IV]}{=} \| \psi_1 \|_{\beta'}^{\mathcal{T}_{<\mathfrak{a}>}} \cup \| \psi_2 \|_{\beta'}^{\mathcal{T}_{<\mathfrak{a}>}} = \| \psi_1 \wedge \psi_2 \|_{\beta'}^{\mathcal{T}_{<\mathfrak{a}>}}$. ✓

□

Proposition 7.2.11:

Sei $\mathcal{T}_{<\mathfrak{a}>}$ ein annotiertes Transitionssystem sowie $\varphi \in Ł_{<\mathfrak{a}>}^{\mu_{BV}}$ beliebig. Es gilt für beliebige μ-Variablenbelegungen β und β', dass

$$\| \varphi \|_{\beta}^{\mathcal{T}_{<\mathfrak{a}>}} = \| \varphi \|_{\beta'}^{\mathcal{T}_{<\mathfrak{a}>}}$$

Beweis.
Da $\varphi \in Ł_{<\mathfrak{a}>}^{\mu_{BV}}$ gilt per Definition $FV(\varphi) = \varnothing$. Die Behauptung ergibt sich unmittelbar mit Proposition 7.2.10.

□

Notation 7.2.12:

Für $\varphi \in Ł_{<\mathfrak{a}>}^{\mu_{BV}}$ schreiben wir auch $\mathcal{T}_{<\mathfrak{a}>}[s_0] \vDash \varphi$ falls $\mathcal{T}_{<\mathfrak{a}>}[s_0] \vDash \varphi[\beta]$ für beliebiges β.

μ-Semantiken

Ziel dieses Abschnitts wird die Definition des Begriffs der μ-Semantik darstellen. Diese stellen Abbildungen auf Zustandsgraphen in die Menge der μ-Formeln dar. Hierbei ist es hilfreich sich μ-Formeln als spezifische Eigenschaften von Zustandsgraphen vorzustellen. Wir werden μ-Semantik in Abhängigkeit von "μ-Formelklassen" definieren. Diese können als Kollektion von μ-Formeln betrachtet werden.

Definition 7.2.13: [μ-Formelklasse]

Als *μ-Formelklasse* bezeichnen wir eine funktionale Klasse $Ł : (APs \times APs) \to \mathscr{P}(Ł^{\mu_{BV}})$ mit

$$(A,B) \;\mapsto\; \Gamma \subseteq Ł_{<A,B>}^{\mu_{BV}}$$

Wir schreiben auch $Ł_{<A,B>}$ anstatt $Ł(A,B)$.

Notation 7.2.14:

Für eine μ-Formelklasse Ł sowie einer $\mathcal{T}$-Annotation $\mathfrak{a}$ eines beliebigen Transitionssystem $\mathcal{T}$ schreiben wir auch $Ł_{<\mathfrak{a}>}$ für $L_{<im(\mathfrak{a}_S),im(\mathfrak{a}_T)>}$.

Eine μ-Formelklasse ist formal eine funktionale Klasse zur Abbildung zweier Mengen atomarer Propositionen auf eine beliebige Menge entsprechender μ-Formeln. In der Regel wird es so sein, dass μ-Formeln verschiedener Bilder einer μ-Formelklasse sich in ihrer Struktur nicht unterscheiden. Eine μ-Formelklasse kann man sich somit vorsichtig als eine Menge von "Formel-Schablonen" vorstellen, welche zu passender Zeit mit gewünschten atomaren Propositionen "instanziiert" werden. Dieses Vorgehen garantiert uns eine strukturierte Vorgehensweise ohne Verwendung echter Klassen.
Wir werden nun den Begriff der μ-Semantik aufbauend auf μ-Formelklasse definieren. Eine μ-Semantik wird dabei eine Abbildungen von Zustandsgraphen in die Menge der implizierten μ-Formeln der zugehörigen μ-Formelklasse darstellen. Anschließend werden wir zeigen, dass μ-Semantiken invariant sind unter Isomorphie.

Definition 7.2.15: [μ-Semantik]

Als *μ-Semantik einer μ-Formelklasse* Ł bezeichnen wir die funktionale Klasse des Typs $TS[*]\text{-}A \to \mathscr{P}(Ł^{\mu_{BV}})$ mit

$$\mathcal{T}_{<\mathfrak{a}>}[s_0] \mapsto \{\varphi \in Ł_{<\mathfrak{a}>} \mid \mathcal{T}_{<\mathfrak{a}>}[s_0] \vDash \varphi\}$$

Lemma 7.2.16:

Seien $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ annotierte Zustandsgraphen mit

$$\mathcal{T}_{<\mathfrak{a}>}[s_0] \approx \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$$

dann gilt $Ł^{\mu}_{<\mathfrak{a}>} = Ł^{\mu}_{<\mathfrak{a}'>}$ und es folgt für beliebiges $\varphi \in L^{\mu}_{<\mathfrak{a}>}$ sowie für beliebige $\beta : \{v_i\}_{i\in\mathbb{N}} \to \mathscr{P}(S^{(\mathcal{T})})$ und $\beta' : \{v_i\}_{i\in\mathbb{N}} \to \mathscr{P}(S^{(\mathcal{T}')})$ mit $f(\beta(v_i)) = \beta'(v_i)$, dass

$$\mathcal{T}_{<\mathfrak{a}>}[s_0] \vDash \varphi[\beta] \text{ gdw. } \mathcal{T}'_{<\mathfrak{a}'>}[s'_0] \vDash \varphi[\beta']$$

Beweis.
Zunächst folgt aus der Isomorphie von $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ die Existenz eines zugehörigen Isomorphismus $f : S^{(\mathcal{T})} \to S^{(\mathcal{T}')}$. Zudem ergibt sich $im(\mathfrak{a}_S) = im(\mathfrak{a}'_S)$ genau wie $im(\mathfrak{a}_T) = im(\mathfrak{a}'_T)$ und folglich auch $Ł^{\mu}_{<\mathfrak{a}>} = Ł^{\mu}_{<\mathfrak{a}'>}$.

Sei nun $\varphi \in L^{\mu}_{<\mathfrak{a}>}$ beliebig sowie $\beta : \{v_i\}_{i\in\mathbb{N}} \to \mathscr{P}(S^{(\mathcal{T})})$ und $\beta' : \{v_i\}_{i\in\mathbb{N}} \to \mathscr{P}(S^{(\mathcal{T}')})$ beliebig mit $f(\beta(v_i)) = \beta'(v_i)$.

Wir zeigen die Behauptung per struktureller Induktion über den Formelaufbau von φ. Man dabei insbesondere Bemerkung 4.1.31.

Gelte die Behauptung nun für alle echten Teilformeln von φ.

- $[\varphi = a \in A]$

  $s \in \| a \|_{\beta}^{\mathcal{T}_{<\mathfrak{a}>}} \Leftrightarrow \mathfrak{a}_S(s) = a \overset{[f \text{ Iso}]}{\Leftrightarrow} \mathfrak{a}'_S(f(s)) = a \Leftrightarrow f(s) \in \| a \|_{\beta'}^{\mathcal{T}'_{<\mathfrak{a}'>}}$ ✓

- $[\varphi = \psi_1 \overset{b}{\to} \psi_2]$

  $s \in \| \psi_1 \overset{b}{\to} \psi_2 \|_{\beta}^{\mathcal{T}_{<\mathfrak{a}>}} \Leftrightarrow s \in \{s' \in \| \psi_1 \|_{\beta}^{\mathcal{T}_{<\mathfrak{a}>}} \mid \exists s'' \in \| \psi_2 \|_{\beta}^{\mathcal{T}_{<\mathfrak{a}>}} \colon s' \to s'' \in T^{(\mathcal{T})} \wedge \mathfrak{a}_T(s' \to s'') = b\} \overset{[IV]}{\Leftrightarrow} f(s) \in \{s' \in \| \psi_1 \|_{\beta'}^{\mathcal{T}'_{<\mathfrak{a}'>}} \mid \exists s'' \in \| \psi_2 \|_{\beta'}^{\mathcal{T}'_{<\mathfrak{a}'>}} \colon s' \to s'' \in T^{(\mathcal{T}')} \wedge \mathfrak{a}'_T(s' \to s'') = b\} \Leftrightarrow f(s) \in \| \psi_1 \overset{b}{\to} \psi_2 \|_{\beta'}^{\mathcal{T}'_{<\mathfrak{a}'>}}$ ✓

- $[\varphi = v_i]$

  $s \in \| v_i \|_{\beta}^{\mathcal{T}_{<\mathfrak{a}>}} \Leftrightarrow s \in \beta(v_i) \Leftrightarrow f(s) \in f(\beta(v_i)) \overset{[Vor]}{\Leftrightarrow} f(s) \in \beta'(v_i) \Leftrightarrow s \in \| v_i \|_{\beta'}^{\mathcal{T}'_{<\mathfrak{a}'>}}$ ✓

- $[\varphi = \mu v_i.\psi]$

  $s \in \| \mu v_i.\psi \|_{\beta}^{\mathcal{T}_{<\mathfrak{a}>}} \Leftrightarrow s \in \bigcap\{S \subseteq S^{(\mathcal{T})} \mid \| \psi \|_{\beta[v_i:=S]}^{\mathcal{T}_{<\mathfrak{a}>}} \subseteq S\} \overset{[IV]}{\Leftrightarrow} f(s) \in \bigcap\{S \subseteq S^{(\mathcal{T}')} \mid \| \psi \|_{\beta'[v_i:=S]}^{\mathcal{T}'_{<\mathfrak{a}'>}} \subseteq S\} \Leftrightarrow f(s) \in \| \mu v_i.\psi \|_{\beta'}^{\mathcal{T}'_{<\mathfrak{a}'>}}$ ✓

- $[\varphi = \neg\psi]$

  $s \in \| \neg\psi \|_{\beta}^{\mathcal{T}_{<\mathfrak{a}>}} \Leftrightarrow s \in S^{(\mathcal{T})} \setminus \| \psi \|_{\beta}^{\mathcal{T}_{<\mathfrak{a}>}} \overset{[IV]}{\Leftrightarrow} f(s) \in S^{(\mathcal{T}')} \setminus \| \psi \|_{\beta'}^{\mathcal{T}'_{<\mathfrak{a}'>}} \Leftrightarrow f(s) \in \| \neg\psi \|_{\beta'}^{\mathcal{T}'_{<\mathfrak{a}'>}}$ ✓

- $[\varphi = \psi_1 \wedge \psi_2]$

  $s \in \| \psi_1 \wedge \psi_2 \|_{\beta}^{\mathcal{T}_{<\mathfrak{a}>}} \Leftrightarrow s \in \| \psi_1 \|_{\beta}^{\mathcal{T}_{<\mathfrak{a}>}} \cup \| \psi_2 \|_{\beta}^{\mathcal{T}_{<\mathfrak{a}>}} \overset{[IV]}{\Leftrightarrow} f(s) \in \| \psi_1 \|_{\beta'}^{\mathcal{T}'_{<\mathfrak{a}'>}} \cup \| \psi_2 \|_{\beta'}^{\mathcal{T}'_{<\mathfrak{a}'>}} \Leftrightarrow f(s) \in \| \psi_1 \wedge \psi_2 \|_{\beta'}^{\mathcal{T}'_{<\mathfrak{a}'>}}$ ✓

□

Proposition 7.2.17:

Sei $\mathfrak{s}$ eine beliebige μ-Semantik. Seien weiter $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ annotierte Zustandsgraphen mit

$$\mathcal{T}_{<\mathfrak{a}>}[s_0] \approx \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$$

dann gilt

$$\mathcal{T}_{<\mathfrak{a}>}[s_0] =_{\mathfrak{s}} \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$$

Beweis.
Sei $Ł^{\mathfrak{s}}$ die zu $\mathfrak{s}$ gehörige μ-Formelklasse. Aufgrund der Isomorphie von $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ gilt $Ł^{\mathfrak{s}}_{<\mathfrak{a}>} = Ł^{\mathfrak{s}}_{<\mathfrak{a}'>}$.Außerdem gilt $Ł^{\mathfrak{s}}_{<\mathfrak{a}>} \subseteq Ł^{\mu}_{<\mathfrak{a}>}$.
Sei nun $\varphi \in Ł^{\mathfrak{s}}_{<\mathfrak{a}>}$ beliebig. Laut Lemma 7.2.16 gilt für beliebiges $\beta : \{v_i\}_{i\in\mathbb{N}} \to \mathscr{P}(S^{(\mathcal{T})})$ sowie $\beta' : \{v_i\}_{i\in\mathbb{N}} \to \mathscr{P}(S^{(\mathcal{T}')})$ mit $v_i \mapsto f(\beta(v_i))$, dass

$$\mathcal{T}_{<\mathfrak{a}>}[s_0] \vDash \varphi[\beta] \text{ gdw. } \mathcal{T}'_{<\mathfrak{a}'>}[s'_0] \vDash \varphi[\beta']$$

und folglich auch

$$\mathcal{T}_{<\mathfrak{a}>}[s_0] \vDash \varphi \text{ gdw. } \mathcal{T}'_{<\mathfrak{a}'>}[s'_0] \vDash \varphi$$

R'_{redMax} als die größte probabilistische $prbRed(\mathcal{T}'_{<\mathfrak{a}'>}[s'_0])$-Bisimulation und somit insbesondere die Behauptung.

□

Beispiel - pTr-Semantik

In diesem Abschnitt werden wir uns mit der μ-Semantik pTr befassen. Wir möchten diese zunächst formal vorstellen und anschließend erläutern. Insbesondere werden wir später in Abschnitt 7.3 feststellen, dass der durch pTr induzierte Beobachter und der $pTrace$ Beobachter aus Beispiel 4.2.3 gleichstarke Beobachter definieren.

Definition 7.2.18: [pTr-Semantik]

Wir definieren die μ-Formelklasse $Ł^{pTr}$ durch

$$(A,B) \;\mapsto\; Lng(\; \varphi := a \mid a \xrightarrow{b} \varphi \;)$$

Die zugehörige μ-Semantik bezeichnen wir mit pTr.

Bemerkung 7.2.19:

Man erinnere sich, dass $Lng(G)$ gerade die Sprache der durch G ausgedrückten Grammatik beschreibt (vgl. hierzu auch Definition 2.0.40).

Bemerkung 7.2.20:

$Ł^{pTr}$ beinhaltet für zwei beliebige Mengen atomarer Proposition gerade die μ-Formeln, welche sich ausschließlich anhand der Regeln (1) und einer Einschränkung von Regel (2) aus Definition 7.2.1 ableiten lassen.

Wir möchten als nächstes vollkommen unabhängig hiervon definieren was es für zwei Zustandsgraphen heißt $pTrace$-äquivalent zu sein. Dies ist nämlich genau dann der Fall, wenn die entsprechenden Zustandsgraphen Pfade mit selben Annotationen besitzen. Anschließend werden wir beweisen, dass $pTrace$-Äquivalenz und pTr-Gleichheit identifizierbar sind, was insbesondere Notation 7.2.22 motiviert. Da die $pTrace$-Äquivalenz einen anschaulichen Charakter vorweist, erhalten wir so auch einen anschaulichen Zugang zur pTr-Semantik.

Definition 7.2.21: [pTrace-Äquivalenz]

Wir bezeichnen zwei Zustandsgraphen $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ als *$pTrace$-äquivalent* gdw.

$$\begin{aligned}&\{\ (a_0, b_0, a_1, b_1, \dots) \mid \exists \pi \in \mathcal{T}[s_0]^{(*)} :\ a_i = \mathfrak{a}_S(\pi_i),\ b_i = \mathfrak{a}_T(\pi_i \to \pi_{i+1})\}\\ =\ &\{\ (a_0, b_0, a_1, b_1, \dots) \mid \exists \pi \in \mathcal{T}'[s'_0]^{(*)} :\ a_i = \mathfrak{a}_S(\pi_i),\ b_i = \mathfrak{a}_T(\pi_i \to \pi_{i+1})\}\end{aligned}$$

Notation 7.2.22:

Für $pTrace$-äquivalente Zustandsgraphen $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ schreiben wir auch $\mathcal{T}_{<\mathfrak{a}>}[s_0] \approx_{pTr} \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$

Proposition 7.2.23:

Seien $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ zwei Zustandsgraphen, dann gilt

$$\mathcal{T}_{<\mathfrak{a}>}[s_0] \approx_{pTr} \mathcal{T}'_{<\mathfrak{a}'>}[s'_0] \ \Leftrightarrow\ \mathcal{T}_{<\mathfrak{a}>}[s_0] =_{pTr} \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$$

Beweis.
Sei

- $Tr := \{\ (a_0, b_0, a_1, b_1, \dots) \mid \exists \pi \in \mathcal{T}[s_0]^{(*)} :\ a_i = \mathfrak{a}_S(\pi_i),\ b_i = \mathfrak{a}_T(\pi_i \to \pi_{i+1})\}$
- $Tr' := \{\ (a_0, b_0, a_1, b_1, \dots) \mid \exists \pi \in \mathcal{T}'[s'_0]^{(*)} :\ a_i = \mathfrak{a}_S(\pi_i),\ b_i = \mathfrak{a}_T(\pi_i \to \pi_{i+1})\}$

Wir zeigen $(a_0, b_0, a_1, b_1, \dots) \in Tr$ gdw. $a_0 \overset{b_0}{\to} a_1 \overset{b_1}{\to} \dots \in pTr(\mathcal{T}_{<\mathfrak{a}>}[s_0])$. Analog ergibt sich $(a_0, b_0, a_1, b_1, \dots) \in Tr'$ gdw. $a_0 \overset{b_0}{\to} a_1 \overset{b_1}{\to} \dots \in pTr(\mathcal{T}'_{<\mathfrak{a}'>}[s'_0])$. Da die der μ-Formelklasse pTr zugeordneten Formeln stets von der Form "$a_0 \overset{b_0}{\to} a_1 \overset{b_1}{\to} \dots$" sind folgt anschließend die Behauptung.
Wir zeigen die Implikationen " $\Rightarrow$ " und " $\Leftarrow$ " getrennt.

- [" $\Rightarrow$ "]

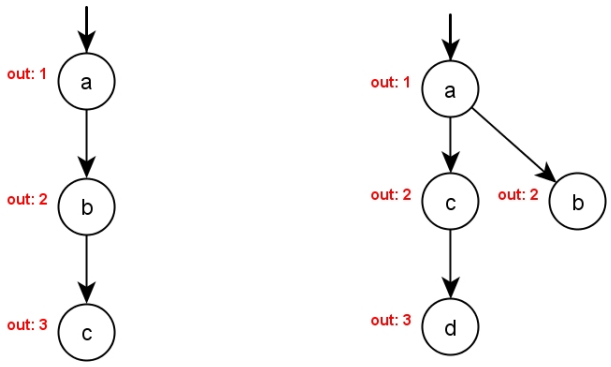


Abbildung 7.5: $pTrace$-äquivalente / pTr-gleiche Zustandsgraphen

Sei $(a_0, b_0, \ldots, b_{n-2}, a_{n-1}) \in Tr$. Per Definition existiert ein $\pi \in {}_{(s_0)}\mathcal{T}^{(n)}$ mit $\mathfrak{a}_S(\pi_i) = a_i$ und $\mathfrak{a}_T(\pi_i \to \pi_{i+1}) = b_i$. Offensichtlich gilt dann $\mathcal{T}_{<\mathfrak{a}>}[\pi_i] \vDash a_i \overset{b_i}{\to} \ldots \overset{b_{n-2}}{\to} a_{n-1}$ und somit insbesondere $\mathcal{T}_{<\mathfrak{a}>}[s_0] \vDash a_0 \overset{b_0}{\to} \ldots \overset{b_{n-2}}{\to} a_{n-1}$. ✓

- [“ $\Leftarrow$ “]

  Sei $a_0 \overset{b_0}{\to} \ldots \overset{b_{n-2}}{\to} a_{n-1} \in pTr(\mathcal{T}_{<\mathfrak{a}>}[s_0])$. Es folgt die Existenz eines Pfads $\pi \in {}_{(s_0)}\mathcal{T}^{(n)}$ mit $\mathcal{T}_{<\mathfrak{a}>}[\pi_i] \vDash a_i \overset{b_i}{\to} \ldots \overset{b_{n-2}}{\to} a_{n-1}$. Folglich gilt $\mathfrak{a}_S(\pi_i) = a_i$ und $\mathfrak{a}_T(\pi_i \to \pi_{i+1}) = b_i$ und somit auch $(a_0, b_0, \ldots, b_{n-2}, a_{n-1}) \in Tr$. ✓

□

Beispiel - $bisim$-Semantik

In diesem Abschnitt werden wir uns mit der μ-Semantik $bisim$ befassen. Wir möchten diese zunächst formal vorstellen und anschließend erläutern.

