technische universität
dortmund

fakultät für
informatik

TECHNICAL REPORTS IN COMPUTER SCIENCE
Technical University of Dortmund

Logik für Informatiker

(Logic for Computer Scientists)

Peter Padawitz
peter.padawitz@udo.edu

Number 867

15. Januar 2023

Technical University of Dortmund — Department of Computer Science
Otto-Hahn-Str. 14, 44227 Dortmund

Inhaltsverzeichnis

1 Vorwort

Dies sind überarbeitete Aufzeichnungen meines von 2012-2018 einmal jährlich angebotenen Logikkurses für Studierende des Bachelor-Studiengangs Informatik an der TU Dortmund.

free Der Kurs beginnt mit den mengentheoretischen Grundlagen abstrakter Datentypen. Mit Hilfe welcher mathematischer Konstruktionen werden Datentypen aufgebaut? Wie lassen sich ihre jeweils gewünschten Eigenschaften beweisen? Wie hängen die Beweismethoden mit der Art zusammen, wie sie aufgebaut wurden? Welche Datenstrukturen lassen sich ineinander überführen? Antworten auf die letzte Frage können beim Softwareentwurf von entscheidender Bedeutung sein und werden deshalb im Abschnitt 3.3 über *Isomorphien* für die wichtigsten Strukturschemata gegeben.

Da es sich hier um einen Einführungskurs handelt, beschränken wir uns in Kapitel 3 und 4 und den in Kapitel 5-9 vorgestellen (aufeinander aufbauenden) Logiken auf *einsortige* Varianten. Prinzipiell leidet die Ausdrucksfähigkeit darunter nicht, weil mit Hilfe von *Prädikaten* mehrsortige auf einsortige Datentypen zurückführbar sind. Allerdings mangelt es darauf basierenden – rein *relationalen* – Programmen oft an Effizienz, da deren Korrektheit nicht statisch, d.h. während ihrer Kompilation, überprüfbar ist. Das erlauben hingegen Sprachen mit Mehrsortigkeit und einem starken Typkonzept. Wir versuchen es deshalb mit einem Kompromiss: Um die Logiken in dieser Einführung nicht zu überfrachten, lassen wir *in ihren Formeln* vorkommende Funktionen und Relationen nur auf einer einzigen, unstrukturierten Datenmenge operieren. Auf der Metaebene, auf der die *Semantik* einer Logik definiert wird, verwenden wir jedoch Funktionen mit z.T. stark strukturierten Argument- bzw. Wertebereichen. So kann zumindest auf dieser Ebene der Umgang mit strukturierten Datenmengen geübt werden – und dabei gezeigt werden, wie man mit adäquaten Funktionen und Datentypen zur Implementierung einer Logik gelangt.

Der funktionsorientierte Zugang zu einzelnen Logiken legt es nahe, mit *Gleichungslogik*, also der Logik algebraischer Modelle, zu beginnen (Kapitel 5). Tatsächlich können die in den darauffolgenden Kapiteln 6-8 behandelten Logiken in weiten Teilen als "Spezialfälle" der Gleichungslogik betrachtet werden (siehe Abschnitt 5.8). So sind die jeweiligen Formeln nichts anderes als *Terme* einer Signatur aussagenlogischer, modaler bzw. prädikatenlogischer Operatoren. Die *Variablen* in den Termen entsprechen *atomaren* Formeln. Deren *Belegung* durch Elemente eines passenden semantischen Bereiches bestimmt den Wert einer Formel φ, der stets durch (eine) *Faltung*(sfunktion) berechnet werden kann. In der Aussagenlogik ist er 0 oder 1, in der Modal- und der Prädikatenlogik besteht er, intuitiv gesprochen, aus den *Zuständen*, die φ erfüllen.

Auch was die semantische *Äquivalenz* von Formeln und darauf aufbauend die Berechnung von *Normalformen* angeht, so entstammen grundlegende Begriffe und Verfahren der Gleichungslogik. Besonders prädikatenlogische Formeln müssen i.d.R. normalisiert werden, bevor effiziente Beweis- oder Lösungsalgorithmen auf sie angesetzt werden können. Außerdem lässt sich Gleichungslogik vollständig in die Prädikatenlogik integrieren (siehe Abschnitt 8.3).

Die hier in Aussagen- und Prädikatenlogik bevorzugten Normalformen sind *Gentzenformeln*

$$atom_1 \wedge \cdots \wedge atom_m \quad \Rightarrow \quad atom_{m+1} \vee \cdots \vee atom_{m+n}.$$

Der Fall $n = 1$ bzw. $m = 1$ liefert *Hornformeln* bzw. *Cohornformeln*, aus denen logische Programme bestehen, die wir in Kapitel 9 behandeln. Deren Semantik basiert auf *Fixpunktsätzen*, welche für die gesamte Informatik zentral sind, da sie die mathematische Begründung für alle *rekursiv definierten* Objekte, Funktionen und Relationen liefern (siehe Kapitel 4). Außerdem basieren die der Ausführung logischer Programme zugrundeliegenden Ableitungsregeln (Abschnitte 9.2 und 9.4) auf den Schnittkalkülen der Aussagen- bzw. Prädikatenlogik (Abschnitte 6.2 bzw. 8.2).

Auch *Modallogik* fügt sich in Aussagen- und Prädikatenlogik ein: Einerseits *erweitert* sie die Aussagenlogik um die modalen Operatoren $\Box$ und $\Diamond$. Im üblichen Fall *endlicher verzweigter Kripke-Strukturen* erlaubt die Bedeutung von $\Box$ und $\Diamond$ andererseits deren Übersetzung in aussagenlogische Konjunktionen bzw. Disjunktionen (Abschnitt 7.1). Betrachtet man die Komponenten einer Kripke-Struktur als Prädikate, dann lassen sich modallogische Formeln

auch in prädikatenlogische übersetzen, wobei □- und ◇-Formeln zu all- bzw. existenzquantifizierten Formeln werden (Abschnitt 8.4).

Gleichungslogik wird für die Modallogik relevant, wenn Zuständsäquivalenz präzisiert und gezeigt oder eine Kripke-Struktur *minimiert* werden soll (siehe Abschnitte 7.3 und 7.4).

In Kapitel 2 weist Fettdruck auf die informelle Einführung eines Begriffs hin, in späteren Kapiteln auf die Textstelle, an der er mathematisch definiert wird. Begriffe sind historisch gewachsen und daher nicht immer so präzise voneinander abgegrenzt wie Definitionen.

Ein mathematisches Symbol mit einer festgelegten Bedeutung ist an der Stelle, an der es zum ersten Mal benutzt wird, rot gefärbt. Manchmal weist Rotfärbung allerdings nur auf entscheidende Teile einer Formel hin.

Die eingestreuten Aufgaben lassen sich ohne besondere Tricks lösen, d.h. nur unter Verwendung hier behandelter Definitionen und Sätze. Auch wer sie nicht bearbeitet, sollte sie lesen und sich ihre jeweilige Aussage klarmachen.

Interne Links – einschließlich der Seitenzahlen im Index – sind an ihrer braunen Färbung, externe Links (z.B. zu Wikipedia) an ihrer magenta-Färbung erkennbar. Vor letzteren steht zusätzlich das Symbol ☞ .

Online-Version dieses Reports: https://fldit-www.cs.tu-dortmund.de/~peter/LogikPadR.pdf
Folienversion: https://fldit-www.cs.tu-dortmund.de/~peter/LogikPad.pdf

Aufzeichnungen zu verwandten Themen sowie zu zwei in Haskell programmierter interaktiver Präsentations- und Verifikationswerkzeuge (die u.a. in diesem Report zur Darstellung logischer Ableitungen verwendet werden) erreicht man durch Klicken auf entsprechende Titel an den Rändern des Pentagons von https://fldit-www.cs.tu-dortmund.de/~peter/SwingLinks.html.

Zu großem Dank verpflichtet bin ich gegenüber meinen Mitarbeitern Pascal Hof, Jos Kusiek, Jens Lechner und Hubert Wagner sowie den Tutoren Niklas Klocke, Felix Laarmann, Lukas Pfahler, Christoph Stahl, Richard Stewing und Jakob Vogt, vor allem hinsichtlich ihrer leidenschaftlichen Unterstützung bei der Vermittlung der Inhalte dieser Lehrveranstaltung.

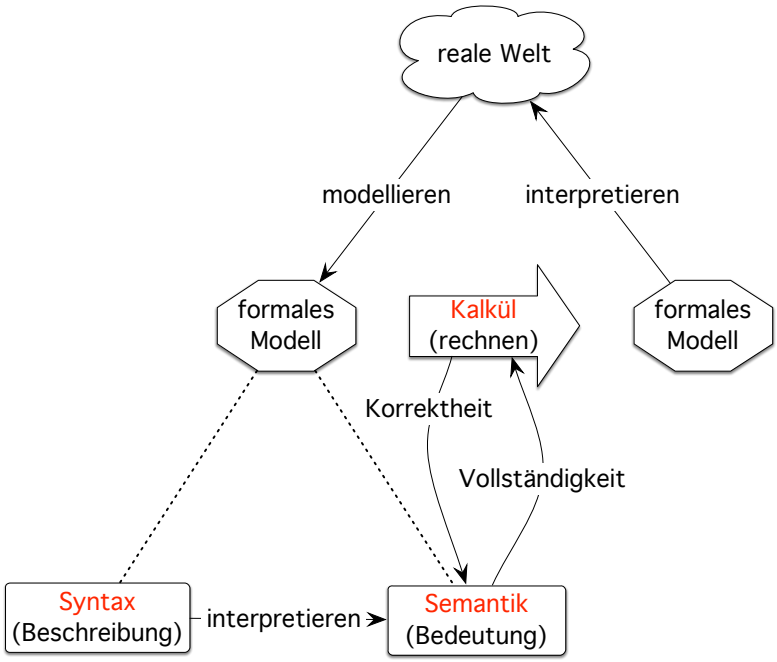
Peter Padawitz Dezember 2019

Neben kleinen Korrekturen wurde zwischenzeitlich das Hennessy-Milner-Theorem (Satz 7.6) verallgemeinert, das tatsächlich nicht nur für endlich verzweigte, sondern auch für *modal gesättigte* Kripke-Strukturen gilt (siehe Kapitel 7).

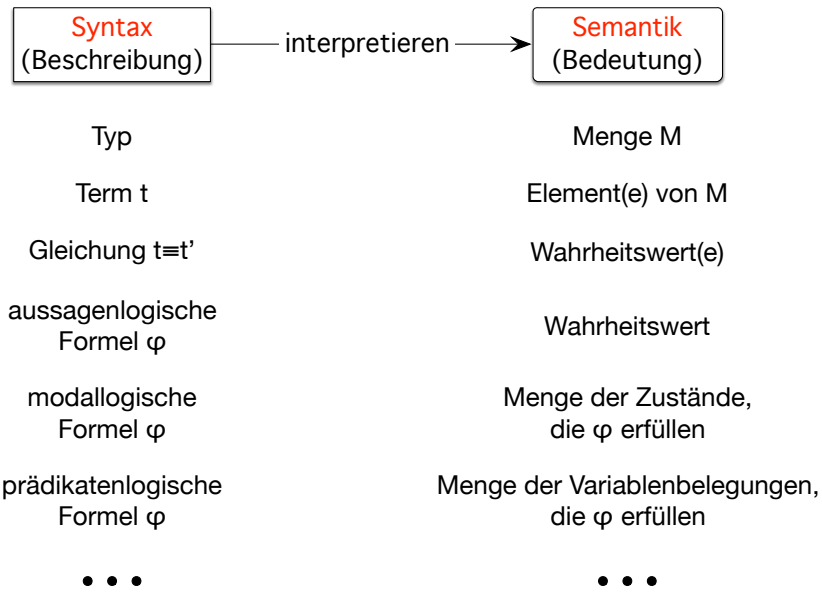
Außerdem wurde im Beweis von Satz 7.8 das in den Beweisen der ursprünglichen Fassung [13], Theorem 2.1, sowie des ähnlichen Theorems 5.15 von [28] verwendete Unendlichkeits-Argument deutlicher herausgearbeitet, das auf endlich verzweigte Kripke-Strukturen zugeschnitten ist. Ob es sich wie Satz 7.6 auch auf modal gesättigte Kripke-Strukturen übertragen lässt, scheint eine offene Frage zu sein.

Peter Padawitz April 2022

2 Grundbegriffe



Logik = Syntax+Semantik+Kalkül



Was wird wie interpretiert?

Syntax: Formeln der jeweiligen Logik

Semantik:

Bedeutung der Formeln und ihrer Komponenten in einem **semantischen Bereich** (*domain*) D, d.i. eine Klasse mathematischer Strukturen, in denen die Formeln als Wahrheitswerte oder von **Zuständen** abhängige Wahrheitswerte **interpretiert** werden können.

Umgekehrt werden Elemente des semantischen Bereichs durch Formeln **repräsentiert**.

Für eine Formel φ nennt man ein Element a von D ein **Modell von** φ, wenn a φ **erfüllt** (oder φ **in** a **gültig** ist), geschrieben: $a \models \varphi$. Was das bedeutet, hängt von der jeweiligen Logik ab. Die Menge aller Modelle von φ wird oft mit φ^D bezeichnet.

Auf D und dem jeweiligen Modellbegriff bauen die folgenden Definitionen bzw. Notationen auf:

φ ist **erfüllbar**, wenn φ ein Modell in D hat. φ ist **widersprüchlich**, wenn φ unerfüllbar ist. φ ist **allgemeingültig** oder **tautologisch**, wenn alle Elemente von D Modelle von φ sind. Eine Formel ψ **folgt aus** φ, geschrieben: $\varphi \models \psi$, wenn jedes Modell von φ auch Modell von ψ ist. Gilt $\varphi^D = \psi^D$, dann sind φ und ψ **äquivalent**.

Werden diese Definitionen für Formelmengen Φ anstelle einzelner Formeln verwendet, dann ist mit Φ stets die logische **Konjunktion** der Elemente von Φ gemeint.

Die Teilformeln φ und ψ einer **Implikation** $\varphi \Rightarrow \psi$ heißt **Prämisse** bzw. **Konklusion** der Implikation.

Im Gegensatz zum Implikationspfeil $\Rightarrow$, der zur Syntax vieler Logiken gehört, also in ihren jeweiligen Formeln vorkommen kann, beschreibt $\models$ stets eine semantische Beziehung zwischen Formeln, also niemals in diesen vorkommt.

Gilt $\varphi \Rightarrow \psi$ bzgl. der jeweiligen Semantik, dann sagt man auch:

- φ **impliziert** (*implies*) ψ,
- φ ist **hinreichend** (*sufficient*) für ψ,
- ψ ist **notwendig** (*necessary*) für φ.

Ein **synthetischer Kalkül** ist eine Menge K von (Schluss-)**Regeln** der Form

$$\frac{\Phi_1 \vdash \varphi_1, \quad \ldots, \quad \Phi_k \vdash \varphi_k}{\Phi \vdash \varphi}$$

Ein **analytischer Kalkül** ist eine Menge K von (Schluss-)**Regeln** der Form

$$\frac{\Phi \vdash \varphi}{\Phi_1 \vdash \varphi_1, \quad \ldots, \quad \Phi_k \vdash \varphi_k}$$

In beiden Fällen sind Φ und Φ_i, $1 \leq i \leq k$, Mengen von Formeln und φ und φ_i, $1 \leq i \leq k$, einzelne Formeln der jeweils zugrundeliegenden Logik sind. Rechts neben einer Regel stehen oft Anwendbarkeitsbedingungen.

Intuitiv beschreibt der – **Urteil** (☞ judgment) genannte – Ausdruck $\Phi \vdash \varphi$ die Gültigkeit von φ in jedem Kontext, in dem die Formeln von Φ gelten.

Die Urteile oberhalb bzw. unterhalb des Bruchstrichs nennt man **Antezedenten** bzw. **Sukzedenten** der Regel. Rechts neben dem Bruchstrich stehen manchmal Eigenschaften der Antezedenten, die diese haben müssen, damit die Regel anwendbar ist.

Der Sukzedent bzw. Antezedent einer Regel der Form

$$\frac{}{\Phi \vdash \varphi} \quad \text{bzw.} \quad \frac{\Phi \vdash \varphi}{}$$

eines synthetischen bzw. analytischen Kalküls heißt **Axiom**.

Ein Urteil a heißt K**-ableitbar**, wenn wiederholte Anwendungen von Regeln von K von Axiomen hin zu a (falls K synthetisch ist) bzw. von a hin zu Axiomen führen (falls K analytisch ist).

Die Folge der Ante- und Sukzedenten der jeweils angewendeten Regeln nennt man eine K**-Ableitung von** a.

Je nach Kalkül und Semantik der Formeln, aus denen seine Regeln gebildet sind, kann man aus der K-Ableitbarkeit von $\Phi \vdash \varphi$ verschiedene Schlüsse ziehen, z.B.

- φ folgt aus Φ,

- Φ ist widersprüchlich,
- φ repräsentiert eine **Lösung** von Φ,
- φ repräsentiert eine **Normalform** von Φ.

Beispiel Sei Φ die Menge aller Gleichungen der Form

$$t_0 * (t_1 + \cdots + t_n) \equiv t_0 * t_1 + \cdots + t_0 * t_n, \tag{1}$$
$$(t_1 + \cdots + t_n) * t_0 \equiv t_1 * t_0 + \cdots + t_n * t_0 \tag{2}$$

zwischen arithmetischen Ausdrücken und NF die Menge aller arithmetischen Ausdrücke, die aus $+$, $*$ und Konstanten bestehen, aber keinen Teilausdruck der Form der linken Seite von (1) oder (2) enthalten.

Mit der folgenden Regel lässt sich jeder arithmetische Ausdruck in einen semantisch äquivalenten Ausdruck von NF überführen:

$$\frac{\Phi \vdash c[L/x]}{\Phi \vdash c[R/x]} \quad L \equiv R \in \Phi \tag{3}$$

Hierbei bezeichnet der Ausdruck $c[u/x]$ die **Instanz** des Ausdrucks c, der aus c entsteht, wenn (die Variable) x durch den Ausdruck u ersetzt wird.

Da sich bei der Anwendung von (3) nur die Teilformel L von $c[L/x]$ verändert, nennt man diese den **Redex** und die korrespondierende Teilformel R von $c[R/x]$ das **Redukt** der Regelanwendung.

Die schrittweise Überführung eines arithmetisches Ausdrucks t in eine Normalform lässt sich als K-Ableitung beschreiben, wobei der analytische Kalkül K aus (3) und folgender Regel besteht:

$$\frac{\Phi \vdash u}{} \quad u \in NF \tag{4}$$

Für den Ausdruck $5 * (6 * ((11 * (x + y + z)) * 14 + (c + g + b) * 22) + 44 * (gg + hh))$ erhält man z.B. die K-Ableitung

$$\begin{array}{lll}
\Phi & \vdash & 5 * (6 * ((11 * (x + y + z)) * 14 + (c + g + b) * 22) + 44 * (gg + hh)) \\
\hline
\Phi & \vdash & 5 * (6 * ((11 * x + 11 * y + 11 * z) * 14 + (c + g + b) * 22) + 44 * (gg + hh)) \\
\hline
\vdots & & \\
\hline
\Phi & \vdash & 5 * 6 * 11 * x * 14 + 5 * 6 * 11 * y * 14 + 5 * 6 * 11 * z * 14 + \\
 & & 5 * 6 * c * 22 + 5 * 6 * g * 22 + 5 * 6 * b * 22 + 5 * 44 * gg + 5 * 44 * hh \\
\hline
\end{array}$$

Die komplette Ableitung steht ☞ hier. Die Ausdrücke sind dort als **Bäume** dargestellt. Teilbäume, die Redexe oder Redukte einer Regelanwendung darstellen, sind rot bzw. grün gefärbt. ❑

Ein Kalkül K ist **korrekt** (*sound*), wenn aus $\Phi \vdash_K \varphi$ stets $\Phi \models \varphi$ folgt.

Ein Kalkül K ist **vollständig** (*adequate*), wenn aus $\Phi \models \varphi$ stets $\Phi \vdash_K \varphi$ folgt.

Eine **Theorie** Th ist eine Menge von Formeln.

Th ist **vollständig** (*complete*), wenn jede Formel φ *oder* die Negation von φ zu Th gehört.

Th ist **konsistent** (widerspruchsfrei), wenn die Zugehörigkeit einer Formel zu Th ausschließt, dass auch die Negation von φ zu Th gehört. Die Konsistenz von Th wird oft durch die Angabe eines aus Th konstruierten "kanonischen" Modells von Th bewiesen.

Th ist **entscheidbar**, wenn es einen Algorithmus gibt, der, auf eine Formel angewendet, feststellt, ob sie zu Th gehört oder nicht, und stets terminiert.

Th ist **semi-entscheidbar**, wenn es einen Algorithmus gibt, der, auf eine Formel angewendet, feststellt, ob sie zu *Th* gehört und in diesem Fall terminiert. Andernfalls terminiert er möglicherweise nicht.

Eine **Präsentation** oder **Spezifikation** von *Th* ist eine – i.d.R. endliche – Menge von Formeln, aus denen alle Formeln von *Th* mit einem gegebenen Kalkül ableitbar sind.

2.1 Wo kommt mathematische Logik in der Informatik vor?

Überall! In Hard- und Software gleichermaßen:

- Schaltnetze und Schaltwerke
- Boolesche Ausdrücke (Darstellungen zweiwertiger Funktionen) kommen in jeder Programmier- oder Entwurfssprache vor.
- **Programmverifikation** besteht klassischerweise im Beweis von **Zusicherungen**, das sind Formeln, die Eigenschaften beschreiben, die Programmvariablen an bestimmten Positionen im Programm besitzen sollen.
- **Programmverifikation** kann auch die komplette Übersetzung der Programme in eine geeignete Logik bedeuten, in der ihre gewünschten Eigenschaften auf abstrakterer Ebene als bei der Zusicherungsmethode überprüft werden können.
- **Programmtransformation** dient der Effizienzsteigerung oder Anpassung an vorgegebene oder veränderte Datenstrukturen (☞ refactoring) oder gar zur Synthese von Programmen aus Formeln, die ihre gewünschten Eigenschaften beschreiben.
- **Logische Programme** lösen Formeln, d.h. (“das heißt”) berechnen Werte logischer Variablen, die die Formeln erfüllen.
- **Datenbankprogramme** operieren auf Relationen, kombinieren sie und stellen Anfragen (berechnen Teilrelationen). Im Gegensatz zu logischen Programmen sind die berechneten Werte hier i.d.R. selbst Relationen.
- **Automatisches Beweisen** befasst sich mit der Implementierung von Inferenzsystemen und ist weniger eine Anwendung der Logik in der Informatik als eine der Informatik in der Logik.

3 Mengen und Funktionen

3.1 Mengen und Relationen

Eine **Menge** ist eine ungeordnete Kombination von **Elementen** ohne Wiederholungen, d.h. ohne mehrfache Vorkommen einzelner Elemente.

Die Elemente einer Menge können selbst Mengen sein, allerdings muss zur Vermeidung logischer Widersprüche für jede Menge M und jedes Element e gelten, dass e *entweder* zu M gehört (in M enthalten ist; geschrieben: $e \in M$) oder *nicht* ($e \notin M$). Das schliesst u.a. Mengen aus, die sich selbst enthalten:

Würde man z.B. die Kombination K aller Mengen, die sich nicht selbst enthalten, als Menge betrachten und annehmen, dass K zu K gehört, dann gilt $K \notin K$ nach Definition von K. Nehmen wir dagegen an, dass K nicht zu K gehört, dann gilt, wieder nach Definition von K, $K \in K$. Um solche Widersprüche zu vermeiden, wird manche Menge von Mengen als **Klasse** eingeführt. Eine Klasse enthält sich *per definitionem* niemals selbst.

- $\emptyset$ ist die leere Menge.
- 1 bezeichnet die einelementige Menge $\{*\}$.
- 2 bezeichnet die zweielementige Menge $\{0, 1\}$.
- $\mathbb{N}$ bezeichnet die Menge $\{0, 1, 2, 3, \dots\}$ der natürlichen Zahlen.
- $\mathbb{N}_{>0}$ bezeichnet die Menge $\{1, 2, 3, \dots\}$ der positiven natürlichen Zahlen.
- $\mathbb{Z}$ und $\mathbb{R}$ bezeichnen die Mengen der ganzen bzw. reellen Zahlen.

Die obige Konvention, dass sich Mengen niemals selbst enthalten, erlaubt uns das *Overloading* von Bezeichnungen für Elemente einerseits und Mengen andererseits – wie z.B. 1 und 2.

Eine Menge B heißt **Teilmenge** (*subset*) einer Menge A oder **unäre Relation auf** A (geschrieben: $B \subseteq A$) und A **Obermenge** von B, wenn alle Elemente von B Elemente von A sind. $\emptyset$ ist also eine Teilmenge *jeder* Menge. $B \subseteq A$ ist eine **echte Teilmenge** von A, wenn A außer den Elementen von B weitere Elemente enthält. A und B sind **gleich** (geschrieben: $A = B$), wenn $A \subseteq B$ und $B \subseteq A$ gilt.

Durch *Komprehension* oder die Anwendung von *Mengenoperatoren* werden aus Mengen neue Mengen gebildet:

- **Komprehension**: Sei A eine Menge und φ eine Eigenschaft von Elementen von A.

  $$B =_{def} \{e \in A \mid e \text{ erfüllt } \varphi\} \quad \text{und} \quad e \in B \Leftrightarrow_{def} e \in A \text{ und } e \text{ erfüllt } \varphi$$

  definieren B als Menge aller Elemente von A, die φ erfüllen.

- **Vereinigung** von Mengen $A_1, \dots, A_n$:

  $$e \in A_1 \cup \dots \cup A_n \Leftrightarrow_{def} \text{es gibt } 1 \leq i \leq n \text{ mit } e \in A_i$$

- **Durchschnitt** von Mengen $A_1, \dots, A_n$:

  $$e \in A_1 \cap \dots \cap A_n \Leftrightarrow_{def} \text{für alle } 1 \leq i \leq n \text{ gilt } e \in A_i$$

- **Differenz** zweier Mengen A und B:

  $$e \in A \setminus B \Leftrightarrow_{def} e \in A \text{ und } e \notin B$$

- (Kartesisches) n**-stelliges Produkt** von Mengen $A_1, \dots, A_n$:

  $$A_1 \times \dots \times A_n =_{def} \{(a_1, \dots, a_n) \mid a_i \in A_i,\ 1 \leq i \leq n\}$$

  $(a_1, \dots, a_n)$ heißt n**-Tupel** mit den **Komponenten** $a_1, \dots, a_n$. Sind $A_1, \dots, A_n$ dieselben Mengen, dann schreibt man A_1^n anstelle von $A_1 \times \dots \times A_n$.

Das nullstellige Produkt ist die einelementige Menge 1 (s.o.).
$\{1, ..., n\}$ ist die **Indexmenge** des Produktes $A_1 \times \cdots \times A_n$.
Für alle $1 \leq i \leq n$ besteht A_i aus den möglichen Werten eines bestimmten Attributs der Elemente des Produkts, z.B. seiner Farbe. Man ersetzt dann den Index i häufig durch das zugehörige Attribut. Im Rahmen vieler Programmiersprachen werden derart attributierte Mengen als **Records** bezeichnet.

- **n-stellige Summe** oder **disjunkte Vereinigung** von Mengen $A_1, \ldots, A_n$:

$$A_1 + \cdots + A_n \;=_{def}\; A_1 \uplus \cdots \uplus A_n \;=_{def}\; \{(a, i) \mid a \in A_i,\ 1 \leq i \leq n\}$$

  Die zweite Komponente eines Paares (a, i) von $A_1 + \cdots + A_n$ informiert über die Herkunft von a.
  Die nullstellige Summe ist die leere Menge $\emptyset$.
  Gibt es $a \in A_1 \cup \cdots \cup A_n$ und $1 \leq i, j \leq n$ mit $i \neq j$ und $a \in A_i \cap A_j$, dann gibt es mindestens zwei Kopien von a in $A_1 + \cdots + A_n$, nämlich (a, i) und (a, j).
  Gibt es kein solches a, dann heißen $A_1 \ldots, A_n$ (paarweise) **disjunkt**, geschrieben: $A_1 \| \ldots \| A_n$.

- Menge der **Listen** oder **Wörter** über einer Menge A:

$$\begin{aligned} A^+ &=_{def} \bigcup_{n>0} A^n \\ A^* &=_{def} A^+ \cup \{\epsilon\}. \end{aligned}$$

  Wie die Definition von A^+ zeigt, lassen sich die oben definierten n-stelligen Mengenoperationen auch auf unendlich viele Mengen anwenden.
  Elemente von A^* werden oft ohne Klammern und Kommas geschrieben, z.B. abc anstatt (a, b, c). Bestehen sie aus mehreren Zeichen, setzt man Punkte dazwischen, schreibt also z.B. $ab \cdot bz \cdot de$ für (ab, bz, de). ϵ wird als **leeres Wort** bezeichnet.
  Listen bilden neben Bäumen und Graphen die wichtigsten Datenstrukturen fast aller Programmiersprachen. Sie werden dort u.a. zur Implementierung von Mengen benutzt. Da der Begriff der Menge sowohl von der Anordnung als auch der Anzahl der Vorkommen ihrer jeweiligen Elemente abstrahiert, gibt es für jede Menge M mit mindestens zwei Elementen mehrere Listen, die A repräsentieren.

- Die **Potenzmenge** $\mathcal{P}(A)$ einer Menge A ist die Menge aller Teilmengen von A:

$$e \in \mathcal{P}(A) \;\Leftrightarrow_{def}\; e \subseteq A$$

- $\mathcal{P}_{fin}(A)$ bezeichnet die Menge aller endlichen Teilmengen von A. $\mathcal{P}_{fin}(A)$ ist also eine Teilmenge von $\mathcal{P}(A)$.

Satz 3.1 Seien A eine Menge und B, C Teilmengen von A. Dann gilt:

$$B \subseteq C \quad \Leftrightarrow \quad B \cap (A \setminus C) = \emptyset.$$

Beweis. "$\Rightarrow$": Sei $B \subseteq C$ und $a \in B \cap (A \setminus C)$. Nach Definition der Mengendifferenz gehört a nicht zu C. Das geht nicht, da nach Voraussetzung $a \in B \subseteq C$ gilt. ↯ (Widerspruch zu $B \subseteq C$). Also sind B und $A \setminus C$ disjunkt.

"$\Leftarrow$": Seien B und $A \setminus C$ disjunkt und $a \in B$. Dann gehört a nicht zu $A \setminus C$. Aus $a \in B \subseteq A$ folgt deshalb $a \in A \setminus (A \setminus C) = A \cap C \subseteq C$ nach Definition der Mengendifferenz. Damit ist B eine Teilmenge von C. ❑

Satz 3.2 Für alle $C, D \subseteq A$ und $CS \subseteq \mathcal{P}(A)$,

$$\begin{aligned} C \subseteq D &\;\Leftrightarrow\; A \setminus D \subseteq A \setminus C, && (1)\\ A \setminus \bigcup CS &\;=\; \bigcap \{A \setminus C \mid C \in CS\}, && (2)\\ A \setminus \bigcap CS &\;=\; \bigcup \{A \setminus C \mid C \in CS\}, && (3)\\ (\forall\, C \in CS : C \subseteq D) &\;\Rightarrow\; \bigcup CS \subseteq D, && (4)\\ (\forall\, C \in CS : D \subseteq C) &\;\Rightarrow\; D \subseteq \bigcap CS. && (5) \end{aligned}$$

Fortan werden gelegentlich die logischen Symbole $\wedge, \vee, \forall$ und $\exists$ in ihrer üblichen umgangssprachlichen Bedeutung verwendet ("und", "oder", "für alle ... gilt" bzw. "es gibt ... mit"). Ihre formale Semantik wird in Kapitel 6 bzw. 8 definiert.

Eine n-stellige **Relation** R ist eine Teilmenge eines Produktes der Form $A_1 \times \cdots \times A_n$. $A_1, \ldots, A_n$ sind die **Komponenten** des Produkts.

Ist $n = 2$, dann wird R **binäre Relation** genannt und binäre Relation **auf** A_1, falls A_1 mit A_2 übereinstimmt.

$R^{-1} =_{def} \{(b,a) \mid (a,b) \in R\}$ heißt **Inverse** von R. R ist eine **Relation auf** A_1, falls A_1 mit A_2 übereinstimmt.

Wegen $A^0 = 1$ gibt es genau zwei nullstellige Relationen, nämlich 1 und $\emptyset$.

Eine Teilmenge Z von $\mathcal{P}(A)$ heißt **Partition** oder **Zerlegung** von A, wenn die leere Menge nicht zu Z gehört, je zwei verschiedene Elemente von Z disjunkt sind und Z ganz A abdeckt, d.h. A mit $\bigcup Z$ übereinstimmt.

Partitionen liefern ein mathematisches Mittel zur **Abstraktion**: Elemente von A, die zur selben Teilmenge von Z gehören, werden als gleich angesehen. Z entspricht daher einer **Äquivalenzrelation** auf A (siehe Kapitel 5).

3.2 Funktionen

Seien $n \in \mathbb{N}$ und A und B Mengen. Eine **n-stellige Funktion** oder **Abbildung** (*function, map*) $f : A \to B$ **von** A **nach** B ordnet jedem – **Argument** oder **Stelle von** f genannten – Element $a \in A$ genau ein Element $f(a) \in B$ zu, das **Bild von** a **unter** f genannt wird. Man sagt auch: f bildet a auf $f(a)$ ab.

Für alle $C \subseteq A$ nennt man die Menge $f(C) = \{f(c) \mid c \in C\}$ ebenfalls das **Bild von** C **unter** f.

Sei $b \in B$ und $D \subseteq B$. $f^{-1}(b) = \{a \in A \mid f(a) = b\}$ und $f^{-1}(D) = \{a \in A \mid f(a) \in D\}$ heißen **Urbilder** (*pre-image*) **von** b bzw. D **unter** f.

- $type(f) = A \to B$ ist der **Typ** von f.
- $arity(f) = n$ ist die **Stelligkeit** von f, falls A ein n-stelliges Produkt ist,
- $dom(f) = A$ ist der **Definitionsbereich** (*domain*) von f,
- $ran(f) = B$ ist der **Wertebereich** (*range*) von f,
- $img(f) = f(A) = \{f(a) \mid a \in A\}$ ist das **Bild** (*image*) von f,
- $ker(f) = \{(a,a') \in A^2 \mid f(a) = f(a')\}$ ist der **Kern** (☞ *kernel*) von f,
- Die Relation $graph(f) = \{(a, f(a)) \mid a \in A\} \subseteq A \times B$ heißt **Graph** von f.

Nullstellige Funktionen haben den Definitionsbereich 1 und heißen auch **Konstanten**.

$f : A \to B$ ist eine **Endofunktion**, wenn $A = B$ gilt.

$f : A \to B$ ist **endlich**, wenn A endlich ist.

$f : A \to B$ ist **surjektiv**, falls $img(f) = B$.

Die Relation $\Delta_A =_{def} \{(a,a) \mid a \in A\}$ heißt **Diagonale von** A^2.

$f : A \to B$ ist **injektiv**, falls $ker(f) = \Delta_A$.

Aufgabe Zeigen Sie, dass f genau dann injektiv ist, wenn je zwei unterschiedliche Elemente von A verschiedene Bilder unter f haben.

Aufgabe Zeigen Sie, dass für alle $C \subseteq A$ und $D \subseteq B$ folgende Implikationen gelten:

$$\begin{array}{rcl} f(C) \subseteq D & \Rightarrow & C \subseteq f^{-1}(D), \\ C \subseteq f^{-1}(D) & \Rightarrow & f(C) \subseteq D, \\ f \text{ surjektiv und } f^{-1}(D) \subseteq C & \Rightarrow & D \subseteq f(C), \\ f \text{ injektiv und } D \subseteq f(C) & \Rightarrow & f^{-1}(D) \subseteq C. \end{array}$$

Funktionen können auf mehrere Weisen definiert werden. Wir schreiben z.B.

$$\begin{array}{rcl} f : \mathbb{N} \times \mathbb{N} & \to & \mathbb{R} \\ (m,n) & \mapsto & m * n/2 \end{array} \tag{1}$$

oder

$$\text{Für alle } m,n \in \mathbb{N},\ f(m,n) =_{def} m * n/2. \tag{2}$$

oder

$$f =_{def} \lambda(m,n).(m * n/2). \tag{3}$$

(3) definiert f durch einen **λ-Ausdruck**. Sie macht deutlich, dass jede Funktion $f : A \to B$ Element einer Menge ist, nämlich der Menge $B^A =_{def} (A \to B)$ aller Funktionen von A nach B. Demnach bedeutet $f \in B^A$ dasselbe wie $f : A \to B$.

In klassischer Algebra und Analysis taucht λ bei der Darstellung von Funktionen nicht auf, wenn Symbole wie x, y, z konventionsgemäß als Variablen betrachtet und daher z.B. für die Polynomfunktion $\lambda x.2 * x^3 + 55 * x^2 + 33 : \mathbb{R} \to \mathbb{R}$ einfach nur $2 * x^3 + 55 * x^2 + 33$ oder sogar nur $2x^3 + 55x^2 + 33$ geschrieben wird.

Da in allgemeiner Logik oder Algebra Symbole wie $x, y, z, *$ unterschiedlich interpretiert werden, können solche Kurzschreibweisen hier zu Missverständnissen führen und sollten deshalb vermieden werden.

Das **Update** von $f : A \to B$ an Stellen $a_1, \ldots, a_n \in A$ durch Werte $b_1, \ldots, b_n \in B$ ist eine neue Funktion, die wie folgt definiert ist:

$$\begin{array}{rcll} f[b_1/a_1, \ldots, b_n/a_n] : A & \to & B & \text{“}f \text{ mit } b_i \text{ für } a_i\text{”} \\ a & \mapsto & \begin{cases} b_i & \text{falls } a = a_i \text{ für ein } 1 \le i \le n \\ f(a) & \text{sonst} \end{cases} & \end{array}$$

Für jede Menge A liefert jede binäre Relation $R \subseteq B \times B$ eine Relation $R' \subseteq B^A \times B^A$:

Für alle $f, g : A \to B$,

$$(f,g) \in R' \quad \Leftrightarrow_{def} \quad \text{für alle } a \in A \text{ gilt } (f(a), g(a)) \in R.$$

Die (sequentielle) **Komposition** $g \circ f$ zweier Funktionen $f : A \to B$ und $g : B \to C$ ist eine Funktion von A nach C:

$$\begin{array}{rcl} g \circ f : & A & \to & C \\ & a & \mapsto & g(f(a)) \end{array}$$

Offenbar ist $\circ$ **assoziativ**, d.h. für alle $f : A \to B$, $g : B \to C$ und $h : C \to D$ gilt

$$h \circ (g \circ f) = (h \circ g) \circ f.$$

Die Klammern werden deshalb oft weggelassen.

Aufgabe Zeigen Sie, dass f genau dann injektiv ist, wenn je zwei Funktionen $g, h : A' \to A$ mit $f \circ g = f \circ h$ übereinstimmen.

Aufgabe Zeigen Sie, dass f genau dann surjektiv ist, wenn je zwei Funktionen $g, h : B \to B'$ mit $g \circ f = h \circ f$ übereinstimmen.

Zu jeder Menge A gibt es die **Identität** $id_A : A \to A$ **auf** A, die jedem Argument dieses selbst zuordnet: Für alle $a \in A$, $id_A(a) =_{def} a$.

Zu jeder Teilmenge B von A gibt es die **Inklusion** $inc_B : B \to A$, die ebenfalls jedem Argument dieses selbst zuordnet: Für alle $b \in B$, $inc_B(b) =_{def} b$.

Eine Funktion $g : B \to A$ heißt **Inverse** (Umkehrfunktion) von $f : A \to B$, falls gilt:

$$g \circ f = id_A, \quad f \circ g = id_B.$$

Die Elemente eines Produktes $A_1 \times \cdots \times A_n$ werden mit **Projektionen**

$$\begin{array}{rcll} \pi_i : A_1 \times \cdots \times A_n & \to & A_i & 1 \leq i \leq n \\ (a_1, \ldots, a_n) & \mapsto & a_i & \end{array}$$

auf ihre n Komponentenmengen $A_1, \ldots, A_n$ abgebildet.

Die Elemente von n Mengen $A_1, \ldots, A_n$ werden mit **Injektionen**

$$\begin{array}{rcll} \iota_i : A_i & \to & A_1 + \cdots + A_n & 1 \leq i \leq n \\ a & \mapsto & (a, i) & \end{array}$$

in die Summe $A_1 + \cdots + A_n$ eingebettet.

Die **Summenextension**

$$[f_1, \ldots, f_n] : A_1 + \cdots + A_n \to B$$

von n Funktionen $f_1 : A_1 \to B, \ldots, f_n : A_n \to B$ ist folgendermaßen definiert:

Für alle $(a, i) \in A_1 + \cdots + A_n$,

$$[f_1, \ldots, f_n](a, i) \quad =_{def} \quad f_i(a).$$

Die **Produktextension**

$$\langle g_1, \ldots, g_n \rangle : B \to A_1 \times \cdots \times A_n$$

von n Funktionen $g_1 : B \to A_1, \ldots, g_n : B \to A_n$ ist folgendermaßen definiert:

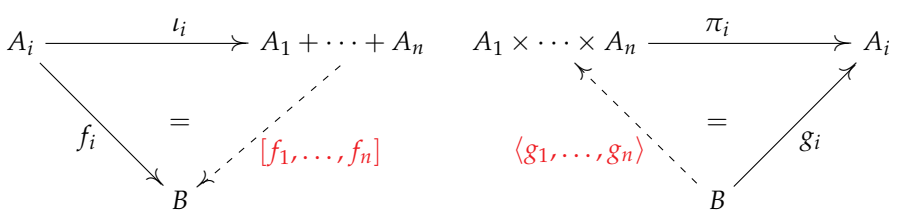
Für alle $b \in B$,

$$\langle g_1, \ldots, g_n \rangle(b) \quad =_{def} \quad (g_1(b), \ldots, g_n(b)).$$

$[f_1, \ldots, f_n]$ und $\langle g_1, \ldots, g_n \rangle$ sind die eindeutigen (!) Funktionen, die für alle $1 \leq i \leq n$ folgende Gleichungen erfüllen:

$$[f_1, \ldots, f_n] \circ \iota_i = f_i \quad \text{bzw.} \quad \pi_i \circ \langle g_1, \ldots, g_n \rangle = g_i.$$

Funktionsgleichungen lassen sich als **kommutative Funktionsdiagramme** darstellen:



Ein gestrichelter Pfeil deutet an, dass die Funktion, mit der er markiert ist, eindeutig ist.

Ebenso sind die **Summe**

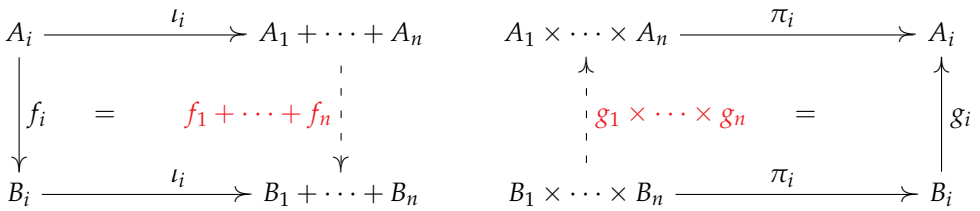
$$f_1 + \cdots + f_n : A_1 + \cdots + A_n \to B_1 + \cdots + B_n$$

von $f_i : A_i \to B_i$, $1 \leq i \leq n$, und das **Produkt**

$$g_1 \times \cdots \times g_n : A_1 \times \cdots \times A_n \to B_1 \times \cdots \times B_n$$

von $g_i : B_i \to A_i$, $1 \leq i \leq n$, als die eindeutigen Funktionen definiert, die für alle $1 \leq i \leq n$ folgende Gleichungen erfüllen:

$$f_1 + \cdots + f_n = [\iota_1 \circ f_1, \ldots, \iota_n \circ f_n] \quad \text{bzw.} \quad g_1 \times \cdots \times g_n = \langle g_1 \circ \pi_1, \ldots, g_n \circ \pi_n \rangle.$$



Im Fall $g_1 = \cdots = g_n = g$ schreibt man auch g^n an Stelle von $g_1 \times \cdots \times g_n$.

Vorsicht: Die Potenzschreibweise wird auch für die **Iteration** einer Endofunktion verwendet, d.h. $f^n : A \to A$ bezeichnet auch die n-fache Komposition $f \circ \cdots \circ f$, wobei $f^0 =_{def} id_A$ (s.o.).

Ähnlich wie die Mengenoperatoren Produkt und Summe oben auf Funktionen definiert werden, lässt sich auch die Potenzmengenbildung zu einem Funktionsoperator erweitern:

Für alle Funktionen $f : A \to B$ bildet die Funktion $\mathcal{P}(f) : \mathcal{P}(A) \to \mathcal{P}(B)$ ein Teilmenge C von A auf das Bild von C unter f ab (s.o.).

3.3 Isomorphien

$f : A \to B$ ist **bijektiv** oder eine **Bijektion**, falls f eine Inverse g hat. In diesem Fall ist g eindeutig und wird deshalb mit f^{-1} bezeichnet.

Beweis. Seien g und h Inverse von f. Dann gilt:

$$h = h \circ id_B = h \circ (f \circ g) = (h \circ f) \circ g = id_A \circ g = g. \qquad \text{❑}$$

Aufgabe Zeigen Sie, dass die Komposition zweier Bijektionen bijektiv ist.

Aufgabe Zeigen Sie, dass eine Funktion genau dann bijektiv ist, wenn sie injektiv und surjektiv ist.

Aufgabe Zeigen Sie, dass für Funktionen zwischen endlichen Mengen die Begriffe *injektiv*, *surjektiv* und *bijektiv* zusammenfallen.

Zwei Mengen A und B sind **isomorph**, geschrieben: $A \cong B$, wenn es eine Bijektion von A nach B (oder B nach A) gibt.

Da die Identitität $id_A : A \to A$ eine Bijektion ist, gilt trivialerweise:

$$A \cong A. \tag{0}$$

Produkt-, Summen- und Funktionsmengenbildung sind unter Isomorphie abgeschlossen, d.h. sind

$$A \cong C \ \wedge \ B \cong D \quad \Rightarrow \quad A + B \cong C + D, \tag{1}$$

$$A \cong C \ \wedge \ B \cong D \quad \Rightarrow \quad A \times B \cong C \times D, \tag{2}$$

$$A \cong C \ \wedge \ B \cong D \quad \Rightarrow \quad B^A \cong D^C. \tag{3}$$

Sind A und B Mengen mit Operationen, dann verlangt Isomorphie zusätzlich, dass die Bijektion mit den Operationen von A und B verträglich ist. Das wird in Kapitel 5 näher ausgeführt.

In diesem Fall folgt aus $A \cong B$, dass A und B **elementar äquivalent** sind, d.h. dieselben – in der jeweils zugrundeliegenden Logik formulierbaren – Eigenschaften haben. Die Umkehrung: Elementare Äquivalenz impliziert Isomorphie, gilt jedoch nicht immer.

Mathematiker betrachten isomorphe Mengen A und B oft als gleich. Informatiker stoßen hingegen manchmal auf große Unterschiede, wenn A und B (als Formate, Datentypen, Klassen o.a.) implementieren und den Aufwand eines bestimmten auf A operierenden Algorithmus mit dem seines Pendants auf B vergleichen. Umgekehrt erleichtert die Kenntnis von Isomorphien die Auswahl geeigneter Datentypen für bestimmte Algorithmen.

Wichtige Isomorphien (jeweils mit der zugehörigen Bijektion und deren Inverser)

- Seien A und B disjunkt. Dann gilt: $A + B \cong A \cup B$. Die zugehörige Bijektion $f : A + B \to A \cup B$ und ihre Inverse $f^{-1} : A \cup B \to A + B$ lauten wie folgt: Für alle $a \in A$ und $b \in B$, $f(a,1) = a$, $f(b,2) = b$, $f^{-1}(a) = (a,1)$ und $f^{-1}(b) = (b,2)$.

- Kommutativität von $+$ und $\times$: $A \times B \cong B \times A$ und $A + B \cong B + A$. (4)
  Die zugehörige Bijektion $f : A \times B \to B \times A$ bzw. $f : A + B \to B + A$ lautet wie folgt: Für alle $a \in A$ und $b \in B$, $f(a,b) = (b,a)$ bzw. $f(a,1) = (a,2)$ und $f(b,2) = (b,1)$.

- $\mathcal{P}(A) \cong 2^A$. (5)
  Die Funktionen $\chi : \mathcal{P}(A) \to 2^A$ und $\chi^{-1} : 2^A \to \mathcal{P}(A)$ sind wie folgt definiert:

  Für alle $B \subseteq A$ und $a \in A$, $\chi(B)(a) = \begin{cases} 1 & \text{falls } a \in B, \\ 0 & \text{sonst.} \end{cases}$

  $\chi(B)$ heißt **charakteristische** oder ☞ **Indikatorfunktion** von B.
  Für alle $g \in 2^A$, $\chi^{-1}(g) = \{a \in A \mid g(a) = 1\}$.

  Ist A ein binäres Produkt, z.B. $A = A_1 \times A_2$, dann wird $\chi(B)$ auch als **Boolesche Matrix** oder **Adjazenzmatrix** bezeichnet.
  χ ist bijektiv.
  Beweis. Nach Definition von Bijektivität und Funktionsgleichheit ist zu zeigen:
  (i) Für alle $B \subseteq A$ gilt $\chi^{-1}(\chi(B)) = B$.
  (ii) Für alle $g : A \to 2$ und $a \in A$ gilt $\chi(\chi^{-1}(g))(a) = g(a)$.
  Beweis von (i). Sei $B \subseteq A$. Dann gilt nach Definition von χ^{-1} bzw. χ:

  $$\chi^{-1}(\chi(B)) = \{a \in A \mid \chi(B)(a) = 1\} = \{a \in A \mid a \in B\} = B.$$

  Beweis von (ii). Sei $g : A \to 2$ und $a \in A$. Dann gilt nach Definition von χ bzw. χ^{-1}:

  $$\chi(\chi^{-1}(g))(a) = \left\{ \begin{array}{ll} 1 & \text{falls } a \in \chi^{-1}(g) \\ 0 & \text{sonst} \end{array} \right\} = \left\{ \begin{array}{ll} 1 & \text{falls } g(a) = 1 \\ 0 & \text{sonst} \end{array} \right\} = g(a). \quad \square$$

- $C^{A \times B} \cong (C^B)^A$. (6)
  Die Funktionen

  $$curry : C^{A \times B} \to C^{B^A} \quad \text{und} \quad uncurry : (C^B)^A \to C^{A \times B}$$

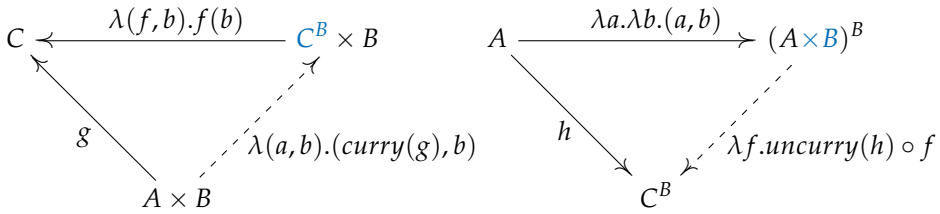
  sind folgt definiert:
  Für alle $g : A \times B \to C$, $a \in A$, $b \in B$ und $h : A \to C^B$,

  $$curry(g)(a)(b) = g(a,b) \quad \text{und} \quad uncurry(h)(a,b) = h(a)(b).$$

  $curry(g) : A \to C^B$ heißt **kaskadierte** oder ☞ **curryfizierte** Version von g.
  $curry$ ist bijektiv.

  Wie die oben definierten Funktionssummen- und produkte, so sind auch $curry$ und $uncurry$ die eindeutigen Funktionen derart, dass bestimmte Funktionsgleichungen gelten oder die entsprechenden Diagramme

kommutieren. Hier lauten letztere wie folgt:



- $A \cong B \;\Rightarrow\; \mathcal{P}(A) \cong \mathcal{P}(B)$. (7)

- $\mathcal{P}(A \times B) \cong \mathcal{P}(B)^A$. (8)
  Die Funktionen
  $$mkfun : \mathcal{P}(A \times B) \to \mathcal{P}(B)^A \text{ und } mkrel : \mathcal{P}(B)^A \to \mathcal{P}(A \times B)$$
  sind wie folgt definiert: Für alle $R \subseteq A \times B$, $a \in A$ und $g : A \to \mathcal{P}(B)$,
  $$mkfun(R)(a) = \{b \in B \mid (a,b) \in R\} \text{ und } mkrel(g) = \{(a,b) \in A \times B \mid b \in g(a)\}.$$
  $mkfun(R)$ wird auch **nichtdeterministische** oder **mehrwertige Funktion** oder **Adjazenzliste** genannt.
  mkfun ist bijektiv.
  Beweis. $\mathcal{P}(A \times B) \overset{(5)}{\cong} 2^{A \times B} \overset{(6)}{\cong} (2^B)^A \overset{(0),(3),(5)}{\cong} \mathcal{P}(B)^A$. ❑

Aufgabe Folgern Sie aus den obigen Isomorphien die folgende:
$$\mathcal{P}(A)^B \cong \mathcal{P}(B)^A.$$

Aufgabe Zeigen Sie die Assoziativität von Summe und Produkt:
$$A + (B + C) \cong (A + B) + C, \qquad A \times (B \times C) \cong (A \times B) \times C.$$

Aufgabe Zeigen Sie folgende Distributivitätsgesetze:
$$A \times (B_1 + \cdots + B_n) \cong (A \times B_1) + \cdots + (A \times B_n) \quad (9)$$
$$B^{A_1 + \cdots + A_n} \cong B^{A_1} \times \ldots \times B^{A_n} \quad (10)$$
$$(B_1 \times \cdots \times B_n)^A \cong B_1^A \times \ldots \times B_n^A \quad (11)$$

Aufgabe Zeigen Sie für alle $f : A \to C^B$ und $g : C \to D$:
$$g \circ uncurry(f) = uncurry((\lambda h.g \circ h) \circ f). \quad (12)$$

Aufgabe Zeigen Sie für alle $R \subseteq A \times B$: $curry(\chi(R)) = \chi \circ mkfun(R)$.

Aufgabe Zeigen Sie für alle $f : A \to B$: $mkfun(graph(f)) = single \circ f$, wobei für alle $b \in B$, $single(b) =_{def} \{b\}$.

Offenbar sind viele zusammengesetzte Mengen zu Funktionsmengen isomorph. Eine endliche Funktion kann durch ein **Feld** oder ☞ **Array** oder eine Liste (s.u.) der Elemente ihres Graphen implementiert werden.

Weitere als Funktionen darstellbare Datenstrukturen:

- **Multimengen** (*bags*) sind wie Mengen ungeordnete Kombinationen von Elementen. Im Gegensatz zu Mengen können sie jedoch Elemente mehrfach enthalten:
  Eine **Multimenge über** einer Menge A, d.h. mit Elementen von A, lässt sich daher als Funktion $f : A \to \mathbb{N}$ darstellen: Für alle $a \in A$ gibt $f(a)$ als Anzahl der Vorkommen von a in f an. Die Menge $\mathcal{B}(A)$ aller Multimengen über A ist daher definiert als die Funktionsmenge $\mathbb{N}^A$.

- Die Menge **Ströme** oder **unendlichen Folgen** von Elementen einer Menge A, kurz: **Ströme über** A, ist umgekehrt definiert als die Funktionsmenge $A^{\mathbb{N}}$: Für alle $s \in A^{\mathbb{N}}$ und $n \in \mathbb{N}$ liefert $s(n)$ das n-te Element von s.

Ist A eine endliche Menge, dann ist die **Mächtigkeit** oder ☞ Kardinalität $|A|$ von A durch die Anzahl der Elemente von A gegeben. Die Mächtigkeit endlicher Produkte, Summen bzw. Funktionsmengen Mengen ergibt sich wie folgt aus den Kardinalitäten der jeweiligen Komponentenmengen:

$$|A \times B| = |A| * |B|, \quad |A + B| = |A| + |B|, \quad |A^B| = |A|^{|B|}.$$

Hier sieht man, wo die Bezeichnungen dieser Mengenoperatoren herkommen.

Isomorphe Mengen nennt man auch **gleichmächtig**.

Die Mächtigkeit oder "Größe" einer Menge A wird durch eine **Kardinalzahl** beschrieben. Die kleinste unendliche Kardinalzahl wird mit ω oder $\aleph_0$ (*aleph*, hebräischer Buchstabe) bezeichnet.

Demgegenüber gibt eine **Ordinalzahl** die "Position" von A innerhalb der totalen Ordnung aller Ordinalzahlen an. Ist A endlich, dann stimmen die Kardinal- und die Ordinalzahl von A miteinander überein. Andernfalls lassen sich A mehrere Ordinalzahlen zuordnen. Die Kardinalität von A ist die kleinste davon.

Lässt sich A z.B. die Ordinalzahl $\omega + 1$ zuordnen, dann hat A dennoch die Kardinalität ω.

Mengen mit einer Kardinalität, die kleiner oder gleich ω ist, heißen **abzählbar**. Nicht abzählbare Mengen werden auch **überabzählbar** genannt. Eine abzählbare Menge A heißt **aufzählbar**, wenn es einen – im Falle von $|A| \geq \omega$ möglicherweise nichtterminierenden – Algorithmus gibt, der die Abzählung der Elemente von A durchführt.

A ist genau dann abzählbar, wenn es eine injektive Funktion $f : A \to \mathbb{N}$ oder eine surjektive Funktion $g : \mathbb{N} \to A$ gibt. Die beiden Bedingungen sind äquivalent zueinander.

Z.B. ist $A = \mathbb{N} \times \mathbb{N}$ abzählbar, weil die folgende Funktion $f : A \to \mathbb{N}$ injektiv ist: Für alle $m, n \in \mathbb{N}$,

$$f(m, n) =_{def} (m + n) * (m + n + 1)/2 + m$$

(siehe [39], §3.2.2).

f basiert auf ☞ Cantors erstem Diagonalargument. An der graphischen Darstellung erkennt man, dass sich Cantors erstes Diagonalargument wie folgt verallgemeinern lässt:

Sind zwei Mengen A und B abzählbar, dann ist auch $A \times B$ abzählbar.

☞ Cantors zweites Diagonalargument dient dem Nachweis der Überabzählbarkeit einer Menge A. Für $A = 2^{\mathbb{N}}$ lautet es wie folgt: Wäre A abzählbar, dann gäbe es eine Bijektion $f : A \to \mathbb{N}$. Sei $h \in 2^{\mathbb{N}}$ wie folgt definiert: Für alle $n \in \mathbb{N}$, $h(n) = 1 - f^{-1}(n)(n)$. Daraus folgt

$$h(f(h)) = 1 - f^{-1}(f(h))(f(h)) = 1 - h(f(h)). \lightning$$

Wieder erkennt man an der graphischen Darstellung, dass sich auch Cantors zweites Diagonalargument wie folgt verallgemeinern lässt:

Ist eine Menge A abzählbar und hat eine Menge B mindestens zwei Elemente, dann ist B^A überabzählbar.

Aufgabe Zeigen Sie unter Verwendung von Cantors erstem Diagonalargument, dass A^* abzählbar ist, falls A abzählbar ist.

3.4 Highlights

$$
\begin{array}{rcl}
B \in \mathcal{P}(A) & \Leftrightarrow_{def} & B \subseteq A \\
e \in A_1 \cup \cdots \cup A_n & \Leftrightarrow_{def} & \exists\, 1 \leq i \leq n : e \in A_i \\
e \in A_1 \cap \cdots \cap A_n & \Leftrightarrow_{def} & \forall\, 1 \leq i \leq n : e \in A_i \\
e \in A \setminus B & \Leftrightarrow_{def} & e \in A \wedge e \notin B \\
A_1 + \cdots + A_n & =_{def} & \{(a,i) \mid a \in A_i,\ 1 \leq i \leq n\} \\
A_1 \times \ldots \times A_n & =_{def} & \{(a_1,\ldots,a_n) \mid a_i \in A_i,\ 1 \leq i \leq n\} \\
B^A & =_{def} & \text{Menge aller Funktionen von } A \text{ nach } B \\
\Delta_A & =_{def} & \{(a,a) \mid a \in A\} \\
A \subseteq B & \Leftrightarrow & C \setminus B \subseteq C \setminus A \\
A \cap (B \cup C) & = & (A \cap B) \cup (A \cap C) \\
A \cup (B \cap C) & = & (A \cup B) \cap (A \cup C) \\
A \setminus (B \cup C) & = & (A \setminus B) \cap (A \setminus C) \\
A \setminus (B \cap C) & = & (A \setminus B) \cup (A \setminus C) \\
graph(f : A \to B) & =_{def} & \{(a, f(a)) \in A^2 \mid a \in A\} \\
img(f : A \to B) & =_{def} & \{f(a) \mid a \in A\} \\
ker(f : A \to B) & =_{def} & \{(a,a') \in A^2 \mid f(a) = f(a')\} \\
f : A \to B \text{ injektiv} & \Leftrightarrow_{def} & ker(f) = \Delta_A \\
f : A \to B \text{ surjektiv} & \Leftrightarrow_{def} & img(f) = B
\end{array}
$$

$$
\begin{array}{rcll}
A & \cong & A & \\
A \cong B & \Rightarrow & B \cong A & \\
A \cong B \ \wedge\ B \cong C & \Rightarrow & A \cong C & \\
A + B & \cong & B + A & \\
A \times B & \cong & B \times A & \\
A \times (B + C) & \cong & (A \times B) + (A \times C) & \\
C^{A+B} & \cong & C^A \times C^B & \\
(B \times C)^A & \cong & B^A \times C^A & \\
A \cong C \ \wedge\ B \cong D & \Rightarrow & A + B \cong C + D \ \wedge\ A \times B \cong C \times D \ \wedge\ B^A \cong D^C & \\
\mathcal{P}(A) & \ncong & A & \\
\mathcal{P}(A) & \cong & 2^A & \textit{(Funktionsdarstellung von Teilmengen)} \\
\mathcal{P}(A \times B) & \cong & \mathcal{P}(B)^A & \textit{(Funktionsdarstellung binärer Relationen)} \\
A \cong B & \Rightarrow & \mathcal{P}(A) \cong \mathcal{P}(B) &
\end{array}
$$

4 (Co)Induktiv Definieren und Beweisen

4.1 (Co)Induktiv definierte Mengen

Mit Mengenoperatoren lassen sich aus gegebenen Mengen neue bilden. Weiterhin haben wir gesehen, wie mit Hilfe der Komprehension eine Teilmenge T einer gegebenen Menge S beschrieben wird, nämlich indem man T als Menge aller Elemente von S definiert, die eine logische Formel φ erfüllen. Damit ist φ eine *endliche* Beschreibung von T – auch wenn T selbst unendlich ist.

Insbesondere unendliche Mengen erhält man oft als Lösungen von Gleichungen der Form $F(X) = X$ in der Mengenvariablen X, wobei $F : \mathcal{P}(S) \to \mathcal{P}(S)$ eine **monotone Funktion** ist, d.h. für alle Teilmengen T, T' von S gilt:

$$T \subseteq T' \quad \Rightarrow \quad F(T) \subseteq F(T').$$

$T \subseteq S$ heißt **Fixpunkt** von F, wenn $F(T)$ mit T übereinstimmt.

Hier ist also die monotone Funktion F (und keine Formel φ wie oben) eine endliche Beschreibung von T. Hat F mehrere Fixpunkte, dann muss man noch dazusagen, welchem (dem kleinsten, dem größten, etc.; s.u.) T entspricht.

Der Beweis, dass T ein Fixpunkt von F ist, besteht – wie immer bei Gleichungen zwischen Mengen – aus dem Nachweis zweier Inklusionen, nämlich

(1) $F(T) \subseteq T$ und (2) $T \subseteq F(T)$.

T heißt **F-abgeschlossen** bzw. **F-dicht**, wenn (1) bzw. (2) gilt.

Die Monotonie von F garantiert die Existenz eines kleinsten wie auch eines größten Fixpunkts von F:

Satz 4.1 Fixpunktsatz für (co)induktiv definierte Mengen

Sei $F : \mathcal{P}(S) \to \mathcal{P}(S)$ monoton und φ eine mögliche Eigenschaft von Elementen von S.

(i) $lfp(F) =_{def} \bigcap\{T \subseteq S \mid T \text{ } F\text{-abgeschlossen}\}$ ist die kleinste F-abgeschlossene Teilmenge von S und der kleinste Fixpunkt von F.

(ii) $gfp(F) =_{def} \bigcup\{T \subseteq S \mid T \text{ } F\text{-dicht}\}$ ist die größte F-dichte Teilmenge von S und der größte Fixpunkt von F.

(iii) **F-Induktion**: Alle Elemente von $lfp(F)$ haben eine gegebene Eigenschaft φ, wenn die Menge T_φ aller Elemente von S, die φ erfüllen, F-abgeschlossen ist; als Beweisregel:

$$\frac{lfp(F) \subseteq T_\varphi}{F(T_\varphi) \subseteq T_\varphi} \Uparrow$$

(iv) **F-Coinduktion**: Alle Elemente von S, die eine gegebene Eigenschaft φ erfüllen, gehören zu $gfp(F)$, wenn die Menge T_φ aller Elemente von S, die φ erfüllen, F-dicht ist; als Beweisregel:

$$\frac{T_\varphi \subseteq gfp(F)}{T_\varphi \subseteq F(T_\varphi)} \Uparrow$$

Beweis von (i). Sei T eine F-abgeschlossene Teilmenge von S. Dann gilt:

$$lfp(F) = \bigcap\{U \subseteq S \mid F(U) \subseteq U\} \subseteq T. \tag{1}$$

Da F monoton und T F-abgeschlossen ist, folgt

$$F(lfp(F)) \subseteq F(T) \subseteq T \tag{2}$$

aus (1). Da T beliebig gewählt wurde, ist $F(lfp(F))$ eine Teilmenge jeder F-abgeschlossenen Teilmenge von S, also auch des Durchschnittes aller F-abgeschlossenen Teilmengen von S, d.h.

$$F(lfp(F)) \subseteq \bigcap\{U \subseteq S \mid F(U) \subseteq U\} = lfp(F), \tag{3}$$

m.a.W.: $lfp(F)$ ist selbst F-abgeschlossen, zusammen mit (1) also die kleinste F-abgeschlossene Teilmenge von S.

Da F monoton ist, folgt

$$F(F(lfp(F))) \subseteq F(lfp(F)) \tag{4}$$

aus (3), d.h. auch $F(lfp(F))$ ist F-abgeschlossen. Also gilt

$$lfp(F) = \bigcap\{U \subseteq S \mid F(U) \subseteq U\} \subseteq F(lfp(F)). \tag{5}$$

Aus (3) und (5) folgt $F(lfp(F)) = lfp(F)$, d.h. $lfp(F)$ ist ein Fixpunkt von F.

Sei T ein beliebiger Fixpunkt von F. Dann ist T F-abgeschlossen. Also gilt $lfp(F) \subseteq T$, d.h. $lfp(F)$ ist der kleinste Fixpunkt von F.

Aufgabe Zeigen Sie (ii) analog zu (i).

Beweis von (iii). Sei T_φ F-abgeschlossen. Aus (i) folgt dann

$$lfp(F) = \bigcap\{U \subseteq S \mid F(U) \subseteq U\} \subseteq T_\varphi.$$

Beweis von (iv). Sei T_φ F-dicht. Aus (ii) folgt dann

$$T_\varphi \subseteq \bigcup\{U \subseteq S \mid U \subseteq F(U)\} = gfp(F).$$

❑

Falls sich beim Versuch, $F(T_\varphi) \subseteq T_\varphi$ bzw. $T_\varphi \subseteq F(T_\varphi)$ zu beweisen, herausstellt, dass die Inklusion *nicht* gilt, muss man φ *verallgemeinern*, indem man φ mit einer weiteren Eigenschaft ψ konjunktiv bzw. disjunktiv verknüpft und versucht, die F-Abgeschlossenheit von $T_{\varphi\wedge\psi}$ bzw. die F-Dichtheit von $T_{\varphi\vee\psi}$ zu zeigen. Gelingt auch das nicht, klappt es vielleicht bei Hinzunahme einer dritten Eigenschaft; usw.

Abbrechen kann man dieses – **inkrementelle (Co)Induktion** genannte – Verfahren, sobald $T_{\varphi\wedge\ldots}$ eine echte Teilmenge von $lfp(F)$ bzw. $gfp(F)$ eine echte Teilmenge von $T_{\varphi\vee\ldots}$ ist. Da jedoch $T_\varphi \setminus lfp(F)$ bzw. $gfp(F) \setminus T_\varphi$ i.d.R. unendlich ist, liefern (iii) und (iv) keine *vollständigen* Verfahren zum Beweis von $lfp(F) \subseteq T_\varphi$ bzw. $T_\varphi \subseteq gfp(F)$. Falls diese Inklusion gilt, *existiert* aber zumindest eine Eigenschaft ψ derart, dass $T_{\varphi\wedge\psi}$ F-abgeschlossen bzw. $T_{\varphi\vee\psi}$ F-dicht ist, d.h. die obigen Beweisregeln können wie folgt verschärft werden:

$$\frac{lfp(F) \subseteq T_\varphi}{\exists\, \psi : F(T_{\varphi\wedge\psi}) \subseteq T_{\varphi\wedge\psi}} \Updownarrow \qquad \frac{T_\varphi \subseteq gfp(F)}{\exists\, \psi : T_{\varphi\vee\psi} \subseteq F(T_{\varphi\vee\psi})} \Updownarrow$$

Rein mengentheoretisch, ohne Bezug auf Eigenschaften φ, ψ:

$$\frac{lfp(F) \subseteq T}{\exists\, T' \subseteq T : F(T') \subseteq T'} \Updownarrow \qquad \frac{T \subseteq gfp(F)}{\exists\, T' \supseteq T : T' \subseteq F(T')} \Updownarrow$$

Aufgabe Folgern Sie die Gültigkeit dieser beiden Äquivalenzen aus der Gültigkeit der Implikationen in (iii) bzw. (iv). ❑

$T \subseteq S$ heißt **induktiv definiert**, wenn es eine monotone Funktion $F : \mathcal{P}(S) \to \mathcal{P}(S)$ mit $lfp(F) = T$ gibt, wenn also T die kleinste F-abgeschlossene Teilmenge von S ist.

$T \subseteq S$ heißt **coinduktiv definiert**, wenn es eine monotone Funktion $F : \mathcal{P}(S) \to \mathcal{P}(S)$ mit $gfp(F) = T$ gibt, wenn also T die größte F-dichte Teilmenge von S ist.

In beiden Fällen nennen wir F die **Schrittfunktion für** T.

Satz 4.2 $\mathbb{N}$ ist induktiv definiert. Eine Schrittfunktion für $\mathbb{N}$ lautet wie folgt:

$$\begin{array}{rcl} F_{nat} : \mathcal{P}(\mathbb{N}) & \to & \mathcal{P}(\mathbb{N}) \\ T & \mapsto & \{0\} \cup \{n+1 \mid n \in T\} \end{array}$$

Beweis. Offenbar ist F_{nat} monoton. $\mathbb{N}$ ist F_{nat}-abgeschlossen:

$$F_{nat}(\mathbb{N}) = \{0\} \cup \{n+1 \mid n \in \mathbb{N}\} \subseteq \mathbb{N}.$$

Sei $T \subseteq \mathbb{N}$ F_{nat}-abgeschlossen, aber $\mathbb{N}$ keine Teilmenge von T. Dann ist $\mathbb{N} \setminus T$ nach Satz 3.2 (1) eine nichtleere Teilmenge von $\mathbb{N} \setminus F_{nat}(T)$.

Sei n das (bzgl. der üblichen kleiner-Relation auf $\mathbb{N}$) kleinste Element von $\mathbb{N} \setminus T$. Dann gilt

$$n \in \mathbb{N} \setminus T \subseteq \mathbb{N} \setminus F_{nat}(T),$$

also $n \neq 0$ und $n \neq m+1$, also $n-1 \neq m$ für alle $m \in T$. Daraus folgt $n-1 \in \mathbb{N} \setminus T$. Das widerspricht der Voraussetzung, dass n das kleinste Element von $\mathbb{N} \setminus T$ ist. Daher gilt $\mathbb{N} \subseteq T$.

Demnach ist die F_{nat}-abgeschlossene Menge $\mathbb{N}$ in jeder F_{nat}-abgeschlossenen Teilmenge von $\mathbb{N}$ enthalten. $\mathbb{N}$ ist also die kleinste F_{nat}-abgeschlossene Teilmenge von $\mathbb{N}$ und stimmt daher nach Satz 4.1 (i) mit $lfp(F_{nat})$ überein. ❑

Beispiel 4.3 Zeigen Sie analog zum Beweis von Satz 4.2, dass die Menge *evens* der geraden Zahlen induktiv definiert und

$$\begin{array}{rcl} F_{even} : \mathcal{P}(\mathbb{N}) & \to & \mathcal{P}(\mathbb{N}) \\ T & \mapsto & \{0\} \cup \{n+2 \mid n \in T\} \end{array}$$

eine Schrittfunktion für *evens* ist. ❑

Satz 4.4 Sei A eine Menge. A^* ist induktiv definiert. Eine Schrittfunktion für A^* (siehe Kapitel 3) lautet wie folgt:

$$\begin{array}{rcl} F_{list} : \mathcal{P}(A^*) & \to & \mathcal{P}(A^*) \\ T & \mapsto & \{\epsilon\} \cup A \cup \{(a, a_1, \ldots, a_n) \mid a \in A,\ (a_1, \ldots, a_n) \in T\} \end{array}$$

Beweis. Offenbar ist F_{list} monoton. A^* ist F_{list}-abgeschlossen:

$$F_{list}(A^*) = \{\epsilon\} \cup A \cup \{(a, a_1, \ldots, a_n) \mid a \in A,\ (a_1, \ldots, a_n) \in A^*,\ n \in \mathbb{N}\} \subseteq A^*.$$

Sei $T \subseteq A^*$ F_{list}-abgeschlossen, aber A^* keine Teilmenge von T. Dann ist $A^* \setminus T$ nach Satz 3.2 (1) eine nichtleere Teilmenge von $A^* \setminus F_{list}(T)$.

