Ein Beweissystem für Hennessy-Milner-Logik mit Rekursion

Studienarbeit im Fach Informatik
vorgelegt von

Michael Siegel
geboren am 2.5.1964
in Heide

und

Martin Steffen
geboren am 6.6.1965
in Bad Honnef

Angefertigt am
Lehrstuhl für Informatik VII
Rechnerarchitektur und Verkehrstheorie
Institut für Mathematische Maschinen und Datenverarbeitung
Friedrich-Alexander-Universität Erlangen-Nürnberg

Betreut von
Norbert Götz und Terry Stroup

Begonnen am 1. September 1990
Abgegeben am 1. Juli 1991

Wir versichern, daß wir diese Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt haben und daß die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Erlangen, den 1. Juli 1991

Danksagung

Unser Dank gilt unseren Betreuern Norbert Götz und Terry Stroup für die Hinführung zu dieser interessanten wissenschaftlichen Aufgabe. Wir bedanken uns weiter bei ihnen sowie bei Thomas Amrhein, Martin Hofmann und Uwe Nestmann für sorgfältige Korrekturlesearbeiten.

Inhaltsverzeichnis

Kapitel 1

Einführung und Grundlagen

Hennessy-Milner-Logik mit Rekursion (im weiteren μHML) ist eine sehr ausdrucksstarke temporale Logik, die seit einigen Jahren für die Spezifikation verteilter Systeme eingesetzt wird. Um aus einer μHML-Spezifikation eine nachweislich korrekte Implementierung gemäß der Methode der schrittweisen Verfeinerung entwickeln zu können, benötigt man ein Beweissystem, mit dem man zeigen kann,daß eine verfeinerte Spezifikation die ursprüngliche Spezifikation erfüllt. In unserer Arbeit konstruieren wir ein Gentzen-System für μHML, mit dem man zielgerichtet diese Beweise führen kann und modifizieren es anschließend, um effizienter beweisen zu können.

In diesem Abschnitt präzisieren wir unsere Vorstellung von schrittweiser Verfeinerung und schrittweiser Verifikation und stellen die verteilten Prozeßsysteme vor, die uns als Implementierungen dienen.

1.1 Schrittweise Verfeinerung und Verifikation

Es gibt prinzipiell zwei Vorgehensweisen, um aus einer anfänglichen Spezifikation eine Implementierung zu entwickeln: die *"Ein-Schritt-Methode"* und die *"Methode der Schrittweisen Verfeinerung"*. Bei der Ein-Schritt-Methode konstruiert man die Implementierung in einem Entwicklungsschritt aus der Anforderungsspezifikation. Die zugehörige Ein-Schritt-Verifikation ist dann der direkte Beweis, daß die Implementierung korrekt bezüglich dieser Spezifikation ist.

Bei der Methode der Schrittweisen Verfeinerung ist die Idee, durch viele kleine Implementierungs- und Verifizierungsschritte aus einer abstrakten Anforderungsspezifikation (SP_0) über korrekte Zwischenspezifikationen (SP_i) ein ausführbares Programm (*Impl*) zu erzeugen. Bei diesem Vorgehen legt jeder Verfeinerungsschritt weitere Implementierungsdetails des späteren Programms fest. Der Programmentwurf sieht schematisch dann folgendermaßen aus:

$$SP_0 \rightsquigarrow SP_1 \rightsquigarrow \ldots\ldots \rightsquigarrow SP_n \rightsquigarrow Impl$$

Die *Verfeinerungsrelation* $\rightsquigarrow$ zwischen Spezifikationen definieren wir modelltheoretisch als Modellinklusion. Das bedeutet, eine Spezifikation SP_{i+1} verfeinert die Spezifikation SP_i, wenn die durch $[\![SP_{i+1}]\!]$ bezeichnete Modellmenge von SP_{i+1} eine Teilmenge von $[\![SP_i]\!]$ ist; ein Prozeß erfüllt eine Spezifikation, wenn er in der Modellmenge der Spezifikation liegt.