Definition 7.2.24: $[bisim]$

Wir definieren die μ-Formelklasse $Ł^{bisim}$ durch

$$(A,B) \;\mapsto\; Lng(\;\varphi \;:=\; a \mid \varphi \overset{b}{\to} \varphi \mid\; \neg\varphi \mid \varphi \wedge \varphi)$$

Die zugehörige μ-Semantik bezeichnen wir mit $bisim$.

Bemerkung 7.2.25:

Man erinnere sich, dass $Lng(G)$ gerade die Sprache der durch G ausgedrückten Grammatik beschreibt (vgl. hierzu auch Definition 2.0.40).

Bemerkung 7.2.26:

Ł^{bisim} beinhaltet für zwei beliebige Mengen atomarer Proposition gerade die μ-Formeln welche sich ausschließlich anahand der Regeln (1) bis (4) aus Definition 7.2.1 ableiten lassen lassen. Diese Teilklasse von Ł$^{\mu}$ sind auch als Formeln der *Henessy-Milner-Logic (HML)* bekannt.

Wir möchten als nächstes vollkommen unabhängig hiervon definieren was es für zwei Zustandsgraphen heißt "bisimilar" zu sein. Dies ist nämlich genau dann der Fall, wenn sich entsprechenden Zustandsgraphen wechselseitig schrittweise simulieren können. Anschließend werden wir beweisen, dass Bisimilarität und *bisim*-Gleichheit dasselbe beschreibt. Da Bisimilarität einen anschaulichen Charakter vorweist, erhalten wir so auch einen anschaulichen Zugang zur *bisim* Semantik.

Definition 7.2.27: [Bisimilarität]

Wir bezeichnen zwei Zustandsgraphen $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ als *bisimilar* gdw. eine Relation $R \subseteq S^{(\mathcal{T})} \times S^{(\mathcal{T}')}$ existiert mit $(s_0,s'_0) \in R$ für welche $(s,s') \in R$ gdw.

1. $\mathfrak{a}_S(s) = \mathfrak{a}'_S(s')$
2. $s \to t \in T^{(\mathcal{T})} \Rightarrow \exists t' : s' \to t' \in T^{(\mathcal{T}')} \wedge \mathfrak{a}_T(s \to t) = \mathfrak{a}'_T(s' \to t') \wedge (t,t') \in R$
3. $s' \to t' \in T^{(\mathcal{T}')} \Rightarrow \exists t : s \to t \in T^{(\mathcal{T})} \wedge \mathfrak{a}'_T(s' \to t') = \mathfrak{a}_T(s \to t) \wedge (t,t') \in R$

Wir bezeichnen R dann auch als *Bisimulation zwischen* $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ *und* $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ und schreiben

$$\mathcal{T}_{<\mathfrak{a}>}[s_0] \approx_{bisim} \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$$

Bemerkung 7.2.28:

Für eine Bisimulation $R \subseteq S^{(\mathcal{T})} \times S^{(\mathcal{T}')}$ zwischen Zustandsgraphen $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ ist $R^{-1} \subseteq S^{(\mathcal{T}')} \times S^{(\mathcal{T})}$ eine Bisimulation zwischen $\mathcal{T}'_{<\mathfrak{a}'>}[s_0]$ und $\mathcal{T}_{<\mathfrak{a}>}[s'_0]$. Insbesondere gilt

$$\mathcal{T}_{<\mathfrak{a}>}[s_0] \approx_{bisim} \mathcal{T}'_{<\mathfrak{a}'>}[s'_0] \text{ gdw. } \mathcal{T}'_{<\mathfrak{a}'>}[s'_0] \approx_{bisim} \mathcal{T}_{<\mathfrak{a}>}[s_0]$$

Lemma 7.2.29:

Seien $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ Zustandsgraphen mit $\mathcal{T}_{<\mathfrak{a}>}[s_0] \approx_{bisim} \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ bzgl. einer Bisimulation $R \subseteq S^{(\mathcal{T})} \times S^{(\mathcal{T}')}$, dann gilt auch für jedes $(s,s') \in R$, dass

$$\mathcal{T}_{<\mathfrak{a}>}[s] \approx_{bisim} \mathcal{T}'_{<\mathfrak{a}'>}[s']$$

Beweis.
Die Behauptung ergibt sich unmittelbar für R als entsprechende Bisimulation.

□

Bemerkung 7.2.30:

Man beachte die parallelen zwischen Bisimulationen und probabilistischen Bisimulationen aus Beispiel 7.1.16. Probabilistische Bisimulationen kann man als spezifische Konkretisierungen von Bisimulationen eigens für probabilistische Systeme auffassen.

Proposition 7.2.31:

Seien $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ zwei annotierte Zustandsgraphen, dann gilt

$$\mathcal{T}_{<\mathfrak{a}>}[s_0] \approx_{bisim} \mathcal{T}'_{<\mathfrak{a}'>}[s'_0] \;\Leftrightarrow\; \mathcal{T}_{<\mathfrak{a}>}[s_0] =_{bisim} \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$$

Beweis.
Wir zeigen die Implikationen " $\Rightarrow$ " und " $\Leftarrow$ " getrennt.

- $[\Rightarrow]$

  Zunächst einmal existiert laut Voraussetzung eine Bisimulation $R \subseteq S^{(\mathcal{T})} \times S^{(\mathcal{T}')}$ zwischen $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$.

  Sei nun $\varphi \in bisim(LT_{<\mathfrak{a}>}[s_0])$ beliebig. Wir zeigen per struktureller Induktion über den Formelaufbau von φ, dass auch $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0] \vDash \varphi$ und somit $\varphi \in bisim(\mathcal{T}'_{<\mathfrak{a}'>}[s'_0])$. Hieraus ergibt sich anschließend $Bism(\mathcal{T}_{<\mathfrak{a}>}[s_0]) \subseteq bisim(\mathcal{T}'_{<\mathfrak{a}'>}[s'_0])$. Analog lässt sich beweisen, dass $Bism(\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]) \subseteq bisim(\mathcal{T}_{<\mathfrak{a}>}[s_0])$. Zusammenfassend ergibt sich dann die Behauptung.

  Gelte die Behauptung also für alle echten Teilformeln von φ.

  - $[\varphi = a]$

    Es gilt $\mathcal{T}_{<\mathfrak{a}>}[s_0] \vDash a$, d.h. $\mathfrak{a}_S(s_0) = a$. Laut Voraussetzung gilt $(s_0,s'_0) \in R$ und folglich $\mathfrak{a}'_S(s'_0) = a$. Es ergibt sich $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0] \vDash a$. ✓

  - $[\varphi = \psi_1 \overset{b}{\rightarrow} \psi_2]$

    Es gilt $\mathcal{T}_{<\mathfrak{a}>}[s_0] \vDash \psi_1 \overset{b}{\rightarrow} \psi_2$, d.h. $\mathcal{T}_{<\mathfrak{a}>}[s_0] \vDash \psi_1$ und es existiert ein $s \in S^{(\mathcal{T})}$ mit $s_0 \rightarrow s \in S^{(\mathcal{T})}$ und $\mathfrak{a}_T(s_0 \rightarrow s) = b$ sowie $\mathcal{T}_{<\mathfrak{a}>}[s] \vDash \psi_2$. Laut Voraussetzung existiert aufgrund von $(s_0,s'_0) \in R$ auch ein $s' \in S^{(\mathcal{T}')}$ mit $s'_0 \rightarrow s' \in T^{(\mathcal{T}')}$ und $\mathfrak{a}'_T(s'_0 \rightarrow s') = b$ sowie $(s,s') \in R$. Laut Induktionsvoraussetzung folgt zusammen mit Lemma 7.2.29 sowohl $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0] \vDash \psi_1$ genau wie $\mathcal{T}'_{<\mathfrak{a}'>}[s'] \vDash \psi_2$. Zusammenfassend ergibt sich $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0] \vDash \psi_1 \overset{b}{\rightarrow} \psi_2$. ✓

– $[\varphi = \neg\psi]$

Es gilt $\mathcal{T}_{<\mathfrak{a}>}[s_0] \vDash \neg\psi$, d.h. $\mathcal{T}_{<\mathfrak{a}>}[s_0] \nvDash \psi$. Laut Induktionsvoraussetzung gilt dann auch $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0] \nvDash \psi$. Es ergibt sich $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0] \vDash \neg\psi$. ✓

– $[\varphi = \psi_1 \wedge \psi_2]$

Es gilt $\mathcal{T}_{<\mathfrak{a}>}[s_0] \vDash \psi_1 \wedge \psi_2$, d.h. es gilt sowohl $\mathcal{T}_{<\mathfrak{a}>}[s_0] \vDash \psi_1$ als auch $\mathcal{T}_{<\mathfrak{a}>}[s_0] \vDash \psi_2$. Laut Induktionsvoraussetzung gilt dann auch $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0] \vDash \psi_1$ genau wie $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0] \vDash \psi_2$. Es ergibt sich $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0] \vDash \psi_1 \wedge \psi_2$. ✓

- $[\Leftarrow]$

Wir zeigen hierfür $R \subseteq S^{(\mathcal{T})} \times S^{(\mathcal{T}')}$ mit $(s,t) \in R$ gdw. $\mathcal{T}_{<\mathfrak{a}>}[s] =_{bisim} \mathcal{T}'_{<\mathfrak{a}'>}[t]$ ist eine Bisimulation zwischen $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$.

Sei $(s,s') \in R$. Wir müssen die Gültigkeit folgender Punkte nachweisen.

– $[(s_0,s'_0) \in R]$

Folgt direkt aus der Voraussetzung. ✓

– $[\mathfrak{a}_S(s) = \mathfrak{a}'_S(s')]$

Gelte $\mathfrak{a}_S(s) = a$. Es folgt $\mathcal{T}_{<\mathfrak{a}>}[s] \vDash a$ und da $(s,s') \in R$ auch $\mathcal{T}'_{<\mathfrak{a}'>}[s'] \vDash a$. Es ergibt sich $\mathfrak{a}'_S(s') = a$ und somit insbesondere $\mathfrak{a}_S(s) = \mathfrak{a}'_S(s')$. ✓

– $[s \to t \in T^{(\mathcal{T})} \Rightarrow \ \exists t' : \ s' \to t' \in T^{(\mathcal{T}')} \ \wedge \ \mathfrak{a}_T(s \to t) = \mathfrak{a}'_T(s' \to t') \ \wedge \ (t,t') \in R]$

Gelte $s \to t \in T^{(\mathcal{T})}$ mit $\mathfrak{a}_T(s \to t) = b$. Wir zeigen die Existenz eines $t' \in S^{(\mathcal{T}')}$ mit $\mathfrak{a}_T(s \to t) = \mathfrak{a}'_T(s' \to t')$ sowie $\mathcal{T}_{<\mathfrak{a}>}[t] \subseteq_{bisim} \mathcal{T}'_{<\mathfrak{a}'>}[t']$.

Anschließend ergibt sich $\mathcal{T}_{<\mathfrak{a}>}[s] =_{bisim} \mathcal{T}'_{<\mathfrak{a}'>}[s']$ und folglich auch die Behauptung, denn sei $\varphi \in bisim(\mathcal{T}'_{<\mathfrak{a}'>}[t'])$. Angenommen $\varphi \notin bisim(\mathcal{T}_{<\mathfrak{a}>}[t])$, dann wäre $\neg\varphi \in bisim(\mathcal{T}_{<\mathfrak{a}>}[t])$ und folglich auch $\neg\varphi \in bisim(\mathcal{T}'_{<\mathfrak{a}'>}[t'])$ im Widerspruch zu $\varphi \in bisim(\mathcal{T}'_{<\mathfrak{a}'>}[t'])$.

Hierfür definieren wir zunächst

$$U' := \{u' \in S^{(\mathcal{T}')} \mid s' \to u' \in T^{(\mathcal{T}')}, \mathfrak{a}'_T(s' \to u') = b, \mathcal{T}_{<\mathfrak{a}>}[t] \notin_{bisim} \mathcal{T}'_{<\mathfrak{a}'>}[u']\}$$

Man beachte, dass da $\mathcal{T}'$ endlich verzweigt U' endlich ist. Für jedes $u' \in U'$ existiert nun ein $\varphi_{u'} \in bisim(\mathcal{T}_{<\mathfrak{a}>}[t])$ mit $\mathcal{T}_{<\mathfrak{a}>}[t] \vDash \varphi_{u'}$ und $\mathcal{T}'_{<\mathfrak{a}'>}[u'] \nvDash \varphi_{u'}$. Folglich gilt

$$\mathcal{T}_{<\mathfrak{a}>}[s] \vDash \top \overset{b}{\to} \bigwedge_{u' \in U'} \varphi_{u'}$$

und laut Voraussetzung somit auch

$$\mathcal{T}'_{<\mathfrak{a}'>}[s'] \vDash \top \overset{b}{\to} \bigwedge_{u' \in U} \varphi_{u'}$$

wobei $\top := \neg(a \wedge \neg a)$ eine stets erfüllte μ-Formel darstellt.

Es folgt nun die Existenz eines $t' \in S^{(\mathcal{T}')}$ mit $s' \to t' \in T^{(\mathcal{T})}$ sowie $\mathfrak{a}'_T(s' \to t') = b$ und $t' \notin U'$, d.h. $\mathcal{T}_{<\mathfrak{a}>}[t] \subseteq_{bisim} \mathcal{T}'_{<\mathfrak{a}'>}[t']$. ✓

– $[s' \to t' \in T^{(\mathcal{T}')} \;\Rightarrow\; \exists t:\; s \to t \in T^{(\mathcal{T})} \quad \wedge\; \mathfrak{a}'_T(s' \to t') = \mathfrak{a}_T(s \to t) \;\wedge\; (t,t') \in R]$

Folgt analog. ✓

□

Bemerkung 7.2.32:

Der Beweis von Proposition 7.2.31 ist der Grund für die Einschränkung unserer Betrachtungen auf endliche verzweigte Systeme. Man könnte dies umgehen indem man unendliche Konjugationen zulässt, also eine Regel der Form $\bigwedge_{m\in M} \varphi_m$ für beliebige Mengen M, mit in den modalen μ-Kalkül integriert. Dies hätte jedoch zur Folge, dass Bilder von μ-Formelklassen und folglich auch Bilder von μ-Semantiken sich möglicherweise zu echten Klassen ausweiten würden, was aus fundamental mathematischen Gründen zu Konflikten mit der Beobachterdefinition führen würde. Wir möchten uns an dieser Stelle mit dem Gedanken zufriedengeben, dass eine Verallgemeinerung auf Kosten der Übersichtlichkeit möglich wäre. Weiterhin möchten wir lediglich ohne nähere Erläuterung anmerken, dass man ebenfalls auf Kosten der Übersichtlichkeit Regeln der Form $\bigwedge_{m\in M} \varphi_m$ für Mengen M mit eingeschränkten Kardinalitäten, d.h. für $|M| < \aleph_i$ für festes aber beliebiges $\aleph_i \in card$ in den modalen μ-Kalkül integrieren könnte, ohne auf den Klassen-Begriff ausweichen zu müssen. Man könnte also die Einschränkung "endliche verzweigt" auf "$\aleph_i$ verzweigt" auflockern ohne eine fundamentale Überarbeitung der bisherigen Theorie. Wir möchten jedoch im Rahmen dieser Arbeit aufgrund der mäßigen Relevanz auf dieses verzichten.

Wir wollen abschließend für spätere Zwecke beweisen, dass Bisimilariät $pTrace$-Äquivalenz impliziert.

Proposition 7.2.33:

Seien $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ beliebige Zustandsgraphen. Es gilt

$$\mathcal{T}_{<\mathfrak{a}>}[s_0] \approx_{bisim} \mathcal{T}'_{<\mathfrak{a}'>}[s'_0] \;\Rightarrow\; \mathcal{T}_{<\mathfrak{a}>}[s_0] \approx_{pTr} \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$$

Beweis.
Seien $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ beliebige Zustandsgraphen mit $\mathcal{T}_{<\mathfrak{a}>}[s_0] \approx_{bisim} \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ bzgl. einer Bisimulation $R \subseteq S^{(\mathcal{T})} \times S^{(\mathcal{T}')}$. Wir müssen für

- $Tr := \{\; (a_0, b_0, a_1, b_1, \dots) \mid \exists \pi \in \mathcal{T}[s_0]^{(*)}:\; a_i = \mathfrak{a}_S(\pi_i),\; b_i = \mathfrak{a}_T(\pi_i \to \pi_{i+1})\}$

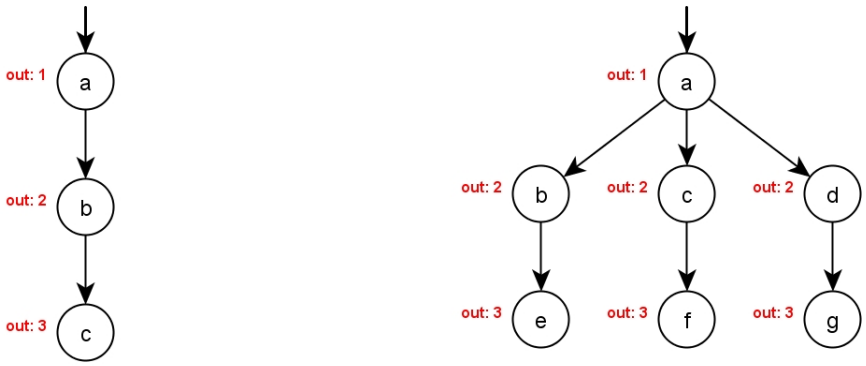


Abbildung 7.6: bisimilare / *bisim*-gleiche Zustandsgraphen

- $Tr' := \{\ (a_0, b_0, a_1, b_1, \dots) \mid \exists \pi \in \mathcal{T}'[s_0']^{(*)} :\ a_i = \mathfrak{a}_S(\pi_i),\ b_i = \mathfrak{a}_T(\pi_i \to \pi_{i+1})\}$

zeigen, dass $Tr = Tr'$.

Wir zeigen hierfür lediglich die Inklusion " $\subseteq$ ", die Inklusion " $\supseteq$ " ergibt sich anlog und es folgt anschließend die Behauptung.

Sei also $(a_0, b_0, \dots, b_{n-2}, a_{n-1}) \in Tr$. Es existiert ein $\pi \in \mathcal{T}[s_0]^{(n)}$ mit $a_i = \mathfrak{a}_S(\pi_i)$ sowie $b_i = \mathfrak{a}_T(\pi_i \to \pi_{i+1})$. Wir konstruieren nun induktiv ein $\pi' \in \mathcal{T}'[s_0']^{(n)}$ mit $(\pi_i, \pi_i') \in R$ sowie $\mathfrak{a}_S'(\pi_i') = a_i$ und $\mathfrak{a}_T'(\pi_i' \to \pi_{i+1}) = b_i$. Anschließend folgt $(a_0, b_0, \dots, b_{n-2}, a_{n-1}) \in Tr'$ und folglich auch die Behauptung.

- $[i = 0]$

  Wir setzen $\pi_0' := s_0'$. Da $(s_0, s_0') \in R$ gilt offensichtlich $\mathfrak{a}_S'(\pi_0) = a_0$.

- $[i \to i+1, i < n]$

  Laut "Induktionsvoraussetzung" gilt $(\pi_i, \pi_i') \in R$. Da $\pi_i \to \pi_{i+1} \in T^{(\mathcal{T})}$ existiert per Definition von R ein $s' \in S^{(\mathcal{T})}$ mit $\pi_i' \to s' \in T^{(\mathcal{T})}$ sowie $\mathfrak{a}_T(\pi_i \to \pi_{i+1}) = \mathfrak{a}_T'(\pi_i' \to s')$ und $(\pi_{i+1}, s') \in R$, woraus sich insbesondere auch $\mathfrak{a}_S'(s') = a_{i+1}$ ergibt. Wir setzten daher $\pi_{i+1}' := s'$.

□

Bemerkung 7.2.34:

Die Rückrichtung von Proposition 7.2.33 gilt im Allgemeinen nicht. Man betrachte hierfür beispielsweise Abbildung 7.5.

7.3 $\mathfrak{F}$-Instanzen

Ziel dieses Abschnitts stellt die Definition der Framework-Instanz dar. Framework-Instanzen werden dabei Beobachter darstellen, welche durch Zustandsgraph-Semantik charakterisiert sind.