Sei w ein Tupel von $A^* \setminus T$ derart, dass kein Tupel von $A^* \setminus T$ kürzer als w ist. Dann gilt

$$w \in A^* \setminus T \subseteq A^* \setminus F_{nat}(T),$$

also $w \neq \epsilon$, $w \neq a$ und $w \neq (a, a_1, \ldots, a_n)$ für alle $a \in A$ und $(a_1, \ldots, a_n) \in T$, also $tail(w) \in A^* \setminus T$, wobei $tail(w)$ des Tupel w ohne dessen erste Komponente ist. Das widerspricht der Voraussetzung, dass kein Tupel von $A^* \setminus T$ kürzer als w ist. Daher gilt $A^* \subseteq T$.

Demnach ist die F_{list}-abgeschlossene Menge A^* in jeder F_{list}-abgeschlossenen Teilmenge von A^* enthalten. A^* ist also kleinste F_{list}-abgeschlossene Teilmenge von A^* und stimmt daher nach Satz 4.1 (i) mit $lfp(F_{list})$ überein. ❑

Beispiel 4.5 Zeigen Sie analog zum Beweis von Satz 4.2, dass die Menge *sorted* aufsteigend sortierter Listen reeller Zahlen induktiv definiert und

$$\begin{array}{rcl} F_{sort} : \mathcal{P}(\mathbb{R}^*) & \to & \mathcal{P}(\mathbb{R}^*) \\ T & \mapsto & \{\epsilon\} \cup \mathbb{R} \cup \{(a, a_1, \ldots, a_n) \mid a \in \mathbb{R},\ a \leq a_1,\ (a_1, \ldots, a_n) \in T,\ n > 0\} \end{array}$$

eine Schrittfunktion für *sorted* ist. ❑

Beispiel 4.6 Zeigen Sie analog zum Beweis von Satz 4.4, dass die Menge

$$\mathbb{R}[x] = \{\lambda x. \sum_{i=0}^{n} a_i * x^i \mid n \in \mathbb{N},\ a_n \neq 0\}$$

reellwertiger Polynome induktiv definiert und

$$\begin{array}{rcl} F_{poly} : \mathcal{P}(\mathbb{R}^{\mathbb{R}}) & \to & \mathcal{P}(\mathbb{R}^{\mathbb{R}}) \\ T & \mapsto & \{\lambda x.a \mid a \in \mathbb{R}\} \cup \{\lambda x.(a + x * p(x)) \mid a \in \mathbb{R},\ p \in T\} \end{array}$$

eine Schrittfunktion für $\mathbb{R}[x]$ ist. ❏

Satz 4.7 $T \subseteq S$ ist genau dann induktiv definiert, wenn $S \setminus T$ coinduktiv definiert ist.

Beweis. "⇒": Sei F eine Schrittfunktion für T und $G : \mathcal{P}(S) \to \mathcal{P}(S)$ wie folgt definiert: Für alle $U \subseteq S$, $G(U) = S \setminus F(S \setminus U)$.

Ist F ist monoton, dann auch G: Sei $T \subseteq U$. Aus Satz 3.2 (1) folgt $S \setminus U \subseteq S \setminus T$, also $F(S \setminus U) \subseteq F(S \setminus T)$ und daher

$$G(T) = S \setminus F(S \setminus T) \overset{\text{Satz 3.2 (1)}}{\subseteq} S \setminus F(S \setminus U) = G(U).$$

G ist eine Schrittfunktion für $S \setminus T$:

$$\begin{array}{l} gfp(G) \overset{\text{Satz 4.1 (ii)}}{=} \bigcup\{T \subseteq S \mid T \subseteq G(T)\} \overset{\text{Def. } G}{=} \bigcup\{T \subseteq S \mid T \subseteq S \setminus F(S \setminus T)\} \\ \overset{\text{Satz 3.2 (1)}}{=} \bigcup\{T \subseteq S \mid F(S \setminus T) \subseteq S \setminus T\} \\ = \bigcup\{S \setminus (S \setminus T) \subseteq S \mid T \subseteq S,\ F(S \setminus T) \subseteq S \setminus T\} \\ = \bigcup\{S \setminus T \mid T \subseteq S,\ F(T) \subseteq T\} \overset{\text{Satz 3.2 (2)}}{=} S \setminus \bigcap\{T \subseteq S \mid F(T) \subseteq T\} \\ \overset{\text{Satz 4.2 (i)}}{=} S \setminus lfp(F) \overset{lfp(F)=T}{=} S \setminus T. \end{array}$$

"⇐": analog. ❏

4.2 Induktion über $n \in \mathbb{N}$

Gezeigt werden mit dieser Beweismethode Eigenschaften φ aller natürlichen Zahlen. Sie besteht darin, φ zunächst für die Null zu zeigen (*Induktionsanfang*) und dann aus der – *Induktionshypothese* (oder *Induktionsvoraussetzung*) genannten – Gültigkeit von φ für n die Gültigkeit von φ für $n + 1$ zu folgern (*Induktionsschritt*).

Sei F_{nat} wie in Satz 4.2 definiert. Wie man leicht sieht, entsprechen Induktionsanfang und Induktionsschritt einem Beweis, dass die Menge $M(\varphi)$ aller natürlichen Zahlen, die φ erfüllen, F_{nat}-abgeschlossen ist.

Die Korrektheit der Induktion über n folgt demnach aus der Monotonie von F_{nat}, Satz 4.1 (iii) und Satz 4.2:

$F_{nat}(M(\varphi)) \subseteq M(\varphi)$ impliziert $\mathbb{N} = lfp(F_{nat}) \subseteq M(\varphi)$, d.h. alle natürlichen Zahlen erfüllen φ.

Induktion über $n \in \mathbb{N}$ ist also F_{nat}-Induktion. ❏

Aufgabe Zeigen Sie durch Induktion über n, dass jede natürliche Zahl gerade oder ungerade ist. ❏

Induktion über $n \in \mathbb{N}$ lässt sich verallgemeinern zur *Noetherschen Induktion* (s.u.), die anstelle von $\mathbb{N}$ eine beliebige Menge mit *wohlfundierter* Relation voraussetzt:

Sei A eine Menge. Eine Relation $R \subseteq A^2$ heißt **wohlfundiert**, wenn jede nichtleere Teilmenge M von A ein bzgl. R minimales Element a enthält, d.h. für alle $b \in M$ gilt $(b, a) \notin R$.

Z.B. ist die kleiner-Relation $<$ auf $\mathbb{N}$ wohlfundiert. Jede nichtleere Teilmenge M enthält hier sogar ein kleinstes Element bzgl. $<$. Die echte-Teilmengen-Relation $\subset$ auf der Potenzmenge $\mathcal{P}(A)$ einer endlichen Menge A ist auch wohlfundiert. Die meisten Elemente von $\mathcal{P}(A)$ (= Teilmengen von A) haben bzgl. $\subset$ mehrere minimale Elemente.

Satz 4.8 Sei $R \subseteq A^2$ eine wohlfundierte Relation. Dann ist A induktiv definiert. Eine Schrittfunktion für A lautet

wie folgt:

$$\begin{array}{rcl} F_R : \mathcal{P}(A) & \to & \mathcal{P}(A) \\ T & \mapsto & \{a \in A \mid \forall\, b \in A : (b,a) \in R \Rightarrow b \in T\} \end{array}$$

Beweis. Offenbar ist F_R monoton. A ist F_R-abgeschlossen:

$$F_R(A) = \{a \in A \mid \forall\, b \in A : (b,a) \in R \Rightarrow b \in A\} \subseteq A.$$

Sei $T \subseteq A$ F_R-abgeschlossen.

Wir nehmen an, dass A keine Teilmenge von T ist. Dann wäre $A \setminus T$ nichtleer. Da R wohlfundiert ist, enthält $A \setminus T$ ein bzgl. R minimales Element a. Wegen $F_R(T) \subseteq T$ folgt $a \notin F_R(T)$, also $b \in A \setminus T$ für ein $(b,a) \in R$. Das widerspricht der Definition von a als minimalem Element von $A \setminus T$. Also gilt $A \subseteq T$.

Demnach ist A die kleinste F_R-abgeschlossene Teilmenge von A und stimmt daher nach Satz 4.1 (i) mit $lfp(F_R)$ überein. ❑

4.3 Noethersche Induktion bzgl. $R \subseteq A^2$

Gezeigt werden mit dieser Beweismethode Eigenschaften φ einer Menge A, für die es eine wohlfundierte Relation $R \subseteq A^2$ gibt.

Ein Beweis von φ durch Noethersche Induktion bzgl. R besteht im Nachweis von φ für alle $a \in A$ *unter der Voraussetzung*, dass φ für alle $b \in A$ mit $(b,a) \in R$ gilt. Das entspricht einem Beweis, dass die Menge $M(\varphi)$ aller Elemente von A, die φ erfüllen, F_R-abgeschlossen ist.

Die Korrektheit Noetherscher Induktion bzgl. R folgt demnach aus der Monotonie von F_R, Satz 4.1 (iii) und Satz 4.8: $F_R(M(\varphi)) \subseteq M(\varphi)$ impliziert $A = lfp(F_R) \subseteq M(\varphi)$, d.h. alle Elemente von A erfüllen φ.

Noethersche Induktion bzgl. R ist also F_R-Induktion.

Induktion über $n \in \mathbb{N}$ ist Noethersche Induktion bzgl. $R = \{(n, n+1) \mid n \in \mathbb{N}\}$. ❑

Da F monoton ist, bilden die Iterationen

$$F^0(\emptyset), F^1(\emptyset), F^2(\emptyset), \ldots \quad \text{bzw.} \quad F^0(S), F^1(S), F^2(S), \ldots$$

von F^∞ bzw. F_∞ eine **auf-** bzw. **absteigende Kette**, d.h. für alle $n \in \mathbb{N}$ gilt

$$F^n(\emptyset) \subseteq F^{n+1}(\emptyset) \quad \text{bzw.} \quad F^n(S) \supseteq F^{n+1}(S).$$

Aus der Ketteneigenschaft folgt sofort, dass im Fall einer *endlichen* Obermenge S der obere bzw. untere Kleene-Abschluss von F mit einer seiner Iterationen übereinstimmt und damit F-abgeschlossen bzw. F-dicht ist.

$$F^\infty =_{def} \bigcup_{n \in \mathbb{N}} F^n(\emptyset) \quad \text{bzw.} \quad F_\infty =_{def} \bigcap_{n \in \mathbb{N}} F^n(S)$$

heißt **oberer** bzw. **unterer Kleene-Abschluss von** F.

Satz 4.9 Sei $F : \mathcal{P}(S) \to \mathcal{P}(S)$ monoton.

(i) $F^\infty \subseteq lfp(F)$.

(ii) $F(F^\infty) \subseteq F^\infty \;\Rightarrow\; F^\infty = lfp(F)$.

(iii) $gfp(F) \subseteq F_\infty$.

(iv) $F_\infty \subseteq F(F_\infty) \;\Rightarrow\; F_\infty = gfp(F)$.

Beweis von (i). Wir zeigen

$$F^n(\emptyset) \subseteq lfp(F) \tag{1}$$

für alle $n \in \mathbb{N}$ durch Induktion über n: Induktionsanfang: $F^0(\emptyset) = \emptyset \subseteq lfp(F)$.

Induktionsschritt: Da F monoton ist und $lfp(F)$ ein Fixpunkt von F, gilt

$$F^{n+1}(\emptyset) = F(F^n(\emptyset)) \overset{(1)}{\subseteq} F(lfp(F)) = lfp(F).$$

Also gilt $F^\infty = \bigcup_{n\in\mathbb{N}} F^n(\emptyset) \subseteq lfp(F)$.

Beweis von (ii). Sei F^∞ F-abgeschlossen. Also gilt $lfp(F) \subseteq F^\infty$ nach Satz 4.1 (iii). Aus (i) folgt daher $F^\infty = lfp(F)$.

Beweis von (iii). Wir zeigen

$$gfp(F) \subseteq F^n(S) \tag{2}$$

für alle $n \in \mathbb{N}$ durch Induktion über n: Induktionsanfang: $gfp(F) \subseteq S = F^0(S)$.

Induktionsschritt: Da F monoton ist und $gfp(F)$ ein Fixpunkt von F, gilt

$$gfp(F) = F(gfp(F)) \overset{(2)}{\subseteq} F(F^n(S)) = F^{n+1}(S).$$

Also gilt $gfp(F) \subseteq \bigcap_{n\in\mathbb{N}} F^n(S) = F_\infty$.

Beweis von (iv). Sei F_∞ F-dicht. Also gilt $F_\infty \subseteq gfp(F)$ nach Satz 4.1 (iv). Aus (iii) folgt daher $F_\infty = gfp(F)$. ❑

F_R^∞ (siehe Satz 4.8) wäre nicht F_R-abgeschlossen, wenn ein Element $a \in F_R(F_R^\infty)$ unendlich viele Vorgänger bzgl. R hätte, die in verschiedenen Approximationen von F_R^∞ vorkommen. Dann gäbe es zwar für alle $(b,a) \in R$ $n_b \in \mathbb{N}$ mit $b \in F_R^{n_b}(\emptyset)$, aber kein $n \in \mathbb{N}$ mit $b \in F_R^n(\emptyset)$ für alle $(b,a) \in R$, also auch kein $n \in \mathbb{N}$ mit $a \in F_R(F_R^n(\emptyset)) = F_R^{n+1}(\emptyset)$, d.h. a würde nicht zu F_R^∞ gehören. Die Wohlfundiertheit von R schließt diesen Fall jedoch aus.

4.4 Logische Ableitungen

Formeln bilden induktive Mengen. Für deren Schrittfunktionen verweisen wir auf spätere Kapitel.

In diesem Abschnitt geht es um die aus einer beliebigen Formelmenge Fo für einen Kalkül K gebildeten K-ableitbaren Ausdrücke der Form $\Phi \vdash \varphi$ (siehe Kapitel 2), die, falls K synthetisch ist, eine induktiv definierte und, falls K analytisch ist, eine coinduktiv definierte Menge bilden:

Laut Kapitel 2 besteht K aus Regeln der Form

$$\frac{\Phi_1 \vdash \varphi_1,\ \ldots,\ \Phi_k \vdash \varphi_k}{\Phi \vdash \varphi},$$

falls K synthetisch ist, oder aus Regeln der Form

$$\frac{\Phi \vdash \varphi}{\Phi_1 \vdash \varphi_1,\ \ldots,\ \Phi_k \vdash \varphi_k},$$

falls K analytisch ist, mit $\Phi_1, \ldots, \Phi_k, \Phi \subseteq Fo$ und $\varphi_1, \ldots, \varphi_k, \varphi \in Fo$.

Mengentheoretisch betrachtet, ist jedes Urteil $\Phi \vdash \varphi$ ein Paar (Φ, φ) aus der Menge

$$\mathcal{R}(Fo) \ =_{def}\ \mathcal{P}(Fo) \times Fo.$$

Der Begiff der K-Ableitung basiert auf folgender Schrittfunktion:

$$\begin{array}{rcl} F_K : \mathcal{P}(\mathcal{R}(Fo)) & \to & \mathcal{P}(\mathcal{R}(Fo)) \\ R & \mapsto & \begin{cases} \{a \mid \frac{a_1,\ldots,a_n}{a} \in K,\ a_1,\ldots,a_n \in R\} & \text{falls } K \text{ synthetisch ist,} \\ \{a \mid \frac{a}{a_1,\ldots,a_n} \in K,\ a_1,\ldots,a_n \in R\} & \text{falls } K \text{ analytisch ist.} \end{cases} \end{array}$$

$\vdash_K =_{def} F_K^\infty$ heißt K**-Ableitungsrelation**.

Die Elemente von $\vdash_K$ heißen K**-ableitbar**. Man benutzt die Infixschreibweise, schreibt also $\Phi \vdash_K \varphi$ anstelle von $(\Phi, \varphi) \subseteq \vdash_K$.

Sei K synthetisch. $(a_1, \ldots, a_n) \in \mathcal{R}(Fo)^*$ heißt K**-Ableitung von** a_n, falls für alle $1 \leq i \leq n$, $a_i \in F_K(\{a_1, \ldots, a_{i-1}\})$.

Sei K analytisch. $(a_1, \ldots, a_n) \in \mathcal{R}(Fo)^*$ heißt K**-Ableitung von** a_1, falls für alle $1 \leq i \leq n$, $a_i \in F_K(\{a_{i-1}, \ldots, a_n\})$.

Satz 4.10 Für alle Kalküle K gilt $\vdash_K = lfp(F_K)$.

Beweis. Wir zeigen zunächst, dass $\vdash_K = F_K^\infty$ F_K-abgeschlossen ist.

Sei K synthetisch. Dann gilt

$$\begin{aligned} F_K(F_K^\infty) &= \{a \mid \tfrac{a_1,\ldots,a_n}{a} \in K,\ a_1, \ldots, a_n \in F_K^\infty\} \\ &= \bigcup_{i\in\mathbb{N}} \{a \mid \tfrac{a_1,\ldots,a_n}{a} \in K,\ a_1, \ldots, a_n \in F_K^i(\emptyset)\} = \bigcup_{i\in\mathbb{N}} F_K^{i+1}(\emptyset) = F_K^\infty. \end{aligned}$$

Sei K analytisch. Dann gilt

$$\begin{aligned} F_K(F_K^\infty) &= \{a \mid \tfrac{a}{a_1,\ldots,a_n} \in K,\ a_1, \ldots, a_n \in F_K^\infty\} \\ &= \bigcup_{i\in\mathbb{N}} \{a \mid \tfrac{a}{a_1,\ldots,a_n} \in K,\ a_1, \ldots, a_n \in F_K^i(\emptyset)\} = \bigcup_{n\in\mathbb{N}} F_K^{i+1}(\emptyset) = F_K^\infty. \end{aligned}$$

Also folgt $\vdash_K = lfp(F_K)$ in beiden Fällen aus Satz 4.9 (i) und (ii), weil F_K monoton ist. ❏

4.5 Weitere (co)induktiv definierte Mengen

Beispiele 4.11 (Induktive Definitionen von Eigenschaften einer Komponente eines unendlichen Objekts)

Ströme über einer Menge A (siehe Kapitel 3) entsprechen Funktionen von $\mathbb{N}$ nach A.

Zeigen Sie analog zum Beweis von Satz 4.2, dass die Mengen *has0* und *unsorted* der Ströme reeller Zahlen mit mindestens einer Null bzw. der unsortierten Ströme reeller Zahlen mit den Schrittfunktionen

$$\begin{aligned} F : \mathcal{P}(\mathbb{R}^\mathbb{N}) &\to \mathcal{P}(\mathbb{R}^\mathbb{N}) \\ T &\mapsto \{s \in \mathbb{R}^\mathbb{N} \mid s(0) = 0 \vee \lambda i.s(i+1) \in T\}, \\ G : \mathcal{P}(\mathbb{R}^\mathbb{N}) &\to \mathcal{P}(\mathbb{R}^\mathbb{N}) \\ T &\mapsto \{s \in \mathbb{R}^\mathbb{N} \mid s(0) > s(1) \vee \lambda i.s(i+1) \in T\} \end{aligned}$$

induktiv definiert sind.

Anschaulich gesprochen ist $s(0)$ das erste Element von s und $\lambda i.s(i+1)$ der Strom, der aus s entsteht, wenn $s(0)$ entfernt wird. ❏

Beispiele 4.12 (Coinduktive Definition von Eigenschaften aller Komponenten eines unendlichen Objekts)

(i) Sei *sorted* die Menge der sortierten Ströme reeller Zahlen, also

$$sorted = \{s \in \mathbb{R}^\mathbb{N} \mid \forall\, n \in \mathbb{N} : s(n) \leq s(n+1)\}.$$

sorted ist coinduktiv definiert. Eine Schrittfunktion für *sorted* lautet wie folgt:

$$\begin{aligned} F_{sorted} : \mathcal{P}(\mathbb{R}^\mathbb{N}) &\to \mathcal{P}(\mathbb{R}^\mathbb{N}) \\ T &\mapsto \{s \in \mathbb{R}^\mathbb{N} \mid s(0) \leq s(1) \wedge \lambda i.s(i+1) \in T\}. \end{aligned}$$

(ii) Die Menge *has*∞*0* der Ströme reeller Zahlen mit unendlich vielen Nullen ist coinduktiv definiert. Eine Schrittfunktion für *has*∞*0* lautet wie folgt:

$$\begin{aligned} F_{has\infty 0} : \mathcal{P}(\mathbb{R}^\mathbb{N}) &\to \mathcal{P}(\mathbb{R}^\mathbb{N}) \\ T &\mapsto \{s \in has0 \mid \lambda i.s(i+1) \in T\}. \end{aligned}$$

Beweis von (i). Offenbar ist $F =_{def} F_{sorted}$ monoton. *sorted* ist F-dicht:

$$sorted \subseteq \{s \in \mathbb{R}^{\mathbb{N}} \mid s(0) \leq s(1),\ \lambda i.s(i+1) \in sorted\} = F(sorted).$$

Sei $T \subseteq \mathbb{R}^{\mathbb{N}}$ F-dicht und $s \in T$.

Wir zeigen durch Induktion über n, dass für alle $n \in \mathbb{N}$ Folgendes gilt:

$$s(n) \leq s(n+1) \text{ und } \lambda i.s(i+n+1) \in T. \tag{1}$$

Induktionsanfang: Aus $s \in T \subseteq F(T)$ folgt $s(0) \leq s(1)$ und $\lambda i.s(i+1) \in T$. Also gilt (1) für $n = 0$.

Induktionsschritt: Es gelte (1) für n. Daraus folgt $\lambda i.s(i+n+1) \in T \subseteq F(T)$, also

$$s(n+1) = (\lambda i.s(i+n+1))(0) \leq (\lambda i.s(i+n+1))(1) = s(n+2)$$

sowie $\lambda i.s(i+n+2) = \lambda i.(\lambda i.s(i+n+1))(i+1) \in T$ und damit (1) für $n+1$.

(1) impliziert $s \in sorted$. Also gilt $T \subseteq sorted$. Demnach ist *sorted* die größte F-dichte Teilmenge von $\mathbb{R}^{\mathbb{N}}$ und stimmt daher nach Satz 4.1 (ii) mit $gfp(F)$ überein.

Beweis von (ii). Offenbar ist $F =_{def} F_{has\infty 0}$ monoton. $has\infty 0$ ist F-dicht:

$$has\infty 0 \subseteq \{s \in has0 \mid \lambda i.s(i+1) \in has\infty 0\} = F(has\infty 0).$$

Sei $T \subseteq \mathbb{R}^{\mathbb{N}}$ F-dicht und $s \in T$.

Wir zeigen durch Induktion über n, dass für alle $n \in \mathbb{N}$ Folgendes gilt:

$$\lambda i.s(i+n) \in has0 \text{ und } \lambda i.s(i+n+1) \in T. \tag{2}$$

Induktionsanfang: Aus $s \in T \subseteq F(T)$ folgt $s \in has0$ und $\lambda i.s(i+1) \in T$. Also gilt (2) für $n = 0$.

Induktionsschritt: Es gelte (2) für n. Daraus folgt $\lambda i.s(i+n+1) \in T \subseteq F(T)$, also $\lambda i.s(i+n+1) \in has0$ sowie

$$\lambda i.s(i+n+2) = \lambda i.(\lambda i.s(i+n+1))(i+1) \in T$$

und damit (2) für $n+1$.

(2) impliziert $s \in has\infty 0$. Also gilt $T \subseteq has\infty 0$. Demnach ist $has\infty 0$ die größte F-dichte Teilmenge von $\mathbb{R}^{\mathbb{N}}$ und stimmt daher nach Satz 4.1 (ii) mit $gfp(F)$ überein. ❑

Beispiel 4.13 (Beweis durch $F_{has\infty 0}$-Coinduktion)

Seien $blink, blink' \in has0$ wie folgt definiert: Für alle $n \in \mathbb{N}$,

$$blink(2*n) = blink'(2*n+1) = 0 \quad \text{und} \quad blink(2*n+1) = blink'(2*n) = 1.$$

Die Menge $BB = \{blink, blink'\}$ ist $F_{has\infty 0}$-dicht:

$$BB \subseteq \{s \in has0 \mid \lambda i.s(i+1) \in BB\} = F_{has\infty 0}(BB).$$

Also gehören *blink* und *blink'* nach Satz 4.1 (iv) zu $gfp(F_{has\infty 0})$, was wegen $gfp(F_{has\infty 0}) = has\infty 0$ (siehe 4.12 (ii)) bedeutet, dass *blink* und *blink'* unendlich viele Nullen enthalten. ❑

Manchmal können $n > 1$ Teilmengen von S nur **wechselseitig** (co)induktiv definiert werden. Die zugehörige Schrittfunktion hat dann den Definitions- und Wertebereich $\mathcal{P}(S)^n$, wobei Inklusion und Vereinigung von Teilmengen von S wie folgt auf n-Tupel $(T_1, \ldots, T_n)$ von Teilmengen von S fortgesetzt werden:

$$\begin{array}{lll} (T_1, \ldots, T_n) \subseteq (T_1', \ldots, T_n') & \Leftrightarrow_{def} & \text{für alle } 1 \leq i \leq n,\ T_i \subseteq T_i', \\ (T_1, \ldots, T_n) \cup (T_1', \ldots, T_n') & =_{def} & (T_1 \cup T_1', \ldots, T_n \cup T_n'). \end{array}$$

Aufgabe Zeigen Sie analog zum Beweis von Satz 4.2, dass das Paar $(evens, odds)$ der Mengen aller geraden bzw. ungeraden Zahlen induktiv definiert und

$$\begin{array}{rcl} F : \mathcal{P}(\mathbb{N})^2 & \to & \mathcal{P}(\mathbb{N})^2 \\ (T_1, T_2) & \mapsto & (\{0\} \cup \{n+1 \mid n \in T_2\}, \{n+1 \mid n \in T_1\}) \end{array}$$

eine Schrittfunktion für $(evens, odds)$ ist. ❑

Bei (co)logischen Programmen (siehe Kapitel 9) geht man noch einen Schritt weiter und interpretiert diese als induktiv definierte Tupel mehrstelliger Relationen. Tatsächlich stimmen die in Kapitel 9 definierten Modelle solcher Programme überein mit dem kleinsten bzw. größten Fixpunkt einer Schrittfunktion des Typs

$$\mathcal{P}(A^{k_1}) \times \ldots \times \mathcal{P}(A^{k_n}) \to \mathcal{P}(A^{k_1}) \times \ldots \times \mathcal{P}(A^{k_n}),$$

die n Relationen $R_1 \subseteq A^{k_1}, \ldots, R_n \subseteq A^{k_n}$ simultan schrittweise vergrößert bzw. verkleinert.

4.6 Konstruktoren und Destruktoren

Sei $\mathcal{C}$ eine endliche Menge von Funktionen mit Wertebereich S und Definitionsbereichen der Form $A \times S^n$. Im Fall $A = 1$ und $n > 0$ oder $n = 0$ entfällt die erste bzw. zweite Komponente von $A \times S^n$. Im Fall $A = 1$ und $n = 0$ wird $A \times S^n$ zu 1.

$\mathcal{C}$ induziert folgende Schrittfunktion, deren kleinster Fixpunkt aus den mittels Anwendungen von $\mathcal{C}$ konstruierten Elementen von S bestehen soll:

$$\begin{array}{rcl} F_{\mathcal{C}} : \mathcal{P}(S) & \to & \mathcal{P}(S) \\ T & \mapsto & \{c(w) \mid c : A \times S^n \to S \in \mathcal{C},\ w \in A \times T^n\} \end{array}$$

$F_{\mathcal{C}}$-Induktion (siehe Satz 4.1 (iii)) wird auch **strukturelle Induktion über** $\mathcal{C}$ genannt.

$F_{\mathcal{C}}$-abgeschlossene Mengen heißen auch **$\mathcal{C}$-Invarianten von** S.

$\mathcal{C} = \{c_i : A_i \to S \mid 1 \leq i \leq k\}$ heißt **Konstruktormenge für** S, falls die Summenextension

$$[c_1, \ldots, c_k] : A_1 \times \ldots \times A_k \to S$$

bijektiv ist (siehe Abschnitt 3.2).

Satz CONSTR $\mathcal{C}$ ist genau dann eine Konstruktormenge für S, wenn jedes Element von S genau eine eindeutige **$\mathcal{C}$-Repräsentation** hat, d.h. eindeutig als Ausdruck, der aus Funktionen von $\mathcal{C}$ und Elementen von $\bigcup_{c \in C} A_c$ besteht, dargestellt werden kann.

Beweis. Siehe [34], Kapitel 2. Man benötigt hier die aus der ☞ **Kategorientheorie** bekannte axiomatische Definition einer Summe, die impliziert, dass auch alle zu einer disjunkten Vereinigung isomorphen Mengen Summen sind. Danach ist $\mathcal{C}$ genau dann eine Konstruktormenge für S, wenn $\coprod_{c \in \mathcal{C}} dom(c)$ eine Summe ist (mit den Funktionen von $\mathcal{C}$ als Injektionen; siehe Abschnitt 3.3). ❑

Satz 4.14 Sei $S = lfp(F_{\mathcal{C}})$ und für alle $c : A \times S^m \to S$, $c' : B \times S^n \to S \in \mathcal{C}$, $t \in A \times S^m$ und $t' \in B \times S^n$ gelte folgende Implikation:

$$c(t) = c'(t') \quad \Rightarrow \quad c = c' \wedge t = t'. \tag{1}$$

Dann ist $\mathcal{C}$ eine Konstruktormenge für S.

Beweis durch $F_{\mathcal{C}}$-Induktion. Sei T die Menge aller Elemente von S mit eindeutiger $\mathcal{C}$-Repräsentation. Nach Satz 4.1 (iii) und Satz CONSTR ist zu zeigen, dass T $F_{\mathcal{C}}$-abgeschlossen ist.

Sei $s \in F_{\mathcal{C}}(T)$. Dann gibt es $c : A \times S^m \to S \in \mathcal{C}$, $a \in A$ und $s_1, \ldots, s_m \in T$ mit $s = c(a, s_1, \ldots, s_m)$. $s_1, \ldots, s_n$ haben also eindeutige $\mathcal{C}$-Repräsentationen. Sei $c' : B \times S^n \to S \in \mathcal{C}$, $b \in B$ und $s'_1, \ldots, s'_n \in S$ mit $s = c'(b, s'_1, \ldots, s'_n)$. Da T eine Teilmenge von S ist, folgt $c = c'$, $a = b$ und $s_i = s'_i$ für alle $1 \leq i \leq m = n$ aus (1). Daher ist auch die $\mathcal{C}$-Repräsentation von s eindeutig, d.h. s gehört zu T. ❑

Beispiel 4.15 Sei $S = \mathbb{N}$ und $\mathcal{C} = \{0 : 1 \to S,\ (+1) : S \to S\}$. Offenbar stimmt $F_{\mathcal{C}}$ mit der Schrittfunktion F_{nat} überein. Deshalb folgt $lfp(F_{\mathcal{C}}) = \mathbb{N}$ aus Satz 4.2. (1) ist in diesem Fall äquivalent zum 3. und 4. ☞ Peano-Axiom, gilt also trivialerweise. Daher ist $\mathcal{C}$ nach Satz 4.14 eine Konstruktormenge für S. ❑

Beispiel 4.16 Sei A eine Menge, $S = A^*$ und $\mathcal{C} = \{\epsilon : 1 \to S,\ cons : A \times S \to S\}$ mit

$$\begin{array}{rcl} cons(a, \epsilon) & =_{def} & a, \\ cons(a, b) & =_{def} & (a, b), \\ cons(a, a_1, \ldots, a_n) & =_{def} & (a, a_1, \ldots, a_n) \end{array}$$

für alle $a, b, a_1, \ldots, a_n \in A$. Offenbar stimmt $F_{\mathcal{C}}$ mit der Schrittfunktion F_{list} von Satz 4.4 überein. Deshalb folgt $lfp(F_{\mathcal{C}}) = A^*$ aus Satz 4.4. Auch hier gilt (1) trivialerweise. Daher ist $\mathcal{C}$ nach Satz 4.14 eine Konstruktormenge für S. ❑

Beispiel 4.17 Sei $S = \mathbb{R}[x]$ und $\mathcal{C} = \{const : \mathbb{R} \to S,\ add : \mathbb{R} \times S \to S\}$ mit

$$\begin{array}{rcl} const(a) & =_{def} & \lambda x.a, \\ add(a, \lambda x. \sum_{i=0}^{n} a_i * x^i) & =_{def} & \lambda x.a + \sum_{i=0}^{n} a_i * x^{i+1} \end{array}$$

für alle $a, a_0, a_1, \ldots, a_n \in \mathbb{R}$. Nach Beispiel 4.6 und Satz 4.14 ist $\mathcal{C}$ eine Konstruktormenge für S. Die Gültigkeit von (1) folgt hier aus der Eindeutigkeit der Darstellung einer Polynomfunktion als Ausdruck der Form $\sum_{i=0}^{n} a_i * x^i$ (siehe z.B. [19], Abschnitt 2.1). ❑

Während Konstruktoren für S den Aufbau der Elemente von S beschreiben, werden diese von *Destruktoren* "zerlegt":

Sei $\mathcal{D}$ eine endliche Menge von Funktionen mit Definitionsbereich S. Diejenigen mit einem Wertebereich der Form S^A nennen wir **Transitionsfunktionen**. Die anderen Elemente von $\mathcal{D}$ heißen **Attribute**. In der objektorientierten Programmierung (OOP) bezeichnet S die Menge der möglichen **Zustände** eines Objekts. Transitionsfunktionen werden dort auch **Methoden** genannt. Attribute bezeichnen zustandsabhängigen, aber kontextunabhängige Eigenschaften eines Objekts wie seine Farbe, Größe, etc., und können durch die Anwendung einer Methode verändert werden. Die Gültigkeit in Kapitel 7 behandelter modallogischer Formeln hängt von einer gegebenen *Kripke-Struktur* ab, die aus einer (nichtdeterministischen) Transitionsfunktion und einer Attributfunktion besteht, die Zuständen keine Attributwerte, sondern Aussagen wie "Die Farbe ist grün" zuordnet.

$\mathcal{D}$ induziert folgende Schrittfunktion, deren größter Fixpunkt die Diagonale von S liefern soll: Für alle $R \subseteq S^2$,

$$\begin{array}{rcl} G_{\mathcal{D}} : \mathcal{P}(S^2) & \to & \mathcal{P}(S^2) \\ R & \mapsto & \{(s, s') \in S^2 \mid \left\{ \begin{array}{l} \forall\, d : S \to S^A \in \mathcal{D},\ a \in A : (d(s)(a), d(s')(a)) \in R, \\ \text{für alle Attribute } d : S \to B \in \mathcal{D} : d(s) = d(s') \end{array} \right\}. \end{array}$$

$\mathcal{D} = \{d_i : S \to B_i \mid 1 \leq i \leq k\}$ heißt **Destruktormenge für** S, falls die Produktextension

$$\langle d_1, \ldots, d_k \rangle : S \to B_1 \times \ldots \times B_k$$

bijektiv ist (siehe Abschnitt 3.2).

Destruktoren tauchen in der objektorientierten Programmierung (OOP) auf. S bezeichnet dort die Menge der möglichen **Zustände** eines Objekts. Destruktoren mit Wertebereich S nennt man **Methoden** oder **Transitionsfunktionen**, Demgegenüber werden Destruktoren mit einem Wertebereich $C \neq S$ als Objekt-**Attribute** bezeichnet.

Satz DESTR $\mathcal{D}$ ist genau dann eine Destruktormenge für S, wenn für alle $s, s' \in S$ Folgendes gilt:

$$\forall\, d : S \to C \in \mathcal{D} : d(s) = d(s') \quad \Rightarrow \quad s = s'. \tag{2}$$

Beweis. Siehe [34], Kapitel 2. Man benötigt hier die aus der ☞ **Kategorientheorie** bekannte axiomatische Definition eines Produkts, aus der folgt, dass auch alle zu einem kartesischen Produkt isomorphen Mengen Produkte sind. Danach ist $\mathcal{D}$ genau dann eine Destruktormenge für S, wenn $\prod_{d \in \mathcal{D}} ran(d)$ ein Produkt ist – mit den Destruktoren als Projektionen (siehe Abschnitt 3.3). ❑

(2) gilt offenbar genau dann, wenn die Diagonale von S ein Fixpunkt von $G_{\mathcal{D}}$ ist.

Der strukturellen oder $F_{\mathcal{C}}$-Induktion als Beweismethode für Eigenschaften von $S = lfp(F_{\mathcal{C}})$ entspricht die $G_{\mathcal{D}}$-Coinduktion, die im Fall $\Delta_S = gfp(G_{\mathcal{D}})$ dem Beweis der Gültigkeit von Gleichungen zwischen Elementen von S dient. Die Diagonale von S wird dann auch **Verhaltensäquivalenz** genannt, weil aus $\Delta_S = gfp(G_{\mathcal{D}})$ (2) folgt, also die Gleichheit zweier "Zustände" s und s' allein von deren "Verhalten" unter $\mathcal{D}$ bestimmt wird. Verhaltensäquivalenzen spielen deshalb in der (mit Zustandsänderungen befassten) Modallogik eine wichtige Rolle (siehe Abschnitt 7.3).

Beispiel 4.18 (Ströme; siehe Abschnitt 4.5) Sei $S = \mathbb{R}^{\mathbb{N}}$ und $\mathcal{D} = \{head : S \to \mathbb{R},\ tail : S \to \mathbb{R}^{\mathbb{N}}\}$ mit

$$\begin{aligned} head(s) &=_{def} s(0), \\ tail(s) &=_{def} \lambda n.s(n+1). \end{aligned}$$

Wir zeigen (2) durch Kontraposition: Sei $s, s' \in S$ mit $s \neq s'$. Dann gibt es $n \in \mathbb{N}$ mit $s(n) \neq s'(n)$. Ist $n = 0$, dann erhalten wir

$$head(s) = s(0) = s(n) \neq s'(n) = s'(0) = head(s').$$

Ist $n > 0$, dann gilt

$$tail(s)(n-1) = (\lambda m.s(m+1))(n-1) = s(n) \neq s'(n) = (\lambda m.s'(m+1))(n-1) = tail(s')(n-1),$$

also $tail(s) \neq tail(s')$. Folglich ist $\mathcal{D}$ eine Destruktormenge für S. ❑

Beispiel 4.19 (unendliche binäre Bäume) Sei $S = \mathbb{R}^{2^*}$ und $\mathcal{D} = \{root : S \to \mathbb{R},\ left, right : S \to S\}$ mit

$$\begin{aligned} root(t) &=_{def} t(\epsilon), \\ left(t) &=_{def} \lambda w.t(0w), \\ right(t) &=_{def} \lambda w.t(1w). \end{aligned}$$

Wir zeigen (2) durch Kontraposition: Sei $t, t' \in S$ mit $t \neq t'$. Dann gibt es $w \in 2^*$ mit $t(w) \neq t'(w)$. Ist $w = \epsilon$, dann erhalten wir

$$root(t) = t(\epsilon) \neq t'(\epsilon) = root(t').$$

Ist $w = 0w'$ für ein $w' \in 2^*$, dann gilt

$$left(t)(w') = (\lambda v.t(0v))(w') = s(w) \neq t'(w) = (\lambda v.t'(0v))(w) = left(t')(w'),$$

also $left(t) \neq left(t')$. Ist $w = 1w'$ für ein $w' \in 2^*$, dann gilt

$$right(t)(w') = (\lambda v.t(1v))(w') = t(w) \neq t'(w) = (\lambda v.t'(1v))(w) = right(t')(w'),$$

also $right(t) \neq right(t')$. Folglich ist $\mathcal{D}$ eine Destruktormenge für S. ❑

Beispiel 4.20 Sei $S = C^{B^*}$ und $\mathcal{D} = \{\beta : S \to C,\ \delta : S \to S^B\}$ mit

$$\begin{aligned} \beta(s) &=_{def} s(\epsilon), \\ \delta(s) &=_{def} \lambda b.\lambda w.s(bw). \end{aligned}$$

Wir zeigen (1) durch Kontraposition: Sei $s, s' \in S$ mit $s \neq s'$. Dann gibt es $w \in B^*$ mit $s(w) \neq s'(w)$. Ist $w = \epsilon$, dann erhalten wir

$$\beta(s) = s(\epsilon) \neq s'(\epsilon) = \beta(s').$$

Ist $w = bw'$ für ein $b \in B$ und ein $w' \in 2^*$, dann gilt

$$\delta(s)(b)(w') = (\lambda v.s(bv))(w') = s(w) \neq s'(w) = (\lambda v.s'(bv))(w) = \delta(s')(b)(w'),$$

also $left(s) \neq left(s')$. Folglich ist $\mathcal{D}$ eine Destruktormenge für S.

$\mathcal{D}$ repräsentiert übrigens den finalen Moore-Automaten mit Eingabemenge B und Ausgabemenge C (siehe Abschnitt 7.6). ❑

4.7 (Co)Induktiv definierte Funktionen

Induktiv definierte Funktionen

Sei $\mathcal{C}$ eine Konstruktormenge für S.

Eine Funktion $f : S \times B \to C$ heißt **induktiv definiert bzgl.** $\mathcal{C}$, wenn für alle $c : A \times S^n \to S \in \mathcal{C}$ eine Funktion $h_c : A \times S^n \times B \times C^n \to C$ existiert derart, dass für alle $a \in A$, $s_1, \ldots, s_n \in S$ und $b \in B$

$$f(c(a, s_1, \ldots, s_n), b) = h_c(a, s_1, \ldots, s_n, b, f(s_1, b), \ldots, f(s_n, b)) \tag{1}$$

genau eine Lösung in der Funktionsvariablen f hat.

Im Fall $\mathcal{C} = \{0, (+1)\}$ (siehe Beispiel 4.15) nennt man eine bzgl. $\mathcal{C}$ induktiv definierte Funktion auch **primitiv-rekursiv**.

Satz 4.21 Sei $lfp(F_\mathcal{C}) = S$ (siehe Abschnitt 4.6). Es gibt höchstens eine Lösung $f : S \times B \to C$ von (1).

Beweis durch $F_\mathcal{C}$-Induktion. Seien $f, g : S \times B \to C$ zwei Lösungen von (1) und T die Menge aller $s \in S$ mit $f(s, b) = g(s, b)$ für alle $b \in B$. Nach Satz 4.1 (iii) ist zu zeigen, dass T $F_\mathcal{C}$-abgeschlossen ist.

Sei also $s \in F_\mathcal{C}(T)$. Dann gibt es $c : A \times S^n \to S \in \mathcal{C}$, $a \in A$ und $s_1, \ldots, s_n \in T$ und $s = c(a, s_1, \ldots, s_n)$. Demnach gilt $f(s_i, b) = g(s_i, b)$ für alle $1 \leq i \leq n$ und $b \in B$. Daraus folgt

$$\begin{aligned} &f(s, b) = f(c(a, s_1, \ldots, s_n), b) = h_c(a, s_1, \ldots, s_n, b, f(s_1, b), \ldots, f(s_n, b)) \\ &= h_c(a, s_1, \ldots, s_n, b, g(s_1, b), \ldots, g(s_n, b), s_1, \ldots, s_n) = g(c(a, s_1, \ldots, s_n), b) = g(s, b), \end{aligned}$$

also $s \in T$, d.h. f und g sind gleich. ❑

Es gibt zahlreiche Verallgemeinerungen von Gleichung (1), die ebenfalls eindeutig lösbar sind. Dazu gehören insbesondere Schemata zur gleichzeitigen, wechselseitig-rekursiven Definition mehrerer Funktionen auf Produkten mit mehreren induktiv definierten Faktoren.

Beispiel 4.22 Sei $S = A^*$. Nach Satz 4.21 bilden die Gleichungen

$$\begin{aligned} length(\epsilon) &= 0, \\ length(cons(a, w)) &= length(w) + 1, \\ conc(\epsilon, w) &= w, \\ conc(cons(a, v), w) &= cons(a, conc(v, w)) \end{aligned}$$

eine induktive Definition der Funktionen

$$\begin{array}{rcl} length : A^* & \to & \mathbb{N} \\ \epsilon & \mapsto & 0 \\ (a_1, \ldots, a_n) & \mapsto & n \end{array}$$

und

$$\begin{array}{rcl} conc : A^* \times A^* & \to & A^* \\ (\epsilon, w) & \mapsto & w \\ (w, \epsilon) & \mapsto & w \\ ((a_1, \ldots, a_m), (b_1, \ldots, b_n)) & \mapsto & (a_1, \ldots, a_m, b_1, \ldots, b_n) \end{array}$$

bzgl. der Konstruktormenge $\mathcal{C}$ von Beispiel 4.16. *length* ordnet einer Liste ihre Länge zu, *conc* **konkateniert** zwei Listen , d.h. fügt sie zu einer Liste zusammen.

Aufgabe Zeigen Sie, dass *length* und *conc* die obigen Gleichungen erfüllen. ❑

Beispiel 4.23 Sei $S = \mathbb{R}[x]$. Nach Satz 4.21 bilden die Gleichungen

$$\begin{array}{rcl} scalar(const(a), b) & = & const(a, b), \\ scalar(add(a, p), b) & = & add(a * b, scalar(p, b)) \end{array}$$

eine induktive Definition der Funktion

$$\begin{array}{rcl} scalar : \mathbb{R}[x] \times \mathbb{R} & \to & \mathbb{R}[x] \\ (\lambda x. \sum_{i=0}^{n} a_i * x^i, b) & \mapsto & \lambda x. \sum_{i=1}^{n} a_i * b * x^{n-1} \end{array}$$

bzgl. der Konstruktormenge $\mathcal{C}$ von Beispiel 4.17. *scalar* multipliziert ein Polynom mit einer reellen Zahl.

Aufgabe Zeigen Sie, dass *scalar* die obigen Gleichungen erfüllt. ❑

Beispiel 4.24 (siehe Beispiel 4.22) Wir zeigen durch strukturelle Induktion, dass für alle $v, w \in A^*$

$$length(conc(v, w)) = length(v) + length(w)$$

gilt. Wegen $A^* = lfp(F_{list})$ (siehe Beispiel 4.16) gilt diese Gleichung nach Satz 4.1 (iii), falls die Menge der Listen über A, die sie erfüllen, eine $\mathcal{C}_{list}$-Invariante (= $F_{\mathcal{C}_{list}}$-abgeschlossene Teilmenge) von A^* ist, falls also für alle $a \in A$ und $v, w \in A^*$ (1) und (2) gilt:

$$length(conc(\epsilon, w)) = length(\epsilon) + length(w), \tag{1}$$

$$\begin{array}{l} length(conc(v, w)) = length(v) + length(w) \\ \qquad \Rightarrow \; length(conc(cons(a, v), w)) = length(cons(a, v)) + length(w). \end{array} \tag{2}$$

Beweis von (2).

$$length(conc(\epsilon, w)) = length(w) = 0 + length(w) = length(\epsilon) + length(w).$$

Beweis von (3). Es gelte die "Induktionshypothese"

$$length(conc(v, w)) = length(v) + length(w). \tag{4}$$

Wir erhalten

$$\begin{array}{l} length(conc(cons(a, v), w)) = length(cons(a, conc(v, w))) = length(conc(v, w)) + 1 \\ \overset{(4)}{=} length(v) + length(w) + 1 = length(v) + 1 + length(w) = length(cons(a, v)) + length(w). \end{array}$$ ❑

Eine weitere Verallgemeinerung von (1) betrifft den Fall von **wechselseitig induktiv definierten** Komponenten eines Funktionstupels $(f_i : S \to A_i)_{1 \le i \le m}$. Dann benötigt man für alle $1 \le i \le m$ eine Funktion

$$h_{i,c} : A \times S^n \times B \times C^n \to C$$

derart, dass für alle $a \in A$, $b \in B$ und $t_1, \ldots, t_n \in S$ folgende Gleichung gilt:

$$f_i(c(a, t_1, \ldots, t_n), b) = h_{i,c}(a, t_1, \ldots, t_n, b, f_1(t_1, b), \ldots, f_1(t_n, b), \ldots, f_m(t_1, b), \ldots, f_m(t_n, b)). \tag{5}$$

Beispiel 4.25 (Hilbertkurven)

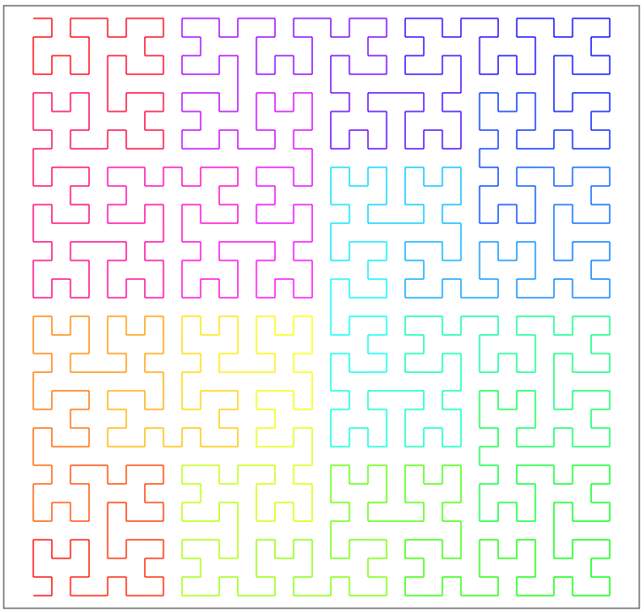
Sei $D = \{east, west, north, south\}$. Das folgende Funktionsquadrupel

$$(go_i : \mathbb{N} \to D^*)_{1 \le i \le 4}$$

berechnet Folgen von Himmelsrichtungen, die ☞ Hilbertkurven repräsentieren:

Wird jedes Element d der Folge $go_i(n)$ als Befehl interpretiert, in Richtung d eine Linie zu ziehen, dann liefert die Ausführung der Befehlsfolge eine Hilbertkurve n-ter Ordnung.

Im Fall $i = 1$ und $n = 5$ entsteht z.B. folgende Kurve:



$(go_1, \ldots, go_4)$ ist wie folgt induktiv definiert: Für alle $0 \le i \le 3$ und $n \in \mathbb{N}$,

$$\begin{aligned}
go_i(0) &= \epsilon, \\
go_1(n+1) &= go_4(n) \cdot east \cdot go_1(n) \cdot south \cdot go_1(n) \cdot west \cdot go_3(n), \\
go_2(n+1) &= go_3(n) \cdot west \cdot go_2(n) \cdot north \cdot go_2(n) \cdot east \cdot go_4(n), \\
go_3(n+1) &= go_2(n) \cdot north \cdot go_3(n) \cdot west \cdot go_3(n) \cdot south \cdot go_1(n), \\
go_4(n+1) &= go_1(n) \cdot south \cdot go_4(n) \cdot east \cdot go_4(n) \cdot north \cdot go_2(n).
\end{aligned}$$

Die Funktion

$$coords : D^* \to (\mathbb{R}^2 \to (\mathbb{R}^2)^*)$$

soll jede Richtungsfolge $s \in D^*$ und jeden Punkt (x, y) in der Ebene in die Folge der Ecken des durch s repräsentierten Linienzuges überführten, wobei jede einzelne Linie die Länge 1 hat.

coords ist wie folgt induktiv definiert: Für alle $s \in D^*$ und $(x,y) \in \mathbb{R}^2$,

$$\begin{aligned} coords(\epsilon) &= \lambda(x,y).(x,y), \\ coords(cons_{east}(s)) &= \lambda(x,y).(x,y)\cdot coords(s)(x+1,y), \\ coords(cons_{west}(s)) &= \lambda(x,y).(x,y)\cdot coords(s)(x-1,y), \\ coords(cons_{north}(s)) &= \lambda(x,y).(x,y)\cdot coords(s)(x,y+1), \\ coords(cons_{south}(s)) &= \lambda(x,y).(x,y)\cdot coords(s)(x,y-1). \quad ❏ \end{aligned}$$

Aufgabe Zeigen Sie, dass jede induktiv definierte Funktion einen induktiv definierten Graphen hat. ❏

Es gibt allgemeinere Schemata für induktive Funktionsdefinitionen als das hier behandelte bis hin zu mehrstelligen Funktionen auf Produkten verschiedener (induktiv definierter) Mengen.

Coinduktiv definierte Funktionen

Sei $\mathcal{D}$ eine Destruktormenge für S (siehe Abschnitt 4.6).

Eine Funktion $f : A \times S^n \to S$ heißt **coinduktiv definiert bzgl.** $\mathcal{D}$, wenn

- für alle Transitionsfunktionen $d : S \to S^B \in \mathcal{D}$ eine Funktion $h_d : A \times B \times S^n \to A \times S^n$ existiert derart, dass für alle $a \in A$ und $s_1, \ldots, s_n \in S$

$$d(f(a, s_1, \ldots, s_n)) = \lambda b.f(h_d(a, b, s_1, \ldots, s_n)) \tag{6}$$

  genau eine Lösung in (der Funktionsvariablen) f hat,

- für alle Attribute $d : S \to C \in \mathcal{D}$ eine Funktion $h_d : A \times S^n \to C$ existiert derart, dass für alle $a \in A$ und $s_1, \ldots, s_n \in S$

$$d(f(a, s_1, \ldots, s_n)) = h_d(a, s_1, \ldots, s_n) \tag{7}$$

  genau eine Lösung in f hat.

Ebenso wie (1) kann auch (6) verallgemeinert werden, z.B. um die simultane, wechselseitig coinduktive Definitionen mehrerer Funktionen zu erlauben.

Satz 4.26 Sei $gfp(G_\mathcal{D}) = \Delta_S$ (siehe Abschnitt 4.6). Es gibt höchstens eine Lösung $f : A \times S^n \to S$ von (6) und (7).

Beweis durch $G_\mathcal{D}$-Coinduktion. Seien $f, g : A \times S^n \to S$ zwei Lösungen von (6) und (7) und R die Menge aller Paare $(f(w), g(w))$ mit $w \in A \times S^n$. Wir zeigen, dass R $G_\mathcal{D}$-dicht und damit nach Satz 4.1 (ii) in $gfp(G_\mathcal{D})$ enthalten ist. Nach Voraussetzung sind daher f und g gleich.

Sei also $a \in A$, $s_1, \ldots, s_n \in S$, $d : S \to S^B \in \mathcal{D}$ und $b \in B$. Dann gilt

$$(d(f(a, s_1, \ldots, s_n))(b), d(g(a, s_1, \ldots, s_n))(b)) = (f(h_d(a, b, s_1, \ldots, s_n)), g(h_d(a, b, s_1, \ldots, s_n))) \in R.$$

Sei $d : S \to C$ eine Attribut von $\mathcal{D}$. Dann erhalten wir

$$d(f(a, s_1, \ldots, s_n)) = h_d(a, s_1, \ldots, s_n) = d(g(a, s_1, \ldots, s_n)),$$

Folglich ist R $G_\mathcal{D}$-dicht. ❏

Analog zu (1) gibt es zahlreiche Verallgemeinerungen von (6) und (7), die ebenfalls eindeutig lösbar sind.

Beispiel 4.27 Sei $S = \mathbb{R}^\mathbb{N}$ und $\mathcal{D}$ die Destruktormenge für von Beispiel 4.18. Nach Satz DESTR gilt daher (2), d.h. die Diagonale von S ist ein Fixpunkt von $G_\mathcal{D}$. Um Satz 4.26 anwenden zu können, müssen wir zunächst zeigen, dass sie der *größte* Fixpunkt von $G_\mathcal{D}$ ist. Da Δ_S $G_\mathcal{D}$-dicht ist und nach Satz 4.1 (ii) $gfp(G_\mathcal{D})$ die Vereinigung aller

$G_{\mathcal{D}}$-dichten Teilmengen von S^2 ist, gilt $\Delta_S \subseteq gfp(G_{\mathcal{D}})$. Es bleibt also zu zeigen, dass alle $G_{\mathcal{D}}$-dichten Teilmengen von S^2 Teilmengen von $gfp(G_{\mathcal{D}})$ sind.

Sei also $R \subseteq G_{\mathcal{D}}(R)$, $(s, s') \in R$ und für alle $n \in \mathbb{N}$ gelte

$$(tail^n(s), tail^n(s')) \in R. \tag{8}$$

Daraus folgt $(tail^n(s), tail^n(s')) \in G_{\mathcal{D}}(R)$, also

$$s(n) = tail^n(s)(0) = head(tail^n(s)) = head(tail^n(s')) = tail^n(s')(0) = s'(n)$$

für alle $n \in \mathbb{N}$, d.h. $s = s'$.

(8) erhält man durch Induktion über n: Im Fall $n = 0$ gilt (8) wegen $tail^0(s) = s$, $tail^0(s') = s'$ und $(s, s') \in R$. Im Fall $n > 0$ gilt $(tail^{n-1}(s), tail^{n-1}(s')) \in R \subseteq G_{\mathcal{D}}(R)$ nach Induktionsvoraussetzung, also auch (8) nach Definition von $G_{\mathcal{D}}(R)$:

$$(tail^n(s), tail^n(s')) = (tail(tail^{n-1}(s)), tail(tail^{n-1}(s'))) \in R.$$

Damit ist $\Delta_S = gfp(G_{\mathcal{D}})$ gezeigt, so dass nach Satz 4.26 die Gleichungen

$$\begin{aligned}
head(nats(n)) &= n, \\
tail(nats(n)) &= nats(n+1), \\
head(cons(x,s)) &= x, \\
tail(cons(x,s)) &= s, \\
head(s \oplus s') &= head(s) + head(s'), \\
tail(s \oplus s') &= tail(s) \oplus tail(s'), \\
head(zip(s,s')) &= head(s), \\
tail(zip(s,s')) &= zip(s', tail(s))
\end{aligned}$$

eine coinduktive Definition der Funktionen

$$\begin{aligned}
nats : \mathbb{N} &\to \mathbb{R}^{\mathbb{N}} \\
n &\mapsto \lambda i.n + i \\
cons : \mathbb{R} \times \mathbb{R}^{\mathbb{N}} &\to \mathbb{R}^{\mathbb{N}} \\
(r,s) &\mapsto \lambda n. \begin{cases} r & \text{falls } n = 0 \\ s(n-1) & \text{falls } n > 0 \end{cases} \\
\oplus : \mathbb{R}^{\mathbb{N}} \times \mathbb{R}^{\mathbb{N}} &\to \mathbb{R}^{\mathbb{N}} \\
(s,s') &\mapsto \lambda n.s(n) + s'(n) \\
zip : \mathbb{R}^{\mathbb{N}} \times \mathbb{R}^{\mathbb{N}} &\to \mathbb{R}^{\mathbb{N}} \\
(s,s') &\mapsto \lambda n. \begin{cases} s(n/2) & \text{falls } n \text{ gerade} \\ s'(n/2+1) & \text{falls } n \text{ ungerade} \end{cases}
\end{aligned}$$

bzgl. $\mathcal{D}$ bilden. $nats(n)$ erzeugt den Strom aller natürlicher Zahlen ab n. $\oplus$ addiert zwei Ströme reeller Zahlen elementweise, zip mischt zwei Ströme s und s' zu einem Strom, an dessen geraden Positionen die Elemente von s und an dessen ungeraden Positionen die Elemente von s' stehen.

Aufgabe Zeigen Sie, dass $nats$, $cons$, $\oplus$ und zip die obigen Gleichungen erfüllen.

$\oplus$ ist kommutativ: Wegen $\Delta_S = gfp(G_{\mathcal{D}})$ genügt es nach Satz 4.1 (iv) zu zeigen, dass die Relation

$$R =_{def} \{(s \oplus s', s' \oplus s) \mid s, s' \in \mathbb{R}^{\mathbb{N}}\}$$

$G_{\mathcal{D}}$-dicht ist. Sei also $s, s' \in \mathbb{R}^{\mathbb{N}}$. Dann gilt

$$\begin{aligned}
&head(s \oplus s') = head(s) + head(s') = head(s') + head(s) = head(s' \oplus s), \\
&(tail(s \oplus s'), tail(s' \oplus s)) = (tail(s) \oplus tail(s'), tail(s') \oplus tail(s)) \in R,
\end{aligned}$$

d.h. $(s \oplus s', s' \oplus s) \in G_{\mathcal{D}}(R)$. ❑

Beispiel 4.28 Sei $S = \mathbb{R}^{2^*}$ und $\mathcal{D}$ die Destruktormenge für von Beispiel 4.19. Nach Satz DESTR gilt daher (2), d.h. die Diagonale von S ist ein Fixpunkt von $G_{\mathcal{D}}$. Um Satz 4.26 anwenden zu können, müssen wir zunächst zeigen, dass sie der *größte* Fixpunkt von $G_{\mathcal{D}}$ ist. Analog zu Beispiel 4.17 bleibt zu zeigen, dass alle $G_{\mathcal{D}}$-dichten Teilmengen von S^2 Teilmengen von $gfp(G_{\mathcal{D}})$ sind.

Sei also $R \subseteq G_{\mathcal{D}}(R)$, $(t, t') \in R$ und für alle $w \in 2^*$ gelte

$$(f(t,w), f(t',w)) \in R, \tag{9}$$

wobei $f : 2^* \times S \to S$ wie folgt induktiv definiert ist: $f(\epsilon, t) = t$ und für alle $w \in 2^*$, $f(t, w0) = left(f(t,w))$ und $f(t, w1) = right(f(t,w))$. Aus (9) folgt $(f(t,w), f(t',w)) \in G_{\mathcal{D}}(R)$, also

$$t(w) = f(t,w)(\epsilon) = root(f(t,w)) = root(f(t',w,)) = f(t,w)(\epsilon) = t'(w)$$

für alle $w \in 2^*$, d.h. $t = t'$.

(9) erhält man durch Induktion über die Länge von w: Im Fall $w = \epsilon$ gilt (9) wegen $f(\epsilon, t) = t$, $f(\epsilon, t') = t'$ und $(t, t') \in R$. Im Fall der Existenz von $v \in 2^*$ und $b \in 2$ mit $w = vb$ gilt $(f(t,v), f(t',v)) \in R \subseteq G_{\mathcal{D}}(R)$ nach Induktionsvoraussetzung, also auch (9) nach Definition von $G_{\mathcal{D}}(R)$:

$$(f(t,w), f(t',w)) = (f(t,v0), f(t',v0)) = (left(f(t,v)), left(f(t',v))) \in R$$

bzw.

$$(f(t,w), f(t',w)) = (f(t,v1), f(t',v1)) = (left(f(t,v)), right(f(t',v))) \in R.$$

Damit ist $\Delta_S = gfp(G_{\mathcal{D}})$ gezeigt, so dass nach Satz 4.26 die Gleichungen

$$\begin{aligned}
root(bnats(n)) &= n,\\
left(bnats(n)) &= nats(2*n),\\
right(bnats(n)) &= nats(2*n+1),\\
root(fork(x,t,t')) &= x,\\
left(fork(x,t,t')) &= t,\\
right(fork(x,t,t')) &= t',\\
root(t \oplus t') &= root(t) + root(t'),\\
left(t \oplus t') &= left(t) \oplus left(t'),\\
right(t \oplus t') &= right(t) \oplus right(t')
\end{aligned}$$

eine coinduktive Definition der Funktionen

$$\begin{aligned}
bnats : \mathbb{N} &\to \mathbb{R}^{2^*}\\
n &\mapsto \lambda w. \begin{cases} n & \text{falls } w = \epsilon \\ 2*n & \text{falls } \exists v \in \mathbb{R}^{2^*} : 0v = w \\ 2*n+1 & \text{falls } \exists v \in \mathbb{R}^{2^*} : 1v = w \end{cases}\\
fork : \mathbb{R} \times \mathbb{R}^{2^*} \times \mathbb{R}^{2^*} &\to \mathbb{R}^{2^*}\\
(r,t,t') &\mapsto \lambda w. \begin{cases} r & \text{falls } w = \epsilon \\ t(v) & \text{falls } \exists v \in \mathbb{R}^{2^*} : 0v = w \\ t'(v) & \text{falls } \exists v \in \mathbb{R}^{2^*} : 1v = w \end{cases}\\
\oplus : \mathbb{R}^{2^*} \times \mathbb{R}^{2^*} &\to \mathbb{R}^{2^*}\\
(t,t') &\mapsto \lambda w. t(w) + t'(w)
\end{aligned}$$

bzgl. $\mathcal{D}$ bilden. $bnats(n)$ erzeugt einen binären Baum mit allen natürlichen Zahlen ab n. Durchläuft man seine Knoten in ☞ Heapordnung, dann erhält man den Strom $nats(n)$ (siehe Beispiel 4.27). $\oplus$ addiert zwei binäre Bäume mit reellwertigen Knoten knotenweise.

Aufgabe Zeigen Sie, dass *bnats*, *fork* und $\oplus$ die obigen Gleichungen erfüllen.

Analog zur Kommutativität der Addition von Strömen (siehe Beispiel 4.27) zeigen wir die folgenden Gleichungen durch *inkrementelle* $G_{\mathcal{D}}$-Coinduktion gemäß Satz 4.1 (iv) und der darauffolgenden Bemerkung: Für alle $n \in \mathbb{N}$,

$$bnats(n) = fork(n, bnats(2*n), bnats(2*n+1)). \tag{10}$$

Für alle $r, s \in \mathbb{R}$ und $t, t', u, u' \in \mathbb{R}^{2^*}$,

$$fork(r,t,t') \oplus fork(s,u,u') = fork(r+s, t \oplus u, t' \oplus u'). \tag{11}$$

Beweis von (10). Wegen $\Delta_S = gfp(G_{\mathcal{D}})$ genügt es zu zeigen, dass die Relation

$$R =_{def} \{(bnats(n), fork(n, bnats(2*n), bnats(2*n+1))) \mid n \in \mathbb{N}\}$$

in einer $G_{\mathcal{D}}$-dichten Teilmenge von S^2 enthalten ist. Sei also $n \in \mathbb{N}$. Dann gilt

$$\begin{aligned}
&root(bnats(n)) = n = root(fork(n, bnats(2*n), bnats(2*n+1))),\\
&(left(bnats(n)), left(fork(n, bnats(2*n), bnats(2*n+1))))\\
&\quad = (\lambda w.bnats(n)(0w), \lambda w.fork(n, bnats(2*n), bnats(2*n+1))(0w))\\
&\quad = (\lambda w.bnats(2*n)(w), \lambda w.bnats(2*n)(w)) \in \Delta_S,\\
&(right(bnats(n)), right(fork(n, bnats(2*n), bnats(2*n+1))))\\
&\quad = (\lambda w.bnats(n)(1w), \lambda w.fork(n, bnats(2*n), bnats(2*n+1))(1w))\\
&\quad = (\lambda w.bnats(2*n+1)(w), \lambda w.bnats(2*n+1)(w)) \in \Delta_S,
\end{aligned}$$

d.h. $(bnats(n), fork(n, bnats(2*n), bnats(2*n+1))) \in G_{\mathcal{D}}(R \cup \Delta_S)$. Daher und wegen $\Delta_S \subseteq G_{\mathcal{D}}(R \cup \Delta_S)$ ist $R \cup \Delta_S$ $G_{\mathcal{D}}$-dicht.

Beweis von (11). Wegen $\Delta_S = gfp(G_{\mathcal{D}})$ genügt es zu zeigen, dass die Relation

$$R =_{def} \{(fork(r,t,t') \oplus fork(s,u,u'), fork(r+s, t \oplus u, t' \oplus u')) \mid r, s \in \mathbb{R},\ t, t', u, u' \in \mathbb{R}^{2^*}\}$$

in einer $G_{\mathcal{D}}$-dichten Teilmenge von S^2 enthalten ist. Sei also $r, s \in \mathbb{R}$ und $t, t', u, u' \in \mathbb{R}^{2^*}$. Dann gilt

$$\begin{aligned}
&root(fork(r,t,t') \oplus fork(s,u,u')) = root(\lambda w.fork(r,t,t')(w) + fork(s,u,u')(w)) = fork(r,t,t')(\epsilon) + fork(s,u,u')(\epsilon)\\
&\quad = r + s = root(fork(r+s, t \oplus u, t' \oplus u')),\\
&(left(fork(r,t,t') \oplus fork(s,u,u')), left(fork(r+s, t \oplus u, t' \oplus u'))) = (left(\lambda w.fork(r,t,t')(w) + fork(s,u,u')(w)), t \oplus u)\\
&\quad = (\lambda v.(\lambda w.fork(r,t,t')(w) + fork(s,u,u')(w))(0v), t \oplus u) = (\lambda v.fork(r,t,t')(0v) + fork(s,u,u')(0v), t \oplus u)\\
&\quad = (\lambda v.left(fork(r,t,t'))(v) + left(fork(s,u,u'))(v), t \oplus u) = (\lambda v.t(v) + u(v), t \oplus u) = (t \oplus u, t \oplus u) \in \Delta_S,\\
&(right(fork(r,t,t') \oplus fork(s,u,u')), right(fork(r+s, t \oplus u, t' \oplus u')))\\
&\quad = (right(\lambda w.fork(r,t,t')(w) + fork(s,u,u')(w)), t \oplus u) = (\lambda v.(\lambda w.fork(r,t,t')(w) + fork(s,u,u')(w))(1v), t' \oplus u')\\
&\quad = (\lambda v.fork(r,t,t')(1v) + fork(s,u,u')(1v), t \oplus u) = (\lambda v.right(fork(r,t,t'))(v) + right(fork(s,u,u'))(v), t' \oplus u')\\
&\quad = (\lambda v.t'(v) + u'(v), t' \oplus u') = (t' \oplus u', t' \oplus u') \in \Delta_S,
\end{aligned}$$

d.h. $(fork(r,t,t') \oplus fork(s,u,u'), fork(r+s, t \oplus u, t' \oplus u')) \in G_{\mathcal{D}}(R \cup \Delta_S)$. Daher und wegen $\Delta_S \subseteq G_{\mathcal{D}}(R \cup \Delta_S)$ ist $R \cup \Delta_S$ $G_{\mathcal{D}}$-dicht.

(11) ist übrigens auch Teil einer *induktiven* Definition von $\oplus$ auf *endlichen* Binärbäumen. In diesem Kontext wäre *fork* ein Konstruktor analog dem Konstruktor *cons* für endliche Listen (siehe Beispiel 4.16). Anstelle von *nil* gäbe es einen Konstruktor *empty* – als Repräsentation des "leeren" Baums. Die Definition von $\oplus$ enthielte neben (11) die Gleichungen $t \oplus empty = t$ und $empty \oplus t = t$. ❏

4.8 Highlights

$F : \mathcal{P}(S) \to \mathcal{P}(S)$ monoton: Für alle $T, T' \subseteq S$ gilt: $T \subseteq T \Rightarrow F(T) \subseteq F(T')$.

$T \subseteq S$ F-abgeschlossen: $F(T) \subseteq T$.

$T \subseteq S$ F-dicht: $T \subseteq F(T)$.

$$\begin{aligned} lfp(F) &=_{def} \bigcap\{T \subseteq S \mid F(T) \subseteq T\}. \\ gfp(F) &=_{def} \bigcup\{T \subseteq S \mid T \subseteq F(T)\}. \end{aligned}$$

Ist $F : \mathcal{P}(S) \to \mathcal{P}(S)$ monoton, dann ist $lfp(F)$ der kleinste und $gfp(F)$ der größte Fixpunkt von F.

F-Induktion

$$\frac{lfp(F) \subseteq T_\varphi}{F(T_\varphi) \subseteq T_\varphi} \Uparrow$$

Alle Elemente von $lfp(F)$ haben eine gegebene Eigenschaft φ, wenn die Menge T_φ aller Elemente von S, die φ erfüllen, F-abgeschlossen ist. Begründung: Jede F-abgeschlossene Teilmenge von S enthält $lfp(F)$.

Inkrementelle F-Induktion

$$\frac{lfp(F) \subseteq T}{\exists\, T' \subseteq T : F(T') \subseteq T'} \Updownarrow$$

F-Coinduktion

$$\frac{T_\varphi \subseteq gfp(F)}{T_\varphi \subseteq F(T_\varphi)} \Uparrow$$

Alle Elemente einer gegebenen F-dichten Teilmenge von S haben eine gegebene Eigenschaft φ, wenn $gfp(F)$ mit der Menge T_φ aller Elemente von S, die φ erfüllen, übereinstimmt. Begründung: $gfp(F)$ enthält jede F-dichte Teilmenge von S.

Inkrementelle F-Coinduktion

$$\frac{T \subseteq gfp(F)}{\exists\, T' \supseteq T : T' \subseteq F(T')} \Updownarrow$$

Oberer Kleene-Abschluss von F: $F^\infty =_{def} \bigcup_{n\in\mathbb{N}} F^n(\emptyset)$.

Unterer Kleene-Abschluss von F: $F_\infty =_{def} \bigcap_{n\in\mathbb{N}} F^n(S)$.

$F(F^\infty) \subseteq F^\infty \;\Rightarrow\; lfp(F) = F^\infty$.

$F_\infty \subseteq F(F_\infty) \;\Rightarrow\; gfp(F) = F_\infty$.

Schrittfunktion für $\mathbb{N}$:

$$\begin{aligned} F_{nat} : \mathcal{P}(\mathbb{N}) &\to \mathcal{P}(\mathbb{N}) \\ T &\mapsto \{0\} \cup \{a+1 \mid a \in T\} \end{aligned}$$

Induktion über $n \in \mathbb{N}$ = F_{nat}-Induktion.

Schrittfunktion für A bzgl. wohlfundierter Relation $R \subseteq A^2$:

$$\begin{aligned} F_R : \mathcal{P}(A) &\to \mathcal{P}(A) \\ T &\mapsto \{a \in A \mid \forall\, b \in A : (b,a) \in R \Rightarrow b \in T\} \end{aligned}$$

Noethersche Induktion bzgl. R = F_R-Induktion.

Schrittfunktion für $\vdash_K$:

$$\begin{aligned} F_K : \mathcal{P}(\mathcal{P}(Fo) \times Fo) &\to \mathcal{P}(\mathcal{P}(Fo) \times Fo) \\ T &\mapsto \{(\Phi,\varphi) \mid \tfrac{\Phi_1 \vdash \varphi_1, \ldots, \Phi_k \vdash \varphi_k}{\Phi \vdash \varphi} \in K,\ (\Phi_1,\varphi_1), \ldots, (\Phi_k\, \varphi_k) \in T\} \end{aligned}$$

Eine Funktionsmenge $\mathcal{C} = \{c_i : A_i \to S \mid 1 \leq i \leq k\}$ heißt Konstruktormenge für S, falls die Summenextension $[c_1, \ldots, c_k] : A_1 + \cdots + A_k \to S$ bijektiv ist.