So, wie man den Implementierungsprozeß hierbei in kleine Stücke zerlegt, unterteilt man auch den Korrektheitsbeweis von *Impl* bezüglich SP_0 in kleine Teilbeweise:

$$SP_0 \Longleftarrow SP_1 \Longleftarrow \ldots\ldots \Longleftarrow SP_n \Longleftarrow Impl$$

Dabei steht $SP_i \Longleftarrow SP_{i+1}$ für den Beweis, daß SP_{i+1} die Spezifikation SP_i verfeinert. Kann man die Korrektheit jedes Verfeinerungsschrittes zeigen, so hat man die Korrektheit von *Impl* bezüglich SP_0 gezeigt, da die Verfeinerungsrelation transitiv ist. Diese Implementierungs- und Verifikations-Methodik liegt im weiteren unserer Arbeit zugrunde. Für das geschilderte Vorgehen benötigt man im wesentlichen drei Dinge:

1. eine Menge von *Modellen*, die die möglichen Implementierungen darstellen,
2. eine formale *Spezifikationssprache* zur Beschreibung von Eigenschaften dieser Modelle und
3. ein *Beweissystem*, mit dem man die Korrektheit der einzelnen Verfeinerungsschritte zeigen kann.

1.2 Prozeßsysteme

Die konkreten Modelle, die Spezifikationssprache und die Beweissysteme, die wir in unserer Arbeit benutzen bzw. konstruieren werden, erläutern wir in den nächsten drei Abschnitten.

Die Modelle

Die Systeme, die wir im Laufe einer Programmentwicklung erzeugen wollen, bestehen aus einer Anzahl von Prozessen, die miteinander und mit ihrer Umwelt kommunizieren. Diese Prozesse bezeichnen wir mit $\mathbf{p}$,$\mathbf{q}$,$\mathbf{r}\ldots$. Durch Ausführen von Aktionen, bezeichnet durch a,b,c...aus einem gegebenen Aktionsalphabet, kann ein Prozeß zu einem neuen Prozeß mit einem anderen Verhalten werden. Solche Übergänge beschreibt man durch eine Übergangsrelation $\mathbf{p} \xrightarrow{a} \mathbf{q}$: der Prozeß $\mathbf{p}$ kann durch Ausführen der Aktion a in den Prozeß $\mathbf{q}$ übergehen. Das Verhalten eines Prozesses ergibt sich somit aus seinen möglichen Übergängen zu anderen Prozessen und dem Verhalten dieser Prozesse.

Die eigentlichen semantischen Modelle sind jetzt Strukturen, die solch ein Prozeßverhalten beschreiben. Wir haben für unsere Arbeit *Transitionssysteme* [Plo81] gewählt. Ein Transitionssystem T=($\mathcal{P}$, $\{\xrightarrow{a}$, a $\in$ Act $\}$) ist dabei gegeben durch eine Zustandsmenge $\mathcal{P}$ und Zustandsübergänge $\xrightarrow{a}$ zwischen den Zuständen. Das Verhalten eines Prozesses wird durch einen Zustand und dem von diesem sog. Anfangszustand aus erreichbaren Transitionssystem modelliert. Bei bekanntem Transitionssystem ist dann das Verhalten des Prozesses eindeutig durch diesen Zustand festgelegt. Deswegen verwenden wir im folgenden diese beiden Begriffe "Prozeß" und "Zustand im Transitionssystem"' synonym und meinen mit letzterem das von diesem Zustand aus erreichbare Transitionssystem. Die Zustandsmenge $\mathcal{P}$ eines gegebenen Transitionssystems entspricht in dieser Sichtweise also gleichzeitig einer Menge von Prozessen.

Je nachdem, an welchen Prozeßeigenschaften man interessiert ist, definiert man eine Verhaltensäquivalenz, die festlegt, wann zwei Prozesse dasselbe Verhalten zeigen. Eine weit verbreitete Äquivalenz ist die *Bisimulations-Äquivalenz* [Par81], die folgendermaßen auf Transitionssystemen definiert ist:

Definition 1.1 (starke Bisimulation) *Gegeben sei ein Aktionsalphabet* Act *sowie zwei Transitionssysteme* $T_1 = (\mathcal{P}_1, \{\xrightarrow{a}, a \in$ Act $\})$ *und* $T_2 = (\mathcal{P}_2, \{\xrightarrow{a}, a \in$ Act $\})$. *Eine binäre Relation* $R \subseteq \mathcal{P}_1 \times \mathcal{P}_2$ *ist eine starke Bisimulation, wenn für alle* (**p**,**q**) $\in$ R *gilt:*

1. *wenn* $\mathbf{p} \xrightarrow{a} \mathbf{p}'$, *dann gibt es ein* $\mathbf{q}$' *mit* $\mathbf{q} \xrightarrow{a} \mathbf{q}'$ *und* ($\mathbf{p}$',$\mathbf{q}$') $\in R$,
2. *wenn* $\mathbf{q} \xrightarrow{a} \mathbf{q}'$, *dann gibt es ein* $\mathbf{p}$' *mit* $\mathbf{p} \xrightarrow{a} \mathbf{p}'$ *und* ($\mathbf{p}$',$\mathbf{q}$') $\in R$.