Definition 7.3.1: [$\mathfrak{F}$-Instanz]

Sei $\mathfrak{s}$ eine μ-Semantik sowie *adap* eine Adaption. Als $\mathfrak{F}$-*Instanz* $\mathfrak{F}^{\ll adap,\mathfrak{s}\gg}$ bezeichnen wir einen Beobachter der Form

$$(\mathcal{S},\ p,\ [\mathfrak{b}]) \mapsto \begin{cases} (\mathfrak{s} \circ adap)(\mathfrak{b}) & : \ \mathfrak{b} \in dom(adap) \\ \blacksquare & : \text{sonst} \end{cases}$$

Notation 7.3.2:

Wir schreiben auch $\mathfrak{F}^{\ll\mathfrak{s}\gg}$ für eine $\mathfrak{F}$-Instanz der Form $\mathfrak{F}^{\ll id_{\mathfrak{B}},\mathfrak{s}\gg}$.

Proposition 7.3.3:

$\mathfrak{F}$-Instanzen sind wohldefiniert

Beweis.
Sei $\mathfrak{F}^{\ll adap,\mathfrak{s}\gg}$ eine beliebige $\mathfrak{F}$-Instanz. Sei weiter $(\mathcal{S},\ p,\ [\mathfrak{b}]),(\mathcal{S}',\ p,\ [\mathfrak{b}']) \in \mathfrak{AK}$ beliebige Beobachter-Kontexte mit $(\mathcal{S},\ p,\ [\mathfrak{b}]) = (\mathcal{S},\ p,\ [\mathfrak{b}'])$.
Es gilt

$$\mathfrak{F}^{\ll adap,\mathfrak{s}\gg}((\mathcal{S},\ p,\ [\mathfrak{b}])) = \begin{cases} (\mathfrak{s} \circ adap)(\mathfrak{b}) & : \ \mathfrak{b} \in dom(adap) \\ \blacksquare & : \text{sonst} \end{cases}$$

sowie

$$\mathfrak{F}^{\ll adap,\mathfrak{s}\gg}((\mathcal{S},\ p,\ [\mathfrak{b}'])) = \begin{cases} (\mathfrak{s} \circ adap)(\mathfrak{b}') & : \ \mathfrak{b}' \in dom(adap) \\ \blacksquare & : \text{sonst} \end{cases}$$

- [1.Fall: $b \in dom(adap)$]

  Es gilt $\mathfrak{b} \simeq \mathfrak{b}'$. Per Definition einer Adaption gilt zusammen mit Bemerkung 7.1.2 sowohl $\mathfrak{b}' \in dom(adap)$ als auch $\mathfrak{b} \simeq_{adap} \mathfrak{b}'$.

  Es bleibt zu zeigen, dass $\mathfrak{s} \circ adap(\mathfrak{b}) = \mathfrak{s} \circ adap(\mathfrak{b}')$. Dies folgt unmittelbar aus Proposition 7.2.17. ✓

- [2.Fall: $b \notin dom(a)$]

  Es gilt $\mathfrak{b} \simeq \mathfrak{b}'$. Laut Bemerkung 7.1.2 gilt dann auch $b' \notin dom(adap)$.

Es folgt $\mathfrak{F}^{\ll adap,\mathfrak{s}\gg}((\mathcal{S},\ p,\ [\mathfrak{b}])) = \blacksquare$ genau wie $\mathfrak{F}^{\ll adap,\mathfrak{s}\gg}((\mathcal{S},\ p,\ [\mathfrak{b}'])) = \blacksquare$ und somit insbesondere $(\mathcal{S},\ p,\ [\mathfrak{b}]) =_{\mathfrak{F}^{\ll adap,\mathfrak{s}\gg}} (\mathcal{S},\ p,\ [\mathfrak{b}'])$. ✓

□

$\mathfrak{F}$-Instanzen stellen Beobachter dar, welche ihre Beobachtungen innerhalb eines Ausführungs-Kontextes, unabhängig von System und Programm, lediglich anhand der Programmausführung treffen. Da Adaptionen als partielle Funktionen definiert sind, existieren womöglich Programmausführungen außerhalb deren Definitionsbereichs. Wir wollen uns dies so vorstellen, dass $\mathfrak{F}$-Instanzen durch Angabe einer Adaption evtl. zu Beobachtern spezieller System-Typen werden. Dies ist beispielsweise bei Beobachtern probabilistischer System der Fall. Da diese Wahrscheinlichkeiten als Übergangs-Annotationen voraussetzen, wären diese ungeeignet für die Beobachtung alternativer Systeme. In solchen Fällen wollen wir davon ausgehen, dass keine informativen Beobachtungen zustande kommen und drücken dies durch die Beobachtung "■" aus, was eine Art Augenklappen vor den Augen des entsprechende Beobachters widerspiegeln soll. Man beachte hierbei insbesondere, dass die Beobachtungen $\varnothing$ und ■ sich unterscheiden.

Proposition 7.3.4:

Sei $pTrace$ der Beobachter aus Beispiel 4.2.3 sowie $\mathfrak{F}^{\ll pTr\gg}$ der durch die μ-Semantik pTr induzierte Beobachter. Es gilt

$$pTrace \equiv \mathfrak{F}^{\ll pTr\gg}$$

Beweis.
Wir zeigen hierfür $pTrace \preceq \mathfrak{F}^{\ll pTr\gg}$ und $\mathfrak{F}^{\ll pTr\gg} \preceq pTrace$.

- $[pTrace \preceq \mathfrak{F}^{\ll pTr\gg}]$

  Laut den Proposition 6.1.18 und 5.1.5 reicht es für beliebige Ausführungs-Kontexte $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},p,[\mathcal{T}_{<\mathfrak{a}>}\langle s_0\rangle]),\ (\mathcal{T}'_{<\mathfrak{a}',\mathfrak{p}'>},p',[\mathcal{T}'_{<\mathfrak{a}'>}\langle s'_0\rangle]) \in \mathfrak{AR}$ mit

  $$(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},p,[\mathcal{T}_{<\mathfrak{a}>}\langle s_0\rangle]) =_{\mathfrak{F}^{\ll pTr\gg}} (\mathcal{T}'_{<\mathfrak{a}',\mathfrak{p}'>},p',[\mathcal{T}'_{<\mathfrak{a}'>}\langle s'_0\rangle])$$

  zu zeigen, dass auch

  $$(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},p,[\mathcal{T}_{<\mathfrak{a}>}\langle s_0\rangle]) =_{pTrace} (\mathcal{T}'_{<\mathfrak{a}',\mathfrak{p}'>},p',[\mathcal{T}'_{<\mathfrak{a}'>}\langle s'_0\rangle])$$

  Wir zeigen hier lediglich die Inklusion " $\subseteq$ ". Die Inklusion " $\supseteq$ " folgt analog.

  Sei also $(a_0,b_0,\ldots,b_{n-2},a_{n-1}) \in pTrace((\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},p,[\mathcal{T}_{<\mathfrak{a}>}\langle s_0\rangle]))$. Es existiert ein $\pi \in (\mathcal{T}^{\downarrow s_0}[(\epsilon,s_0)])^{(n)}$ mit $\mathfrak{a}_S(\pi_i) = a_i$ sowie $\mathfrak{a}_T(\pi_i \to \pi_{i+1}) = b_i$. Offensichtlich gilt dann

auch $\mathcal{T}^{\downarrow s_0}_{<\mathfrak{a}\downarrow>}[\pi_i] \vDash a_i \overset{b_i}{\to} \ldots \overset{b_{n-2}}{\to} a_{n-1}$ und somit insbesondere

$$\mathcal{T}_{<\mathfrak{a}>}\langle s_0\rangle \vDash a_0 \overset{b_0}{\to} \ldots \overset{b_{n-2}}{\to} a_{n-1}$$

Laut Voraussetzung gilt nun auch

$$\mathcal{T}'_{<\mathfrak{a}'>}[s'_0] \vDash a_0 \overset{b_0}{\to} \ldots \overset{b_{n-2}}{\to} a_{n-1}$$

Es folgt die Existenz eines Pfads $\pi' \in (\mathcal{T}'^{\downarrow s'_0}[(\epsilon, s'_0)])^{(n)}$ mit $\mathcal{T}'^{\downarrow s'_0}_{<\mathfrak{a}'\downarrow>}[\pi'_i] \vDash a_i \overset{b_i}{\to} \ldots \overset{b_{n-2}}{\to} a_{n-1}$.

Offensichtlich gilt dann aber auch $\mathfrak{a}'_S(\pi'_i) = a_i$ sowie $\mathfrak{a}'_T(\pi'_i \to \pi'_{i+1}) = b_i$ und somit insbesondere $(a_0,b_0,\ldots,b_{n-2},a_{n-1}) \in pTrace((\mathcal{T}'_{<\mathfrak{a}',\mathfrak{p}'>},p',[\mathcal{T}'_{<\mathfrak{a}'>}\langle s'_0\rangle]))$. ✓

- $[\mathfrak{F}^{\ll pTr\gg} \preceq pTrace]$
  Laut Proposition 6.1.18 und 5.1.5 reicht es für beliebige Ausführungs-Kontexte $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},p,[\mathcal{T}_{<\mathfrak{a}>}\langle s_0\rangle])$ und $(\mathcal{T}'_{<\mathfrak{a}',\mathfrak{p}'>},p',[\mathcal{T}'_{<\mathfrak{a}'>}\langle s'_0\rangle])$ mit

  $$(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},p,[\mathcal{T}_{<\mathfrak{a}>}\langle s_0\rangle]) =_{pTrace} (\mathcal{T}'_{<\mathfrak{a}',\mathfrak{p}'>},p',[\mathcal{T}'_{<\mathfrak{a}'>}\langle s'_0\rangle])$$

  zu zeigen, dass auch

  $$(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},p,[\mathcal{T}_{<\mathfrak{a}>}\langle s_0\rangle]) =_{\mathfrak{F}^{\ll pTr\gg}} (\mathcal{T}'_{<\mathfrak{a}',\mathfrak{p}'>},p',[\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]])$$

  Wir zeigen hier lediglich die Inklusion " $\subseteq$ ". Die Inklusion " $\supseteq$ " folgt analog.

  Gelte also $\mathcal{T}_{<\mathfrak{a}>}\langle s_0\rangle \vDash a_0 \overset{b_0}{\to} \ldots \overset{b_{n-2}}{\to} a_{n-1}$. Es folgt die Existenz eines Pfads $\pi \in (\mathcal{T}^{\downarrow s_0}[(\epsilon,s_0)])^{(n)}$ mit $\mathcal{T}^{\downarrow s_0}_{<\mathfrak{a}\downarrow>}[\pi_i] \vDash a_i \overset{b_i}{\to} \ldots \overset{b_{n-2}}{\to} a_{n-1}$. Offensichtlich gilt dann auch $\mathfrak{a}_S(\pi_i) = a_i$ sowie $\mathfrak{a}_T(\pi_i \to \pi_{i+1}) = b_i$ und somit insbesondere

  $$(a_0,b_0,\ldots,b_{n-2},a_{n-1}) \in pTrace((\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},p,[\mathcal{T}_{<\mathfrak{a}>}\langle s_0\rangle]))$$

  Laut Voraussetzung gilt nun auch

  $$(a_0,b_0,\ldots,b_{n-2},a_{n-1}) \in pTrace((\mathcal{T}'_{<\mathfrak{a}',\mathfrak{p}'>},p',[\mathcal{T}'_{<\mathfrak{a}'>}\langle s'_0\rangle]))$$

  Es folgt die Existenz eines $\pi' \in (\mathcal{T}'[s'_0])^{(n)}$ mit $\mathfrak{a}'_S(\pi_i) = a_i$ sowie $\mathfrak{a}'_T(\pi'_i \to \pi'_{i+1}) = b_i$. Offensichtlich gilt dann aber auch $\mathcal{T}'^{\downarrow s'_0}_{<\mathfrak{a}'\downarrow>}[\pi'_i] \vDash a_i \overset{b_i}{\to} \ldots \overset{b_{n-2}}{\to} a_{n-1}$ und somit insbesondere $\mathcal{T}'_{<\mathfrak{a}'>}\langle s'_0\rangle \vDash a_0 \overset{b_0}{\to} \ldots \overset{b_{n-2}}{\to} a_{n-1}$. ✓

□

Beispiel 7.3.5:

$\mathfrak{F}^{\ll pTr \gg}$ ist ein *r.a.i.* Beobachter.

Beweis.
Hierfür müssen wir zeigen, dass $\mathfrak{F}^{\ll pTr \gg}$ sowohl *z.r.a.i.* als auch ü.*r.a.i.*. Wir beschränken uns auf den Beweis der *z.r.a.i.*-Eingenschaft. Die ü.*r.a.i.*-Eigenschaft folgt analog.
Was wir also zeigen müssen ist

$$\forall M \in \mathfrak{U}_{<\mathfrak{F}^{\ll pTr \gg}>} : \{(\mathcal{T}_{<(f \circ \mathfrak{a}_S, \mathfrak{a}_T), \mathfrak{p}>}, p, [\mathcal{T}_{<(f \circ \mathfrak{a}_S, \mathfrak{a}_T)>}\langle s \rangle]) \mid (\mathcal{T}_{<\mathfrak{a}, \mathfrak{p}>}, p, [\mathcal{T}_{<\mathfrak{a}>}\langle s \rangle]) \in M\} \in \mathfrak{U}_{<\mathfrak{F}^{\ll pTr \gg}>}$$

für beliebiges f mit passenden Definitionsbereich. Wir wissen bereits laut 6.2.11, dass $pTrace$ *z.r.a.i.* ist, d.h. es gilt

$$\forall M \in \mathfrak{U}_{<pTrace>} : \{(\mathcal{T}_{<(f \circ \mathfrak{a}_S, \mathfrak{a}_T), \mathfrak{p}>}, p, [\mathcal{T}_{<(f \circ \mathfrak{a}_S, \mathfrak{a}_T)>}\langle s \rangle]) \mid (\mathcal{T}_{<\mathfrak{a}, \mathfrak{p}>}, p, [\mathcal{T}_{<\mathfrak{a}>}\langle s \rangle]) \in M\} \in \mathfrak{U}_{<pTrace>}$$

Außerdem wissen wir laut Proposition 7.3.4, dass $pTrace \equiv \mathfrak{F}^{\ll pTr \gg}$. Mit Proposition 6.1.19 folgt dann $\mathfrak{U}_{<pTrace>} = \mathfrak{U}_{<\mathfrak{F}^{\ll pTr \gg}>}$ und folglich auch die Behauptung.

□

Beispiel 7.3.6:

Der Beobachter $weak_{bisim} := \mathfrak{F}^{\ll silent, bisim \gg}$ stellt einen Beobachter dar, welche stille Übergänge nicht wahrnimmt

Beispiel 7.3.7:

Der Beobachter $prb_{bisim} := \mathfrak{F}^{\ll prob, bisim \gg}$ stellt einen Beobachter dar, welche probabilistische Systeme spezialisiert ist.

7.4 Hierarchien

In Anlehnung an Abschnitt 6.1 werden wir eine Ordnungsstruktur auf μ-Semantiken definieren. Diese wird sich auf natürliche Weise auf die zugehörigen Framework-Instanzen übertragen. Aussagen bzgl. Noninterferenz in Bezug auf Beobachtern sind somit direkt übertragbar. Die Betrachtung von Ordnungsstrukturen auf μ-Semantiken wird dadurch motiviert, dass der Nachweis dieser in der Regel simpler bzw. anschaulicher ausfällt als selbiges für entsprechenden Framework-Instanzen. Darüber hinaus bietet die Literatur Hierarchisierungen für diverse bedeutende Semantiken der theoretischen Informatik. Diese

werden anschließend mühelos auf zugehörige Beobachter übertragbar sein, dazu später mehr.
Die folgenden Definition und Propositionen orientieren sich an Abschnitt 6.1. Diese sind ähnlich motiviert und bedürfen somit keiner näheren Erläuterung.

Definition 7.4.1: [Ordnung μ-Semantiken]

Wir definieren folgende folgende relationale Ordnungsklassen auf der Klasse der μ-Semantiken

- $\mathfrak{s} \preceq \mathfrak{s}' \quad :\Leftrightarrow \quad \exists f : \ \mathfrak{s} = f \circ \mathfrak{s}'$
- $\mathfrak{s} \preceq_{\mathcal{K}} \mathfrak{s}' \ :\Leftrightarrow \ \exists f : \ \mathfrak{s}|_{\mathcal{K}} = f \circ \mathfrak{s}'|_{\mathcal{K}}$ für $\mathcal{K} \subseteq TS[*]\text{-}A$

Wir bezeichnen f in beiden Fällen als *Reduktion*.

Proposition 7.4.2:

Folgende Aussagen sind äquivalent

(i) $\mathfrak{s} \preceq \mathfrak{s}'$

(ii) $\mathfrak{s} \preceq_{TS[*]\text{-}A} \mathfrak{s}'$

(iii) $\forall \mathcal{K} \subseteq TS[*]\text{-}A : \ \mathfrak{s} \preceq_{\mathcal{K}} \mathfrak{s}'$

Beweis.
Der Beweis verläuft vollkommen analog zu 6.1.3.

□

Proposition 7.4.3:

(i) $\preceq$ ist Präordnung

(ii) $\preceq_{\mathcal{K}}$ ist Präordnung, für jedes $\mathcal{K} \subseteq TS[*]\text{-}A$

Beweis.
Der Beweis verläuft vollkommen analog zu 6.1.4.

□

Definition 7.4.4: [W]

ir definieren folgende Relationen.

- $\equiv$ ist die zu $\preceq$ kanonische Äquivalenzrelation
- $\equiv_{\mathcal{K}}$ ist die zu $\preceq_{\mathcal{K}}$ kanonische Äquivalenzrelation für beliebiges $\mathcal{K} \subseteq TS[*]\text{-}A$

Proposition 7.4.5:

Für beliebige μ-Semantiken $\mathfrak{s}$ und $\mathfrak{s}'$ sind folgende Aussagen äquivalent

(i) $\mathfrak{s} \equiv \mathfrak{s}'$

(ii) $\mathfrak{s} \equiv_{TS(*)\text{-}A} \mathfrak{s}'$

(iii) $\forall \mathcal{K} \in TS(*)\text{-}A : \ \mathfrak{s} \equiv_{\mathcal{K}} \mathfrak{s}'$

Beweis.
Der Beweis verläuft vollkommen analog zu 6.1.11.

□

Proposition 7.4.6:

Seien $\mathfrak{s}$ und $\mathfrak{s}'$ beliebige μ-Semantiken mit $\mathfrak{s} \equiv_{\mathcal{K}} \mathfrak{s}'$. Seien $\mathfrak{s}_1$ und $\mathfrak{s}_2$ weitere μ-Semantiken mit $\mathfrak{s}_1 \preceq_{\mathcal{K}} \mathfrak{s} \preceq_{\mathcal{K}} \mathfrak{s}_2$, dann gilt auch

$$\mathfrak{s}_1 \preceq_{\mathcal{K}} \mathfrak{s}' \preceq_{\mathcal{K}} \mathfrak{s}_2$$

Beweis.
Der Beweis verläuft vollkommen analog zu 6.1.12. □

Proposition 7.4.7:

Seien $\mathfrak{s}$ und $\mathfrak{s}'$ beliebige μ-Semantiken mit $\mathfrak{s} \equiv \mathfrak{s}'$. Seien $\mathfrak{s}_1$ und $\mathfrak{s}_2$ weitere μ-Semantiken mit $\mathfrak{s}_1 \preceq \mathfrak{s} \preceq \mathfrak{s}_2$, dann gilt auch

$$\mathfrak{s}_1 \preceq \mathfrak{s}' \preceq \mathfrak{s}_2$$

Beweis.
Der Beweis verläuft vollkommen analog zu 6.1.13.

□

Wir wollen jetzt für μ-Semantiken analoge Zusammenhänge wie in Proposition 6.1.18 und Proposition 6.1.19 nachweisen. Beachte man Proposition 5.1.5, dann kann für eine μ-Semantik $\mathfrak{s}$ die Relation $(=_{\mathfrak{s}})$ als eine Form von “Unterscheidbarkeit“ aufgefasst werden.