Eine Funktionsmenge $\mathcal{D} = \{d_i : S \to B_i \mid 1 \leq i \leq k\}$ heißt Destruktormenge für S, falls die Produktextension $\langle d_1, \ldots, d_k \rangle : S \to B_1 \times \ldots \times B_k$ bijektiv ist.

Schrittfunktion $F_\mathcal{C} : \mathcal{P}(S) \to \mathcal{P}(S)$:

Für alle $T \subseteq S$,

$$F_\mathcal{C}(T) = \{c(w) \mid c : A \times S^n \in \mathcal{C},\ w \in A \times T^n\}.$$

Strukturelle Induktion über $\mathcal{C}$ = $F_\mathcal{C}$-Induktion

$\mathcal{C}$-Invariante = $F_\mathcal{C}$-abgeschlossene Menge

Schrittfunktion $G_\mathcal{D} : \mathcal{P}(S^2) \to \mathcal{P}(S^2)$:

Für alle $R \subseteq S^2$,

$$G_\mathcal{D}(R) = \{(s, s') \in S^2 \mid \left\{ \begin{array}{l} \forall\, d : S \to S^A \in \mathcal{D},\ a \in A : (d(s)(a), d(s')(a)) \in R, \\ \text{für alle Attribute } d : S \to B \in \mathcal{D} : d(s) = d(s') \end{array} \right\}.$$

Bzgl. $\mathcal{C}$ induktiv definierte Funktion $f : S \times B \to C$:

Für alle $c : A \times S^n \to S \in \mathcal{C}$ existiert h_c derart, dass für alle $a \in A$ und $t_1, \ldots, t_n \in S$

$$f(c(a, s_1, \ldots, s_n), b) = h_c(a, s_1, \ldots, s_n, b, f(s_1, b), \ldots, f(s_n, b))$$

genau eine Lösung in f hat.

Bzgl. $\mathcal{D}$ coinduktiv definierte Funktion $f : A \times S^n \to S$:

- Für alle Transitionsfunktionen $d : S \to S^B \in \mathcal{D}$ existiert h_d derart, dass für alle $a \in A$ und $s_1, \ldots, s_n \in S$

  $$d(f(a, s_1, \ldots, s_n)) = \lambda b. f(h_d(a, b, s_1, \ldots, s_n))$$

  genau eine Lösung in f hat.
- Für alle Attribute $d : S \to C \in \mathcal{D}$ existiert h_d derart, dass für alle $a \in A$ und $s_1, \ldots, s_n \in S$

  $$d(f(a, s_1, \ldots, s_n)) = h_d(a, s_1, \ldots, s_n)$$

  genau eine Lösung in f hat.

5 Gleichungslogik (equational logic)

Im vorangegangenen Kapitel war häufig von Ausdrücken im Sinne zusammengesetzter Funktionsaufrufe die Rede. Da diese Sprechweise den Unterschied zwischen Syntax und Semantik (siehe Kapitel 2) verschwimmen lässt, führen wir in diesem Kapitel mathematische Begriffe ein, die eine klare Trennlinie ziehen zwischen syntaktischen Ausdrücken (*Termen*), die nichts anderes als Symbolfolgen sind, und deren variierender Bedeutung als Elemente strukturierter Mengen (*Algebren*).

5.1 Signaturen, Terme und Algebren

Eine **Signatur** Σ ist eine Menge von – **Operationen** genannten – Funktions*symbolen* f mit Stelligkeit $n \in \mathbb{N}$. Wir schreiben $f : n \to 1$ für eine Operation mit Stelligkeit n.

Sei X eine Menge von **Variablen** und S die Menge aller Folgen von runden Klammern, Kommas und Elementen der Menge $\Sigma \cup X$.

Die Menge $T_\Sigma(X)$ der **Σ-Terme über** X ist induktiv definiert. Ihre Schrittfunktion lautet wie folgt:

$$\begin{aligned} F_\Sigma : \mathcal{P}(S) &\to \mathcal{P}(S) \\ T &\mapsto X \cup \{ft \mid f : n \to 1 \in \Sigma,\ t \in T^n\} \end{aligned}$$

Jedes *Funktionssymbol* $f : n \to 1 \in \Sigma$ induziert eine *Funktion* $f^{T_\Sigma(X)} : S^n \to S$: Für alle $t \in S$, $f^{T_\Sigma(X)}(t) =_{def} ft$.

Zusammen mit den (als nullstellige Funktionen betrachteten) Variablen von X bilden diese Funktionen eine Konstruktormenge für $T_\Sigma(X)$:

$$\mathcal{C} = X \cup \{f^{T_\Sigma(X)} \mid f : n \to 1 \in \Sigma\}.$$

Deren induzierte Schrittfunktion $F_\mathcal{C}$ (siehe Abschnitt 4.6) stimmt offenbar mit F_Σ überein.

Die Elemente von $T_\Sigma =_{def} T_\Sigma(\emptyset)$, also die Terme ohne Variablen, heißen **Grundterme**.

Σ-Terme werden in Σ-Algebren ausgewertet:

Eine **Σ-Algebra** besteht aus

- einer – Daten-, Grund- oder **Trägermenge** – genannten Menge A und
- einer Menge $\{f^A : A^n \to A \mid f : n \to 1 \in \Sigma\}$ von Funktionen, den **Interpretationen von Σ in** A.

Falls keine Verwechslungen möglich sind, dann werden eine Algebra und ihre Trägermenge gleich bezeichnet.

Da ein nullstelliges Produkt mit der Menge $1 = \{*\}$ gleichgesetzt wird (siehe Abschnitt 3.1), wird eine Konstante $f : 0 \to 1$ (s.o.) als Element $f^A \in A$ interpretiert.

Alg_Σ bezeichnet die Klasse aller Σ-Algebren.

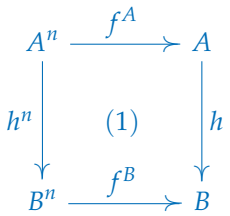
$T_\Sigma(X)$ ist die Trägermenge der gleichnamigen Σ-Algebra, die $f \in \Sigma$ durch (die Einschränkung von) $f^{T_\Sigma(X)}$ (auf $T_\Sigma(X)$; s.o.) interpretiert. Da $T_\Sigma(X)$ aus Termen besteht, wird sie als **Termalgebra** bezeichnet.

Sei $h : A \to B$ eine Funktion zwischen zwei Σ-Algebren A und B und $h^n : A^n \to B^n$ das n-fache Produkt von h (siehe Abschnitt Funktionen in Kapitel 3). Man schreibt oft nur h anstelle von h^n.

h heißt **Σ-Homomorphismus**, **Σ-homomorph** oder **mit Σ verträglich**, wenn für alle $f : n \to 1 \in \Sigma$

$$h \circ f^A = f^B \circ h^n$$

gilt, m.a.W., wenn das folgende Funktionsdiagramm kommutiert:



(1) veranschaulicht die Orthogonalität zwischen Operationen und Homomorphismen: Erstere transformieren Elemente *einer* Algebra, letztere stellen Bezüge zwischen *mehreren* Algebren her.

Bijektive Σ-Homomorphismen heißen **Σ-Isomorphismen**.

Lemma HOM Seien A, B, C Σ-Algebren, $g : A \to B$ ein surjektiver Σ-Homomorphismus und $h : B \to C$ derart, dass $h \circ g$ Σ-homomorph ist. Dann ist auch h Σ-homomorph. ❑

Lemma HOM ergibt sich aus folgender Charakterisierung surjektiver Funktionen:

$g : A \to B$ ist genau dann surjektiv, wenn g rechtskürzbar ist, d.h. für alle $h, h' : B \to C$ folgt $h = h'$ aus $h \circ g = h' \circ g$.

Die hier behandelten Signaturen Σ sind *einsortig*, d.h. jede Σ-Algebra hat genau eine Trägermenge. In realistischen Anwendungen kommen häufig Operationen vor, deren Definitionsbereich aus *verschiedenen* Mengen zusammengesetzt ist. Fast alle hier behandelten Begriffe und Resultate lassen sich problemlos auf mehrsortige Signaturen übertragen.

Die zugrundeliegende Signatur hat dann für jede dieser Mengen einen Namen, den man Sorte nennt. Z.B. kann der Konstruktor *cons* bzw. *poly* (siehe Beispiel 4.16 bzw. 4.17) nur als Interpretation einer Operation einer Signatur betrachtet werden, die für A und A^* bzw. $\mathbb{R}$ und $\mathbb{R}[x]$ jeweils eine eigene Sorte hat.

Dementsprechend ist die Trägermenge einer Algebra einer n-sortigen Signatur ein n-Tupel von Mengen: Jeder Sorte wird eine eigene Menge zugeordnet. Die Menge der Terme setzt sich aus n wechselseitig induktiv definierten Termmengen zusammen (siehe Abschnitt Induktiv definierte Mengen in Kapitel 4).

In der Prädikatenlogik kann man jede Sorte als einstelliges Prädikat realisieren und so mehrsortige Modelle durch einsortige simulieren (siehe Beispiel 9.1).

Im *Übersetzerbau* [32] werden die Terme einer (aus einer kontextfreien Grammatik gebildeten) mehrsortigen Signatur *Syntaxbäume* genannt. Sie liefern abstrakte Beschreibungen von *Quellprogrammen*, auf deren Basis sie ein Compiler in *Zielprogramme* übersetzt.

Auch logische Formeln sind Terme geeigneter Signaturen. Während die Syntax der aussagen- und modallogischer Formeln durch eine feste Signatur gegeben ist, baut dort die Signatur prädikatenlogischer Formeln auf einer beliebigen Signatur Σ von Operationen auf einem Datenbereich auf, dessen Eigenschaften durch die Formeln beschrieben werden. Sollen Bezüge zwischen $n > 1$ Datenbereichen beschrieben werden, dann muss Σ n-sortig sein.

Beispiel 5.1 Die Bitalgebra

Die in Kapitel 6 behandelte Aussagenlogik ist auf folgender Signatur aufgebaut:

$$AL \;=\; \{\bot, \top : 0 \to 1,\;\; \neg : 1 \to 1,\;\; \vee, \wedge, \Rightarrow : 2 \to 1\}.$$

Die AL-Algebra mit Trägermenge $2 = \{0,1\}$ und folgender Interpretation von AL nennen wir Bitalgebra:

$$\begin{aligned} \bot^2 &= 0, \\ \top^2 &= 1, \\ \neg^2 &= \lambda b.1 - b, \\ \wedge^2 &= min, \\ \vee^2 &= max, \\ \Rightarrow^2 &= \lambda(b,c).max(1-b,c) \;=\; \chi(\leq). \end{aligned}$$

$\bot$ (*bottom*) und $\top$ (*top*) werden also durch die Wahrheitswerte 0 bzw. 1 interpretiert.

Wir bezeichnen die Bitalgebra wie ihre Trägermenge mit 2.

AL-Terme nennt man üblicherweise **Boolesche Ausdrücke**. ❑

Aufgabe Zeigen Sie: Sind zwei Mengen A und B isomorph und ist A (die Trägermenge) eine(r) Σ-Algebra, dann ist auch B eine Σ-Algebra. ❑

Sei A eine Σ-Algebra und C eine Menge. A lässt sich wie folgt zu einer Σ-Algebra mit Trägermenge A^C **liften**:

- Für alle $f : n \to 1 \in \Sigma$, $g_1, \ldots, g_n \in A^C$ und $c \in C$,

$$f^{A^C}(g_1, \ldots, g_n)(c) \;=_{def}\; f^A(g_1(c), \ldots, g_n(c)).$$

Aufgabe Sei $h : A \to B$ ein Σ-Homomorphismus und C eine Menge. Zeigen Sie, dass dann auch $\lambda g.(h \circ g) : A^C \to B^C$ Σ-homomorph ist, wobei die Σ-Algebren A und B wie oben zu Σ-Algebren A^C bzw. B^C geliftet wurden. ❑

5.2 Termauswertung und -substitution

Die induktiv definierte Funktion $var : T_\Sigma(X) \to \mathcal{P}(X)$ liefert die Menge der Variablen von X, die in t vorkommen:

- Für alle $x \in X$, $var(x) = \{x\}$.
- Für alle $f : n \to 1 \in \Sigma$ und $t_1, \ldots, t_n \in T_\Sigma(X)$,

$$var(f(t_1, \ldots, t_n)) \;=_{def}\; \bigcup_{i=1}^{n} var(t_i).$$

Wie in Kapitel 3 bei der Einführung von Funktionsprodukten bemerkt wurde, kann jede Funktion $f : A \to B$ (wie z.B. *var*) auch auf *Tupel* von Elementen von A angewendet werden: Für alle $n > 0$ und $a_1, \ldots, a_n \in A$,

$$f(a_1, \ldots, a_n) \;=\; (f(a_1), \ldots, f(a_n)).$$

Funktionen von X in eine Σ-Algebra A heißen **Belegungen** *(valuations)* von X in A.

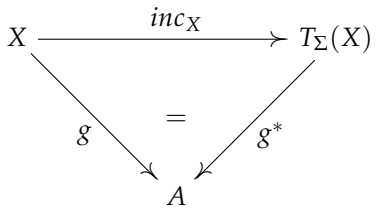
Gibt man eine Belegung $g : X \to A$ vor, dann ist der **Wert eines Terms** t **unter** g definiert als das Bild von t unter der induktiv definierten **Extension** oder **Fortsetzung von** g **auf** $T_\Sigma(X)$,

$$g^* : T_\Sigma(X) \to A.$$

- Für alle $x \in X$, $g^*(x) =_{def} g(x)$.
- Für alle $f : n \to 1 \in \Sigma$ und $t_1, \ldots, t_n \in T_\Sigma(X)$,

$$g^*(f(t_1, \ldots, t_n)) =_{def} f^A(g^*(t_1), \ldots, g^*(t_n)).$$

Aus dem ersten Teil der Definition von g^* folgt die Gleichung $g^* \circ inc_X = g$, die folgendem Funktionsdiagramm entspricht:



Anschaulich gesprochen, wertet g^* einen Term t unter g aus, d.h. nachdem g die Variablen von t mit Werten aus A belegt hat. Wir nennen $g^*(t)$ deshalb **den Wert von t unter g**.

Aufgabe Sei $\mathcal{C} = X \cup \{f^{T_\Sigma(X)} \mid f \in \Sigma\}$. Zeigen Sie durch strukturelle Induktion, dass g^* der einzige Σ-Homomorphismus von $T_\Sigma(X)$ nach A mit $g^* \circ inc_X = g$ ist. ❑

In Termen oder Funktionsanwendungen wird eine zweistellige Operation wie $\wedge : 2 \to 1$ bzw. $\wedge^2 : 2 \to 2$ (siehe Beispiel 5.1) häufig *zwischen* ihre beiden Argumente geschrieben (*Infixschreibweise*), d.h. man schreibt $t \wedge t'$ anstelle von $\wedge(t, t')$ und $0 \wedge^2 1$ anstelle von $\wedge^2(0, 1)$.

Beispiel 5.2

Sei $X = \{a, b, c, d\}$ und $\Sigma = \{\neg : 1 \to 1,\ \vee, =, +, * : 2 \to 1\}$.

☞ Schrittweise Auswertung des Terms

$$(a * (b + c + d) = a * b + a * c + a * d) \ \vee \ \neg(a * (b + c + d) = a * b + a * c + a * d)$$

in einer Σ-Algebra mit Trägermenge $\mathbb{N}$ unter der Belegung

$$g \ = \ (\lambda x.0)[5/a, 23/b, 45/c, 111/d] : X \to \mathbb{N}$$

Man sieht in der Animation, wie die Funktion g^* (hier auch mit g bezeichnet) zunächst *top-down* den Term traversiert, bis sie Variablen erreicht und durch Werte ersetzt. Danach führen *bottom-up* Anwendungen der Interpretationen von Σ-Operationen zur schrittweisen Auswertung des Terms. ❑

Da Grundterme keine Variablen enthalten, hängt die Einschränkung von g^* auf T_Σ nicht von g ab. Sie wird auch **Termfaltung** genannt und mit $fold^A$ bezeichnet.

Eine Σ-Algebra A heißt **erreichbar**, wenn $fold^A$ surjektiv ist, wenn also für alle $a \in A$ ein Σ-Grundterm t mit $fold^A(t) = a$ existiert.

Gemäß obiger Aufgabe ist $fold^A$ der einzige Σ-Homomorphismus von T_Σ nach A. Man nennt T_Σ deshalb die **initiale Σ-Algebra**.

Aufgabe Sei $f : T_\Sigma \to A$ eine induktiv definierte Funktion, wobei die Interpretationen der Operationen von Σ in der Termalgebra T_Σ als Konstruktoren von T_Σ (siehe Kapitel 4) dienen. Erweitern Sie A so zu einer Σ-Algebra, dass f mit $fold^A$ übereinstimmt. ❑

Belegungen durch Terme heißen **Substitutionen**. Belegungen durch Grundterme heißen **Grundsubstitutionen**.

Aufgabe Sei $\sigma : X \to T_\Sigma(X)$ eine Substitution. Zeigen Sie, dass ihre Extension

$$\sigma^* : T_\Sigma(X) \to T_\Sigma(X)$$

die folgende induktive Definition hat:

- Für alle $x \in X$, $\sigma^*(x) = \sigma(x)$.
- Für alle $f : n \to 1 \in \Sigma$ und $t_1, \ldots, t_n \in T_\Sigma(X)$,

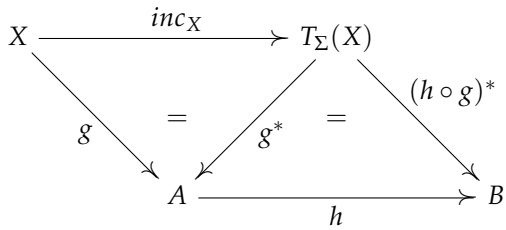
$$\sigma^*(f(t_1, \ldots, t_n)) = f(\sigma^*(t_1), \ldots, \sigma^*(t_n)). \qquad \square$$

Das Bild eines Terms t unter σ^* heißt σ-**Instanz** von t.

Lemma 5.3 (Substitutionslemma für Terme)

(i) Für alle Σ-Algebren A, $g : X \to A$ und Σ-Homomorphismen $h : A \to B$ gilt

$$(h \circ g)^* = h \circ g^*.$$



(ii) Für alle Σ-Algebren A, $g : X \to A$ und $\sigma : X \to T_\Sigma(X)$ gilt $(g^* \circ \sigma)^* = g^* \circ \sigma^*$.

(iii) Für alle erreichbaren Σ-Algebren A und $g : X \to A$ gibt es $\sigma : X \to T_\Sigma$ mit $fold^A \circ \sigma = g$, also $g^* = fold^A \circ \sigma^*$ wegen (i).

Beweis.

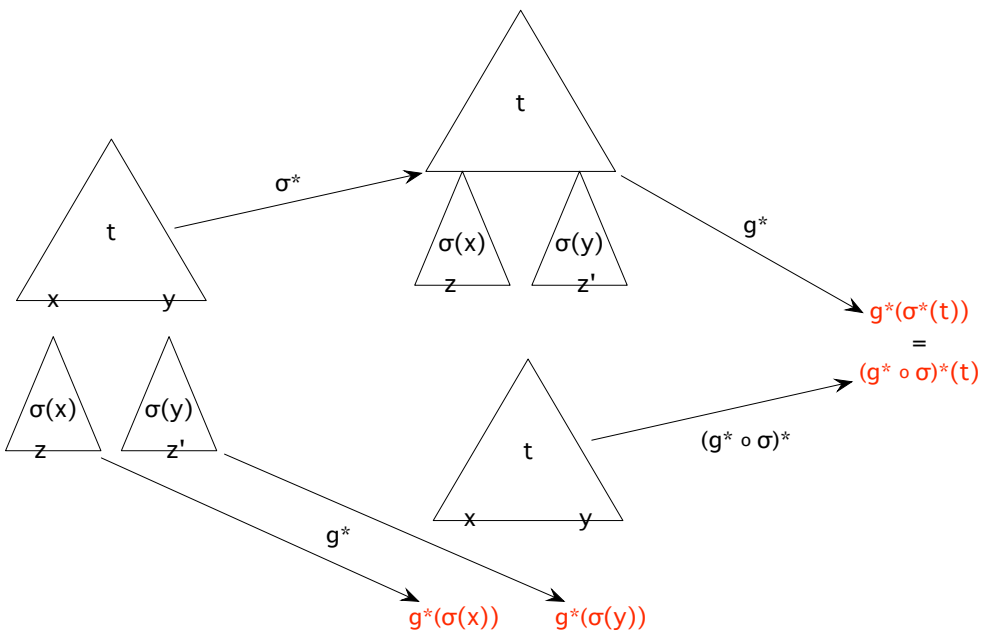
(i) Wegen $(h \circ g)^* \circ inc_X = h \circ g$ und $h \circ g^* \circ inc_X = h \circ g$ sind $(h \circ g)^*$ und $h \circ g^*$ zwei Σ-Homomorphismen $h' : T_\Sigma(X) \to B$, die $h'^* \circ inc_X = h \circ g$ erfüllen und deshalb miteinander übereinstimmen.

(ii) folgt aus (i), weil σ^* ein Σ-Homomorphismus ist.

(iii) Da $fold^A$ surjektiv ist, gibt es eine Auswahlfunktion $f : A \to T_\Sigma$ mit $fold^A \circ f = id_A$. Daraus folgt

$$fold^A \circ \sigma = fold^A \circ f \circ g = g$$

für $\sigma = f \circ g$. $\square$



Sei $x_1, \ldots, x_k \in X$, $t_1, \ldots, t_k, t \in T_\Sigma(X)$ und $\sigma, \sigma_1, \ldots, \sigma_n : X \to T_\Sigma(X)$.

Fortan schreiben wir auch

$$\begin{array}{rll} \{t_1/x_1, \ldots, t_k/x_k\} & \text{anstelle von} & inc_X[t_1/x_1, \ldots, t_k/x_k], \quad \text{(siehe Abschnitt 3.2)} \\ \sigma_1 \ldots \sigma_n & \text{anstelle von} & \sigma_n^* \circ \cdots \circ \sigma_2^* \circ \sigma_1, \\ t\sigma & \text{anstelle von} & \sigma^*(t). \end{array}$$

$\sigma_1 \ldots \sigma_n$ bezeichnet man auch als **Kleisli-Komposition** der Substitutionen $\sigma_1, \ldots, \sigma_n$. Wie die Funktionskomposition $\circ$, so ist auch die Kleisli-Komposition assoziativ:

Aufgabe

Folgern Sie aus Lemma 5.3 (ii), dass für alle $\rho, \sigma, \tau : X \to T_\Sigma(X)$ $(\rho\sigma)\tau = \rho(\sigma\tau)$ gilt.

Aufgabe Bezeichnen $\sigma\{t/x\}$ und $\sigma[t/x]$ dieselben Substitutionen?

$u \in T_\Sigma(X)$ ist ein **Teilterm** von $t \in T_\Sigma(X)$, wenn es einen Term v und $x \in var(v)$ gibt mit $v\{u/x\} = t$.

5.3 Termgleichungen

Sei $n \in \mathbb{N}$ und $t_1, \ldots, t_n, t_1', \ldots, t_n', t, t' \in T_\Sigma(X)$. Die Formel

$$e \quad = \quad t_1 \equiv t_1' \wedge \cdots \wedge t_n \equiv t_n' \;\Rightarrow\; t \equiv t'$$

heißt **Σ-Gleichung über** X.

Im Fall $n = 0$ ist e **unbedingt**, andernfalls **bedingt**.

Sei e eine Σ-Gleichung.

Eine Σ-Algebra A **erfüllt** e und e ist **gültig in** A, geschrieben: $A \models e$, wenn für alle Belegungen $g : X \to A$ Folgendes gilt:

$$g^*(t_1) = g^*(t_1') \wedge \cdots \wedge g^*(t_n) = g^*(t_n') \;\Rightarrow\; g^*(t) = g^*(t'). \tag{1}$$

In e sind $\wedge$ und $\Rightarrow$ lediglich Symbole, in (1) werden sie in ihrer umgangssprachlichen Bedeutung ("und" bzw. "impliziert") verwendet! Nur bei der Gleichheit unterscheiden wir zwischen dem Symbol ($\equiv$) und der Bedeutung als Diagonale ($=$; siehe Kapitel 3).

Eine Klasse $\mathcal{K}$ von Σ-Algebren **erfüllt** e und e ist **gültig in** $\mathcal{K}$, geschrieben: $\mathcal{K} \models e$, wenn e in allen Elementen von $\mathcal{K}$ gültig ist.

A bzw. $\mathcal{K}$ erfüllen eine Menge E von Σ-Gleichungen, geschrieben: $A \models E$ bzw. $\mathcal{K} \models E$, wenn für alle $e \in E$ $A \models e$ bzw. $\mathcal{K} \models e$ gilt.

Die **A-Äquivalenz** $\equiv^A$ ist die Menge aller Termpaare (t, t') und aller Substitutionspaare (σ, τ) mit $A \models t \equiv t'$ bzw. $A \models \sigma(x) \equiv \tau(x)$ für alle $x \in X$.

Lemma 5.4 (Äquivalenzlemma für Terme)

Sei $t \in T_\Sigma(X)$, $\sigma, \tau : X \to T_\Sigma(X)$ und A eine Σ-Algebra.

$$\sigma \equiv^A \tau \quad \Rightarrow \quad t\sigma \equiv^A t\tau.$$

Beweis. Sei $g : X \to A$ und $\sigma \equiv^A \tau$. Daraus folgt

$$(g^* \circ \sigma)(x) = g^*(\sigma(x)) = g^*(\tau(x)) = (g^* \circ \tau)(x). \tag{1}$$

Aus zweimaliger Anwendung von Lemma 5.3 (ii) folgt daher

$$g^*(\sigma^*(t)) = (g^* \circ \sigma)^*(t) \overset{(1)}{=} (g^* \circ \tau)^*(t) = g^*(\tau^*(t)),$$

also $t\sigma = \sigma^*(t) \equiv^A \tau^*(t) = t\tau$. ❑

5.4 Normalformen

Eine Teilmenge NF von $T_\Sigma(X)$ mit $X \subseteq NF$ ist eine Menge von **Normalformen** bzgl. einer Σ-Algebra A, wenn NF für jeden Σ-Term t einen Term u mit $t \equiv^A u$ enthält. Man nennt u dann eine Normalform **von** t bzgl. A.

Idealerweise sind Normalformen eindeutig, m.a.W.: äquivalente Normalformen sind gleich. In diesem Fall liefert jeder Algorithmus, der Terme in äquivalente Normalformen überführt, ein Entscheidungsverfahren für die Äquivalenz zweier Terme t und t':

- Berechne Normalformen u und u' von t bzw. t'.
- Sind u und u' gleich, dann gilt $t \equiv^A u = u' \equiv^A t'$. Also sind t und t' äquivalent. Sind u und u' verschieden, dann erhält man $t \not\equiv^A t'$ durch Kontraposition: Wären t und t' äquivalent, dann wären auch u und u' äquivalent. Also wären nach Voraussetzung u und u' gleich. ↯

Die Eindeutigkeit von Normalformen ist oft weit schwieriger nachzuweisen als ihre Existenz und erfordert Begriffe aus der **Termersetzungstheorie** (siehe z.B. [18], Kapitel 5). Hier interessiert uns vor allem, wie äquivalente Normalformen berechnet werden können.

Beispiel 5.5

Sei $\Sigma = \{+, * : 2 \to 1\}$, $X = \{x, y, z, z'\}$,

$$E = \{x * (y + z) \equiv (x * y) + (x * z),\ \ (x + y) * z \equiv (x * z) + (y * z)\}$$

und NF die Menge aller Σ-Terme über X, deren Teilterme der Form $t * u$ keine Teilterme der Form $t + u$ haben.

Sei A eine Σ-Algebra mit Trägermenge $\mathbb{N}$ und der üblichen Interpretation von Σ in $\mathbb{N}$. Dann gibt es für jeden Σ-Term t einen Term u, der bzgl. A zu t äquivalent ist. Unten werden zwei Verfahren beschrieben, mit deren Hilfe das gezeigt werden kann (siehe Beispiel 5.8 bzw. 5.10).

Aufgabe Zeigen Sie, dass der Σ-Term $(x + y) * (z + z')$ mehrere Normalformen bzgl. A hat. ❑

Normalisierungen logischer Formeln sind erforderlich, weil logische Ableitungsregeln oft nur auf Normalformen anwendbar sind. Das verkleinert den **Suchraum** (die Menge möglicher Regelanwendungen auf eine einzelne Formel), führt aber vielfach zu Ableitungen, die für Menschen nur schwer nachvollziehbar sind, weil Normalisierungen die intuitive Verständlichkeit logischer Formeln i.a. verschlechtern.

Das spielt keine Rolle, wenn Ableitungen vollautomatisch erzeugt werden. Sind jedoch Eingriffe in den Ableitungsprozess notwendig – weil der Suchraum zu groß oder der verwendete Kalkül unvollständig ist (siehe Kapitel 2) –, dann müssen die am Eingriffspunkt vorliegenden Zwischenergebnisse schnell verstehbar sein, damit entscheiden werden kann, wie die Ableitung fortgesetzt werden soll.

Im Gegensatz zur **rekursiven** Auswertung eines Terms t (siehe oben) werden Normalformen von t üblicherweise durch **iterative** – d.h. wiederholte – Anwendung von Gleichungen auf t berechnet.

5.5 Äquivalenzrelationen und Quotienten

Um die Korrektheit solcher Ableitungen beweisen zu können, widmen wir uns zunächst den wichtigen Eigenschaften von $\equiv^A$ (die z.T. schon im o.g. Entscheidungsverfahren für Termäquivalenz verwendet wurden).

Sei A eine Menge. Eine Relation $R \subseteq A^2$ heißt

- reflexiv, wenn die Diagonale von A^2 eine Teilmenge von R ist; (1)
- transitiv: Für alle $(a,b),(b,c) \in R$ auch (a,c) zu R gehört; (2)
- symmetrisch, wenn für alle $(a,b) \in R$ auch (b,a) zu R gehört; (3)
- antisymmetrisch, wenn alle $a,b \in A$ mit $(a,b),(b,a) \in R$ gleich sind; (4)

Eine reflexive und transitive Relation heißt **Prä-** oder **Quasiordnung**.

Eine antisymmetrische Präordnung heißt **Halbordnung** oder **partielle Ordnung**.

Eine Halbordnung R heißt **totale Ordnung**, wenn für alle $a,b \in A$ (a,b) oder (b,a) zu R gehört.

Eine wohlfundierte (siehe Kapitel 4) und totale Ordnung heißt **Wohlordnung**.

R heißt **Äquivalenzrelation auf** A, wenn (1)-(3) gelten.

Eine Äquivalenzrelation R liefert folgende Zerlegung von A:

$$A/R \;=_{def}\; \{[a]_R \mid a \in A\}.$$

$[a]_R =_{def} \{b \in A \mid (a,b) \in R\}$ heißt **Äquivalenzklasse** von a und A/R der nach R **faktorisierte Quotient** von A.

Die **natürliche Abbildung** $nat_R : A \to A/R$ bildet $a \in A$ auf $[a]_R$ ab.

Umgekehrt liefert jede Zerlegung $Z = \{A_1,\ldots,A_n\}$ von A die Äquivalenzrelation

$$R_Z = \{(a,b) \in A^2 \mid \text{es gibt } 1 \le i \le n \text{ mit } a,b \in A_i\}.$$

Aufgabe Zeigen Sie $A/R_Z = Z$. ❑

Sei A eine Σ-Algebra. Eine Äquivalenzrelation R auf A heißt **Σ-Kongruenz**, wenn R mit allen $f : n \to 1 \in \Sigma$ **verträglich** ist, d.h. wenn für alle $a_1,\ldots,a_n,b_1,\ldots,b_n \in A^n$ Folgendes gilt:

$$(a_1,b_1),\ldots,(a_n,b_n) \in R \quad\Rightarrow\quad (f^A(a_1,\ldots,a_n), f^A(b_1,\ldots,b_n)) \in R. \tag{5}$$

Aufgabe Zeigen Sie, dass für alle Σ-Algebren B $\equiv^B$ eine Σ-Kongruenz auf $T_\Sigma(X)$ ist, also (1)-(3) und (5) für $A = T_\Sigma(X)$ erfüllt. ❑

Sei R eine Σ-Kongruenz. Die Quotientenmenge A/R ist die Trägermenge der gleichnamigen **Quotientenalgebra**, die alle $f : n \to 1 \in \Sigma$ wie folgt interpretiert: Für alle $a \in A^n$,

$$f^{A/R}([a]_R) \;=_{def}\; nat_R(f^A(a)).$$

Aufgabe Zeigen Sie, dass $f^{A/R}$ wohldefiniert ist, dass also aus $[a]_R = [b]_R$

$$f^{A/R}([a]_R) = f^{A/R}([b]_R)$$

folgt. ❑

Aufgabe Zeigen Sie, dass die natürliche Abbildung (s.o) Σ-homomorph ist. ❑

5.6 Gleichungskalkül

Sei E eine Menge von Σ-Gleichungen.

Der **Gleichungskalkül** GK ist synthetisch und besteht aus fünf Regeln:

Reflexivitätsregel

$$\frac{}{E \vdash t \equiv t} \qquad t \in T_\Sigma(X)$$

Symmetrieregel

$$\frac{E \vdash t \equiv t'}{E \vdash t' \equiv t} \qquad t, t' \in T_\Sigma(X)$$

Transitivitätsregel

$$\frac{E \vdash t \equiv t', \quad E \vdash t' \equiv t''}{E \vdash t \equiv t''} \qquad t, t', t'' \in T_\Sigma(X)$$

Kongruenzregel

$$\frac{E \vdash t_1 \equiv t_1', \quad \ldots, \quad E \vdash t_n \equiv t_n'}{E \vdash f(t_1, \ldots, t_n) \equiv f(t_1', \ldots, t_n')} \qquad \begin{array}{l} f : n \to 1 \in \Sigma, \\ t_1, \ldots, t_n, t_1', \ldots, t_n' \in T_\Sigma(X) \end{array}$$

Modus ponens

$$\frac{E \vdash t_1\sigma \equiv t_1'\sigma, \quad \ldots, \quad E \vdash t_n\sigma \equiv t_n'\sigma}{E \vdash t\sigma \equiv t'\sigma} \qquad \begin{array}{l} (t_1 \equiv t_1' \wedge \cdots \wedge t_n \equiv t_n' \Rightarrow t \equiv t') \in E, \\ \sigma : X \to T_\Sigma(X) \end{array}$$

$Alg_{\Sigma,E}$ bezeichnet die Klasse aller Σ-Algebren, die E erfüllen:

$$Alg_{\Sigma,E} =_{def} \{A \in Alg_\Sigma \mid A \models E\}.$$

Σ-Algebren, die E erfüllen, heißen (Σ, E)**-Algebren**.

Satz 5.6 Korrektheit des Gleichungskalküls

Für alle Mengen E von Σ-Gleichungen und unbedingten Σ-Gleichungen e gilt:

$$E \vdash_{GK} e \quad \Rightarrow \quad Alg_{\Sigma,E} \models e.$$

Beweis. Nach Satz 4.1 (iii) gilt die Behauptung, wenn die Menge P aller Paare $(E, e) \in \mathcal{P}(T_\Sigma(X)^2) \times T_\Sigma(X)^2$ mit $Alg_{\Sigma,E} \models e$ F_{GK}-abgeschlossen ist (siehe Abschnitt 4.4).

Sei $(E, t \equiv t') \in F_{GK}(P)$.

Fall 1 (Reflexivitätsregel): $t = t'$. Dann gilt $Alg_{\Sigma,E} \models t \equiv t'$.

Fall 2 (Symmetrieregel): $(E, t' \equiv t) \in P$. Dann gilt $Alg_{\Sigma,E} \models t' \equiv t$, also auch $Alg_{\Sigma,E} \models t \equiv t'$.

Fall 3 (Transitivitätsregel): Es gibt einen Σ-Term t'' mit $(E, t \equiv t'')$, $(E, t'' \equiv t') \in P$. Dann gilt $Alg_{\Sigma,E} \models t \equiv t''$ und $Alg_{\Sigma,E} \models t'' \equiv t'$, also auch $Alg_{\Sigma,E} \models t \equiv t'$.

Fall 4 (Kongruenzregel): Es gibt $f : n \to 1 \in \Sigma$ und Σ-Terme $t_1, \ldots, t_n, t_1', \ldots, t_n'$ mit $t = f(t_1, \ldots, t_n)$, $t' = f(t_1', \ldots, t_n')$ und $(E, t_i \equiv t_i') \in P$ für alle $1 \le i \le n$. Dann gilt $g^*(t_i) = g^*(t_i')$ für alle (Σ, E)-Algebren A und $g \in A^X$, also auch

$$\begin{aligned} &g^*(t) = g^*(f(t_1, \ldots, t_n)) = f^A(g^*(t_1), \ldots, g^*(t_n)) = f^A(g^*(t_1'), \ldots, g^*(t_n')) \\ &= g^*(f(t_1', \ldots, t_n')) = g^*(t'). \end{aligned}$$

Daher gilt $t \equiv t'$ in allen (Σ, E)-Algebren.

Fall 5 (Modus ponens): Es gibt $(t_1 \equiv t_1' \wedge \cdots \wedge t_n \equiv t_n' \Rightarrow u \equiv u') \in E$, Σ-Terme $t_1, \ldots, t_n, t_1', \ldots, t_n'$ und $\sigma : X \to T_\Sigma(X)$ mit $t = u\sigma$, $t' = u'\sigma$ und $(E, t_i\sigma \equiv t_i'\sigma) \in P$ für alle $1 \le i \le n$. Dann gilt $g^*(t_i\sigma) = g^*(t_i'\sigma)$ für alle (Σ, E)-Algebren A und $g \in A^X$, also auch

$$(g^* \circ \sigma)^*(t_i) \overset{\text{Lemma 5.3 (ii)}}{=} g^*(t_i\sigma) = g^*(t_i'\sigma) \overset{\text{Lemma 5.3 (ii)}}{=} (g^* \circ \sigma)^*(t_i'). \tag{1}$$

Da A E erfüllt, folgt

$$g^*(t) = g^*(u\sigma) \overset{\text{Lemma 5.3 (ii)}}{=} (g^* \circ \sigma)^*(u) = (g^* \circ \sigma)^*(u') \overset{\text{Lemma 5.3 (ii)}}{=} g^*(u'\sigma) = g^*(t')$$

aus (1). Daher gilt $t \equiv t'$ in allen (Σ, E)-Algebren.

Demnach gehört $(E, t \equiv t')$ in allen fünf Fällen zu P. ❑

Die Menge aller Paare (t, t') von Σ-Termen mit $E \vdash_{GK} t \equiv t'$ heißt E**-Äquivalenz** und wird mit $\equiv_E$ bezeichnet.

Aus Satz 5.6 folgt, dass für alle (Σ, E)-Algebren A $\equiv_E$ in $\equiv^A$ enthalten ist. (2)

5.7 Berechnung äquivalenter Normalformen

Erstes Normalisierungsverfahren

Die E**-Reduktionsrelation** $\to_E$ besteht aus allen Paaren

$$(u\{t\sigma/x\}, u\{t'\sigma/x\})$$

von Σ-Termen mit $u \in T_\Sigma(X)$, $t \equiv t' \in E$ und $\sigma : X \to T_\Sigma(X)$.

Ein Σ-Term t heißt E**-irreduzibel**, wenn kein t' mit $t \to_E t'$ existiert.

Die kleinste transitive Relation auf $T_\Sigma(X)$, die $\to_E$ enthält, wird mit $\overset{+}{\to}_E$ bezeichnet.

Aufgabe Zeigen sie, dass $\overset{+}{\to}_E$ die kleinste transitive und mit Σ verträgliche Relation ist, die

$$Inst(E) \;=_{def}\; \{(t\sigma, t'\sigma) \mid (t, t') \in E,\ \sigma : X \to T_\Sigma(X)\}$$

enthält. Folgern Sie daraus, dass $\overset{+}{\to}_E$ eine Teilmenge von $\equiv_E$ ist. ❑

Satz 5.7 Sei $NF \subseteq T_\Sigma(X)^2$ und A eine (Σ, E)-Algebra.

Ist $\to_E$ ist wohlfundiert und gehört jeder E-irreduzible Term zu NF, dann hat jeder Σ-Term t eine Normalform u in NF, d.h. es gilt $t \equiv^A u$.

Beweis. Aus der Wohlfundiertheit von $\to_E$ folgt die Existenz eines E-irreduziblen Σ-Terms u mit $t \overset{+}{\to}_E u$. Also gehört u zu NF. Aus $\overset{+}{\to}_E \subseteq \equiv_E \overset{(2)}{\subseteq} \equiv^A$ folgt $t \equiv^A u$. ❑

Ein Kriterium für Wohlfundiertheit von $\to_E$ ist die Existenz einer **Gewichtsfunktion**

$$weight : T_\Sigma(X) \to W,$$

die Termen Elemente einer Menge W zuordnet, auf der es eine wohlfundierte Halbordnung $<$ mit folgender Eigenschaft gibt:

- Für alle $t, t' \in T_\Sigma(X)$ mit $t \to_E t'$ gilt $weight(t) > weight(t')$. (3)

Gäbe es eine unendliche Reduktion $t_1 \to_E t_2 \to_E t_3 \to_E \ldots$, dann gälte

$$weight(t_1) > weight(t_2) > weight(t_3) > \ldots,$$

was der Wohlfundiertheit von $<$ widerspricht.

Die einfachste Gewichtsfunktion ordnet einem Term die Anzahl seiner Symbole zu. Sie erfüllt aber selten Bedingung (3).

Häufig genügt es, natürliche Zahlen als Gewichte und $<$ als die übliche Ordnung auf $\mathbb{N}$ zu wählen. Manchmal wird aber eine komplexere Ordnung benötigt, die z.B. auf Tupeln oder Multimengen natürlicher Zahlen basiert.

Hierzu zählen die **lexikographische** bzw. **Multimengen-Erweiterung** einer wohlfundierten Ordnung $<$ auf einer Menge A:

Für alle $n > 1$, $a = (a_1, \ldots, a_n), b = (b_1, \ldots, b_n) \in A^n$ und $f, g \in \mathcal{B}(A) = \mathbb{N}^A$,

$$a <_{lex} b \quad \Leftrightarrow_{def} \quad \text{es gibt } 1 \leq i \leq n \text{ mit } a_i < b_i \text{ und } a_j \leq b_j \text{ für alle } 1 \leq j < i.$$

Für alle $f, g \in \mathcal{B}(A) = \mathbb{N}^A$ (siehe Isomorphien),

$$f <_{mult} g \quad \Leftrightarrow_{def} \quad f \neq g \text{ und für alle } a \in A_f \setminus A_g \text{ gibt es } b \in A_g \setminus A_f \text{ mit } a < b,$$

wobei $A_f = f^{-1}(\mathbb{N}_{>0})$ und $A_g = g^{-1}(\mathbb{N}_{>0})$.

Beispiel 5.8 Seien Σ, X, E und NF wie in Beispiel 5.5. Man sieht sofort, dass E-irreduzible Terme zu NF gehören. ☞ Schrittweise E-Reduktion des Terms

$$dist \;=\; 5 * (6 * (11 * (x + y + z) * 14 + (c + g + b) * 22) + 44 * (gg + hh))$$

zu einer Normalform. $+$ und $*$ treten hier auch als dreistellige Funktionssymbole auf.

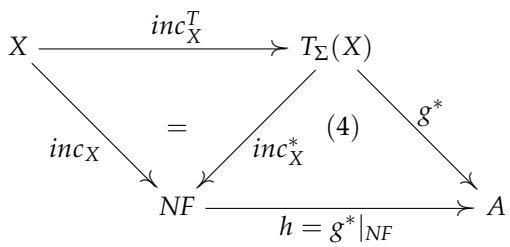
Ein Termgewicht, das Bedingung (3) erfüllt, ist die Multimenge der Längen aller *Pfade* von einem mit $*$ markierten *Knoten* zu einem *Blatt*, wobei sich die Begriffe Pfad, Knoten und Blatt auf die Baumdarstellung des gewichteten Terms beziehen. ❑

Zweites Normalisierungsverfahren

Eine andere Methode, äquivalente Normalformen bzgl. einer Algebra zu berechnen, erfordert die Erweiterung von NF zur Σ-Algebra: Eine Normalform von t erhält man dann als Bild unter $inc_X^* : T_\Sigma(X) \to NF$:

Satz 5.9 Sei $NF \subseteq T_\Sigma(X)$ (die Trägermenge) eine(r) Σ-Algebra, inc_X^* surjektiv, A eine Σ-Algebra und $g : X \to A$.

In folgendem Diagramm kommutiert (4) genau dann, wenn $h =_{def} g^*|_{NF}$, d.i. die Einschränkung von $g^* : T_\Sigma(X) \to A$ auf NF, Σ-homomorph ist.



Beweis. Für alle $x \in X$ gilt $h(inc_X(x)) = h(x) = g^*(x) = g(x)$, also

$$h \circ inc_X = g. \tag{5}$$

Ist h homomorph, dann ist auch $h \circ inc_X^*$ homomorph und $h \circ inc_X^* = g^*$ folgt wegen

$$h \circ inc_X^* \circ inc_X^T \overset{Lemma\ 5.3}{=} (h \circ inc_X)^* \circ inc_X^T \overset{(5)}{=} g^* \circ inc_X^T$$

aus der Eindeutigkeit von g^*. Ist umgekehrt (4) kommutativ, dann folgt aus der Surjektivität von inc_X^* und Lemma HOM (siehe Abschnitt 5.1), dass h Σ-homomorph ist. ❑

Ist (4) kommutativ, dann sind ein Term t und seine Normalform $inc_X^*(t)$ äquivalent bzgl. A:

$$g^*(t) = h \circ inc_X^*(t) = g^*(inc_X^*(t)).$$

Die in Satz 5.9 verlangte Surjektivität von inc_X^* gilt z.B. dann, wenn für alle $t \in NF$ $inc_X^*(t) = t$ ist.

Um einen Term zu normalisieren, muss man ihn also nur in NF auswerten.

Beispiel 5.10

Seien Σ, NF und A wie in Beispiel 5.5. $+$ und $*$ werden in NF als induktiv definierte Funktionen interpretiert: Für alle $t, t' \in NF$,

$$
\begin{aligned}
t +^{NF} t' &= t + t', \\
t *^{NF} t' &= \begin{cases} t * t' & \text{falls } t, t' \in T_{\{*\}}(X), \\ (u *^{NF} t') + (v *^{NF} t') & \text{falls es } u, v \in NF \text{ gibt mit } t = u + v, \\ (t *^{NF} u) + (t *^{NF} v) & \text{falls } t \in T_{\{*\}}(X) \text{ und es } u, v \in NF \text{ gibt} \\ & \text{mit } t' = u + v. \end{cases}
\end{aligned}
$$

Die Definition von $*^{NF}$ basiert auf folgender induktiver Defintion des Produktes NF^2:

- $t, t' \in T_{\{*\}}(X) \;\Rightarrow\; (t, t') \in NF^2$,
- $(u, t'), (v, t') \in NF^2 \;\Rightarrow\; (u + v, t') \in NF^2$,
- $t \in T_{\{*\}}(X) \wedge (t, u), (t, v) \in NF^2 \;\Rightarrow\; (t, u + v) \in NF^2$.

Man kürzt deshalb die Definition von $*^{NF}$ wie folgt ab:

Für alle $t, t' \in T_{\{*\}}(X)$ und $u, v \in NF$,

$$
\begin{aligned}
t *^{NF} t' &= t * t', \\
(u + v) *^{NF} t &= (u *^{NF} t) + (v *^{NF} t''), \\
t *^{NF} (u + v) &= (t'' *^{NF} u) + (t *^{NF} v).
\end{aligned}
$$

h in Satz 5.9 ist Σ-homomorph.

Beweis durch strukturelle Induktion über NF^2 (siehe Abschnitt 4.6). Sei $t, t' \in NF$. Dann gilt

$$h(t +^{NF} t') = h(t + t') = h(t +^{T_\Sigma(X)} t') = h(t) +^A h(t').$$

Bei der Multiplikation müssen wir zwei Fälle unterscheiden:

Fall 1: $t, t' \in T_{\{*\}}(X)$. Dann gilt

$$h(t *^{NF} t') = h(t * t') = h(t *^{T_\Sigma(X)} t') = h(t) *^A h(t').$$

Fall 2: Es gibt $u, v \in NF$ mit $t = u + v$. Dann gilt

$$
\begin{aligned}
&h(t *^{NF} t') = h((u *^{NF} t') + (v *^{NF} t')) = h((u *^{NF} t') +^{T_\Sigma(X)} (v *^{NF} t')) \\
&= h(u *^{NF} t') +^A h(v *^{NF} t') \overset{\text{ind. hyp.}}{=} (h(u) *^A h(t')) +^A (h(v) *^A h(t')) \\
&= (h(u) +^A h(v)) *^A h(t') \overset{\text{ind. hyp.}}{=} h(u +^{NF} v) *^A h(t') = h(u + v) *^A h(t') = h(t) *^A h(t').
\end{aligned}
$$

Fall 3: Es gibt $u, v \in NF$ mit $t' = u + v$.

Analog zu Fall 2 erhält man $h(t *^{NF} t') = h(t) *^A h(t')$. ❑

Sei $X = \{x, y, z, c, g, b, gg, hh\}$. In der ☞ schrittweisen Auswertung des Terms *dist* steht *nf* für inc_X^* und *add* bzw. *mul* für die Interpretation von $+$ bzw. $*$ in NF. $+$ bzw. $*$ kommen in der Auswertung auch als dreistellige Operationen vor.

Beispiel 5.11 (siehe Beispiel 5.1)

Sei $nand = \lambda(x,y).\neg(x \wedge y) : T_{AL}(X)^2 \to T_{AL}(X)$ und $NF \subseteq T_{AL}(X)$ die folgendermaßen induktiv definierte Menge von Normalformen:

- $X \cup \{\top\} \subseteq NF$,
- $t, u \in NF \;\Rightarrow\; nand(t,u) \in NF$.

Die Operationen von AL werden in NF wie folgt interpretiert: Für alle $t, u \in NF$,

$$\begin{aligned} \top^{NF} &= \top, \\ \bot^{NF} &= nand(\top,\top), \\ \neg^{NF}(t) &= nand(t,t), \\ t \wedge^{NF} u &= nand(nand(t,u), nand(t,u)), \\ t \vee^{NF} u &= nand(nand(t,t), nand(u,u)), \\ t \Rightarrow^{NF} u &= nand(t, nand(u,u)). \end{aligned}$$

Aufgabe

Sei A die Bitalgebra. Zeigen Sie, dass h in Satz 5.9 AL-homomorph ist. Im Gegensatz zu Beispiel 5.10 wird hier keine strukturelle Induktion benötigt, weil die Interpretation der Operationen von AL nicht induktiv definiert ist. ❑

Satz 5.12 Vollständigkeit des Gleichungskalküls

Für jede unbedingte Σ-Gleichung e gilt:

$$E \vdash_{GK} e \;\Leftrightarrow\; Alg_{\Sigma,E} \models e \;\Leftrightarrow\; T_{\Sigma,E}(X) = GT(X)/\equiv_E \models e.$$

Beweis. Sei $E \vdash_{GK} e$. Aus Satz 5.6 folgt $Alg_{\Sigma,E} \models e$.

Sei $Alg_{\Sigma,E} \models e$. Da $\equiv_E$ eine Σ-Kongruenz ist, lässt sich die Quotientenmenge $T_{\Sigma,E}(X)$ wie oben beschrieben zur Σ-Algebra erweitern.

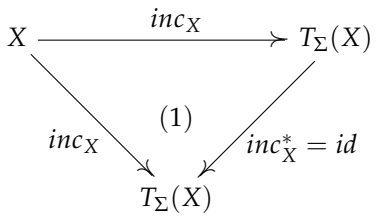
Sei $\varphi = (t_1 \equiv t_1' \wedge \cdots \wedge t_n \equiv t_n' \Rightarrow t \equiv t') \in E$ und $g : X \to T_{\Sigma,E}(X)$ mit $g^*(t_i) = g^*(t_i')$ für alle $1 \le i \le n$. Dann gibt es $\sigma : X \to T_\Sigma(X)$ mit $nat_{\equiv_E} \circ \sigma = g$. Daraus folgt

$$\begin{aligned} nat_{\equiv_E}(t_i\sigma) &\overset{\text{Lemma 5.3 } (i)}{=} (nat_{\equiv_E} \circ \sigma)^*(t_i) = g^*(t_i) = g^*(t_i') = (nat_{\equiv_E} \circ \sigma)^*(t_i') \\ &\overset{\text{Lemma 5.3 } (i)}{=} nat_{\equiv_E}(t_i'\sigma), \end{aligned}$$

also $t_i\sigma \equiv_E t_i'\sigma$, d.h. es gibt eine GK-Ableitung von $E \vdash t_i\sigma \equiv t_i'\sigma$. Eine Anwendung des Modus ponens auf die Gleichungen $t_1\sigma \equiv t_1'\sigma, \ldots, t_n\sigma \equiv t_n'\sigma$ und φ liefert die Gleichung $t\sigma \equiv t'\sigma$. Demnach gilt $t\sigma \equiv_E t'\sigma$ und wir erhalten

$$\begin{aligned} g^*(t) = (nat_{\equiv_E} \circ \sigma)^*(t) &\overset{\text{Lemma 5.3 } (i)}{=} nat_{\equiv_E}(t\sigma) = nat_{\equiv_E}(t'\sigma) \\ &\overset{\text{Lemma 5.3 } (i)}{=} (nat_{\equiv_E} \circ \sigma)^*(t') = g^*(t'). \end{aligned}$$

Also gilt φ in $T_{\Sigma,E}(X)$. Wegen $Alg_{\Sigma,E} \models e$ folgt $T_{\Sigma,E}(X) \models e$.



Sei $T_{\Sigma,E}(X) \models e$ und $(t,t') = e$. Daraus folgt

$$\begin{aligned} nat_{\equiv_E}(t) = nat_{\equiv_E}(id(t)) &\overset{(1)}{=} nat_{\equiv_E}(inc_X^*(t)) \overset{\text{Lemma 5.3 } (i)}{=} (nat_{\equiv_E} \circ inc_X)^*(t) \\ &\overset{T_\Sigma(X)/\equiv_E \text{ erfüllt } t\equiv t'}{=} (nat_{\equiv_E} \circ inc_X)^*(t') \overset{\text{Lemma 5.3 } (i)}{=} nat_{\equiv_E}(inc_X^*(t')) \overset{(1)}{=} nat_{\equiv_E}(id(t')) \\ &= nat_{\equiv_E}(t'), \end{aligned}$$

also $E \vdash_{GK} e$. ❑

5.8 Gleichungslogik in anderen Logiken

Bei allen drei in den nächsten Kapiteln behandelten Logiken folgen Syntax und Semantik dem gleichen Definitionsschema. Zunächst wird eine Signatur L logischer Operationen angegeben. Die Formelmenge wird als Menge $T_L(At)$ der L-Terme über einer Menge At von **Atomen** definiert.

In der Aussagenlogik ($L = AL$); siehe Beispiel 5.1) und der Modallogik ($L = ML$) ist At eine beliebige – i.d.R. endliche – Menge.

In der Prädikatenlogik ($L = PL$) ist At eine Menge von Ausdrücken der Form $p(t_1, \ldots, t_n)$, die aus Termen $t_1, \ldots, t_n$ einer beliebigen Signatur Σ und – neben den Operationen ebenfalls zu Σ gehörigen Prädikaten p besteht. Demnach ist eine prädikatenlogische Formel ein Term mit Symbolen aus *zwei* Signaturen: PL und Σ.

Während die Bedeutung einer aussagenlogischen Formel durch 0 oder 1 gegeben ist, entspricht sie bei einer modal- oder prädikatenlogischen Formel der Menge aller **Zustände** ("Welten"), die sie erfüllen.

An die Stelle einer Belegung $g : At \to 2$ durch Wahrheitswerte tritt in der Modallogik eine **Kripke-Struktur**

$$\mathcal{K} : State \to \mathcal{P}(State) \times \mathcal{P}(At),$$

die jedem Zustand s seine (direkten) Folgezustände sowie alle atomaren Formeln zuordnet, die im Zustand s gelten (sollen).

In der Prädikatenlogik besteht die Zustandsmenge $State$ aus allen **Belegungen** von X in einer **Σ-Struktur** A, d.i. eine Σ-Algebra, die die Prädikate von Σ als Relationen auf A interpretiert.

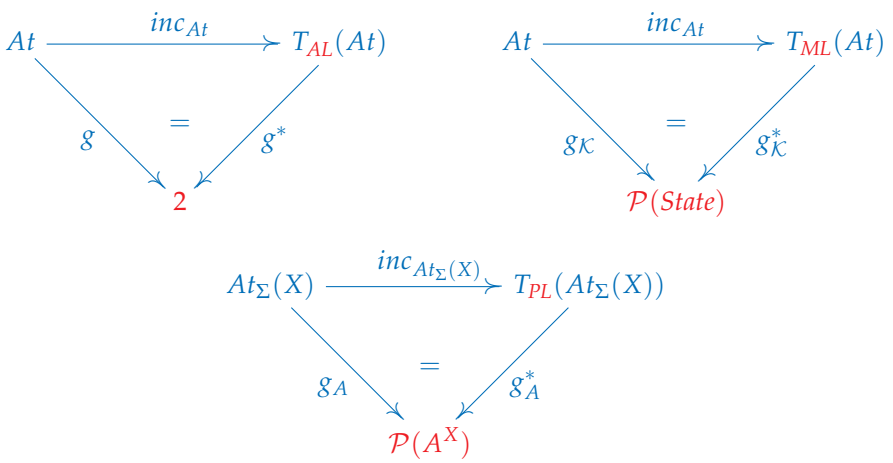
Die jeweilige Kripke-Struktur $\mathcal{K}$ bzw. Σ-Struktur A liefert eine Belegung

$$g_\mathcal{K} : At \to \mathcal{P}(State) \quad \text{bzw.} \quad g_A : At_\Sigma(X) \to \mathcal{P}(A^X)$$

atomarer Formeln in der ML- bzw. PL-Algebra $\mathcal{P}(State)$ bzw. $\mathcal{P}(A^X)$, deren Extension

$$g_\mathcal{K}^* : T_{ML}(At) \to \mathcal{P}(State) \quad \text{bzw.} \quad g_A^* : T_{PL}(At_\Sigma(X)) \to \mathcal{P}(A^X)$$

modal- bzw. prädikatenlogischen Formeln ihre jeweilige Bedeutung zuordnet:



Die Formeln mancher Logiken – wie z.B. SQL, XPath oder sog. Beschreibungslogiken (*description logics*) – beschreiben auch mehrstellige Relationen. Dann besteht die Signatur L aus Operatoren, die Relationen bilden oder mit anderen Relationen verknüpfen, während der semantische Bereich, in dem die Formeln interpretiert werden, die Potenzmenge eines Produkts oder – im Fall binärer Relationen – eine Menge mehrwertiger Funktionen ist (siehe Kapitel 3). Eine solche Logik wird z.B. in [34], Kapitel 29, behandelt.

5.9 Highlights

Eine **Signatur** Σ besteht aus Operation(ssymbol)en der Form $f : n \to 1$ mit $n \in \mathbb{N}$.

Induktive Definition der Menge $T_\Sigma(X)$ der **Σ-Terme** über X:

- $X \subseteq T_\Sigma(X)$,
- Für alle $f : n \to 1 \in \Sigma$ und $t \in T_\Sigma(X)^n$, $ft \in T_\Sigma(X)$.

Eine **Σ-Algebra** A besteht aus einer – oft ebenfalls mit A bezeichneten – **Trägermenge** und für alle $f : n \to 1 \in \Sigma$ einer Funktion $f^A : A^n \to A$.

Die Σ-Algebra $T_\Sigma(X)$ interpretiert f wie folgt: $f^{T_\Sigma(X)}(t_1, \ldots, t_n) =_{def} f(t_1, \ldots, t_n)$.

Für Σ-Algebren A und B heißt eine Funktion $h : A \to B$ **Σ-Homomorphismus**, wenn für alle $f \in \Sigma$ $h \circ f^A = f^B \circ h$ gilt.

Bijektive Σ-Homomorphismen heißen Σ-Isomorphismen.

Die Termfortsetzung $g^* : T_\Sigma(X) \to A$ einer **Belegung** $g : X \to A$ von X in einer Σ-Algebra A ist wie folgt induktiv definiert:

- Für alle $x \in X$, $g^*(x) = g(x)$.
- Für alle $f : n \to 1$ und $t_1, \ldots, t_n \in T_\Sigma(X)$,
$$g^*(f(t_1, \ldots, t_n)) = f^A(g^*(t_1), \ldots, g^*(t_n)).$$

g^* ist der einzige Σ-Homomorphismus von $T_\Sigma(X)$ nach A mit $g^* \circ inc_X = g$.

Folglich ist die Einschränkung $fold^A : T_\Sigma \to A$ von g^* auf Grundterme der einzige Σ-Homomorphismus von $T_\Sigma(X)$ nach A

Belegungen durch Terme heißen **Substitutionen**.

Substitutionslemma (Lemma 5.3): Für alle Σ-Algebren A, Belegungen $g : X \to A$ und Substitutionen $\sigma : X \to T_\Sigma(X)$ gilt $(g^* \circ \sigma)^* = g^* \circ \sigma^*$.

Sei $t_1, \ldots, t_n, t'_1, \ldots, t'_n, t, t' \in T_\Sigma(X)$. Die Formel
$$\varphi = (t_1 \equiv t'_1 \wedge \cdots \wedge t_n \equiv t'_n \Rightarrow t \equiv t')$$
heißt **Σ-Gleichung**.

Eine Σ-Algebra A **erfüllt** φ, falls für alle Belegungen $g : X \to A$ mit $g^*(t_i) = g^*(t'_i)$ für alle $1 \leq i \leq n$ auch $g^*(t) = g^*(t')$ gilt.

Sei E eine Menge von Σ-Gleichungen.

Der **Gleichungskalkül** GK besteht aus den in Abschnitt 5.6 angegebenen fünf Regeln zur Transformation von Gleichungen. Die Relation
$$\equiv_E \;=_{def} \{(t, t') \in T_\Sigma(X)^2 \mid E \vdash_{GK} t \equiv t'\}$$
heißt **E-Äquivalenz**.

Korrektheit von GK (Satz 5.6): Für alle unbedingten Σ-Gleichungen e gilt:
$$E \vdash_{GK} e \;\Rightarrow\; Alg_{\Sigma,E} \models e.$$

Vollständigkeit von GK (Satz 5.12): Für alle unbedingten Σ-Gleichungen e gilt:

$$Alg_{\Sigma,E} \models e \;\Rightarrow\; E \vdash_{GK} e.$$

Die Quotientenalgebra $A = T_\Sigma(X)/\equiv_E$ ist ein Modell von E und jeder unbedingten Gleichung e, die in $Alg_{\Sigma,E}$ gilt:

$$A \models e \;\Leftrightarrow\; Alg_{\Sigma,E} \models e.$$

Zweites Verfahren zur Normalformkonstruktion:

Sei $NF \subseteq T_\Sigma(X)$ die Trägermenge einer Σ-Algebra, inc_X die Inklusion von X in NF, A eine Σ-Algebra, $g : X \to A$ und die Einschränkung von $g^* : T_\Sigma(X) \to A$ auf NF Σ-homomorph. Dann erfüllt A für alle $t \in T_\Sigma(X)$ die Gleichung $t \equiv inc_X^*(t)$, d.h. $inc_X^*(t)$ liefert eine Normalform von t.

6 Aussagenlogik (propositional logic)

Aussagenlogische Formeln (*propositional formulas*) sind logische Verknüpfungen elementarer Sachverhalte, die durch eine Menge At von **Atomen** gegeben sind.

Die Symbole der Signatur AL (siehe Beispiel 5.1) heißen **aussagenlogische Operationen**.

Ein AL-Term über At heißt **aussagenlogische Formel über** At.

Ein AL-Term der Form $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$ oder $\varphi \Rightarrow \psi$ heißt **Negation**, **Konjunktion**, **Disjunktion** bzw. **Implikation**.

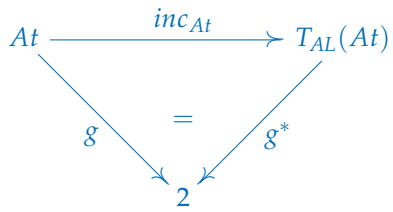
φ und ψ heißen **Faktoren** von $\varphi \wedge \psi$ und **Summanden** von $\varphi \vee \psi$.

$prem(\varphi \Rightarrow \psi) =_{def} \varphi$ heißt **Prämisse** von $\varphi \Rightarrow \psi$.
$conc(\varphi \Rightarrow \psi) =_{def} \psi$ heißt **Konklusion** von $\varphi \Rightarrow \psi$.

Um Klammern zu sparen, vereinbaren wir die folgenden (üblichen) ☞ Prioritäten aussagenlogischen Operationen: Negation vor Konjunktion, Konjunktion vor Disjunktion, Disjunktion vor Implikation.

In den verlinkten Ableitungsbeispielen steht *false* für $\bot$, *true* für $\top$ und – wie in Java – das Ausrufezeichen für $\neg$.

Sei $g : At \to 2$. Den Wahrheitswert einer aussagenlogischen Formel φ unter g erhält man durch Anwendung der Extension $g^* : T_{AL}(At) \to 2$ von g auf φ.



g^* wertet aussagenlogische Formeln in der Bitalgebra aus.

Sei $\varphi, \psi \in T_{AL}(At)$ und $\Phi \subseteq T_{AL}(At)$.

$g : At \to 2$ **erfüllt** φ oder ist ein **Modell von** φ, geschrieben: $g \models \varphi$, wenn $g^*(\varphi) = 1$ gilt.

Darauf aufbauend sind **Erfüllbarkeit**, **Allgemeingültigkeit**, **Folgerung** und **Äquivalenz** aussagenlogischer Formeln wie am Anfang von Kapitel 2 definiert, wobei der zugrundeliegende semantische Bereich D durch die Funktionsmenge 2^{At} gegeben ist.

Aufgaben Sei $\varphi, \psi, \vartheta \in T_{AL}(At)$. Zeigen Sie:

- φ ist genau dann allgemeingültig, wenn φ aus $\top$ folgt. (1)
- φ ist genau dann unerfüllbar, wenn $\bot$ aus φ folgt. (2)
- Folgerungssatz: $\varphi \wedge \psi \models \vartheta$ gilt genau dann, wenn $\varphi \models (\psi \Rightarrow \vartheta)$ gilt. (3)
- Für alle Substitutionen $\sigma : At \to T_{AL}(At)$ gilt: Ist $\varphi\sigma$ erfüllbar, dann ist auch φ erfüllbar. (4)
- φ und ψ sind äquivalent, wenn die AL-Gleichung $\varphi \equiv \psi$ in der Bitalgebra gültig ist (siehe Abschnitt 5.1).

❑

Aus (2) und (3) folgt sofort der Unerfüllbarkeitssatz: Sei $\varphi, \psi \in T_{AL}(At)$.

$$\psi \text{ folgt genau dann aus } \varphi, \text{ wenn } \varphi \wedge \neg\psi \text{ unerfüllbar ist.} \quad (5)$$

Aufgabe Zeigen Sie, dass φ und ψ genau dann äquivalent sind, wenn ψ aus φ und φ aus ψ folgt. ❑

Aufgabe Sei $x, y, z \in At$. Zeigen Sie, dass die Bitalgebra folgende AL-Gleichungen erfüllt:

$x \vee (y \vee z) \equiv (x \vee y) \vee z$	$x \wedge (y \wedge z) \equiv (x \wedge y) \wedge z$	(Assoziativität)
$x \vee y \equiv y \vee x$	$x \wedge y \equiv y \wedge x$	(Kommutativität)
$x \vee x \equiv x$	$x \wedge x \equiv x$	(Idempotenz) ❑

Wegen der Gültigkeit dieser sechs Gleichungen in der Bitalgebra können die Operationen $\vee$ und $\wedge$ ohne innere Klammerung auf mehr als zwei Formeln wie z.B. hier: $\varphi \vee \psi \vee \vartheta$ oder auf eine endliche Formel*menge* wie z.B hier: $\bigvee\{\varphi, \psi, \vartheta\}$ angewendet werden.

$\bigvee \emptyset$ wird mit $\bot$ identifiziert, $\bigwedge \emptyset$ mit $\top$ und $\bigvee\{\varphi\}$ und $\bigwedge\{\varphi\}$ mit φ.

Weitere 2-Äquivalenzen (= in der Bitalgebra gültige AL-Gleichungen)

$x \vee \bot \equiv x$	$x \wedge \top \equiv x$	(Neutralität)
$x \vee \top \equiv \top$	$x \wedge \bot \equiv \bot$	(Annihilation oder Extremalgesetze)
$x \vee (x \wedge y) \equiv x$	$x \wedge (x \vee y) \equiv x$	(Absorption)
$x \vee (y \wedge z) \equiv (x \vee y) \wedge (x \vee z)$	$x \wedge (y \vee z) \equiv (x \wedge y) \vee (x \wedge z)$	(Distributivität)
$x \vee \neg x \equiv \top$	$x \wedge \neg x \equiv \bot$	(Auslöschung oder Komplementarität)
$(x \vee z \equiv y \vee z) \wedge (x \wedge z \equiv y \wedge z)$	$\Rightarrow\ x \equiv y$	(Kürzungsregel)

$\neg\bot \equiv \top$	$\neg\top \equiv \bot$	$\neg\neg x \equiv x$	$\neg(x \vee y) \equiv \neg x \wedge \neg y$	
$\neg(x \wedge y) \equiv \neg x \vee \neg y$		$\neg(x \Rightarrow y) \equiv x \wedge \neg y$	$x \Rightarrow y \equiv \neg x \vee y$	($negAL$)

Die Menge der Assoziativitäts-, Kommutativitäts-, Absorptions-, Distributivitäts- und Auslöschungsgleichungen bezeichnen wir fortan mit BE.

Eine AL-Algebra, die BE erfüllt, heißt **Boolesche Algebra**.

Aufgabe Zeigen Sie, dass jede Potenzmenge eine Boolesche Algebra ist. ❑

Da die Bitalgebra eine Boolesche Algebra ist, ist $\varphi \in T_{AL}(At)$ nach Satz 5.6 allgemeingültig bzw. unerfüllbar, wenn $\varphi \equiv_{BE} \top$ bzw. $\varphi \equiv_{BE} \bot$ gilt.

Idempotenz, Neutralität, Annihilation und alle Gleichungen von $negAL$ (außer denen mit $\Rightarrow$) lassen sich aus BE ableiten, d.h. sie gehören zur BE-Äquivalenz.

Umgekehrt erfüllt jede Boolesche Algebra nach Satz 5.6 alle aus BE ableitbaren Gleichungen. Laut Satz 6.2 (s.u.) gilt sogar jede in der Bitalgebra gültige unbedingte Gleichung auch in jeder anderen Booleschen Algebra, was nach Satz 5.12 impliziert, dass alle diese Gleichungen zur BE-Äquivalenz gehören. Letztere ist also nicht nur in der semantischen Äquivalenz $\equiv^2$ enthalten, sie stimmt sogar mit $\equiv^2$ überein!

Aufgabe Die folgenden AL-Gleichungen $\varphi \equiv \psi$ werden häufig in Beweisen mathematischer Sätze verwendet, indem ψ an Stelle der zu beweisenden Aussage φ gezeigt wird:

Sei $x, y, z, x_1, \ldots, x_n \in At$.

$$\begin{aligned}
x &\equiv \neg x \Rightarrow \bot && \text{(Ableitung eines Widerspruchs)}\\
x \Rightarrow y &\equiv \neg y \Rightarrow \neg x && \text{(Kontraposition)}\\
x \Rightarrow (y \Rightarrow z) &\equiv (x \wedge y) \Rightarrow z && \text{(Dekaskadierung)}\\
x_1 \vee \cdots \vee x_n \Rightarrow x &\equiv (x_1 \Rightarrow x) \wedge \cdots \wedge (x_n \Rightarrow x) && \text{(Zerlegung einer Implikation)}\\
x \Rightarrow x_1 \wedge \cdots \wedge x_n &\equiv (x \Rightarrow x_1) \wedge \cdots \wedge (x \Rightarrow x_n) && \text{(Zerlegung einer Implikation)}\\
x &\equiv (z \Rightarrow x) \wedge (\neg z \Rightarrow x) && \text{(Vollständige Fallunterscheidung)}
\end{aligned}$$

Die Dekaskadierung ähnelt der Isomorphie (6) im Abschnitt Isomorphien auf, wenn man A, B, C durch Formeln, $\times$ durch $\wedge$ und $\to$ durch $\Rightarrow$ ersetzt.

Ersetzt man noch + durch $\vee$, dann werden auch die Kommutativitäts-, Assoziativitäts- und Distributivitätsgesetze im Abschnitt Isomorphien zu gleichnamigen aussagenlogischen Äquivalenzen (s.o).

Diese Analogie zwischen Mengenverknüpfungen einerseits und logischen Operationen andererseits ist unter dem Namen ☞ Curry–Howard Korrespondenz bekannt. ❑

6.1 Normalformen

Eine aussagenlogische Formel heißt **Negationsnormalform (NNF)**, wenn $\Rightarrow$ in ihr nicht vorkommt und $\neg$ nur direkt vor Atomen.

Atome nennt man auch **positive Literale**, während Formeln $\neg\varphi$ mit $\varphi \in At$ **negative Literale** genannt werden. Für alle $x \in At$ heißen x und $\neg x$ **komplementär** zueinander.

Eine Disjunktion von Konjunktionen paarweise verschiedener Literale heißt **disjunktive Normalform (DNF)**.

Eine Konjunktion von Disjunktionen paarweise verschiedener Literale heißt **konjunktive Normalform (KNF)**.

Eine Disjunktion oder Konjunktion von Literalen $\varphi_1, \ldots, \varphi_n$ heißt **geschlossen**, wenn es $1 \leq i, j \leq n$ gibt mit $\varphi_i = \neg\varphi_j$. Andernfalls heißt sie **offen**.