Proposition 7.4.8:

Es gilt

(i) $(=_{\mathfrak{s}'}) \subseteq (=_{\mathfrak{s}})$ gdw. $\mathfrak{s} \preceq \mathfrak{s}'$

(ii) $(=_{\mathfrak{s}'|_{\mathcal{K}}}) \subseteq (=_{\mathfrak{s}|_{\mathcal{K}}})$ gdw. $\mathfrak{s} \preceq_{\mathcal{K}} \mathfrak{s}'$, für jedes $\mathcal{K} \subseteq \mathfrak{AR}$

Beweis.
Wir zeigen nur (ii), anschließend ergibt sich (i) für $\mathcal{K} = TS[*]\text{-}A$ wegen $\mathfrak{s} = \mathfrak{s}|_{TS[*]\text{-}A}$ sowie Proposition 7.4.5.
Wir zeigen die Implikationen " $\Rightarrow$ " und " $\Leftarrow$ " getrennt.

- $[\Rightarrow]$

  Wir zeigen $f : im(\mathfrak{s}'|_{\mathcal{K}}) \to im(\mathfrak{s}|_{\mathcal{K}})$ mit $f(\mathfrak{s}'|_{\mathcal{K}}(\mathcal{G})) \mapsto \mathfrak{s}|_{\mathcal{K}}(\mathcal{G})$ ist eine wohldefinierte funktionale Klasse. Anschließend ergibt sich direkt $\mathfrak{s}|_{\mathcal{K}} = f \circ \mathfrak{s}'|_{\mathcal{K}}$.

  Die Wohldefiniertheit von f ergibt sich dabei wie folgt.

  Seien $\mathfrak{s}'|_{\mathcal{K}}(\mathcal{G}), \mathfrak{s}'|_{\mathcal{K}}(\mathcal{G}') \in im(\mathfrak{s}'|_{\mathcal{K}})$ mit $\mathcal{G} =_{\mathfrak{s}'|_{\mathcal{K}}} \mathcal{G}'$. Es folgt direkt aus der Voraussetzung, dass dann auch $\mathcal{G} =_{\mathfrak{s}|_{\mathcal{K}}} \mathcal{G}'$. ✓

- $[\Leftarrow]$

  Gelte $\mathcal{G} =_{\mathfrak{s}'_{\mathcal{K}}} \mathcal{G}'$. Laut Voraussetzung existiert eine Reduktion f mit $\mathfrak{s}_{\mathcal{K}} = f \circ \mathfrak{s}'_{\mathcal{K}}$. Es ergibt sich direkt $\mathcal{G} =_{f \circ \mathfrak{s}'_{\mathcal{K}}} \mathcal{G}'$ was gleichbedeutend ist mit $\mathcal{G} =_{\mathfrak{s}_{\mathcal{K}}} \mathcal{G}'$. ✓

□

Proposition 7.4.9:

Es gilt

(i) $(=_{\mathfrak{s}}) = (=_{\mathfrak{s}'})$ gdw. $\mathfrak{s} \equiv \mathfrak{s}'$

(ii) $(=_{\mathfrak{s}|_{\mathcal{K}}}) = (=_{\mathfrak{s}'|_{\mathcal{K}}})$ gdw. $\mathfrak{s} \equiv_{\mathcal{K}} \mathfrak{s}'$, für jedes $\mathcal{K} \subseteq \mathfrak{AR}$

Beweis.
Wir zeigen nur (ii), anschließend ergibt sich (ii) analog.

Sei $\mathcal{K} \subseteq TS[*]\text{-}A$ beliebig. Es gilt $(=_{\mathfrak{s}|_{\mathcal{K}}}) = (=_{\mathfrak{s}'|_{\mathcal{K}}})$ gdw. $(=_{\mathfrak{s}|_{\mathcal{K}}}) \subseteq (=_{\mathfrak{s}'|_{\mathcal{K}}})$ sowie $(=_{\mathfrak{s}'|_{\mathcal{K}}}) \subseteq (=_{\mathfrak{s}|_{\mathcal{K}}})$. Laut Proposition 7.4.8 gilt dies wiederum gdw. $\mathfrak{s}' \preceq_{\mathcal{K}} \mathfrak{s}$ als auch $\mathfrak{s} \preceq_{\mathcal{K}} \mathfrak{s}'$. Dies ist aber per Definition gleichbedeutend mit $\mathfrak{s} \equiv_{\mathcal{K}} \mathfrak{s}'$. ✓

□

Wir wollen nun beginnen die Ordnungsstrukturen der μ-Semantiken auf die der Beobachter zu übertragen.

Proposition 7.4.10:

Seien $\mathfrak{F}^{\ll adap,\mathfrak{s}\gg}$ und $\mathfrak{F}^{\ll adap,\mathfrak{s}'\gg}$ zwei $\mathfrak{F}$-Instanzen mit

(i) $s \preceq_{\mathcal{K}} s'$

(ii) $\mathcal{K}' := \{(\mathcal{S}, p, [\mathfrak{b}]) \in \mathfrak{AR} \mid adap(\mathfrak{b}) \in \mathcal{K} \ \vee \ \mathfrak{b} \notin dom(adap)\}$
(wohldefiniert laut Definition 7.1.1 und Bemerkung 7.1.2)

dann gilt

$$\mathfrak{F}^{\ll adap,\mathfrak{s}\gg} \preceq_{\mathcal{K}'} \mathfrak{F}^{\ll adap,\mathfrak{s}'\gg}$$

Beweis.
Wir müssen die Existenz einer Reduktion f nachweisen mit $\mathfrak{F}^{\ll adap,\mathfrak{s}\gg}|_{\mathcal{K}'} = f \circ \mathfrak{F}^{\ll adap,\mathfrak{s}'\gg}|_{\mathcal{K}'}$.

Laut Voraussetzung existiert eine Reduktion g mit $\mathfrak{s}|_{\mathcal{K}} = g \circ \mathfrak{s}'|_{\mathcal{K}}$.
Sei nun $f : im(\mathfrak{F}^{\ll adap,\mathfrak{s}'\gg}|_{\mathcal{K}'}) \to im(\mathfrak{F}^{\ll adap,\mathfrak{s}\gg}|_{\mathcal{K}'})$ mit

$$f(L) = \begin{cases} g(L) & : L \neq \blacksquare \\ \blacksquare & : L = \blacksquare \end{cases}$$

Es gilt

$$f \circ \mathfrak{F}^{\ll adap,\mathfrak{s}'\gg}|_{\mathcal{K}'}((\mathcal{S},\ p,\ [\mathfrak{b}])) = \begin{cases} (f \circ \mathfrak{s}'|_{\mathcal{K}} \circ adap)(\mathfrak{b}) & : \ \mathfrak{b} \in dom(adap) \\ f(\blacksquare) & : \text{ sonst} \end{cases}$$

Nun gilt $f(\blacksquare) = \blacksquare$. Außerdem gilt, da $\blacksquare \notin im(\mathfrak{s}'|_{\mathcal{K}})$, dass $f \circ \mathfrak{s}'|_{\mathcal{K}} = g \circ \mathfrak{s}'|_{\mathcal{K}}$ und somit aufgrund von $g \circ \mathfrak{s}'|_{\mathcal{K}} = \mathfrak{s}|_{\mathcal{K}}$ insbesondere auch $f \circ \mathfrak{s}'|_{\mathcal{K}} = \mathfrak{s}|_{\mathcal{K}}$. Es ergibt sich $\mathfrak{F}^{\ll adap,\mathfrak{s}\gg}|_{\mathcal{K}'} = f \circ \mathfrak{F}^{\ll adap,\mathfrak{s}'\gg}|_{\mathcal{K}'}$.

□

Bemerkung 7.4.11:

Man beachte, dass wir im Beweis von 7.4.10 explizit die Unterscheidung der Beobachtungen "$\varnothing$" und "$\blacksquare$" ausgenutzt haben.

Proposition 7.4.12:

Seien $\mathfrak{F}^{\ll adap,\mathfrak{s}\gg}$ und $\mathfrak{F}^{\ll adap,\mathfrak{s}'\gg}$ zwei $\mathfrak{F}$-Instanzen mit $s \preceq s'$, dann gilt

$$\mathfrak{F}^{\ll adap,\mathfrak{s}\gg} \preceq \mathfrak{F}^{\ll adap,\mathfrak{s}'\gg}$$

Beweis.
Sei $\mathcal{K} := TS[*]\text{-}A$. Laut Proposition 7.4.2 gilt $\mathfrak{s} \preceq \mathfrak{s}'$ gdw. $\mathfrak{s} \preceq_{\mathcal{K}} \mathfrak{s}'$. Es folgt mit Proposition

7.4.10, dass $\mathfrak{F}^{\ll adap,\mathfrak{s}\gg} \preceq_{\mathcal{K}'} \mathfrak{F}^{\ll adap,\mathfrak{s}'\gg}$ für $\mathcal{K}' := \{(\mathcal{S}, p, [\mathfrak{b}]) \in \mathfrak{AR} \mid adap(\mathfrak{b}) \in \mathcal{K} \ \vee \ b \notin dom(adap)\}$. Offensichtlich gilt aber $\mathcal{K}' = \mathfrak{AR}$ und somit laut Proposition 6.1.3 auch $\mathfrak{F}^{\ll adap,\mathfrak{s}\gg} \preceq \mathfrak{F}^{\ll adap,\mathfrak{s}'\gg}$.

□

Proposition 7.4.13:

Seien $\mathfrak{F}^{\ll adap,\mathfrak{s}\gg}$ und $\mathfrak{F}^{\ll adap,\mathfrak{s}'\gg}$ zwei $\mathfrak{F}$-Instanzen mit

(i) $\mathfrak{s} \equiv_{\mathcal{K}} \mathfrak{s}'$

(ii) $\mathcal{K}' := \{(\mathcal{S}, p, [\mathfrak{b}]) \in \mathfrak{AR} \mid adap(\mathfrak{b}) \in \mathcal{K} \ \vee \ \mathfrak{b} \notin dom(adap)\}$
(wohldefiniert laut Definition 7.1.1 und Bemerkung 7.1.2)

dann gilt

$$\mathfrak{F}^{\ll adap,\mathfrak{s}\gg} \equiv_{\mathcal{K}'} \mathfrak{F}^{\ll adap,\mathfrak{s}'\gg}$$

Beweis.
Per Definition gilt $\mathfrak{s} \equiv_{\mathcal{K}} \mathfrak{s}'$ gdw. $\mathfrak{s} \preceq_{\mathcal{K}} \mathfrak{s}'$ sowie $\mathfrak{s}' \preceq_{\mathcal{K}} \mathfrak{s}$. Aufgrund von Proposition 7.4.10 ergibt sich $\mathfrak{F}^{\ll adap,\mathfrak{s}\gg} \preceq_{\mathcal{K}'} \mathfrak{F}^{\ll adap,\mathfrak{s}'\gg}$ und $\mathfrak{F}^{\ll adap,\mathfrak{s}'\gg} \preceq_{\mathcal{K}'} \mathfrak{F}^{\ll adap,\mathfrak{s}\gg}$. Dies ist wiederum per Definition gleichbedeutend mit $\mathfrak{F}^{\ll adap,\mathfrak{s}\gg} \equiv_{\mathcal{K}'} \mathfrak{F}^{\ll adap,\mathfrak{s}'\gg}$.

□

Proposition 7.4.14:

Seien $\mathfrak{F}^{\ll adap,\mathfrak{s}\gg}$ und $\mathfrak{F}^{\ll adap,\mathfrak{s}'\gg}$ zwei $\mathfrak{F}$-Instanzen mit $\mathfrak{s} \equiv \mathfrak{s}'$, dann gilt

$$\mathfrak{F}^{\ll a,\mathfrak{s}\gg} \equiv \mathfrak{F}^{\ll a,\mathfrak{s}'\gg}$$

Beweis.
Sei $\mathcal{K} := TS[*]\text{-}A$. Laut Proposition 7.4.5 gilt $\mathfrak{s} \equiv \mathfrak{s}'$ gdw. $\mathfrak{s} \equiv_{\mathcal{K}} \mathfrak{s}'$. Es folgt mit Proposition 7.4.13, dass $\mathfrak{F}^{\ll adap,\mathfrak{s}\gg} \equiv_{\mathcal{K}'} \mathfrak{F}^{\ll adap,\mathfrak{s}'\gg}$ für

$$\mathcal{K}' := \{(\mathcal{S}, p, [\mathfrak{b}]) \in \mathfrak{AR} \mid adap(\mathfrak{b}) \in \mathcal{K} \ \vee \ b \notin dom(adap)\}$$

Offensichtlich gilt aber $\mathcal{K}' = \mathfrak{AR}$ und somit laut Proposition 6.1.11 auch $\mathfrak{F}^{\ll adap,\mathfrak{s}\gg} \equiv \mathfrak{F}^{\ll adap,\mathfrak{s}'\gg}$.

□

Nachdem wir einen erfreulichen Zusammenhang zwischen Ordnungen auf Semantiken und Beobachtern hergestellt haben, wollen wir unseren Fokus abschließend auf deterministische Systeme lenken. Diese stellen eine breite Klasse von Systemen dar, welche zugleich im Fokus vieler Theorien stehen. Wir wollen diese aufgrund ihrer besonderen Eigenschaften in Bezug auf Noninterferenz näher betrachten. Genauer wollen wir beweisen, dass geordnete Semantik-Ketten, eingeschlossen zwischen pTr und $bisim$, im Kontexte deterministischer

Systeme gleichstarke Beobachter definieren. Um dies zu motivieren möchten wir auf Abbildung 7.7 aus [G01] verweisen.

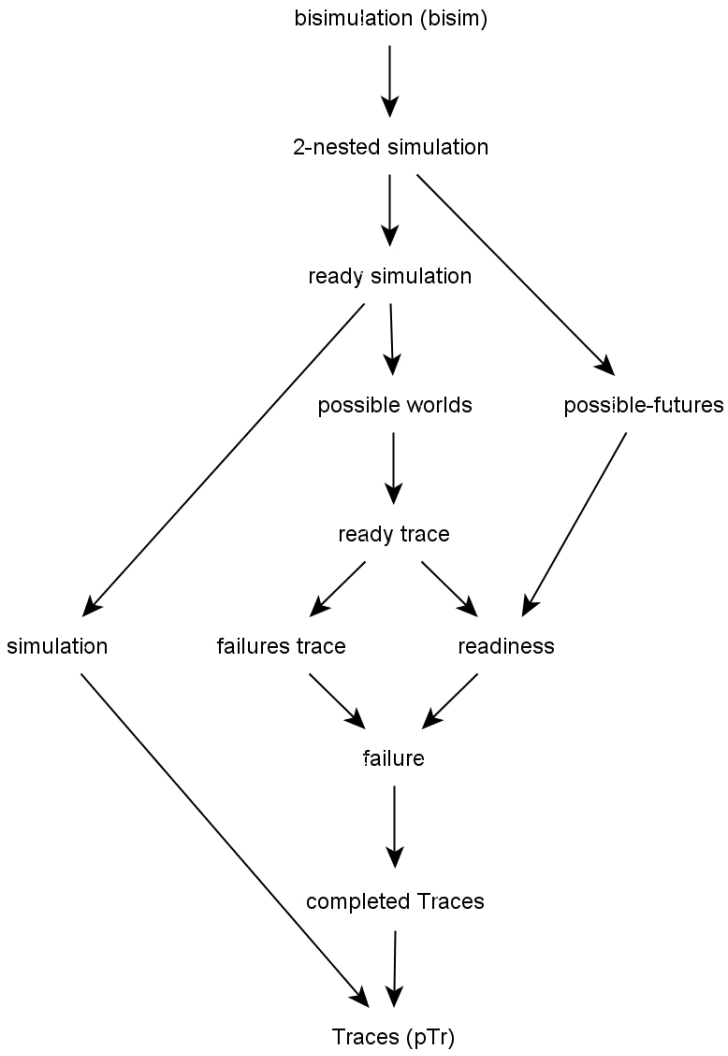


Abbildung 7.7: The linear time - branching time spectrum

In dieser sind zwölf bedeutende Semantiken der theoretischen Informatik dargestellt. Das Diagramm ist dabei als eine Art Hasse-Diagramm zu interpretieren, d.h. eine Semantik an einer Pfeilquelle stellt eine stärkere Semantik dar als diese an der Pfeilspitze. Man erkennt leicht dass die vorgestellten Semantiken sich allesamt zwischen pTr und $bisim$ einordnen lassen. Wir werden zunächst feststellen, dass die $bisim$-Semantik eine stärkere Semantik als pTr beschreibt. Hierfür werden wir zwei alternative Beweisführungen vorstellen. Erstere orientiert sich dabei direkt an der Definition und nutzt die Eigenschaft aus, dass $Ł^{pTr}$ in gewissen Sinne eine Teilklasse von $Ł^{bisim}$ darstellen. Dieser Beweis ist in analoger Ausführung auf alle dargestellten Ordnungsverhältnisse aus Abbildung 7.7 übertragbar.

Alternativ stellen wir einen Beweisführung vor, welche sich eher der Anschauung widmet. Diese orientiert sich an Proposition 7.2.33 und somit an der Tatsache, dass bisimilare Zustandsgraphen dieselben "annotierten Pfadmengen" vorweisen.

Proposition 7.4.15:

Es gilt

$$pTr \preceq bisim$$

Beweis.
Laut Proposition 7.4.8 reicht es für beliebige Zustandsgraphen $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ mit

$$\mathcal{T}_{<\mathfrak{a}>}[s_0] =_{bisim} \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$$

zu zeigen, dass auch

$$\mathcal{T}_{<\mathfrak{a}>}[s_0] =_{pTr} \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$$

Wir zeigen " $\subseteq$ ", anschließend folgt " $\supseteq$ " analog.

Sei also $\varphi \in pTr(\mathcal{T}_{<\mathfrak{a}>}[s_0])$ beliebig. Es gilt $\varphi \in \{\varphi \in Ł^{pTr}_{<\mathfrak{a}>} \mid \mathcal{T}_{<\mathfrak{a}>}[s_0] \vDash \varphi\}$ und da $L^{pTr}_{<\mathfrak{a}>} \subseteq L^{bisim}_{<\mathfrak{a}'>}$ offensichtlich auch $\varphi \in \{\varphi \in Ł^{bisim}_{<\mathfrak{a}>} \mid \mathcal{T}_{<\mathfrak{a}>}[s_0] \vDash \varphi\}$. Laut Voraussetzung gilt nun ebenfalls $\varphi \in \{\varphi \in Ł^{bisim}_{<\mathfrak{a}'>} \mid \mathcal{T}'_{<\mathfrak{a}'>}[s'_0] \vDash \varphi\}$ und da φ von der Form $\varphi = a_0 \overset{b_0}{\to} a_1 \overset{b_1}{\to} \dots$ folglich auch $\varphi \in \{\varphi \in Ł^{pTr}_{<\mathfrak{a}'>} \mid \mathcal{T}'_{<\mathfrak{a}'>}[s'_0] \vDash \varphi\}$ was gleichbedeutend ist mit $\varphi \in pTr(\mathcal{T}'_{<\mathfrak{a}'>}[s'_0])$. □

Beweis (alternativ).
Laut Proposition 7.4.8 reicht es für beliebige Zustandsgraphen $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ mit

$$\mathcal{T}_{<\mathfrak{a}>}[s_0] =_{bisim} \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$$

zu zeigen, dass auch

$$\mathcal{T}_{<\mathfrak{a}>}[s_0] =_{pTr} \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$$

Laut Proposition 7.2.31 und 7.2.23 könnten wir hierfür zeigen, dass wenn $\mathcal{T}_{<\mathfrak{a}>}[s_0] \approx_{bisim} \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ auch $\mathcal{T}_{<\mathfrak{a}>}[s_0] \approx_{pTr} \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$. Dies folgt unmittelbar anhand von Proposition 7.2.33.

□

Sei im Folgenden

$$det := \{\mathcal{T}_{<\mathfrak{a}>}[s_0] \in TS[*]\text{-}A \mid \mathcal{T} \text{ ist deterministisch}\}$$

die Klasse der deterministischen annotierten Zustandsgraphen sowie

$$Det := \{(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>}, p, [\mathfrak{b}]) \mid \mathcal{T} \text{ ist deterministsch}\}$$

die Klasse der Ausführungs-Kontexte deterministischer Systeme.

Lemma 7.4.16:

Sei $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},p,[\mathfrak{b}]) \in Det$, dann gilt

$$\mathfrak{b} \in det$$

Beweis.
Sei $\mathfrak{b} = \mathcal{T}_{<\mathfrak{a}>}\langle s\rangle$ für passendes $s \in S^{(\mathcal{T})}$.
Wir zeigen zunächst die Repräsentantenunabhängigkeit der obigen Aussage. Sei also $\mathcal{T}_{<\mathfrak{a}>}\langle s'\rangle \in [\mathcal{T}_{<\mathfrak{a}>}\langle s\rangle]$ beliebig. Da $\mathcal{T}_{<\mathfrak{a}>}\langle s\rangle \in det$ folgt dass $\mathcal{T}^{\downarrow s}$ deterministisch ist. Aufgrund von $\mathcal{T}_{<\mathfrak{a}>}\langle s\rangle \simeq \mathcal{T}_{<\mathfrak{a}>}\langle s'\rangle$ folgt aber auch, dass $\mathcal{T}^{\downarrow s'}$ deterministisch und folglich auch $\mathcal{T}_{<\mathfrak{a}>}\langle s'\rangle \in det$
Für den Beweis der eigentlichen Aussagen nehmen wir an, dass $\mathcal{T}_{<\mathfrak{a}>}\langle s\rangle$ nicht deterministisch ist. Dann existiert ein $(V,s) \in S^{(\mathcal{T}^{\downarrow s})}$ sowie $(V_1,s_1),(V_2,s_2) \in S^{(\mathcal{T}^{\downarrow s})}$ für die $(V_1,s_1) \neq (V_2,s_2)$ mit $(V,s) \to (V_1,s_1) \in T^{(\mathcal{T}^{\downarrow s})}$ und $(V,s) \to (V_2,s_2) \in T^{(\mathcal{T}^{\downarrow s})}$. Folglich gilt auch $s \to s_1 \in T^{(\mathcal{T})}$ genau wie $s \to s_2 \in T^{(\mathcal{T})}$. Dies steht jedoch im direkten Widerspruch zu $(\mathcal{T}_{<\mathfrak{a},\mathfrak{p}>},p,[\mathfrak{b}]) \in Det$. $\lightning$ □

Proposition 7.4.17:

Für μ-Semantiken $\mathfrak{s}_1,\ldots,\mathfrak{s}_n$ mit

$$pTr \preceq_{(det)} \mathfrak{s}_1 \preceq_{(det)} \ldots \preceq_{(det)} \mathfrak{s}_n \preceq_{(det)} bisim$$

gilt

$$pTr \equiv_{(det)} \mathfrak{s}_1 \equiv_{(det)} \ldots \equiv_{(det)} \mathfrak{s}_n \equiv_{(det)} bisim$$

Beweis.
Hierfür reicht es zu zeigen, dass $bisim \preceq_{det} pTr$. Anschließend folgt mit der in Proposition 7.4.3 gezeigten Transitivität von $\preceq_{det}$ die Behauptung.

Hierfür reicht es laut Proposition 7.4.8 für zwei beliebige Zustandsgraphen $\mathcal{T}_{<\mathfrak{a}>}[s_0], \mathcal{T}'_{<\mathfrak{a}'>}[s'_0] \in det$ mit $\mathcal{T}_{<\mathfrak{a}>}[s_0] =_{pTr} \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ zu zeigen, dass $\mathcal{T}_{<\mathfrak{a}>}[s_0] =_{bisim} \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$. Wir greifen hierfür auf die Proposition 7.2.23 und 7.2.31 zurück und zeigen, dass wenn $\mathcal{T}_{<\mathfrak{a}>}[s_0] \approx_{pTr} \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ auch $\mathcal{T}_{<\mathfrak{a}>}[s_0] \approx_{bisim} \mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$. Dies machen wir indem wir zeigen, dass $R \subseteq S^{(\mathcal{T})} \times S^{(\mathcal{T}')}$ mit $(s,s') \in R$ gdw. wenn Pfade $\pi \in (\mathcal{T}[s_0])^{(*)}$ und $\pi' \in (\mathcal{T}'[s'_0])^{(*)}$ existieren mit

(i) $|\pi| = |\pi'|$

(ii) $\pi_{|\pi|-1} = s$

(iii) $\pi'_{|\pi'|-1} = s'$

eine Bisimulation zwischen $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ darstellt. Vorher beweisen wir folgendes Lemma.

Lemma 7.4.18:

Sei $\mathcal{T}$ ein deterministisches Transitionssystem sowie $\mathcal{T}[s_0]$ ein entsprechender Zustandsgraph, dann ist jeder $\mathcal{T}[s_0]$-Pfad eindeutig durch seine Länge bestimmt.

Beweis.
Wir zeigen die Behauptung für endliche Pfade. Für unendliche Pfade verläuft der Beweis analog.
Seien $\pi,\pi' \in (\mathcal{T}[s_0])^{(n)}$ beliebige Pfade der Länge $n \in \mathbb{N}$. Wir zeigen $\pi_i = \pi'_i$ per Induktion über $i \in \{0,\ldots,n\}$.

- [Induktionsanfang]

  Per Definition gilt sowohl $\pi_0 = s_0$ als auch $\pi'_0 = s_0$ und somit insbesondere $\pi_0 = \pi'_0$. ✓

- [Induktionsvoraussetzung]

  Die Behauptung gelte für festes beliebiges $i \in \{0,\ldots,n-1\}$.

- [Induktionsschritt]

  Laut Induktionsvoraussetzung gilt $\pi_i = \pi'_i$ für $i \in \{0,\ldots,n-1\}$. Da $\mathcal{T}$ deterministisch ist existiert ein eindeutig bestimmtes $s \in S^{(\mathcal{T})}$ mit $(\pi_i,s) \in T^{(\mathcal{T})}$ und folglich gilt $\pi_{i+1} = s$. Analog ergibt sich die Existenz eines eindeutig bestimmten $s' \in S^{(\mathcal{T})}$ mit $(\pi'_i,s') \in T^{(\mathcal{T})}$ und folglich gilt $\pi'_{i+1} = s'$. Da $\pi_i = \pi'_i$ gilt aber auch $s = s'$ und folglich auch $\pi_{i+1} = \pi'_{i+1}$. ✓

□

Wir müssen nun für $(s,s') \in R$ die Gültigkeit folgender Punkte nachweisen.

- $[(s_0,s'_0) \in R]$

  Offensichtlich, betrachte dafür die Pfade (s_0) und (s'_0). ✓

- $[\mathfrak{a}_S(s) = \mathfrak{a}'_S(s')]$

  Laut Voraussetzung existieren Pfade $\pi \in (\mathcal{T}[s_0])^{(*)}$ und $\pi' \in (\mathcal{T}'[s'_0])^{(*)}$ mit $|\pi| = |\pi'|$ sowie $\pi_{|\pi|-1} = s$ und $\pi'_{|\pi'|-1} = s'$. Da $\mathcal{T}$ und $\mathcal{T}'$ deterministisch sind ergibt sich mit Lemma 7.4.18, dass π und π' die einzigen Pfade entsprechender Länge innerhalb ihrer Zustandsgraphen sind. Aus der $pTrace$-Äquivalenz von $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ ergibt sich somit $\mathfrak{a}_S(\pi_{|\pi|-1}) = \mathfrak{a}_S(\pi'_{|\pi'|-1})$ und folglich auch die Behauptung. ✓

- $[s \to t \in T^{(\mathcal{T})} \Rightarrow \exists t' :\ s' \to t' \in T^{(\mathcal{T}')} \ \wedge\ \mathfrak{a}_T(s \to t) = \mathfrak{a}'_T(s' \to t') \ \wedge\ (t,t') \in R]$

  Gelte $s \to t \in T^{(\mathcal{T})}$. Laut Voraussetzung existieren Pfade $\pi \in (\mathcal{T}[s_0])^{(*)}$ und $\pi' \in (\mathcal{T}'[s'_0])^{(*)}$ mit $|\pi| = |\pi'|$ sowie $\pi_{|\pi|-1} = s$ und $\pi'_{|\pi'|-1} = s'$. Betrachte nun den Pfad $\pi.(t)$. Da $\mathcal{T}$ deterministisch ist, ist dieser laut Lemma 7.4.18 einziger $\mathcal{T}[s_0]$-Pfad entsprechender Länge. Aus der $pTrace$-Äquivalenz von $\mathcal{T}_{<\mathfrak{a}>}[s_0]$ und $\mathcal{T}'_{<\mathfrak{a}'>}[s'_0]$ ergibt sich nun auch die Existenz eines $\mathcal{T}'[s'_0]$-Pfads selbiger Länge. Da auch $\mathcal{T}'$ deterministisch ist ergibt sich mit Lemma 7.4.18, dass dieser von der Form $\pi'.(t')$ für ein eindeutig bestimmtes $t' \in S^{(\mathcal{T}')}$. Aus der Eindeutig von $\pi.(t)$ und $\pi'.(t')$ ergibt sich zusammen mit der $pTrace$-Äquivalenz der zugehörigen Zustandsgraphen, dass $\mathfrak{a}_T(\pi_{|\pi|-1} \to t) = \mathfrak{a}'_T(\pi'_{|\pi'|-1} \to t')$ und es ergibt sich aufgrund von $|\pi.(t)| = |\pi'.(t')|$, dass $(t,t') \in R$. ✓

- $[s' \to t' \in T^{(\mathcal{T}')} \ \Rightarrow\ \exists t :\ s \to t \in T^{(\mathcal{T})} \wedge\ \mathfrak{a}'_T(s' \to t') = \mathfrak{a}_T(s \to t) \ \wedge\ (t,t') \in R]$

  Folgt analog. ✓

□

Proposition 7.4.19:

Für μ-Semantiken $\mathfrak{s}_1, \ldots, \mathfrak{s}_n$ mit

$$pTr \preceq_{(det)} \mathfrak{s}_1 \preceq_{(det)} \ldots \preceq_{(det)} \mathfrak{s}_n \preceq_{(det)} bisim$$

gilt

$$\mathfrak{F}^{\ll pTr \gg} \equiv_{(Det)} \mathfrak{F}^{\ll \mathfrak{s}_1 \gg} \equiv_{(Det)} \ldots \equiv_{(Det)} \mathfrak{F}^{\ll \mathfrak{s}_n \gg} \equiv_{(Det)} \mathfrak{F}^{\ll bisim \gg}$$

Beweis.
Laut Proposition 7.4.17 gilt $pTr \equiv_{(det)} \mathfrak{s}_1 \equiv_{(det)} \ldots \equiv_{(det)} \mathfrak{s}_n \equiv_{(det)} bisim$. Es folgt mit Proposition 7.4.13, dass $\mathfrak{F}^{\ll pTr \gg} \equiv_{\mathcal{K}'} \mathfrak{F}^{\ll \mathfrak{s}_1 \gg} \equiv_{\mathcal{K}'} \ldots \equiv_{\mathcal{K}'} \mathfrak{F}^{\ll \mathfrak{s}_n \gg} \equiv_{\mathcal{K}'} \mathfrak{F}^{\ll bisim \gg}$ für $\mathcal{K}' := \{(\mathcal{S}, p, [\mathfrak{b}]) \in \mathfrak{AK} \mid \mathfrak{b} \in det \ \vee\ \mathfrak{b} \notin dom(id|_{BK})\} = \{(\mathcal{S}, p, [\mathfrak{b}]) \in \mathfrak{AK} \mid \mathfrak{b} \in det\}$. Laut Lemma 7.4.16 gilt $Det \subseteq \mathcal{K}'$ und folglich auch die Behauptung.

□

Proposition 7.4.20:

Für μ-Semantiken $\mathfrak{s}_1, \ldots, \mathfrak{s}_n$ mit

$$pTr \preceq \mathfrak{s}_1 \preceq \ldots \preceq \mathfrak{s}_n \preceq bisim$$

gilt

$$\mathfrak{F}^{\ll pTr \gg} \equiv_{(Det)} \mathfrak{F}^{\ll \mathfrak{s}_1 \gg} \equiv_{(Det)} \ldots \equiv_{(Det)} \mathfrak{F}^{\ll \mathfrak{s}_n \gg} \equiv_{(Det)} \mathfrak{F}^{\ll bisim \gg}$$

Beweis.
Für $pTr \preceq \mathfrak{s}_1 \preceq \ldots \preceq \mathfrak{s}_n \preceq bisim$ folgt laut Proposition 7.4.2 insbesondere $pTr \preceq_{(det)} \mathfrak{s}_1 \preceq_{(det)} \ldots \preceq_{(det)} \mathfrak{s}_n \preceq_{(det)} bisim$. Die Behauptung ergibt sich somit direkt mit Proposition 7.4.19.

□

KAPITEL 8

Validierung des Noninterferenz-Begriffs

Ziel dieses Kapitels stellt die abschließende Validierung unseres Konzepts dar. Dies machen wir indem wir für zwei Beispiele von Noninterferenz-Sicherheit zeigen, dass sich durch die von uns erschaffenen Mittel ausdrücken lassen. Hierbei werden wir zunächst eine sequentielle Sprache betrachten, welche in verschiedenen Variationen als Basis vieler wissenschaftlicher Betrachtungen der theoretischen Informatik steht. Anschließend werden wir im zweiten Kapitel eine durch Pseudo-Parallelität verzweigte Sprache betrachten, welche auf den Arbeiten von Sabelfeld und Sands in [SS00] basiert. Hierbei werden wir insbesondere feststellen, dass wir mit unserem Ansatz die Intuition der Autoren bzgl. Noninterferenz-Sicherheit besser umsetzen können, als in [SS00] vorgeschlagen.

8.1 Sequentielle Sprache

In diesem Abschnitt betrachten wir die in der theoretischen Informatik oft verwendete *while*-Sprache. Diese stellt einen minimalen berechnungs-universellen Formalismus dar und erfreut sich großer Beliebtheit als Standard für theoretische Betrachtungen. Wir werden diese im Folgenden syntaktisch und semantisch definieren. Hierbei wollen wir simultan eine Hierarchie von Benutzersichten in die Sprache integrieren. Zu diesem Zweck legen wir einen mehrfach partitionierten Zustandsraum zugrunde, dessen Partition spezifischen Benutzergruppen fest zugeordnet sind. Anschließend werden wir beweisen, dass ein typischer Sicherheitsbegriff des *while*-Systems, basierend auf dem Konzept der Noninterferenz, auf natürliche Weise mit unseren Mitteln formuliert werden kann. Da die *while*-Sprache jedem ambitionierten Informatik vertraut sein sollte werden wir auf detaillierte Beispiele und Erläuterungen verzichten.

Syntax & Semantik

Wir beginnen mit der syntaktischen Definition der Sprache. Da wir in diesem Kontext nur nebensächlich an der Form und Auswertung von Ausdrücken interessiert sind verweisen wir hier der Einfachheit halber auf Ausdrücke der C-basierten Sprachfamilie ohne näher auf diese einzugehen.

Definition 8.1.1: [*while*-Sprache]

Die Syntax eines gültigen *while*-Programms wird durch folgende eindeutige kontextfreie Grammatik, beschrieben (in BNF):

$$W ::= skip \mid Id := A \mid W;W \mid if(B)\ then\ \{W\}\ else\ \{W\} \mid while\ (B)\ do\ \{W\}$$

$$A ::=< \text{arith. Ausdruck} > ,\ B ::=< \text{bool. Ausdruck} >$$

Definition 8.1.2: [Programme]

Mit

$$Prog := \{p \mid W \Rightarrow^* p\}$$

bezeichnen wir die Menge aller gültigen *while*-Programme

Nachdem wir die Syntax der *while*-Sprache definiert haben, definieren wir nun einen zugehörigen Zustandsraum auf welche später die Semantik operieren soll. Den Zustandsraum wollen wir uns dabei fest partitioniert vorstellen. Hierfür sei im Folgenden $c \in \mathbb{N}$ eine beliebige aber feste natürliche Zahl. Unser Zustandsraum wird aus $c+2$ geordneten Partitionen bestehen. Die erste hiervon beinhaltet für allgemeine Benutzergruppen unzugängliche Daten und kann im Folgenden als geheim angesehen werden. Die restlichen Partitionen sind $c+1$ verschiedenen Benutzersichten zugeordnet. Dabei sollen einem Benutzer der Sicht $s \in \{0,\ldots,c\}$ lesender Zugriff auf die Partitionen $0,\ldots,s$ gewährt sein. Wir wollen diesen Zustandsraum nun formal definieren, sowie aufbauend einen zugehörigen Konfigurationsraum. Der Einfachheit halber stellen wir uns die Inhalte der jeweiligen Partitionen als natürliche Zahlen vor. Dies ist im Allgemeinen keine Einschränkung wenn man bedenkt, dass endliche Inhalte stets in natürliche Zahlen codiert werden können. Anschließend definieren wir Benutzersichten als Funktion, welche eine konkrete Ausprägung des Zustandsraum auf die zugehörigen Partitionen abbilden.

Definition 8.1.3: [Zustands & Konfiguration]

Wir bezeichnen mit

- $St := \{(h,(l_i)_{i\in[0,c]_{\mathbb{N}}}) \mid h,l_0,\ldots,l_c \in \mathbb{N}\}$ die Menge der *Programmzustände*
- $Conf := \{\langle\!\langle p,z \rangle\!\rangle \mid p \in Prog, z \in St\}$ als die Menge der *Konfigurationen*

- $TC := \{\langle\!|\langle\rangle,z|\!\rangle \mid z \in St\} \subseteq Conf$ als die Menge der *terminalen Konfigurationen*

Bemerkung 8.1.4:

Wir nutzen "$\langle\!|$" und "$|\!\rangle$" sowie "$\langle$" und "$\rangle$" aus Gründen der Übersicht als alternative Folgenklammern für Konfigurationen bzw. Programmen.

Definition 8.1.5: [Benutzersicht]

Wir definieren für $s \in [0,c]_{\mathbb{N}}$ die Funktion $sL : Conf \to \mathbb{N}^s$ mit

$$\langle\!|p,\ (h,(l_i)_{i\in[0,c]_{\mathbb{N}}})|\!\rangle \mapsto (l_i)_{i\in[0,s]_{\mathbb{N}}}$$

Beispiel 8.1.6:

Betrachten wir $s := \langle\!|p,(h,(l_0,\ldots,l_c))|\!\rangle \in Conf$, dann ist

- $0L(s) = (l_0)$
- $1L(s) = (l_0,l_1)$
- $2L(s) = (l_0,l_1,l_2)$
- $\ldots$
- $cL(s) = (l_0,\ldots,l_c)$

Wir werden nun eine für die *while*-Sprache übliche *SOS*-Semantik definieren. Da, wie bereits erwähnt, die Form und Auswertung von Ausdrücken in unserem Kontext eine nebensächliche Rolle einnimmt, sei im Folgenden $\downarrow$ ein Operator zur atomaren Auswertung von Ausdrücken.

Definition 8.1.7: [*SOS*.Semantik]

Wir spezifizieren den Ableitungsoperator $\rightarrowtriangle$ für die *while*-Sprache wie üblich durch Angabe entsprechender Inferenzregeln.

1. $$\frac{-}{\langle\!|skip,z|\!\rangle \rightarrowtriangle \langle\!|\langle\rangle,z|\!\rangle}$$

2. $$\frac{-}{\langle\!|Id := Exp,z|\!\rangle \rightarrowtriangle \langle\!|\langle\rangle,z[Id = Exp^{\downarrow}]|\!\rangle}\ [Id \in \{h,l_0,\ldots,l_c\}]$$

3. $$\frac{\langle\!|C_1,z|\!\rangle \rightarrowtriangle \langle\!|\langle\rangle,z'|\!\rangle}{\langle\!|\langle C_1;C_2\rangle,z|\!\rangle \rightarrowtriangle \langle\!|C_2,z'|\!\rangle}\ ,\ \frac{\langle\!|C_1,z|\!\rangle \rightarrowtriangle \langle\!|C_1',z'|\!\rangle}{\langle\!|\langle C_1;C_2\rangle,z|\!\rangle \rightarrowtriangle \langle\!|\langle C_1';C_2\rangle,z'|\!\rangle}$$

4.
$$\frac{-}{\langle\!\langle if\ (B)\ then\ \{C_1\}\ else\ \{C_2\}, z\rangle\!\rangle \rightarrowtriangle \langle\!\langle C_1, z\rangle\!\rangle}\ [\langle\!\langle B,z\rangle\!\rangle \downarrow True]$$

5.
$$\frac{-}{\langle\!\langle if\ (B)\ then\ \{C_1\}\ else\ \{C_2\}, z\rangle\!\rangle \rightarrowtriangle \langle\!\langle C_2, z\rangle\!\rangle}\ [\langle\!\langle B,z\rangle\!\rangle \downarrow False]$$

6.
$$\frac{-}{\langle\!\langle while\ (B)\ do\ \{C\}, z\rangle\!\rangle \rightarrowtriangle \langle\!\langle C; while\ (B)\ do\ \{C\}, z\rangle\!\rangle}\ [\langle\!\langle B,z\rangle\!\rangle \downarrow True]$$

7.
$$\frac{-}{\langle\!\langle while\ (B)\ do\ \{C\}, z\rangle\!\rangle \rightarrowtriangle \langle\!\langle\rangle, z\rangle\!\rangle}\ [\langle\!\langle B,z\rangle\!\rangle \downarrow False]$$

Sicherheit

In diesem Abschnitt werden wir zunächst die $pTrace$-Sicherheit für *while*-Programme vorstellen. Anschließend werden wir eine Noninterferenz-Spezifikation formulieren, von der wir zeigen werden, dass sie dieselbe Sicherheit definiert.