Eine KNF φ ist genau dann allgemeingültig, wenn alle ihre Faktoren geschlossen sind. (6)

Beweis. Da φ eine KNF ist, gibt es Literale $\varphi_{11}, \ldots, \varphi_{1n_1}, \ldots, \varphi_{m1}, \ldots, \varphi_{mn_m}$ mit

$$\varphi = \bigwedge_{i=1}^{m} \bigvee_{j=1}^{n_i} \varphi_{ij}.$$

Sei $e = (x \vee \neg x \equiv \top)$ und $e' = (x \wedge \top \equiv x)$. Ist $\bigvee_{j=1}^{n_i} \varphi_{ij}$ für alle $1 \leq i \leq m$ geschlossen, dann gilt

$$\varphi \quad \equiv_e \quad \top \wedge \cdots \wedge \top \quad \equiv_{e'} \quad \top.$$

Da die Bitalgebra e und e' erfüllt, erfüllt sie nach Satz 5.6 auch die Gleichung $\varphi \equiv \top$.

Sei umgekehrt φ allgemeingültig. Gäbe es $1 \leq i \leq m$ derart, dass $\psi = \bigvee_{j=1}^{n_i} \varphi_{ij}$ offen ist, dann würde $g : At \to 2$ mit $g^{-1}(0) = \{x \in At \mid \exists\, 1 \leq j \leq n_i : x = \varphi_{ij}\}$ ψ, also auch φ nicht erfüllen. ↯

Beweis von $g^*(\psi) = 0$. Sei $1 \leq j \leq n_i$. Für alle Atome x, die in φ_{ij} vorkommen, gilt $g^*(x) = g(x) = 0$. Für alle Literale $\neg z$, die in ψ vorkommen, gilt

$$g^*(\neg z) = 1 - g*(z) = 1 - g(z) = 1 - 0 = 1,$$

weil ψ offen ist und deshalb z nicht zu ψ gehört. ❑

Eine DNF φ ist genau dann unerfüllbar, wenn alle ihre Summanden geschlossen sind. (7)

Beweis. Da φ eine DNF ist, gibt es Literale $\varphi_{11}, \ldots, \varphi_{1n_1}, \ldots, \varphi_{m1}, \ldots, \varphi_{mn_m}$ mit

$$\varphi \;=\; \bigvee_{i=1}^{m} \bigwedge_{j=1}^{n_i} \varphi_{ij}.$$

Sei $e = (x \wedge \neg x \equiv \bot)$ und $e' = (x \vee \bot \equiv x)$. Ist $\bigwedge_{j=1}^{n_i} \varphi_{ij}$ für alle $1 \leq i \leq m$ geschlossen ist, dann gilt

$$\varphi \;\equiv_e\; \bot \vee \cdots \vee \bot \;\equiv_{e'}\; \bot.$$

Sei umgekehrt φ unerfüllbar. Gäbe es $1 \leq i \leq m$ derart, dass $\psi = \bigwedge_{j=1}^{n_i} \varphi_{ij}$ offen ist, dann würde $g : At \to 2$ mit $g^{-1}(1) = \{x \in At \mid \exists\, 1 \leq j \leq n_i : x = \varphi_{ij}\}$ ψ, also auch φ erfüllen. ϟ

Beweis von $g^*(\psi) = 1$. Sei $1 \leq j \leq n_i$. Für alle Atome x, die in φ_{ij} vorkommen, gilt $g^*(x) = g(x) = 1$. Für alle Literale $\neg z$, die in ψ vorkommen, gilt

$$g^*(\neg z) = 1 - g*(z) = 1 - g(z) = 1 - 1 = 0,$$

weil ψ offen ist und deshalb z nicht zu ψ gehört. ❑

Jede Implikation $\varphi \Rightarrow \psi$ mit folgenden Eigenschaften heißt **Gentzenformel über** At (englisch: *sequent*):

- $\varphi = \top$ oder φ ist eine Konjunktion paarweise verschiedener Atome,
- $\psi = \bot$ oder ψ ist eine Disjunktion paarweise verschiedener Atome.

Wegen $(\top \Rightarrow \psi) \equiv^2 \psi$ kürzt man die Gentzenformel $\top \Rightarrow \psi$ oft mit ψ ab.

Wegen $(\varphi \Rightarrow \bot) \equiv^2 \neg\varphi$ kürzt man die Gentzenformel $\varphi \Rightarrow \bot$ oft mit $\neg\varphi$ ab.

Eine Gentzenformel heißt **minimal**, wenn in ihr kein Atom zweimal vorkommt.

Aufgabe Zeigen Sie, dass jede Gentzenformel ist zu $\top$ oder zu einer minimalen Gentzenformel äquivalent ist. ❑

Eine Konjunktion von Gentzenformeln heißt **implikative Normalform** (INF).

Zwei Gentzenformeln, die sich nur in der Anordnung der Atome der Prämisse oder der Konklusion voneinander unterscheiden, werden als gleich angesehen. Deshalb werden sie manchmal als Implikationen zwischen Mengen notiert:

$$\{x_1, \ldots, x_m\} \Rightarrow \{y_1, \ldots, y_n\} \text{ steht dann für } (x_1 \wedge \cdots \wedge x_m) \Rightarrow (y_1 \vee \cdots \vee y_n).$$

Der Vorteil von Gentzenformeln liegt in ihrer "Natürlichkeit" im Sinne der leichten Erfassbarkeit der konkrete Aussage hinter den Formeln. Ebenso entsprechen Anwendungen der Schnittregel (s.u.), aus denen Beweise der Allgemeingültigkeit oder Erfüllbarbarkeit von Gentzenformeln bestehen, eher natürlichen Schlüssen als es z.B. Anwendungen von Gleichungen tun.

Beispiel ☞ Schrittweise Transformation der DNF

$$(\neg x \wedge y \wedge z) \vee (x \wedge \neg y \wedge z) \vee (x \wedge y \wedge \neg z)$$

in eine INF.

Satz 6.1 Sei $\varphi \in T_{AL}(At)$, ψ eine NNF und ϑ eine KNF.

- φ ist zu einer NNF äquivalent. (8)

- ψ ist zu einer DNF äquivalent. (9)
- ψ ist zu einer KNF äquivalent. (10)
- ϑ ist zu einer INF äquivalent. (11)

Beweis von (8). Da jede Teilformel $\neg\varphi$ einer *negAL*-irreduziblen Formel ein Literal ist, sind alle *negAL*-irreduziblen Formeln Negationsnormalformen.

Ein Gewicht einer Formel φ, das Bedingung (10) in Kapitel 5 erfüllt, ist die Summe der Symbole aller Teilformeln $\neg\psi$ oder $\psi \Rightarrow \vartheta$ von φ. Damit ist (8) nach dem ersten Verfahren, äquivalente Normalformen zu berechnen, bewiesen.

Zur Anwendung des zweiten Verfahrens, äquivalente Normalformen zu berechnen, definieren wir *NF* als die Menge aller Negationsnormalformen und interpretieren Σ in *NF* wie folgt: Für alle $c \in \{\bot, \top\}$, $\otimes \in \{\vee, \wedge\}$, $\varphi, \psi \in NF$ und $x \in At$,

$$\begin{array}{lcl}
c^{NF} & = & c, \\
\varphi \otimes^{NF} \psi & = & \varphi \otimes \psi, \\
\varphi \Rightarrow^{NF} \psi & = & \neg^{NF}(\varphi) \vee \psi, \\
\neg^{NF}(\bot) & = & \top, \\
\neg^{NF}(\top) & = & \bot, \\
\neg^{NF}(x) & = & \neg x, \\
\neg^{NF}(\neg\varphi) & = & \varphi, \\
\neg^{NF}(\varphi \vee \psi) & = & \neg^{NF}(\varphi) \wedge \neg^{NF}(\psi), \\
\neg^{NF}(\varphi \wedge \psi) & = & \neg^{NF}(\varphi) \vee \neg^{NF}(\psi), \\
\neg^{NF}(\varphi \Rightarrow \psi) & = & \varphi \wedge \neg^{NF}(\psi).
\end{array}$$

Dass $\neg^{NF}$ induktiv definiert ist, folgt sofort aus der induktiven Definition von *NF* als kleinste Menge aussagenlogischer Formeln mit folgenden Eigenschaften:

- $x \in At \;\Rightarrow\; x, \neg x \in T$,
- $\bot, \top \in T$,
- $\varphi, \psi \in T \;\Rightarrow\; \varphi \vee \psi,\ \varphi \wedge \psi \in T$.

Nach Satz 5.9 ist für jede aussagenlogische Formel t $inc^*_{At}(t)$ eine bzgl. der Bitalgebra zu t äquivalente NNF, falls für alle $g : At \to 2$ die Einschränkung von $g^* : T_{AL}(At) \to A$ *AL*-homomorph ist.

Aufgabe Zeigen Sie das durch strukturelle Induktion über *NF*.

Aufgabe Zeigen Sie auch (9) und (10) durch Anwendung von Satz 5.9. Folgen Sie dem Vorgehen in Beispiel 5.10 oder im Beweis von (8).

Beweis von (11).

Sei ϑ eine KNF. Für alle Faktoren d von ϑ definieren wir:

$$\begin{array}{rcl}
neg(d) & = & \{x \in At \mid \neg x \text{ kommt in } d \text{ vor}\} = \{x_1, \ldots, x_m\}, \\
pos(d) & = & \{x \in At \mid x \text{ kommt in } d \text{ vor}\} = \{y_1, \ldots, y_n\},
\end{array}$$

$$gen(d) \;=\; \begin{cases}
\bot & \text{falls } neg(d) = \emptyset = pos(d), \\
\neg(x_1 \wedge \cdots \wedge x_m) & \text{falls } neg(d) \neq \emptyset,\ pos(d) = \emptyset, \\
y_1 \vee \cdots \vee y_n & \text{falls } neg(d) = \emptyset,\ pos(d) \neq \emptyset, \\
x_1 \wedge \cdots \wedge x_m \Rightarrow y_1 \vee \cdots \vee y_n & \text{sonst.}
\end{cases}$$

Kurz gesagt, die negativen Literale von d bilden, konjunktiv verknüpft, die Prämisse der Gentzenformel $gen(d)$, während die Atome der positiven Literale von d, disjunktiv verknüpft, die Konklusion von $gen(d)$ bilden.

Aufgabe Zeigen Sie, dass ϑ zu $inf(\vartheta)$ äquivalent ist, wobei

$$inf(\vartheta) \quad =_{def} \quad \bigwedge\{gen(d) \mid d \text{ ist Faktor von } \vartheta\}. \qquad \square$$

Zusammengefasst erhält man eine zu einer aussagenlogischen Formel φ äquivalente

- NNF durch Anwendung von *negAL*-Gleichungen oder Faltung von φ in der im Beweis von (8) definierten Normalformalgebra,
- DNF und KNF durch Anwendung von 2-Äquivalenzen auf die NNF von φ;
- INF durch Anwendung der Funktion *inf* auf eine KNF von φ.

Sei die Menge At der Atome endlich. Dann können wir ihre Elemente anordnen:

$$At = \{x_1, \ldots, x_n\}.$$

Folglich gibt es für jede aussagenlogische Formel φ über At eine *eindeutige* Normalform, und zwar in der Menge aller maximalen DNFs:

Eine DNF φ ist eine **maximale DNF (MDNF)**, wenn jeder Summand von φ die Form $f_1(x_1) \wedge \cdots \wedge f_n(x_n)$ hat, wobei für alle $1 \leq i \leq n$ und $x \in At$ $f_i(x) \in \{x, \neg x\}$ gilt.

Mit Hilfe von Neutralitäts-, Komplementaritäts-, Distributivitäts- und Kommutativitätsgleichungen lässt sich jede DNF in eine *BE*-äquivalente MDNF überführen. Deren Eindeutigkeit liefert das folgende – bereits oben erwähnte – Theorem:

Satz 6.2 Jede in der Bitalgebra gültige unbedingte Σ-Gleichung ist in jeder Booleschen Algebra gültig.

Beweis. Nach Satz 5.6 und Satz 5.12 gilt eine *AL*-Gleichung $t \equiv t'$ genau dann in allen Booleschen Algebren, wenn sie zur *BE*-Äquivalenz gehört. Es genügt also Folgendes zu zeigen: Für alle $t, t' \in T_{AL}(At)$,

$$t \equiv^2 t' \quad \Rightarrow \quad t \equiv_{BE} t'. \tag{12}$$

Wir zeigen (12) unter der Annahme, dass für alle MDNFs u, u' Folgendes gilt:

$$u \equiv^2 u' \quad \Rightarrow \quad u = u'. \tag{13}$$

Sei also $t \equiv^2 t'$. Dann erhalten wir zwei MDNFs u, u' mit $t \equiv_{BE} u$ und $t' \equiv_{BE} u'$. Nach Satz 5.6 folgt $t \equiv^2 u$ und $t' \equiv^2 u'$, also auch $u \equiv^2 u'$ und damit $u = u'$ wegen (13). Folglich sind t und t' *BE*-äquivalent, womit (12) bewiesen ist.

Den Beweis von (13) führen wir durch Kontraposition:

Seien u und u' zwei unterschiedliche MDNFs. Dann hat u einen Summanden, den u' nicht hat (Fall 1), oder u' hat einen Summanden, den u nicht hat (Fall 2). Sei $\varphi = f_1(x_1) \wedge \cdots \wedge f_n(x_n)$ dieser Faktor und $g : At \to 2$ definiert durch

$$g^{-1}(1) = \{x \in At \mid \exists\, 1 \leq i \leq n : f_i = id_{At}\}.$$

Im Fall 1 gilt $g^*(u) = 1$ und $g^*(u') = 0$. Im Fall 2 gilt $g^*(u) = 0$ und $g^*(u') = 1$. In beiden Fällen folgt $g^*(u) \neq g^*(u')$, also $u \not\equiv^2 u'$. $\square$

Aufgabe Zeigen Sie unter Verwendung von (13) und Satz 5.6, dass $T_{AL}(At)$ und $\mathcal{P}(At)$ *AL*-isomorph sind. $\square$

Zur Größe von Normalformen

Die mit *negAL* gebildete NNF von φ besteht aus höchstens zweimal so vielen Symbolen wie φ. Bei der Transformation einer NNF zur DNF oder KNF mit Hilfe von Distributivitätsgleichungen werden Teilformeln verdoppelt.

Finden solche Verdopplungen auf mehreren Ebenen der Baumdarstellung von φ statt, dann werden sie bis zu n-mal geschachtelt, wobei n die *Höhe* (maximale Pfadlänge) des Baumes ist. Die DNF bzw. KNF von φ besteht daher aus bis zu 2^n Symbolen.

Aufgabe Zeigen Sie, dass es genau $3^{|At|} + 1$ minimale Gentzenformeln gibt. ❑

6.2 Schnittkalkül

Der **aussagenlogische Schnittkalkül** *ASK* ist synthetisch und besteht aus zwei Regeln:

Sei Φ eine Menge von Gentzenformeln.

Einführungsregel

$$\frac{}{\Phi \vdash \varphi} \qquad \varphi \in \Phi$$

Aussagenlogische Schnittregel

$$\frac{\Phi \vdash \varphi \Rightarrow \psi \vee x, \qquad \Phi \vdash x \wedge \varphi' \Rightarrow \psi'}{\Phi \vdash \varphi \wedge \varphi' \Rightarrow \psi \vee \psi'} \qquad \varphi, \varphi', \psi, \psi' \in T_{AL}(At),\ x \in At$$

Jede zur Gentzenformel des Sukzedenten der Schnittregel äquivalente Gentzenformel heißt **Resolvente** der beiden Gentzenformeln des Antezedenten. Letztere haben ein gemeinsames Atom x, das wegen der Kommutativität von $\vee$ bzw. $\wedge$ nicht notwendig am Ende bzw. Anfang steht.

Eine *ASK*-**Widerlegung** einer Menge Φ von Gentzenformeln ist eine *ASK*-Ableitung von $\Phi \vdash \bot$.

Beispiel Schnittwiderlegung der Gentzenformelmenge $\Phi = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4\}$ mit

$$\begin{aligned} \varphi_1 &= \top \Rightarrow x \vee y \vee z, \\ \varphi_2 &= x \Rightarrow y \vee z, \\ \varphi_3 &= z \Rightarrow y, \\ \varphi_4 &= y \Rightarrow \bot. \end{aligned}$$

Φ	$\vdash$	$\top \Rightarrow x \vee y \vee z$	Einführung von φ_1
Φ	$\vdash$	$x \Rightarrow y \vee z$	Einführung von φ_2
Φ	$\vdash$	$\top \Rightarrow y \vee z$	Resolvente von φ_1 und φ_2
Φ	$\vdash$	$z \Rightarrow y$	Einführung von φ_3
Φ	$\vdash$	$\top \Rightarrow y$	Resolvente von $\top \Rightarrow y \vee z$ und φ_3
Φ	$\vdash$	$y \Rightarrow \bot$	Einführung von φ_4
Φ	$\vdash$	$\top \Rightarrow \bot$	Resolvente von $\top \Rightarrow y$ und φ_4 ❑

Satz 6.3 (Korrektheit von *ASK*-Ableitungen) Sei Φ eine endliche Menge von Gentzenformeln.

(i) Aus $\Phi \vdash_{ASK} \varphi$ folgt $\Phi \models \varphi$.

(ii) Aus $\Phi \vdash_{ASK} \bot$ folgt, dass Φ unerfüllbar ist.

Beweis von (i). Nach Satz 4.1 (iii) gilt die Behauptung, wenn die Menge P aller Paare (Φ, φ) einer Menge Φ von Gentzenformeln und einer einzelnen Gentzenformel φ mit $\Phi \models \varphi$ F_{ASK}-abgeschlossen ist (siehe Abschnitt 4.4), wenn also Folgendes gilt:

$$\text{Aus } \varphi \in \Phi \text{ folgt } \Phi \models \varphi, \tag{14}$$

$$\Phi \models \varphi \Rightarrow \psi \vee x \text{ und } \Phi \models x \wedge \varphi' \Rightarrow \psi' \text{ folgt } \Phi \models \varphi \wedge \varphi' \Rightarrow \psi \vee \psi'. \tag{15}$$

(14) gilt, weil jedes Modell einer Formelmenge ein Modell jedes ihrer Elemente ist.

Es gelte also:

$$\Phi \models \varphi \Rightarrow \psi \vee x \text{ und } \Phi \models x \wedge \varphi' \Rightarrow \psi', \tag{16}$$
$$g : At \to 2 \text{ ist ein Modell von } \Phi, \tag{17}$$
$$g \text{ ist ein Modell von } \varphi \wedge \varphi'. \tag{18}$$

Zu zeigen ist:

$$g \text{ ist ein Modell von } \varphi \wedge \varphi'. \tag{19}$$

Beweis von (19).

Fall 1: $g(x) = 0$. Wegen (18) ist g ein Modell von φ. Wegen (16) und (17) ist g ein Modell von $\varphi \Rightarrow \psi \vee x$. Also ist g ein Modell von $\psi \vee x$. Daraus folgt (19):

$$g^*(\psi \vee \psi') = g^*(\psi) \vee^2 g^*(\psi') = g^*(\psi) \vee^2 0 \vee^2 g^*(\psi')$$
$$= g^*(\psi) \vee^2 g(x) \vee^2 g^*(\psi') = g^*(\psi \vee x) \vee^2 g^*(\psi') = 1 \vee^2 g^*(\psi') = 1.$$

Fall 2: $g(x) = 1$. Wegen (18) ist g ein Modell von φ'. Wegen (16) und (17) ist g ein Modell von $x \wedge \varphi' \Rightarrow \psi'$. Also ist g ein Modell von ψ'. Daraus folgt (19):

$$g^*(\psi \vee \psi') = g^*(\psi) \vee^2 g^*(\psi') = g^*(\psi) \vee^2 1 = 1.$$

Beweis von (ii). Sei $\Phi \vdash_{ASK} \bot$. Aus (i) folgt $\Phi \models \bot$, also ist Φ wegen (2) unerfüllbar. ❑

Aufgabe Zeigen Sie, dass eine Menge von Gentzenformeln ohne $\bot$ erfüllbar ist, wenn die Schnittregel auf keine der Formeln von Φ anwendbar ist.

Geben Sie dazu eine Belegung $g \in 2^{At}$ mit $g^*(\bigwedge \Phi) = 1$ an oder schließen Sie die Behauptung aus folgendem Satz:

Satz 6.4 (Vollständigkeit von *ASK*-Widerlegungen)

Sei Φ eine unerfüllbare endliche Menge von Gentzenformeln. Dann gilt $\Phi \vdash_{ASK} \bot$.

Beweis. Wir zeigen $\Phi \vdash_{ASK} \bot$ durch Induktion über die Anzahl n der Atome von Φ. Im Fall $n = 0$ ist $\Phi = \{\bot\}$. Sei x ein in Φ vorkommendes Atom. Da Φ unerfüllbar ist, sind wegen (4) auch

$$\Phi_\bot = \{\varphi\{\bot/x\} \mid \varphi \in \Phi,\ \varphi\{\bot/x\} \not\equiv^2 \top\} \text{ und}$$
$$\Phi_\top = \{\varphi\{\top/x\} \mid \varphi \in \Phi,\ \varphi\{\top/x\} \not\equiv^2 \top\}$$

unerfüllbar. Da $\Phi_\bot$ und $\Phi_\top$ ein Atom weniger enthalten als Φ, gelten $\Phi_\bot \vdash_{ASK} \bot$ und $\Phi_\top \vdash_{ASK} \bot$ nach Induktionsvoraussetzung.

Sei $nx(\Phi)$ die Mengen aller Gentzenformeln von Φ, die x nicht enthalten. Wie man leicht sieht, lässt sich $\Phi_\bot$ bzw. $\Phi_\top$ wie folgt darstellen:

$$\Phi_\bot = \{\vartheta \Rightarrow \vartheta' \mid \vartheta \Rightarrow \vartheta' \vee x \in \Phi\} \cup nx(\Phi), \tag{20}$$
$$\Phi_\top = \{\vartheta \Rightarrow \vartheta' \mid x \wedge \vartheta \Rightarrow \vartheta' \in \Phi\} \cup nx(\Phi). \tag{21}$$

Aus (20) und $\Phi_\bot \vdash_{ASK} \bot$ folgt $\Phi \vdash_{ASK} \bot$ oder $\Phi \vdash_{ASK} \top \Rightarrow x$ (siehe folgende Aufgabe).

Analog folgt $\Phi \vdash_{ASK} \bot$ oder $\Phi \vdash_{ASK} x \Rightarrow \bot$ aus (21) und $\Phi_\top \vdash_{ASK} \bot$. Zusammengefasst gilt also $\Phi \vdash_{ASK} \bot$ oder sowohl $\Phi \vdash_{ASK} \top \Rightarrow x$ als auch $\Phi \vdash_{ASK} x \Rightarrow \bot$. Da $\bot$ die Resolvente von $\top \Rightarrow x$ und $x \Rightarrow \bot$ ist, ist $\bot$ auch im zweiten Fall aus Φ *ASK*-ableitbar. ❑

Aufgabe Sei $\Phi_\bot \vdash_{ASK} \vartheta \Rightarrow \vartheta'$. Zeigen Sie durch Induktion über die Anzahl der Schnittregelanwendungen einer kürzesten Ableitung von $\Phi_\bot \vdash \vartheta \Rightarrow \vartheta'$ und unter Verwendung von (20), dass $\Phi \vdash \vartheta \Rightarrow \vartheta'$ oder $\Phi \vdash \vartheta \Rightarrow \vartheta' \vee x$

ASK-ableitbar ist. ❑

Die Frage, ob eine endliche Menge Φ von Gentzenformeln unerfüllbar ist. lässt sich mit Hilfe von (7) (siehe Abschnitt 6.1) **entscheiden**: Man bilde die DNF von $\bigwedge \Phi$ und prüfe, ob alle ihre Summanden geschlossen sind.

Einen anderen Algorithmus, der die Unerfüllbarkeit entscheidet, erhält man wie folgt:

Für jede *ASK*-Ableitung $(\Phi_1 \vdash \varphi_1, \ldots, \Phi_n \vdash \varphi_n)$ gilt offenbar $\Phi_1 = \Phi_i$ für alle $1 < i \leq n$.

Darüberhinaus gibt es nur endlich viele *ASK*-Ableitungen $(\Phi_1 \vdash \varphi_1, \ldots, \Phi_n \vdash \varphi_n)$, wenn Φ_1 endlich ist.

Um zu zeigen, dass Φ unerfüllbar ist, konstruiert man nun alle *ASK*-Ableitungen

$$(\Phi \vdash \varphi_1, \ldots, \Phi \vdash \varphi_n).$$

Befindet sich darunter eine mit $\varphi_n = \bot$, dann gilt $\Phi \vdash_{ASK} \bot$, und wir schließen aus Satz 6.3 (ii), dass Φ unerfüllbar ist. Andernfalls gilt offenbar $\Phi \nvdash_{ASK} \bot$, so dass aus Satz 6.4 die Erfüllbarkeit von Φ folgt.

6.3 Knotenmarkierte Bäume

Während mit der Schnittregel INFs transformiert werden, die aus KNFs hervorgehen, werden mit den unten eingeführten Tableauregeln NNFs in Baumdarstellungen (*Tableaus*) ihrer DNFs überführt. Die Summanden einer DNF entsprechen den Pfaden in der Baumdarstellung. Demnach folgt aus (7), dass eine NNF genau dann unerfüllbar ist, wenn das Tableau ihrer DNF aus geschlossenen Pfaden besteht.

Bei der mathematischen Darstellung eines Baumes muss unterschieden werden zwischen seinen Knoten und deren Markierung, die für mehrere Knoten übereinstimmen kann.

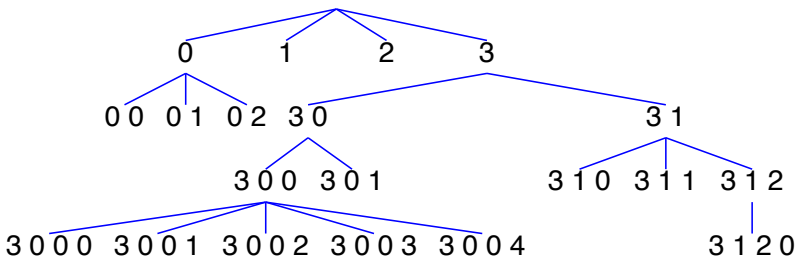
Sei L eine Menge von Markierungen (*labels*). Eine Funktion $t : \mathbb{N}^* \to 1 + L$ heißt **L-markierter Baum**, wenn $t(\epsilon) \in L$ und für alle $w \in \mathbb{N}^*$ und $n \in \mathbb{N}$ Folgendes gilt:

$$w(n+1) \in L \quad \Rightarrow \quad wn \in L, \tag{22}$$

$$wn \in L \quad \Rightarrow \quad w \in L. \tag{23}$$

$w \in \mathbb{N}^*$ heißt **Knoten** (*node*) von t, wenn $t(w)$ zu L gehört. ϵ heißt **Wurzel** von t.

Die Menge $dom(t) =_{def} \{w \in \mathbb{N}^* \mid t(w) \in L\}$ heißt **Domäne** von t. t ist **endlich**, wenn $dom(t)$ endlich ist. Beispiel einer Baumdomäne:



Jedes Element von $dom(t)$ identifiziert eindeutig einen Knoten von t. (22) drückt aus, dass jeder Knoten außer der Wurzel einen Vorgänger in t hat. (23) erlaubt die eindeutige Identifizierung von Geschwisterknoten.

$Tree(L)$ bezeichnet die Menge aller L-markierten Bäume.

Sei $t \in Tree(L)$ und $w \in \mathbb{N}^*$ mit $t(w) \in L$.

$t' \in Tree(L)$ mit $t'(v) = t(wv)$ für alle $v \in \mathbb{N}^*$ heißt **Unterbaum von t mit Wurzel** w. Ist $w \in \mathbb{N}$, dann ist t' ein **maximaler Unterbaum von** t. $w \in \mathbb{N}^*$ heißt **Blatt** von t, wenn für alle $n > 0$ $t(wn) = *$ ist.

Die **Präfixrelation** $\leq$ auf $\mathbb{N}^*$ ist wie folgt definiert: Für alle $v, w \in \mathbb{N}^*$,

$$\begin{aligned} v \leq w &\quad \Leftrightarrow_{def} \quad \text{es gibt } u \in \mathbb{N}^* \text{ mit } vu = w, \\ v < w &\quad \Leftrightarrow_{def} \quad v \leq w \text{ und } v \neq w. \end{aligned}$$

Ist $v \leq w$, dann heißt w von v aus **erreichbar**.

Ist $v < w$, dann heißt v **Vorgänger** von w und w **Nachfolger** von v.

$leaves(t, w)$ bezeichnet die Menge aller von w aus erreichbaren Blätter von t.

$path(v, w) =_{def} \{u \in \mathbb{N}^* \mid v \leq u \leq w\}$ heißt **Pfad** oder **Weg von** v **nach** w.

$\leq$ überträgt sich auf Bäume wie folgt und heißt dort **Subsumptionsrelation**: Für alle L-markierten Bäume t und t',

$$t \leq t' \quad \Leftrightarrow_{def} \quad \text{für alle } w \in f^{-1}(L),\ t(w) = t'(w).$$

6.4 Tableaukalkül

Ein (aussagenlogisches) **Tableau** ist ein endlicher Baum t, dessen Knoten mit aussagenlogischen Formeln markiert sind. t repräsentiert die aussagenlogische Formel

$$form(t) \ = \ \bigvee \{ \bigwedge path(\epsilon, w) \mid w \text{ ist ein Blatt von } t \}.$$

Da die Summe der Längen aller Pfade eines Baums t i.d.R. größer ist als die Anzahl aller Knoten von t, ist die Darstellung einer aussagenlogischen Formel als Tableau platzsparender als die Formel selbst.

Sei φ eine NNF. Eine Folge $(t_1, \ldots, t_n)$ von Tableaus heißt (aussagenlogische) **Tableau-Ableitung von** t_n **aus** φ, wenn

- t_1 ein mit φ markiertes Blatt ist,
- für alle $1 \leq i < n$ t_{i+1} durch Anwendung einer Tableauregel (siehe unten) aus t_i gebildet ist,
- t_n **gesättigt** (*saturated*) ist, d.h. auf t_n ist keine Tableauregel anwendbar, m.a.W.: $form(t_n)$ ist eine DNF.

Enthält $form(t_n)$ eine geschlossene Konjunktion, dann heißt $(t_1, \ldots, t_n)$ **Tableau-Widerlegung**.

Wir schreiben $\varphi \vdash_{\mathcal{T}} t$, falls es eine Tableau-Ableitung von t aus φ gibt.

Aussagenlogische Tableauregeln Sei t ein Tableau und w ein Knoten von t. Wir nennen ihn den **Redex** der jeweiligen Regelanwendung, die t in das neue Tableau t' transformiert.

- **$\vee$-Regel** Sei $t(w) = \varphi \vee \psi$. Für alle $u \in \mathbb{N}^*$,

$$t'(u) \ =_{def} \begin{cases} \top & \text{falls } u = w, \\ \varphi & \text{falls } u = v0 \text{ und } v \in leaves(t, w), \\ \psi & \text{falls } u = v1 \text{ und } v \in leaves(t, w), \\ t(u) & \text{sonst.} \end{cases}$$

- **$\wedge$-Regel** Sei $t(w) = \varphi \wedge \psi$. Für alle $u \in \mathbb{N}^*$,

$$t'(u) \ =_{def} \begin{cases} \top & \text{falls } u = w, \\ \varphi & \text{falls } u = v0 \text{ und } v \in leaves(t, w), \\ \psi & \text{falls } u = v00 \text{ und } v \in leaves(t, w), \\ t(u) & \text{sonst.} \end{cases}$$

Anschaulich gesprochen, bildet die $\vee$- bzw. $\wedge$-Regel t' aus t, indem sie den Knoten w mit $\top$ markiert sowie φ und ψ parallel bzw. hintereinander an alle von w aus erreichbaren Blätter von t anhängt.

Aufgabe Die Gleichungen von *negAL* legen weitere Tableauregeln nahe, die zusammen mit den vier obigen nicht nur NNFs, sondern beliebige aussagenlogische Formeln verarbeiten können. Wie lauten die zusätzlichen Regeln?

Satz 6.5 (Korrektheit aussagenlogischer Tableau-Ableitungen)

Sei $\varphi \vdash_{\mathcal{T}} t$.

(i) $\varphi \equiv^2 form(t)$.

(ii) Alle Summanden von $form(t)$ sind geschlossen. Dann ist φ unerfüllbar.

Beweis. (i): Sei $(t_1, \ldots, t_n)$ eine Tableau-Ableitung von t aus φ. Dann ist $form(t_1) = \varphi$ und $t_n = t$. Es genügt also zu zeigen, dass für alle $1 \leq i < n$

$$form(t_i) \text{ und } form(t_{i+1}) \text{ in der Bitalgebra äquivalent sind.} \tag{19}$$

Aufgabe Zeigen Sie (19).

(ii): Da $form(t)$ eine DNF ist, folgt aus (7), dass $form(t)$ unerfüllbar ist. Also ist wegen (i) auch φ unerfüllbar. ❑

Satz 6.6 (Vollständigkeit aussagenlogischer Tableau-Widerlegungen)

Eine NNF φ sei unerfüllbar. Dann gibt es ein Tableau t mit $\varphi \vdash_{\mathcal{T}} t$ und der Eigenschaft, dass $form(t)$ keine offene Konjunktion von Literalen enthält.

Beweis. Wiederholte Anwendungen der Tableauregeln führen von φ (als einknotigem Tableau t) zu einem gesättigten Tableau t, weil jede Regelanwendung die Knotenanzahl eines Tableaus zwar erhöht, die Gesamtzahl der Vorkommen von $\vee$ und $\wedge$ jedoch verkleinert, so dass irgendwann ein Tableau ohne diese Symbole, also ein gesättigtes Tableau t, erreicht ist.

Sei φ unerfüllbar. Aus Satz 6.5 (i) folgt, dass φ und t in der Bitalgebra äquivalent sind, womit sich die Unerfüllbarkeit von φ auf $form(t)$ überträgt. Da t gesättigt und $form(t)$ unerfüllbar ist, folgt aus (7), dass alle Summanden von $form(t)$ geschlossen sind. ❑

Erfüllbarkeitstest für aussagenlogische NNFs

Sei φ eine NNF. Wende Tableauregeln in beliebiger Reihenfolge auf φ an, bis ein gesättigtes Tableau t erreicht ist.

Prüfe, ob $form(t)$ eine offene Konjunktion von Literalen enthält.

Wenn ja, ist $form(t)$ – wegen (6) – erfüllbar. Wenn nein, ist $form(t)$ – ebenfalls wegen (6) – unerfüllbar. Aus Satz 6.5 (i) folgt, dass sich die Erfüllbarkeit bzw. Unerfüllbarkeit von $form(t)$ auf φ überträgt.

Jede in $form(t)$ enthaltene offene Konjunktion $\varphi_1 \wedge \cdots \wedge \varphi_n$ von Literalen liefert eine Belegung g, die $form(t)$, also auch φ erfüllt: Sei $1 \leq i \leq n$. Setze $g(\varphi_i) = \begin{cases} 1 & \text{falls } \varphi_i \in At, \\ 0 & \text{sonst.} \end{cases}$

Effizienter wird der Test, wenn man nicht erst im gesättigten Tableau nach offenen Konjunktionen sucht, sondern schon vorher Teiltableaus entfernt, deren Pfade komplementäre Literale enthalten. Dazu muss der Kalkül um folgende Regeln erweitert werden, die alle Unterbäume t' von t durch ein Blatt mit der Markierung $\perp$ ersetzen, sofern die Wurzel w von t' mit einem Literal φ und ein Vorgänger von w in t mit dem zu φ komplementären Literal markiert ist:

- **Atomregel** Sei $t(w) = \varphi \in At$. Für alle $u \in \mathbb{N}^*$,

$$t'(u) =_{def} \begin{cases} \perp & \text{falls } u > w \text{ und } t(u) = \neg\varphi, \\ * & \text{falls es } v > w \text{ gibt mit } t(v) = \neg\varphi \text{ und } u > v, \\ t(u) & \text{sonst.} \end{cases}$$

- **Literalregel** Sei $t(w) = \neg\varphi$ und $\varphi \in At$. Für alle $u \in \mathbb{N}^*$,

$$t'(u) =_{def} \begin{cases} \bot & \text{falls } u > w \text{ und } t(u) = \varphi, \\ * & \text{falls es } v > w \text{ gibt mit } t(v) = \varphi \text{ und } u > v, \\ t(u) & \text{sonst.} \end{cases}$$

Außerdem ist in der $\vee$- und der $\wedge$-Regel die Menge $leaves(t, w)$ der von w aus erreichbaren Blätter von t auf diejenigen zu beschränken, die nicht mit $\bot$ markiert sind.

Schließlich wird der Erfüllbarkeitstest für NNFs zu einem Erfüllbarkeitstest für beliebige aussagenlogische Formeln, wenn er die oben erwähnten, aus den Gleichungen von *negAL* gebildeten Tableauregeln mitverwendet.

Beispiele

1. Sei $At = \{x, y, z, x1, y1, z1, x2\}$. ☞ Erfüllbarkeitstest der Formel

$$(((x \wedge y) \vee z) \wedge x1) \vee y1 \vee z1 \vee \neg x2$$

Die Formel ist allgemeingültig, da das letzte Tableau der Ableitung kein mit *false* markiertes Blatt enthält.

2. Sei $At = \{x, y, z\}$. ☞ Erfüllbarkeitstest der Formel

$$(\neg x \wedge (y \vee x) \wedge (y \Rightarrow x)) \vee (x \wedge ((\neg x \wedge y) \vee \neg y))$$

Aus den mit *false* markierten Blättern des letzten Tableaus der Ableitung ergibt sich, dass $g \in 2^{At}$ die Formel genau dann erfüllt, wenn $g(x) = 1$ und $g(y) = 0$ ist.

Wer selbst solche Tableau-Reduktionen erzeugen will, installiere Haskell, lade das Painter-Paket herunter, starte *ghci* mit *Tableau.hs* und berechne Tableaus für aussagenlogische Formeln wie 1. und 2. durch entsprechende Aufrufe der Haskell-Funktion *reduceTab* (siehe Beispiele am Ende von *Tableau.hs*).

6.5 Highlights

Signatur der Aussagenlogik: $AL = \{\bot, \top : 0 \to 1,\ \neg : 1 \to 1,\ \vee, \wedge, \Rightarrow : 2 \to 1\}$

Semantik der Aussagenlogik: AL-Algebra mit Trägermenge $2 = \{0, 1\}$ ("Bitalgebra") und folgender Interpretation von AL:

$$\begin{aligned} \bot^2 &= 0, \\ \top^2 &= 1, \\ \neg^2 &= \lambda b.1 - b, \\ \wedge^2 &= min, \\ \vee^2 &= max, \\ \Rightarrow^2 &= \lambda(b, c).max(1 - b, c) \ = \ \chi(\leq). \end{aligned}$$

$\varphi, \psi \in T_{AL}(X)$ sind **äquivalent**, wenn die AL-Gleichung $\varphi \equiv \psi$ in der Bitalgebra gültig ist.

$g : At \to 2$ **erfüllt** φ oder ist ein **Modell von** φ, geschrieben: $g \models \varphi$, wenn $g^*(\varphi) = 1$ gilt.

Unerfüllbarkeitssatz: ψ folgt genau dann aus φ, wenn $\varphi \wedge \neg\psi$ unerfüllbar ist.

Eine **Boolesche Algebra** ist eine AL-Algebra, die folgende AL-Gleichungen erfüllt:

$x \vee (y \vee z) \equiv (x \vee y) \vee z$	$x \wedge (y \wedge z) \equiv (x \wedge y) \wedge z$	(Assoziativität)
$x \vee y \equiv y \vee x$	$x \wedge y \equiv y \wedge x$	(Kommutativität)
$x \vee (x \wedge y) \equiv x$	$x \wedge (x \vee y) \equiv x$	(Absorption)
$x \vee (y \wedge z) \equiv (x \vee y) \wedge (x \vee z)$	$x \wedge (y \vee z) \equiv (x \wedge y) \vee (x \wedge z)$	(Distributivität)
$x \vee \neg x \equiv \top$	$x \wedge \neg x \equiv \bot$	(Auslöschung)

Atome heißen auch **positive Literale**, Formeln $\neg\varphi$ mit $\varphi \in At$ **negative Literale**.

Eine Disjunktion von Konjunktionen paarweise verschiedener Literale heißt **disjunktive Normalform (DNF)**.

Eine Konjunktion von Disjunktionen paarweise verschiedener Literale heißt **konjunktive Normalform (KNF)**.

Satz 6.1 (9) und (10): Jede aussagenlogische Formel hat äquivalente DNFs und KNFs, wobei BE die Menge der obigen zehn Gleichungen ist.

Eine Disjunktion oder Konjunktion von Literalen $\varphi_1, \ldots, \varphi_n$ heißt **geschlossen**, wenn es $1 \leq i, j \leq n$ gibt mit $\varphi_i = \neg\varphi_j$.

Eine KNF (DNF) φ ist genau dann allgemeingültig (unerfüllbar), wenn alle ihre Faktoren (Summanden) geschlossen sind.

$\top$ sowie jede Implikation $\varphi \Rightarrow \psi$ mit folgenden Eigenschaften heißt **Gentzenformel über** At (englisch: *sequent*):

- $\varphi = \top$ oder φ ist eine Konjunktion paarweise verschiedener Atome,
- $\psi = \bot$ oder ψ ist eine Disjunktion paarweise verschiedener Atome.

Eine Gentzenformel heißt **minimal**, wenn in ihr kein Atom zweimal vorkommt.

Eine Konjunktion von Gentzenformeln heißt **implikative Normalform (INF)**.

Satz 6.1 (11): Jede aussagenlogische Formel hat eine BE-äquivalente INF.

Der **aussagenlogische Schnittkalkül** ASK besteht aus den in Abschnitt 6.2 angegebenen zwei Regeln zur Transformation aussagenlogischer Gentzenformeln. Die Gentzenformel des Sukzedenten der Schnittregel heißt **Resolvente**.

Eine ASK**-Widerlegung** einer Menge Φ von Gentzenformeln ist eine ASK-Ableitung von $\Phi \vdash \bot$.

Korrektheit und Vollständigkeit von ASK**-Ableitungen bzw. -Widerlegungen** (Sätze 6.3 und 6.4): Für endliche Mengen Φ aussagenlogischer Gentzenformeln gilt:

Aus $\Phi \vdash_{ASK} \psi$ folgt $\Phi \models \psi$.

$\Phi \vdash_{ASK} \bot$ gilt genau dann, wenn Φ unerfüllbar ist.

7 Modallogik

Während mit Aussagen- und Prädikatenlogik überall dasselbe gemeint ist, gibt es sowohl von der Syntax her als auch bzgl. der Semantik viele Modallogiken (siehe unten). Allen gemeinsam ist die Möglichkeit, Formeln in Abhängigkeit von verschiedenen Zuständen, Situationen, Welten, etc., auszuwerten. Die (Unterschiede zwischen den) Welten sind entweder in der Realität bereits vorhanden oder – und so wird Modallogik am häufigsten in der Informatik angewendet – werden im Rahmen eines Hardware- oder Softwareentwurfs entwickelt. Dann beschreiben, anschaulich gesprochen, verschiedene Zustände verschiedene Gedächtnisinhalte.

Nehmen wir eine Zahltaste eines Handys, die je nachdem wie oft sie gedrückt wird, die Buchstaben A, B oder C anzeigt. Wie funktioniert das? Jeder Druck auf die Taste bringt das Handy in einen neuen Zustand. Die Gültigkeit elementarer Aussagen wie "A wird angezeigt" oder "B wird angezeigt" hängt vom aktuellen Zustand ab, der selbst vom Vorgängerzustand abhängt, usw.

Deshalb wird die Bedeutung einer modallogischen Formel nicht nur von einer zustandsabhängigen Belegung der Atome bestimmt, sondern auch von einer Transitionsfunktion, die allen im jeweiligen System möglichen Zustandsübergänge definiert.

Die hier behandelte Modallogik baut direkt auf der Aussagenlogik auf und erweitert diese um die einstelligen **modalen Operationen** $\Diamond$ (*diamond*) und $\Box$ (*box*), die es erlauben, Aussagen zu formulieren, deren Gültigkeit im gegenwärtigen Zustand von der Gültigkeit einer anderen Aussage in zukünftigen Situationen bestimmt wird.

Sei $ML = AL \cup \{\Box, \Diamond : 1 \to 1\}$.

Ein ML-Term über At heißt **modallogische Formel über** At.

Die Bedeutung modallogischer Formeln hängt ab von einer **Kripke-Struktur** genannten Funktion

$$\mathcal{K} : State \to (\mathcal{P}(State) \times \mathcal{P}(At)),$$

die jedem – manchmal auch "Welt" genannten – Zustand $s \in State$ dessen direkten Folgezustände sowie die Atome zuordnet, die s erfüllt.

Wegen Isomorphie (14) für $n = 2$ (siehe Kapitel 3) kann $\mathcal{K}$ geschrieben werden als Produktextension $\langle \delta, \beta \rangle$ zweier Funktionen

$$\delta : State \to \mathcal{P}(State) \quad \text{und} \quad \beta : State \to \mathcal{P}(At).$$

δ heißt auch **Kripke-Rahmen** oder **Transitionsfunktion** von $\mathcal{K}$ und bestimmt die Interpretation der modalen Operationen von ML, während β die Belegung $g_{\mathcal{K}}$ (s.u.) der Atome in $\mathcal{P}(State)$ festlegt.

Kripke-Strukturen und andere zustandsbasierte Modelle sind – im Gegensatz zu induktiv definierten Mengen – *black-box-Modelle*, weil der Aufbau ihrer einzelnen Elemente verborgen bleibt. Die Identität eines Zustands kann nur durch *Beobachtung* seines *Verhaltens* ermittelt werden, das hier durch β und δ bestimmt wird.

An die Stelle der Bitalgebra mit Trägermenge 2, auf der wir die Semantik aussagenlogischer Formeln aufgebaut haben, tritt jetzt eine ML-Algebra mit Trägermenge $\mathcal{P}(State)$, die wir $Pow(\delta)$ nennen, weil die Interpretation der Signatur ML in dieser Algebra nicht nur von $State$, sondern auch von δ abhängt:

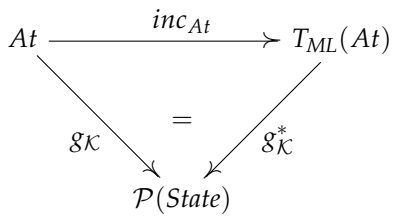
Für alle $S, S' \subseteq State$,

$$\begin{aligned}
\bot^{Pow(\delta)} &= \emptyset, \\
\top^{Pow(\delta)} &= State, \\
\neg^{Pow(\delta)}(S) &= State \setminus S, \\
\wedge^{Pow(\delta)}(S, S') &= S \cap S', \\
\vee^{Pow(\delta)}(S, S') &= S \cup S', \\
\Rightarrow^{Pow(\delta)}(S, S') &= (State \setminus S) \cup S', \\
\Box^{Pow(\delta)}(S) &= \{s \in State \mid \delta(s) \subseteq S\} = \{s \in State \mid \forall\, s' \in \delta(s) : s' \in S\}, \\
\Diamond^{Pow(\delta)}(S) &= \{s \in State \mid \delta(s) \cap S \neq \emptyset\} = \{s \in State \mid \exists\, s' \in \delta(s) : s' \in S\}.
\end{aligned}$$

β bestimmt die Belegung $g_{\mathcal{K}} : At \to \mathcal{P}(State)$, unter der modallogische Formeln in $Pow(\delta)$ – durch Anwendung der aus $g_{\mathcal{K}}$ abgeleiteten Auswertungsfunktion $g^*_{\mathcal{K}}$ (siehe Kapitel 5) – ausgewertet werden:

Für alle $x \in At$ und $s \in State$,

$$g_{\mathcal{K}}(x) =_{def} \{s \in State \mid x \in \beta(s)\}.$$



Sei $\varphi \in T_{ML}(At)$ und $s \in State$. Das Paar (φ, s) heißt **Zustandsformel**.

$(\mathcal{K}, s)$ ist ein **Modell** von φ, geschrieben: $\mathcal{K} \models_s \varphi$, wenn s zu $g^*_{\mathcal{K}}(\varphi)$ gehört, in Worten: wenn φ im Zustand s gilt.

Darauf aufbauend sind **Erfüllbarkeit**, **Allgemeingültigkeit**, **Folgerung** und **Äquivalenz** modallogischer Formeln wie am Anfang von Kapitel 2 definiert, wobei der zugrundeliegende semantische Bereich D durch die Menge aller Paare gegeben ist, die aus einer Kripke-Struktur $\mathcal{K} : State \to (\mathcal{P}(State) \times \mathcal{P}(At))$ und einem Zustand $s \in State$ bestehen.

Aufgabe Zeigen Sie (1)-(4) aus Kapitel 6 für modallogische Formeln. ❑

$\mathcal{K} = \langle \delta, \beta \rangle$ heißt **endlich verzweigt** (*finitely branching*), wenn für alle $s \in State$ die Menge $\delta(s)$ endlich ist.

$\mathcal{K} = \langle \delta, \beta \rangle$ heißt **modal gesättigt** (*modally saturated*), wenn für alle $s \in State$ und $\Phi \subseteq T_{ML}(At)$ eine der beiden folgenden äquivalenten Bedingungen gilt (vgl. [5], Def. 53; [9], Def. 6.9; [11], Def. 5 und Lemma 3):

$$\forall\, t \in \delta(s)\ \exists\, \varphi \in \Phi : \mathcal{K} \models_t \varphi \quad \Rightarrow \quad \text{Es gibt eine endliche Teilmenge } \Phi' \text{ von } \Phi \text{ mit } \mathcal{K} \models_s \Box \bigvee \Phi'. \tag{1}$$

$$\text{Für alle endlichen Teilmengen } \Phi' \text{ von } \Phi \text{ gilt } \mathcal{K} \models_s \Diamond \bigwedge \Phi' \quad \Rightarrow \quad \exists\, t \in \delta(s)\ \forall\, \varphi \in \Phi : \mathcal{K} \models_t \varphi. \tag{2}$$

(2) ist tatsächlich die Kontraposition von (1) – wenn Φ in (2) durch $\{\neg\varphi \mid \varphi \in \Phi\}$ ersetzt wird.

$\mathcal{K}$ endlich verzweigt $\Rightarrow$ $\mathcal{K}$ modal gesättigt: Ist $\mathcal{K}$ endlich verzweigt und gilt die Prämisse von (1), dann ist $\delta(s)$ endlich und für alle $t \in \delta(s)$ gibt es $\varphi_t \in \Phi$ mit $\mathcal{K} \models_t \varphi_t$. Also erfüllt $\Phi' = \{\varphi_t \mid t \in \delta(s)\}$ die Konklusion von (1).

7.1 Übersetzung modallogischer in aussagenlogische Formeln

Ist $\mathcal{K}$ endlich verzweigt, dann lässt sich die Frage, ob $\mathcal{K}$ eine modallogische Formel φ im Zustand s erfüllt, dadurch beantworten, dass man die Zustandsformel (φ, s) in eine aussagenlogische Formel über $At \times State$ übersetzt und

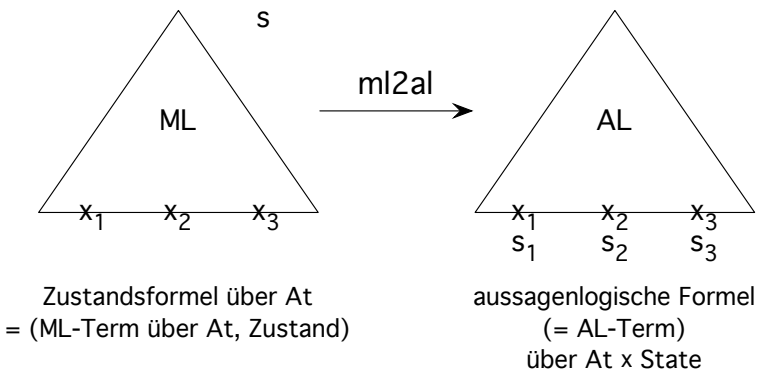
anschließend in der Bitalgebra auswertet. Der Übersetzer

$$ml2al : T_{ML}(At) \times State \to T_{AL}(At \times State)$$

ist wie folgt induktiv definiert:

Für alle $x \in At$, $s \in State$, $c \in \{\bot, \top\}$, $\otimes \in \{\vee, \wedge, \Rightarrow\}$ und $\varphi, \psi \in T_{ML}(At)$,

$$\begin{aligned}
ml2al(x,s) &= (x,s),\\
ml2al(c,s) &= c,\\
ml2al(\neg\varphi,s) &= \neg ml2al(\varphi,s),\\
ml2al(\varphi \otimes \psi,s) &= ml2al(\varphi,s) \otimes ml2al(\psi,s),\\
ml2al(\Box\varphi,s) &= \bigwedge\{ml2al(\varphi,s') \mid s' \in \delta(s)\},\\
ml2al(\Diamond\varphi,s) &= \bigvee\{ml2al(\varphi,s') \mid s' \in \delta(s)\}.
\end{aligned}$$



Auf semantischer Ebene wird eine Kripke-Struktur $\mathcal{K}$ in die aussagenlogische Belegung

$$h_{\mathcal{K}} =_{def} uncurry(\chi \circ g_{\mathcal{K}}) : At \times State \to 2$$

transformiert.

Satz 7.1 (Korrektheit von *ml2al*)

Für alle Kripke-Strukturen $\mathcal{K}$ mit Zustandsmenge $State$, $s \in State$ und modallogischen Formeln φ gilt:

$$\mathcal{K} \models_s \varphi \quad \Leftrightarrow \quad h_{\mathcal{K}} \models ml2al(\varphi,s), \tag{1}$$

in Worten: $\mathcal{K}$ erfüllt die modallogische Formel φ im Zustand s genau dann, wenn die aussagenlogische Belegung $h_{\mathcal{K}}$ die aussagenlogische Formel $ml2al(\varphi,s)$ erfüllt.

Beweis. Sei $s \in State$. Nach Definition der modal- bzw. aussagenlogischen Erfüllbarkeitsrelation ist (1) äquivalent zu:

$$s \in g^*_{\mathcal{K}}(\varphi) \;\Leftrightarrow\; h^*_{\mathcal{K}}(ml2al(\varphi,s)) = 1. \tag{2}$$

Wir zeigen (2) durch strukturelle Induktion über $T_{ML}(At)$.

Für alle $x \in At$,

$$\begin{aligned}
s \in g^*_{\mathcal{K}}(x) &\;\Leftrightarrow\; s \in g_{\mathcal{K}}(x)\\
&\;\Leftrightarrow\; h^*_{\mathcal{K}}(ml2al(x,s)) = h^*_{\mathcal{K}}(x,s) = uncurry(\chi \circ g_{\mathcal{K}})^*(x,s)\\
&\quad = uncurry(\chi \circ g_{\mathcal{K}})(x,s) = (\chi \circ g_{\mathcal{K}})(x)(s) = \chi(g_{\mathcal{K}}(x))(s) = 1,\\
s \in g^*_{\mathcal{K}}(\bot) = \emptyset &\;\Leftrightarrow\; h^*_{\mathcal{K}}(ml2al(\bot,s)) = h^*_{\mathcal{K}}(\bot) = 0,\\
s \in g^*_{\mathcal{K}}(\top) = State &\;\Leftrightarrow\; h^*_{\mathcal{K}}(ml2al(\bot,s)) = h^*_{\mathcal{K}}(\top) = 1.
\end{aligned}$$

Für alle $\varphi, \psi \in T_{ML}(At)$,

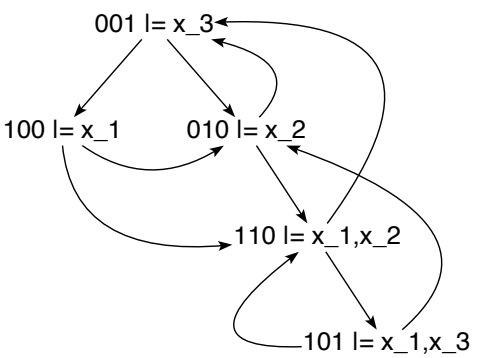
$$
\begin{aligned}
s \in g^*_{\mathcal{K}}(\neg\varphi) &\Leftrightarrow s \in State \setminus g^*_{\mathcal{K}}(\varphi) \\
&\Leftrightarrow h^*_{\mathcal{K}}(ml2al(\neg\varphi, s)) = h^*_{\mathcal{K}}(\neg ml2al(\varphi, s)) = 1 - h^*_{\mathcal{K}}(ml2al(\varphi, s))) \\
&\overset{ind.\ hyp.}{=} 1 - 0 = 1, \\
s \in g^*_{\mathcal{K}}(\varphi \wedge \psi) &\Leftrightarrow s \in g^*_{\mathcal{K}}(\varphi) \cap g^*_{\mathcal{K}}(\psi) \\
&\Leftrightarrow h^*_{\mathcal{K}}(ml2al(\varphi \wedge \psi, s)) = h^*_{\mathcal{K}}(ml2al(\varphi, s) \wedge ml2al(\psi, s)) \\
&= min\{h^*_{\mathcal{K}}(ml2al(\varphi, s)), h^*_{\mathcal{K}}(ml2al(\psi, s))\} \overset{ind.\ hyp.}{=} 1, \\
s \in g^*_{\mathcal{K}}(\varphi \vee \psi) &\Leftrightarrow s \in g^*_{\mathcal{K}}(\varphi) \cup g^*_{\mathcal{K}}(\psi) \\
&\Leftrightarrow h^*_{\mathcal{K}}(ml2al(\varphi \vee \psi, s)) = h^*_{\mathcal{K}}(ml2al(\varphi, s) \vee ml2al(\psi, s)) \\
&= max\{h^*_{\mathcal{K}}(ml2al(\varphi, s)), h^*_{\mathcal{K}}(ml2al(\psi, s))\} \overset{ind.\ hyp.}{=} 1, \\
s \in g^*_{\mathcal{K}}(\varphi \Rightarrow \psi) &\Leftrightarrow s \in (State \setminus g^*_{\mathcal{K}}(\varphi)) \cup g^*_{\mathcal{K}}(\psi) \\
&\Leftrightarrow h^*_{\mathcal{K}}(ml2al(\varphi \Rightarrow \psi, s)) = h^*_{\mathcal{K}}(ml2al(\varphi, s) \Rightarrow ml2al(\psi, s)) \\
&= max\{1 - h^*_{\mathcal{K}}(ml2al(\varphi, s)), h^*_{\mathcal{K}}(ml2al(\psi, s))\} \overset{ind.\ hyp.}{=} 1, \\
s \in g^*_{\mathcal{K}}(\Box\varphi) &\Leftrightarrow \delta(s) \subseteq g^*_{\mathcal{K}}(\varphi) \Leftrightarrow \forall\, s' \in \delta(s) : s' \in g^*_{\mathcal{K}}(\varphi) \\
&\Leftrightarrow h^*_{\mathcal{K}}(ml2al(\Box\varphi, s)) = h^*_{\mathcal{K}}(\bigwedge\{ml2al(\varphi, s') \mid s' \in \delta(s)\}) \\
&= min\{h^*_{\mathcal{K}}(ml2al(\varphi, s')) \mid s' \in \delta(s)\} \overset{ind.\ hyp.}{=} min\{1, \ldots, 1\} = 1, \\
s \in g^*_{\mathcal{K}}(\Diamond\varphi) &\Leftrightarrow \delta(s) \cap g^*_{\mathcal{K}}(\varphi) \neq \emptyset \Leftrightarrow \exists\, s' \in \delta(s) : s' \in g^*_{\mathcal{K}}(\varphi) \\
&\Leftrightarrow h^*_{\mathcal{K}}(ml2al(\Diamond\varphi, s)) = h^*_{\mathcal{K}}(\bigvee\{ml2al(\varphi, s') \mid s' \in \delta(s)\}) \\
&= max\{h^*_{\mathcal{K}}(ml2al(\varphi, s')) \mid s' \in \delta(s)\} \overset{ind.\ hyp.}{=} max\{\ldots, 1, \ldots\} = 1. \qquad \text{❑}
\end{aligned}
$$

Beispiel 7.2 "Murmelspiel"

Sei $State = 2^3$, $At = \{x_1, x_2, x_3\}$, $\mathcal{K} = \langle \delta, \beta \rangle : State \to \mathcal{P}(State) \times \mathcal{P}(At)$ mit

$$
\begin{aligned}
&\delta(001) = \{100, 010\},\ \delta(010) = \{001, 110\},\ \delta(100) = \{110, 010\}, \\
&\delta(110) = \{001, 101\},\ \delta(101) = \{010, 110\},\ \delta(000) = \delta(011) = \delta(111) = \emptyset
\end{aligned}
$$

und für alle $a_1a_2a_3 \in State$, $\beta(a_1a_2a_3) = \{x_i \mid a_i = 1,\ 1 \leq i \leq 3\}$. Die Werte von x_1, x_2, x_3 repräsentieren die jeweiligen Stellungen dreier Weichen auf dem Weg einer Murmel (siehe [38], Abschnitt B 5.1).



Von ☞ Expander2 erzeugter Transitionsgraph von $\mathcal{K}$ inkl. die Zustände erfüllende Atome

Dann gilt

$$
\begin{aligned}
&ml2al(\Diamond\Diamond\Box x_3, 001)\\
&= ml2al(\Diamond\Box x_3, 100) \vee ml2al(\Diamond\Box x_3, 010)\\
&= ml2al(\Box x_3, 110) \vee ml2al(\Box x_3, 010) \vee ml2al(\Box x_3, 001) \vee ml2al(\Box x_3, 110)\\
&= ml2al(\Box x_3, 110) \vee ml2al(\Box x_3, 010) \vee ml2al(\Box x_3, 001)\\
&= ml2al(x_3, 001) \wedge ml2al(x_3, 101) \vee ml2al(x_3, 001) \wedge ml2al(x_3, 110)\\
&\quad \vee ml2al(x_3, 100) \wedge ml2al(x_3, 010)\\
&= (x_3, 001) \wedge (x_3, 101) \vee (x_3, 001) \wedge (x_3, 110) \vee (x_3, 100) \wedge (x_3, 010) \qquad (3)
\end{aligned}
$$

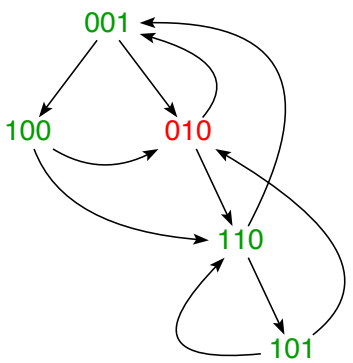
Die o.g. Belegung $h_{\mathcal{K}} : At \times State \to 2$ (s.o.) hat hier folgende Werte: Für alle $1 \leq i \leq 3$ und $a_1a_2a_3 \in State$,

$$
\begin{aligned}
&h_{\mathcal{K}}(x_i, a_1a_2a_3) = uncurry(\chi \circ g_{\mathcal{K}})(x_i, a_1a_2a_3) = (\chi \circ g_{\mathcal{K}})(x_i)(a_1a_2a_3)\\
&= \chi(g_{\mathcal{K}}(x_i))(a_1a_2a_3) = \chi(\{b_1b_2b_3 \in State \mid x_i \in \beta(b_1b_2b_3)\})(a_1a_2a_3)\\
&= \chi(\{b_1b_2b_3 \in State \mid b_i = 1\})(a_1a_2a_3) = a_i. \qquad (4)
\end{aligned}
$$

Daraus folgt:

$$
\begin{aligned}
&h^*_{\mathcal{K}}(ml2al(\Diamond\Diamond\Box x_3, 001))\\
&\overset{(3)}{=} h^*_{\mathcal{K}}((x_3, 001) \wedge (x_3, 101) \vee (x_3, 001) \wedge (x_3, 110) \vee (x_3, 100) \wedge (x_3, 010)\\
&= h^*_{\mathcal{K}}(x_3, 001) \wedge^2 h^*_{\mathcal{K}}(x_3, 101) \vee^2 h^*_{\mathcal{K}}(x_3, 001) \wedge^2 h^*_{\mathcal{K}}(x_3, 110)\\
&\quad \vee^2 h^*_{\mathcal{K}}(x_3, 100) \wedge^2 h^*_{\mathcal{K}}(x_3, 010)\\
&= h_{\mathcal{K}}(x_3, 001) \wedge^2 h_{\mathcal{K}}(x_3, 101) \vee^2 h_{\mathcal{K}}(x_3, 001) \wedge^2 h_{\mathcal{K}}(x_3, 110)\\
&\quad \vee^2 h_{\mathcal{K}}(x_3, 100) \wedge^2 h_{\mathcal{K}}(x_3, 010)\\
&\overset{(4)}{=} 1 \wedge^2 1 \vee^2 1 \wedge^2 0 \vee^2 0 \wedge^2 0 = 1.
\end{aligned}
$$

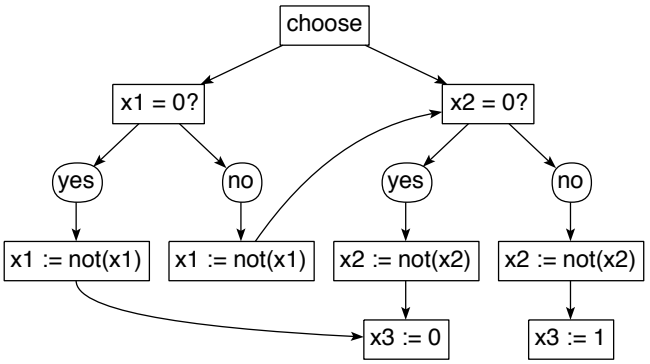
Also ist $(\mathcal{K}, 001)$ nach Satz 7.1 ein Modell von $\Diamond\Diamond\Box x_3$.



Transitionsgraph von $\mathcal{K}$ mit grüner Markierung aller Zustände von $\mathcal{K}$, die $\Diamond\Diamond\Box x_3$ erfüllen, also aller Elemente von $g^*_{\mathcal{K}}(\Diamond\Diamond\Box x_3)$

Das Beispiel ist typisch für die Bildung vieler Zustandsmengen: Jeder Zustand ist ein Tupel von Werten (hier: 0 oder 1) einzelner Atome (hier: x_1, x_2 oder x_3). Man nennt solche Atome auch **Zustandsvariablen**.

Die Transitionsfunktion δ lässt sich dann auch als Flussgraph darstellen, dessen Pfade von der Wurzel zu einem Blatt alle möglichen Zustandsänderungen wiedergeben, und zwar in Form der Zuweisungen auf dem jeweiligen Pfad. Im Beispiel sieht der Flussgraph folgendermaßen aus:



❑

Aufgabe Sei $\mathcal{K} : State \to (\mathcal{P}(State) \times \mathcal{P}(At))$ eine Kripke-Struktur und $x, y \in At$. Zeigen Sie, dass die *ML*-Algebra $Pow(\delta)$ alle in der Bitalgebra gültigen *AL*-Gleichungen und darüberhinaus folgende *ML*-Gleichungen erfüllt:

$$\neg\Diamond x \equiv \Box\neg x \qquad \neg\Box x \equiv \Diamond\neg x \qquad (negM)$$
$$\Diamond(x \vee y) \equiv \Diamond x \vee \Diamond y \qquad \Box(x \wedge y) \equiv \Box x \wedge \Box y \qquad (distDia / distBox)$$

Eine modallogische Formel heißt **Negationsnormalform über** At (NNF), wenn $\Rightarrow$ in ihr nicht vorkommt und $\neg$ nur direkt vor Atomen.

Der Beweis von Satz 6.1 (8) lässt sich leicht zu einem Beweis dafür fortsetzen, dass jede modallogische Formel über At zu einer NNF $(BE \cup negM)$-äquivalent ist.

7.2 Tableaukalkül

Fortan nennen wir eine Zustandsformel (φ, s) **Zustandsatom** bzw. **Zustandsliteral**, wenn φ ein Atom bzw. Literal ist.

$\mathcal{K} = \langle \delta, \beta \rangle$ **erfüllt** eine **Transition** (s, s'), geschrieben: $\mathcal{K} \models (s, s')$, wenn s' zu $\delta(s)$ gehört.

Die Erweiterung des aussagenlogischen Tableaukalküls um Regeln für die modalen Operationen $\Diamond$ und $\Box$ dient dem Beweis der Erfüllbarkeit einer gegebenen Zustandsformel. δ wird schrittweise im Laufe des Ableitungsprozesses aufgebaut.

Demgegenüber geht **Model checking** von einer Kripke-Struktur $\mathcal{K}$ aus und prüft die modallogischer Formeln für genau diese Kripke-Struktur. Hier kann der Tableaukalkül dazu dienen festzustellen, ob es für eine gegebene Formel φ überhaupt einen Zustand s mit $\mathcal{K} \models_s \varphi$ geben kann. Diese Frage lässt sich allerdings i.d.R. einfacher beantworten, indem man $g^*_{\mathcal{K}}(\varphi)$ auf Leerheit testet.

Der Tableaukalkül wird deshalb eher in einem Szenario angewendet, in dem die Details der vom Kalkül konstruierten Kripke-Struktur keine Rolle spielen. Dann werden $\Box$ und $\Diamond$ meistens auch ohne direkte Verwendung einer Transitionsfunktion interpretiert; stattdessen werden $\Box\varphi$ und $\Diamond\varphi$ als *Möglichkeit* bzw. *Notwendigkeit* der Gültigkeit von φ betrachtet.

Sei $State$ eine mindestens abzählbar unendliche Menge von Zuständen. Für alle Zustandsformeln φ nennen wir den Ausdruck $box(\varphi)$ eine **besuchte Box-Formel**.

Ein (modallogisches) **Tableau** ist ein endlicher Baum t, dessen Knoten mit $\top$, einer Transition, einer Zustandsformel oder einer besuchten Box-Formel markiert sind (siehe Knotenmarkierte Bäume).

Ein **Knoten** w von t heißt **geschlossen**, wenn es zwei Knoten $u \leq w$ und $v \leq w$ gibt mit $\pi_1(t(u)) \equiv^2 \neg\pi_1(t(v))$ und $\pi_2(t(u)) = \pi_2(t(v))$. w heißt **offen**, wenn w nicht geschlossen ist.

t heißt **geschlossen**, wenn alle Blätter von t geschlossen sind. t heißt **offen**, wenn t nicht geschlossen ist.

Eine Kripke-Struktur $\mathcal{K}$ **erfüllt** t, geschrieben: $\mathcal{K} \models t$, wenn es ein Blatt w von t gibt mit $\mathcal{K} \models \varphi$ für alle Zustandsformeln und Transitionen $\varphi \in t(path(\epsilon, w))$.

Eine Folge $(t_1, \ldots, t_n)$ von Tableaus heißt (modallogische) **Tableau-Ableitung von t_n aus t_1**, wenn

- t_1 ein mit einer Zustandsformel markiertes Blatt ist,
- für alle $1 < i \leq n$ t_i durch Anwendung einer Tableauregel (siehe unten) aus t_{i-1} gebildet ist,
- t_n **gesättigt** (*saturated*) ist, d.h. auf t_n ist keine Tableauregel anwendbar, m.a.W.: alle noch nicht besuchten Zustandsformeln von t_n sind Zustandsliterale.

Ist t_n geschlossen, dann heißt $(t_1, \ldots, t_n)$ **Tableau-Widerlegung**.

Wir schreiben $t \vdash_{\mathcal{T}} t'$, falls es eine Tableau-Ableitung von t' aus t gibt.

Für alle $s \in State$ und $w \in \mathbb{N}^*$ sei

$$\begin{aligned} boxes(s,t,w) &= \{\varphi \in T_{ML}(At) \mid \text{es gibt } v \leq w \text{ mit } t(v) = box(\varphi, s)\}, \\ succs(s,t,w) &= \{s' \in State \mid \text{es gibt } v \leq w \text{ mit } t(v) = (s, s')\}. \end{aligned}$$

- φ gehört genau dann zu $boxes(s,t,w)$, wenn $box(s,\varphi)$ einen Vorgänger von w markiert.
- s' gehört genau dann zu $succs(s,t,w)$, wenn (s,s') einen Vorgänger von w markiert.

Modallogische Tableauregeln Sei t ein Tableau und w ein Knoten von t. Wir nennen ihn den **Redex** der jeweiligen Regelanwendung, die t in das neue Tableau t' transformiert.

- **$\vee$-Regel** Sei $t(w) = (\varphi \vee \psi, s)$. Für alle $u \in \mathbb{N}^*$,

$$t'(u) =_{def} \begin{cases} \top & \text{falls } u = w, \\ (\varphi, s) & \text{falls } u = v0 \text{ und } v \in leaves(t,w), \\ (\psi, s) & \text{falls } u = v1 \text{ und } v \in leaves(t,w), \\ t(u) & \text{sonst.} \end{cases}$$

- **$\wedge$-Regel** Sei $t(w) = (\varphi \wedge \psi, s)$. Für alle $u \in \mathbb{N}^*$,

$$t'(u) =_{def} \begin{cases} \top & \text{falls } u = w, \\ (\varphi, s) & \text{falls } u = v0 \text{ und } v \in leaves(t,w), \\ (\psi, s) & \text{falls } u = v00 \text{ und } v \in leaves(t,w), \\ t(u) & \text{sonst.} \end{cases}$$

- **$\Diamond$-Regel** Sei $t(w) = (\Diamond\varphi, s)$ und $s' \in State$ komme in t nicht vor. Für alle $u \in \mathbb{N}^*$,

$$t'(u) =_{def} \begin{cases} \top & \text{falls } u = w, \\ (s, s') & \text{falls } u = v0 \text{ und } v \in leaves(t,w), \\ (\varphi, s') & \text{falls } u = v00 \text{ und } v \in leaves(t,w), \\ (\varphi_i, s') & \text{falls } u = v000^i, v \in leaves(t,w), \\ & \{\varphi_1, \ldots, \varphi_n\} = boxes(s,t,v) \text{ und } 1 \leq i \leq n, \\ t(u) & \text{sonst.} \end{cases}$$

- **$\Box$-Regel** Sei $t(w) = (\Box\varphi, s)$. Für alle $u \in \mathbb{N}^*$,

$$t'(u) =_{def} \begin{cases} box(\varphi, s) & \text{falls } u = w, \\ (\varphi, s_i) & \text{falls } u = v0^i, v \in leaves(t,w), \\ & \{s_1, \ldots, s_n\} = succs(s,t,v) \text{ und } 1 \leq i \leq n, \\ t(u) & \text{sonst.} \end{cases}$$

Aufgabe Die Gleichungen von *negAL* und *negM* legen weitere Tableauregeln nahe, die zusammen mit den beiden obigen nicht nur NNFs, sondern beliebige modallogische Formeln verarbeiten können. Wie lauten die zusätzlichen Regeln?