Definition 8.1.8: [$pTrace$-Sicherheit]

Wir bezeichnen ein $p \in Prog$ als *$pTrace$-sicher in Stufe* $s \in [0,n]_{\mathbb{N}}$ bzw. kurz als *s-$pTrace$-sicher* gdw.

$$\forall \langle\!\langle p,z\rangle\!\rangle =_{sL} \langle\!\langle p,z'\rangle\!\rangle : \langle\!\langle p,z\rangle\!\rangle \rightarrowtriangle \langle\!\langle p_1,z_1\rangle\!\rangle \rightarrowtriangle \ldots \rightarrowtriangle \langle\!\langle p_k,z_k\rangle\!\rangle$$
$$\Rightarrow\ \langle\!\langle p,z'\rangle\!\rangle \rightarrowtriangle \langle\!\langle p'_1,z'_1\rangle\!\rangle \rightarrowtriangle \ldots \rightarrowtriangle \langle\!\langle p'_k,z'_k\rangle\!\rangle\ \wedge\ \forall i \in \{1,\ldots,k\} :\ \langle\!\langle p,z_i\rangle\!\rangle =_{sL} \langle\!\langle p,z'_i\rangle\!\rangle$$

Definition 8.1.9: [*while*-System]

Wir definieren das *while-System* $\mathcal{W}^{(s)}_{<\mathfrak{a},\mathfrak{p}>}$ in Abhängigkeit von dem Parameter $s \in [0,c]_{\mathbb{N}}$ durch

- $\mathcal{W} = (Conf, \rightarrowtriangle)$
- $\mathfrak{a} = (sL, \tau)$
- $\mathfrak{p}(\langle\!\langle p,z\rangle\!\rangle) = p$

Bemerkung 8.1.10:

Es ist offensichtlich, dass $\mathcal{W}$ deterministisch und somit insbesondere endlich verzweigt ist.

Definition 8.1.11: [Noninterferenz-Spezifikation]

Wir definieren die mit $s \in [0,c]_{\mathbb{N}}$ parametrisierte kanonischen p-Noninterferenz-Spezifikation $pTrace^{(s)}_{<p>}$ durch

$$pTrace^{(s)}_{<p>} := (\mathcal{W}^{(s)}_{<\mathfrak{a},\mathfrak{p}>}, \mathcal{C}_{kan}, \mathfrak{F}^{\ll pTr \gg})$$

Theorem 8.1.12: [Kongruenz-Theorem]

Sei $p_0 \in Prog$ ein beliebiges Programm. Es gilt

$$p_0 \text{ ist } s\text{-}pTrace\text{-sicher } gdw.\ pTrace^{(s)}_{<p>} \text{ ist erfüllt}$$

Beweis.
Wir zeigen die Implikationen " $\Rightarrow$ " und " $\Leftarrow$ " getrennt.

- [$\Rightarrow$]

  Laut Proposition 5.1.5 reicht es für beliebige Konfigurationen $\langle\!|p_0,z_0|\!\rangle$ und $\langle\!|p_0',z_0'|\!\rangle$ mit $p_0 = p_0'$ und $\langle\!|p,z_0|\!\rangle =_{sL} \langle\!|p,z_0'|\!\rangle$ zu zeigen, dass

  $$(\mathcal{W}_{<sL,\mathfrak{p}>},\ p_0,\ [\mathcal{W}_{<sL>}\langle\langle\!|p_0,z_0|\!\rangle\rangle]) =_{\mathfrak{F}^{\ll pTr \gg}} (\mathcal{W}_{<sL,\mathfrak{p}>},\ p_0',\ [\mathcal{W}_{<sL>}\langle\langle\!|p_0',z_0'|\!\rangle\rangle])$$

  Sei im Folgenden $s_0 := \langle\!|p_0,z_0|\!\rangle$ sowie $s_0' := \langle\!|p_0',z_0'|\!\rangle$

  Wir zeigen hier lediglich die Inklusion " $\subseteq$ ". Die Inklusion " $\supseteq$ " folgt analog.

  Sei also
  $$a_0 \xrightarrow{\tau} \ldots \xrightarrow{\tau} a_{n-1} \in \mathfrak{F}^{\ll pTr \gg}((\mathcal{W}_{<sL,\mathfrak{p}>}, p_0, [\mathcal{W}_{<sL>}\langle s_0\rangle]))$$

  Es gilt $\mathcal{W}_{<sL>}\langle s_0\rangle \vDash a_0 \xrightarrow{\tau} \ldots \xrightarrow{\tau} a_{n-1}$. Folglich existiert ein Pfad $\pi \in (\mathcal{W}^{\downarrow\{p_0,z_0\}}[(\epsilon,s_0)])^{(n)}$ der Form
  $$\pi = (V_0,s_0) \to (V_1,s_1) \to \ldots (V_{n-1},s_{n-1})$$
  mit $\mathcal{W}^{\downarrow s_0}_{<sL>}[\pi_i] \vDash a_i \xrightarrow{\tau} \ldots \xrightarrow{\tau} a_{n-1}$.

  Laut Voraussetzung existiert dann aber auch ein Pfad $\pi' \in (\mathcal{W}^{\downarrow s_0'}[(\epsilon,s_0')])^{(n)}$ der Form
  $$\pi' = (V_0',s_0') \to (V_1',s_1') \to \ldots (V_{n-1}',s_{n-1}')$$
  mit $sL(\pi_i) = sL(\pi_i')$. Es ergibt sich $\mathcal{W}^{\downarrow s_0'}_{<sL>}[\pi_i'] \vDash a_i \xrightarrow{\tau} \ldots \xrightarrow{\tau} a_{n-1}$ und somit insbesondere $\mathcal{W}_{<sL>}\langle s_0'\rangle \vDash a_0 \xrightarrow{\tau} \ldots \xrightarrow{\tau} a_{n-1}$ was gleichbedeutend ist mit
  $$a_0 \xrightarrow{\tau} \ldots \xrightarrow{\tau} a_{n-1} \in \mathfrak{F}^{\ll pTr \gg}((\mathcal{W}_{<sL,\mathfrak{p}>}, p_0', [\mathcal{W}_{<sL>}\langle s_0'\rangle]))$$

- [$\Leftarrow$]

Wir müssen für beliebige Konfigurationen $\langle p_0,z_0\rangle$ und $\langle p'_0,z'_0\rangle$ mit $p_0 = p'_0$ und $\langle p_0,z_0\rangle =_{sL} \langle p_0,z'_0\rangle$ zeigen, dass

$$\langle p_0,z_0\rangle \rightarrowtriangle \langle p_1,z_1\rangle \rightarrowtriangle \ldots \rightarrowtriangle \langle p_k,z_k\rangle$$

die Existenz von Konfigurationen $\langle p'_1,z'_1\rangle, \ldots \langle p'_k,z'_k\rangle \in Conf$ impliziert mit

$$\langle p'_0,z'_0\rangle \rightarrowtriangle \langle p'_1,z'_1\rangle \rightarrowtriangle \ldots \rightarrowtriangle \langle p'_k,z'_k\rangle$$

sowie $\langle p_i,z_i\rangle =_{sL} \langle p'_i,z'_i\rangle$.
Sei im Folgenden $s_0 := \langle p_0,z_0\rangle$ sowie $s'_0 := \langle p'_0,z'_0\rangle$.

Gelte also

$$s_0 \rightarrowtriangle s_1 \rightarrowtriangle \ldots \rightarrowtriangle s_k$$

Es gilt für $\pi \in (\mathcal{W}^{\downarrow s_0}[(\epsilon, s_0)])^{(n)}$ mit

$$\pi = (\epsilon,s_1) \rightarrow \ldots \rightarrow ((s_0,\ldots,s_{k-1}),s_k)$$

dass $\mathcal{W}^{\downarrow s_0}_{<sL>}[\pi_i] \vDash sL(s_i) \stackrel{\tau}{\rightarrow} \ldots \stackrel{\tau}{\rightarrow} sL(s_k)$ und somit insbesondere

$$\mathcal{W}_{<sL>}\langle s_0\rangle \vDash sL(s_0) \stackrel{\tau}{\rightarrow} \ldots \stackrel{\tau}{\rightarrow} sL(s_k)$$

Laut Voraussetzung gilt dann aber auch

$$\mathcal{W}_{<sL>}\langle s'_0\rangle \vDash sL(s_0) \stackrel{\tau}{\rightarrow} \ldots \stackrel{\tau}{\rightarrow} sL(s_k)$$

Es folgt die Existenz eines Pfades $\pi' \in (\mathcal{W}^{\downarrow s'_0}[(\epsilon, s'_0)])^{(n)}$ der Form

$$\pi' = (\epsilon,s'_0) \rightarrow \ldots \rightarrow ((s'_0,\ldots,s'_{k-1}), s'_k)$$

mit $\mathcal{W}^{\downarrow s'_0}_{<sL>}[\pi'_i] \vDash sL(s_i) \stackrel{\tau}{\rightarrow} \ldots \stackrel{\tau}{\rightarrow} sL(s_k)$. Es ergibt sich direkt

$$s'_0 \rightarrowtriangle s'_1 \rightarrowtriangle \ldots \rightarrowtriangle s'_k$$

sowie $s_i =_{sL} s'_i$. ✓

□

Aufgrund unserer Vorüberlegungen sind wir dazu in der Lage auf einfache Weise zwei weitreichende Theoreme abzuleiten. Erstere garantiert nachgewiesene Sicherheit bzgl. einer konkreten Benutzersicht auch für alle tieferen Sichten. Zweites Theorem belegt, dass die Angriffsstärke vieler bedeutender Beobachter bzgl. der *while*-Sprache nicht zur Geltung kommt.

Theorem 8.1.13: [sL-Verträglichkeit]

Sei $p_0 \in Prog$ eine beliebiges s-$pTrace$-sicheres Programm, dann ist p_0 auch s'-$pTrace$-sicher für jedes $s' \in [0,s]_{\mathbb{N}}$

Beweis.
Aufgrund des Kongruenz-Theorem 8.1.12 reicht es hierfür zu zeigen, dass $s'L \preceq sL$, die Behauptung folgt anschließend anhand der in Beispiel 7.3.5 gezeigten $r.a.i.$-Eigenschaft von $\mathfrak{F}^{\ll pTr \gg}$, zusammen mit dem Ordnungstheorem 6.2.9.
Dies ist aber offensichtlich, da $s'L = f \circ sL$ für $f((l_i)_{i \in [0,s]_{\mathbb{N}}}) = (l_i)_{i \in [0,s']_{\mathbb{N}}}$.
□

Theorem 8.1.14: [Koinzidenz-Theorem]

Für jede Semantik $\mathfrak{s}$ aus Abbildung 7.7 gilt

$$(\mathcal{W}, sL, \mathfrak{p}, \mathfrak{F}^{\ll pTrace \gg}) \equiv (\mathcal{W}, sL, \mathfrak{p}, \mathfrak{F}^{\ll \mathfrak{s} \gg})$$

Beweis.
Sei $\mathfrak{s}$ eine beliebige Semantik aus Abbildung 7.7. Es gilt $pTr \preceq \mathfrak{s} \preceq bisim$ und laut Proposition 7.4.20 ebenfalls $\mathfrak{F}^{\ll pTr \gg} \equiv_{(Det)} \mathfrak{F}^{\ll \mathfrak{s} \gg} \equiv_{(Det)} \mathfrak{F}^{\ll bisim \gg}$. Da $\mathcal{W}$ deterministisch ist ergibt sich die Behauptung mit Proposition 6.1.22.
□

8.2 Verzweigte Sprache

In diesem Abschnitt werden wir die in Abschnitt 8.1 eingeführte *while*-Sprache um Konzepte der nebenläufigen Programmierung erweitern. Anschließend werden wir eine Form von Noninterferenz-Sicherheit vorstellen, basierend auf probabilistischen Bisimulationen, von der wir zeigen werden, dass sie eine von uns passend definierte Noninterferenz-Spezifikation impliziert. Die Ungültigkeit der Rückrichtung werden mit einem simplen Gegenbeispiel widerlegen können welches zugleich belegt, dass die von uns definierte Noninterferenz-Spezifikation der intuitive Vorstellung der Autoren zu Noninterferenz-Sicherheit besser gerecht wird. Wir möchten erneut auf detaillierte Beispiele und Erläuterungen verzichten und verweisen hierfür auf [SS00]. Insbesondere setzten wir Grundkenntnisse der parallelen Programmierung voraus.

Syntax & Semantik

Die Syntax der parallelisierten *while*-Sprache deckt sich weitestgehend mit der aus Abschnitt 8.1. Diese wird lediglich um den Befehl $fork$ erweitert, welcher später der Thread-Generierung dienen soll.

Definition 8.2.1: [Syntax]

Die Syntax eines gültigen Programms der *while*-Sprache wird durch folgende eindeutige kontextfreie Grammatik, beschrieben (in BNF):

$$W ::= skip \mid Id := A \mid W;W \mid if(B)\ then\ \{W\}\ else\ \{W\} \mid while\ (B)\ do\ \{W\} \mid fork(\vec{W})$$

$$\vec{W} ::= W \mid \vec{W},W$$

$$A ::= < \text{arith. Ausdruck} > \ , \ B ::= < \text{bool. Ausdruck} >$$

Definition 8.2.2: [Programme]

Wir bezeichnen mit

- $Prog := \{p \mid W \Rightarrow^* p\}$ die Menge aller *Programme*
- $\vec{Prog} := \bigcup_{n\in\mathbb{N}} Prog^n = \{\langle p_0,\ldots,p_n\rangle \mid p_i \in Prog\}$ die Menge aller *Programm-Vektoren*

Nachdem wir die Syntax der *while* Sprache definiert haben, definieren wir nun einen geeignete Zustandsraum auf welche später die Semantik operieren soll. Wir verzichten hier der Übersicht halber auf die Generierung einer Hierarchie von Benutzersichten und betrachten lediglich einen bi-partitionierten Zustandsraum, bestehend aus einer H-Partition zur Kapselung geheimer Daten sowie einer L-Partition zur Kapselung öffentlicher Daten. Die Partitionen werden wir dabei durch natürliche Zahlen repräsentieren.

Definition 8.2.3: [Zustandsraum]

Wir bezeichnen mit $St := \{\ (h,l) \mid h,l \in \mathbb{N}\}$ die Menge der *Programmzustände*

Bemerkung 8.2.4:

Wir nutzen "$\langle\!|$" und "$|\!\rangle$" sowie "$\langle$" und "$\rangle$" aus Gründen der Übersicht als alternative Folgenklammern für Konfigurationen bzw. Programmen.

Definition 8.2.5: [Benutzersicht]

Wir definieren die Funktion $L : St \to \mathbb{N}$ durch

$$(h,l) \mapsto l$$

Wir wollen nun für obige Sprache eine passende Semantik definieren. Wir werden zu diesem Zweck zunächst den Begriff des Schedulers mathematisch erfassen und darauf aufbauend eine entsprechende Semantik definieren. Ein Scheduler soll auf formaler Ebene ein Konstrukt zur Selektion einzelner Komponenten eines Programm-Vektors darstellen. Die Selektion soll hierbei von folgenden Parametern abhängen.

1. Aktuelle Länge des Programm-Vektors
2. Historie des bisherigen Scheduler-Verhaltens

Unter der Historie des bisheriges Scheduler-Verhalten verstehen wir Informationen über getroffene Selektionen des Schedulers von Programmstart bis zum gegenwärtigen Zeitpunkt. Wie üblich werden wir Scheduler als nicht-deterministische Konstrukte mit einem stochastischen Verhalten auffassen. Dies bedeutet, dass Selektionen mit Wahrscheinlichkeiten einhergehen und in der Regel nicht fest vorbestimmt sind. Wir werden Scheduler daher als Abbildungen in Wahrscheinlichkeitsverteilungen definieren.

Definition 8.2.6: [History]

Als *History* bezeichnen wir Elemente aus

$$Hist := \mathbb{N} \times \mathbb{N}^*$$

wobei $\mathbb{N}^*$ für die Menge aller $\mathbb{N}$-Folgen steht. Wir bezeichnen $(1,\epsilon)$ auch als *initiale History*. Die Menge aller Historys bezeichnen wir mit $Hist$.

Definition 8.2.7: [Scheduler]

Ein *Scheduler* ist eine Funktion $\sigma : Hist \to \mathcal{D} \cup \{\bot\}$, wobei $\bot$ die eindeutig bestimmte Funktion ist mit $dom(\bot) = \varnothing$

$$\sigma((n,N)) \in \begin{cases} \mathcal{D}[n] & : n > 0 \\ \bot & : \text{sonst} \end{cases}$$

Ein Scheduler stellt auf formaler Ebene somit eine Funktion zur Abbildung von Tupel der Form (n,N) auf entsprechende Wahrscheinlichkeitsverteilung dar. Hierbei soll n die Anzahl der aktiven Threads widerspiegeln, N hingegen den bisherigen Selektions-Verlauf

des Schdedulers. Anhand dieser Werte wird anschließend eine passende Wahrscheinlichkeitsverteilung bestimmt, welche repräsentierend für die Ausführungswahrscheinlichkeiten der aktiven Threads steht. Wir wollen anhand eines kurzen Beispiels die Formalisierung zweier existierender Scheduler betrachten.

Beispiel 8.2.8:

Scheduler]

- $uni(n,N)(x) = \frac{1}{n}$
  formalisiert den Uniformen Scheduler.
- $roundr(n,(h_1,\ldots,h_k))(x) = \begin{cases} 1 & , (h_k < n \ \wedge \ x = i+1 \ MOD \ n) \vee \\ & \quad (h_k = n \ \wedge \ x = 0) \\ 0 & , sonst \end{cases}$
  formalisiert den Round-Robin-Scheduler

Wir werden nun als Basis unserer Semantik Konfigurationen der *while*-Sprache definieren. Diese werden sich aufteilen in einen deterministischen und einem nicht-deterministischen Part, welche sich dadurch unterscheiden, dass nicht-deterministische Konfigurationen zusätzliche Metainformationen, in Form von History's für den operierenden Scheduler bereitstellen.

Definition 8.2.9: [Konfigurationen]

Wir bezeichnen ...

- $Conf_{ndet} := \{ \ \langle\!\langle H,\vec{p},z \rangle\!\rangle \mid H \in Hist, \ \vec{p} \in \vec{Com}, \ z \in St\}$
  als die Menge der *nichtdeterministischen Konfigurationen*
- $Conf_{det} := \{ \ \langle\!\langle p,z \rangle\!\rangle \mid p \in Com, \ z \in St\}$
  als die Menge der *deterministischen Konfigurationen*
- $IK := \{ \ \langle\!\langle H,\vec{p},z \rangle\!\rangle \in Conf_{ndet} \mid H = (1,\epsilon)\}$
  als die Menge der *initialen Konfigurationen*
- $TK := \{ \ \langle\!\langle H,\vec{p},z \rangle\!\rangle \in Conf_{ndet} \mid \vec{p} = \langle\rangle\}$
  als die Menge der *terminalen Konfigurationen*

Definition 8.2.10: [SOS-Semantik]

Wir spezifizieren zwei Ableitungsoperatoren $\rightharpoonup$ und $\longrightarrow$ für die Ableitung deterministischer- bzw. nicht-deterministischer Konfigurationen der parallelisierten *while*-Sprache wie üblich durch Angabe entsprechender Inferenzregeln.

1. $$\frac{-}{\langle\!|skip,z|\!\rangle \rightharpoonup \langle\!|\langle\rangle,z|\!\rangle}$$

2. $$\frac{-}{\langle\!|Id := Exp,z|\!\rangle \rightharpoonup \langle\!|\langle\rangle,z[Id = n]|\!\rangle}\ [\langle\!|Exp,z|\!\rangle \downarrow n]$$

3. $$\frac{\langle\!|C_1,z|\!\rangle \rightharpoonup \langle\!|\langle\rangle,z'|\!\rangle}{\langle\!|C_1;C_2,z|\!\rangle \rightharpoonup \langle\!|\langle C_2\rangle,z'|\!\rangle}\ ,\ \frac{\langle\!|C_1,z|\!\rangle \rightharpoonup \langle\!|\langle C_1'\vec{D}\rangle,z'|\!\rangle}{\langle\!|C_1;C_2,z|\!\rangle \rightharpoonup \langle\!|\langle (C_1';C_2)\vec{D}\rangle,z'|\!\rangle}$$

4. $$\frac{-}{\langle\!|if\ (B)\ then\ \{C_1\}\ else\ \{C_2\},z|\!\rangle \rightharpoonup \langle\!|C_1,z|\!\rangle}\ [\langle\!|B,z|\!\rangle \downarrow True]$$

5. $$\frac{-}{\langle\!|if\ (B)\ then\ \{C_1\}\ else\ \{C_2\},z|\!\rangle \rightharpoonup \langle\!|C_2,z|\!\rangle}\ [\langle\!|B,z|\!\rangle \downarrow False]$$

6. $$\frac{-}{\langle\!|while\ (B)\ do\ \{C\},z|\!\rangle \rightharpoonup \langle\!|C;while\ (B)\ do\ \{C\},z|\!\rangle}\ [\langle\!|B,z|\!\rangle \downarrow True]$$

7. $$\frac{-}{\langle\!|while\ (B)\ do\ \{C\},z|\!\rangle \rightharpoonup \langle\!|\langle\rangle,z|\!\rangle}\ [\langle\!|B,z|\!\rangle \downarrow False]$$

8. $$\frac{-}{\langle\!|fork(C_0\ \dots\ C_{n-1}),z|\!\rangle \rightharpoonup \langle\!|C_0\dots C_{n-1},z|\!\rangle}$$

9. $$\frac{\langle\!|C_i,\ z|\!\rangle \rightharpoonup \langle\!|\vec{C},\ z'|\!\rangle}{\langle\!|(n,N),\ \langle C_0\ \dots\ C_{n-1}\rangle,\ z|\!\rangle \ \longrightarrow^{\sigma}\ \langle\!|(n-1+|\vec{C}|,N.(i)),\ \langle C_0\ \dots\ \underset{\underset{i}{\uparrow}}{\vec{C}}\ \dots\ C_{n-1}\rangle,\ z'|\!\rangle}$$

Bemerkung 8.2.11:

Man beachte dass die Regelen 1-8 Regeln der deterministischen Ableitungen von Programmen dienen und unabhängig von einem konkreten Schedulers σ operieren. Regel 9 beschreibt die nicht deterministische Ableitung von Programm-Vektoren in Abhängigkeit von den Regeln $1-8$.

Notation 8.2.12:

- Für eine Scheduler σ schreiben wir

$$\langle\!|(n,N),\ \vec{p},\ (h,l)|\!\rangle \longrightarrow_{\rho}^{\sigma} \langle\!|(m,N.(i)),\ \vec{p'},\ (h',l')|\!\rangle$$

  gdw.

  (i) $\langle\!|(n,N),\ \vec{p},\ (h,l)|\!\rangle \longrightarrow^{\sigma} \langle\!|(m,N.(i)),\ \vec{p'},\ (h',l')|\!\rangle$

  (ii) $\rho = \sigma((n,N))(i)$

- Für eine Scheduler σ, $s \in Conf_{ndet}$ und $M \subseteq Conf_{ndet}$ schreiben wir

$$s \longrightarrow_{\rho}^{\sigma} M \quad \text{gdw.} \quad \rho = \sum_{\substack{s \longrightarrow_{\rho_m}^{\sigma} m, \\ m \in M}} \rho_m$$

Sicherheit

In diesem Abschnitt werden wir zunächst probabilistische Bisimulationen definieren und aufbauend eine entsprechende Noninterferenz-Sicherheit, welche wir als σ-Sicherheit bezeichnen werden. Anschließend konstruieren wir eine korrespondierende Nointerferenz-Spezifikation und erarbeite die Zusammenhänge zur σ-Sicherheit. Dafür sei im Folgenden σ ein beliebiger aber fester Scheduler.

Definition 8.2.13: [probabilistische Bisimulation]

Eine partielle Äquivalenzrelation $R \in per(Conf_{ndet})$ ist eine *σ-spezifische partielle probabilistische Bisimulation ($\sigma - ppb$)* gdw. falls $(c,d) \in R$, dann

(i) $\forall S \in Conf_{ndet}/R:\ c \longrightarrow_{p}^{\sigma} S \ \Leftrightarrow d \longrightarrow_{p}^{\sigma} S$

(ii) $c,d \longrightarrow_{0}^{\sigma} Conf_{ndet} \setminus dom(R)$

Definition 8.2.14: [σ-spezifische partielle probabilistische L-Bisimulation]

Eine partielle Äquivalenzrelation $R \in per(\vec{Com})$ ist eine *σ-spezifische partielle probabilistische L-Bisimulation ($\sigma - ppLb$)* gdw. folgende Relation eine $\sigma - ppb_{Conf_{ndet}}$ ist

$$(=_{\sigma}) \otimes R \otimes (=_{L})$$

Proposition 8.2.15:

$\sim_L^\sigma := \{R \mid R \text{ ist } \sigma - ppLb\}$ ist größte $\sigma - ppLb$

Beweis.
Wir verweisen hier auf [SS00].

□

Definition 8.2.16: [σ-Sicherheit]

Wir bezeichnen ein Programm p als *σ-sicher* gdw.

$$p \in dom(\sim_L^\sigma)$$

Definition 8.2.17: [parallelisiertes *while*-System]

Wir definieren das parallelisierte *while*-System $\mathcal{W}_{<\sigma L,\mathfrak{p}>}$ durch

- $\mathcal{W} := (Conf_{ndet}, \longrightarrow^\sigma)$
- $(\sigma L)_S : Conf_{ndet} \to (\mathcal{D} \times \mathbb{N})$ mit

$$\langle\!|H,\vec{p},z|\!\rangle \mapsto (\sigma(H),L(z))$$

- $(\sigma L)_T : Conf_{ndet}^2 \to [0,1]_{\mathbb{R}}$ mit

$$(\ \langle\!|(n,N),\vec{p},z|\!\rangle \to \langle\!|(m,N.(i)),\vec{p'},z'|\!\rangle\ ) \ \mapsto\ \sigma((n,N))(i)$$

- $\mathfrak{p} : Conf_{ndet} \to \vec{Prog}$ mit

$$\langle\!|H,\vec{p},z|\!\rangle \mapsto \vec{p}$$

Bemerkung 8.2.18:

Es ist offensichtlich, dass $\mathcal{W}$ endlich verzweigt ist.

Definition 8.2.19: [Noninterferenz-Spezifikation]

Wir definieren die mit σ parametrisierte kanonischen p-Noninterferenz-Spezifikation

$$prbBsm_{<p>}^{(\sigma)} := (\mathcal{W}_{<\sigma L,\mathfrak{p}>}, C_{kan}, \mathfrak{F}^{\ll prob,bisim\gg})$$

Theorem 8.2.20: [Kongruenz-Theorem]

Sei $p_0 \in P_{\mathcal{W}}$ ein beliebiges Programm. Es gilt

$$p_0 \text{ ist } \sigma\text{-sicher} \Rightarrow prbBsm^{(\sigma)}_{<p_0>} \text{ ist erfüllt}$$

Beweis.
Laut Proposition 5.1.5 reicht es für beliebige Konfigurationen $s_0 := \langle\!| H_0, p_0, z_0 |\!\rangle$ und $s_0' := \langle\!| H_0', p_0', z_0' |\!\rangle$ mit $p_0 = p_0'$ und $s_0 =_{\sigma L} s_0'$ zu zeigen, dass

$$(\mathcal{W}_{<\sigma L, \mathfrak{p}>},\ p_0,\ [\mathcal{W}_{<\sigma L>}\langle s_0\rangle]) =_{\mathfrak{F}^{\ll prob, bisim \gg}} (\mathcal{W}_{<\sigma L, \mathfrak{p}>},\ p_0',\ [\mathcal{W}_{<sL>}\langle s_0'\rangle])$$

Laut Proposition 7.2.31 reicht es hierfür wiederum die Existenz einer Bisimulation zwischen $prob(\mathcal{W}_{<\sigma L>}\langle s_0\rangle)$ und $prob(\mathcal{W}_{<\sigma L>}\langle s_0'\rangle$ nachzuweisen.

Im Folgenden sei $\mathcal{W}^{\Downarrow_s}$ gerade das Transitionssystem mit

$$\mathcal{W}^{\Downarrow_s}_{<\sigma L^{\downarrow}>}[(\epsilon, s)] = prbRed(\mathcal{W}^{\downarrow_s}_{<\sigma L^{\downarrow}>}[(\epsilon, s)])$$

d.h. das durch $prbRed(\mathcal{W}^{\downarrow_s}_{<\sigma L^{\downarrow}>}[(\epsilon, s)]$ induzierte Transitionssystem.
Weiterhin seien

- R_{max} die größte probabilistische $prbRed(\mathcal{W}_{<\sigma L>}\langle s_0\rangle)$-Bisimulation
- R'_{max} die größte probabilistische $prbRed(\mathcal{W}_{<\sigma L>}\langle s_0'\rangle)$-Bisimulation

Außerdem sei $R := (=_\sigma) \otimes \sim_L^\sigma \otimes (=_L)$.
Wir werden den Beweis nun in verschiedene Schritte zerlegen. Im ersten Schritt zeigen wir

- $R_{ppb} \in rel(S^{(\mathcal{W}^{\Downarrow_{s_0}})})$ mit

  $$((V, s), (V', s')) \in R_{ppb} \text{ gdw. } (s, s') \in R$$

  ist probabilistische $prbRed(\mathcal{W}_{<\sigma L>}\langle s_0\rangle)$-Bisimulation

- $R'_{ppb} \in rel(S^{(\mathcal{W}^{\Downarrow_{s_0'}})})$ mit

  $$((V, s), (V', s')) \in R'_{ppb} \text{ gdw. } (s, s') \in R$$

  ist probabilistische $prbRed(\mathcal{W}_{<\sigma L>}\langle s_0'\rangle)$-Bisimulation

Hieraus ergibt sich, dass R_{ppb} eine Verfeinerung von R_{max} darstellt genau wie R'_{ppb} eine Verfeinerung R'_{max} darstellt. Ignorieren wir die durch die Ausfaltung gegebene Struktur besagt dies anschaulich, dass R_{max}-Äquivalenzklassen die disjunkte Vereinigung "passend eingeschränkter" R-Partitionen darstellen.

Anschließend zeigen wir im zweiten Schritt

- Für jedes $S \in S^{(\mathcal{W}^{\Downarrow s_0})}/_{R_{max}}$ existiert ein $S' \in S^{(\mathcal{W}^{\Downarrow s'_0})}/_{R'_{max}}$ mit

$$\bigcup_{(V,s)\in S} [s]_R = \bigcup_{(V',s')\in S'} [s']_R$$

- Für jedes $S' \in S^{(\mathcal{W}^{\Downarrow s'_0})}/_{R'_{max}}$ existiert ein $S \in S^{(\mathcal{W}^{\Downarrow s_0})}/_{R_{max}}$ mit

$$\bigcup_{(V',s')\in S} [s']_R = \bigcup_{(V,s)\in S} [s]_R$$

Anschaulich besagt dies für beliebiges $S \in S^{(\mathcal{W}^{\Downarrow s_0})}/_{R_{max}}$, dass wenn dies durch die Partitionen $(P_i \in Conf/_R)_{i\in I}$ induziert wird es stets ein $S' \in S^{(\mathcal{W}^{\Downarrow s'_0})}/_{R'_{max}}$ gibt welches ebenfalls durch die P_i induziert wird.

Im dritten und letzten Schritt zeigen wir dann, dass $R_{bsm} \subseteq S^{(\mathcal{W}^{\Downarrow s_0})} \times S^{(\mathcal{W}^{\Downarrow s'_0})}$ mit

$$([(V,s)]_{R_{max}},[V',s']_{R'_{max}}) \in R_{bsm} \text{ gdw.}$$

$$\exists (V_{tmp},s_{tmp}) \in [(V,s)]_{R_{max}}, (V'_{tmp},s'_{tmp}) \in [(V',s')]_{R'_{max}} : \ (s_{tmp},s'_{tmp}) \in R$$

eine Bisimulation zwischen $prob(\mathcal{W}_{<\sigma L>}\langle s_0\rangle)$ und $prob(\mathcal{W}_{<\sigma L>}\langle s'_0\rangle)$ darstellt.

1.Schritt:

Wir zeigen $R_{ppb} \in rel(S^{(\mathcal{W}^{\Downarrow s_0})})$ mit

$$((V,s),(V',s')) \in R_{ppb} \text{ gdw. } (s,s') \in R$$

ist probabilistische $prbRed(\mathcal{W}_{<\sigma L>}\langle s_0\rangle)$-Bisimulation. Der Beweis dass R'_{ppb} eine probabilistische $prbRed(\mathcal{W}_{<\sigma L>}\langle s'_0\rangle)$-Bisimulation darstellt ergibt sich analog.
Wir müssen also für $((V,s),(V',s')) \in R_{ppb}$ die Gültigkeit folgender Punkte nachweisen.

- [R_{ppb} ist Äquivalenzrelation]

  Wir weisen lediglich die Reflexivität nach. Symmetrie und Transitivität ergeben sich unmittelbar aus den entsprechenden Eigenschaften von R.

  Für den Nachweis der Reflexivität sei $(V,s) \in S^{(\mathcal{W}^{\Downarrow s_0})}$ beliebig. Da $\mathcal{W}^{\Downarrow s_0}_{<\sigma L^{\downarrow}>}[(\epsilon,s_0)]$ den probabilistisch reduzierten Zustandsgraph von $\mathcal{W}^{\downarrow s_0}_{<\sigma L^{\downarrow}>}[(\epsilon,s_0)]$ beschreibt existieren $V_1,\ldots V_n \in \mathcal{W}^{(*)}$ sowie $s_1,\ldots,s_n \in S^{(\mathcal{W})}$ mit

$$(\epsilon,s_0) \xrightarrow{\rho_0>0} (V_1,s_1) \xrightarrow{\rho_1>0} \ldots \xrightarrow{\rho_{n-1}>0} (V_n,s_n) \xrightarrow{\rho_n>0} (V,s)$$

  Folglich gilt auch

$$s_0 \xrightarrow{\rho_0>0} s_1 \xrightarrow{\rho_1>0} \ldots \xrightarrow{\rho_{n-1}>0} s_n \xrightarrow{\rho_n>0} s$$

und somit insbesondere $s \in dom(R)$. Es folgt somit $(s,s) \in R$ und somit ebenfalls die Behauptung. ✓

- $[(V,s) =_{\sigma L_S^{\downarrow}} (V',s')]$

  Es gilt $s,s' \in R$. Per Definition von R gilt $s =_\sigma s'$ sowie $s =_L s'$. Es folgt $s =_{\sigma L_s} s'$ und somit auch die Behauptung. ✓

- $[\forall S \in S^{(\mathcal{W}^{\Downarrow s_0})}/_{R_{ppb}} : \ (V,s) \xrightarrow{\rho} S \Leftrightarrow (V',s') \xrightarrow{\rho} S]$

  Sei $S \in S^{(\mathcal{W}^{\Downarrow s_0})}/_{R_{ppb}}$ beliebig. Per Definition von R_{ppb} existiert ein $U \in S^{(\mathcal{W})}/_R$ mit $\{s \mid (V,s) \in S\} \subseteq U$. Für dieses folgt per Definition von R die Existenz eines $\rho \in [0,1]_{\mathbb{R}}$ mit $s \xrightarrow{\rho} U \Leftrightarrow s' \xrightarrow{\rho} U$. Hieraus ergibt sich für

  – $T := \{(V.(s), u) \mid (V.(s), u) \in S^{(\mathcal{W}^{\Downarrow s_0})}\}_{u \in U}$
  – $T' := \{(V'.(s'), u) \mid (V'.(\langle\!| p'.z' |\!\rangle), u) \in S^{(\mathcal{W}^{\Downarrow s_0})}\}_{u \in U}$

  dass $(V,s) \xrightarrow{\rho} T \Leftrightarrow (V',s') \xrightarrow{\rho} T'$ denn für beliebiges $u \in U$ gilt $s \xrightarrow{\rho'>0} u$ gdw. $(V.(s), u) \in T$ und folglich auch $(V,s) \xrightarrow{\rho} S \Leftrightarrow (V',s') \xrightarrow{\rho} S'$, da $(V,s) \xrightarrow{0} S \setminus T$ genau wie $(V,s') \xrightarrow{0} S \setminus T'$. ✓

Dass $R'_{ppb} \in rel(S^{(\mathcal{W}^{\Downarrow s'_0})})$ mit

$$((V,s),(V',s')) \in R'_{ppb} \text{ gdw. } (s,s') \in R$$

probabilistische $prbRed(\mathcal{W}_{<\sigma L>}\langle s'_0\rangle)$-Bisimulation ist folgt analog.

2.Schritt:

Wir zeigen für beliebiges $S \in S^{(\mathcal{W}^{\Downarrow s_0})}/_{R_{max}}$ die Existenz eines $S' \in S^{(\mathcal{W}^{\Downarrow s'_0})}/_{R'_{max}}$ mit

$$\bigcup_{(V,s)\in S} [s]_R = \bigcup_{(V',s')\in S'} [s']_R$$

Der Beweis, dass für beliebiges $S' \in S^{(\mathcal{W}^{\Downarrow s'_0})}/_{R'_{max}}$ ein $S \in S^{(\mathcal{W}^{\Downarrow s_0})}/_{R_{max}}$ existiert mit

$$\bigcup_{(V',s')\in S} [s']_R = \bigcup_{(V,s)\in S} [s]_R$$

ergibt sich analog.
Hierfür zeigen wir, dass $R_{tmp} \in rel(S^{(\mathcal{W}^{\Downarrow s_0})})$ mit

$$((V,s),(V',s')) \in R_{tmp} \text{ gdw. } \exists S' \in S^{(\mathcal{W}^{\Downarrow s'_0})}/_{R'_{max}} : \ [s]_R \cup [s']_R \subseteq \bigcup_{(V,s)\in S'} [s]_R$$

eine probabilistische $prbRed(\mathcal{W}_{<\sigma L>}\langle s_0\rangle)$-Bisimulation darstellt.

Vollkommen analog lässt sich beweisen, dass $R'_{tmp} \in rel(S^{(\mathcal{W}^{\Downarrow_{s'_0}})})$ mit

$$((V,s),(V',s')) \in R'_{tmp} \text{ gdw. } \exists S \in S^{(\mathcal{W}^{\Downarrow_{s_0}})}/_{R_{max}} : \ [s]_R \cup [s']_R \subseteq \bigcup_{(V,s)\in S} [s]_R$$

eine probabilistische $prbRed(\mathcal{W}_{<\sigma L>}\langle s'_0\rangle)$-Bisimulation darstellt.
Fasst man nun beide Aussagen zusammen ergibt sich $R_{tmp} = R_{max}$ genau wie $R'_{tmp} = R'_{max}$ und folglich auch die Behauptung.
Wir müssen also für $(V,s),(V',s') \in R_{tmp}$ die Gültigkeit folgender Punkte nachweisen.

- $[R_{tmp}$ ist Äquivalenzrelation]

  Wir weisen lediglich die Reflexivität nach. Symmetrie und Transitivität sind offensichtlich.

  Für den Nachweis der Reflexivität sei $(V,s) \in S^{(\mathcal{W}^{\Downarrow_{s_0}})}$ beliebig. Wir müssen zeigen $\exists S' \in S^{(\mathcal{W}^{\Downarrow_{s'_0}})}/_{R'_{max}} : \ [s]_R \subseteq \bigcup_{(V,s)\in S'}[s]_R$. Wir zeigen hierfür die Existenz eines $(V',s') \in S^{(\mathcal{W}^{\Downarrow_{s'_0}})}$ mit $(s,s') \in R$.