Lemma 7.3 Ein gesättigtes Tableau t ist genau dann erfüllbar, wenn es offen ist.

Beweis. Sei t erfüllbar. Dann gibt es eine Kripke-Struktur

$$\mathcal{K} : State \to \mathcal{P}(State) \times \mathcal{P}(At)$$

und ein Blatt w von t, das von $\mathcal{K}$ erfüllt wird.

Seien $\Phi = \{(x_1, s_1), \ldots (x_m, s_m)\}$ und $\{(\neg y_1, s'_1), \ldots, (\neg y_n, s'_n)\}$ die Mengen der positiven bzw. negativen Zustandsliterale von $t(path(\epsilon, w))$. Dann gilt für alle $1 \leq i \leq m$,

$$g_{\mathcal{K}}(x_i)(s_i) = g^*_{\mathcal{K}}(x_i)(s_i) = 1,$$

und für alle $1 \leq i \leq n$,

$$\neg^2(g_{\mathcal{K}}(y_i)(s'_i)) = \neg^2(g^*_{\mathcal{K}}(y_i)(s'_i)) = g^*_{\mathcal{K}}(\neg y_i)(s'_i) = 1,$$

also $g_{\mathcal{K}}(y_i)(s'_i) = 0$. Daraus folgt, dass Φ und $\{(y_1, s'_1), \ldots, (y_n, s'_n)\}$ disjunkt sind.

Für alle Knoten $u \leq w$ und $v \leq w$, $s \in State$ und Literale φ, ψ mit $t(u) = (\varphi, s)$ und $t(v) = (\psi, s)$ gilt demnach $\varphi \neq \neg\psi$. Also ist t offen.

Sei t offen. Dann gibt es ein Blatt w von t mit der Eigenschaft, dass für je zwei Knoten $u \leq w$ und $v \leq w$, $s \in State$ und ein Literal φ mit $t(u) = (\varphi, s)$ und $t(v) = (\psi, s)$ $\varphi \neq \neg\psi$ gilt.

Seien R_1 die Menge der Transitionen und $R_2 = \{(x_1, s_1), \ldots (x_m, s_m)\}$ und $\{(\neg y_1, s'_1), \ldots, (\neg y_n, s'_n)\}$ die Mengen der positiven bzw. negativen Zustandsliterale von $t(path(\epsilon, w))$.

Sei $\mathcal{K} = \langle mkfun(R_1), mkfun(R_2^{-1})\rangle$ (siehe Isomorphien). Wir zeigen, dass $\mathcal{K}$ t erfüllt. Da t offen ist, sind β und $\{(y_1, s'_1), \ldots, (y_n, s'_n)\}$ disjunkt. Daraus folgt für alle $1 \leq i \leq m$,

$$g^*_{\mathcal{K}}(x_i)(s_i) = g_{\mathcal{K}}(x_i)(s_i) = 1,$$

und für alle $1 \leq i \leq n$,

$$g^*_{\mathcal{K}}(\neg y_i)(s'_i) = \neg^2(g^*_{\mathcal{K}}(y_i)(s'_i)) = \neg^2(g_{\mathcal{K}}(y_i)(s'_i)) = \neg^2(0) = 1.$$

Also erfüllt $\mathcal{K}$ alle Formeln von $t(path(\epsilon, w))$, d.h. $\mathcal{K}$ erfüllt t. ❑

Für alle Tableaus t, t' definieren wir:

$$t \leq t' \Leftrightarrow_{def} \text{ für alle } w \in \mathbb{N}^*,\ t'(w) \begin{cases} = t(w) & \text{falls } t(w) \text{ Zustandsliteral ist,} \\ \in \{t(w), box(\varphi, s)\} & \text{falls } t(w) = (\Box\varphi, s), \\ \in \{t(w), \top\} & \text{sonst.} \end{cases}$$

Sei $(t_1, \ldots, t_n)$ eine Tableau-Ableitung. Da jede Anwendung einer Tableauregel keine Knoten entfernt, sondern höchstens neue Knoten an vorhandene Blätter anhängt, gilt $t_i \leq t_{i+1}$ für alle $1 \leq i < n$.

Satz 7.4 (Korrektheit modallogischer Tableau-Ableitungen)

Sei $\mathcal{K} = \langle \delta, \beta\rangle$ eine Kripke-Struktur.

(i) Aus $t \vdash_{\mathcal{T}} t'$ und $\mathcal{K} \models t$ folgt $\mathcal{K} \models t'$.

(ii) Sei t' geschlossen. Dann ist t unerfüllbar.

Beweis. (i): Sei $(t_1, \ldots, t_n)$ eine Tableau-Ableitung von t' aus t. Wir zeigen, dass für alle $1 \leq i < n$ $\mathcal{K} \models t_{i+1}$ aus $\mathcal{K} \models t_i$ folgt.

Sei $\mathcal{K} \models t_i$. Dann gibt es ein Blatt v von t_i derart, dass $\mathcal{K}$ alle Zustandsformeln und Transitionen von $t_i(path(\epsilon, v))$ erfüllt. Ist v auch ein Blatt von t_{i+1}, dann folgt $\mathcal{K} \models t_{i+1}$ aus $t_i \leq t_{i+1}$.

Andernfalls gilt $w \leq v$ für den Redex w der Regelanwendung, die von t_i zu t_{i+1} führt, und es gibt einen Teilbaum t_v von t_{i+1}, der das Blatt v von t_i ersetzt.

Angenommen,

$$\begin{array}{l}\text{es gibt ein Blatt } v' \text{ von } t_v \text{ derart, dass } \mathcal{K} \text{ alle Zustandsformeln} \\ \text{und Transitionen von } t_v(path(\epsilon, v')) \text{ erfüllt.}\end{array} \tag{1}$$

Dann erfüllt $\mathcal{K}$ auch alle Zustandsformeln und Transitionen von $t_{i+1}(path(\epsilon, v'))$.

Also gilt $\mathcal{K} \models t_{i+1}$.

Zu zeigen bleibt (1).

Fall 1: t_{i+1} wurde durch Anwendung der $\vee$-Regel aus t_i gebildet. Dann gilt für alle $u \in \mathbb{N}^*$,

$$t_v(u) =_{def} \begin{cases} (\varphi, s) & \text{falls } u = 0, \\ (\psi, s) & \text{falls } u = 1, \\ t_i(u) & \text{sonst.} \end{cases}$$

Aus $\mathcal{K} \models t_i(w) = (\varphi \vee \psi, s)$ folgt $\mathcal{K} \models (\varphi, s) = t_v(0)$ oder $\mathcal{K} \models (\psi, s) = t_v(1)$. Also gilt (1) für $v' = 0$ oder $v' = 1$.

Fall 2: t_{i+1} wurde durch Anwendung der $\wedge$-Regel aus t_i gebildet. Dann gilt für alle $u \in \mathbb{N}^*$,

$$t_v(u) =_{def} \begin{cases} (\varphi, s) & \text{falls } u = 0, \\ (\psi, s) & \text{falls } u = 00, \\ t_i(u) & \text{sonst.} \end{cases}$$

Aus $\mathcal{K} \models t_i(w) = (\varphi \wedge \psi, s)$ folgt $\mathcal{K} \models (\varphi, s) = t_v(0)$ und $\mathcal{K} \models (\psi, s) = t_v(00)$. Also gilt (1) für $v' = 11$.

Fall 3: t_{i+1} wurde durch Anwendung der $\Diamond$-Regel aus t_i gebildet.

Sei $\{\varphi_1, \ldots, \varphi_n\} = boxes(s, t, v)$. Dann gilt für alle $u \in \mathbb{N}^*$,

$$t_v(u) =_{def} \begin{cases} (s, s') & \text{falls } u = 0, \\ (\varphi, s') & \text{falls } u = 00, \\ (\varphi_i, s') & \text{falls } 1 \leq i \leq n \text{ und } u = 000^i, \\ t_i(u) & \text{sonst.} \end{cases}$$

Aus $\mathcal{K} \models t_i(w) = (\Diamond\varphi, s)$ folgt, dass es $s' \in \delta(s)$ mit $\mathcal{K} \models (\varphi, s')$ gibt. Demnach gilt $\mathcal{K} \models (s, s') = t_v(0)$ und $\mathcal{K} \models (\varphi, s') = t_v(00)$.

Sei $1 \leq i \leq n$. Da es $u \leq v$ gibt mit $t_i(u) = (\Box\varphi_i, s)$, gilt $\mathcal{K} \models (\Box\varphi_i, s)$, also wegen $(s, s') \in \delta$ auch $\mathcal{K} \models (\varphi_i, s') = t_v(000^i)$. Demnach gilt (1) für $v' = 000^n$.

Fall 4: t_{i+1} wurde durch Anwendung der $\Box$-Regel aus t_i gebildet.

Sei $\{s_1, \ldots, s_n\} = succs(s, t, v)$. Dann gilt für alle $u \in \mathbb{N}^*$,

$$t_v(u) =_{def} \begin{cases} (\varphi, s_i) & \text{falls } 1 \leq i \leq n \text{ und } u = 0^i, \\ t_i(u) & \text{sonst.} \end{cases}$$

Aus $\mathcal{K} \models t_i(w) = (\Box\varphi, s)$ folgt, dass für alle $s' \in \delta(s)$ $\mathcal{K} \models (\varphi, s')$ gilt.

Sei $1 \leq i \leq n$. Da es $u \leq v$ gibt mit $t_i(u) = (s, s_i)$, gehört (s, s_i) zu δ. Daraus folgt $\mathcal{K} \models (\varphi, s_i)$. Demnach gilt (1) für $v' = 0^n$.

(ii): Da t' geschlossen ist, folgt aus Lemma 7.3, dass t' unerfüllbar ist. Also ist wegen (i) auch t unerfüllbar. ❑

Satz 7.5 (Vollständigkeit modallogischer Tableau-Ableitungen)

Sei (φ, s) eine unerfüllbare Zustandsformel und ein erstes Tableau wie folgt definiert: Für alle $w \in \mathbb{N}^*$,

$$t_1(w) = \begin{cases} (\varphi, s) & \text{falls } w = \epsilon, \\ * & \text{sonst.} \end{cases}$$

Dann gibt es eine Tableau-Widerlegung $(t_1, \ldots, t_n)$.

Beweis. Wiederholte Anwendungen der Tableauregeln führen von t_1 zu einem gesättigten Tableau t, weil jede Regelanwendung die Knotenanzahl eines Tableaus zwar erhöht, die Multimenge der Größen aller noch nicht besuchten Zustandsformeln jedoch verkleinert, so dass irgendwann ein Tableau ohne solche Formeln, also ein gesättigtes Tableau, erreicht ist, m.a.W.: es gibt eine Tableau-Ableitung $(t_1, \ldots, t_n)$. Es bleibt zu zeigen, dass t_n geschlossen ist.

Wir führen den Beweis durch Kontraposition, nehmen also an, dass t_n offen ist. Da t_n gesättigt ist, folgt aus Lemma 7.3, dass t_n erfüllbar ist, d.h. es gibt eine Kripke-Struktur $\mathcal{K} = \langle \delta, \beta \rangle$ mit $\mathcal{K} \models t_n$, also ein Blatt w von t_n derart, dass $\mathcal{K}$ alle Zustandsformeln und Transitionen von $t_n(path(\epsilon, w))$ erfüllt.

Sei $\mathcal{K}' = \langle \delta', \beta \rangle$ mit $\delta'(s) = \{s' \in \delta(s) \mid (s, s') \in t_n(path(\epsilon, w))\}$ für alle $s \in State$. Da jede Zustandsformel von t_n ein Zustandsliteral ist, also keine modalen Operationen enthält, deren Gültigkeit von δ abhängen könnte,

$$\text{erfüllt auch } \mathcal{K}' \text{ alle Zustandsformeln und Transitionen von } t_n(path(\epsilon, w)). \tag{2}$$

Wir zeigen zunächst durch Induktion über $k = n - i$, dass es für alle $1 \leq i \leq n$ ein Blatt $v \leq w$ gibt derart, dass

$$\mathcal{K}' \text{ alle Zustandsformeln und Transitionen von } t_i(path(\epsilon, v)) \text{ erfüllt.} \tag{3}$$

Fall 1: $k = 0$. Dann ist $i = n$. Also folgt (3) aus (2).

Fall 2: $k > 0$. Dann ist $i < n$ und nach Induktionsvoraussetzung gibt es ein Blatt $v \leq w$ von t_{i+1} derart, dass $\mathcal{K}'$ alle Zustandsformeln und Transitionen von $t_{i+1}(path(\epsilon, v))$ erfüllt. Nach Konstruktion von t_{i+1} aus t_i gibt es ein Blatt $u \leq v$ von t_i. Es bleibt zu zeigen, dass

$$\mathcal{K}' \text{ alle Zustandsformeln und Transitionen von } t_i(path(\epsilon, u)) \text{ erfüllt.} \tag{4}$$

Sei $u' \leq u$ und $t_i(u')$ eine Zustandsformel oder Transition.

Fall 2.1: $t_i(u') = t_{i+1}(u')$. Wegen $u' \in path(\epsilon, v)$ folgt $\mathcal{K}' \models t_i(u')$.

Fall 2.2: Es gibt $\varphi \in T_{ML}(At)$ und $s \in State$ mit $t_i(u') = (\varphi, s)$ und $t_{i+1}(u') = \top$.

Fall 2.2.1: $\varphi = (\varphi' \vee \psi)$. Dann führt eine Anwendung der $\vee$-Regel von t_i nach t_{i+1}, d.h. es gibt $v' > v$ mit $v' \leq w$ und $t_{i+1}(v') \in \{(\varphi', s), (\psi, s)\}$. Wegen $v' \in path(\epsilon, w)$ folgt $\mathcal{K}' \models (\varphi', s)$ oder $\mathcal{K}' \models (\psi, s)$, also $\mathcal{K}' \models (\varphi' \vee \psi, s) = (\varphi, s)$.

Fall 2.2.2: $\varphi = (\varphi' \wedge \psi)$. Dann führt eine Anwendung der $\wedge$-Regel von t_i nach t_{i+1}, d.h. es gibt $v', v'' > v$ mit $v', v'' \leq w$, $t_{i+1}(v') = (\varphi', s)$ und $t_{i+1}(v'') = (\psi, s)$. Wegen $v', v'' \in path(\epsilon, w)$ folgt $\mathcal{K}' \models (\varphi', s)$ und $\mathcal{K}' \models (\psi, s)$, also $\mathcal{K}' \models (\varphi' \wedge \psi, s) = (\varphi, s)$.

Fall 2.2.3: $\varphi = \Diamond\psi$. Dann gibt es $v' > v$, $v'' > v$ und $s' \in State$ mit $v', v'' \leq w$, $t_{i+1}(v') = (s, s')$ und $t_{i+1}(v'') = (s', \psi)$. Wegen $v', v'' \in path(\epsilon, w)$ folgt $s' \in \delta(s)$ und $\mathcal{K}' \models (\psi, s')$, also $\mathcal{K}' \models (\Diamond\psi, s) = (\varphi, s)$.

Fall 2.3: $\varphi = \Box\psi$ und $t_{i+1}(u') = box(\psi, s)$. Sei $\delta(s) = \{s_1, \ldots, s_k\}$. Nach Konstruktion von $\mathcal{K}'$ und $(t_{i+1} \ldots, t_n)$ gibt es $u_1, \ldots, u_k \in t_n(path(\epsilon, w))$ mit $t_n(u_i) = (\psi, s_i)$ für alle $1 \leq i \leq k$. Daraus folgt $\mathcal{K}' \models (\psi, s_i)$ für alle $1 \leq i \leq k$, also $\mathcal{K} \models (\Box\psi, s) = (\varphi, s)$.

Da $u' \in path(\epsilon, u)$ beliebig gewählt wurde, haben wir (4) gezeigt, womit der Beweis von Fall 2 des Induktionsbeweises von (3) beendet ist.

Damit gilt (3) insbesondere für t_1. Also erfüllt $\mathcal{K}'$ die Zustandsformel $t_1(\epsilon) = (\varphi, s)$. $\lightning$ ❑

Erfüllbarkeitstest für modallogische NNFs

Sei (φ, s) eine NNF.

Wende Tableauregeln in beliebiger Reihenfolge auf das durch $t_1(\epsilon) = (\varphi, s)$ und $t_1(w) = *$ für alle $w \in \mathbb{N}^+$ definierte Tableau an, bis ein gesättigtes Tableau t erreicht ist.

Prüfe, ob t offen ist.

Wenn ja, ist t – nach Lemma 7.3 – erfüllbar. Wenn nein, ist t – ebenfalls nach Lemma 7.3 – unerfüllbar. Im ersten Fall folgt aus Satz 7.5, dass (φ, s) erfüllbar ist. Im zweiten Fall folgt aus Satz 7.4 (ii), dass t_1, also auch (φ, s) unerfüllbar ist. ❑

Wie im aussagenlogischen Fall wird der Test effizienter, wenn man nicht erst im gesättigten Tableau nach offenen Blättern sucht, sondern schon vorher Teiltableaus entfernt, deren Pfade komplementäre Literale enthalten. Dazu muss der Kalkül um folgende Regeln erweitert werden, die alle Unterbäume t' von t durch ein Blatt mit der Markierung $\perp$ ersetzen, sofern die Wurzel w von t' mit einem Literal φ und ein Vorgänger von w in t mit dem zu φ komplementären Literal markiert ist:

- **Atomregel** Sei $t(w) = (\varphi, s)$ ein Zustandsatom. Für alle $u \in \mathbb{N}^*$,

$$t'(u) =_{def} \begin{cases} \perp & \text{falls } u > w \text{ und } t(u) = (\neg\varphi, s), \\ * & \text{falls es } v > w \text{ gibt mit } t(v) = (\neg\varphi, s) \text{ und } u > v, \\ t(u) & \text{sonst.} \end{cases}$$

- **Literalregel** Sei $t(w) = (\neg\varphi, s)$ ein Zustandsliteral. Für alle $u \in \mathbb{N}^*$,

$$t'(u) =_{def} \begin{cases} \perp & \text{falls } u > w \text{ und } t(u) = (\varphi, s), \\ * & \text{falls es } v > w \text{ gibt mit } t(v) = (\varphi, s) \text{ und } u > v, \\ t(u) & \text{sonst.} \end{cases}$$

Außerdem ist in den anderen Tableauregeln die Menge $leaves(t, w)$ der von w aus erreichbaren Blätter von t auf diejenigen zu beschränken, die nicht mit $\perp$ markiert sind.

Schließlich wird der Erfüllbarkeitstest für NNFs zu einem Erfüllbarkeitstest für beliebige modallogische Formeln, wenn er die oben erwähnten, aus den Gleichungen von *negAL* und *negM* gebildeten Tableauregeln mitverwendet.

Beispiele ([38], Abschnitt B 6.2) Sei $State = \{s_1, s_2, s_3, \dots\}$.

1. Sei $At = \{x\}$. ☞ Erfüllbarkeitstest der Zustandsformel

$$(\Diamond\Diamond x \wedge \Box(x \Rightarrow \Box\neg x), s_1)$$

Aus dem letzten Tableau der Ableitung ergibt sich, dass die Formel von jeder Kripke-Struktur $\langle \delta, \beta \rangle$ mit $\delta(s_1) = \{s_2\}, \delta(s_2) = \{s_3\}, \delta(s) = \emptyset$ für alle $s \in State \setminus \{s_1, s_2\}$ und $x \in \beta(s_3) \setminus \beta(s_2)$ erfüllt wird.

2. Sei $At = \{x, y\}$. ☞ Erfüllbarkeitstest der Zustandsformel

$$(\Diamond x \wedge \Box\neg y \wedge \Diamond(x \Rightarrow y), s_1)$$

Aus dem letzten Tableau der Ableitung ergibt sich, dass die Formel von jeder Kripke-Struktur $\langle \delta, \beta \rangle$ mit $\delta(s_1) = \{s_2\}, \delta(s_1) = \{s_3\}, \delta(s) = \emptyset$ für alle $s \in State \setminus \{s_1, s_2\}$, $x \in \beta(s_2) \setminus \beta(s_3)$ und $y \notin \beta(s_2) \cup \beta(s_3)$ erfüllt wird.

3. Sei $At = \{x, y\}$. ☞ Erfüllbarkeitstest der Zustandsformel

$$(\Box(x \Rightarrow y) \wedge \Diamond x \wedge \Box\neg y, s_1)$$

Aus dem letzten Tableau der Ableitung ergibt sich, dass die Formel unerfüllbar ist. Dementsprechend ist ihre Negation erfüllbar, und zwar – wie ihr ☞ Erfüllbarkeitstest zeigt – von jeder Kripke-Struktur $\langle \delta, \beta \rangle$ mit

- $\delta(s_1) = \{s_2\}$, $x \in \beta(s_2)$ und $y \notin \beta(s_2)$, oder
- $\delta = \emptyset$, oder
- $\delta(s_1) = \{s_3\}$ und $y \in \beta(s_3)$.

Wer selbst solche Tableau-Reduktionen erzeugen will, installiere Haskell, lade das Painter-Paket herunter, starte *ghci* mit *Tableau.hs* und berechne Tableaus für modallogische Formeln durch entsprechende Aufrufe der Haskell-Funktion *reduceTab* (siehe Beispiele am Ende von *Tableau.hs*).

Da jedes Tableau höchstens endlich viele Transitionen enthält und jedes Tableau, auf das eine Regel angewendet wird, die es um eine Transition (s, s') erweitert, s' nicht enthält, schließen wir aus den Sätzen 7.4 (ii) und 7.5, dass jede erfüllbare Zustandsformel ein endliches Baummodell hat, das ist eine Kripke-Struktur mit endlicher Zustandsmenge, deren Transitionsfunktion δ folgende Eigenschaft hat:

$$\text{Für alle } s \in State, |\delta(s)| \leq 1.$$

7.3 Zustandsäquivalenzen

Wie oben schon erwähnt wurde, sind Kripke-Strukturen black-box-Modelle: Ein Zustand ist nicht wie ein Term aus Konstruktoren zusammengesetzt und lässt sich daher nicht anhand seines jeweiligen Aufbaus von anderen Zuständen unterscheiden. Stattdessen ist die Gleichheit oder Ungleichheit von Zuständen nur indirekt erkennbar, indem man mit ihnen *experimentiert*, d.h. im Fall von Kripke-Strukturen δ und β solange anwendet, bis die Ergebnisse der Anwendung das gleiche oder ungleiche *Verhalten* der getesteten Zustände offenbart.

Das führt zu einer coinduktiven Definition der $\mathcal{K}$**-Verhaltensäquivalenz** (*behavioral equivalence, bisimilarity*) $\sim_{\mathcal{K}}$ von Zuständen der Kripke-Struktur

$$\mathcal{K} = \langle \delta, \beta \rangle : State \to \mathcal{P}(State) \times \mathcal{P}(At)$$

bzgl. folgender Schrittfunktion $F_{\mathcal{K}} : \mathcal{P}(State^2) \to \mathcal{P}(State^2)$ (siehe Kapitel 4):

Für alle $R \subseteq State^2$,

$$\begin{aligned} F_{\mathcal{K}}(R) \quad = \quad \{(s, s') \in State^2 \mid\ & \beta(s) = \beta(s'), \\ & \forall\, t \in \delta(s)\ \exists\, t' \in \delta(s') : (t, t') \in R, \\ & \forall\, t' \in \delta(s')\ \exists\, t \in \delta(s) : (t, t') \in R\}. \end{aligned}$$

Eine $F_{\mathcal{K}}$-dichte Teilmenge von $State^2$ heißt $\mathcal{K}$**-Bisimulation**.

$\sim_{\mathcal{K}}$ ist definiert als größte $\mathcal{K}$-Bisimulation, also $\sim_{\mathcal{K}} = gfp(F_{\mathcal{K}})$.

Zwei Zustände $s, s' \in State$ heißen $\mathcal{K}$**-verhaltensäquivalent**, wenn (s, s') zu $\sim_{\mathcal{K}}$ gehört.

Aufgabe Zeigen Sie, dass die Diagonale von $State^2$ eine $\mathcal{K}$-Bisimulation ist.

Satz 7.5 $\sim_{\mathcal{K}}$ ist eine Äquivalenzrelation (siehe Kapitel 5).

Beweis. Sei $\sim_{\mathcal{K}}^{eq}$ der **Äquivalenzabschluss** von $\sim_{\mathcal{K}}$, das ist die kleinste Äquivalenzrelation, die $\sim_{\mathcal{K}}$ enthält. Wir nehmen an, dass

$$\sim_{\mathcal{K}}^{eq} \text{ eine } \mathcal{K}\text{-Bisimulation ist.} \tag{1}$$

Da $\sim_{\mathcal{K}}$ die größte $\mathcal{K}$-Bisimulation ist, folgt aus (1), dass $\sim_{\mathcal{K}}^{eq}$ in $\sim_{\mathcal{K}}$ enthalten ist. Andererseits ist $\sim_{\mathcal{K}}$ nach Definition von $\sim_{\mathcal{K}}^{eq}$ in $\sim_{\mathcal{K}}^{eq}$ enthalten. Beide Relationen stimmen also überein, so dass mit $\sim_{\mathcal{K}}^{eq}$ auch $\sim_{\mathcal{K}}$ eine Äquivalenzrelation ist.

Zu zeigen bleibt (1), d.h. für alle $(s, s') \in State^2$ muss folgende Implikation gelten:

$$s \sim_{\mathcal{K}}^{eq} s' \;\Rightarrow\; \begin{cases} \beta(s) = \beta(s'), & (2) \\ \forall\, t \in \delta(s)\ \exists\, t' \in \delta(s') : t \sim_{\mathcal{K}}^{eq} t', & (3) \\ \forall\, t' \in \delta(s')\ \exists\, t \in \delta(s) : t \sim_{\mathcal{K}}^{eq} t'. & (4) \end{cases}$$

Beweis von (1)-(3). Nach Definition von $\sim_{\mathcal{K}}^{eq}$ stimmt diese Relation mit $lfp(G)$ überein, wobei $G : \mathcal{P}(State^2) \to \mathcal{P}(State^2)$ wie folgt definiert ist: Für alle $R \subseteq State^2$,

$$G(R) \;=\; \sim_{\mathcal{K}} \cup\, \Delta_{State} \cup \{(s', s) \mid (s, s') \in R\} \cup \{(s, s') \mid \exists\, s'' : (s, s''), (s'', s') \in R\}.$$

Offenbar gelten (2)-(4) genau dann, wenn

$$R \;=_{def}\; \{(s, s') \in State^2 \mid (s, s') \text{ erfüllt (2)-(4)}\}$$

$\sim_{\mathcal{K}}^{eq}$ enthält. Nach Satz 4.2 (iii) ist $lfp(G) = \sim_{\mathcal{K}}^{eq}$ in jeder G-abgeschlossenen Teilmenge von $State^2$ enthalten. Also gelten (2)-(4), wenn R G-abgeschlossen ist.

Sei $(s, s') \in G(R)$.

Fall 1: $s \sim_{\mathcal{K}} s'$. Daraus folgt $(s, s') \in R$, weil $\sim_{\mathcal{K}}$ eine $\mathcal{K}$-Bisimulation und in $\sim_{\mathcal{K}}^{eq}$ enthalten ist.

Fall 2: $s = s'$. Daraus folgt $(s, s') \in R$, weil die Diagonale von $State^2$ eine $\mathcal{K}$-Bisimulation und in $\sim_{\mathcal{K}}^{eq}$ enthalten ist.

Fall 3: $(s', s) \in R$. Dann gilt $\beta(s') = \beta(s)$, für alle $t' \in \delta(s')$ gibt es $t \in \delta(s)$ mit $t' \sim_{\mathcal{K}}^{eq} t$ und für alle $t \in \delta(s)$ gibt es $t' \in \delta(s')$ mit $t' \sim_{\mathcal{K}}^{eq} t$. Daraus folgt $(s, s') \in R$, weil die Diagonale von $\mathcal{P}(At)^2$ und $\sim_{\mathcal{K}}^{eq}$ symmetrisch sind.

Fall 4: Es gibt $s'' \in State$ mit $(s, s''), (s'', s') \in R$. Dann gilt $\beta(s) = \beta(s'') = \beta(s')$, für alle $t \in \delta(s)$ gibt es $t'' \in \delta(s'')$ mit $t \sim_{\mathcal{K}}^{eq} t''$, für alle $t'' \in \delta(s'')$ gibt es $t \in \delta(s)$ mit $t \sim_{\mathcal{K}}^{eq} t''$, für alle $t'' \in \delta(s'')$ gibt es $t' \in \delta(s')$ mit $t'' \sim_{\mathcal{K}}^{eq} t'$ und für alle $t' \in \delta(s')$ gibt es $t'' \in \delta(s'')$ mit $t'' \sim_{\mathcal{K}}^{eq} t'$. Daraus folgt $(s, s') \in R$, weil die Diagonale von $\mathcal{P}(At)^2$ und $\sim_{\mathcal{K}}^{eq}$ transitiv sind.

Damit haben wir gezeigt, dass R G-abgeschlossen ist. ❑

$\mathcal{K}$-Bisimulationen setzen Zustände ein und derselben Kripke-Struktur in Beziehung. Eine allgemeinere Definition erlaubt den Vergleich von Zuständen zweier Kripke-Strukturen

$$\mathcal{K} = \langle \delta, \beta \rangle : State \to (\mathcal{P}(State) \times \mathcal{P}(At)),\ \mathcal{K}' = \langle \delta', \beta' \rangle : State' \to (\mathcal{P}(State') \times \mathcal{P}(At))$$

mit denselben Atomen, aber möglicherweise verschiedenen Zustandsmengen:
$R \subseteq State \times State'$ heißt $\mathcal{K}, \mathcal{K}'$**- Bisimulation**, wenn R $F_{\mathcal{K},\mathcal{K}'}$-dicht ist, wobei

$$F_{\mathcal{K},\mathcal{K}'} : \mathcal{P}(State \times State') \to \mathcal{P}(State \times State')$$

wie folgt definiert ist: Für alle $R \subseteq State \times State'$,

$$\begin{aligned} F_{\mathcal{K},\mathcal{K}'}(R) \;=\; \{(s, s') \in State \times State' \mid\ & \beta(s) = \beta(s'), \\ & \forall\, t \in \delta(s)\ \exists\, t' \in \delta(s') : (t, t') \in R, \\ & \forall\, t' \in \delta(s')\ \exists\, t \in \delta(s) : (t, t') \in R\}. \end{aligned}$$

Eine Funktion $h : State \to State'$ heißt **Kripke-Morphismus**, wenn das folgende Diagramm kommutiert:

$$\begin{array}{ccc} State & \xrightarrow{\mathcal{K}} & \mathcal{P}(State) \times \mathcal{P}(At) \\ \Big\downarrow h & = & \Big\downarrow \mathcal{P}(h) \times id \\ State' & \xrightarrow{\mathcal{K}'} & \mathcal{P}(State') \times \mathcal{P}(At) \end{array}$$

In Worten: h ist ein Kripke-Morphismus, wenn h jeden Zustand $s \in State$ auf einen Zustand $s' \in State'$ abbildet, der dieselben Atome erfüllt wie s, sowie jeden direkten Folgezustand von s auf einen direkten Folgezustand von s'.

Kripke-Morphismen sind für Kripke-Strukturen offenbar das, was Σ-Homomorphismen für Σ-Algebren sind. Dementsprechend sind zwei Kripke-Strukturen $\mathcal{K}$ und $\mathcal{K}'$ **Kripke-isomorph**, wenn es Kripke-Morphismen $g : \mathcal{K} \to \mathcal{K}'$ und $h : \mathcal{K}' \to \mathcal{K}$ mit $g \circ h = id_{State'}$ und $h \circ g = id_{State}$ gibt.

Aufgabe Zeigen Sie, dass $h : State \to State'$ genau dann ein Kripke-Morphismus ist, wenn der Graph von h (siehe Abschnitt 3.2) eine $\mathcal{K}, \mathcal{K}'$-Bisimulation ist.

Die größte $\mathcal{K}, \mathcal{K}'$-Bisimulation heißt $\mathcal{K}, \mathcal{K}'$**-Verhaltensäquivalenz** und wird mit $\sim_{\mathcal{K},\mathcal{K}'}$ bezeichnet. Offenbar ist $\sim_{\mathcal{K},\mathcal{K}} = \sim_{\mathcal{K}}$.

$s \in State$ und $s' \in State'$ heißen **elementar** $\mathcal{K}, \mathcal{K}'$**-äquivalent**, wenn $\mathcal{K}$ im Zustand s dieselben modallogischen Formeln erfüllt wie $\mathcal{K}'$ im Zustand s'.

Die Menge aller Paare elementar $\mathcal{K}, \mathcal{K}'$-äquivalenter Zustände bezeichnen wir mit $\approx_{\mathcal{K},\mathcal{K}'}$, im Fall $\mathcal{K} = \mathcal{K}'$ mit $\approx_{\mathcal{K}}$.

Satz 7.6 Hennessy-Milner-Theorem

Seien $\mathcal{K}$ und $\mathcal{K}'$ modal gesättigte Kripke-Strukturen. Dann gilt $\sim_{\mathcal{K},\mathcal{K}'} = \approx_{\mathcal{K},\mathcal{K}'}$.

Beweis. Wir zeigen zunächst:

$$\sim_{\mathcal{K},\mathcal{K}'} \quad \subseteq \quad \approx_{\mathcal{K},\mathcal{K}'} . \tag{1}$$

(1) folgt aus der Definition elementarer $\mathcal{K}$-Äquivalenz und folgender Eigenschaft:

Für alle $\varphi \in T_{ML}(At)$ und $s, s' \in State$ gilt:

$$s \sim_{\mathcal{K},\mathcal{K}'} s' \;\Rightarrow\; (s \in g^*_{\mathcal{K}}(\varphi) \Leftrightarrow s' \in g^*_{\mathcal{K}'}(\varphi)). \tag{2}$$

Beweis von (2) durch strukturelle Induktion über $T_{ML}(At)$:

Sei $s \sim_{\mathcal{K},\mathcal{K}'} s'$. Dann gilt für alle $x \in At$,

$$s \in g^*_{\mathcal{K}}(x) = g_{\mathcal{K}}(x) \;\Leftrightarrow\; x \in \beta(s) \overset{s \sim_{\mathcal{K},\mathcal{K}'} s'}{=} \beta'(s') \;\Leftrightarrow\; s' \in g_{\mathcal{K}'}(x) = g^*_{\mathcal{K}'}(x).$$

Für alle $\varphi, \psi \in T_{ML}(At)$,

$$s \in g^*_{\mathcal{K}}(\neg\varphi) = State \setminus g^*_{\mathcal{K}}(\varphi) \overset{ind.\ hyp.}{\Leftrightarrow} s' \in State' \setminus g^*_{\mathcal{K}'}(\varphi) = g^*_{\mathcal{K}'}(\neg\varphi)(s'),$$

$$s \in g^*_{\mathcal{K}}(\varphi \wedge \psi) = g^*_{\mathcal{K}}(\psi) \cap g^*_{\mathcal{K}}(\vartheta) \overset{ind.\ hyp.}{\Leftrightarrow} s' \in g^*_{\mathcal{K}'}(\varphi) \cap g^*_{\mathcal{K}'}(\psi) = g^*_{\mathcal{K}'}(\varphi \wedge \psi),$$

$$s \in g^*_{\mathcal{K}}(\varphi \vee \psi) = g^*_{\mathcal{K}}(\psi) \cup g^*_{\mathcal{K}}(\vartheta) \overset{ind.\ hyp.}{\Leftrightarrow} s' \in g^*_{\mathcal{K}'}(\varphi) \cup g^*_{\mathcal{K}'}(\psi) = g^*_{\mathcal{K}'}(\varphi \vee \psi).$$

Wegen $s \sim_{\mathcal{K},\mathcal{K}'} s'$ gilt

$$\forall\, t \in \delta(s)\ \exists\, t' \in \delta'(s') : t \sim_{\mathcal{K}} t' \quad \text{und} \quad \forall\, t' \in \delta'(s')\ \exists\, t \in \delta(s) : t \sim_{\mathcal{K},\mathcal{K}'} t'. \tag{3}$$

Daraus folgt

$$s \in g^*_{\mathcal{K}}(\Box\varphi) \;\Leftrightarrow\; \delta(s) \subseteq g^*_{\mathcal{K}}(\varphi) \overset{(3),\ ind.\ hyp.}{\Leftrightarrow} \delta'(s') \subseteq g^*_{\mathcal{K}'}(\varphi) \;\Leftrightarrow\; s' \in g^*_{\mathcal{K}'}(\Box\varphi),$$

$$s \in g^*_{\mathcal{K}}(\Diamond\varphi) \;\Leftrightarrow\; \delta(s) \cap g^*_{\mathcal{K}}(\varphi) \neq \emptyset \overset{(3),\ ind.\ hyp.}{\Leftrightarrow} \delta'(s') \cap g^*_{\mathcal{K}'}(\varphi) \neq \emptyset \;\Leftrightarrow\; s' \in g^*_{\mathcal{K}'}(\Diamond\varphi).$$

Zu zeigen bleibt die Umkehrung von (1):

$$\approx_{\mathcal{K},\mathcal{K}'} \;\subseteq\; \sim_{\mathcal{K},\mathcal{K}'}.$$

Da $\sim_{\mathcal{K},\mathcal{K}'}$ die größte $\mathcal{K},\mathcal{K}'$-Bisimulation ist, genügt es zu zeigen, dass $\approx_{\mathcal{K},\mathcal{K}'}$ eine solche ist, dass also folgende Implikation gilt:

$$s \approx_{\mathcal{K},\mathcal{K}'} s' \;\Rightarrow\; \begin{cases} \beta(s) = \beta'(s'), & (4) \\ \forall\, t \in \delta(s)\ \exists\, t' \in \delta'(s') : t \approx_{\mathcal{K},\mathcal{K}'} t', & (5) \\ \forall\, t' \in \delta'(s')\ \exists\, t \in \delta(s) : t \approx_{\mathcal{K},\mathcal{K}'} t'. & (6) \end{cases}$$

Sei $s \approx_{\mathcal{K},\mathcal{K}'} s'$. Zunächst erhalten wir (4) wie folgt: Für alle $x \in At$,

$$x \in \beta(s) \;\Leftrightarrow\; s \in g_{\mathcal{K}}(x) = g^*_{\mathcal{K}}(x) \;\Leftrightarrow\; s' \in g^*_{\mathcal{K}'}(x) = g_{\mathcal{K}'}(x) \;\Leftrightarrow\; x \in \beta'(s').$$

Ab hier folgen wir dem Beweis von [5], Proposition 2.54, oder [9], Proposition 6.10, und zeigen (5).

Sei $t \in \delta(s)$, $\Phi = \{\varphi \in T_{ML}(At) \mid \mathcal{K} \models_t \varphi\}$ und Φ' eine endliche Teilmenge von Φ. Dann gilt $\mathcal{K} \models_s \Diamond \bigwedge \Phi'$, also auch $\mathcal{K}' \models_{s'} \Diamond \bigwedge \Phi'$ wegen $s \approx_{\mathcal{K},\mathcal{K}'} s'$. Da $\mathcal{K}'$ modal gesättigt ist, gibt es $t' \in \delta'(s')$ mit $\mathcal{K}' \models_{t'} \varphi$ für alle $\varphi \in \Phi$. Daher gilt $t \approx_{\mathcal{K},\mathcal{K}'} t'$.

Analog erhält man (6). ❑

Ursprünglich wurde das Hennessy-Milner-Theorem für endlich verzweigte Kripke-Strukturen formuliert (siehe [13], Theorem 2.1, oder [10], Satz 3.17). Es charakterisiert die Verhaltensäquivalenz zweier Zustände s und s' als deren *Ununterscheidbarkeit*: s und s' sind genau dann verhaltensäquivalent, wenn es keine modallogische Formel gibt, die in s einen anderen Wahrheitswert hat als in s'.

Sei $\mathcal{K} = \langle \delta, \beta \rangle : State \to (\mathcal{P}(State) \times \mathcal{P}(At))$ eine Kripke-Struktur.

Die Quotientenmenge $State/\!\sim_{\mathcal{K}}$ (siehe Kapitel 5) lässt sich zu folgender Kripke-Struktur erweitern:

$$\begin{array}{rcl} \mathcal{K}_{min} = \langle \delta_{min}, \beta_{min} \rangle : State/\!\sim_{\mathcal{K}} & \to & \mathcal{P}(State/\!\sim_{\mathcal{K}}) \times \mathcal{P}(At) \\ [s] & \mapsto & (\{[s'] \mid s' \in \delta(s)\},\ \beta(s)) \end{array}$$

Hier bleibt zu prüfen, ob $\mathcal{K}_{min}([s])$ unabhängig vom Repräsentanten s der Äquivalenzklasse $[s]$ stets denselben Wert hat, d.h. ob für alle $s \sim_{\mathcal{K}} t$ Folgendes gilt:

$$\delta_{min}([s]) = \delta_{min}([t]), \quad (1)$$

$$\beta_{min}([s]) = \beta_{min}([t]). \quad (2)$$

Beweis von (1). Sei $[s'] \in \delta_{min}([s])$. Nach Definition von δ_{min} gibt es $u \in \delta(s)$ mit $u \sim_{\mathcal{K}} s'$. Da $\sim_{\mathcal{K}}$ eine $\mathcal{K}$-Bisimulation ist, existiert $v \in \delta(t)$ mit $u \sim_{\mathcal{K}} v$. Also folgt $[s'] \in \delta_{min}([t])$ aus $v \sim_{\mathcal{K}} u \sim_{\mathcal{K}} s'$. Die Umkehrung $\delta_{min}([t]) \subseteq \delta_{min}([s])$ erhält man analog.

Beweis von (2). Sei $x \in \beta_{min}([s])$. Nach Definition von β_{min} gilt $x \in \beta(s)$. Da $\sim_{\mathcal{K}}$ eine $\mathcal{K}$-Bisimulation ist, folgt $x \in \beta(s) = \beta(t) = \beta_{min}([t])$. Die Umkehrung $\delta_{min}([t]) \subseteq \delta_{min}([s])$ erhält man analog. ❑

Aufgabe Zeigen Sie, dass die natürliche Abbildung $nat_{\sim_{\mathcal{K}}} : State \to State/\!\sim_{\mathcal{K}}$ (siehe Kapitel 5) ein Kripke-Morphismus ist.

Demnach ist der Graph von $nat_{\sim_\mathcal{K}}$ eine $\mathcal{K}, \mathcal{K}_{min}$-Bisimulation (siehe Aufgabe vor Satz 7.6), also in $\sim_{\mathcal{K},\mathcal{K}'}$ enthalten. Folglich ist jeder Zustand s von $\mathcal{K}$ nach Satz 7.6 zu seiner Äquivalenzklasse $[s]_{\sim_\mathcal{K}}$ elementar äquivalent, d.h. erfüllt dieselben modallogischen Formeln wie $[s]_{\sim_\mathcal{K}}$.

Approximative Konstruktion der Verhaltensäquivalenz

Der untere Kleene-Abschluss $F_{\mathcal{K},\infty} = \bigcap_{n\in\mathbb{N}} F^n_\mathcal{K}(State^2)$ von $F_\mathcal{K}$ stimmt nach Satz 4.9 (iv) mit $\sim_\mathcal{K} = gfp(F_\mathcal{K})$ überein, falls er $F_\mathcal{K}$-dicht ist.

Um das zu zeigen, definieren wir zunächst die folgenden Approximationen von $\sim_\mathcal{K}$ induktiv wie folgt: Für alle $n \in \mathbb{N}$,

$$\begin{aligned} \sim_0 &= ker(\beta), \\ \sim_{n+1} &= \{s \sim_n s' \mid \left\{ \begin{array}{l} \forall\, t \in \delta(s)\ \exists\, t' \in \delta(s') : t \sim_n t', \\ \forall\, t' \in \delta(s')\ \exists\, t \in \delta(s) : t \sim_n t' \end{array} \right\}. \end{aligned}$$

Lemma 7.7

Für alle $n \in \mathbb{N}$, $\sim_n = F^n_\mathcal{K}(ker(\beta))$.

Beweis durch Induktion über n. $\sim_0 = ker(\beta) = F^0_\mathcal{K}(ker(\beta))$.

Sei $s \sim_{n+1} s'$, $t \in \delta(s)$ und $t' \in \delta(s')$. Nach Definition von $s \sim_{n+1} s'$ gilt $\beta(s) = \beta(s')$ und gibt es $u' \in \delta(s')$ und $u \in \delta(s)$ mit $t \sim_n u'$ und $u \sim_n t'$. Nach Induktionsvoraussetzung folgt $(t, u'), (u, t') \in F^n_\mathcal{K}(ker(\beta)) \subseteq ker(\beta)$. Nach Definition von $F_\mathcal{K}$ gehört daher (s, s') zu $F^{n+1}_\mathcal{K}(ker(\beta))$.

Sei $(s, s') \in F^{n+1}_\mathcal{K}(ker(\beta))$. Nach Definition von $F_\mathcal{K}$ gilt

$$F^{n+1}_\mathcal{K}(ker(\beta)) = F^n_\mathcal{K}(F_\mathcal{K}(ker(\beta))) \subseteq F^n_\mathcal{K}(ker(\beta)),$$

also $s \sim_n s'$ nach Induktionsvoraussetzung. Sei $t \in \delta(s)$ und $t' \in \delta(s')$. Nach Definition von $F_\mathcal{K}$ gibt es $u' \in \delta(s')$ und $u \in \delta(s)$ mit $(t, u'), (u, t') \in F^n_\mathcal{K}(ker(\beta))$. Nach Induktionsvoraussetzung folgt $t \sim_n u'$ und $u \sim_n t'$. Nach Definition von $\sim_{n+1}$ gilt daher $s \sim_{n+1} s'$. ❑

Satz 7.8 ([13], Theorem 2.1)

Sei $\mathcal{K} = \langle \delta, \beta \rangle$ eine endlich verzweigte Kripke-Struktur. Die $\mathcal{K}$-Verhaltensäquivalenz stimmt mit $\sim_\omega = \bigcap_{n\in\mathbb{N}} \sim_n$ überein.

Beweis. Nach Lemma 7.7 gilt

$$\begin{aligned} &F_{\mathcal{K},\infty} = \textstyle\bigcap_{n\in\mathbb{N}} F^n_\mathcal{K}(State^2) = \bigcap_{n\in\mathbb{N}} F^{n+2}_\mathcal{K}(State^2) \\ &= \textstyle\bigcap_{n\in\mathbb{N}} F^{n+1}_\mathcal{K}(F_\mathcal{K}(ker(\beta) \cap \{(s,s') \in State^2 \mid \delta(s) = \emptyset \Leftrightarrow \delta(s') = \emptyset\}) = \bigcap_{n\in\mathbb{N}} F^{n+1}_\mathcal{K}(F_\mathcal{K}(ker(\beta))) \\ &= \textstyle\bigcap_{n\in\mathbb{N}} F^{n+2}_\mathcal{K}(ker(\beta)) = \bigcap_{n\in\mathbb{N}} \sim_{n+2} = \sim_\omega . \end{aligned}$$

Wegen $\sim_\mathcal{K} = gfp(F_\mathcal{K})$ bleibt nach Satz 4.9 (iv) zu zeigen, dass $F_{\mathcal{K},\infty} = \sim_\omega$ $F_\mathcal{K}$-dicht ist.

Sei $s \sim_\omega s'$. Zu zeigen ist $(s, s') \in F_\mathcal{K}(\sim_\omega)$, also

$$\beta(s) = \beta(s'), \tag{3}$$
$$\forall\, t \in \delta(s)\ \exists\, t' \in \delta(s') : t \sim_\omega t', \tag{4}$$
$$\forall\, t' \in \delta(s')\ \exists\, t \in \delta(s) : t \sim_\omega t'. \tag{5}$$

(3) folgt aus $s \sim_0 s'$.

Beweis von (4). Sei $t \in \delta(s)$. Nach Definition von $\sim_\omega$ gibt es für alle $n > 0$ $t_n \in \delta(s')$ mit $t \sim_{n-1} t_n$. Da $\delta(s)$ endlich ist, gibt es $t' \in State$ derart, dass $t' = t_n$ für unendlich viele natürliche Zahlen n gilt. Demnach gibt es für alle $n \in \mathbb{N}$ $i_n \geq n$ mit $t_{i_n} = t'$. (Dieses Argument wird auch im Beweis von [13], Theorem 2.1 verwendet). Aus $t \sim_{i_n-1} t_{i_n}$ folgt $t \sim_{i_n-1} t'$, also wegen $\sim_{i_n-1} \subseteq \sim_{n-1} \subseteq \sim_0$ auch $t \sim_{n-1} t'$ und $t \sim_0 t'$. Folglich gilt $t \sim_\omega t'$.

(5) erhält man analog. ❑

Dass $\mathcal{K}_{min}$ unter allen endlich verzweigten Kripke-Strukturen $\mathcal{K}'$, deren Zustandsmenge das Bild eines Kripke-Morphismus $h : \mathcal{K} \to \mathcal{K}'$ ist, bzgl. der Anzahl der Zustände minimal ist, zeigt folgender Satz.

Satz 7.9

Sei $\mathcal{K} = \langle \delta, \beta \rangle$ eine endlich verzweigte Kripke-Struktur. $\mathcal{K}_{min}$-verhaltensäquivalente Zustände sind gleich. Die Anzahl der Zustände von $\mathcal{K}_{min}$ kann also nicht weiter verringert werden, ohne dass nicht verhaltensäquivalente Zustände miteinander verschmolzen werden.

Beweis. Nach Satz 7.8 gilt $\sim_{\mathcal{K}} \; = \; \sim_{\omega} \; = F_{\mathcal{K},\infty} = \bigcap_{n\in\mathbb{N}} F^n_{\mathcal{K}}(State^2)$. Also genügt es zu zeigen, dass für alle $n \in \mathbb{N}$ und $s, s' \in State$ Folgendes gilt:

$$[s] \sim_{\mathcal{K}_{min}} [s'] \quad \Longrightarrow \quad (s, s') \in F^n_{\mathcal{K}}(State^2). \tag{6}$$

Wir zeigen (6) durch Induktion über n. Wegen $F^0_{\mathcal{K}}(State^2) = State^2$ gilt (3) im Fall $n = 0$.

Sei $[s] \sim_{\mathcal{K}_{min}} [s']$. Dann ist $\beta_{min}([s]) = \beta_{min}([s'])$ und für alle $[t] \in \delta_{min}([s])$ bzw. $[t'] \in \delta_{min}([s'])$ gibt es $[u'] \in \delta_{min}([s'])$ bzw. $[u] \in \delta_{min}([s])$ mit $[t] \sim_{\mathcal{K}_{min}} [u']$ bzw. $[u] \sim_{\mathcal{K}_{min}} [t']$.

Also ist $\beta(s) = \beta(s')$ nach Definition von β_{min} und für alle $t \in \delta(s)$ bzw. $t' \in \delta(s')$ gibt es $u' \in \delta(s')$ bzw. $u \in \delta(s)$ mit $(t, u') \in F^n_{\mathcal{K}}(State^2)$ bzw. $(u, t') \in F^n_{\mathcal{K}}(State^2)$ nach Definition von δ_{min} und Induktionsvoraussetzung. Nach Definition von $F_{\mathcal{K}}$ folgt $(s, s') \in F_{\mathcal{K}}(F^n_{\mathcal{K}}(State^2)) = F^{n+1}_{\mathcal{K}}(State^2)$. ❏

7.4 Minimierungsalgorithmus

Sei $\mathcal{K} = \langle \delta, \beta \rangle$ eine Kripke-Struktur mit endlicher Zustandsmenge *State* und $<$ eine totale Ordnung auf *State* (siehe Kapitel 5).

Nach dem sog. **Paull-Unger-Verfahren** wird die Approximation $\sim_n$, $n \in \mathbb{N}$, von $\sim_{\mathcal{K}}$ über die Funktion

$$M_n : \{(s, s') \in State^2 \mid s < s'\} \to 1 + \mathcal{P}(State)^2, \qquad n \in \mathbb{N},$$

berechnet, die folgendermaßen definiert ist: Für alle $s, s' \in State$ mit $s < s'$,

$$M_0(s, s') \;=\; \begin{cases} (\delta(s), \delta(s')) & \text{falls } \beta(s) = \beta(s'), \\ * & \text{sonst,} \end{cases}$$

$$M_{n+1}(s, s') \;=\; \begin{cases} M_n(s, s') & \text{falls für alle } (S, S') \in M_n(s, s') \text{ Folgendes gilt:} \\ & \forall\, t \in S \;\exists\, t' \in S' : t = t' \vee M_n(min\{t, t'\}, max\{t, t'\}) \neq *, \\ & \forall\, t' \in S' \;\exists\, t \in S : t = t' \vee M_n(min\{t, t'\}, max\{t, t'\}) \neq *, \\ * & \text{sonst.} \end{cases}$$

Während $\sim_0$ auf β und alle Relationen $\sim_n$ mit $n > 0$ auf δ zugreifen, wird sowohl β als auch δ nur von M_0, aber von keinem M_n mit $n > 0$ benutzt!

M_n lässt sich als Dreiecksmatrix darstellen (siehe Beispiel unten und [33], Abschnitt 2.3, wo mit dem gleichen Verfahren deterministische Automaten minimiert werden).

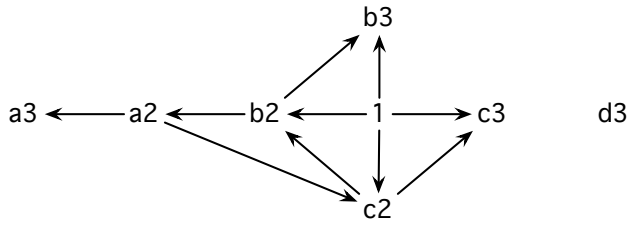
Für alle $n \in \mathbb{N}$ sei $R_n = \{(s, s') \in State^2 \mid s < s',\ M_n(s, s') \neq *\}$. Durch Induktion über n erhält man:

$$\sim_n \;=\; \Delta_{State} \cup R_n \cup R_n^{-1}.$$

Da *State* endlich ist, gibt es eine kleinste natürliche Zahl k mit $M_k = M_n$, also auch $R_k = R_n$ für alle $n \geq k$. Daraus folgt nach Satz 7.8:

$$\sim_{\mathcal{K}} \;=\; \sim_{\omega} \;=\; \Delta_{State} \cup R_k \cup R_k^{-1}.$$

Beispiel 7.10



Der Graph stellt folgende Kripke-Struktur $\mathcal{K} = \langle \delta, \lambda s.\emptyset \rangle$ dar:

$$State = \{1, a2, b2, c2, a3, b3, c3, d3\},$$
$$\delta(1) = \{b2, b3, c2, c3\}, \quad \delta(a2) = \{a3, c2\}, \quad \delta(b2) = \{a2, b3\}, \quad \delta(c2) = \{b2, c3\},$$
$$\delta(s) = \emptyset \text{ für alle } s \in \{a3, b3, c3, d3\}.$$

Wir erhalten:

$$\begin{aligned}
\sim_0 &= State^2, \\
\sim_1 &= \{s \sim_0 s' \mid \left\{ \begin{array}{l} \forall\, t \in \delta(s)\ \exists\, t' \in \delta(s') : t \sim_0 t', \\ \forall\, t' \in \delta(s')\ \exists\, t \in \delta(s) : t \sim_0 t' \end{array} \right\}\} \\
&= \{1, a2, b2, c2\}^2 \cup \{a3, b3, c3, d3\}^2, \\
\sim_2 &= \{s \sim_1 s' \mid \left\{ \begin{array}{l} \forall\, t \in \delta(s)\ \exists\, t' \in \delta(s') : t \sim_1 t', \\ \forall\, t' \in \delta(s')\ \exists\, t \in \delta(s) : t \sim_1 t' \end{array} \right\}\} \\
&= \sim_1
\end{aligned}$$

Also stimmt $\sim_{\mathcal{K}}$ mit $\sim_1$ überein. Folglich ist

$$State/\sim_{\mathcal{K}} = \{\{1, a2, b2, c2\}, \{a3, b3, c3, d3\}\}$$

die Zustandsmenge von $\mathcal{K}_{min}$.

Die (mit ☞ Expander2 erstellten) $\sim_0$ und $\sim_1$ entsprechenden "Dreiecksmatrizen" M_0 bzw. M_1 lauten wie folgt:

	1	d3	b2	c2	b3	c3	a2	a3
1		[b2,c2,b3,c3],[]	[b2,c2,b3,c3],[b3,a2]	[b2,c2,b3,c3],[b2,c3]	[b2,c2,b3,c3],[]	[b2,c2,b3,c3],[]	[b2,c2,b3,c3],[c2,a3]	[b2,c2,b3,c3],[]
d3			[],[b3,a2]	[],[b2,c3]	[],[]	[],[]	[],[c2,a3]	[],[]
b2				[b3,a2],[b2,c3]	[b3,a2],[]	[b3,a2],[]	[b3,a2],[c2,a3]	[b3,a2],[]
c2					[b2,c3],[]	[b2,c3],[]	[b2,c3],[c2,a3]	[b2,c3],[]
b3						[],[]	[],[c2,a3]	[],[]
c3							[],[c2,a3]	[],[]
a2								[c2,a3],[]
a3								

	1	d3	b2	c2	b3	c3	a2	a3
1			[b2,c2,b3,c3],[b3,a2]	[b2,c2,b3,c3],[b2,c3]			[b2,c2,b3,c3],[c2,a3]	
d3					[],[]	[],[]		[],[]
b2				[b3,a2],[b2,c3]			[b3,a2],[c2,a3]	
c2							[b2,c3],[c2,a3]	
b3						[],[]		[],[]
c3								[],[]
a2								
a3								

7.5 Verhaltensmodelle und finale Strukturen

Die Funktion, die jede Menge A auf ihre Potenzmenge $\mathcal{P}(A)$ abbildet, ist ein – mit $\mathcal{P}$ bezeichneter – **Funktor**, der nicht nur Mengen auf Mengen, sondern auch Funktionen auf Funktionen abbildet. Dabei ist $\mathcal{P}$ wie folgt auf Funktionen $f : A \to B$ definiert:

$$\begin{aligned}
\mathcal{P}(f) : \mathcal{P}(A) &\to \mathcal{P}(B) \\
C &\mapsto \lambda C.f(C)
\end{aligned}$$

$\mathcal{P}$ darf sich Funktor nennen, weil für alle $f : A \to B$ und $g : B \to C$ Folgendes gilt:

$$\mathcal{P}(g \circ f) = \mathcal{P}(g) \circ \mathcal{P}(f) \quad \text{und} \quad \mathcal{P}(id_A) = id_{\mathcal{P}(A)}.$$

Allgemein bildet ein Funktor jedes *Objekt* und jeden *Morphismus* einer *Kategorie* auf ein Objekt bzw. einen Morphismus derselben oder einer anderen Kategorie ab. All diese Begriffe entstammen der ☞ **Kategorientheorie**, die sich mit *universellen* mathematischen Konstruktionen und Eigenschaften befasst.

Der Typ eines Funktors ist also immer von der Form $\mathcal{K} \to \mathcal{L}$, wobei $\mathcal{K}$ und $\mathcal{L}$ Kategorien sind. Im Fall des oben definierten Funktors $\mathcal{P}$ ist sowohl $\mathcal{K}$ als auch $\mathcal{L}$ die Kategorie Set der Mengen (als Objekte) und Funktionen (als Morphismen), also $\mathcal{P} : Set \to Set$.

Jeder Funktor $F : Set \to Set$ definiert die Kategorie $coAlg_F$ der **F-Coalgebren** mit allen Funktionen $\alpha : A \to F(A)$ (mit irgendeiner Menge A) als Objekten und allen Funktionen $h : A \to B$, für die folgendes Diagramm für alle F-Coalgebren $\alpha : A \to F(A)$ und $\beta : B \to F(B)$ kommutiert, als Morphismen:

$$\begin{array}{ccc} A & \longrightarrow & F(A) \\ {\scriptstyle h}\downarrow & = & \downarrow{\scriptstyle F(h)} \\ B & \longrightarrow & F(B) \end{array}$$

Z.B. sind die endlich verzweigten Kripke-Strukturen mit fester Atommenge At die Objekte der Kategorie der F_{At}-Coalgebren, wobei $F_{At} : Set \to Set$ wie folgt definiert ist: Für alle Mengen $State$, $f : State \to State'$, $S \subseteq State$ und $A \subseteq At$,

$$\begin{array}{rcl} F_{At}(State) & = & \mathcal{P}_{fin}(State) \times \mathcal{P}(At), \\ F_{At}(f)(S, A) & = & (f(S), A). \end{array}$$

Wir nennen F_{At} den von At induzierten **Kripke-Funktor**.

Die Morphismen von $coAlg_{F_{At}}$ sind offenbar Kripke-Morphismen (siehe Abschnitt 7.3).

Die Menge $Tree(\mathcal{P}(At))$ aller $\mathcal{P}(At)$-markierten Bäume $t : \mathbb{N}^* \to 1 + \mathcal{P}(At)$ (siehe Abschnitt 6.3) lässt sich zu folgender Kripke-Struktur erweitern:

$$\begin{array}{rcccl} \mathcal{T}(At) & = & \langle \delta, \beta \rangle : Tree(\mathcal{P}(At)) & \to & \mathcal{P}_{fin}(Tree(\mathcal{P}(At))) \times \mathcal{P}(At) \\ & & t & \mapsto & (\{\lambda w.t(iw) \mid i \in dom(t)\},\ t(\epsilon)). \end{array}$$

Anschaulich gesprochen, ist $\delta(t)$ die Menge der maximalen echten Unterbäume von t und $\beta(t)$ die Markierung der Wurzel von t.

Aufgabe Zeigen Sie, dass $\mathcal{T}(At)$ in $coAlg_{F_{At}}$ **schwach final** ist, d.h. es gibt von jeder F_{At}-Coalgebra einen Kripke-Morphismus nach $\mathcal{T}(At)$.

Nach Definition von $\mathcal{T}(At)$ gilt für die Schrittfunktion

$$F_{\mathcal{T}(At)} : \mathcal{P}(Tree(\mathcal{P}(At))^2) \to \mathcal{P}(Tree(\mathcal{P}(At))^2)$$

der Verhaltensäquivalenz $\sim_{\mathcal{T}(At)}$ Folgendes: Für alle $R \subseteq Tree(\mathcal{P}(At))^2$,

$$\begin{array}{rcl} F_{\mathcal{T}(At)}(R) & = & \{(t, t') \in Tree(\mathcal{P}(At))^2 \mid t(\epsilon) = t'(\epsilon), \\ & & \forall\, i \in dom(t)\ \exists\, j \in dom(t') : (\lambda w.t(iw), \lambda w.t'(jw)) \in R, \\ & & \forall\, j \in dom(t')\ \exists\, i \in dom(t) : (\lambda w.t(iw), \lambda w.t'(jw)) \in R\}. \end{array}$$

M.a.W.: Zwei Bäume $t, t' \in Tree(\mathcal{P}(At))$ sind genau dann $\mathcal{T}(At)$-verhaltensäquivalent, wenn für die maximalen Unterbäume $t_1, \ldots, t_m$ bzw. $t'_1, \ldots, t'_n$ von t bzw. t' Folgendes gilt: Für alle $1 \leq i \leq m$ gibt es $1 \leq j \leq n$ mit $t_i \sim_{\mathcal{T}(At)} t'_j$ und für alle $1 \leq j \leq n$ gibt es $1 \leq i \leq m$ mit $t_i \sim_{\mathcal{T}(At)} t'_j$.

Den Quotienten $Beh(At) =_{def} \mathcal{T}(At)_{min}$ (siehe Abschnitt 7.3) nennen wir **Verhaltensmodell**.

Er ist **final** in $coAlg_{F_{At}}$, d.h. von jeder F-Coalgebra gibt es *genau einen* Kripke-Morphismus nach $Beh(At)$.

Aufgabe Zeigen Sie, dass alle finalen F_{At}-Coalgebren Kripke-isomorph sind.

Ähnlich wie eine Σ-Algebra A durch die Termfaltung

$$fold^A : T_\Sigma \to A$$

mit der Termalgebra T_Σ in Beziehung steht (siehe Kapitel 5), so verbindet ein Kripke-Morphismus

$$unfold^{\mathcal{K}} : \mathcal{K} \to Beh(At)$$

eine F-Coalgebra $\mathcal{K}$ mit dem Verhaltensmodell $Beh(At)$.

Während $fold^A(t)$ den Term t zu einem Wert in A *faltet*, entfaltet $unfold^{\mathcal{K}}(s)$ den Zustand s in zur $\sim_{\mathcal{T}(At)}$-Äquivalenzklasse aller $\mathcal{P}(At)$-markierten Bäume, deren Pfade alle möglichen, in s beginnenden Zustandsfolgen repäsentieren.

Die Dualität zwischen $fold^A$ und $unfold^{\mathcal{K}}$ geht noch weiter:

- $fold^A$ ist der einzige Σ-Homomorphismus von T_Σ nach A (siehe Kapitel 5);
- $unfold^{\mathcal{K}}$ ist der einzige Kripke-Morphismus von $\mathcal{K}$ nach $Beh(At)$ (s.o.).

So wie Faltungen für alle Signaturen gleich definiert sind, so sind werden Entfaltungen für andere Automatentypen als den durch F_{At} analog zu $unfold^{\mathcal{K}}$ definiert. Konsequenzen der Eindeutigkeit von Entfaltungen wie das folgende Lemma gelten daher auch für andere Funktoren.

Satz 7.11 Lambeks Lemma (für Kripke-Strukturen)

Sei $State = Tree(\mathcal{P}(At))/\sim_{\mathcal{T}(At)}$. Das oben definierte Verhaltensmodell $Beh(At) : State \to F_{At}(State)$ ist bijektiv.

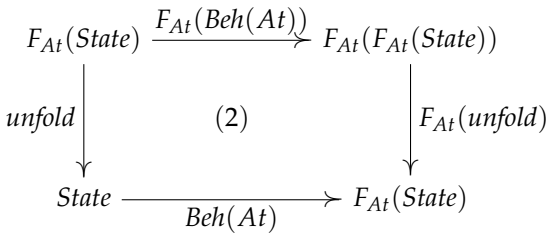
Beweis. Zunächst stellen wir fest, dass die Aussage nur gelten kann, weil $F_{At}(State)$ $\mathcal{P}_{fin}(State)$ und nicht $\mathcal{P}(State)$ enthält. Andernfalls könnte $Beh(At)$ wegen $S \ncong \mathcal{P}(S)$ für alle Mengen S (siehe Isomorphien) gar nicht bijektiv sein.

Das folgende kommutative Diagramm zeigt, dass $Beh(At)$ selbst ein Kripke-Morphismus von $Beh(At)$ in die Kripke-Struktur $F_{At}(Beh(At))$ ist:

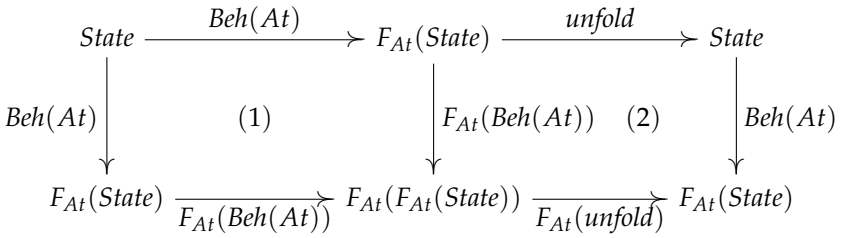
$$\begin{array}{ccc} State & \xrightarrow{\quad Beh(At) \quad} & F_{At}(State) \\ {\scriptstyle Beh(At)}\downarrow & (1) & \downarrow{\scriptstyle F_{At}(Beh(At))} \\ F_{At}(State) & \xrightarrow[F_{At}(Beh(At))]{} & F_{At}(F_{At}(State)) \end{array}$$

In umgekehrter Richtung gibt es einen weiteren Kripke-Morphismus:

Sei $unfold = unfold^{F_{At}(Beh(At))}$.



Aus der Kommutativität von (1) und (2) und der Funktoreigenschaft von F_{At} folgt, dass auch die Komposition $unfold \circ Beh(At)$ ein Kripke-Morphismus ist:



Da es von jeder Kripke-Struktur (hier: $Beh(At)$) nach $Beh(At)$ genau einen Kripke-Morphismus gibt, stimmt $unfold \circ Beh(At)$ mit der Identität auf $State$ überein, d.h.

$$unfold \circ Beh(At) = id_{State}. \tag{3}$$

Damit ist $Beh(At)$ bijektiv, wenn auch die Umkehrung gilt:

$$Beh(At) \circ unfold = id_{F_{At}(State)}. \tag{4}$$

Beweis von (4).

$$Beh(At) \circ unfold \overset{(2)}{=} F_{At}(unfold) \circ F_{At}(Beh(At)) = F_{At}(unfold \circ Beh(At))$$
$$\overset{(3)}{=} F_{At}(id_{State}) = id_{F_{At}(State)}. \qquad \text{❑}$$

Der Beweis von Lambeks Lemma zeigt an einem einfachen Beispiel, wie Eigenschaften von Modellen mit Hilfe von Funktionsdiagrammen ohne Bezug auf – die manchmal recht komplexen – konkreten Repräsentationen der Modelle gezeigt werden können. Das ist durchaus vergleichbar mit der Formulierung von Algorithmen in höheren Programmiersprachen, die gegenüber der Implementierung in maschinenorientierten Sprachen in der Regel weitaus kompakter und überschaubarer ist.

Modelleigenschaften sollten deshalb stets auf der Basis von allen irrelevanten Details befreiter Repräsentationen ihrer zugrundeliegenden Datenstrukturen verifiziert werden. In der Folge werden nicht nur die Beweise kürzer, nachvollziehbarer und fehlerfreier, sondern auch die Modelle flexibler, d.h. leichter an neue Anforderungen anpassbar (Stichwort *adaptive* und *agile* Software).

Bezogen auf die oben definierte Baumrepräsentation der Elemente von $Beh(At)$ besagt Lambeks Lemma, dass sich jeder Baum eindeutig zerlegen lässt in das Paar (T, A), das aus der Menge T seiner maximalen Unterbäume und seiner Wurzelmarkierung A besteht, und dass umgekehrt jedes solche Paar eindeutig einem Baum entspricht, dessen Menge maximaler Unterbäume mit T und dessen Wurzelmarkierung mit A übereinstimmt.

So gesehen, ist die Aussage von Lambeks Lemma trivial. Das Entscheidende ist aber, dass die Bijektivität von $Beh(At)$ nicht aus der Baumrepräsentation abgeleitet wurde, sondern allein aus der Charakterisierung von $Beh(At)$ als finale F-Coalgebra (s.o.). Da alle finalen F-Coalgebren Kripke-isomorph sind, gilt Lambeks Lemma also nicht nur für $Beh(At)$, sondern auch jede andere finale F-Coalgebra.

Der kategorientheoretische Zugang zu Kripke-Strukturen kann ähnlich auch für Σ-Algebren (siehe Kapitel 5) gewählt werden, allerdings in dualer Weise:

Eine Signatur Σ entspricht einem Funktor $H_\Sigma : Set \to Set$. Eine Σ-Algebra mit Trägermenge A entspricht keiner H_Σ-Coalgebra, sondern einer H_Σ-Algebra, d.h. einer Funktion $\alpha : H_\Sigma(A) \to A$. α ist die Summenextension aller Operationen der Σ-Algebra. Σ-Homomorphismen zwischen Σ-Algebren entsprechen Morphismen zwischen H_Σ-Algebren.

Näheres darüber in [34], Kapitel 13.

7.6 Jenseits von Kripke-Strukturen

Modallogik, Verhaltensäquivalenz, elementare Äquivalenz und das Verhaltensmodell lassen sich analog zur Kategorie $coAlg_{F_{At}}$ endlich verzweigter Kripke-Strukturen für andere Klassen von Coalgebren definieren. Intuitiv entspricht jede dieser Klassen einem **Automatentyp**.

Betrachten wir als herausragendes Beispiel Moore-Automaten, die in vielen Bereichen der Informatik – vom Schaltwerksentwurf bis zum Übersetzerbau – bei der Modellierung eingesetzt werden.

Ein **Moore-Automat** mit Eingabemenge In, Ausgabemenge Out ist eine Funktion

$$\mathcal{M} : State \to G(State),$$

wobei der Funktor G wie folgt definiert ist:

Für alle Mengen $State, State'$, $f : State \to State'$, $g : In \to State$ und $z \in Out$,

$$\begin{aligned} G(State) &= State^{In} \times Out, \\ G(f)(g, z) &= (f \circ g, z). \end{aligned}$$

$\mathcal{M}$ ordnet also jedem Zustand $s \in State$ ein Paar (g, z) zu, wobei $g : In \to State$ angibt, welche im Zustand s an den Automaten übergebene Eingabe zu welchem eindeutigen (!) Folgezustand führt, und z die im Zustand s erfolgte Ausgabe darstellt.

Im Gegensatz zu Kripke-Strukturen sind Moore-Automaten **deterministisch**. weil zwar auch hier jeder Zustand mehrere Nachfolger haben kann. Die jeweilige Eingabe legt aber fest, welcher davon ausgewählt wird. Außerdem gibt es immer einen Folgezustand.

Um Aussagen über Moore-Automaten zu formulieren, kann man die Operationen von AL verwenden und anstelle der modalen Operationen $\Diamond$ und $\Box$ eine Operation $\langle x \rangle$ für jede Eingabe $x \in In$. Die Signatur logischer Operationen lautet hier also

$$\Sigma = AL \cup \{\langle x \rangle \mid x \in In\}.$$

Out ist die Menge der atomaren Formeln.

Interpretiert werden alle Formeln in folgender Σ-Algebra $Pow(\delta)$ mit Trägermenge $\mathcal{P}(State)$: Für alle $x \in In$, $S \subseteq State$ und $s \in State$,

$$s \in \langle x \rangle^{Pow(\delta)}(S) \Leftrightarrow_{def} \delta(s)(x) \in S.$$

Sei $\mathcal{M} = \langle \delta, \beta \rangle : State \to G(State)$ ein Moore-Automat. Dann ist $g_\mathcal{M} : Out \to \mathcal{P}(State)$ wie folgt definiert: Für alle $z \in Out$ und $s \in State$,

$$s \in g_\mathcal{M}(z) \Leftrightarrow_{def} z = \beta(s).$$

Wieder liefert $g^*_\mathcal{M}(\varphi)$ die Menge der Zustände, die φ erfüllen.

Aufgabe Wie lauten wohl die Definitionen einer $\mathcal{M}$-Bisimulation, der $\mathcal{M}$-Verhaltensäquivalenz und eines Moore-Morphismus?

Aufgabe Wie lautet der Minimierungsalgorithmus für Moore-Automaten?

Das dem Verhaltensmodell $Beh(At)$ für Kripke-Strukturen (s.o.) entsprechende Verhaltensmodell $Beh(In, Out)$ für Moore-Automaten lässt sich als finale G-Coalgebra darstellen, also als diejenige G-Coalgebra, in die es von jeder G-Coalgebra aus genau einen Moore-Morphismus gibt.

Da Moore-Automaten deterministisch sind, besteht die Zustandsmenge $Beh(In, Out)$ nicht wie $Beh(At)$ aus (Äquivalenzklassen von) Bäumen, sondern aus der Menge der Funktionen von In^* nach Out. δ und β sind darauf wie folgt definiert:

$$\begin{array}{rcl} \delta : Out^{In^*} & \to & (In \to Out^{In^*}) \\ f & \mapsto & \lambda x.\lambda w.f(xw) \end{array} \qquad \begin{array}{rcl} \beta : Out^{In^*} & \to & Out \\ f & \mapsto & f(\epsilon) \end{array}$$

Aufgabe Zeigen Sie, dass Out^{In^*} im Fall $In = 1$ isomorph ist zur Menge $Out^{\mathbb{N}}$ der Ströme über Out.

Aufgabe Zeigen Sie, dass Out^{In^*} im Fall $Out = 2$ isomorph ist zur Menge $\mathcal{P}(In^*)$ der Teilmengen von In^*, die auch als Menge der **Sprachen über** In bezeichnet wird. Moore-Automaten vom Typ $State \to State^{In} \times 2$ heißen folglich auch (Sprach-)**erkennende Automaten**.

Mit Formeln der Modallogik für Moore-Automaten kann z.B. die Bedeutung imperativer (= befehlsorientierter) Programme beschrieben werden. In ist dann die Menge der jeweils verwendeten Sprache Zuweisungen, Prozeduraufrufe, etc., Out die Menge der Ausgabewerte und $State$ die Menge der Belegungen von Programmvariablen mit Ausgabewerten.

Eine – **Zusicherung** genannte – Formel $\varphi \Rightarrow \langle p \rangle \psi$ drückt aus, dass nach Ausführung des Programms p ψ gilt, falls vorher φ galt. Z.B. besagt

$$(x = a) \Rightarrow \langle x := x * 2 \rangle (x = a * 2),$$

dass die Zuweisung $x := x * 2$ den Wert der Programmvariablen x verdoppelt.

Programme, die nicht auf jeder Eingabe terminieren, lassen sich auf der Basis einer weiteren Klasse von Coalgebren verifizieren, den **partiellen Automaten**. Der entsprechende Funktor H wie folgt:

Für alle Mengen $State$,

$$H(State) = (1 + State)^{In} \times Out.$$

Hier bieten sich *zwei* modale Operatoren für jede Eingabe an, so dass wir

$$\Sigma = AL \cup \{\langle x \rangle \mid x \in In\} \cup \{[x] \mid x \in In\}$$

als Signatur logischer Operationen für partielle Automaten erhalten. Out ist wieder die Menge der atomaren Formeln.

Interpretiert werden alle Formeln in folgender Σ-Algebra $Pow(\delta)$ mit Trägermenge $\mathcal{P}(State)$: Für alle $x \in In$, $S \subseteq State$ und $s \in State$,

$$\begin{array}{rcl} s \in \langle x \rangle^{Pow(\delta)}(S) & \Leftrightarrow_{def} & \delta(s)(x) \in S, \\ s \in [x]^{Pow(\delta)}(S) & \Leftrightarrow_{def} & \delta(s)(x) \in 1 + S. \end{array}$$

Die Zusicherung $\varphi \Rightarrow [p]\psi$ bedeutet, dass

- nach Ausführung von p ψ gilt, falls vorher φ galt, oder
- p nicht terminiert,

während $\varphi \Rightarrow \langle p \rangle \psi$ die Termination von p miteinschließt.

Andere Automatentypen werden verwendet, um zu zeigen, dass eine gegebene Formel φ von bestimmten Belegungen ihrer Variablen/Atome erfüllt wird. Das erfordert eine Semantik von φ in Form einer Teilmenge $Sem(\varphi)$ der Menge $\mathcal{B}$ aller in der jeweiligen Logik möglichen Belegungen, wie z.B. $\mathcal{B} = 2^{At}$ in der Aussagenlogik, $\mathcal{B} = \mathcal{P}(State)^{At}$ in der Modallogik und $\mathcal{B} = A^X$ in der Prädikaten- und der Gleichungslogik.

Ausgehend von einem Anfangszustand verarbeitet ein aus φ gebildeter **erkennender Automat** A_φ schrittweise jede Belegung $g \in \mathcal{B}$ so, dass der am Ende der Verarbeitung von g erreichte Zustand von A_φ angibt, ob g zu $Sem(\varphi)$ gehört oder nicht:

$$g \models \varphi \;\Leftrightarrow\; A_\varphi \text{ erkennt } g.$$

Der Automat A_φ wird induktiv über dem Aufbau von φ konstruiert. Damit er g *schrittweise* verarbeiten kann, muss auch jede Belegung von $\mathcal{B}$ eine charakteristische Struktur aufweisen, an der er sich "entlanghangeln" kann.

7.7 Highlights

Signatur der Modallogik: $ML = AL \cup \{\Box, \Diamond : 1 \to 1\}$
Semantik der Modallogik: ML-Algebra $Pow(\delta)$ mit Trägermenge $\mathcal{P}(State)$ und – von einer **Kripke-Struktur**

$$\mathcal{K} = \langle \delta, \beta \rangle : State \to (\mathcal{P}(State) \times \mathcal{P}(At))$$

abhängiger – folgender Interpretation von ML: Für alle $S, S' \subseteq State$,

$$\begin{aligned}
\bot^{Pow(\delta)} &= \emptyset, \\
\top^{Pow(\delta)} &= State, \\
\neg^{Pow(\delta)}(S) &= State \setminus S, \\
\wedge^{Pow(\delta)}(S, S') &= S \cap S', \\
\vee^{Pow(\delta)}(S, S') &= S \cup S', \\
\Rightarrow^{Pow(\delta)}(S, S') &= (State \setminus S) \cup S', \\
\Box^{Pow(\delta)}(S) &= \{s \in State \mid \forall\, s' \in \delta(s) : s' \in S\}, \\
\Diamond^{Pow(\delta)}(S) &= \{s \in State \mid \exists\, s' \in \delta(s) : s' \in S\}.
\end{aligned}$$

Die zweite Komponente von $\mathcal{K}$ induziert eine Belegung $g_\mathcal{K} : At \to \mathcal{P}(State)$ der Atome: Für alle $x \in At$ und $s \in State$,

$$g_\mathcal{K}(x) =_{def} \{s \in State \mid x \in \beta(s)\}.$$

$(\mathcal{K}, s)$ **erfüllt** φ oder $(\mathcal{K}, s)$ ist ein **Modell** von φ, geschrieben: $\mathcal{K} \models_s \varphi$, wenn s zu $g^*_\mathcal{K}(\varphi)$ gehört.

$\mathcal{K}$ heißt **endlich verzweigt**, wenn für alle $s \in State$ $\delta(s)$ endlich ist.

Sei $\mathcal{K}$ endlich verzweigt. Die folgende Funktion

$$ml2al : T_{ML}(At) \times State \to T_{AL}(At \times State)$$

übersetzt modallogische in aussagenlogische Formeln: Für alle $x \in At$, $s \in State$, $c \in \{\bot, \top\}$, $\otimes \in \{\vee, \wedge, \Rightarrow\}$ und $\varphi, \psi \in T_{ML}(At)$,

$$\begin{aligned}
ml2al(x, s) &= (x, s), \\
ml2al(c, s) &= c, \\
ml2al(\neg\varphi, s) &= \neg ml2al(\varphi, s), \\
ml2al(\varphi \otimes \psi, s) &= ml2al(\varphi, s) \otimes ml2al(\psi, s), \\
ml2al(\Box\varphi, s) &= \bigwedge\{ml2al(\varphi, s') \mid s' \in \delta(s)\}, \\
ml2al(\Diamond\varphi, s) &= \bigvee\{ml2al(\varphi, s') \mid s' \in \delta(s)\}.
\end{aligned}$$

Korrektheit der Übersetzung (Satz 7.1):

$$\mathcal{K} \models_s \varphi \quad\Leftrightarrow\quad uncurry(\chi \circ g_\mathcal{K}) \models ml2al(\varphi, s).$$

Alle in der Bitalgebra gültigen *AL*-Gleichungen sind auch in $Pow(\delta)$ gültig. Darüberhinaus erfüllt $Pow(\delta)$ die folgenden *AL*-Gleichungen:

$$\neg\Diamond x \equiv \Box\neg x \qquad \neg\Box x \equiv \Diamond\neg x \qquad (negM)$$
$$\Diamond(x \vee y) \equiv \Diamond x \vee \Diamond y \qquad \Box(x \wedge y) \equiv \Box x \wedge \Box y \qquad (distDia/distBox)$$

Sei $\mathcal{K}' = \langle \delta', \beta' \rangle : State' \to (\mathcal{P}(State') \times \mathcal{P}(At))$ eine weitere Kripke-Struktur.

$\mathcal{K}, \mathcal{K}'$-Verhaltensäquivalenz: $\sim_{\mathcal{K},\mathcal{K}'} =_{def} gfp(F_{\mathcal{K},\mathcal{K}'})$, wobei

$$\begin{array}{rcl} F_{\mathcal{K},\mathcal{K}'} : \mathcal{P}(State \times State') & \to & \mathcal{P}(State \times State') \\ & & \{(s,s') \in State \times State' \mid \beta(s) = \beta'(s'), \\ R & \mapsto & \quad \forall\, t \in \delta(s)\ \exists\, t' \in \delta'(s') : (t,t') \in R, \\ & & \quad \forall\, t' \in \delta'(s')\ \exists\, t \in \delta(s) : (t,t') \in R\} \end{array}$$

Elementare $\mathcal{K}, \mathcal{K}'$-Äquivalenz:

$$\approx_{\mathcal{K},\mathcal{K}'} =_{def} \{(s,s') \in State^2 \mid \forall\, \varphi \in T_{ML}(At) : \mathcal{K} \models_s \varphi \Leftrightarrow \mathcal{K}' \models_{s'} \varphi\}$$

Hennessy-Milner-Theorem: Sind $\mathcal{K}$ und $\mathcal{K}'$ endlich verzweigt, dann gilt $\sim_{\mathcal{K},\mathcal{K}'} = \approx_{\mathcal{K},\mathcal{K}'}$.

Minimale Kripke-Struktur:

$$\begin{array}{rcl} \mathcal{K}_{min} = \langle \delta_{min}, \beta_{min} \rangle : State/\!\sim_{\mathcal{K}} & \to & \mathcal{P}_{fin}(State/\!\sim_{\mathcal{K}}) \times \mathcal{P}(At) \\ [s] & \mapsto & (\{[s'] \mid s' \in \delta(s)\},\ \beta(s)) \end{array}$$

Für alle $s \in State$, $s \sim_{\mathcal{K},\mathcal{K}_{min}} [s]$.

Approximative Konstruktion der $\mathcal{K}$-Verhaltensäquivalenz $\sim_{\mathcal{K}} =_{def} \sim_{\mathcal{K},\mathcal{K}}$:

$$\begin{array}{rcl} \sim_0 & = & ker(\beta), \\ \sim_{n+1} & = & \{s \sim_n s' \mid \left\{ \begin{array}{l} \forall\, t \in \delta(s)\ \exists\, t' \in \delta(s') : t \sim_n t', \\ \forall\, t' \in \delta(s')\ \exists\, t \in \delta(s) : t \sim_n t' \end{array} \right\}, \\ \sim_\omega & = & \bigcap_{n \in \mathbb{N}} \sim_n . \end{array}$$

Satz 7.8: Ist $\mathcal{K}$ endlich verzweigt, dann gilt $\sim_\omega = \sim_{\mathcal{K}}$.

Satz 7.9: Ist $\mathcal{K}$ endlich verzweigt, dann gilt $\sim_{\mathcal{K}_{min}} = \Delta_{State/\sim_{\mathcal{K}}}$.

Minimierung durch schrittweise Bildung von Dreiecksmatrizen:

Für alle $n \in \mathbb{N}$ und $s, s' \in State$ mit $s < s'$,

$$\begin{array}{rcl} M_0(s,s') & = & \left\{ \begin{array}{ll} (\delta(s), \delta(s')) & \text{falls } \beta(s) = \beta(s'), \\ * & \text{sonst,} \end{array} \right. \\ M_{n+1}(s,s') & = & \left\{ \begin{array}{ll} M_n(s,s') & \text{falls für alle } (S,S') \in M_n(s,s') \text{ Folgendes gilt:} \\ & \forall\, t \in S\ \exists\, t' \in S' : t = t' \vee M_n(min\{t,t'\}, max\{t,t'\}) \neq *, \\ & \forall\, t' \in S'\ \exists\, t \in S : t = t' \vee M_n(min\{t,t'\}, max\{t,t'\}) \neq *, \\ * & \text{sonst,} \end{array} \right. \\ R_n & = & \{(s,s') \in State^2 \mid s < s',\ M_n(s,s') \neq *\}. \end{array}$$

Daraus folgt $\sim_n \ = \ \Delta_{State} \cup R_n \cup R_n^{-1}$. (1)

Ist *State* endlich, dann gibt es eine kleinste natürliche Zahl k mit $M_k = M_n$, also auch $R_k = R_n$ für alle $n \geq k$. Folglich stimmt die $\mathcal{K}$-Verhaltensäquivalenz wegen (1) und nach Satz 7.8 mit $\Delta_{State} \cup R_k \cup R_k^{-1}$ überein.

8 Prädikatenlogik (predicate logic, first-order logic)

In der Prädikatenlogik sind Atome keine Variablen, sondern Ausdrücke pt mit einem *Prädikat* p und einem Tupel $t = (t_1, \ldots, t_n)$ von Termen über einer Menge X von **Individuenvariablen**.

p wird als n-stellige Relation interpretiert, Individuenvariable stehen für beliebige Elemente der Trägermenge einer Algebra A, die die Terme interpretiert (siehe Kapitel 5). Die Erweiterung von A um die Interpretation der Prädikate wird **Struktur** genannt.

Zu den aussagenlogischen Operationen kommen für jede Individuenvariable x die beiden einstelligen Operationen $\exists x$ und $\forall x$ (**Existenz-** bzw. **Allquantor**) hinzu, die x *binden*, was bedeutet, dass der Wert einer Formel $\exists x\varphi$ oder $\forall x\varphi$ unter einer Belegung $g : X \to A$ nicht von $g(x)$ abhängt.

Modallogisch ausgedrückt, sind die Belegungen von X in A die *Zustände*, die den Wahrheitswert einer Formel bestimmen. Dementsprechend könnte man Belegungen von At in 2 als aussagenlogische Zustände bezeichnen, was suggeriert, dass – zumindest von der Semantik her – Aussagenlogik und Prädikatenlogik spezielle Modallogiken sind. Was die Parameter der Auswertung einer Formel angeht, so liegt die Prädikatenlogik jedoch eher *zwischen* Aussagen- und Modallogik:

- In der Aussagenlogik gibt es nur einen Parameter, nämlich eine Belegung der Atome durch Wahrheitswerte.
- In der Modallogik hängt der Wert einer Formel von einer Kripke-Struktur ab.
- In der Prädikatenlogik gibt es drei Parameter: eine Belegung der Individuenvariablen durch Elemente (der Trägermenge) einer Algebra A, eine Interpretation der (nicht-logischen) Operationen, die in den Termen der Formel vorkommen, und eine Interpretation der Prädikate durch Relationen auf A. Die letzten beiden Parameter werden zu einer *Struktur* im u.g. Sinne zusammengefasst.

Syntax

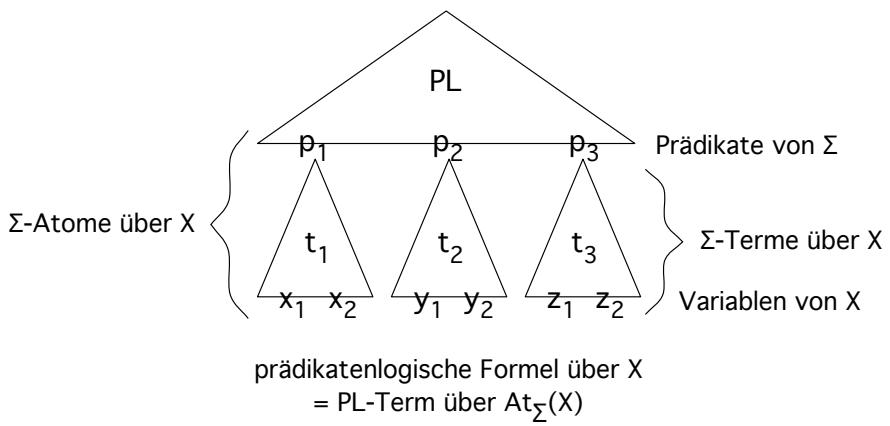
Sei X eine Menge von **Individuenvariablen**.

$$PL = AL \cup \{(\exists x) : 1 \to 1 \mid x \in X\} \cup \{(\forall x) : 1 \to 1 \mid x \in X\}$$

ist die Signatur der **prädikatenlogischen Operationen**.

Eine Signatur Σ gemäß Kapitel 5 besteht aus Operationen. Zusätzlich kann Σ jetzt auch – **Prädikate** genannte – Relations*symbole* enthalten, die – wie die Operationen – jeweils eine feste Stelligkeit (> 0) haben. Zur Unterscheidung von Operationen $f : n \to 1$ schreiben wir $p : n$ für ein Prädikat mit Stelligkeit n. Ein-, zwei- bzw. dreistellige Prädikate werden **unär**, **binär** bzw. **ternär** genannt.

Ein Ausdruck der Form pt heißt **Σ-Atom über** X, falls $p : n$ ein Prädikat von Σ und $t = (t_1, \ldots, t_n)$ ein n-Tupel von Σ-Termen über X ist. $At_\Sigma(X)$ bezeichnet die Menge aller Σ-Atome über X. $At_\Sigma =_{def} At_\Sigma(\emptyset)$.



Ein PL-Term φ über $At_\Sigma(X)$ heißt **(prädikatenlogische) Σ-Formel über** X.

Semantik

Eine **Σ-Struktur** A ist eine Σ-Algebra, die für alle $n \in \mathbb{N}$ jedem n-stelligen Prädikat $p \in \Sigma$ eine n-stellige Relation $p^A \subseteq A^n$ zuordnet.

Hat A die Trägermenge $T_\Sigma(X)$ oder T_Σ und sind die Operationen von Σ dort so interpretiert wie in Kapitel 5, dann heißt A **Termstruktur** oder **Herbrand-Struktur**.

$Struct_\Sigma$ bezeichnet die Klasse aller Σ-Strukturen.

Σ-Homomorphismen $h : A \to B$ zwischen Σ-Strukturen A und B sind mit den jeweiligen Interpretationen nicht nur der Operationen, sondern auch der Prädikate von Σ verträglich, d.h. für alle Prädikate $p \in \Sigma$ ist $h(p^A) \subseteq p^B$.

Entsprechendes gilt für **Σ-Kongruenzen** R auf einer Σ-Struktur A (siehe Kapitel 5):

Neben der Verträglichkeit von $\sim$ mit Σ wird gefordert, dass für alle Prädikate $p : n \in \Sigma$ und $a_1, \ldots, a_n, b_1, \ldots, b_n \in A$ Folgendes gilt:

$$(a_1\, b_1) \in R \wedge \cdots \wedge (a_n\, b_n) \in R \quad \Rightarrow \quad (a_1, \ldots, a_n) \in p^A \Leftrightarrow (b_1, \ldots, b_n) \in p^A.$$

Die Quotientenalgebra A/R wird zur **Quotientenstruktur** erweitert, die alle Prädikate p von Σ wie folgt interpretiert: Für alle $a_1, \ldots, a_n \in A^n$,

$$p^{A/R} = \{nat_R(a) \mid a \in p^A\}.$$

In der Modallogik trat an die Stelle der Bitalgebra, auf der die Semantik aussagenlogischer Formeln aufbaut, die *ML*-Algebra $Pow(\delta)$, wobei δ die Transitionsfunktion der jeweils zugrundeliegenden Kripke-Struktur $\mathcal{K}$ ist. Hat $\mathcal{K}$ die Zustandsmenge *State*, dann ist $\mathcal{P}(State)$ die Trägermenge von $Pow(\delta)$. Wie im Abschnitt Gleichungslogik in anderen Logiken von Kapitel 5 erwähnt wurde, werden Zustände in der Prädikatenlogik als Belegungen von X in einer Σ-Struktur A dargestellt.

Folglich erhält man anstelle von $Pow(\delta)$ die wie folgt definierte *PL*-Algebra $Pow(A)$ mit Trägermenge $\mathcal{P}(A^X)$: Für alle Σ-Strukturen A und $S, S' \subseteq A^X$,

$$\begin{aligned}
\bot^{Pow(A)} &= \emptyset, \\
\top^{Pow(A)} &= A^X, \\
\neg^{Pow(A)}(S) &= A^X \setminus S, \\
\wedge^{Pow(A)}(S, S') &= S \cap S', \\
\vee^{Pow(A)}(S, S') &= S \cup S', \\
\Rightarrow^{Pow(A)}(S, S') &= (A^X \setminus S) \cup S', \\
(\forall x)^{Pow(A)}(S) &= \{h \in A^X \mid \forall\, a \in A : h[a/x] \in S\}, \\
(\exists x)^{Pow(A)}(S) &= \{h \in A^X \mid \exists\, a \in A : h[a/x] \in S\}.
\end{aligned}$$

Die Σ-Struktur A bestimmt die Belegung $g_A : At_\Sigma(X) \to \mathcal{P}(A^X)$, unter der prädikatenlogische Formeln in $Pow(A)$ – durch Anwendung der aus g_A abgeleiteten Auswertungsfunktion g_A^* (siehe Kapitel 5) – ausgewertet werden:

Für alle $pt \in At_\Sigma(X)$,

$$g_A(pt) =_{def} \{h \in A^X \mid h^*(t) \in p^A\}. \tag{1}$$

$$\begin{array}{ccc}
At_\Sigma(X) & \xrightarrow{inc_{At_\Sigma(X)}} & T_{PL}(At_\Sigma(X)) \\
& {\scriptstyle g_A} \searrow \quad = \quad \swarrow {\scriptstyle g_A^*} & \\
& \mathcal{P}(A^X) &
\end{array}$$

Sei $\varphi, \psi \in T_{PL}(At_\Sigma(X))$ und $h : X \to A$.

(A,h) **erfüllt** φ oder ist ein **Modell** von φ, geschrieben: $A \models_h \varphi$, wenn h zu $g^*_A(\varphi)$ gehört. h heißt dann auch **Lösung von** φ **in** A.

A **erfüllt** φ oder ist ein **Modell** von φ, geschrieben: $A \models \varphi$, wenn $A \models_h \varphi$ für alle $h \in A^X$ gilt. Eine Termstruktur, die φ erfüllt, heißt **Termmodell von** φ.

Für alle $\Phi \subseteq T_{PL}(At_\Sigma(X))$ bezeichnet $Struct_{\Sigma,\Phi}$ die Klasse der Σ-Strukturen, die (alle Formeln von) Φ erfüllen.

Darauf aufbauend sind **Erfüllbarkeit**, **Allgemeingültigkeit**, **Folgerung** und **Äquivalenz** prädikatenlogischer Formeln wie am Anfang von Kapitel 2 definiert, wobei der zugrundeliegende semantische Bereich D gegeben ist durch $Struct_\Sigma$ bzw. die Menge aller Paare, die aus einer Σ-Struktur A und einer Variablenbelegung $h : X \to A$ bestehen.

Wir schreiben $\varphi \Leftrightarrow^A_h \psi$, falls (A,h) genau dann φ erfüllt, wenn (A,h) ψ erfüllt, und $\varphi \Leftrightarrow^A \psi$, falls A genau dann φ erfüllt, wenn A ψ erfüllt.

Lemma 8.1 (Vollständigkeit von Termstrukturen)

Sei A eine Σ-Struktur und $H(A)$ die Termstruktur mit folgender Interpretation der Prädikate $p : n \in \Sigma$:

$$p^{H(A)} =_{def} \{t \in T^n_\Sigma \mid fold^A(t) \in p^A\} \tag{2}$$

(siehe Kapitel 5).

Sei φ eine quantorenfreie Σ-Formel mit $A \models \varphi$. Dann ist auch $H(A)$ ein Modell von φ.

Beweis. Durch strukturelle Induktion über $T_{AL}(At_\Sigma(X))$ zeigen wir, dass für alle quantorenfreien Σ-Formeln φ Folgendes gilt:

$$g^*_{H(A)}(\varphi) \;=\; \{\sigma \in T^X_\Sigma \mid fold^A \circ \sigma \in g^*_A(\varphi)\}. \tag{3}$$

Für alle $pt \in At_\Sigma(X)$,

$$\begin{aligned} &g^*_{H(A)}(pt) = g_{H(A)}(pt) \overset{(1)}{=} \{\sigma \in T^X_\Sigma \mid t\sigma \in p^{H(A)}\} \overset{(2)}{=} \{\sigma \in T^X_\Sigma \mid fold^A(t\sigma) \in p^A\} \\ &\overset{\text{Lemma 5.3 } (i)}{=} \{\sigma \in T^X_\Sigma \mid (fold^A \circ \sigma)^*(t) \in p^A\} \\ &= \{\sigma \in T^X_\Sigma \mid fold^A \circ \sigma \in g_A(pt)\} = \{\sigma \in T^X_\Sigma \mid fold^A \circ \sigma \in g^*_A(pt)\}. \end{aligned}$$

Für alle $\varphi, \psi \in T_{AL}(At_\Sigma(X))$,

$$\begin{aligned} &g^*_{H(A)}(\neg\varphi) = T^X_\Sigma \setminus g^*_{H(A)}(\varphi) \overset{\text{ind. hyp.}}{=} T^X_\Sigma \setminus \{\sigma \in T^X_\Sigma \mid fold^A \circ \sigma \in g^*_A(\varphi)\} \\ &= \{\sigma \in T^X_\Sigma \mid fold^A \circ \sigma \in A^X \setminus g^*_A(\varphi)\} = \{\sigma \in T^X_\Sigma \mid fold^A \circ \sigma \in g^*_A(\neg\varphi)\}, \end{aligned}$$

$$\begin{aligned} &g^*_{H(A)}(\varphi \wedge \psi) = g^*_{H(A)}(\varphi) \cap g^*_{H(A)}(\varphi) \overset{\text{ind. hyp.}}{=} \{\sigma \in T^X_\Sigma \mid fold^A \circ \sigma \in g^*_A(\varphi) \cap g^*_A(\psi)\} \\ &= \{\sigma \in T^X_\Sigma \mid fold^A \circ \sigma \in g^*_A(\varphi \wedge \psi)\}, \end{aligned}$$

$$\begin{aligned} &g^*_{H(A)}(\varphi \vee \psi) = g^*_{H(A)}(\varphi) \cup g^*_{H(A)}(\varphi) \overset{\text{ind. hyp.}}{=} \{\sigma \in T^X_\Sigma \mid fold^A \circ \sigma \in g^*_A(\varphi) \cup g^*_A(\psi)\} \\ &= \{\sigma \in T^X_\Sigma \mid fold^A \circ \sigma \in g^*_A(\varphi \vee \psi)\}. \end{aligned}$$

Damit endet der Beweis von (3).

Sei A ein Modell von φ und $\sigma \in T^X_\Sigma$. Dann gilt $g^*_A(\varphi) = A^X$, also insbesondere $h^* \circ \sigma \in g^*_A(\varphi)$ für alle $h \in A^X$. Also folgt aus (3), dass σ zu $g^*_{H(A)}(\varphi)$ gehört. Demnach gilt $g^*_{H(A)}(\varphi) = T^X_\Sigma$. Also ist $H(A)$ ein Modell von φ. ❑

Sei φ eine beliebige Σ-Formel über X.

$var(\varphi)$ bezeichnet die Menge aller Variablen von X, die in φ vorkommen, aber nicht nur als Teil einer Operation wie $\exists x$ oder $\forall x$.

$x \in V$ **kommt in** φ **gebunden vor**, wenn es eine Teilformel $\exists x\psi$ oder $\forall x\psi$ von φ mit $x \in var(\psi)$ gibt.

$x \in V$ ist eine **freie Variable von** φ, wenn x mindestens einmal nicht gebunden in φ vorkommt. $free(\varphi)$ bezeichnet die Menge der freien Variablen von φ.

φ heißt **geschlossen**, wenn $free(\varphi)$ leer ist.

Sei $\{x_1, \ldots, x_n\} = free(\varphi)$.

$all(\varphi) =_{def} \forall x_1 \ldots \forall x_n \varphi$ heißt **Allabschluss von** φ.

$any(\varphi) =_{def} \exists x_1 \ldots \exists x_n \varphi$ heißt **Existenzabschluss von** φ.

Aufgabe Zeigen Sie (1)-(3) von Kapitel 6 für geschlossene prädikatenlogische Formeln.

Substitution von Variablen einer prädikatenlogischen Formel durch Terme

Sei φ eine prädikatenlogische Σ-Formel über X und

$$\sigma : X \to T_\Sigma(X).$$

Da $T_\Sigma(X)$ eine Σ-Algebra ist, gibt es laut Kapitel 5 den Σ-Homomorphismus

$$\sigma^* : T_\Sigma(X) \to T_\Sigma(X),$$

der jedem Term t die Σ-Instanz von t zuordnet, d.h. denjenigen Term, der aus t durch Ersetzung der Variablen von t durch ihre jeweiligen Bilder unter σ zuordnet.

Wollen wir eine solche Ersetzung in den Termen einer prädikatenlogischen Formel durchführen, müssen wir berücksichtigen, dass Variablen durch Quantoren gebunden sein können. Diese Variablen dürfen nicht ersetzt werden!

Außerdem würde durch die Ersetzung freier Variablen durch Terme, in denen zufälligerweise Variablen vorkommen, die in der Formel gebunden sind, eine zusätzliche Bindung entstehen, die durch die Umbenennung der gebundenen Variablen verhindert werden muss.

Die Funktion

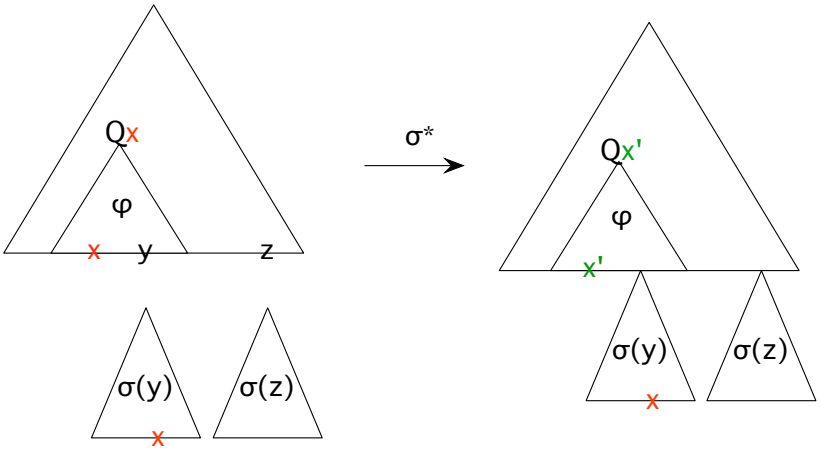
$$\sigma^* : T_{PL}(At_\Sigma(X)) \to T_{PL}(At_\Sigma(X)),$$

die die Ersetzung durchführt und dabei die Existenz von Quantoren in obiger Weise berücksichtigt, setzt die gleichnamige Funktion $\sigma^* : T_\Sigma(X) \to T_\Sigma(X)$ von Termen auf Formeln fort. Sie ist wie folgt definiert:

- Für alle $f : n \to 1 \in \Sigma$ und $t \in T_\Sigma(X)^n$, $\sigma^*(ft) = f\sigma^*(t)$.
- Für alle $p : n \in \Sigma$ und $t \in T_\Sigma(X)^n$, $\sigma^*(pt) = p\sigma^*(t)$.
- Für alle $f : n \to 1 \in AL$ und $\varphi \in T_{PL}(At_\Sigma(X))^n$, $\sigma^*(ft) = f\sigma^*(\varphi)$.
- Für alle $Q \in \{\exists, \forall\}$, $x \in X$ und $\varphi \in T_{PL}(At_\Sigma(X))$,

$$\sigma^*(Qx\varphi) = \begin{cases} Qx'\sigma[x'/x]^*(\varphi) & \text{falls } x \in V_{\varphi,\sigma,x} =_{def} \bigcup var(\sigma(free(\varphi) \setminus \{x\})), \\ Qx\sigma[x/x]^*(\varphi) & \text{sonst.} \end{cases}$$

Wegen $\sigma[x/x](x) = x$ verhindert $\sigma[x/x]$ die Ersetzung von x durch $\sigma(x)$.



Offenbar gehört x genau dann zu $V_{\varphi,\sigma,x}$, wenn x in einem Term t vorkommt, der bei der Anwendung von $\sigma[x/x]^*$ auf φ eine freie Variable von φ substituiert und damit ein neues gebundenes Vorkommen von x in φ einführen würde, wenn wir x nicht durch x' ersetzen würden. Wie folgendes Beispiel zeigt, kann das bewirken, dass $Qx\varphi$ erfüllbar ist, die σ-Instanz von $Qx\varphi$ jedoch nicht!

Sei $\varphi \in T_{PL}(At_\Sigma(X))$ und $\sigma, \sigma_1, \ldots, \sigma_n : X \to T_\Sigma(X)$.

Fortan schreiben wir analog zu Kapitel 5

$$\begin{array}{rll} \varphi\sigma & \text{anstelle von} & \sigma^*(\varphi), \\ \sigma_1 \ldots \sigma_n & \text{anstelle von} & \sigma_n^* \circ \cdots \circ \sigma_2^* \circ \sigma_1. \qquad \text{(Kleisli-Komposition)} \end{array}$$

Beispiel 8.2

Sei $\varphi = \exists x p(x,y)$, $A = \{a,b\}$, $p^A = \{(a,b)\}$ und $h = \lambda z.b$. Wir erhalten

$$\begin{aligned} A \models_h \varphi \Leftrightarrow h \in g_A^*(\varphi) \quad &\Leftrightarrow \{h[a/x], h[b/x]\} \cap g_A(p(x,y)) \neq \emptyset \\ &\Leftrightarrow \{h[a/x], h[b/x]\} \cap \{h' \in A^X \mid (h'(x), h'(y)) \in p^A\} \neq \emptyset \\ &\Leftrightarrow \{(h[a/x](x), h[a/x](y)), (h[b/x](x), h[b/x](y))\} \cap p^A \neq \emptyset \\ &\Leftrightarrow \{(a,b),(b,b)\} \cap p^A \neq \emptyset. \qquad (1) \end{aligned}$$

Aus der Gültigkeit von (1) folgt

$$A \models_h \varphi. \qquad (2)$$

Sei $\sigma = \{x/y\}$ (siehe Abschnitt 5.2). Dann ist $V_{\varphi,\sigma,x} = \{x\}$. Ohne Umbenennung erhält man

$$\begin{aligned} &A \models_h \varphi\sigma \Leftrightarrow h \in g_A^*(\varphi\sigma) = g_A^*((\exists x p(x,y))\sigma) \overset{!!}{=} g_A^*(\exists x p(x,y)\sigma[x/x]) = g_A^*(\exists x p(x,x)) \\ &\Leftrightarrow \{h[a/x], h[b/x]\} \cap g_A(p(x,x)) \neq \emptyset \\ &\Leftrightarrow \{h[a/x], h[b/x]\} \cap \{h' \in A^X \mid (h'(x), h'(x)) \in p^A\} \neq \emptyset \\ &\Leftrightarrow \{(h[a/x](x), h[a/x](x)), (h[b/x](x), h[b/x](x))\} \cap p^A \neq \emptyset \\ &\Leftrightarrow \{(a,a),(b,b)\} \cap p^A \neq \emptyset. \qquad (3) \end{aligned}$$

Da (3) nicht gilt, ist auch $A \not\models_h \varphi\sigma$ verletzt. ☹

Mit Umbenennung gilt jedoch:

$$\begin{aligned} &A \models_h \varphi\sigma \Leftrightarrow h \in g_A^*(\varphi\sigma) = g_A^*((\exists x p(x,y))\sigma) \overset{!!}{=} g_A^*(\exists x' p(x,y)\sigma[x'/x]) = g_A^*(\exists x' p(x',x)) \\ &\Leftrightarrow \{h[a/x'], h[b/x']\} \cap \{h' \in A^X \mid (h'(x'), h'(x)) \in p^A\} \neq \emptyset \\ &\Leftrightarrow \{(h[a/x'](x'), h[a/x'](x)), (h[b/x'](x'), h[b/x'](x))\} \cap p^A \neq \emptyset \\ &\Leftrightarrow \{(a,b),(b,b)\} \cap p^A \neq \emptyset. \qquad (4) \end{aligned}$$

Aus der Gültigkeit von (4) folgt $A \models_h \varphi\sigma$. ☺

Da h eine konstante Funktion ist, gilt $h = h^* \circ \sigma$, also wegen (2) auch $A \models_{h^* \circ \sigma} \varphi$. ❑

Lemma 8.3 (Substitutionslemma für prädikatenlogische Formeln; vgl. Lemma 5.3)

Für alle Σ-Strukturen A, $h : X \to A$, $\sigma : X \to T_\Sigma(X)$ und $\varphi \in T_{PL}(At_\Sigma(X))$,

$$A \models_h \varphi\sigma \;\Leftrightarrow\; A \models_{h^* \circ \sigma} \varphi. \tag{1}$$

Beweis. Wir zeigen zunächst, dass für alle $\varphi \in T_{PL}(At_\Sigma(X))$, $x \in X$, $y \in X \setminus V_{\varphi,\sigma,x}$, $z \in free(\varphi)$ und $a \in A$ Folgendes gilt:

$$(h[a/y]^* \circ \sigma[y/x])(z) = (h^* \circ \sigma)[a/x](z). \tag{2}$$

Fall 1: $z = x$. Wegen

$$(h[a/y]^* \circ \sigma[y/x])(x) = h[a/y]^*(\sigma[y/x](x)) = h[a/y]^*(y) = h[a/y](y) = a = (h^* \circ \sigma)[a/x](x)$$

gilt (2).

Fall 2: $z \in free(\varphi) \setminus \{x\}$. Aus $var(\sigma(z)) \subseteq V_{\varphi,\sigma,x}$ und $y \notin V_{\varphi,\sigma,x}$ folgt $y \notin var(\sigma(z))$, also auch (2):

$$\begin{aligned}&(h[a/y]^* \circ \sigma[y/x])(z) = h[a/y]^*(\sigma[y/x](z)) \overset{x \neq z}{=} h[a/y]^*(\sigma(z)) \overset{y \notin var(\sigma(z))}{=} h^*(\sigma(z))\\ &= (h^* \circ \sigma)(z) \overset{x \neq z}{=} (h^* \circ \sigma)[a/x](z).\end{aligned}$$

Nun zeigen wir (1) durch strukturelle Induktion über $T_{PL}(At_\Sigma(X))$.

Für alle $pt \in At_\Sigma(X)$,

$$\begin{aligned}&A \models_h pt\sigma \;\Leftrightarrow\; h \in g_A^*(pt\sigma) = g_A(pt\sigma) \;\Leftrightarrow\; (h^* \circ \sigma)^*(t) \overset{\text{Lemma 5.3 (i)}}{=} h^*(t\sigma) \in p^A\\ &\Leftrightarrow\; h^* \circ \sigma \in g_A(pt) = g_A^*(pt) \;\Leftrightarrow\; A \models_{h^* \circ \sigma} pt.\end{aligned}$$

Für alle $\varphi, \psi \in T_{PL}(At_\Sigma(X))$,

$$\begin{aligned}&A \models_h \neg\varphi\sigma \;\Leftrightarrow\; h \in g_A^*(\neg\varphi\sigma) = A^X \setminus g_A^*(\varphi\sigma) \overset{\text{ind. hyp.}}{\Leftrightarrow} h^* \circ \sigma \in A^X \setminus g_A^*(\varphi) = g_A^*(\neg\varphi)\\ &\Leftrightarrow\; A \models_{h^* \circ \sigma} \neg\varphi,\\ &A \models_h (\varphi \wedge \psi)\sigma \;\Leftrightarrow\; h \in g_A^*((\varphi \wedge \psi)\sigma) = g_A^*(\varphi\sigma) \cap g_A^*(\psi\sigma)\\ &\overset{\text{ind. hyp.}}{\Leftrightarrow} h^* \circ \sigma \in g_A^*(\varphi) \cap g_A^*(\psi) = g_A^*(\varphi \wedge \psi) \;\Leftrightarrow\; A \models_{h^* \circ \sigma} \varphi \wedge \psi\end{aligned}$$

und analog $A \models_h (\varphi \vee \psi)\sigma \;\Leftrightarrow\; A \models_{h^* \circ \sigma} \varphi \vee \psi$.

Sei $x \in X$ und $\varphi \in T_{PL}(At_\Sigma(X))$.

Nach Definition von σ^* gibt es eine Variable $y \in X \setminus V_{\forall x\varphi,\sigma,x}$ mit $(\forall x\varphi)\sigma = \forall y\varphi\sigma[y/x]$. Daraus folgt

$$\begin{aligned}&A \models_h (\forall x\varphi)\sigma \;\Leftrightarrow\; h \in g_A^*((\forall x\varphi)\sigma) = g_A^*(\forall y\varphi\sigma[y/x])\\ &\Leftrightarrow\; \forall\, a \in A : h[a/y] \in g_A^*(\varphi\sigma[y/x]) \overset{\text{ind. hyp.}}{\Leftrightarrow} \forall\, a \in A : h[a/y]^* \circ \sigma[y/x] \in g_A^*(\varphi)\\ &\overset{(2)}{\Leftrightarrow} \forall\, a \in A : (h^* \circ \sigma)[a/x] \in g_A^*(\varphi) \;\Leftrightarrow\; h^* \circ \sigma \in g_A^*(\forall x\varphi) \;\Leftrightarrow\; A \models_{h^* \circ \sigma} \forall x\varphi.\end{aligned}$$

Analog erhält man $A \models_h (\exists x\varphi)\sigma \;\Leftrightarrow\; A \models_{h^* \circ \sigma} \exists x\varphi$. ❑

Lemma 8.4 (Äquivalenzlemma für prädikatenlogische Formeln)

Sei $\varphi \in T_{PL}(At_\Sigma(X))$ und A eine Σ-Struktur.

(i) Für alle $\sigma, \tau : X \to T_\Sigma(X)$ mit $\sigma \equiv^A \tau$ gilt $\varphi\sigma \Leftrightarrow^A \varphi\tau$.

(ii) Für alle Σ-Kongruenzen $\sim$ auf A und $h, h' \in A^X$ gilt:

$$A \models_h \varphi \;\wedge\; h \sim h' \;\Rightarrow\; A \models_{h'} \varphi.$$

Beweis.

(i): Sei $h : X \to A$. $\sigma \equiv^A \tau$ impliziert

$$(h^* \circ \sigma)(x) = h^*(\sigma(x)) = h^*(\tau(x)) = (h^* \circ \tau)(x) \tag{3}$$

für alle $x \in X$. Daraus folgt

$$A \models_h \varphi\sigma \overset{(1)}{\Leftrightarrow} A \models_{h^* \circ \sigma} \varphi \overset{(3)}{\Leftrightarrow} A \models_{h^* \circ \tau} \varphi \overset{(1)}{\Leftrightarrow} A \models_h \varphi\tau,$$

also $\varphi\sigma \Leftrightarrow^A \varphi\tau$.

(ii) erhält man durch strukturelle Induktion über $T_{PL}(At_\Sigma(X))$. Sei $h \sim h'$.

Für alle $pt = p(t_1, \ldots, t_n) \in At_\Sigma(X)$ und $1 \leq i \leq n$ gilt $h^*(t_i) \sim h'^*(t_i)$, also

$$\begin{aligned} &A \models_h pt \;\Leftrightarrow\; h \in g_A^*(pt) = g_A(pt) \;\Leftrightarrow\; h^*(t) \in p^A \;\Leftrightarrow\; h'^*(t) \in p^A \\ &\Leftrightarrow\; h' \in g_A(pt) = g_A(pt) \;\Leftrightarrow\; A \models_{h'} pt. \end{aligned}$$

Die Induktionsschritte ergeben sich analog zum Beweis von (1) aus der induktiven Definition von g_A^*. ❑

8.1 Normalformen

Die Normalisierung prädikatenlogischer Formeln zielt darauf ab, sie in erfüllbarkeitsäquivalente aussagenlogische Formeln über $At_\Sigma(X)$ umzuwandeln, so dass ihre Erfüllbarkeit mit Hilfe zu prädikatenlogischen Verallgemeinerungen aussagenlogischer Schnitt- oder Tableau-Ableitungen überprüft werden kann.

Wir werden hier nur Schnittableitungen verallgemeinern. Tableau-Kalküle für prädikatenlogische Formeln werden in [8], [24] und [41] vorgestellt.

Aufgabe Sei A eine Σ-Struktur, $x, y \in X$, $Q \in \{\exists, \forall\}$ und φ, ψ, ϑ Σ-Formeln mit $x \notin free(\vartheta)$. Zeigen Sie, dass $Pow(A)$ folgende *PL*-Gleichungen erfüllt:

$\neg\exists x\varphi \equiv \forall x\neg\varphi$	$\neg\forall x\varphi \equiv \exists x\neg\varphi$	(*negQ*)
$\exists x(\varphi \vee \psi) \equiv \exists x\varphi \vee \exists x\psi$ $\exists x(\varphi \wedge \vartheta) \equiv \exists x\varphi \wedge \vartheta$	$\exists x(\vartheta \wedge \varphi) \equiv \vartheta \wedge \exists x\varphi$	(*distEx*)
$\forall x(\varphi \wedge \psi) \equiv \forall x\varphi \wedge \forall x\psi$ $\forall x(\varphi \vee \vartheta) \equiv \forall x\varphi \vee \vartheta$	$\forall x(\vartheta \vee \varphi) \equiv \vartheta \vee \forall x\varphi$	(*distAll*)
$Qx\vartheta \equiv \vartheta$		(*removeQ*)
$Qx\varphi \equiv Qy(\varphi\{y/x\})$		(*rename*)

Eine Σ-Formel heißt **Negationsnormalform über** $At_\Sigma(X)$ (NNF), wenn $\Rightarrow$ in ihr nicht vorkommt und $\neg$ nur direkt vor Atomen.

Seien Σ, Σ' Signaturen mit $\Sigma \subseteq \Sigma'$ und A eine Σ'-Struktur.

Das **Σ-Redukt** $A|_\Sigma$ von A ist die Σ-Struktur mit derselben Trägermenge und Interpretation der Symbole von Σ wie A.

Die Normalisierung einer prädikatenlogischen Formel zielt vor allem auf die Eliminierung ihrer Existenzquantoren ab. Wegen der dazu erforderlichen Einführung zusätzlicher Operationen in die Formel (s.u.) ist die normalisierte zur ursprünglichen nur in folgendem Sinne äquivalent:

Eine Σ-Formel φ und eine Σ'-Formel ψ heißen **erfüllbarkeitsäquivalent bzgl.** Σ, geschrieben: $\varphi \approx_\Sigma \psi$, wenn es für alle Σ-Strukturen A und $h \in A^X$ eine Σ'-Struktur B gibt, deren Σ-Redukt mit A übereinstimmt und die folgende

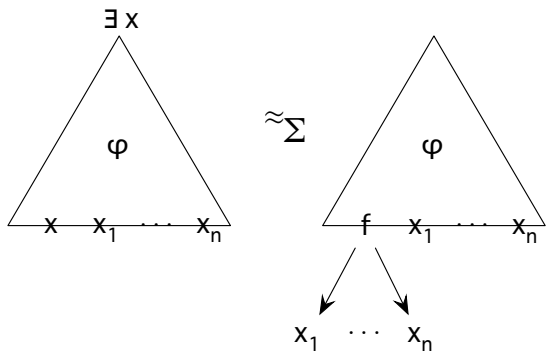
Äquivalenz erfüllt:

$$A \models_h \varphi \;\Leftrightarrow\; B \models_h \psi.$$

Aufgabe Zeigen Sie, dass die Erfüllbarkeitsäquivalenz bzgl. Σ' (!) mit der Äquivalenz in allen Σ'-Strukturen übereinstimmt. ❑

Satz 8.6

Sei $x \in X$, $\varphi \in T_{PL}(At_\Sigma(X))$, $\{x_1, \ldots, x_n\} = free(\varphi) \setminus \{x\}$, f eine – **Skolemfunktion** genannte – n-stellige Operation, $\Sigma' = \Sigma \uplus \{f\}$ und $\sigma = \{f(x_1, \ldots, x_n)/x\}$.



(i) $\exists x\varphi \;\approx_\Sigma\; \varphi\sigma$.

(ii) Sei $\vartheta \in T_{PL}(At_\Sigma(X))$ und $z \in X$. Dann gilt:

$$\vartheta\{\exists x\varphi/z\} \;\approx_\Sigma\; \vartheta\{\varphi\sigma/z\}.$$

Beweis von (i).

Sei B eine Σ'-Struktur mit $B|_\Sigma = A$, $h \in A^X$ und $a = f^B(h(x_1), \ldots, h(x_n))$.

Wir zeigen zunächst:

$$h^* \circ \sigma \;=\; h[a/x]. \tag{4}$$

Nach Definition von h^* gilt

$$h^*(\sigma(x)) = h^*(f(x_1, \ldots, x_n)) = f^B(h(x_1), \ldots, h(x_n)) = a = h[a/x](x)$$

und $h^*(\sigma(y)) = h(y) = h[a/x](y)$ für alle $y \in X \setminus \{x\}$. Also gilt (4). Daraus folgt

$$B \models_h \varphi\sigma \overset{(1)}{\Leftrightarrow} B \models_{h^* \circ \sigma} \varphi \overset{(4)}{\Leftrightarrow} B \models_{h[a/x]} \varphi. \tag{5}$$

Sei nun $A \models_h \exists x\varphi$. Dann gibt es $a \in A$ mit $A \models_{h[a/x]} \varphi$. Wir definieren eine Σ'-Struktur B wie folgt: $B|_\Sigma = A$ und für alle $(a_1, \ldots, a_n) \in A^n$, $f^B(a_1, \ldots, a_n) = a$. Da f in φ nicht vorkommt, gilt $B \models_{h[a/x]} \varphi$, also $B \models_h \varphi\sigma$ wegen (5).

Sei $B \models_h \varphi\sigma$ und $a = f^B(h(x_1), \ldots, h(x_n))$. (5) impliziert $B \models_{h[a/x]} \varphi$, also $A \models_{h[a/x]} \varphi$, weil f in φ nicht vorkommt. Daraus folgt $A \models_h \exists x\varphi$.

(ii) folgt aus (i), weil $\approx_\Sigma$ mit PL verträglich ist. ❑

Eine prädikatenlogische Formel $\forall x_1 \ldots \forall x_n \varphi$ heißt **Skolemnormalform (SNF)**, wenn φ eine implikative Normalform über $At_\Sigma(X)$ ist (siehe Kapitel 6).

Satz 8.7 Jede prädikatenlogische Formel ψ hat eine erfüllbarkeitsäquivalente Skolemnormalform.

Beweis. Zunächst wird ψ durch Anwendung von *negAL*- und *negQ*-Gleichungen in eine NNF ψ_1 überführt. Dann werden die Quantoren von ψ_1 durch Anwendung von *distEx*-, *distAll*-, *removeQ*- und *rename*-Gleichungen soweit wie möglich nach rechts geschoben.

Anschließend wird die resultierende Formel ψ_2 durch Anwendung von Gleichungen der Form

$$\vartheta\{\exists x\varphi/z\} \equiv \vartheta\{\varphi\sigma/z\}, \tag{6}$$

deren Komponenten die Bedingungen von Satz 8.6 (ii) erfüllen, in eine zu ψ_2 erfüllbarkeitsäquivalente Formel ψ_3 ohne Existenzquantoren transformiert.

Durch Anwendung von *rename*-Gleichungen und Inversen von *distAll*-Gleichungen werden die Allquantoren von ψ_3 nach links geschoben, bis eine äquivalente Formel

$$\forall x_1 \ldots \forall x_n \psi_4$$

mit paarweise verschiedenen Variablen $x_1, \ldots, x_n$ und quantorenfreier Formel ψ_4 entsteht.

ψ_4 wird durch die Anwendung gültiger *AL*-Gleichungen (siehe Kapitel 6) in eine äquivalente KNF ψ_5 transformiert und diese in die INF $\psi_6 = inf(\psi_5)$ (siehe Satz 6.1 (11)).

Zusammengefasst ergibt sich folgende Kette von Äquivalenzen, wobei A eine Σ-Struktur, $h \in A^X$, Σ' die durch die Anwendungen von (6) definierte Erweiterung von Σ und B die gemäß Satz 8.6 (i) aus (A, h) gebildete Σ'-Struktur ist:

$$\psi \Leftrightarrow_h^A \psi_1 \Leftrightarrow_h^A \psi_2 \approx_\Sigma \psi_3 \Leftrightarrow_h^B \forall x_1 \ldots \forall x_n \psi_4 \Leftrightarrow_h^B \forall x_1 \ldots \forall x_n \psi_5 \Leftrightarrow_h^B \forall x_1 \ldots \forall x_n \psi_6.$$

$\forall x_1 \ldots \forall x_n \psi_6$ ist also die gewünschte Skolemnormalform.

(6) könnte auch an anderer Stelle des Normalisierungsprozesses angewendet werden. Die hier genannte Schrittfolge stellt jedoch sicher, dass bei jeder Anwendung von (6) die Teilformel φ möglichst klein ist und daher die auf der rechten Seite von (6) eingeführte Operation f (siehe Satz 8.6 (i)) eine möglichst niedrige Stelligkeit hat. ❑

8.2 Schnittkalkül

Während die aussagenlogische Schnittregel das gleiche Atom aus zwei Gentzenformeln herausschneidet, ist die unten definierte prädikatenlogische Schnittregel auch auf zwei verschieden Atome anwendbar, sofern diese *unifizierbar* sind, d.h. eine gemeinsame Instanz haben (siehe Kapitel 5):

Sei $\sigma, \tau : X \to T_\Sigma(X)$. σ **subsumiert** τ, geschrieben: $\sigma \leq \tau$, wenn es $\rho : X \to T_\Sigma(X)$ mit $\sigma\rho = \tau$ gibt.

Seien t, u Atome oder Terme. σ **unifiziert** oder ist ein **Unifikator** von t und u, wenn $t\sigma = u\sigma$ gilt.

Subsumiert ein Unifikator von t und u alle Unifikatoren von t und u, dann nennen wir ihn einen **allgemeinster Unifikator** (most general unifier; **mgu**) von t und u.

Die folgenden Funktionen

$$\begin{aligned} unify : (At_\Sigma(X) \cup T_\Sigma(X))^2 &\to 1 + T_\Sigma(X)^X, \\ unify' : (T_\Sigma(X)^*)^2 &\to 1 + T_\Sigma(X)^X \end{aligned}$$

berechnen einen mgu zweier Atome oder Terme bzw. zweier gleichlanger Termtupel, sofern diese unifizierbar sind.

Funktionen eines Typs der Form $A \to 1 + B$ werden auf besondere Weise komponiert. Sei $f : A \to 1 + B$ und $g : B \to 1 + C$. Dann ist $g \odot f : A \to 1 + C$ wie folgt definiert: Für alle $a \in A$,

$$(g \odot f)(a) = \begin{cases} g(f(a)) & \text{falls } f(a) \neq *, \\ * & \text{falls } f(a) = *. \end{cases}$$

unify und *unify'* liefern den Wert $*$, wenn die zu unifizierenden Ausdrücke an der gleichen Baumposition mit verschiedenen Operationen oder Prädikaten markiert sind oder der **occur check** $x \in var(ft)$ (s.u.) erfolgreich ist. Er verhindert, dass eine Variable x durch einen Term t ersetzt wird, der keine Variable ist, aber x enthält. Wenn sie

nicht verhindert wird, würde sich eine solche Ersetzung unendlich oft wiederholen. M.a.W.: Ein Unifikator von x und t müsste x und t den *unendlichen* Term $t\{t/x\}\{t/x\}\ldots$ zuweisen. Der gehört aber nicht zu $T_\Sigma(X)$.

unify und *unify'* sind wie folgt definiert:

Für alle $f, g \in \Sigma$, $t, u \in T_\Sigma(X)$, $t', u' \in T_\Sigma(X)^*$ und $x \in X$,

$$\begin{array}{lcl}
unify(f(t'), g(u')) & = & \begin{cases} unify'(t', u') & \text{falls } f = g, \\ * & \text{sonst,} \end{cases} \\
unify(f(t'), x) & = & \begin{cases} \{f(t')/x\} & \text{falls } x \notin var(f(t')), \\ * & \text{sonst,} \end{cases} \\
unify(x, f(t')) & = & unify(f(t'), x), \\
unify(x, y) & = & \{x/y\}, \\
unify'(cons_t(t'), cons_u(u')) & = & unify'(t'\sigma, u'\sigma) \odot \sigma, \text{ wobei } \sigma = unify(t, u), \\
unify'(cons_t(t'), \epsilon) & = & *, \\
unify'(\epsilon, cons_t(t')) & = & *, \\
unify'(\epsilon, \epsilon) & = & inc_X
\end{array}$$

(siehe Abschnitt 4.7).

Liefert *unify* oder *unify'* den Wert $*$, dann scheitert der Algorithmus. Jeder von $*$ verschiedene Wert ist ein mgu der jeweiligen Parameterterme bzw. -atome. Da die Argumente von *unify* nur endlich viele Variablen enthalten, hat jede von *unify* berechnete Substitution σ endlichen **Support**

$$supp(\sigma) =_{def} \{x \in X \mid x\sigma \neq x\}.$$

Außerdem garantiert der – durch die Abfrage $x \in var(ft)$ realisierte – o.g. occur check, dass kein Term von $\sigma(supp(\sigma))$ Variablen von $supp(\sigma)$ enthält.

Der occur check kann auch dann zum Scheitern von $unify(t, t')$ führen, wenn keine Variable von t in t' vorkommt. Z.B. ersetzt *unify* bei der Unifikation von $f(x, x)$ und $f(y, g(y))$ zunächst die Variable x durch y, so dass der Algorithmus nun $f(y, y)$ und $f(y, g(y))$, also u.a. y mit $g(y)$ unifizieren muss.

Würde er jetzt nicht abbrechen, sondern y durch $g(y)$ ersetzen, dann hätte er im nächsten Schritt $f(g(y), g(y))$ und $f(g(y), g(g(y)))$ zu unifizieren, usw.

Eine injektive Substitution σ heißt **Variablenumbenennung**, wenn $\sigma(X)$ eine Teilmenge von X ist.

Aufgabe Zeigen Sie, dass es für alle $t, t' \in T_\Sigma(X)$ genau dann eine Variablenumbenennung σ mit $t\sigma = t'$ gibt, wenn t t' subsumiert und t' t subsumiert. ❑

Der **prädikatenlogische Schnittkalkül** *PSK* ist synthetisch und besteht aus vier Regeln:

Einführungsregel

$$\frac{}{\Phi \vdash \varphi} \quad \varphi \in \Phi \subseteq T_{PL}(At_\Sigma(X))$$

Summandregel

$$\frac{\Phi \vdash \varphi \Rightarrow \psi \vee \vartheta \vee \vartheta'}{\Phi \vdash \varphi\sigma \Rightarrow \psi\sigma \vee \vartheta\sigma} \quad \varphi, \psi \in T_{PL}(At_\Sigma(X)),\ \vartheta, \vartheta' \in At_\Sigma(X),\ \sigma = unify(\vartheta, \vartheta')$$

Faktorregel

$$\frac{\Phi \vdash \vartheta \wedge \vartheta' \wedge \varphi \Rightarrow \psi}{\Phi \vdash \vartheta\sigma \wedge \varphi\sigma \Rightarrow \psi\sigma} \quad \varphi, \psi \in T_{PL}(At_\Sigma(X)),\ \vartheta, \vartheta' \in At_\Sigma(X),\ \sigma = unify(\vartheta, \vartheta')$$

Prädikatenlogische Schnittregel (*binäre Resolution*)

$$\frac{\Phi \vdash \varphi \Rightarrow \psi \vee \vartheta, \quad \Phi \vdash \vartheta' \wedge \varphi' \Rightarrow \psi'}{\Phi \vdash \varphi\tau\sigma \wedge \varphi'\sigma \Rightarrow \psi\tau\sigma \vee \psi'\sigma}$$

$\varphi, \psi, \varphi', \psi' \in T_{PL}(At_\Sigma(X))$, $\vartheta, \vartheta' \in At_\Sigma(X)$,
$\sigma = unify(\vartheta\tau, \vartheta')$,
τ ist eine Variablenumbenennung mit
$var(\varphi\tau \vee \psi\tau \vee \vartheta\tau) \parallel var(\vartheta' \vee \varphi' \vee \psi')$

Jede zur Gentzenformel des Sukzedenten der Schnittregel äquivalente Gentzenformel heißt **Resolvente** der beiden Gentzenformeln des Antezedenten. Letzterer enthält zwei unifizierbare Atome ϑ und ϑ', die wegen der Kommutativität von $\vee$ bzw. $\wedge$ nicht notwendig am Ende bzw. Anfang stehen.

Eine *PSK*-**Widerlegung** einer Menge Φ von Gentzenformeln ist eine *PSK*-Ableitung von $\Phi \vdash \bot$.

Beispiele

Schnittwiderlegung der Gentzenformelmenge $\Phi = \{\varphi_1, \varphi_2, \varphi_3\}$ mit

$$\begin{aligned} \varphi_1 &= p(y) \wedge p(f(a)) \Rightarrow q(x), \\ \varphi_2 &= \top \Rightarrow p(x), \\ \varphi_3 &= p(g(b,y)) \wedge q(b) \Rightarrow \bot. \end{aligned}$$

$\Phi \vdash$		$p(y) \wedge p(f(a)) \Rightarrow q(x)$	Einführung von φ_1
$\Phi \vdash$	(φ_4)	$p(f(a)) \Rightarrow q(x)$	Faktorregelanwendung auf φ_1 wegen $unify(p(y), p(f(a))) = \{f(a)/y\}$
$\Phi \vdash$		$\top \Rightarrow p(x)$	Einführung von φ_2
$\Phi \vdash$		$\top \Rightarrow p(x')$	Variablenumbenennung $\tau = \{x'/x\}$
$\Phi \vdash$	(φ_5)	$\top \Rightarrow q(x)$ $unify(p(x'), p(f(a))) = \{f(a)/x'\}$	Resolvente von $\varphi_2\tau$ und φ_4 wegen
$\Phi \vdash$		$p(g(b,y)) \wedge q(b) \Rightarrow \bot$	Einführung von φ_3
$\Phi \vdash$	(φ_6)	$p(g(b,y)) \Rightarrow \bot$	Resolvente von φ_5 und φ_3 wegen $unify(q(x), q(b)) = \{b/x\}$
$\Phi \vdash \top \Rightarrow \bot$		Resolvente von φ_2 und φ_6 wegen $unify(p(x), p(g(b,y))) = \{g(b,y)/x\}$	

Schnittwiderlegung der Gentzenformelmenge $\Phi = \{\varphi_1, \varphi_2, \varphi_3\}$ mit

$$\begin{aligned} \varphi_1 &= \top \Rightarrow p(f(y)) \vee q(y), \\ \varphi_2 &= p(x) \Rightarrow \bot, \\ \varphi_3 &= q(x) \wedge q(f(b)) \Rightarrow \bot. \end{aligned}$$

$\Phi \vdash$		$\top \Rightarrow p(f(y)) \vee q(y)$	Einführung von φ_1
$\Phi \vdash$		$p(x) \Rightarrow \bot$	Einführung von φ_2
$\Phi \vdash$	(φ_4)	$\top \Rightarrow q(y)$	Resolvente von φ_1 und φ_2 wegen $unify(p(f(y)), p(x)) = \{f(y)/x\}$
$\Phi \vdash$	(φ_5)	$q(x) \wedge q(f(b)) \Rightarrow \bot$	Einführung von φ_3
$\Phi \vdash$	(φ_6)	$q(f(b)) \Rightarrow \bot$	Faktorregelanwendung auf φ_5 wegen $unify(q(x), q(f(b))) = \{f(b)/x\}$
$\Phi \vdash$		$\top \Rightarrow \bot$	Resolvente von φ_4 und φ_6 wegen $unify(q(y), q(f(b))) = \{f(b)/y\}$

Schnittwiderlegung der Gentzenformelmenge $\Phi = \{\varphi_1, \varphi_2\}$ mit

$$\begin{aligned} \varphi_1 &= \top \Rightarrow p(f(x), y) \vee p(y, f(x)), \\ \varphi_2 &= p(f(c), x) \Rightarrow \bot. \end{aligned}$$

$\Phi \vdash$		$\top \Rightarrow p(f(x), y) \vee p(y, f(x))$	Einführung von φ_1
$\Phi \vdash$		$\top \Rightarrow p(f(x), f(x))$	Faktorregelanwendung auf φ_1 wegen $unify(p(f(x), y), p(y, f(x)))$ $= unify'((f(x), y), (y, f(x)))$ $= unify(y\sigma, f(x)\sigma) \odot \sigma$ $= unify(f(x), f(x)) \odot \sigma$ $= inc_X \odot \sigma$ $= \sigma$, wobei $\sigma = unify(f(x), y) = \{f(x)/y\}$
$\Phi \vdash$	(φ_3)	$\top \Rightarrow p(f(x'), f(x'))$	Variablenumbenennung $\tau = \{x'/x\}$
$\Phi \vdash$		$p(f(c), x) \Rightarrow \bot$	Einführung von φ_2
$\Phi \vdash$	$\top \Rightarrow \bot$	Resolvente von φ_3 und φ_2 wegen $unify(p(f(x'), f(x')), p(f(c), x))$ $= unify'((f(x'), f(x')), (f(c), x))$ $= unify(f(x')\sigma, x\sigma) \odot \sigma$ $= unify(f(c), x) \odot \sigma$ $= \{c/x', f(c)/x\}$,	wobei $\sigma = unify(f(x'), f(c)) = \{c/x'\}$

Aufgabe Zeigen Sie, dass die Widerlegung der Gentzenformelmenge

$$\{\top \Rightarrow p(x) \vee p(x'),\ p(y) \wedge p(y') \Rightarrow \bot\}$$

eine Summand- oder Faktorregelanwendung benötigt. ❑

Satz 8.8 (Korrektheit prädikatenlogischer Schnittableitungen)

Sei Φ eine endliche Menge von Gentzenformeln über $At_\Sigma(X)$ und ψ eine Gentzenformel.

(i) Aus $\Phi \vdash_{PSK} \psi$ folgt $\Phi \models \psi$.

(ii) Aus $\Phi \vdash_{PSK} \bot$ folgt, dass Φ unerfüllbar ist.

Beweis von (i). Es genügt zu zeigen, dass aus dem Antezendenten der Summand-, Faktor- und Schnittregel der jeweilige Sukzedent folgt. Daraus erhält man (i) durch Induktion über die Anzahl der Summand- bzw. Faktorregelanwendungen und Resolventen einer *PSK*-Ableitung von $\Phi \vdash \psi$.

Fall 1: $\frac{\varphi_1}{\varphi_2}$ ist die Summandregel.

Dann gilt $\varphi_1 = (\varphi \Rightarrow \psi \vee \vartheta \vee \vartheta')$ und $\varphi_2 = (\varphi\sigma \Rightarrow \psi\sigma \vee \vartheta\sigma)$ für Formeln bzw. Substitutionen $\varphi, \vartheta, \vartheta', \psi, \sigma$, die die Anwendbarkeitsbedingungen der Summandregel erfüllen.

Sei $A \models \varphi_1$ und $h \in A^X$ mit $A \models_h \varphi\sigma$. Nach Lemma 8.3 folgt $A \models_{h^* \circ \sigma} \varphi$, also

$$A \models_{h^* \circ \sigma} \psi \vee \vartheta \vee \vartheta'$$

wegen $A \models \varphi_1$. Also gilt $A \models_h \psi\sigma \vee \vartheta\sigma \vee \vartheta'\sigma$, wieder nach Lemma 8.3. Wegen $\vartheta\sigma = \vartheta'\sigma$ folgt $A \models_h \psi\sigma \vee \vartheta\sigma$. Da h beliebig gewählt wurde, schließen wir $A \models \varphi_2$.

Fall 2: $\frac{\varphi_1}{\varphi_2}$ ist die Faktorregel. Der Beweis verläuft analog zu Fall 1.

Fall 3: $\frac{\varphi_1 \quad \varphi_2}{\varphi_3}$ ist die Schnittregel. Dann gilt $\varphi_1 = (\varphi \Rightarrow \psi \vee \vartheta)$, $\varphi_2 = (\vartheta' \wedge \varphi' \Rightarrow \psi')$ und

$$\varphi_3 \quad = \quad \varphi\tau\sigma \wedge \varphi'\sigma \Rightarrow \psi\tau\sigma \vee \psi'\sigma$$

für Formeln bzw. Substitutionen $\varphi, \varphi', \vartheta, \vartheta', \psi, \psi', \tau, \sigma$, die die Anwendbarkeitsbedingungen der Schnittregel erfüllen.

Sei $A \models \varphi_1 \wedge \varphi_2$ und $h \in A^X$ mit $A \models_h \varphi\tau\sigma \wedge \varphi'\sigma$. Nach Lemma 8.3 folgt $A \models_{h^* \circ \tau\sigma} \varphi$ und $A \models_{h^* \circ \sigma} \varphi'$, also $A \models_{h^* \circ \tau\sigma} \psi \vee \vartheta$ wegen $A \models \varphi_1$. Nach Lemma 8.3 folgt $A \models_h \psi\tau\sigma \vee \vartheta\tau\sigma$.

Fall 3.1: $A \models_h \psi\tau\sigma$. Dann gilt $A \models_h \psi\tau\sigma \vee \psi'\sigma$. Da h beliebig gewählt wurde, schließen wir $A \models \varphi_3$.

Fall 3.2: $A \models_h \vartheta\tau\sigma$. Dann gilt $A \models_h \vartheta'\sigma$ wegen $\vartheta\tau\sigma = \vartheta'\sigma$. Nach Lemma 8.3 folgt $A \models_{h^* \circ \sigma} \vartheta'$.

Wegen $A \models_{h^* \circ \sigma} \varphi'$ und $A \models \varphi_2$ folgt $A \models_{h^* \circ \sigma} \psi'$, also $A \models_h \psi'\sigma$ wieder nach Lemma 8.3. Daraus folgt $A \models_h \psi\tau\sigma \vee \psi'\sigma$. Da h beliebig gewählt wurde, schließen wir $A \models \varphi_3$.

Beweis von (ii). Sei $\Phi \vdash_{PSK} \bot$. Aus (i) folgt $\bigwedge \Phi \models \bot$, also ist $\bigwedge \Phi$ unerfüllbar. ❑

Satz 8.9

Eine Menge Φ prädikatenlogischer Gentzenformeln ist genau dann erfüllbar, wenn ihre **Termexpansion**

$$Exp(\Phi) \quad = \quad \{\varphi\sigma \mid \varphi \in \Phi,\ \sigma \in T_\Sigma^X\}$$

(im aussagenlogischen Sinne) erfüllbar ist.

Beweis. Wegen Lemma 8.1 genügt es zu zeigen, dass φ genau dann ein Termmodell hat, wenn $Exp(\Phi)$ ein aussagenlogisches Modell hat.

Sei H ein Termmodell von Φ und $\sigma \in T_\Sigma^X$. Dann gibt es für alle Gentzenformeln $\varphi \Rightarrow \psi$ von Φ ein Atom $at_{\varphi,\psi}$ in φ oder ψ mit $\sigma \notin g_H(at_{\varphi,\psi})$ bzw. $\sigma \in g_H(at_{\varphi,\psi})$.

Wir bilden daraus folgende (aussagenlogische) Belegung $g : At_\Sigma(X) \to 2$:

Für alle Σ-Atome at über X, $g(at) =_{def} g_H(at)(\sigma)$. Dann gilt $g(at_{\varphi,\psi}) = 0$ bzw. $g(at_{\varphi,\psi}) = 1$, also $g(\varphi \Rightarrow \psi) = 1$ für alle Gentzenformeln $\varphi \Rightarrow \psi$ von Φ. Da σ beliebig gewählt wurde, ist g ist ein aussagenlogisches Modell von $Exp(\Phi)$.