  Da $(V,s) \in S^{(\mathcal{W}^{\Downarrow_{s_0}})}$ folgt anlog zu oben die Existenz von $s_1,\ldots,s_n \in S^{(\mathcal{W})}$ mit $s_0 \xrightarrow{\rho_0>0} s_1 \xrightarrow{\rho_1>0} \ldots \xrightarrow{\rho_{n-1}>0} s_n \xrightarrow{\rho_n>0} s$. Da laut Voraussetzung $(s_0,s'_0) \in R$ folgt die Existenz von von $s'_1,\ldots,s'_n \in S^{(\mathcal{W})}$ mit $s'_0 \xrightarrow{\rho_0} s'_1 \xrightarrow{\rho_1} \ldots \xrightarrow{\rho_{n-1}} s'_n \xrightarrow{\rho_n>0} s'$ sowie $(s_i,s'_i) \in R$, als auch $(s,s') \in R$. Es folgt für $V' := (s'_0,\ldots,s_n)$, dass $(V',s') \in S^{(\mathcal{W}^{\Downarrow_{s'_0}})}$. ✓

- $[(V,s) =_{\sigma L^{\downarrow}_S} (V',s')]$

  Laut Voraussetzung gibt es ein $S' \in S^{(\mathcal{W}^{\Downarrow_{s'_0}})}/_{R'_{max}}$ mit $[s]_R \cup [s']_R \subseteq \bigcup_{(V,s)\in S'}[s]_R$, d.h. es existieren $(V_t,t),(V_{t'},t') \in S'$ mit $(s,t) \in R$ und $(s',t') \in R$. Es gilt nun $(V_t,t) =_{\sigma L^{\downarrow}_S} (V_{t'},t')$ und somit auch $t =_{\sigma L_S} t'$. Folglich gilt dann aber auch $s =_{\sigma L_S} s'$ und somit auch $(V,s) =_{\sigma L^{\downarrow}_S} (V',s')$. ✓

- $[\forall S \in S^{(prbRed(\mathcal{W}_{<\sigma L>}\langle s_0\rangle))}/_{R_{tmp}} : \ (V,s) \xrightarrow{p} S' \Leftrightarrow (V',s') \xrightarrow{p} S]$

  Wir zeigen lediglich die Implikation " $\Rightarrow$ ", die Implikation " $\Leftarrow$ " folgt analog.

  Sei also $S \in S^{(\mathcal{W}^{\Downarrow_{s_0}})}/_{R_{tmp}}$ beliebig. Per Definition von R_{tmp} existiert ein $S' \in S^{(\mathcal{W}^{\Downarrow_{s'_0}})}/_{R'_{max}}$ mit $\bigcup_{(V,s)\in S}[s]_R = \bigcup_{(V,s)\in S'}[s]_R$. Ebenfalls folgt die Existenz von $(V_t,t),(V_{t'},t') \in S^{(\mathcal{W}^{(\Downarrow_{s'_0})})}$ mit $((V_t,t),(V_{t'},t')) \in R'_{max}$ für die $(s,t) \in R$ und $(s',t') \in R$. Nun gilt $(V_t,t) \xrightarrow{p} S' \Leftrightarrow (V_{t'},t') \xrightarrow{p} S'$. Laut "Schritt 1" gilt auch

  $$t \xrightarrow{p} \bigcup_{(V,s)\in S'} [s]_R \Leftrightarrow t' \xrightarrow{p} \bigcup_{(V,s)\in S'} [s]_R$$

und da $\bigcup_{(V,s)\in S'}[s]_R$ eine Vereinigung von R-Äquivalenzklassen darstellt somit auch $s \xrightarrow{p} \bigcup_{(V,s)\in S'}[s]_R \Leftrightarrow s' \xrightarrow{p} \bigcup_{(V,s)\in S'}[s]_R$ was insbesondere bedeutet $s \xrightarrow{p} \bigcup_{(V,s)\in S}[s]_R \Leftrightarrow s' \xrightarrow{p} \bigcup_{(V,s)\in S}[s]_R$. Hieraus ergibt sich mit analoger Argumentation wie im Beweis von "Schritt 1", dass $(V,s) \xrightarrow{p} S \Leftrightarrow (V',s') \xrightarrow{p} S$. ✓

3.Schritt:

Wir müssen zeigen, dass $R_{bsm} \subseteq S^{(\mathcal{W}^{\downarrow s_0})} \times S^{(\mathcal{W}^{\downarrow s'_0})}$ mit

$$([(V,s)]_{R_{max}},\ [(V',s')]_{R'_{max}}) \in R_{bsm} \text{ gdw.}$$

$$\exists (V_{tmp},s_{tmp}) \in [(V,s)]_{R_{max}},\ (V'_{tmp},s'_{tmp}) \in [(V',s')]_{R'_{max}} :\ (s_{tmp},s'_{tmp}) \in R$$

eine Bisimulation zwischen $prob(\mathcal{W}_{<\sigma L>}\langle s_0\rangle)$ und $prob(\mathcal{W}_{<\sigma L>}\langle s'_0\rangle)$ darstellt.
Sei im Folgenden

- $\mathcal{P}_{<\mathfrak{a}>}[[(\epsilon,s_0)]] := prob(\mathcal{W}_{<\sigma L>}\langle s_0\rangle)$
- $\mathcal{P}'_{<\mathfrak{a}'>}[[(\epsilon,s'_0)]] := prob(\mathcal{W}_{<\sigma L>}\langle s'_0\rangle)$

Wir müssen die Gültigkeit folgender Punkte nachweisen.

- $[([(\epsilon,s_0)]_{R_{max}},[(\epsilon,s_1)]_{R_{max}}) \in R_{bsm}]$
  Offensichtlich gilt $(\epsilon,s_0) \in [(\epsilon,s_0)]_{R_{max}}$ genau wie $(\epsilon,s_1) \in [(\epsilon,s_1)]_{R'_{max}}$. Laut Voraussetzung gilt außerdem $(s_0,s_1) \in R$ und somit auch $([(\epsilon,s_0)]_{R_{max}},[(\epsilon,s_1)]_{R'_{max}}) \in R_{bsm}$. ✓
- $[\mathfrak{a}_S([(V,s)]_{R_{max}} = \mathfrak{a}'_S([(V',s')]_{R_{max}})]$
  Es gibt $(V_{tmp},s_{tmp}) \in [(V,s)]_{R_{max}}$ sowie $(V'_{tmp},s'_{tmp}) \in [(V',s')]_{R'_{max}}$ mit $(s_{tmp},s'_{tmp}) \in R$. Per Definition gilt $s_{tmp} =_{\sigma L_S} s'_{tmp}$. Folglich gilt $(V_{tmp},s_{tmp}) =_{\sigma L^{\downarrow}_S} (V_{tmp},s_{tmp})$ und somit auch $\mathfrak{a}_S([(V,s)]_{R_{max}}) =_{\mathfrak{a}'_S} ([(V',s')]_{R'_{max}})$. ✓
- $[\ [(V,s)]_{R_{max}} \to [(W,t)]_{R_{max}} \ \Rightarrow$
  $\exists [(W',t')]_{R'_{max}} :\ [(V',s')]_{R'_{max}} \to [(W',t')]_{R'_{max}} \ \wedge$
  $\mathfrak{a}_T([(V,s)]_{R_{max}} \to [(W,t)]_{R_{max}}) = \mathfrak{a}'_T([(V',s')]_{R'_{max}} \to [(W',t')]_{R'_{max}}) \ \wedge$
  $([(W,t)]_{R_{max}},[(W',t')]_{R'_{max}}) \in R_{bsm}\ ]$
  Gelte $[(V,s)]_{R_{max}} \to [(W,t)]_{R_{max}} \in T^{(\mathcal{P})}$. In "Schritt 1" haben wir die Existenz von $(P_i \in S^{(\mathcal{W})}/_R)_{i\in I}$ bewiesen, mit $[(W,t)]_{R_{max}} = \{(V,s) \in S^{(\mathcal{W}^{\Downarrow s_0})} \mid s \in \bigcup_{i\in I} P_i\}$.
  In "Schritt 2" haben wir gezeigt, dass $\{(V',s') \in S^{(\mathcal{W}^{\Downarrow s'_0})} \mid s' \in \bigcup_{i\in I} P_i\} \in S^{(\mathcal{W}^{\Downarrow s'_0})}/_{R'_{max}}$.
  Sei im Folgenden $(W',t') \in \{(V',s') \in S^{(\mathcal{W}^{\Downarrow s'_0})} \mid s' \in \bigcup_{i\in I} P_i\}$ beliebig aber fest.
  Seien nun $(\rho_i \in [0,1]_{\mathbb{R}})_{i\in I}$ mit $s \xrightarrow{\rho_i} P_i$, dann gilt auch $s' \xrightarrow{\rho_i} P'_i$ und somit insbesondere $(V,s) \xrightarrow{\sum_i \rho_i} [(W,t)]_{R_{max}}$ genau wie $(V',s') \xrightarrow{\sum_i \rho_i} [(W',t')]_{R'_{max}}$, was

gleichbedeutend ist mit

$$\mathfrak{a}_T([(V,s)]_{R_{max}} \to [(W,t)]_{R_{max}}) = \mathfrak{a}'_T([(V',s')]_{R'_{max}} \to [(W',t')]_{R'_{max}})$$

Es bleibt zu zeigen, dass $([(W,t)]_{R_{max}},\ [(W',t')]_{R'_{max}}) \in R_{bsm}$.

Da $[(W,t)]_{R_{max}} = \{(V,s) \in S^{(\mathcal{W}^{\Downarrow s_0})} \mid s \in \bigcup_{i\in I} P_i\}$ sowie $[(W',t')]_{R'_{max}}\{(V',s') \in S^{(\mathcal{W}^{\Downarrow s'_0})} \mid s' \in \bigcup_{i\in I} P_i\}$ folgt für beliebiges festes i die Existenz von Repräsentanten $(W_0,t_0) \in [(W,t)]_{R_{max}}$ sowie $(W'_0,t'_0) \in [(W',t')]_{R'_{max}}$ mit $(t_0,t'_0) \in P_i \subseteq R$. ✓

- $[\ [(V',s')]_{R'_{max}} \to [(W',t')]_{R'_{max}} \ \Rightarrow$
  $\exists [(W,t)]_{R_{max}} :\ [(V,s)]_{R_{max}} \to [(W,t)]_{R_{max}} \ \wedge$
  $\mathfrak{a}'_T([(V',s')]_{R'_{max}} \to [(W','t)]_{R'_{max}}) = \mathfrak{a}_T([(V,s)]_{R_{max}} \to [(W,t)]_{R_{max}}) \ \wedge$
  $([(W,t)]_{R_{max}},[(W,t')]_{R'_{max}}) \in R_{bsm}\ ]$

  Folgt analog. ✓

□

Folgendes Beispiel soll verdeutlichen, dass die Rückrichtung in Theorem 8.2.20 keine Gültigkeit besitzt.

Beispiel 8.2.21:

Es gilt für $p_0 := \langle h := 1; l := h\rangle$, dass $prbBsm^{(\sigma)}_{<p_0>}$ erfüllt ist aber p_0 nicht σ-sicher ist.

Beweis.
Wir wollen den Beweis lediglich Skizzenhaft führen.
Wir zeigen

- [$prbBsm^{(p_0)}_{<\sigma>}$ ist erfüllt]

  Man mache sich hierfür klar, dass $\mathcal{W}_{<\sigma L>}\langle\!|H,p_0,(h,l)|\!\rangle$ unabhängig von h und H stets von der Form wie in Abbildung 8.1 ist. ✓

- [p_0 ist nicht σ-sicher]

  Man mache sich hierfür klar, dass da $\langle h := 1; l := h\rangle$ nach einem Schritt stets in $\langle l = h\rangle$ übergeht und somit $\langle l = h\rangle$ ebenfalls σ-sicher sein müsste was offensichtlich nicht der Fall ist. ✓

□

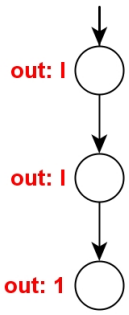


Abbildung 8.1: Skizzenhafte Darstellung von $\mathcal{W}_{<\sigma L>}\langle\langle H,p_0,(h,l)\rangle\rangle$ aus Beispiel 8.2.21

Betrachten wir bzgl. Beispiel 8.2.21 die Intention der Autoren bzgl. der σ-Sicherheit, dann lautet diese wie folgt:

> *"The intuition behind the construction is that a program is secure iff for any two states which differ only in the values of high variables, two configurations containing the program and each of the states, execute in such a way that their behaviour is indistinguishable (or "low-equivalent") from the attacker's observation of the low parts of the state and the probability with which they occur."[SS00]*

Betrachten wir nun $p_0 := \langle h := 1; l := h\rangle$ aus Beispiel 8.2.21 sowie zwei Zustände $(h,l),(h',l) \in St$ welche sich nur im H-Wert unterscheiden sowie die zugehörigen Konfigurationen $\langle\langle p_0,z\rangle\rangle,\langle\langle p_0,z'\rangle\rangle \in Conf_{ndet}$ dann sind die jeweiligen Programmausführungen von der Form

$$\langle\langle\langle h := 1; l := h\rangle,(h,l)\rangle\rangle \longrightarrow_1^\sigma \langle\langle\langle l := h\rangle,(1,l)\rangle\rangle \longrightarrow_1^\sigma \langle\langle\langle\rangle,(1,1)\rangle\rangle$$

$$\langle\langle\langle h := 1; l := h\rangle,(h',l)\rangle\rangle \longrightarrow_1^\sigma \langle\langle\langle l := h\rangle,(1,l)\rangle\rangle \longrightarrow_1^\sigma \langle\langle\langle\rangle,(1,1)\rangle\rangle$$

Es ist offensichtlich, dass diese für Beobachter, welche Sicht auf die L-Werte der Programmzustände der jeweiligen Konfigurationen sowie den Übergangswahrscheinlichkeiten haben, ununterscheidbar sind. Man würde von daher erwarten, dass p_0 die Eigenschaften der σ-Sicherheit erfüllt was hier jedoch nicht der Fall ist.
Natürlich kann man eine Äquivalenz in Theorem 8.2.20 erzwingen, indem man künstlich Anpassungen der $prbBsm^{(\sigma)}_{<p>}$ Noninterferenz-Spezifikation vornimmt. Da wir mit $prbBsm^{(\sigma)}_{<p>}$ der Intuition der Autoren bzgl. Noninterferenz-Sicherheit passend umgesetzt haben wollen wir im Rahmen dieser Arbeit hierauf verzichten.
Abschließend wollen wir zeigen, dass sofern ein Programm der parallelen *while*-Sprache sicher gegenüber dem Beobachter $\mathfrak{F}^{\ll prob,bisim\gg}$ ist, ebenfalls sicher gegenüber jedem Beobachter $\mathfrak{F}^{\ll prob,\mathfrak{s}\gg}$ für jede beliebige Semantik aus $\mathfrak{s}$ aus Abbildung 7.7.

Theorem 8.2.22: [Koinzidenz-Theorem]

Für jede Semantik $\mathfrak{s}$ aus 7.7 gilt

$$(\mathcal{W}_{<\sigma L,\mathfrak{p}>}, \mathcal{C}_{kan}, \mathfrak{F}^{\ll prob,\mathfrak{s}\gg}) \preceq (\mathcal{W}_{<\sigma L,\mathfrak{p}>}, \mathcal{C}_{kan}, \mathfrak{F}^{\ll prob,bisim\gg})$$

Beweis.
Sei $\mathfrak{s}$ eine beliebige Semantik aus Abbildung 7.7. Es gilt $\mathfrak{s} \preceq bisim$. Laut Proposition 7.4.12 gilt dann auch $\mathfrak{F}^{\ll prob,\mathfrak{s}\gg} \preceq \mathfrak{F}^{\ll prob,bisim\gg}$ und laut dem 1. Ordnungstheorem 6.1.21 ebenfalls die Behauptung.

□

KAPITEL 9

Zusammenfassung

Ziel dieser Arbeit war die formal Erfassung des Konzepts der Noninterferenz, sodass eine intuitive, generische Definition von Noninterferenz, in welche sich spezifische Konkretisierungen auf natürliche Weise einbetten lassen. Zu diesem Zwecke haben wir uns in Kapitel 2 auf die Struktur der Transitionssysteme als Basis der Systemmodellierung geeignet. Diese beschreiben eine abstrakte Modellierungsstruktur, in welche sich auf dem Zustandsbegriff basierende Systeme einbetten lassen. Diese haben wir im Laufe des Kapitel um Annoation, repräsentierend für System-Ausgaben, sowie um Programmfunktionen, um auch auf Modellebene eine formale Vorstellung von Programmen zu erhalten, erweitert. Anschließend haben wir in Kapitel 2 erarbeitet wie Programmausführungen innerhalb unserer Systemmodelle formal zu verstehen sind. Wir haben hierbei zwischen "formalisierten Programmausführungen" und "externen Programmausführungen" unterschieden. Formalisierte Programmausführungen dienten der formalen Repräsentation von Programmausführungen aus interner Sicht. Externe Programmausführungen waren entsprechend die formale Darstellung von Programmausführungen aus externer Sicht. Wir haben dabei festgestellt, dass die wesentlichen Unterschiede zwischen diesen darin bestehen, dass externe Programmausführungen in der Regel azyklische wahrgenommen werden und man aus externer Sicht keinen direkten lesenden Zugriff auf Systemzustände hat. Aufbauend auf den externen Programmausführungen haben wir Beobachter als funktionale Klassen definiert, welche Programmausführungen zusammen mit dem entsprechenden Systemmodell und zugehörigem Programm, was wir als theoretisches Maximum als nach außen ersichtlicher Information spezifiziert haben, auf spezifische Beobachtungen abbilden. Uns war es anschließend in Kapitel 5 möglich Noninterferenz, sowohl in Bezug auf Programme als auch auf Systemen, in einer allgemeingültigen Form zu definieren. Anschließend haben wir in Kapitel 6 Ordnungen auf Beobachter und Annotationen definiert und anschließend festgestellt, dass diese mit der Noninterfernez-Sicherheit verträglich sind. Wir sind dadurch beispielsweise in der Lage zu bestimmen, wann ein Beobachter stärker ist als ein andere und wissen zeitgleich, dass Noninterferenz-Sicherheit bzgl. des stärkeren Beobachters auch entsprechende Sicherheit bzgl. des schwächeren Beobachters impliziert.

Entsprechend haben wir festgestellt, dass Nointerferenz-Sicherheit bzgl. einer stärkeren Annotation unter günstigen Voraussetzungen auch Noninterferenz-Sicherheit bzgl. einer schwächeren Annotation impliziert. Da wir Annotationen mit Benutzersichten auf Modell-Ebene identifizieren können ist dies gleichbedeutend damit, dass Noninterferenz-Sicherheit bzgl. einer höheren Benutzersicht entsprechend Noninterferenz-Sicherheit aller tieferen Sichten impliziert. Die darauffolgenden Kapitel 7 und 8 beschäftigen sich mit der Einbettung fremder Arbeiten in unsere Theorie. In Kapitel 7 haben wir Beobachter durch einen Formalismus zur Einbettung von Zustandsgraph-Semantiken definiert. Zustandsgraph-Semantik sind dabei ein gut ausgearbeitetes Gebiet der theoretischen Informatik, welche nach Abschluss von Kapitel 7 strukturiert und simpel in unsere Theorie eingebettet werden konnten. Abschließend haben wir in Kapitel 8 zwei prominente Beispiele von Noninterferenz-Sicherheit der Literatur vorgestellt und bewiesen, dass diese sich mit Hilfe unserer erschaffenen Mittel formulieren lassen. Insbesondere konnten wir hierbei bei letzterem Beispiel feststellen, dass wir durch unseren Ansatz die Intuitionen der Autoren in Bezug auf Noninterferenz-Sicherheit besser umsetzen konnten.

Literaturverzeichnis

[GM82] J Goguen / J.Meseguer. Security Policies and Security Models (1982)

[E11] Claudia Eckert. IT-Sicherheit: Konzepte - Verfahren - Protokolle (2011)

[N27] John von Neumann. Die Axiomatisierung der Mengenlehre (1927)

[S09] Ralf Schindler. Logische Grundlagen der Mathematik (2009)

[M10] Markus Müller-Olm. Theorie der Programmierung (2010)

[G01] R.J. van Glabbeek. The Linear Time-Branching Time Spectrum I - The Semantics of Concrete, Sequential Processes (2001)

[SS00] Andrei Sabelfeld/ David Sands. Probabilistic Noninterference for Multi-threaded Programs (2000)

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Gronau, den 06. September 2013