Sei umgekehrt $g : At_\Sigma(X) \to 2$ ein aussagenlogisches Modell von $Exp(\Phi)$ und H die Termstruktur mit folgender Interpretation der Prädikate $p \in \Sigma$. Sei $n = arity(\sigma)$. Für alle $t \in T_\Sigma^n$,

$$t \in p^H \ \Leftrightarrow_{def} \ g(pt) = 1. \tag{7}$$

Sei $\sigma : X \to T_\Sigma(X)$. Nach Voraussetzung gibt es für alle Gentzenformeln $\varphi \Rightarrow \psi$ von Φ

- einen Faktor $pt_{\varphi,\psi}$ von ϑ mit $g(pt_{\varphi,\psi}\sigma) = 0$ (*)
- oder einen Summanden $pt_{\varphi,\psi}$ von ϑ' mit $g(pt_{\varphi,\psi}\sigma) = 1$. (**)

Im Fall (*) erhalten wir

$$\begin{aligned} &g(pt_{\varphi,\psi}\sigma) = 0 \overset{(7)}{\Leftrightarrow} t_{\varphi,\psi}\sigma \notin p^H \ \Leftrightarrow \ \sigma \notin g_H(pt_{\varphi,\psi}) \\ &\Rightarrow \ \sigma \notin g_H^*(\varphi) \ \Rightarrow \ \sigma \in g_H^*(\varphi \Rightarrow \psi). \end{aligned}$$

Im Fall (**) erhalten wir

$$\begin{aligned} &g(pt_{\varphi,\psi}\sigma) = 1 \overset{(7)}{\Leftrightarrow} t_{\varphi,\psi}\sigma \in p^H \ \Leftrightarrow \ \sigma \in g_H(pt_{\varphi,\psi}) \\ &\Rightarrow \ \sigma \in g_H^*(\psi) \ \Rightarrow \ \sigma \in g_H^*(\varphi \Rightarrow \psi). \end{aligned}$$

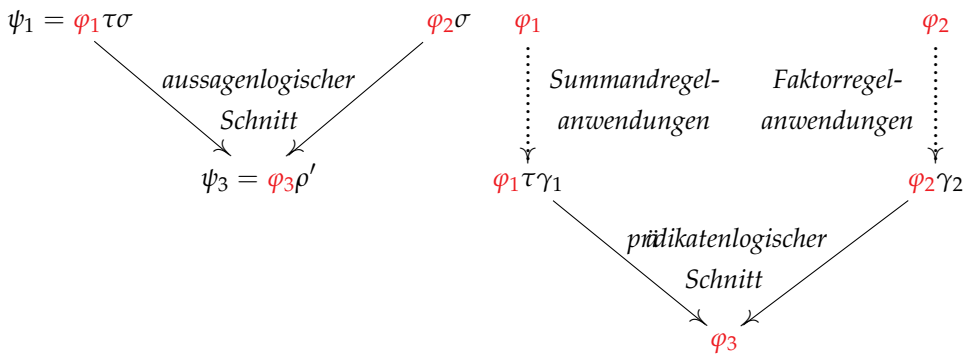
Also gilt $H \models_\sigma \varphi \Rightarrow \psi$ für alle Gentzenformeln $\varphi \Rightarrow \psi$ von Φ. Da σ beliebig gewählt wurde, ist H ein Modell von Φ. ❑

Lemma 8.10 (Lifting-Lemma)

Seien φ_1, φ_2 Gentzenformeln über $At_\Sigma(X)$, $\psi_1 \in Exp(\varphi_1)$, $\sigma : X \to T_\Sigma$, und ψ_3 eine aussagenlogische Resolvente von ψ_1 und $\varphi_2\sigma$.

Dann gibt es eine Gentzenformel φ_3, eine *PSK*-Ableitung von $\{\varphi_1, \varphi_2\} \vdash \varphi_3$ und $\rho' : X \to T_\Sigma$ mit $\varphi_3\rho' = \psi_3$ und $\sigma = \gamma'\rho'$, wobei γ' die Komposition der im Laufe der Ableitung gebildeten mgus ist.

Die aus der aussagenlogischen Resolvente gebildete *PSK*-Ableitung besteht aus möglicherweise leeren Folgen von Anwendungen der Summand -bzw. Faktorregel auf φ_1 bzw. φ_2 und einer anschließenden Anwendung der Schnittregel:



Beweis. Wegen $\psi_1 \in Exp(\varphi_1)$ gibt es eine Variablenumbenennung τ mit $var(\varphi_1\tau) \| var(\varphi_2)$ und $\psi_1 = \varphi_1\tau\sigma$. Da ψ_3 eine aussagenlogische Resolvente von ψ_1 und $\varphi_2\sigma$ ist, gibt es Formeln $\varphi, \varphi', \psi, \psi'$ und Atome

$$\vartheta_1, \ldots, \vartheta_m, \vartheta'_1, \ldots, \vartheta'_n$$

mit

$$\begin{array}{rclcrcl} \varphi_1 & = & \varphi \Rightarrow \psi \vee \vartheta_1 \vee \cdots \vee \vartheta_m, & \quad & \varphi_2 & = & \vartheta'_1 \wedge \cdots \wedge \vartheta'_n \wedge \varphi' \Rightarrow \psi', \\ \psi_3 & = & \varphi\tau\sigma \wedge \varphi'\sigma \Rightarrow \psi\tau\sigma \vee \psi'\sigma & & & & \end{array}$$

und $\vartheta_i\tau\sigma = \vartheta'_j\sigma$ für alle $1 \le i \le m$ und $1 \le j \le n$. Insbesondere werden sowohl die Atome $\vartheta_1\tau, \ldots, \vartheta_m\tau$ als auch die Atome $\vartheta'_1, \ldots, \vartheta'_n$ von σ unifiziert.

Also gibt es einen mgu $\gamma_1 \le \sigma$ der Atome $\vartheta_1\tau, \ldots, \vartheta_m\tau$ und einen mgu $\gamma_2 \le \sigma$ der Atome $\vartheta'_1, \ldots, \vartheta'_n$ mit $var(\varphi_1\tau\gamma_1) \| var(\varphi_2\gamma_2)$. Folglich ist $\tau^{-1}\gamma_1$ ein mgu von $\vartheta_1, \ldots, \vartheta_m$, und wir erhalten durch (*m-1*)-fache Anwendung der Summandregel auf φ_1 und (*n-1*)-fache Anwendung der Faktorregel auf φ_2 die Formeln

$$\varphi_1\tau\gamma_1 = \varphi\tau\gamma_1 \Rightarrow \psi\tau\gamma_1 \vee \vartheta_1\tau\gamma_1 \quad \text{bzw.} \quad \varphi_2\gamma_2 = \vartheta'_1\gamma_2 \wedge \varphi'\gamma_2 \Rightarrow \psi'\gamma_2.$$

Da $\varphi_1\tau$ und φ_2 keine gemeinsamen Variablen haben, lässt sich X als Summe der Mengen $var(\varphi_1\tau)$ und $X \setminus var(\varphi_2)$ darstellen.

Also gilt $\varphi_1\tau\gamma = \varphi_1\tau\gamma_1$, $\varphi_2\gamma = \varphi_2\gamma_2$ und $\gamma \le \sigma$ für die Summenextension $\gamma = [\gamma_1, \gamma_2]$.

Wegen $\gamma \le \sigma$ gibt es eine Substitution ρ mit $\gamma\rho = \sigma$. Wegen $\vartheta_i\tau\sigma = \vartheta'_j\sigma$ unifiziert ρ die Atome $\vartheta_1\tau\gamma$ und $\vartheta'_1\gamma$. Also gibt es einen mgu ξ dieser Atome und eine Substitution ρ' mit $\xi\rho' = \rho$. Da $var(\varphi_1\tau\gamma) = var(\vartheta_1\tau\gamma_1)$ und $var(\vartheta'_1\gamma) = var(\vartheta'_1\gamma_2)$ disjunkt sind, benötigen wir keine weitere Variablenumbenennung, um die Schnittregel auf $\varphi_1\tau\gamma$ und $\varphi_2\gamma$ anzuwenden, was zur Resolvente

$$\varphi_3 = \varphi\tau\gamma\xi \wedge \varphi'\gamma\xi \Rightarrow \psi\tau\gamma\xi \vee \psi'\gamma\xi$$

führt.

Also gibt es eine *PSK*-Ableitung von $\{\varphi_1, \varphi_2\} \vdash \varphi_3$.

Aus $\sigma = \gamma\rho = \gamma\xi\rho'$ folgt

$$\psi_3 = \varphi\tau\sigma \wedge \varphi'\sigma \Rightarrow \psi\tau\sigma \vee \psi'\sigma = \varphi\tau\gamma\xi\rho' \wedge \varphi'\gamma\xi\rho' \Rightarrow \psi\tau\gamma\xi\rho' \vee \psi'\gamma\xi\rho' = \varphi_3\rho'.$$

Demnach gilt die Behauptung des Lemmas für $\gamma' = \gamma\xi$. ❑

Das folgende Theorem überträgt Satz 6.4 auf *unendliche* Mengen aussagenlogischer Gentzenformeln. Es wird in Satz 8.12 auf Termexpansionen angewendet.

Satz 8.11 (Vollständigkeit von *ASK*-Widerlegungen von Gentzenformelmengen)

Sei Φ eine unerfüllbare Menge von Gentzenformeln über einer *abzählbaren* Menge von Atomen. Dann gilt $\Phi \vdash_{ASK} \bot$.

Beweis.

Wir folgen dem Beweis von [14], Satz 1.49, und zeigen die Behauptung durch Kontraposition. Sei $\Phi \nvdash_{ASK} \bot$ und $x_1, x_2, x_3, \ldots$ eine Abzählung der Atome von Φ.

Der **konsistente Abschluss** von Φ ist die folgendermaßen definierte Menge:

$$\begin{aligned} closure(\Phi) &= \bigcup_{n\in\mathbb{N}} \Phi_n, \\ \Phi_0 &= \Phi, \\ \Phi_{n+1} &= \begin{cases} \Phi_n \cup \{\neg x_n\} & \text{falls } \Phi_n \cup \{x_n\} \vdash_{ASK} \bot, \\ \Phi_n \cup \{x_n\} & \text{falls } \Phi_n \cup \{x_n\} \nvdash_{ASK} \bot. \end{cases} \end{aligned}$$

Durch Induktion über n zeigen wir zunächst, dass $\Phi_n \nvdash_{ASK} \bot$ für alle $n \in \mathbb{N}$ gilt:

Nach Voraussetzung gilt $\Phi_0 = \Phi \nvdash_{ASK} \bot$.

Sei $\bot$ nicht aus Φ_n ableitbar (Induktionsvoraussetzung).

Fall 1: $\Phi_n \cup \{x_n\} \nvdash_{ASK} \bot$. Dann gilt $\Phi_{n+1} = \Phi_n \cup \{x_n\} \nvdash_{ASK} \bot$.

Fall 2: $\Phi_n \cup \{x_n\} \vdash_{ASK} \bot$. Dann ist $\Phi_{n+1} = \Phi_n \cup \{\neg x_n\}$. Wäre $\Phi_{n+1} \vdash_{ASK} \bot$, dann müsste $\bot$ wegen $\Phi_n \nvdash_{ASK} \bot$ die Resolvente des Schnittes von $\neg x_n$ mit einer Gentzenformel von Φ_n sein. Die kann aber nur mit x_n übereinstimmen, d.h. x_n müsste zu Φ_n gehören. Also würde gelten:

$$\Phi_n \cup \{x_n\} \;=\; \Phi_n \;\nvdash_{ASK}\; \bot. \;\lightning$$

Also gilt $\Phi_n \nvdash_{ASK} \bot$ für alle $n \in \mathbb{N}$.

Sei $(closure(\Phi) \vdash \varphi_1, \ldots, closure(\Phi) \vdash \varphi_n)$ eine *ASK*-Ableitung. Dann gibt es eine endliche Teilmenge Ψ von $closure(\Phi)$ derart, dass auch $(\Psi \vdash \varphi_1, \ldots, \Psi \vdash \varphi_n)$ eine *ASK*-Ableitung ist. Also existiert $n \in \mathbb{N}$ mit $\Psi \subseteq \Phi_n$, so dass auch $(\Phi_n \vdash \varphi_1, \ldots, \Phi_n \vdash \varphi_n)$ eine *ASK*-Ableitung ist.

Insbesondere gäbe es im Fall $closure(\Phi) \vdash_{ASK} \bot$ $n \in \mathbb{N}$ mit $\Phi_n \vdash_{ASK} \bot$. $\lightning$

Also gilt $closure(\Phi) \nvdash_{ASK} \bot$.

Nun zeigen wir durch Kontraposition, dass die Belegung $g : At \to 2$ mit

$$g(x) = 1 \;\Leftrightarrow\; x \in closure(\Phi)$$

ein (aussagenlogisches) Modell von Φ ist.

Angenommen, es gibt eine Gentzenformel $\varphi = x_1 \wedge \cdots \wedge x_m \Rightarrow y_1 \vee \cdots \vee y_n$ von Φ, die g nicht erfüllt. Dann wäre für alle $1 \le i \le m$ und $1 \le j \le n$ $g(x_i) = 1$ und $g(y_j) = 0$. Nach Definition von g würde x_i zu $closure(\Phi)$ gehören, y_j jedoch nicht. Aus Letzterem würde folgen, dass $\neg y_j$ zu $closure(\Phi)$ gehört. Damit wäre

$$\{x_i \mid 1 \le i \le m\} \cup \{\varphi\} \cup \{\neg y_j \mid 1 \le j \le n\}$$

eine Teilmenge von $closure(\Phi)$, aus der sich durch $m + n$ Anwendungen der (aussagenlogischen) Schnittregel $\bot$ ableiten ließe. Also würde $closure(\Phi) \vdash_{ASK} \bot$ gelten. $\lightning$

Folglich erfüllt g alle Gentzenformeln von Φ, ist also ein Modell von Φ.

Ist Φ unerfüllbar, dann gilt demnach $\Phi \vdash_{ASK} \bot$. ❑

Um Satz 8.11 im folgenden Theorem anwenden zu können, setzen wir ab jetzt voraus, dass Σ, X und damit auch $At_\Sigma(X)$ abzählbar sind.

Satz 8.12 (Vollständigkeit prädikatenlogischer Schnittwiderlegungen)

Sei Φ eine endliche Menge von Gentzenformeln über $At_\Sigma(X)$.

(i) Aus $Exp(\Phi) \vdash_{ASK} \psi$ folgt $\Phi \vdash_{PSK} \varphi$ für eine Gentzenformel φ mit $\psi \in Exp(\varphi)$.

(ii) Ist Φ unerfüllbar, dann gilt $\Phi \vdash_{PSK} \bot$.

Beweis von (i) *durch Induktion über eine – bzgl. der Anzahl n ihrer Resolventen – minimale ASK-Ableitung von* $Exp(\Phi) \vdash \psi$.

Fall 1: $n = 0$. Dann gibt es $\varphi \in \Phi$ mit $\psi \in Exp(\varphi)$. Also ist $(\Phi \vdash \varphi)$ eine *PSK*-Ableitung von $\Phi \vdash \varphi$.

Fall 2: $n > 0$. Dann ist ψ eine Resolvente zweier Gentzenformeln ψ_1 und ψ_2 und es gibt *ASK*-Ableitungen von $Exp(\Phi) \vdash \psi_1$ bzw. $Exp(\Phi) \vdash \psi_1$ mit weniger als n Resolventen.

Nach Induktionsvoraussetzung gibt es Gentzenformeln φ_1, φ_2 mit $\Phi \vdash_{PSK} \varphi_1$, $\psi_1 \in Exp(\varphi_1)$, $\Phi \vdash_{PSK} \varphi_2$ und $\psi_2 \in Exp(\varphi_2)$.

Also liefert Lemma 8.10 eine Gentzenformel φ mit $\{\varphi_1, \varphi_2\} \vdash_{PSK} \varphi$ und $\psi \in Exp(\varphi)$.

Zusammengenommen liefern die drei *PSK*-Ableitungen die gewünschte *PSK*-Ableitung von $\Phi \vdash \varphi$.

Beweis von (ii). Nach Voraussetzung und Satz 8.9 ist die Termexpansion von Φ unerfüllbar. Da sie eine Menge variablenfreier Gentzenformeln über der abzählbaren (!) Menge $At_\Sigma(X)$ ist, folgt $Exp(\Phi) \vdash_{ASK} \bot$ aus Satz 8.11 und damit $\Phi \vdash_{PSK} \bot$ aus (i). ❑

8.3 Prädikatenlogik mit Gleichheit

Eine Σ-Struktur A heißt **Σ-Gleichheitsstruktur**, wenn das zweistellige **Gleichheitsprädikat** $\equiv$ durch die Diagonale Δ_A und das zweistellige **Ungleichheitsprädikat** $\not\equiv$ durch ihr Komplement $A^2 \setminus \Delta_A$ interpretiert werden.

Hat A die Trägermenge $T_\Sigma/\equiv_E$ für eine Menge E von Gleichungen (siehe Kapitel 5) und sind die Operationen von Σ so auf $T_\Sigma/\equiv_E$ interpretiert wie in Kapitel 5, dann heißt A **Quotienten-Termstruktur**.

Da $T_\Sigma(X)$ hier nur als Algebra, aber nicht als (Term-)Struktur eingeht, ist eine Quotienten-Termstruktur nicht notwendig eine Quotientenstruktur in obigem Sinne.

Ein Modell einer prädikatenlogischen Formel φ heißt **Gleichheitsmodell von** φ, wenn es eine Gleichheitsstruktur ist.

Seien $\varphi, \psi \in T_{PL}(At_\Sigma(X))$.

Wir schreiben $\varphi \models_{EQ} \psi$, wenn jedes Gleichheitsmodell von φ auch ψ erfüllt.

Seien $x, y, x_1, \ldots, x_n, y_1, \ldots, y_n \in X$, f eine Operation und p ein Prädikat von Σ mit $arity(f) = arity(p) = n$.

Die Menge $cong_\Sigma$ der **Σ-Kongruenzaxiome** besteht aus folgenden Gentzenformeln:

$$
\begin{array}{ll}
x \equiv x & \text{(Reflexivität)} \\
x \equiv y \;\Rightarrow\; y \equiv x & \text{(Symmetrie)} \\
x_1 \equiv y_1 \wedge \cdots \wedge x_n \equiv y_n \;\Rightarrow\; f(x_1, \ldots, x_n) \equiv f(y_1, \ldots, y_n) & \text{(Verträglichkeit mit } f\text{)} \\
p(x_1, \ldots, x_n) \wedge x_1 \equiv y_1 \wedge \cdots \wedge x_n \equiv y_n \;\Rightarrow\; p(y_1, \ldots, y_n) & \text{(Verträglichkeit mit } p\text{)}
\end{array}
$$

Die Menge $disg_\Sigma$ der **Σ-Disgruenzaxiome** besteht aus folgenden Gentzenformeln:

$$\begin{array}{ll}
\neg\, x \not\equiv x & \text{(Reflexivität)} \\
x \not\equiv y \;\Rightarrow\; y \not\equiv x & \text{(Symmetrie)} \\
f(x_1,\ldots,x_n) \not\equiv f(y_1,\ldots,y_n) \;\Rightarrow\; x_1 \not\equiv y_1 \vee \cdots \vee x_n \not\equiv y_n & \text{(Verträglichkeit mit } f\text{)} \\
p(x_1,\ldots,x_n) \;\Rightarrow\; x_1 \not\equiv y_1 \vee \cdots \vee x_n \not\equiv y_n \vee p(y_1,\ldots,y_n) & \text{(Verträglichkeit mit } p\text{)}
\end{array}$$

Aufgabe Zeigen Sie, dass Reflexivität und Verträglichkeit mit Prädikaten die Transitivität von $\equiv$ bzw. $\not\equiv$ implizieren, die als Gentzenformel folgendermaßen lautet:

$$x \equiv y \wedge y \equiv z \;\Rightarrow\; x \equiv z \quad \text{bzw.} \quad x \not\equiv z \;\Rightarrow\; x \not\equiv y \vee y \not\equiv z. \qquad \square$$

Aufgabe Zeigen Sie, dass jede Σ-Gleichheitsstruktur ein Modell von $cong_\Sigma$ bzw. $disg_\Sigma$ ist. $\square$

Die in Kapitel 5 behandelte Gleichungslogik ist der Spezialfall der Prädikatenlogik, in dem die Prädikate von Σ auf $\equiv$, Formeln auf Σ-Gleichungen und Σ-Strukturen auf Gleichheitsstrukturen beschränkt sind.

Aufgabe Sei A ein Modell von $disg_\Sigma$. Zeigen Sie, dass $\sim^A =_{def} A^2 \backslash \not\equiv^A$ eine Σ-Kongruenz ist. $\square$

Aus obiger Interpretation von Prädikaten in Quotientenstrukturen folgt für alle $a, b \in A$:

$$\begin{array}{rcl}
nat_{\equiv^A}(a) \equiv^{A/\equiv^A} nat_{\equiv^A}(b) & \Leftrightarrow & a \equiv^A b \;\Leftrightarrow\; nat_{\equiv^A}(a) = nat_{\equiv^A}(b), \\
nat_{\sim^A}(a) \equiv^{A/\sim^A} nat_{\sim^A}(b) & \Leftrightarrow & a \sim^A b \;\Leftrightarrow\; nat_{\sim^A}(a) = nat_{\sim^A}(b).
\end{array}$$

Demnach sind $A/\equiv^A$ und $A/\sim^A$ Gleichheitsstrukturen.

Satz 8.13

(i) Für jede Σ-Kongruenz $\sim$ auf A, $nat = nat_\sim$ und alle $h \in A^X$ gilt folgende Äquivalenz:

$$A/\sim \models_{nat\circ h} \varphi \quad\Leftrightarrow\quad A \models_h \varphi.$$

(ii) Sei $\varphi \in T_{PL}(At_\Sigma(X))$ und A eine Σ-Struktur.

$$\begin{array}{rcl}
A/\equiv^A \;\models\; \varphi & \Leftrightarrow & A \;\models\; \varphi \wedge \bigwedge cong_\Sigma, \\
A/\sim^A \;\models\; \varphi & \Leftrightarrow & A \;\models\; \varphi \wedge \bigwedge disg_\Sigma.
\end{array}$$

(iii) Sei φ eine implikative Normalform über $At_\Sigma(X)$.

$$\begin{array}{rcl}
\varphi \text{ hat kein Gleichheitsmodell} & \Leftrightarrow & \varphi \wedge \bigwedge cong_\Sigma \text{ ist unerfüllbar}, \\
\varphi \text{ hat kein Gleichheitsmodell} & \Leftrightarrow & \varphi \wedge \bigwedge disg_\Sigma \text{ ist unerfüllbar}.
\end{array}$$

Beweis von (i) durch strukturelle Induktion über $T_{PL}(At_\Sigma(X))$.

Für alle $pt \in At_\Sigma(X)$,

$$\begin{array}{l}
A/\sim \models_{nat\circ h} pt \;\Leftrightarrow\; nat \circ h \in g^*_{A/\sim}(pt) = g_{A/\sim}(pt) \;\Leftrightarrow\; nat(h^*(t)) \\
\overset{\text{Lemma 5.3 (i)}}{=} (nat \circ h)^*(t) \in p^{A/\sim} \Leftrightarrow h^*(t) \in p^A \;\Leftrightarrow\; h \in g_A(pt) = g^*_A(pt) \;\Leftrightarrow\; A \models_h pt.
\end{array}$$

Für alle $\varphi, \psi \in T_{PL}(At_\Sigma(X))$,

$$\begin{array}{l}
A/\sim \models_{nat\circ h} \neg\varphi \;\Leftrightarrow\; nat \circ h \in g^*_{A/\sim}(\neg\varphi) = \neg^{Pow(A/\sim)}(g^*_{A/\sim}(\varphi)) = (A/\sim)^X \setminus g^*_{A/\sim}(\varphi) \\
\overset{\text{ind. hyp.}}{\Leftrightarrow} h \in A^X \setminus g^*_A(\varphi) = \neg^{Pow(A)}(g^*_A(\varphi)) = g^*_A(\neg\varphi) \;\Leftrightarrow\; A \models_h \varphi,
\end{array}$$

$$
\begin{aligned}
& A/\sim \models_{nat\circ h} \varphi \wedge \psi \\
& \Leftrightarrow\ nat \circ h \in g^*_{A/\sim}(\varphi \wedge \psi) = g^*_{A/\sim}(\varphi) \wedge^{Pow(A/\sim)} g^*_{A/\sim}(\psi) = g^*_{A/\sim}(\varphi) \cap g^*_{A/\sim}(\psi) \\
& \overset{ind.\ hyp.}{\Leftrightarrow}\ h \in g^*_A(\varphi) \cap g^*_A(\psi) = g^*_A(\varphi) \wedge^{Pow(A)} g^*_A(\psi) = g^*_A(\varphi \wedge \psi)\ \Leftrightarrow\ A \models_h \varphi \wedge \psi
\end{aligned}
$$

und analog $A/\sim \models_{nat\circ h} \varphi \vee \psi \ \Leftrightarrow\ A \models_h \varphi \vee \psi$.

Für alle $x \in X$ und $\varphi \in T_{PL}(At_\Sigma(X))$,

$$
\begin{aligned}
& A/\sim \models_{nat\circ h} \forall x\varphi \ \Leftrightarrow\ nat \circ h \in g^*_{A/\sim}(\forall x\varphi) = (\forall x)^{Pow(A/\sim)}(g^*_{A/\sim}(\varphi)) \\
& \Leftrightarrow\ \forall\, [a] \in A/\sim : nat \circ h[a/x] = (nat \circ h)[[a]/x] \in g^*_{A/\sim}(\varphi) \\
& \overset{ind.\ hyp.}{\Leftrightarrow}\ \forall\, a \in A : h[a/x] \in g^*_A(\varphi) = (\forall x)^{Pow(A)}(g^*_A(\varphi)) = g^*_A(\forall x\varphi)\ \Leftrightarrow\ A \models_h \forall x\varphi
\end{aligned}
$$

und analog $A/\sim \models_{nat\circ h} \exists x\varphi \ \Leftrightarrow\ A \models_h \exists x\varphi$.

Sei $\sim\ \in \{\equiv^A, \sim^A\}$.

Beweis von (ii). Da jede Σ-Gleichheitsstruktur die Σ-Kongruenz- und -Disgruenzaxiome erfüllt und es zu jeder Belegung $h' : X \to A/\sim$ eine Belegung $h : X \to A$ mit $nat \circ h = h'$ gibt, folgt (ii) aus (i).

Beweis von (iii). φ habe kein Gleichheitsmodell. Dann gilt für alle Σ-Strukturen A, $A/\sim \not\models \varphi$, also $A \not\models \varphi \wedge \bigwedge cong_\Sigma$ bzw. $A \not\models \varphi \wedge \bigwedge disg_\Sigma$ wegen (ii), d.h. $\varphi \wedge \bigwedge cong_\Sigma$ bzw. $\varphi \wedge \bigwedge disg_\Sigma$ ist unerfüllbar.

Sei umgekehrt $\varphi \wedge \bigwedge cong_\Sigma$ bzw. $\varphi \wedge \bigwedge disg_\Sigma$ erfüllbar. Dann gibt es ein Modell A von $\varphi \wedge \bigwedge cong_\Sigma$ bzw. $\varphi \wedge \bigwedge disg_\Sigma$ und damit wegen (ii) das Gleichheitsmodell $A/\sim$ von φ. ❑

Wir fassen zusammen:

Sei Φ eine endliche Menge von Gentzenformeln über $At_\Sigma(X)$.

Aus den Sätzen 8.8 (ii) und 8.12 (ii) folgt:

$$
\boxed{\Phi \text{ ist unerfüllbar} \ \Leftrightarrow\ \Phi \vdash_{PSK} \bot.} \tag{4}
$$

Mit Hilfe von Satz 8.13 (iii) lässt sich (4) auf Gleichheitsmodelle übertragen:

$$
\boxed{\begin{array}{ll}
 & \Phi \text{ hat kein Gleichheitsmodell} \\
\overset{8.13\ (iii)}{\Leftrightarrow} & \Phi \cup cong_\Sigma \text{ ist unerfüllbar} \\
\overset{(4)}{\Leftrightarrow} & \Phi \cup cong_\Sigma \ \vdash_{PSK}\ \bot.
\end{array}} \tag{5}
$$

$$
\boxed{\begin{array}{ll}
 & \Phi \text{ hat kein Gleichheitsmodell} \\
\overset{8.13\ (iii)}{\Leftrightarrow} & \Phi \cup disg_\Sigma \text{ ist unerfüllbar} \\
\overset{(4)}{\Leftrightarrow} & \Phi \cup disg_\Sigma \ \vdash_{PSK}\ \bot.
\end{array}} \tag{6}
$$

Für Gleichheitsstrukturen, deren Gleichheit über Gleichungen definiert ist (siehe Kapitel 5), lässt sich Lemma 8.1 auf Quotienten-Termstrukturen übertragen:

Lemma 8.14 (Vollständigkeit von Quotienten-Termstrukturen)

Sei Σ eine Signatur mit Prädikatenmenge P, E eine Menge von Gleichungen, die die Σ-Kongruenzaxiome enthält, φ eine quantorenfreie Σ-Formel, $A \in Struct_\Sigma$ ein Gleichheitsmodell von E und φ und $H(A)$ die folgende Erweiterung der $(\Sigma \setminus P)$-Algebra $Q = T_\Sigma/\equiv_E$ zur Σ-Gleichheitsstruktur.

Sei *nat* die natürliche Abbildung von T_Σ nach Q. Für alle $p : n \in P$,

$$p^{H(A)} =_{def} \{nat(t) \mid t \in T_\Sigma^n, fold^A(t) \in p^A\}.$$

$H(A)$ ist ein Modell von φ.

Beweis. Durch strukturelle Induktion über φ zeigen wir

$$g^*_{H(A)}(\varphi) = \{nat \circ \sigma \mid \sigma \in T_\Sigma^X, fold^A \circ \sigma \in g^*_A(\varphi)\}. \tag{7}$$

Für alle $pt \in At_\Sigma(X)$,

$$\begin{aligned}
&g^*_{H(A)}(pt) = g_{H(A)}(pt) \overset{Def.\ g_{H(A)}}{=} \{nat \circ \sigma \mid \sigma \in T_\Sigma^X, (nat \circ \sigma)^*(t) \in p^{H(A)}\} \\
&\overset{Lemma\ 5.3\ (i)}{=} \{nat \circ \sigma \mid \sigma \in T_\Sigma^X, nat(t\sigma) \in p^{H(A)}\} \\
&\overset{Def.\ p^{H(A)}}{=} \{nat \circ \sigma \mid \sigma \in T_\Sigma^X, fold^A(t\sigma) \in p^A\} \\
&\overset{Lemma\ 5.3\ (i)}{=} \{nat \circ \sigma \mid \sigma \in T_\Sigma^X, (fold^A \circ \sigma)^*(t) \in p^A\} \\
&= \{nat \circ \sigma \mid \sigma \in T_\Sigma^X, fold^A \circ \sigma \in g_A(pt)\} = \{nat \circ \sigma \mid \sigma \in T_\Sigma^X, fold^A \circ \sigma \in g^*_A(pt)\}.
\end{aligned}$$

Für alle $\varphi, \psi \in T_{AL}(At_\Sigma(X))$,

$$\begin{aligned}
&g^*_{H(A)}(\neg\varphi) = Q^X \setminus g^*_{H(A)}(\varphi) \overset{ind.\ hyp.}{=} Q^X \setminus \{nat \circ \sigma \mid \sigma \in T_\Sigma^X, fold^A \circ \sigma \in g^*_A(\varphi)\} \\
&= \{nat \circ \sigma \mid \sigma \in T_\Sigma^X, fold^A \circ \sigma \in A^X \setminus g^*_A(\varphi)\} \\
&= \{nat \circ \sigma \mid \sigma \in T_\Sigma^X, fold^A \circ \sigma \in g^*_A(\neg\varphi)\}, \\
&g^*_{H(A)}(\varphi \wedge \psi) = g^*_{H(A)}(\varphi) \cap g^*_{H(A)}(\varphi) \\
&\overset{ind.\ hyp.}{=} \{nat \circ \sigma \mid \sigma \in T_\Sigma^X, fold^A \circ \sigma \in g^*_A(\varphi) \cap g^*_A(\psi)\} \\
&= \{nat \circ \sigma \mid \sigma \in T_\Sigma^X, fold^A \circ \sigma \in g^*_A(\varphi \wedge \psi)\}, \\
&g^*_{H(A)}(\varphi \vee \psi) = g^*_{H(A)}(\varphi) \cup g^*_{H(A)}(\varphi) \\
&\overset{ind.\ hyp.}{=} \{nat \circ \sigma \mid \sigma \in T_\Sigma^X, fold^A \circ \sigma \in g^*_A(\varphi) \cup g^*_A(\psi)\} \\
&= \{nat \circ \sigma \mid \sigma \in T_\Sigma^X, fold^A \circ \sigma \in g^*_A(\varphi \vee \psi)\}.
\end{aligned}$$

Damit endet der Beweis von (7).

Sei A ein Modell von φ und $\sigma \in T_\Sigma^X$. Dann gilt $g^*_A(\varphi) = A^X$, also insbesondere $h^* \circ \sigma \in g^*_A(\varphi)$ für alle $h \in A^X$. (7) impliziert $nat \circ \sigma \in g^*_{H(A)}(\varphi)$. Da alle Belegungen von X in Q die Form $nat \circ \sigma$ haben, gilt $g^*_{H(A)}(\varphi) = Q^X$. Also ist $H(A)$ ein Modell von φ. ❏

8.4 Übersetzung modallogischer in prädikatenlogische Formeln

Sei At eine Menge unstrukturierter Atome wie in Kapitel 6 und 7, $X = \{x_1, x_2, \dots\}$ eine abzählbare Menge von Variablen für Zustände und Σ eine Signatur, die das zweistellige Prädikat *trans* und jedes Atom von At als einstelliges Prädikat enthält.

Die im Folgenden induktiv definierte Funktion

$$ml2pl : T_{ML}(At) \times X \to T_{PL}(At_\Sigma(X))$$

übersetzt jede modallogische Formel über At in eine prädikatenlogische Σ-Formel über X_{state}: Für alle $p \in At$,

$\otimes \in \{\vee, \wedge, \Rightarrow\}$, $\varphi, \psi \in T_{ML}(At)$, $x \in X$ und $i \in \mathbb{N}$,

$$\begin{aligned}
ml2pl(p, x) &= p(x), \\
ml2pl(\bot, x) &= \bot, \\
ml2pl(\top, x) &= \top, \\
ml2pl(\neg\varphi, x) &= \neg ml2pl(\varphi, x), \\
ml2pl(\varphi \otimes \psi, x) &= ml2pl(\varphi, x) \otimes ml2pl(\psi, x), \\
ml2pl(\Box\varphi, x_i) &= \forall x_{i+1}(trans(x_i, x_{i+1}) \Rightarrow ml2pl(\varphi, x_{i+1})), \\
ml2pl(\Diamond\varphi, x_i) &= \exists x_{i+1}(trans(x_i, x_{i+1}) \wedge ml2pl(\varphi, x_{i+1})).
\end{aligned}$$

Aufgabe Zeigen Sie, dass für alle $\varphi \in T_{ML}(At)$ und $x \in X$ x die einzige freie Variable von $ml2pl(\varphi, x)$ ist. ❑

Im Gegensatz zum Übersetzer von Zustandsformeln in aussagenlogische Formeln (siehe Kapitel 7) ist *ml2pl* nicht von einer Kripke-Struktur

$$\mathcal{K} = \langle \delta, \beta \rangle : State \to (\mathcal{P}(State) \times \mathcal{P}(At))$$

abhängig. $\mathcal{K}$ bestimmt jedoch die *Interpretation* von *trans* und At in einer Σ-Struktur: Sei $\Sigma' = \Sigma \setminus \{trans\} \cup At$ und A eine Σ'-Struktur mit Trägermenge *State*. Dann bezeichnet $A_\mathcal{K}$ die Σ-Struktur mit

$$\begin{aligned}
trans^{A_\mathcal{K}} &= \chi^{-1}(\delta) = \{(s, s') \in State^2 \mid s' \in \delta(s)\}, \\
p^{A_\mathcal{K}} &= \{s \in State \mid p \in \beta(s)\}
\end{aligned}$$

für alle $p \in At$.

Satz 8.15 (Korrektheit von *ml2pl*)

Für alle Kripke-Strukturen $\mathcal{K}$ mit Zustandsmenge *State*, Σ'-Strukturen A mit Trägermenge *State*, $\varphi \in T_{ML}(At)$, $h \in State^X$ und $x \in X$ gilt:

$$\mathcal{K} \models_{h(x)} \varphi \quad \Leftrightarrow \quad A_\mathcal{K} \models_h ml2pl(\varphi, x). \tag{1}$$

Beweis. Nach Definition der modal- bzw. prädikatenlogischen Erfüllbarkeitsrelation ist (1) äquivalent zu:

$$h(x) \in g^*_\mathcal{K}(\varphi) \quad \Leftrightarrow \quad h \in g^*_{A_\mathcal{K}}(ml2pl(\varphi, x)). \tag{2}$$

Wir zeigen (2) durch Induktion über $T_{ML}(At)$.

Für alle $p \in At$ und $x \in X$,

$$\begin{aligned}
&h(x) \in g^*_\mathcal{K}(p) = g_\mathcal{K}(p) \Leftrightarrow p \in \beta(h(x)) \Leftrightarrow h^*(x) = h(x) \in p^{A_\mathcal{K}} \\
&\quad \Leftrightarrow h \in g^*_{A_\mathcal{K}}(ml2pl(p, x)) = g^*_{A_\mathcal{K}}(p(x)), \\
&h(x) \in g^*_\mathcal{K}(\bot) = \emptyset \Leftrightarrow h \in \emptyset = g^*_{A_\mathcal{K}}(\bot) = g^*_{A_\mathcal{K}}(ml2pl(\bot)) \\
&h(x) \in g^*_\mathcal{K}(\top) = State \Leftrightarrow h \in State^X = g^*_{A_\mathcal{K}}(\top) = g^*_{A_\mathcal{K}}(ml2pl(\top)).
\end{aligned}$$

Für alle $\varphi, \psi \in T_{ML}(At)$ und $x \in X$,

$$\begin{aligned}
&h(x) \in g^*_\mathcal{K}(\neg\varphi) = State \setminus g^*_\mathcal{K}(\varphi) \\
&\overset{ind.\ hyp.}{\Leftrightarrow} h \in State^X \setminus g^*_{A_\mathcal{K}}(ml2pl(\varphi, x)) = g^*_{A_\mathcal{K}}(\neg(ml2pl(\varphi, x))) \\
&\qquad = g^*_{A_\mathcal{K}}(ml2pl(\neg\varphi, x)),
\end{aligned}$$

$$
\begin{aligned}
&h(x) \in g^*_{\mathcal{K}}(\varphi \wedge \psi) = g^*_{\mathcal{K}}(\varphi) \cap g^*_{\mathcal{K}}(\psi)\\
&\overset{\text{ind. hyp.}}{\Leftrightarrow} \quad h \in g^*_{A_{\mathcal{K}}}(ml2pl(\varphi, x)) \cap g^*_{A_{\mathcal{K}}}(ml2pl(\psi, x))\\
&\qquad\qquad = g^*_{A_{\mathcal{K}}}(ml2pl(\varphi, x) \wedge ml2pl(\psi, x)) = g^*_{A_{\mathcal{K}}}(ml2pl(\varphi \wedge \psi, x)),\\
&h(x) \in g^*_{\mathcal{K}}(\varphi \vee \psi) = g^*_{\mathcal{K}}(\varphi) \cup g^*_{\mathcal{K}}(\psi)\\
&\overset{\text{ind. hyp.}}{\Leftrightarrow} \quad h \in g^*_{A_{\mathcal{K}}}(ml2pl(\varphi, x)) \cup g^*_{A_{\mathcal{K}}}(ml2pl(\psi, x))\\
&\qquad\qquad = g^*_{A_{\mathcal{K}}}(ml2pl(\varphi, x) \vee ml2pl(\psi, x)) = g^*_{A_{\mathcal{K}}}(ml2pl(\varphi \vee \psi, x)),\\
&h(x) \in g^*_{\mathcal{K}}(\varphi \Rightarrow \psi) = (State \setminus g^*_{\mathcal{K}}(\varphi)) \cup g^*_{\mathcal{K}}(\psi)\\
&\overset{\text{ind. hyp.}}{\Leftrightarrow} \quad h \in (State^X \setminus g^*_{A_{\mathcal{K}}}(ml2pl(\varphi, x))) \cup g^*_{A_{\mathcal{K}}}(ml2pl(\psi, x))\\
&\qquad\qquad = g^*_{A_{\mathcal{K}}}(ml2pl(\varphi, x) \Rightarrow ml2pl(\psi, x)) = g^*_{A_{\mathcal{K}}}(ml2pl(\varphi \Rightarrow \psi, x)).
\end{aligned}
$$

Für alle $\varphi \in T_{ML}(At)$ und $i \in \mathbb{N}$,

$$
\begin{aligned}
&h(x_i) \in g^*_{\mathcal{K}}(\Box\varphi) = \{s \in State \mid \delta(s) \subseteq g^*_{\mathcal{K}}(\varphi)\} \;\Leftrightarrow\; \delta(h(x_i)) \subseteq g^*_{\mathcal{K}}(\varphi)\\
&\Leftrightarrow\; \forall\, s \in State : s \notin \delta(h(x_i)) \text{ oder } h[s/x_{i+1}](x_{i+1}) = s \in g^*_{\mathcal{K}}(\varphi)\\
&\overset{\text{ind. hyp.}}{\Leftrightarrow}\; \forall\, s \in State : s \notin \delta(h(x_i)) \text{ oder } h[s/x_{i+1}] \in g^*_{A_{\mathcal{K}}}(ml2pl(\varphi, x_{i+1}))\\
&\Leftrightarrow\; \forall\, s \in State : (h[s/x_{i+1}](x_i), h[s/x_{i+1}](x_{i+1})) = (h(x_i), s) \notin trans^{A_{\mathcal{K}}}\\
&\qquad\qquad \text{oder } h[s/x_{i+1}] \in g^*_{A_{\mathcal{K}}}(ml2pl(\varphi, x_{i+1}))\\
&\Leftrightarrow\; \forall\, s \in State : h[s/x_{i+1}] \notin g^*_{A_{\mathcal{K}}}(trans(x_i, x_{i+1}))\\
&\qquad\qquad \text{oder } h[s/x_{i+1}] \in g^*_{A_{\mathcal{K}}}(ml2pl(\varphi, x_{i+1}))\\
&\Leftrightarrow\; \forall\, s \in State : h[s/x_{i+1}] \in g^*_{A_{\mathcal{K}}}(trans(x_i, x_{i+1}) \Rightarrow ml2pl(\varphi, x_{i+1}))\\
&\Leftrightarrow\; h \in g^*_{A_{\mathcal{K}}}(\forall x_{i+1}(trans(x_i, x_{i+1}) \Rightarrow ml2pl(\varphi, x_{i+1}))) = g^*_{A_{\mathcal{K}}}(ml2pl(\Box\varphi, x_i)),
\end{aligned}
$$

$$
\begin{aligned}
&h(x_i) \in g^*_{\mathcal{K}}(\Diamond\varphi) = \{s \in State \mid \delta(s) \cap g^*_{\mathcal{K}}(\varphi) \neq \emptyset\} \;\Leftrightarrow\; \delta(h(x_i)) \cap g^*_{\mathcal{K}}(\varphi) \neq \emptyset\\
&\Leftrightarrow\; \exists\, s \in \delta(h(x_i)) : h[s/x_{i+1}](x_{i+1}) = s \in g^*_{\mathcal{K}}(\varphi)\\
&\overset{\text{ind. hyp.}}{\Leftrightarrow}\; \exists\, s \in \delta(h(x_i)) : h[s/x_{i+1}] \in g^*_{A_{\mathcal{K}}}(ml2pl(\varphi, x_{i+1}))\\
&\Leftrightarrow\; \exists\, s \in State : (h[s/x_{i+1}](x_i), h[s/x_{i+1}](x_{i+1})) = (h(x_i), s) \in trans^{A_{\mathcal{K}}}\\
&\qquad\qquad \text{und } h[s/x_{i+1}] \in g^*_{A_{\mathcal{K}}}(ml2pl(\varphi, x_{i+1}))\\
&\Leftrightarrow\; \exists\, s \in State : h[s/x_{i+1}] \in g^*_{A_{\mathcal{K}}}(trans(x_i, x_{i+1}))\\
&\qquad\qquad \text{und } h[s/x_{i+1}] \in g^*_{A_{\mathcal{K}}}(ml2pl(\varphi, x_{i+1}))\\
&\Leftrightarrow\; \exists\, s \in State : h[s/x_{i+1}] \in g^*_{A_{\mathcal{K}}}(trans(x_i, x_{i+1}) \Rightarrow ml2pl(\varphi, x_{i+1}))\\
&\Leftrightarrow\; h \in g^*_{A_{\mathcal{K}}}(\exists x_{i+1}(trans(x_i, x_{i+1}) \Rightarrow ml2pl(\varphi, x_{i+1}))) = g^*_{A_{\mathcal{K}}}(ml2pl(\Diamond\varphi, x_i)).
\end{aligned}
$$

❑

Eine modallogische Formel φ heißt **bisimulationsinvariant**, wenn für alle Kripke-Strukturen $\mathcal{K}$ $\sim_{\mathcal{K}}$ zum Kern von $g^*_{\mathcal{K}}(\varphi)$ gehört. Demnach sind laut Satz 7.9 alle modallogischen Formeln bisimulationsinvariant.

Analog nennen wir eine prädikatenlogische Σ-Formel φ **bisimulationsinvariant**, wenn für alle Kripke-Strukturen $\mathcal{K}$ mit Zustandsmenge *State*, Σ'-Strukturen A $\sim_{\mathcal{K}}$ zum Kern von $g^*_{A_{\mathcal{K}}}(\varphi)$ gehört.

Satz 8.15 ([4], Theorem 4.18) Eine prädikatenlogische Formel ist genau dann bisimulationsinvariant, wenn sie zum Bild von *ml2pl* gehört. ❑

8.5 Highlights

Signatur der Prädikatenlogik: $PL = AL \cup \{Qx : 1 \to 1 \mid Q \in \{\forall, \exists\},\ x \in X\}$

Sei Σ eine Signatur der in prädikatenlogischen Formeln vorkommenden Operationen $f : n \to 1$ und **Prädikate** $p : n$.

Σ-Atome:

$At_\Sigma(X) = \{pt \mid p : n \in \Sigma,\ t \in T_\Sigma(X)^n\}$.

Semantik der Prädikatenlogik:

Σ-Struktur A: Interpretation der Operationen und Prädikate von Σ

Termstruktur A: Σ-Struktur mit der Trägermenge und den Operationen von $T_\Sigma(X)$ oder T_Σ

PL-Algebra $Pow(A)$ mit Trägermenge $\mathcal{P}(A^X)$: Für alle $S, S' \subseteq A^X$,

$$\begin{aligned}
\bot^{Pow(A)} &= \emptyset, \\
\top^{Pow(A)} &= A^X, \\
\neg^{Pow(A)}(S) &= A^X \setminus S, \\
\wedge^{Pow(A)}(S, S') &= S \cap S', \\
\vee^{Pow(A)}(S, S') &= S \cup S', \\
\Rightarrow^{Pow(A)}(S, S') &= (A^X \setminus S) \cup S', \\
(\forall x)^{Pow(A)}(S) &= \{h \in A^X \mid \forall\, a \in A : h[a/x] \in S\}, \\
(\exists x)^{Pow(A)}(S) &= \{h \in A^X \mid \exists\, a \in A : h[a/x] \in S\}.
\end{aligned}$$

A induziert folgende Belegung $g_A : At_\Sigma(X) \to \mathcal{P}(A^X)$ der Atome:

Für alle $pt \in At_\Sigma(X)$,

$$g_A(pt) =_{def} \{h \in A^X \mid h^*(t) \in p^A\}.$$

Sei $\varphi, \psi \in T_{PL}(At_\Sigma(X))$ und $h : X \to A$.

(A, h) **erfüllt** φ oder ist ein **Modell** von φ, geschrieben: $A \models_h \varphi$, wenn h zu $g_A^*(\varphi)$ gehört. h heißt dann auch **Lösung von** φ **in** A.

A **erfüllt** φ oder ist ein **Modell** von φ, geschrieben: $A \models \varphi$, wenn $g_A^*(\varphi) = A^X$ gilt.

Wir schreiben $\varphi \Leftrightarrow_h^A \psi$, falls (A, h) genau dann φ erfüllt, wenn (A, h) ψ erfüllt, und $\varphi \Leftrightarrow^A \psi$, falls A genau dann φ erfüllt, wenn A ψ erfüllt.

Termmodell von φ: Termstruktur, die φ erfüllt.

Lemma 8.1: Jede erfüllbare quantorenfreie Σ-Formel hat ein Termmodell.

Sei $\sigma : X \to T_\Sigma(X)$. Die Erweiterung $\sigma^* : T_{PL}(At_\Sigma(X)) \to T_{PL}(At_\Sigma(X))$ von σ auf prädikatenlogische Formeln verlangt eine spezielle Behandlung der Quantoren:

- Für alle $f : n \to 1 \in \Sigma$ und $t \in T_\Sigma(X)^n$, $\sigma^*(ft) = f\sigma^*(t)$.
- Für alle $p : n \in \Sigma$ und $t \in T_\Sigma(X)^n$, $\sigma^*(pt) = p\sigma^*(t)$.
- Für alle $f : n \to 1 \in AL$ und $\varphi \in T_{PL}(At_\Sigma(X))^n$, $\sigma^*(ft) = f\sigma^*(\varphi)$.
- Für alle $Q \in \{\exists, \forall\}$, $x \in X$ und $\varphi \in T_{PL}(At_\Sigma(X))$,

$$\sigma^*(Qx\varphi) = \begin{cases} Qx'\sigma[x'/x]^*(\varphi) & \text{falls } x \in V_{\varphi,\sigma,x} =_{def} \bigcup var(\sigma(free(\varphi) \setminus \{x\})), \\ Qx\sigma[x/x]^*(\varphi) & \text{sonst.} \end{cases}$$

Substitutionslemma (Lemma 8.3):

Für alle Σ-Strukturen A, $h : X \to A$, $\sigma : X \to T_\Sigma(X)$ und $\varphi \in T_{PL}(At_\Sigma(X))$ gilt

$$A \models_h \varphi\sigma \ \Leftrightarrow\ A \models_{h^* \circ \sigma} \varphi.$$

$Pow(A)$ erfüllt alle in der Bitalgebra gültigen AL-Gleichungen sowie die folgenden PL-Gleichungen:

$$\neg\exists x\varphi \equiv \forall x\neg\varphi \qquad \neg\forall x\varphi \equiv \exists x\neg\varphi$$
$$\exists x(\varphi \vee \psi) \equiv \exists x\varphi \vee \exists x\psi \qquad \forall x(\varphi \wedge \psi) \equiv \forall x\varphi \wedge \forall x\psi$$

Seien Σ, Σ' Signaturen mit $\Sigma \subseteq \Sigma'$ und A eine Σ'-Struktur.

Eine Σ-Formel φ und eine Σ'-Formel ψ heißen **erfüllbarkeitsäquivalent bzgl.** Σ, geschrieben: $\varphi \approx_\Sigma \psi$, wenn es für alle Σ-Strukturen A und $h \in A^X$ eine Σ'-Struktur B gibt, deren Σ-Redukt mit A übereinstimmt und die folgende Äquivalenz erfüllt:

$$A \models_h \varphi \;\Leftrightarrow\; B \models_h \psi.$$

Eine prädikatenlogische Formel $\forall x_1 \ldots \forall x_n \varphi$ heißt **Skolemnormalform (SNF)**, wenn φ eine implikative Normalform über $At_\Sigma(X)$ ist (siehe Kapitel 6).

Satz 8.7: Jede prädikatenlogische Formel hat eine erfüllbarkeitsäquivalente Skolemnormalform.

Der **prädikatenlogische Schnittkalkül** PSK besteht aus den in Abschnitt 8.2 angegebenen vier Regeln zur Transformation prädikatenlogischer Gentzenformeln. Die Gentzenformel des Sukzedenten der Schnittregel heißt **Resolvente**.

Eine PSK**-Widerlegung** einer Menge Φ von Gentzenformeln ist eine PSK-Ableitung von $\Phi \vdash \bot$.

Satz 8.11: Die Vollständigkeit von ASK-Widerlegungen gilt auch für unendliche Gentzenformelmengen Φ mit höchstens abzählbar vielen Atomen.

Korrektheit und Vollständigkeit von PSK**-Ableitungen bzw. -Widerlegungen** (Sätze 8.8 und 8.12): Für endliche Mengen Φ prädikatenlogischer Gentzenformeln gilt:

Aus $\Phi \vdash_{PSK} \psi$ folgt $\Phi \models \psi$.

$\Phi \vdash_{PSK} \bot$ gilt genau dann, wenn Φ unerfüllbar ist.

Gleichheitsstruktur: Σ-Struktur A, die das **Gleichheitsprädikat** $\equiv$: 2 durch Δ_A und das **Unleichheitsprädikat** $\not\equiv$: 2 durch $A^2 \setminus \Delta_A$ interpretiert.

Gleichheitsmodell von φ: Gleichheitsstruktur, die φ erfüllt.

Sei $\sim^A = A^2 \setminus \not\equiv^A$. Für alle Σ-Strukturen A sind $A/\equiv^A$ und $A/\sim^A$ Gleichheitsstrukturen und genau dann Modelle von φ, wenn A ein Modell von $\varphi \wedge \bigwedge cong_\Sigma$ bzw. $\varphi \wedge \bigwedge disg_\Sigma$ ist.

Satz 8.13 (iii): Eine implikative Normalform φ über $At_\Sigma(X)$ hat genau dann kein Gleichheitsmodell, wenn $\varphi \wedge \bigwedge cong_\Sigma$ oder $\varphi \wedge \bigwedge disg_\Sigma$ unerfüllbar ist.

Korrektheit und Vollständigkeit von PSK**-Widerlegungen in Gleichheitsmodellen**: Für endliche Mengen Φ prädikatenlogischer Gentzenformeln gilt:

$$\Phi \cup cong_\Sigma \vdash_{PSK} \bot \;\Leftrightarrow\; \Phi \text{ hat kein Gleichheitsmodell} \;\Leftrightarrow\; \Phi \cup disg_\Sigma \vdash_{PSK} \bot.$$

9 Logische Programmierung

Im weitesten Sinne ist ein logisches Programm eine Menge LP prädikatenlogischer Formeln. Seine Eingabe ist eine – Anfrage genannte – weitere prädikatenlogische Formel. Es berechnet eine Menge von *Lösungen* der Anfrage in einem Modell von LP.

Funktionale oder objektorientierte Programme repräsentieren Funktionen, logische Programme sollen Relationen repräsentieren. Das schließt schon mal alle unerfüllbaren Formeln von ihrer Verwendung als logische Programme aus.

Sei f eine Operation von Σ. Eine Σ-Gleichung über X (siehe Abschnitt 5.3) der Form $\varphi \Rightarrow ft \equiv u$ heißt **Hornformel für** f.

Sei p eine Prädikat von Σ, das weder Gleichheits- noch Ungleichheitsprädikat ist (siehe Abschnitt 8.3). Eine Gentzenformel der Form $\varphi \Rightarrow pt$ heißt **Hornformel für** p.

Damit die durch eine Hornformel "definierte" Funktion bzw. das durch sie "definierte" Prädikat am Anfang der Formel steht, schreiben wir $ft \equiv u \Leftarrow \varphi$ bzw. $pt \Leftarrow \varphi$ an Stelle von $\varphi \Rightarrow ft \equiv u$ bzw. $\varphi \Rightarrow pt$.

Eine Gentzenformel $pt \Rightarrow \varphi$ mit $pt \in At_\Sigma(X)$ heißt **Cohornformel für** p.

Eine endliche Menge LP von Horn- bzw. Cohornformeln für eine Teilmenge P von Σ heißt **logisches** bzw. **cologisches Σ-Programm für** P.

9.1 Beispiele logischer Programme

Beispiel 9.1

Die Signatur Σ bestehe aus den nullstelligen Operationen *zero* und *nil* (*leere Liste*), der unären Operation *succ* (*successor, Nachfolger*), den binären Operationen $+$, $:$, $++$ und $\setminus$, den unären Prädikaten *Nat*, *sorted* und *unsorted* und den binären Prädikaten $\leq$, $>$, $\in$ und *perm*.

Sei $x, y, s, s' \in X$. Die folgenden (Co)Hornformeln bilden (co)logische Programme für einzelne Elemente von Σ:

$$
\begin{array}{ll}
(nat) & Nat(zero) \\
 & Nat(succ(x)) \Leftarrow Nat(x) \\
(less) & zero \leq x \Leftarrow Nat(x) \\
 & succ(x) \leq succ(y) \Leftarrow x \leq y \\
(greater) & zero > x \Rightarrow \bot \\
 & succ(x) > succ(y) \Rightarrow x > y \\
(plus) & zero + x \equiv x \Leftarrow Nat(x) \\
 & succ(x) + y \equiv succ(x + y) \Leftarrow Nat(x) \wedge Nat(y) \\
(sorted) & sorted(nil) \\
 & sorted(x : nil)) \Leftarrow Nat(x) \\
 & sorted(x : (y : s)) \Leftarrow x \leq y \wedge sorted(y : s) \\
(unsorted) & unsorted(nil) \Rightarrow \bot \\
 & unsorted(x : nil) \Rightarrow \bot \\
 & unsorted(x : (y : s)) \Rightarrow x > y \vee unsorted(y : s) \\
 & unsorted(x : (y : s)) \Rightarrow Nat(x) \\
 & unsorted(x : (y : s)) \Rightarrow Nat(y) \\
(concat) & nil{+}{+}s \equiv s \\
 & (x : s){+}{+}s' \equiv x : (s{+}{+}s') \Leftarrow Nat(x)
\end{array}
$$

$$
\begin{array}{ll}
(remove) & nil \setminus x \equiv nil \Leftarrow Nat(x) \\
& (x : s) \setminus x \equiv s \Leftarrow Nat(x) \\
& (x : s) \setminus y \equiv x : (s \setminus y) \Leftarrow Nat(x) \wedge Nat(y) \wedge x \not\equiv y \\
(elem) & x \in x : s \Leftarrow Nat(x) \\
& x \in y : s \Leftarrow x \in s \Leftarrow Nat(x) \wedge Nat(y) \\
(perm) & perm(nil, nil) \\
& perm(s, x : s') \Leftarrow x \in s \wedge Nat(x) \wedge perm(s \setminus x, s')
\end{array}
$$

Die Menge $\mathbb{N}^*$ wird mit folgender Interpretation von Σ zur Σ-Struktur $\mathcal{A}$:

Für alle $v, w \in \mathbb{N}^*$ und $a_1, \ldots, a_n \in \mathbb{N}$,

$$
\begin{array}{rcl}
zero^{\mathcal{A}} & = & 0, \\
succ^{\mathcal{A}}(w) & = & \begin{cases} w+1 & \text{falls } w \in \mathbb{N}, \\ \epsilon & \text{sonst,} \end{cases} \\
v +^{\mathcal{A}} w & = & \begin{cases} v+w & \text{falls } v, w \in \mathbb{N}, \\ \epsilon & \text{sonst,} \end{cases} \\
nil^{\mathcal{A}} & = & \epsilon, \\
v :^{\mathcal{A}} w & = & \begin{cases} cons_v(w) & \text{falls } v \in \mathbb{N}, \quad \text{(siehe Abschnitt 4.7)} \\ \epsilon & \text{sonst,} \end{cases} \\
v +\!\!+^{\mathcal{A}} w & = & conc(v, w), \qquad \text{(siehe Abschnitt 4.7)} \\
(a_1, \ldots, a_n) \setminus^{\mathcal{A}} w & = & \begin{cases} (a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n) & \text{falls } w = a_i \in \mathbb{N} \text{ und} \\ & \quad a_j \neq w \text{ für alle } 1 \leq j < i, \\ (a_1, \ldots, a_n) & \text{falls } w \in \mathbb{N} \text{ und} \\ & \quad a_i \neq w \text{ für alle } 1 \leq j \leq n, \\ \epsilon & \text{sonst,} \end{cases} \\
Nat^{\mathcal{A}} & = & \mathbb{N}, \\
\leq^{\mathcal{A}} & = & \{(a, b) \in \mathbb{N}^2 \mid a \leq b\}, \\
>^{\mathcal{A}} & = & \{(a, b) \in \mathbb{N}^2 \mid a > b\}, \\
\equiv^{\mathcal{A}} & = & \Delta_{\mathbb{N}^*}, \qquad \text{(siehe Abschnitt 3.2)} \\
\not\equiv^{\mathcal{A}} & = & (\mathbb{N}^*)^2 \setminus \Delta_{\mathbb{N}^*}, \\
\in^{\mathcal{A}} & = & \{(a, (a_1, \ldots, a_n)) \in \mathbb{N} \times \mathbb{N}^* \mid \exists\, 1 \leq i \leq n : a = a_i\}, \\
sorted^{\mathcal{A}} & = & \{(a_1, \ldots, a_n) \in \mathbb{N}^* \mid \forall\, 1 \leq i < n \in \mathbb{N} : a_i \leq a_{i+1}\}, \\
unsorted^{\mathcal{A}} & = & \{(a_1, \ldots, a_n) \in \mathbb{N}^+ \mid \exists\, 1 \leq i < n \in \mathbb{N} : a_i > a_{i+1}\}, \\
perm^{\mathcal{A}} & = & \{(v, w) \in (\mathbb{N}^*)^2 \mid w \text{ ist eine ☞ Permutation von } v\}.
\end{array}
$$

Aufgabe Zeigen Sie $\mathcal{A} \models \bigwedge LP$. ❑

Beispiel 9.2 ☞ Türme von Hanoi

n Scheiben sind von 1 nach 3 zu transportieren, wobei ein 2 als Zwischenlager dient. Die Scheiben dürfen nur mit nach oben abnehmender Größe gestapelt werden. Jeder während des Transports erreichte Zustand ist ein Tripel von Listen der bei 1, 2 bzw. 3 gelagerten Scheiben. Z.B. lässt sich die Folge der Zustände, die beim Transport von fünf Scheiben von 1 nach 2 durchlaufen werden, wie folgt graphisch darstellen:

Das logische Programm soll keine Zustandsfolge, sondern die Folge der Ortswechsel, die erforderlich sind, um n Scheiben von Position 1 zu Position 3 zu transportieren. Wir modellieren die Aufgabe mit Hilfe einer Signatur Σ, die aus den Operationen von Beispiel 9.1 und dem fünfstelligen Prädikat *actions* besteht. Deren intendierte Bedeutung ist unten als Σ-Struktur beschrieben.

Sei $n, x, y, z, act, acts, acts' \in X$.

Die folgenden Hornformeln bilden ein logisches Programm LP für $actions$:

$$actions(zero, x, y, z, nil) \tag{1}$$

$$actions(succ(n), x, y, z, acts{+}{+}((x, z) : acts') \Leftarrow actions(n, x, z, y, acts) \wedge actions(n, y, x, z, acts') \tag{2}$$

(2) drückt aus, dass $n+1$ Scheiben von Turm x über Turm y nach Turm z getragen werden können, indem zunächst n Scheiben von x über z nach y, dann eine Scheibe von x nach z und schließlich n Scheiben von y über x nach z transportiert werden. Das Paar (x, z) steht für "Transportiere die oberste Scheibe von x nach z".

Das Bild auf der vorigen Seite visualisiert die Zustandsfolge, die von der (einzigen) Aktionsfolge $acts''$ erzeugt wird, die $actions(5, 1, 2, 3, acts'')$ erfüllt. Wegen (2) gibt es Aktionsfolgen $acts$ und $acts'$, die das Goal

$$actions(4, 1, 3, 2, acts) \wedge actions(4, 2, 1, 3, acts') \wedge acts{+}{+}((1, 3) : acts') \equiv acts''$$

erfüllen. Im Bild führen $acts$ vom Zustand links oben zum Zustand links unten, die Aktion (1,3) vom Zustand links unten zum Zustand rechts oben und $acts'$ vom Zustand rechts oben zum Zustand rechts unten.

Sei $A = \mathbb{N} \times \mathbb{N}^3 \times (\mathbb{N}^2)^*$. Eine Schrittfunktion $F : \mathcal{P}(A) \to \mathcal{P}(A)$ sei wie folgt definiert: Für alle $T \subseteq A$,

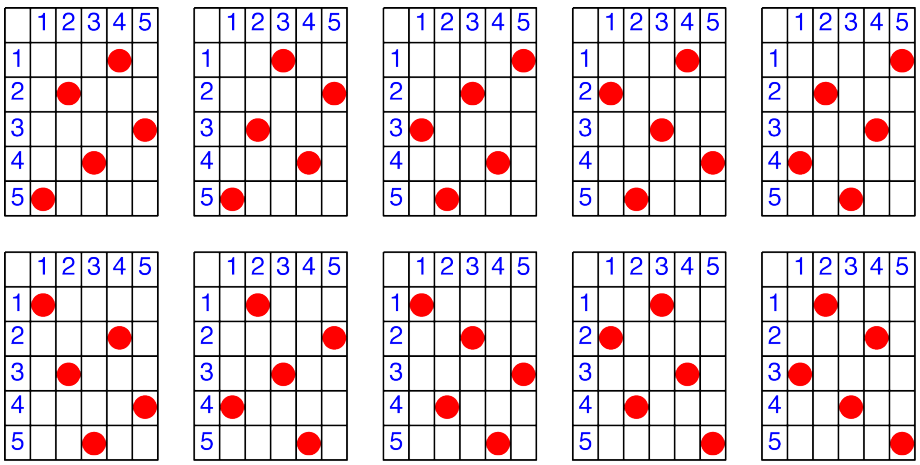
$$\begin{aligned} F(T) \quad = \quad & \{(0,a,b,c,\epsilon) \mid a,b,c \in \mathbb{N}\} \cup \\ & \{(n+1,a,b,c,conc(v,cons_{(a,c)}(w)) \mid (n,a,c,b,v),(n,b,a,c,w) \in T\}. \end{aligned}$$

Wir wählen A als Trägermenge einer Σ-Algebra $\mathcal{B}$, interpretieren dort *actions* durch den kleinsten Fixpunkt von F (siehe Kapitel 4) und *zero*, *succ*, *nil*, : und $++$ wie in Beispiel 9.1.

Aufgabe Zeigen Sie $\mathcal{B} \models \bigwedge LP$. ❑

Beispiel 9.3 ☞ Damenproblem

n Damen auf einem $(n \times n)$-Schachbrett sollen so platziert werden, dass sie sich nach den Schachregeln nicht schlagen können, also keine zwei Damen auf derselben Diagonalen, Horizontalen oder Vertikalen stehen. Z.B. gibt es für fünf Damen auf einem 5×5-Schachbrett genau zehn sichere Platzierungen:



Die Platzierung einer Dame in jeder Zeile eines $(n \times n)$-Schachbrettes wird als Permutation $(k_1,\ldots,k_n)$ von $(1,2,\ldots,n)$ dargestellt: k_i ist die Spaltenposition der Dame in der i-ten Zeile. Der Brettbelegung links oben entspricht z.B. die Permutation $(4,2,5,3,1)$.

Wir modellieren die Aufgabe mit Hilfe einer Signatur Σ, die neben Operationen und Prädikaten von Beispiel 9.1 die unäre Operation *enum* und die binären Prädikate *queens*, *board* und *safe* besteht. Deren intendierte Bedeutung ist unten als Σ-Struktur $\mathcal{C}$ beschrieben, die folgende Bedingungen erfüllt:

$((1,\ldots,n),(k_1,\ldots,k_n))$ gehört genau dann zu $board^{\mathcal{C}}$, wenn $(k_1,\ldots,k_n)$ eine sichere Platzierung von n Damen repräsentiert, also eine, bei der sich keine zwei Damen schlagen können.

$(1,(k,k_1,\ldots,k_{n-1}))$ gehört genau dann zu $safe^{\mathcal{C}}$, wenn unter der Voraussetzung, dass $(k_1,\ldots,k_{n-1})$ eine sichere Damenplatzierung auf den unteren $n-1$ Zeilen repräsentiert, $(k,k_1,\ldots,k_{n-1})$ eine sichere Damenplatzierung auf dem gesamten $(n \times n)$-Brett darstellt.

Sei $n,x,y,s,s',val \in X$. Die folgenden Hornformeln bilden ein logisches Programm LP für die Prädikate von Σ:

$$\begin{aligned} &(queens) && queens(n,val) \Leftarrow board(enum(n),nil,val) \\ &(board) && board(nil,val,val) \\ & && board(s,val,val') \\ & && \quad \Leftarrow x \in s \wedge safe(succ(zero),x:val) \wedge board(s \setminus x, x:val,val') \end{aligned}$$

$$
\begin{array}{ll}
(safe) & safe(n, x : nil) \\
 & safe(n, x : (y : s)) \Leftarrow x \not\equiv y + n \wedge y \not\equiv x + n \wedge safe(succ(n), x : s) \\
(enum) & enum(zero) \equiv nil \\
 & enum(succ(n)) \equiv enum(n) +\!\!+ (succ(n) : nil)
\end{array}
$$

Hornformeln für $\in$, $\not\equiv$ und $+\!\!+$ stehen in Beispiel 9.1.

Die im Programm blau bzw. rot gefärbte Liste von Zahlen besteht aus den noch nicht vergebenen bzw. vergebenen Spalten, in denen Damen sicher platziert werden können.

$\mathcal{C}$ ist wie folgt definiert: Für alle $n \in \mathbb{N}^*$,

$$
\begin{array}{rcl}
enum^{\mathcal{C}}(n) & = & \begin{cases} (1, \ldots, n) & \text{falls } n \in \mathbb{N}, \\ \epsilon & \text{sonst,} \end{cases} \\
safe^{\mathcal{C}} & = & \{(i, (a, a_1, \ldots, a_n)) \mid i, a, a_1, \ldots, a_n \in \mathbb{N} : \\
 & & \qquad \forall\, i \leq j \leq n : a_j - j \neq a \neq a_j + j\}, \\
queens^{\mathcal{C}} & = & \{(n, w) \in \mathbb{N} \times \mathbb{N}^* \mid ((1, \ldots, n), nil, w) \in board^{\mathcal{C}}\}, \\
board^{\mathcal{C}} & = & lfp(F), \qquad \text{(siehe Kapitel 4)}
\end{array}
$$

wobei die Schrittfunktion $F : \mathcal{P}((\mathbb{N}^*)^3) \to \mathcal{P}((\mathbb{N}^*)^3)$ wie folgt definiert ist:

Für alle $T \subseteq (\mathbb{N}^*)^3$,

$$
\begin{array}{rcl}
F(T) & = & \{(\epsilon, w, w) \mid w \in \mathbb{N}^*\} \cup \\
 & & \{(v, w, w') \mid \exists\, a \in^{\mathcal{C}} v \wedge (1, aw) \in safe^{\mathcal{C}} \wedge (v \setminus^{\mathcal{C}} a, aw, w') \in T\}.
\end{array}
$$

$zero$, $succ$, $+$, nil, $:$, $+\!\!+$, $\setminus$, $\equiv$, $\not\equiv$ und $\in$ werden in $\mathcal{C}$ wie in $\mathcal{A}$ interpretiert (siehe Beispiel 9.1).

Aufgabe Zeigen Sie $\mathcal{C} \models \bigwedge LP$. ❑

Aufgabe Überführen Sie LP in eine Kripke-Struktur mit Zustandsmenge $\mathbb{N}^* \times \mathbb{N}^*$ und Transitionsfunktion δ, so dass durch wiederholte Anwendung von δ alle sicheren Platzierungen von n Damen auf einem $(n \times n)$-Schachbrett berechnet werden können. ❑

9.2 SLD-Kalküle

Eine endliche Konjunktion oder Disjunktion von Σ-Atomen über X heißt **Σ-Goal** bzw. **Σ-Cogoal** über X. $\top$ bzw. $\bot$ ist ein weiteres Goal bzw. Cogoal.

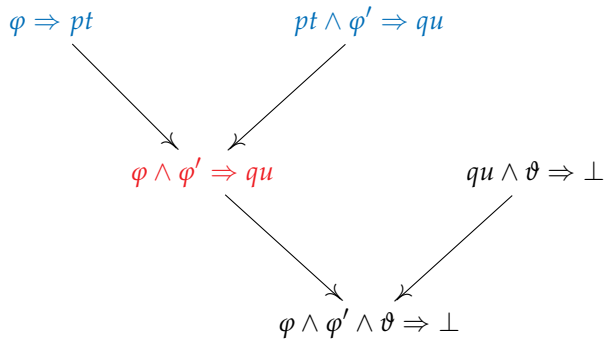
$G_\Sigma(X)$ und $coG_\Sigma(X)$ bezeichnen die Mengen der Σ-Goals bzw. Σ-Cogoals über X.

Berechnungen auf der Basis eines logischen Programms LP entsprechen Schnittwiderlegungen von $\Phi = LP \cup \{\varphi\}$, wobei φ ein (Co)Goal ist. Wegen der speziellen Form von LP bzw. φ lässt sich der Schnittkalkül vereinfachen, ohne dass seine Vollständigkeit bzgl. Widerlegungen von Φ verlorengeht.

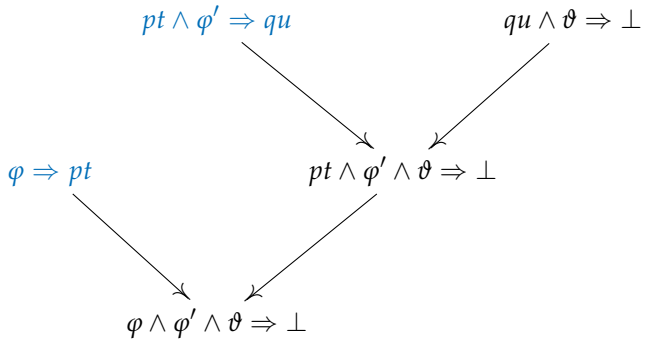
Dazu betrachten wir zunächst variablenfreie PSK-Ableitungen der Form $\Phi \vdash \psi$ mit einem Goal ψ.

Jede solche Ableitung s lässt sich schrittweise in eine Ableitung von $\Phi \vdash \psi$ transformieren, in der keine zwei Hornformeln $\varphi \Rightarrow pt$ und $pt \wedge \varphi' \Rightarrow qu$ geschnitten werden.

Fall 1: Die Resolvente $\varphi \wedge \varphi' \Rightarrow qu$ eines solchen Schnittes wird in s mit einem Goal – das die Form $qu \wedge \vartheta \Rightarrow \bot$ haben müsste – geschnitten. Dann kann sie aus s entfernt werden, weil dieser Schnitt dieselbe Resolvente liefert wie der Schnitt der ersten Hornformel $\varphi \Rightarrow pt$ mit der Resolvente $pt \wedge \varphi' \wedge \vartheta \Rightarrow \bot$ der zweiten Hornformel $pt \wedge \varphi' \Rightarrow qu$ und dem Goal:



wird transformiert in:



Fall 2: Die Resolvente $\varphi \wedge \varphi' \Rightarrow qu$ wird in s mit keinem Goal geschnitten. Dann trägt sie zur Ableitung von $\Phi \vdash \psi$ nichts bei, kann also ebenfalls aus s entfernt werden.

Demnach gilt $\Phi \vdash_{ASK} \psi$ genau dann, wenn es eine Folge $(\psi_1, \ldots, \psi_n)$ von Goals gibt derart, dass $\psi_1 \in \Phi$, $\psi_n = \psi$ und für alle $1 < i \leq n$ ψ_i die Resolvente von ψ_{i-1} und einer Hornformel von Φ ist — und nicht die Resolvente zweier *beliebiger* Vorgänger von ψ_i.

Analog kann man alle Resolventen zweier Cohornformeln aus variablenfreien Schnittableitungen von $\Phi = LP \cup \{\varphi\} \vdash \psi$ entfernen, wenn LP eine cologisches Programm ist und φ, ψ Cogoals sind, so dass $\Phi \vdash_{PSK} \psi$ genau dann gilt, wenn es eine Folge $(\psi_1, \ldots, \psi_n)$ von Cogoals gibt derart, dass $\psi_1 \in \Phi$, $\psi_n = \psi$ und für alle $1 < i \leq n$ ψ_i die Resolvente von ψ_{i-1} und einer Cohornformel von Φ ist.

Die Einschränkung des prädikatenlogischen Schnittkalküls auf solche Regelanwendungen führt zum *(co)SLD-Kalkül*, mit dem (Co)Goals gelöst werden sollen. SLD steht für *selective linear definite clause resolution*. *selective* weist auf eine mögliche Strategie hin, nach der das Atom ausgewählt wird, das bei einem Schnitt entfernt wird.

linear bedeutet, dass sich jede Formel φ einer *SLD*-Ableitung nur aus ihrem jeweiligen direkten Vorgänger (und LP) ergibt, dass also φ nicht wie beim Schnittkalkül auch von früheren Vorgängern abhängt (s.o.). Der Begriff *clause* (deutsch: Klausel) umfasst Hornformeln (*definite clauses*) und Goals.

Um sowohl Goals als auch Cogoals zu verarbeiten, benötigen wir einen hierarchischen Ansatz, in dem LP auch **Basisprädikate** enthalten darf, die von LP verwendet, aber nicht "definiert" werden.

Dazu wird von der gegebenen Signatur Σ eine **Basissignatur** $\Sigma' \subseteq \Sigma$ abgespalten, die alle Operationen von Σ enthält. Die Prädikate von Σ' nennen wir **Basisprädikate**.

Zu jedem $p : n \in \Sigma'$ gehöre auch das **Komplementprädikat** $\overline{p} : n$ zu Σ'. Im Fall $\equiv \in \Sigma'$ sei $\overline{\equiv} = \not\equiv$.

Stellen alle Operationen von Σ Konstruktoren dar, dann können Gleichheits- bzw. Ungleichheitsprädikate zu $\Sigma \setminus \Sigma'$ gehören. Gemäß Abschnitt 8.3 spezifiziert man sie dann durch das (co)logische Programm $cong_\Sigma$ bzw. $disg_\Sigma$, damit diese Prädikate im Fixpunktmodell als Gleichheit bzw. Ungleichheit interpretiert werden (siehe Abschnitt 9.3). Sollen jedoch einige Operationen durch Gleichungen (siehe Abschnitt 5.3) spezifiziert werden, dann

gehört $\equiv$ zu Σ'.

Ein (co)logisches Σ-Programm LP heißt **Programm mit Basisprädikaten**, wenn für alle $pt \Leftarrow \varphi \in LP$ bzw. $pt \Rightarrow \varphi \in LP$ p nicht zu Σ' gehört.

Sei A eine erreichbare **komplementabgeschlossene** Σ'-Gleichheitsstruktur (siehe Abschnitte 5.2 und 8.3), d.h. für alle $p : n \in \Sigma'$ gilt $\overline{p}^B = A^n \setminus p^B$.

$Struct_{\Sigma,A}$ bezeichnet die Klasse aller Σ-Strukturen B, deren Σ'-Redukt mit A übereinstimmt. $Struct_{\Sigma,A,LP} =_{def} \{B \in Struct_{\Sigma,A} \mid B \models LP\}$.

Die prädikatenlogische Folgerungsrelation für $Struct_{\Sigma,A}$ bezeichnen wir mit $\models_A$:

Für alle $\Phi \subseteq T_{PL}(At_\Sigma(X))$ und $\psi \in T_{PL}(At_\Sigma(X))$,

$$\Phi \models_A \psi \quad \Leftrightarrow_{def} \quad \forall B \in Struct_{\Sigma,A} : B \models \bigwedge \Phi \Rightarrow B \models \psi.$$

Φ ist A**-erfüllbar**, wenn es $B \in Struct_{\Sigma,A}$ mit $B \models \bigwedge \Phi$ gibt.

Sei LP ein (co)logisches Σ-Programm mit Basisprädikaten und LP' die Menge aller (Co)Hornformeln φ', für die es $\varphi \in LP$ und eine Variablenumbenennung τ mit $\varphi\tau = \varphi'$ gibt.

Sei LP logisch. Der **SLD-Kalkül für** (LP, A) ist analytisch und besteht aus vier Regeln:

Faktorregel

$$\frac{LP \vdash \vartheta \wedge \vartheta' \wedge \varphi}{LP \vdash \vartheta\sigma \wedge \varphi\sigma} \qquad \vartheta, \vartheta' \in At_\Sigma(X),\ \varphi \in G_\Sigma(X),\ \sigma = unify(\vartheta, \vartheta')$$

Resolution über LP

$$\frac{LP \vdash \vartheta' \wedge \psi}{LP \vdash \varphi\sigma \wedge \psi\sigma} \qquad \begin{array}{l} \vartheta' \in At_\Sigma(X),\ \psi \in G_\Sigma(X),\ \vartheta \Leftarrow \varphi \in LP', \\ \sigma = unify(\vartheta, \vartheta'),\ var(\vartheta \Leftarrow \varphi) \parallel var(\vartheta' \wedge \psi) \end{array}$$

Basisregel

$$\frac{LP \vdash \vartheta \wedge \varphi}{\varphi\sigma} \qquad \vartheta \in At_{\Sigma'}(X),\ \varphi \in G_\Sigma(X),\ \sigma \in T_\Sigma^X,\ A \models \vartheta\sigma$$

Stopregel

$$\frac{}{LP \vdash \top}$$

Analog zu Satz 8.8 (i) erhält man die Korrektheit des SLD-Kalküls für (LP, A):

Für alle SLD-Ableitungen $(LP \vdash \varphi_1, \ldots, LP \vdash \varphi_n)$ gilt

$$LP \models_A \varphi_n \Rightarrow \exists Z \varphi_1, \tag{1}$$

wobei $Z = var(\varphi_1) \setminus var(\varphi_n)$. Im Fall $\varphi_n = \top$ gilt demnach $LP \models_A any(\varphi_1)$ (siehe Kapitel 8).

Sei LP cologisch. Der **coSLD-Kalkül für** (LP, A) ist analytisch und besteht aus vier Regeln:

Summandregel

$$\frac{LP \vdash \vartheta \vee \vartheta' \vee \varphi}{LP \vdash \vartheta\sigma \vee \varphi\sigma} \qquad \vartheta, \vartheta' \in At_\Sigma(X),\ \varphi \in coG_\Sigma(X),\ \sigma = unify(\vartheta, \vartheta')$$

Coresolution über LP

$$\frac{LP \vdash \vartheta' \vee \psi}{LP \vdash \varphi\sigma \vee \psi\sigma} \qquad \begin{array}{l} \vartheta' \in At_\Sigma(X),\ \psi \in coG_\Sigma(X),\ \vartheta \Rightarrow \varphi \in LP', \\ \sigma = unify(\vartheta, \vartheta'),\ var(\vartheta \Rightarrow \varphi) \parallel var(\vartheta' \vee \psi) \end{array}$$

Basisregel

$$\frac{LP \vdash \vartheta \vee \varphi}{\varphi\sigma} \quad \vartheta \in At_{\Sigma'}(X),\ \varphi \in coG_\Sigma(X),\ \sigma \in T_\Sigma^X,\ A \not\models \vartheta\sigma$$

Stopregel

$$\frac{LP \vdash \bot}{}$$

Analog zu Satz 8.8 (i) erhält man die Korrektheit des *coSLD*-Kalküls für (LP, A):

Für alle *coSLD*-Ableitungen $(LP \vdash \varphi_1, \ldots, LP \vdash \varphi_n)$ gilt

$$LP \models_A \forall Z \varphi_1 \Rightarrow \varphi_n, \tag{2}$$

wobei $Z = var(\varphi_1) \setminus var(\varphi_n)$. Im Fall $\varphi_n = \bot$ gilt demnach $LP \models_A any(\neg\varphi_1)$ (siehe Kapitel 8).

Aufgrund obiger Bemerkungen über variablenfreie *PSK*-Ableitungen und da diese *ASK*-Ableitungen entsprechen, gilt Satz 8.11 auch für variablenfreie *SLD*-Ableitungen (die auch keine Faktor- bzw. Summandregelanwendungen enthalten):

Satz 9.4 (Vollständigkeit variablenfreier $(co)SLD$-Ableitungen)

(i) Sei LP ein variablenfreies logisches Programm mit Basisprädikaten und φ ein Σ-Goal über X. Ist $LP \cup \{\neg\varphi\}$ A-unerfüllbar, dann gilt $LP \vdash_{SLD,A} \top$.

(ii) Sei LP ein variablenfreies cologisches Programm mit Basisprädikaten und φ ein Σ-Cogoal über X. Ist $LP \cup \{\varphi\}$ A-unerfüllbar, dann gilt $LP \vdash_{SLD,A} \bot$. ❑

Satz 9.5 (Lifting und Vollständigkeit von $(co)SLD$-Ableitungen)

(i) Sei LP ein logisches Σ-Programm mit Basisprädikaten, φ ein Σ-Goal, $\sigma : X \to T_\Sigma$ und

$$abl_0 = (Exp(LP) \vdash \varphi\sigma, \ldots, Exp(LP) \vdash \psi')$$

eine variablenfreie *SLD*-Ableitung. Dann gibt es eine *SLD*-Ableitung

$$abl = (LP \vdash \varphi, \ldots, LP \vdash \psi)$$

und $\rho : X \to T_\Sigma(X)$ mit $\psi\rho = \psi'$ und $\gamma\rho = \sigma$, wobei γ die Komposition der mgus von abl ist.

(ii) Aus $LP \models_A any(\varphi)$ folgt die Existenz von abl mit $\psi = \top$.

(iii) Sei LP ein cologisches Σ-Programm mit Basisprädikaten, φ ein Σ-Cogoal, $\sigma : X \to T_\Sigma$ und

$$abl_0 = (Exp(LP) \vdash \varphi\sigma, \ldots, Exp(LP) \vdash \psi')$$

eine variablenfreie *coSLD*-Ableitung. Dann gibt es eine *coSLD*-Ableitung

$$abl = (LP \vdash \varphi, \ldots, LP \vdash \psi)$$

und $\rho : X \to T_\Sigma(X)$ mit $\psi\rho = \psi'$ und $\gamma\rho = \sigma$, wobei γ die Komposition der mgus von abl ist.

(iv) Aus $LP \models_A any(\neg\varphi)$ folgt die Existenz von abl mit $\psi = \bot$.

Beweis von (i) *durch Induktion über die Länge n von* abl_0.

Fall 1: $n = 0$. Dann ist $\psi' = \varphi\sigma$. Also ist $(LP \vdash \varphi)$ eine *SLD*-Ableitung, die die Bedingungen von (i) erfüllt, wenn man $\rho = \sigma$ und $\gamma = id_X$ setzt.

Fall 2: $n > 0$. Dann ist ψ' die Resolvente einer Hornformel θ von $Exp(LP)$ und eines Goals ψ'_1 und es gibt eine variablenfreie *SLD*-Ableitung

$$(Exp(LP) \vdash \varphi\sigma, \ldots, Exp(LP) \vdash \psi'_1)$$

mit weniger als n Resolventen. Nach Induktionsvoraussetzung gibt es ein Goal ψ_1, eine *SLD*-Ableitung

$$abl' = (LP \vdash \varphi, \ldots, LP \vdash \psi_1)$$

und $\rho' : X \to T_\Sigma(X)$ mit $\psi_1\rho' = \psi'_1$ und $\gamma\rho' = \sigma$, wobei γ die Komposition der mgus von abl' ist.

Wegen $\psi_1\rho' = \psi'_1$ und da ψ' eine aussagenlogische Resolvente von θ und ψ'_1 ist, liefert Lemma 8.10 ein Goal ψ, eine *SLD*-Ableitung

$$abl'' = (LP \vdash \psi_1, \ldots, LP \vdash \psi)$$

und $\rho : X \to T_\Sigma(X)$ mit $\psi\rho = \psi'$ und $\gamma'\rho = \rho'$, wobei γ' die Komposition der mgus von abl'' ist.

Setzt man abl' und abl'' zusammen, erhält man eine *SLD*-Ableitung abl, die die Bedingungen von (i) erfüllt: $\psi\rho = \psi'$, $\gamma\gamma'\rho = \gamma\rho' = \sigma$ und $\gamma'\gamma$ ist die Komposition der mgus von abl.

Beweis von (ii).

Sei $LP \models_A any(\varphi)$. Dann ist $LP \cup \{\neg any(\varphi)\}$, also auch $LP \cup \{\neg\varphi\}$ A-unerfüllbar. Nach Satz 8.9 ist daher auch die Termexpansion von $LP \cup \{\neg\varphi\}$ A-unerfüllbar. Da diese abzählbar ist und mit $Exp(LP) \cup \{\neg\varphi\sigma \mid \sigma \in T_\Sigma^X\}$ übereinstimmt, gibt es abl_0 mit $\psi' = \top$ nach Satz 9.4 (i). Aus dem ersten Teil des Beweises folgt dann die Existenz von abl.

Beweis von (iii).

Die Existenz einer eine *coSLD*-Ableitung

$$abl = (LP \vdash \varphi, \ldots, LP \vdash \psi)$$

mit den unter (ii) genannten Eigenschaften kann analog zu (i) gezeigt werden.

Beweis von (iv).

Sei $LP \models_A any(\neg\varphi)$. Dann ist $LP \cup \{any(\varphi)\}$, also auch $LP \cup \{\varphi\}$ A-unerfüllbar. Nach Satz 8.9 ist daher auch die Termexpansion von $LP \cup \{\varphi\}$ A-unerfüllbar. Da diese abzählbar ist und mit $Exp(LP) \cup \{\varphi\sigma \mid \sigma \in T_\Sigma^X\}$ übereinstimmt, gibt es abl_0 nach Satz 9.4 (ii). Aus dem ersten Teil des Beweises folgt dann die Existenz von abl. ❑

Zusammenfassend erhalten wir:

Sei A eine erreichbare komplementabgeschlossene Σ'-Struktur und $\models_{A,EQ} = \models_A \cap \models_{EQ}$.

Sei LP ein logisches Σ-Programm mit Basisprädikaten und φ ein Σ-Goal.

$$\boxed{LP \models_A any(\varphi) \;\Leftrightarrow\; LP \vdash_{SLD,A} \top.} \tag{3}$$

Enthält $\Sigma \setminus \Sigma'$ die Gleichheitsprädikate von Σ, dann folgt (4) aus (3) und Satz 8.13 (iii):

$$\boxed{LP \models_{A,EQ} any(\varphi) \;\Leftrightarrow\; LP \cup cong_\Sigma \vdash_{SLD,A} \top.} \tag{4}$$

Sei LP ein cologisches Σ-Programm mit Basisprädikaten und φ ein Σ-Cogoal.

$$\boxed{LP \models_A any(\neg\varphi) \;\Leftrightarrow\; LP \vdash_{SLD,A} \bot.} \tag{5}$$

Enthält $\Sigma \setminus \Sigma'$ die Ungleichheitsprädikate von Σ, dann folgt (6) aus (5) und Satz 8.13 (iii):

$$\boxed{LP \models_{A,EQ} any(\neg\varphi) \;\Leftrightarrow\; LP \cup dis_\Sigma \vdash_{SLD,A} \bot.} \tag{6}$$

9.3 Fixpunktmodelle logischer Programme

Schnitte zweier (Co)Hornformeln können niemals zu $\bot$ führen. Demnach ist jedes endliche (co)logische Σ-Programm LP nach Satz 8.12 (ii) erfüllbar.

Sei A eine Σ'-Struktur. $Struct_{\Sigma,A}$ ist halbgeordnet:

Für alle $B, C \in Struct_{\Sigma,A}$,

$$B \leq C \quad \Leftrightarrow_{def} \quad \text{für alle Prädikate } p \in \Sigma \setminus \Sigma' \text{ gilt } p^B \subseteq p^C.$$

Darüberhinaus ist $Struct_{\Sigma,A}$ ein **vollständiger Verband** mit Null- und Einselement:

Für alle $\mathcal{K} \subseteq Struct_{\Sigma,A}$ und $p : n \in \Sigma \setminus \Sigma'$,

$$p^0 = \emptyset, \;\; p^1 = A^n, \; p^{\bigsqcup \mathcal{K}} = \bigcup_{B \in \mathcal{K}} p^B, \; p^{\bigsqcap \mathcal{K}} = \bigcap_{B \in \mathcal{K}} p^B.$$

Folglich können wir die in Kapitel 4 für Potenzmengenverbände formulierten Ergebnisse (insbesondere die Sätze 4.1 und 4.9) auch auf $Struct_{\Sigma,A}$ anwenden.

Satz 9.6

Sei LP ein (co-)logisches Σ-Programm mit Basisprädikaten und $B = Ext(A, LP)$ die Σ-Struktur, die $p : n \in \Sigma$ wie folgt interpretiert:

$$p^B = \begin{cases} \{fold^A(t) \mid t \in T_\Sigma^n, \; LP \vdash_{SLD,A} pt\} & \text{falls } LP \text{ logisch ist,} \\ \{fold^A(t) \mid t \in T_\Sigma^n, \; LP \nvdash_{SLD,A} pt\} & \text{falls } LP \text{ cologisch ist.} \end{cases}$$

(i) Für alle Prädikate $p \in \Sigma'$ gilt $p^B = p^A$. B gehört also zu $Struct_{\Sigma,A}$.

(ii) B erfüllt LP.

(iii) Sei LP logisch. B ist der kleinste Fixpunkt der Schrittfunktion

$$F_{\Sigma,A} = F : Struct_{\Sigma,A} \to Struct_{\Sigma,A},$$

die wie folgt definiert ist:

Für alle $C \in Struct_{\Sigma,A}$ und $p : n \in \Sigma \setminus \Sigma'$,

$$p^{F(C)} = \{a \in A^n \mid \exists \; pt \Leftarrow \varphi \in LP, \; h \in A^X : h^*(t) = a \wedge C \models_h \varphi\}.$$

F^∞ ist F-abgeschlossen. Satz 4.9 (ii) impliziert daher $B = F^\infty$.

(iv) Sei LP cologisch. B ist der größte Fixpunkt der Schrittfunktion

$$F_{\Sigma,A} = F : Struct_{\Sigma,A} \to Struct_{\Sigma,A},$$

die wie folgt definiert ist: Für alle $C \in Struct_{\Sigma,A}$ und $p : n \in \Sigma \setminus \Sigma'$,

$$p^{F(C)} = \{a \in A^n \mid \forall \; pt \Rightarrow \varphi \in LP, \; h \in A^X : h^*(t) \neq a \vee C \models_h \varphi\}.$$

F_∞ ist F-abgeschlossen. Satz 4.9 (iv) impliziert daher $B = F_\infty$.

Beweis von (i). Sei $p : n \in \Sigma'$ und $a \in A^n$. Ist LP logisch, dann gibt es $t \in T_\Sigma^n$ mit

$$a \in p^B \;\Leftrightarrow\; a = fold^A(t) \in p^B \;\Leftrightarrow\; LP \vdash_{SLD,A} pt \;\Leftrightarrow\; A \models pt \;\Leftrightarrow\; a = fold^A(t) \in p^A.$$

Ist LP cologisch, dann gibt es $t \in T_\Sigma^n$ mit

$$a \in p^B \;\Leftrightarrow\; a = fold^A(t) \in p^B \;\Leftrightarrow\; LP \nvdash_{SLD,A} pt \;\Leftrightarrow\; A \models pt \;\Leftrightarrow\; a = fold^A(t) \in p^A.$$

Beweis von (ii). Fall 1: LP ist logisch. Sei $pt \Leftarrow p_1t_1 \wedge \cdots \wedge p_nt_n \in LP$ und $h \in g_B^*(p_it_i)$, also $h^*(t_i) \in p_i^B$ für alle $1 \leq i \leq n$. Nach Lemma 5.3 (iii) gibt es $\sigma : X \to T_\Sigma$ mit $fold^A \circ \sigma = h$ und $h^* = fold^A \circ \sigma^*$. Daraus folgt $fold^A(t_i\sigma) = h^*(t_i) \in p_i^B$, also

$$LP \vdash_{SLD,A} p_it_i\sigma$$

nach Definition von p_i^B. Folglich gibt es für alle $1 \leq i \leq n$ eine *SLD*-Ableitung

$$(LP \vdash p_it_i\sigma,\ LP \vdash \varphi_{i1},\ \ldots,\ LP \vdash \varphi_{ik_i},\ LP \vdash \top).$$

Diese Ableitungen lassen sich zu folgender Ableitung zusammensetzen:

$$\begin{aligned}
&(LP \vdash pt\sigma,\\
&LP \vdash p_1t_1\sigma \wedge \cdots \wedge p_nt_n\sigma, && \text{(Resolution mit } pt \Leftarrow \varphi)\\
&LP \vdash \varphi_{11} \wedge p_2t_2\sigma \wedge \cdots \wedge p_nt_n\sigma,\\
&\ldots,\\
&LP \vdash \varphi_{1k_1} \wedge p_2t_2\sigma \wedge \cdots \wedge p_nt_n\sigma,\\
&LP \vdash \top \wedge p_2t_2\sigma \wedge p_3t_3\sigma \wedge \cdots \wedge p_nt_n\sigma,\\
&LP \vdash \top \wedge \varphi_{21} \wedge p_3t_3\sigma \wedge \cdots \wedge p_nt_n\sigma,\\
&\ldots,\\
&LP \vdash \top \wedge \varphi_{2k_2} \wedge p_3t_3\sigma \wedge \cdots \wedge p_nt_n\sigma,\\
&LP \vdash \top \wedge \top \wedge p_3t_3\sigma \wedge \cdots \wedge p_nt_n\sigma,\\
&\ldots,\\
&LP \vdash \top \wedge \top \wedge \top \wedge \cdots \wedge \top = \top)
\end{aligned}$$

Also gilt $LP \vdash_{SLD,A} pt\sigma$, d.h. $h^*(t) = fold^A(t\sigma) \in p^B$ nach Definition von p^B und damit $h \in g_B^*(pt)$.

Fall 2: LP ist cologisch. Sei $pt \Rightarrow p_1t_1 \vee \cdots \vee p_nt_n \in LP$ und $h \in g_B^*(pt)$, also $h^*(t) \in p^B$. Nach Lemma 5.3 (iii) gibt es $\sigma : X \to T_\Sigma$ mit $fold^A \circ \sigma = h$ und $h^* = fold^A \circ \sigma^*$. Daraus folgt $fold^A(t\sigma) = h^*(t) \in p^B$, also

$$LP \nvdash_{SLD,A} pt\sigma \tag{1}$$

nach Definition von p^B. Angenommen, für alle $1 \leq i \leq n$ gilt

$$LP \vdash_{SLD,A} p_it_i\sigma.$$

Dann gäbe es für alle $1 \leq i \leq n$ eine *SLD*-Ableitung

$$(LP \vdash p_it_i\sigma,\ LP \vdash \varphi_{i1},\ \ldots,\ LP \vdash \varphi_{ik_i},\ LP \vdash \bot).$$

Diese Ableitungen ließen sich zu folgender Ableitung zusammensetzen:

$$\begin{aligned}
&(LP \vdash pt\sigma,\\
&LP \vdash p_1t_1\sigma \vee \cdots \vee p_nt_n\sigma, && \text{(Coresolution mit } pt \Rightarrow \varphi)\\
&LP \vdash \varphi_{11} \vee p_2t_2\sigma \vee \cdots \vee p_nt_n\sigma,\\
&\ldots,\\
&LP \vdash \varphi_{1k_1} \vee p_2t_2\sigma \vee \cdots \vee p_nt_n\sigma,\\
&LP \vdash \bot \vee p_2t_2\sigma \vee p_3t_3\sigma \vee \cdots \vee p_nt_n\sigma,\\
&LP \vdash \bot \vee \varphi_{21} \vee p_3t_3\sigma \vee \cdots \vee p_nt_n\sigma,\\
&\ldots,\\
&LP \vdash \bot \vee \varphi_{2k_2} \vee p_3t_3\sigma \vee \cdots \vee p_nt_n\sigma,\\
&LP \vdash \bot \vee \bot \vee \cdots \vee p_nt_n\sigma,\\
&\ldots,\\
&LP \vdash \bot \vee \bot \vee \bot \vee \cdots \vee \bot = \bot)
\end{aligned}$$

Das widerspricht (1). Folglich gibt es $1 \leq i \leq n$ mit $LP \not\vdash_{SLD,A} p_i t_i \sigma$, d.h. $h^*(t_i) = fold^A(t_i\sigma) \in p_i^B$ nach Definition von p^B und daher $h \in g_B^*(p_i t_i)$, also auch

$$h \in g_B^*(p_1 t_1 \vee \cdots \vee p_n t_n).$$

Beweis von (iii). Wir zeigen zunächst für alle $C \in Struct_{\Sigma,A}$:

$$C \models LP \;\Leftrightarrow\; F(C) \leq C. \tag{2}$$

"$\Rightarrow$": Sei $p : n \in \Sigma \setminus \Sigma'$ und $a \in p^{F(C)}$. Nach Definition von $p^{F(C)}$ gibt es $pt \Leftarrow \varphi \in LP$ und $h \in A^X$ mit $h^*(t) = a$ und $C \models_h \varphi$. Aus $C \models LP$ folgt $C \models_h pt$, also $a = h^*(t) \in p^C$.

"$\Leftarrow$": Sei $pt \Leftarrow \varphi \in LP$ und $h \in g_C^*(\varphi)$. Nach Definition von $p^{F(C)}$ gilt $h^*(t) \in p^{F(C)}$. Wegen $F(C) \subseteq C$ folgt $h^*(t) \in p^C$, also $C \models_h pt$. Daher gilt $C \models pt \Leftarrow \varphi$.

Damit ist (2) gezeigt. Daraus folgt insbesondere $lfp(F) \models LP$. Ferner erhalten wir für alle $p : n \in \Sigma \setminus \Sigma'$ und $t \in T_\Sigma^n$,

$$p^{lfp(F)} \overset{Satz\ 4.1(i)}{=} \bigcap\{p^C \mid C \in Struct_{\Sigma,A},\ F(C) \leq C\} \overset{(2),(i)}{\subseteq} p^B$$

sowie

$$fold^A(t) \in p^B \Leftrightarrow LP \vdash_{SLD,A} pt \Rightarrow LP \models_A pt \overset{lfp(F)\models LP}{\Rightarrow} lfp(F) \models pt \Leftrightarrow fold^A(t) \in p^{lfp(F)}$$

wegen der Korrektheit des *SLD*-Kalküls für (LP, A), also $lfp(F) = B$.

Sei $a \in p^{F(F^\infty)}$. Dann gibt es $pt \Leftarrow \bigwedge_{i=1}^k p_i t_i \in LP$ und $h \in A^X$ mit $h^*(t) = a$ und $F^\infty \models_h p_i t_i$, also $h^*(t_i) \in p_i^{F^\infty} = p_i^{\bigsqcup_{j\in\mathbb{N}} F^j(0)} = \bigcup_{j\in\mathbb{N}} p_i^{F^j(0)}$ für alle $1 \leq i \leq k$. Daher existieren $n_1, \ldots, n_k \in \mathbb{N}$ mit $h^*(t_i) \in p_i^{F^{n_i}(0)}$ für alle $1 \leq i \leq k$. Sei $m = max\{n_1, \ldots, n_k\}$. Wegen $F^{n_i}(0) \leq F^m(0)$ für alle $1 \leq i \leq k$ folgt $h^*(t_i) \in p_i^{F^m(0)}$ für alle $1 \leq i \leq k$, also

$$F^m(0) \;\models_h \bigwedge_{i=1}^k p_i t_i.$$

Demnach gilt $a = h^*(t) \in p^{F(F^m(0))} = p^{F^{m+1}(0)} \subseteq p^{F^\infty}$. Damit ist $p^{F(F^\infty)} \subseteq p^{F^\infty}$ gezeigt.

Daraus folgt $F(F^\infty) \leq F^\infty$, d.h. F^∞ ist F-abgeschlossen.

Beweis von (iv). Wir zeigen zunächst für alle $C \in Struct_{\Sigma,A}$:

$$C \models LP \;\Leftrightarrow\; C \leq F(C). \tag{3}$$

"$\Rightarrow$": Sei $p : n \in \Sigma \setminus \Sigma'$ und $a \in A^n \setminus p^{F(C)}$. Nach Definition von $p^{F(C)}$ gibt es $pt \Rightarrow \varphi \in LP$ und $h \in A^X$ mit $h^*(t) = a$ und $C \not\models_h \varphi$. Aus $C \models LP$ folgt $C \not\models_h pt$, also $a = h^*(t) \notin p^C$.

"$\Leftarrow$": Sei $pt \Rightarrow \varphi \in LP$ und $C \not\models_h \varphi$. Nach Definition von $p^{F(C)}$ gilt $h^*(t) \notin p^{F(C)}$, also $C \not\models_h pt$. Daher gilt $C \models pt \Rightarrow \varphi$.

Damit ist (3) gezeigt. Daraus folgt insbesondere $gfp(F) \models LP$. Ferner erhalten wir für alle $p : n \in \Sigma \setminus \Sigma'$ und $t \in T_\Sigma^n$,

$$p^B \overset{(3),(i)}{\subseteq} \bigcup\{p^C \mid C \in Struct_{\Sigma,A},\ C \leq F(C)\} \overset{Satz\ 4.1(ii)}{=} p^{gfp(F)}$$

sowie

$$\begin{aligned} &fold^A(t) \notin p^B \Leftrightarrow LP \vdash_{SLD,A} pt \Rightarrow LP \models_A \neg pt \overset{gfp(F)\models LP}{\Rightarrow} gfp(F) \models \neg pt \\ &\Leftrightarrow fold^A(t) \notin p^{gfp(F)} \end{aligned}$$

wegen der Korrektheit des *coSLD*-Kalküls für (LP, A), also $gfp(F) = B$.

Sei $a \in A^n \setminus p^{F(F_\infty)}$. Dann gibt es $pt \Rightarrow \bigvee_{i=1}^k p_i t_i \in LP$ und $h \in A^X$ mit $h^*(t) = a$ und $F_\infty \not\models_h p_i t_i$, also $h^*(t_i) \notin p_i^{F_\infty} = p_i^{\bigsqcap_{j\in\mathbb{N}} F^j(1)} = \bigcap_{j\in\mathbb{N}} p_i^{F^j(1)}$ für alle $1 \leq i \leq k$. Daher existieren $n_1, \ldots, n_k \in \mathbb{N}$ mit $h^*(t_i) \notin p_i^{F^{n_i}(0)}$ für alle

$1 \leq i \leq k$. Sei $m = max\{n_1, \ldots, n_k\}$. Wegen $F^m(1) \leq F^{n_i}(1)$ für alle $1 \leq i \leq k$ folgt $h^*(t_i) \notin p_i^{F^m(1)}$ für alle $1 \leq i \leq k$, also

$$F^m(1) \not\models_h \bigvee_{i=1}^{k} p_i t_i.$$

Demnach gilt $a = h^*(t) \notin p^{F(F^m(1))} = p^{F^{m+1}(1)}$, also $a \notin p^{F_\infty}$ wegen $p^{F_\infty} \subseteq p^{F^{m+1}(1)}$. Damit ist $p^{F_\infty} \subseteq p^{F(F_\infty)}$ gezeigt.

Daraus folgt $F_\infty \leq F(F_\infty)$, d.h. F_∞ ist F-dicht. ❑

Kongruenzaxiome sind Hornformeln, Disgruenzaxiome Cohornformeln (siehe Kapitel 8). Die einen erhält man durch Kontraposition aus den anderen. Analog lassen sich aus den Horn- bzw. Cohornformeln von LP durch Kontraposition Cohorn- bzw. Hornformeln gewinnen, die $Ext(A, LP)$ um die Komplemente der Interpretationen der Prädikate von $\Sigma \setminus \Sigma'$ erweitern:

Sei LP ein (co)logisches Programm für $P = \Sigma \setminus \Sigma'$ und $\overline{P} = \{\overline{p} : n \mid p : n \in P\}$. Das (co)logische Programm

$$\overline{LP} = \begin{cases} \{\overline{p}(t) \Rightarrow \overline{p_1}t_1 \vee \cdots \vee \overline{p_n}t_n \mid pt \Leftarrow p_1t_1 \wedge \cdots \wedge p_nt_n \in LP\} \cup \\ \{\overline{p}(t) \Rightarrow \bot \mid pt \Leftarrow \top \in LP\} & \text{falls } LP \text{ logisch ist,} \\ \{\overline{p}(t) \Leftarrow \overline{p_1}t_1 \wedge \cdots \wedge \overline{p_n}t_n \mid pt \Rightarrow p_1t_1 \vee \cdots \vee p_nt_n \in LP\} \cup \\ \{\overline{p}(t) \Leftarrow \top \mid pt \Rightarrow \bot \in LP\} & \text{falls } LP \text{ cologisch ist,} \end{cases}$$

nennen wir **Kontraposition** von LP.

Satz 9.7

Sei $B = Ext(A, LP)$ und $\overline{B} = Ext(A, \overline{LP})$. Für alle $p \in \Sigma$ ist $\overline{p}^{\overline{B}}$ das Komplement von p^B, d.h.

$$\overline{p}^{\overline{B}} = A^n \setminus p^B. \tag{1}$$

Beweis. Durch Induktion über die Länge von *SLD*-Ableitungen lässt sich zeigen, dass für alle Prädikate $p : n \in \Sigma$ und $t \in T_\Sigma^n$ Folgendes gilt:

$$LP \vdash_{SLD,A} pt \quad \Leftrightarrow \quad \overline{LP} \vdash_{SLD,A} \overline{p}t. \tag{2}$$

Sei LP logisch. Dann ist $\overline{LP}$ cologisch und nach Satz 9.6

$$\begin{aligned} &\overline{p}^{\overline{B}} = \{fold^A(t) \mid t \in T_\Sigma^n,\ \overline{LP} \nvdash_{SLD,A} \overline{p}t\} \overset{(1)}{=} \{fold^A(t) \mid t \in T_\Sigma^n,\ LP \nvdash_{SLD,A} pt\} \\ &= A^n \setminus \{fold^A(t) \mid t \in T_\Sigma^n,\ LP \vdash_{SLD,A} pt\} = A^n \setminus p^B. \end{aligned}$$

Sei LP cologisch. Dann ist $\overline{LP}$ logisch und nach Satz 9.6

$$\begin{aligned} &\overline{p}^{\overline{B}} = \{fold^A(t) \mid t \in T_\Sigma^n,\ \overline{LP} \vdash_{SLD,A} \overline{p}t\} \overset{(2)}{=} \{fold^A(t) \mid t \in T_\Sigma^n,\ LP \vdash_{SLD,A} pt\} \\ &= A^n \setminus \{fold^A(t) \mid t \in T_\Sigma^n,\ LP \nvdash_{SLD,A} pt\} = A^n \setminus p^B. \end{aligned}$$

Der folgende alternative Beweis von (1) verwendet anstelle von (2) die Sätze 3.2 (2/3) und 9.6 (iii/iv).

Sei $\overline{\Sigma} = \Sigma \setminus \Sigma' \cup \overline{P}$. Wir zeigen zunächst durch Induktion über i, dass für $F = F_{\Sigma,A}$, $\overline{F} = F_{\overline{\Sigma},A}$ (siehe Satz 9.6) und alle $i \in \mathbb{N}$ Folgendes gilt:

$$\overline{p}^{\overline{F}^i(1)} = A^n \setminus p^{F^i(0)}. \tag{3}$$

Beweis von (3). Sei LP logisch. Dann gilt $\overline{p}^{\overline{F}^0(1)} = \overline{p}^1 = A^n = A^n \setminus \emptyset = A^n \setminus p^0$ und nach Induktionsvoraussetzung

für alle $i > 0$:

$$\begin{aligned}
&\overline{p}^{\overline{F}^i(1)} = \overline{p}^{\overline{F}(\overline{F}^{i-1}(1))} \\
&= \{a \in A^n \mid \forall\, \overline{p}t \Rightarrow \bigvee_{j=1}^k \overline{p_j}t_j \in \overline{LP},\ h \in A^X : h^*(t) \neq a \vee h \in g^*_{\overline{F}^{i-1}(1)}(\bigvee_{j=1}^k \overline{p_j}t_j)\} \\
&= \{a \in A^n \mid \forall\, \overline{p}t \Rightarrow \bigvee_{j=1}^k \overline{p_j}t_j \in \overline{LP},\ h \in A^X : \\
&\qquad\qquad h^*(t) \neq a \vee \exists\, 1 \leq j \leq k : h^*(t_j) \in \overline{p_j}^{\overline{F}^{i-1}(1)}\} \\
&\overset{\text{ind. hyp.}}{=} \{a \in A^n \mid \forall\, \overline{p}t \Rightarrow \bigvee_{j=1}^k \overline{p_j}t_j \in \overline{LP},\ h \in A^X : \\
&\qquad\qquad h^*(t) \neq a \vee \exists\, 1 \leq j \leq k : h^*(t_j) \in A^n \setminus p_j^{F^{i-1}(0)}\} \\
&= A^n \setminus \{a \in A^n \mid \exists\, pt \Leftarrow \bigwedge_{j=1}^k p_j t_j \in LP,\ h \in A^X : \\
&\qquad\qquad h^*(t) = a \wedge \forall\, 1 \leq j \leq k : h^*(t_j) \in p_j^{F^{i-1}(0)}\} \\
&= A^n \setminus p^{F(F^{i-1}(0))} = A^n \setminus p^{F^i(0)}.
\end{aligned}$$

Ist LP cologisch, dann folgt (3) analog.

Sei LP logisch. Da F^∞ F-abgeschlossen und $\overline{F}_\infty$ $\overline{F}$-dicht ist, erhält man (1) aus (3) wie folgt:

$$\begin{aligned}
&\overline{p}^{\overline{B}} \overset{\text{Satz } 9.6(iv)}{=} \overline{p}^{\overline{F}_\infty} = \overline{p}^{\bigsqcap_{i\in\mathbb{N}} \overline{F}^i(1)} = \bigcap_{i\in\mathbb{N}} \overline{p}^{\overline{F}^i(1)} \overset{(3)}{=} \bigcap_{i\in\mathbb{N}} (A^n \setminus p^{F^i(0)}) \\
&\overset{\text{Satz } 3.2(2)}{=} A^n \setminus \bigcup_{i\in\mathbb{N}} (p^{F^i(0)}) = A^n \setminus p^{\bigsqcup_{i\in\mathbb{N}} F^i(0)} = A^n \setminus p^{F^\infty} \overset{\text{Satz } 9.6(iii)}{=} A^n \setminus p^B.
\end{aligned}$$

Ist LP cologisch, dann folgt (3) analog. ❑

Die (Fixpunkt-)Semantik logischer Programme funktioniert offenbar immer dann, wenn die Menge aller Prädikate von Σ so in Teilmengen $P_1, \ldots P_n$ zerlegt und ein Tupel $LP = (LP_1, \ldots, LP_n)$ logischer Programme gebildet werden kann, dass für alle $1 \leq i \leq n$ LP_i zwar nur die Prädikate von P_i "definiert", aber dazu auch Prädikate von $P_1, \ldots P_{i-1}$ verwendet. In diesem Fall nennt man LP ein **stratifiziertes Programm**. Damit arbeitet z.B. die Antwortmengenprogrammierung (siehe z.B. [2], Kapitel 9).

Die *cologischen* Komponenten von LP sind dort stets Kontrapositionen logischer Programme. Sie tauchen zwar explizit nicht auf, bestimmen aber implizit die Semantik der dort in Hornformeln erlaubten *negierten* Atome. Ein anderes Beispiel für die Verwendung von Basisprädikaten ist die constraint-logische Programmierung: Σ' ist dort z.B. eine Signatur von Operationen und Prädikaten auf reellen Zahlen. Die Constraints entsprechen Σ'-Atomen, die – analog zu den obigen Basisregeln – in einer festen Σ'-Struktur mit Trägermenge $\mathbb{R}$ ausgewertet werden.

Laut Satz 9.6 (iii) und (iv) ist das "Standardmodell" $Ext(A, LP)$ eines (co)logischen Programms LP der kleinste bzw. größte Fixpunkt einer Schrittfunktion F auf $Struct_{\Sigma,A}$. Eigenschaften von $Ext(A, LP)$ werden deshalb gemäß Satz 4.1 (iii/iv) mit Hilfe von F-Induktion bzw. -Coinduktion bewiesen, die als spezielle Ableitungsregeln in einen erweiterten (co)SLD-Kalkül eingebaut werden können, der beliebige prädikatenlogische Formeln – auch über mehrsortigen (getypten) Signaturen – verarbeitet (siehe [34], Kapitel 17).

Dieser Kalkül enthält zur Anwendung von Gleichungen (siehe Abschnitt 5.3) auch Varianten der folgenden Regeln, deren Benutzung (die i.d.R. aufwändigen) Resolutionen mit Kongruenzaxiomen überflüssig machen (siehe [26], Theorem 5.3.4; [27], Theorem 4.4):

Paramodulation über E

$$\frac{LP \vdash \vartheta\{u/x\} \wedge \psi}{E \vdash \vartheta\{t'/x\}\sigma \wedge \varphi\sigma \wedge \psi\sigma} \qquad \begin{array}{l} \vartheta \in At_\Sigma(X),\ u \in T_\Sigma(X),\ \psi \in G_\Sigma(X),\ t \equiv t' \Leftarrow \varphi \in LP', \\ \sigma = unify(t, u),\ var(t \equiv t' \Leftarrow \varphi) \parallel var(\vartheta \wedge \psi) \end{array}$$

Unifikationsregel

$$\frac{LP \vdash t \equiv t' \wedge \varphi}{LP \vdash \varphi\sigma \wedge \bigwedge\{x \equiv x\sigma \mid x \in supp(\sigma)\}} \qquad t, t' \in T_\Sigma(X),\ \varphi \in G_\Sigma(X),\ \sigma = unify(t, t')$$

9.4 Lösungskalküle für logische Programme

Wir erweitern die Sukzedenten der Regeln des $(co)SLD$-Kalküls um eine Konjunktion von Gleichungen $x \equiv x\sigma$ bzw. eine Disjunktion von Ungleichungen $x \not\equiv x\sigma$, wobei σ der mgu der jeweiligen Regelanwendung ist.

Sei A eine erreichbare Σ'-Struktur mit Komplementen (siehe Abschnitt 5.1), LP ein (co)logisches Σ-Programm und LP' die Menge aller (Co)Hornformeln φ', für die es $\varphi \in LP$ und eine Variablenumbenennung τ mit $\varphi\tau = \varphi'$ gibt. Σ' enthalte $\equiv$ und $\not\equiv$.

Sei LP logisch. Der **SLDL-Kalkül für** (LP, A) ist analytisch und besteht aus drei Regeln:

Faktorregel

$$\frac{LP \vdash \vartheta \wedge \vartheta' \wedge \varphi}{LP \vdash \vartheta\sigma \wedge \varphi\sigma \wedge \bigwedge\{x \equiv x\sigma \mid x \in supp(\sigma)\}} \qquad \begin{array}{l}\vartheta, \vartheta' \in At_\Sigma(X),\ \varphi \in G_\Sigma(X),\\ \sigma = unify(\vartheta, \vartheta')\end{array}$$

Resolution über LP

$$\frac{LP \vdash \vartheta' \wedge \psi}{LP \vdash \varphi\sigma \wedge \psi\sigma \wedge \bigwedge\{x \equiv x\sigma \mid x \in supp(\sigma)\}} \qquad \begin{array}{l}\vartheta' \in At_\Sigma(X),\ \psi \in G_\Sigma(X),\\ \vartheta \Leftarrow \varphi \in LP',\ \sigma = unify(\vartheta, \vartheta'),\\ var(\vartheta \Leftarrow \varphi) \parallel var(\vartheta' \wedge \psi)\end{array}$$

Stopregel

$$\frac{LP \vdash \varphi}{} \qquad \varphi \in G_{\Sigma'}(X)$$

Sei LP cologisch. Der **coSLDL-Kalkül für** (LP, A) ist analytisch und besteht aus drei Regeln:

Summandregel

$$\frac{LP \vdash \vartheta \vee \vartheta' \vee \varphi}{LP \vdash \vartheta\sigma \vee \varphi\sigma \vee \bigvee\{x \not\equiv x\sigma \mid x \in supp(\sigma)\}} \qquad \begin{array}{l}\vartheta, \vartheta' \in At_\Sigma(X),\ \varphi \in coG_\Sigma(X),\\ \sigma = unify(\vartheta, \vartheta')\end{array}$$

Coresolution über LP

$$\frac{LP \vdash \vartheta' \vee \psi}{LP \vdash \varphi\sigma \vee \psi\sigma \vee \bigvee\{x \not\equiv x\sigma \mid x \in supp(\sigma)\}} \qquad \begin{array}{l}\vartheta' \in At_\Sigma(X),\ \psi \in coG_\Sigma(X),\\ \vartheta \Rightarrow \varphi \in LP',\ \sigma = unify(\vartheta, \vartheta'),\\ var(\vartheta \Leftarrow \varphi) \parallel var(\vartheta' \wedge \psi)\end{array}$$

Stopregel

$$\frac{LP \vdash \varphi}{} \qquad \varphi \in coG_{\Sigma'}(X)$$

Goals der Form $\varphi \wedge \bigwedge_{i=1}^n x_i \equiv t_i$ und Cogoals der Form $\varphi \vee \bigvee_{i=1}^n x_i \not\equiv t_i$ heißen **gelöst**, wenn φ ein Σ'-(Co)Goal ist und $x_1, \ldots, x_n$ paarweise verschiedene Variablen sind, die in $t_1, \ldots, t_n$ nicht vorkommen.

Haben alle in einer $(co)SLDL$-Ableitung angewendeten (Co)Hornformeln von LP' paarweise disjunkte Variablenmengen, dann sind die aus dem jeweiligen mgu gebildeten Konjunktionen von Gleichungen bzw. Disjunktionen von Ungleichungen gelöst.

Satz 9.8 (Korrektheit von $(co)SLDL$-Ableitungen)

Sei LP ein logisches Σ-Programm mit Basisprädikaten und $(LP \vdash \varphi_1, \ldots, LP \vdash \varphi_n)$ eine $SLDL$-Ableitung.

(i) $LP \models_A \varphi_n \Rightarrow \varphi_1$.

(ii) Sei ψ ein Σ'-Goal, $\varphi_n = \psi \wedge \bigwedge_{i=1}^k x_i \equiv t_i$ gelöst und $\sigma = \{t_1/x_1, \ldots, t_k/x_k\}$.
Dann gilt $LP \models_A \psi\sigma \Rightarrow \varphi_1\sigma$.

Sei LP ein cologisches Σ-Programm mit Basisprädikaten und $(LP \vdash \varphi_1, \ldots, LP \vdash \varphi_n)$ eine *coSLDL*-Ableitung.

(iii) $LP \models_A \varphi_n \Rightarrow \varphi_1$.

(iv) Sei ψ ein Σ'-Cogoal, $\varphi_n = \psi \vee \bigvee_{i=1}^{k} x_i \not\equiv t_i$ gelöst und $\sigma = \{t_1/x_1, \ldots, t_k/x_k\}$.
Dann gilt $LP \models_A \varphi_1\sigma \Rightarrow \psi\sigma$.

Beweis von (i). Wegen

$$\begin{aligned} &LP \models_A \varphi_n \Rightarrow \varphi_1 \quad \Leftrightarrow \quad \forall\, B \in Struct_{\Sigma,A,LP} : B \models \varphi_n \Rightarrow \varphi_1 \\ &\Leftrightarrow \quad \forall\, B \in Struct_{\Sigma,A,LP} : g_B^*(\varphi_n) \subseteq g_B^*(\varphi_1) \end{aligned}$$

genügt es zu zeigen, dass für alle $B \in Struct_{\Sigma,A,LP}$ und $1 \leq i < n$ Folgendes gilt:

$$g_B^*(\varphi_{i+1}) \subseteq g_B^*(\varphi_i). \tag{1}$$

Fall 1: Der Ableitungsschritt von $LP \vdash \varphi_i$ nach $LP \vdash \varphi_{i+1}$ ist eine Anwendung der Faktorregel, d.h.

$$\varphi_i = \vartheta \wedge \vartheta' \wedge \varphi \quad \text{und} \quad \varphi_{i+1} = \vartheta\sigma \wedge \varphi\sigma \wedge \bigwedge\{x \equiv x\sigma \mid x \in supp(\sigma)\},$$

wobei $\vartheta, \vartheta', \varphi, \sigma$ die Anwendbarkeitsbedingungen der Faktorregel erfüllen.

Beweis von (1). Sei $h \in g_B^*(\varphi_{i+1})$, also

$$\begin{aligned} h \quad \in \quad & g_B^*(\vartheta\sigma) \cap g_B^*(\varphi\sigma) \cap \bigcap\{g_B(x \equiv x\sigma) \mid x \in supp(\sigma)\} \\ & = g_B^*(\vartheta\sigma) \cap g_B^*(\varphi\sigma) \cap \bigcap\{\{h \in A^X \mid h(x) = h^*(x\sigma)\} \mid x \in supp(\sigma)\}. \end{aligned} \tag{2}$$

Wegen $\vartheta\sigma = \vartheta'\sigma$ folgt

$$h^* \circ \sigma \in g_B^*(\vartheta) \cap g_B^*(\vartheta') \cap g_B^*(\varphi) \tag{3}$$

aus (2) und Lemma 8.3.

(2) impliziert außerdem $h(x) = h^*(x\sigma)$ für alle $x \in supp(\sigma)$, also $h = h^* \circ \sigma$. Deshalb folgt

$$h \in g_B^*(\vartheta) \cap g_B^*(\vartheta') \cap g_B^*(\varphi) = g_B^*(\vartheta \wedge \vartheta' \wedge \varphi) = g_B^*(\varphi_i)$$

aus (3).

Fall 2: Der Ableitungsschritt von $LP \vdash \varphi_i$ nach $LP \vdash \varphi_{i+1}$ ist eine Anwendung der Resolutionsregel, d.h.

$$\varphi_i = \vartheta' \wedge \psi \quad \text{und} \quad \varphi_{i+1} = \varphi\sigma \wedge \psi\sigma \wedge \bigwedge\{x \equiv x\sigma \mid x \in supp(\sigma)\},$$

wobei $\vartheta, \vartheta', \varphi, \psi, \sigma$ die Anwendbarkeitsbedingungen der Resolutionsregel erfüllen.

Beweis von (1). Sei $h \in g_B^*(\varphi_{i+1})$, also

$$\begin{aligned} h \quad \in \quad & g_B^*(\varphi\sigma) \cap g_B^*(\psi\sigma) \cap \bigcap\{g_B(x \equiv x\sigma) \mid x \in supp(\sigma)\} \\ & = g_B^*(\varphi\sigma) \cap g_B^*(\psi\sigma) \cap \bigcap\{\{h \in A^X \mid h(x) = h^*(x\sigma)\} \mid x \in supp(\sigma)\}. \end{aligned} \tag{4}$$

nach Lemma 8.3 folgt $h^* \circ \sigma \in g_B^*(\varphi) \cap g_B^*(\psi)$ aus (4), also

$$h^* \circ \sigma \in g_B^*(\vartheta') \cap g_B^*(\psi) \tag{5}$$

wegen $A \models \vartheta \Leftarrow \varphi$ und $\vartheta\sigma = \vartheta'\sigma$.

(4) impliziert außerdem $h(x) = h^*(x\sigma)$ für alle $x \in supp(\sigma)$, also $h = h^* \circ \sigma$. Deshalb folgt

$$h \in g_B^*(\vartheta') \cap g_B^*(\psi) = g_B^*(\vartheta' \wedge \psi) = g_B^*(\varphi_i)$$

aus (5).

Beweis von (ii). Da φ_n gelöst ist, gilt

$$g_B^*(\varphi_n\sigma) = g_B^*(\psi\sigma \wedge \bigwedge_{i=1}^{k} x_i\sigma \equiv t_i\sigma) = g_B^*(\psi\sigma \wedge \bigwedge_{i=1}^{k} t_i \equiv t_i) = g_B^*(\psi\sigma). \tag{6}$$

Sei $h \in g_B^*(\psi\sigma)$. (6) impliziert $h \in g_B^*(\varphi_n\sigma)$, also $h^* \circ \sigma \in g_B^*(\varphi_n) \overset{(i)}{\subseteq} g_B^*(\varphi_1)$ nach Lemma 8.3. Daraus folgt $h \in g_B^*(\varphi_1\sigma)$, wieder nach Lemma 8.3.

Beweis von (iii). Wegen

$$\begin{aligned} &LP \models_A \varphi_1 \Rightarrow \varphi_n \;\Leftrightarrow\; \forall\, B \in Struct_{\Sigma,A,LP} : B \models \varphi_1 \Rightarrow \varphi_n \\ &\Leftrightarrow\; \forall\, B \in Struct_{\Sigma,A,LP} : g_B^*(\varphi_1) \subseteq g_B^*(\varphi_n) \end{aligned}$$

genügt es zu zeigen, dass für alle $B \in Struct_{\Sigma,A,LP}$ und $1 \leq i < n$

$$g_B^*(\varphi_i) \subseteq g_B^*(\varphi_{i+1}) \tag{7}$$

gilt.

Fall 1: Der Ableitungsschritt von $LP \vdash \varphi_i$ nach $LP \vdash \varphi_{i+1}$ ist eine Anwendung der Summandregel, d.h.

$$\varphi_i = \vartheta \vee \vartheta' \vee \varphi \quad \text{und} \quad \varphi_{i+1} = \vartheta\sigma \vee \varphi\sigma \vee \bigvee\{x \not\equiv x\sigma \mid x \in supp(\sigma)\},$$

wobei $\vartheta, \vartheta', \varphi, \sigma$ die Anwendbarkeitsbedingungen der Summandregel erfüllen.

Beweis von (7). Sei $h \in g_B^*(\varphi_i)$, also

$$h \in g_B^*(\vartheta) \cup g_B^*(\vartheta') \cup g_B^*(\varphi). \tag{8}$$

Fall 1.1: $h = h^* \circ \sigma$. Dann gilt $h^* \circ \sigma \in g_B^*(\vartheta) \cup g_B^*(\vartheta') \cup g_B^*(\varphi)$ wegen (8), also

$$h \in g_B^*(\vartheta\sigma) \cup g_B^*(\varphi\sigma) \tag{9}$$

wegen Lemma 8.3 und $\vartheta\sigma = \vartheta'\sigma$.

Fall 1.2: Es gibt $x \in supp(\sigma)$ mit $h(x) = h^*(x\sigma)$. Also gilt

$$\begin{aligned} h \;\in\; & \bigcup\{\{h \in A^X \mid h(x) \neq h^*(x\sigma)\} \mid x \in supp(\sigma)\} \\ & = \bigcup\{g_B(x \not\equiv x\sigma) \mid x \in supp(\sigma)\} = g_B^*(\bigvee\{x \not\equiv x\sigma \mid x \in supp(\sigma)\}). \end{aligned} \tag{10}$$

In beiden Fällen folgt

$$\begin{aligned} h \;\in\; & g_B^*(\vartheta\sigma \vee \varphi\sigma) \cup g_B^*(\bigvee\{x \not\equiv x\sigma \mid x \in supp(\sigma)\}) \\ & g_B^*(\vartheta\sigma \vee \varphi\sigma \vee \bigvee\{x \not\equiv x\sigma \mid x \in supp(\sigma)\}) = g_B^*(\varphi_{i+1}) \end{aligned}$$

aus (9) bzw. (10).

Fall 2: Der Ableitungsschritt von $LP \vdash \varphi_i$ nach $LP \vdash \varphi_{i+1}$ ist eine Anwendung der Coresolutionsregel, d.h.

$$\varphi_i = (\vartheta' \wedge \psi) \quad \text{und} \quad \varphi_{i+1} = \varphi\sigma \vee \psi\sigma \vee \bigvee\{x \not\equiv x\sigma \mid x \in supp(\sigma)\},$$

wobei $\vartheta, \vartheta', \varphi, \psi, \sigma$ die Anwendbarkeitsbedingungen der Summandregel erfüllen.

Beweis von (7). Sei $h \in g_B^*(\varphi_i)$, also

$$h \in g_B^*(\vartheta') \cup g_B^*(\psi). \tag{11}$$

Fall 2.1: $h = h^* \circ \sigma$. Dann gilt $h^* \circ \sigma \in g_B^*(\vartheta') \cup g_B^*(\varphi)$ wegen (11), also

$$h^* \circ \sigma \in g_B^*(\varphi) \cup g_B^*(\psi)$$

wegen $\vartheta'\sigma = \vartheta\sigma$ und $A \models \vartheta \Rightarrow \varphi$. Daraus folgt

$$h \in g_B^*(\varphi\sigma) \cup g_B^*(\psi\sigma) \tag{12}$$

nach Lemma 8.3.

Fall 2.2: Es gibt $x \in supp(\sigma)$ mit $h(x) = h^*(x\sigma)$. Also gilt (10) wie in Fall 1.2.

In beiden Fällen folgt

$$\begin{array}{rcl} h & \in & g_B^*(\varphi\sigma \vee \psi\sigma) \cup g_B^*(\bigvee\{x \not\equiv x\sigma \mid x \in supp(\sigma)\}) \\ & & g_B^*(\varphi\sigma \vee \psi\sigma \vee \bigvee\{x \not\equiv x\sigma \mid x \in supp(\sigma)\}) = g_B^*(\varphi_{i+1}) \end{array}$$

aus (12) bzw. (10).

Beweis von (iv). Da φ_n gelöst ist, gilt

$$g_B^*(\varphi_n\sigma) = g_B^*(\psi\sigma \vee \bigvee_{i=1}^{k} x_i\sigma \not\equiv t_i\sigma) = g_B^*(\psi\sigma \vee \bigvee_{i=1}^{k} t_i \not\equiv t_i) = g_B^*(\psi\sigma). \tag{13}$$

Sei $h \in A^X \setminus g_B^*(\psi\sigma)$. (13) impliziert $h \in A^X \setminus g_B^*(\varphi_n\sigma) = g_B^*(\neg\varphi_n\sigma)$, also

$$h^* \circ \sigma \in g_B^*(\neg\varphi_n) = A^X \setminus g_B^*(\varphi_n) \overset{(iii)}{\subseteq} A^X \setminus g_B^*(\varphi_1) = g_B^*(\neg\varphi_1)$$

nach Lemma 8.3. Daraus folgt $h \in g_B^*(\neg\varphi_1\sigma)$, wieder nach Lemma 8.3. ❑

Satz 9.9 (Lösungsvollständigkeit von $(co)SLDL$-Ableitungen)

(i) Sei LP ein logisches Σ-Programm, φ ein Σ-Goal und $\sigma : X \to T_\Sigma$ mit $LP \models_A \varphi\sigma$. Dann gibt es eine Substitution $\tau \leq \sigma$ und eine *SLDL*-Ableitung

$$(LP \vdash \varphi, \ldots, LP \vdash \psi\tau \wedge \bigwedge_{x \in supp(\tau)} x \equiv x\tau).$$

(ii) Sei LP ein cologisches Σ-Programm, φ ein Σ-Cogoal und $\sigma : X \to T_\Sigma$ mit $LP \models_A \neg\varphi\sigma$. Dann gibt es eine Substitution $\tau \leq \sigma$ und eine *coSLDL*-Ableitung

$$(LP \vdash \varphi, \ldots, LP \vdash \psi\tau \vee \bigvee_{x \in supp(\tau)} x \not\equiv x\tau).$$

(i) und (ii) lassen sich analog zu Satz 9.5 (ii) bzw. (iv) beweisen, indem man die dortigen Beweisschritte auf $(co)SLDL$-Ableitungen überträgt und berücksichtigt, dass eine $(co)SLD$-Ableitung $abl = (LP \vdash \varphi_1, LP \vdash \varphi_2, \ldots, LP \vdash \varphi_n)$ einer $(co)SLDL$-Ableitung

$$(LP \vdash \varphi_1, LP \vdash \varphi_2 \wedge \psi_2 \ldots, LP \vdash \varphi_n \wedge \psi_n)$$

mit gelösten (Co)Goals $\psi_2, \ldots, \psi_n$ entspricht, wobei für alle $2 \leq i \leq n$ ψ_i den mgu repräsentiert, der im i-ten Schritt von abl (von $LP \vdash \varphi_{i-1}$ nach $LP \vdash \varphi_i$) gebildet wird. ❑

Beispielhafte Anwendungen der Lösungskalküle

Die Spezifikation ☞ log4 gibt die logischen Programme von Abschnitt 9.1 so wieder, dass der Beweiseditor ☞ Expander2 den Lösungskalkül auf sie anwenden kann.

Das jeweils ausgewählte Atom eines (Co)Goals wird vollständig **expandiert**, d.h. gleichzeitig mit allen (Co)Hornformeln – auch mittels Paramodulation – geschnitten, deren Konklusion (Prämisse) mit ihm unifizierbar ist. Folglich besteht das jeweilige Redukt aus der Disjunktion (Konjunktion) der Prämissen (Konklusionen) aller angewendeten (Co)Hornformeln. Variablen, die nur im Redukt vorkommen, werden dort existenz- bzw. allquantifiziert. `True` steht dort für $\top$, `False` für $\bot$, `&` für $\wedge$, `|` für $\vee$, `Any` für $\exists$, `All` für $\forall$, `=` für $\equiv$, `=/=` für $\not\equiv$, `0` für $zero$, `1` für $succ(zero)$, `n+1` für $succ(n)$, `[]` für nil, `'in'` für $\in$, `-` für $\setminus$ und `[1..n]` für $enum(n)$.

Programme für diese Operationen bzw. Prädikate sind in ☞ Expander2 eingebaut und tauchen deshalb in ☞ log4 nicht auf. Stattdessen folgen auf jede Expansion Simplifikationsschritte, die Teilausdrücke durch äquivalente Teilausdrücke ersetzen und so das jeweilige Redukt soweit wie möglich vereinfachen.

Beweis von ☞ `sorted[1,3,4]` (siehe Beispiel 9.1)
Widerlegung von ☞ `unsorted[1,3,4]` (siehe Beispiel 9.1)

Berechnung der Lösungen von ☞ `perm([1,3,4],s)` (siehe Beispiel 9.1)
Berechnung der Lösungen von ☞ `actions(3,A,B,C,acts)` (siehe Beispiel 9.2)
Berechnung der Lösungen von ☞ `queens(5,val)` (siehe Beispiel 9.3)

9.5 Highlights

Hornformel: $\varphi \Rightarrow pt$ **Cohornformel**: $pt \Rightarrow \psi$

Ein **(co)logisches Programm** LP besteht aus (Co)Hornformeln.

Goal: $p_1t1 \wedge \cdots \wedge p_nt_n$ **Cogoal**: $p_1t1 \vee \cdots \vee p_nt_n$

Sei Σ' eine Teilsignatur von Σ, die alle Operationen von Σ enthält. Die Prädikate von Σ' nennen wir **Basisprädikate**. Zu jedem $p : n \in \Sigma'$ gehöre auch das **Komplementprädikat** $\overline{p} : n$ zu Σ'. Im Fall $\equiv \in \Sigma'$ sei $\overline{\equiv} = \not\equiv$.

Ein (co)logisches Σ-Programm LP heißt **Programm mit Basisprädikaten**, wenn für alle $pt \Leftarrow \varphi \in LP$ bzw. $pt \Rightarrow \varphi \in LP$ p nicht zu Σ' gehört.

Sei A eine erreichbare **komplementabgeschlossene** Σ'-Gleichheitsstruktur, d.h. für alle $p : n \in \Sigma'$ gilt $\overline{p}^B = A^n \setminus p^B$.

Der (analytische) **(co)SLD-Kalkül für** (LP, A) besteht aus den in Abschnitt 9.2 angegebenen Regeln zur Transformation eines (Co)Goals nach $LP \vdash \top$ bzw. $LP \vdash \bot$.

Korrektheit und Vollständigkeit des (co)SLD-Kalküls:

Sei LP ein logisches Σ-Programm mit Basisprädikaten und φ ein Σ-Goal.

$$\boxed{LP \models_A any(\varphi) \;\Leftrightarrow\; LP \vdash_{SLD,A} \top.}$$

Sei LP ein cologisches Σ-Programm mit Basisprädikaten und φ ein Σ-Cogoal.

$$\boxed{LP \models_A any(\neg\varphi) \;\Leftrightarrow\; LP \vdash_{SLD,A} \bot.}$$

Modelle

$Struct_{\Sigma,A}$ = Klasse aller Σ-Strukturen B mit $B|_{\Sigma'} = A$

Für alle $B, C \in Struct_{\Sigma,A}$,

$$B \leq C \quad \Leftrightarrow_{def} \quad \text{für alle Prädikate } p \in \Sigma \setminus \Sigma' \text{ gilt } p^B \subseteq p^C.$$

Sei LP ein (co)logisches Σ-Programm mit Basisprädikaten und $Ext(A, LP)$ die Σ-Struktur, die $p : n \in \Sigma$ wie folgt interpretiert:

$$p^{Ext(A,LP)} = \begin{cases} \{fold^A(t) \mid t \in T_\Sigma^n,\ LP \vdash_{SLD,A} pt\} & \text{falls } LP \text{ logisch ist,} \\ \{fold^A(t) \mid t \in T_\Sigma^n,\ LP \not\vdash_{SLD,A} pt\} & \text{falls } LP \text{ cologisch ist.} \end{cases}$$

$Ext(A, LP)$ ist das bzgl. $\leq$ kleinste bzw. größte Modell von LP, das zu $Struct_{\Sigma,A}$ gehört.

Komplemente

Sei LP ein (co)logisches Programm für $P = \Sigma \setminus \Sigma'$ und $\overline{P} = \{\overline{p} : n \mid p : n \in P\}$.

Das (co)logische Programm

$$\overline{LP} = \begin{cases} \{\overline{p}(t) \Rightarrow \overline{p_1}t_1 \vee \cdots \vee \overline{p_n}t_n \mid pt \Leftarrow p_1t_1 \wedge \cdots \wedge p_nt_n \in LP\} \cup \\ \{\overline{p}(t) \Rightarrow \bot \mid pt \Leftarrow \top \in LP\} & \text{falls } LP \text{ logisch ist,} \\ \{\overline{p}(t) \Leftarrow \overline{p_1}t_1 \wedge \cdots \wedge \overline{p_n}t_n \mid pt \Rightarrow p_1t_1 \vee \cdots \vee p_nt_n \in LP\} \cup \\ \{\overline{p}(t) \Leftarrow \top \mid pt \Rightarrow \bot \in LP\} & \text{falls } LP \text{ cologisch ist,} \end{cases}$$

nennen wir **Kontraposition** von LP.

Sei $B = Ext(A, LP)$ und $\overline{B} = Ext(A, \overline{LP})$. Für alle $p \in \Sigma$ gilt

$$\overline{p}^{\overline{B}} = A^n \setminus p^B.$$

Lösungen

Goals der Form $\varphi \wedge \bigwedge_{i=1}^n x_i \equiv t_i$ und Cogoals der Form $\varphi \vee \bigvee_{i=1}^n x_i \not\equiv t_i$ heißen **gelöst**, wenn φ ein Σ'-(Co)Goal ist und $x_1, \ldots, x_n$ paarweise verschiedene Variablen sind, die in $t_1, \ldots, t_n$ nicht vorkommen.

Der (analytische) **(co)SLDL-Kalkül für** (LP, A) besteht aus den in Abschnitt 9.4 angegebenen Regeln zur Transformation eines (Co)Goals in seine – als Gleichungen bzw. Ungleichungen – dargestellten Lösungen.

Korrektheit und Vollständigkeit des (co)SLDL-Kalküls:

Sei LP ein logisches Σ-Programm mit Basisprädikaten, $(LP \vdash \varphi_1, \ldots, LP \vdash \varphi_n)$ eine *SLDL*-Ableitung, ψ ein Σ'-Goal und $\varphi_n = \psi \wedge \bigwedge_{i=1}^k x_i \equiv t_i$ gelöst. Dann gilt

$$LP \models_A \psi\{t_1/x_1, \ldots, t_k/x_k\} \Rightarrow \varphi_1\{t_1/x_1, \ldots, t_k/x_k\}.$$

Gilt umgekehrt $LP \models_A \varphi\sigma$ für ein $\sigma : X \to T_\Sigma$, dann gibt es eine Substitution $\tau \leq \sigma$ und eine *SLDL*-Ableitung

$$(LP \vdash \varphi, \ldots, LP \vdash \psi\tau \wedge \bigwedge_{x \in supp(\tau)} x \equiv x\tau).$$

Sei LP ein cologisches Σ-Programm mit Basisprädikaten und $(LP \vdash \varphi_1, \ldots, LP \vdash \varphi_n)$ eine *coSLDL*-Ableitung, ψ ein Σ'-Cogoal und $\varphi_n = \psi \vee \bigvee_{i=1}^k x_i \not\equiv t_i$ gelöst. Dann gilt

$$LP \models_A \varphi_1\{t_1/x_1, \ldots, t_k/x_k\} \Rightarrow \psi\{t_1/x_1, \ldots, t_k/x_k\}.$$

Gilt umgekehrt $LP \models_A \neg\varphi\sigma$, dann gibt es eine Substitution $\tau \leq \sigma$ und eine *SLDL*-Ableitung

$$(LP \vdash \varphi, \ldots, LP \vdash \psi\tau \vee \bigvee_{x \in supp(\tau)} x \not\equiv x\tau).$$

Literatur

[1] Pierre Basieux, Die Architektur der Mathematik: Denken in Strukturen, rororo science 2000

[2] Christoph Beierle, Gabriele Kern-Isberner, Methoden wissensbasierter Systeme, Vieweg+Teubner 2008

[3] Mordechai Ben-Ari, Mathematical Logic for Computer Science, Springer 2001

[4] Johan van Benthem, Exploring Logical Dynamics, CSLI Publications, Stanford University 1996

[5] Patrick Blackburn, Maarten de Rijke, Yde Venema, Modal Logic, Cambridge Tracts in Theoretical Computer Science 53, Cambridge University Press 2001

[6] Jürgen Dassow, Logik für Informatiker, Lehrbuch, Teubner 2005

[7] Hartmut Ehrig et al., Mathematisch-strukturelle Grundlagen der Informatik, Springer 2001

[8] Melvin Fitting, First-Order Logic and Automated Theorem Proving, Springer 1996

[9] Valentin Goranko, Martin Otto, Model Theory of Modal Logic, in: Handbook of Modal Logic 3 (2007) 249-329

[10] Erich Grädel, Mathematische Logik, Vorlesungsskript, RWTH Aachen, Sommersemester 2009

[11] Heinz Peter Gumm, Mona Taheri, Saturated Kripke Structures as Vietoris Coalgebras, arXiv:2202.07786 (2022)

[12] Heinz Peter Gumm, Universelle Coalgebra, in: Th. Ihringer, Allgemeine Algebra, Heldermann Verlag 2003

[13] Matthew Hennessy, Robin Milner, Algebraic laws for nondeterminism and concurrency, Journal ACM 32 (1985) 137-161

[14] Dieter Hofbauer, Ralf-Detlef Kutsche, Grundlagen des maschinellen Beweisens, Vieweg 1991

[15] Michael Huth, Mark Ryan, Logic in Computer Science, Cambridge University Press 2004

[16] Bart Jacobs, Jan Rutten, A Tutorial on (Co)Algebras and (Co)Induction, EATCS Bulletin 62 (1997) 222-259

[17] Uwe Kastens und Hans Kleine Büning, Modellierung: Grundlagen und formale Methoden, Hanser 2018

[18] Martin Kreuzer, Stefan Kühling, Logik für Informatiker, Pearson 2006

[19] Klaus Kriegel, Mathematik für Informatiker II: Analysis, FU Berlin 2009

[20] Alexander Kurz, Coalgebras and Modal Logic, Kurs am CWI Amsterdam, 2001

[21] Zohar Manna, Mathematical Theory of Computation, McGraw-Hill 1974

[22] Christoph Meinel, Martin Mundhenk, Mathematische Grundlagen der Informatik - Mathematisches Denken und Beweisen, Vieweg+Teubner 2011

[23] Faron Moller, Georg Struth, Modelling Computing Systems - Mathematics for Computer Science, Springer 2013

[24] Anil Nerode, Richard A. Shore, Logic for Applications, Springer 1997

[25] Peter Padawitz, Dirk Siefkes, Einführung in die Syntax und Semantik der Prädikatenlogik, Vorlesungsskript, TU Berlin 1981

[26] Peter Padawitz, Computing in Horn Clause Theories, EATCS Monographs on Theoretical Computer Science 16, Springer 1988; Freiexemplare gibt es beim Autor.

[27] Peter Padawitz, Deduction and Declarative Programming, Cambridge Tracts in Theoretical Computer Science 28, Cambridge University Press 1992

[28] P. Padawitz, Swinging Types = Functions + Relations + Transition Systems, Theoretical Computer Science 243 (2000) 93-165

[29] Peter Padawitz, Modellieren und Implementieren in Haskell, Technical Report No. 868, TU Dortmund

[30] Peter Padawitz, Formale Methoden des Systementwurfs, Vorlesungsskript, TU Dortmund 2006

[31] Peter Padawitz, From Modal Logic to (Co)Algebraic Reasoning, Folienskript, TU Dortmund

[32] Peter Padawitz, Übersetzerbau, Folienskript, TU Dortmund 2019

[33] Peter Padawitz, Übersetzerbau, Vorlesungsskript, TU Dortmund 2010

[34] Peter Padawitz, Fixpoints, Categories, and (Co)Algebraic Modeling, Folienskript, TU Dortmund

[35] Michael Richter, Logikkalküle, Teubner 1978

[36] Jan Rutten, Universal Coalgebra: A Theory of Systems, Theoretical Computer Science 249 (2000) 3-80

[37] Uwe Schöning, Logik für Informatiker, Spektrum Akademischer Verlag 2000

[38] Thomas Schwentick, Logik für Informatiker, Vorlesungsfolien, TU Dortmund, Wintersemester 2012/13

[39] Bernhard Steffen, Oliver Rüthing, Malte Isberner, Grundlagen der höheren Informatik - Induktives Vorgehen, Springer 2014

[40] Walter Vogler, Logik für Informatiker, Vorlesungsskript, Universität Augsburg, Wintersemester 2011/12

[41] Hubert Wagner, Logische Systeme der Informatik, Vorlesungsskript, TU Dortmund, Wintersemester 2000/01

Index