Zwei Prozesse heißen stark bisimulationsäquivalent*, wenn eine starke Bisimulation R zwischen ihnen existiert.*

Man spricht hier von starker Bisimulation, weil alle Aktionen der Prozesse nach außen hin sichtbar sind. Diese Annahme werden wir stets machen, wenn von Zustandsübergängen die Rede ist.

Somit haben wir jetzt festgelegt, was wir als Implementierungen betrachten wollen. Wie man nun Eigenschaften solcher Transitionssysteme beschreibt, erläutern wir im nächsten Abschnitt.

Der Spezifikationsformalismus

Für die Methode der schrittweisen Verfeinerung benötigen wir als nächstes einen geeigneten Spezifikationsformalismus, um auf abstrakter Ebene Eigenschaften von Transitionssystemen formulieren zu können. Geeignet bedeutet hierbei, daß der Formalismus ein möglichst guter Kompromiß bezüglich der folgenden Kriterien ist:

1. *ausdrucksstark*: der Formalismus sollte möglichst ausdrucksstark sein, um die gewünschten Eigenschaften der Prozesse präzise formulieren zu können,

2. *adäquat und expressiv*: er muß mit der zugrundegelegten Verhaltensäquivalenz zwischen Prozessen verträglich sein; das bedeutet einerseits, daß man keine Eigenschaft formulieren kann, die zwei verhaltensäquivalente Modelle voneinander unterscheidet (Adäquatheit), daß andererseits jedoch stets Eigenschaften formulierbar sind, die zwei nicht-äquivalente Modelle voneinander unterscheiden (Expressivität). Diese Begriffe wurden von Pnueli in [Pnu85] vorgeschlagen.

3. *verifizierbar*: die Verfeinerungsrelation und der Formalismus sollten so gewählt sein, daß man die Verfeinerungsschritte auf ihre Korrektheit überprüfen kann.

Gemäß dieser Kriterien haben wir uns für *Hennessy-Milner-Logik mit Rekursion* entschieden. Diese Logik ist eine Erweiterung der multi-modalen Hennessy-Milner-Logik, die Matthew Hennessy und Robin Milner in [HM85] für eine alternative Charakterisierung der Bisimulation benutzt haben. Sie wird seit einigen Jahren auch für die Spezifikation von verteilten Systemen benutzt.

Durch die Erweiterung um Rekursion wird aus der ausdrucksschwachen modalen Logik HML eine sehr ausdrucksstarke temporale Logik. Die üblichen Modelle temporaler Logiken sind *Kripke-Strukturen*, eine Verallgemeinerung von Transitionssystemen. So kann man einer μHML-Formel sehr natürlich eine Menge von Prozessen (= Zustände in einem Transitionssystem) als formale Semantik zuordnen.

Somit haben wir jetzt auch einen Spezifikationsformalismus für den schrittweisen Programmentwurf. Die genaue Einordnung von μHML entsprechend der drei erwähnten Kriterien erfolgt am Ende des 2. Kapitels, wenn wir die Grundlagen von μHML erörtert haben.

Das Beweissystem

Nachdem wir präzisiert haben, welche Modelle wir entwickeln wollen und von welchen Spezifikationen wir ausgehen, benötigen wir jetzt noch ein Beweissystem, mit dem man die Korrektheit der Verfeinerungsschritte zeigen kann. Der Programmentwurf sieht schematisch so aus:

$$\Gamma_0 \rightsquigarrow \Gamma_1 \rightsquigarrow \ldots\ldots \rightsquigarrow \Gamma_n \rightsquigarrow \mathit{Impl}$$

wobei $\Gamma_i \in \mu\text{HML}$, was modelltheoretisch der folgenden Sequenz entspricht:

$$[\![\, \Gamma_0 \,]\!] \supseteq [\![\, \Gamma_1 \,]\!] \supseteq \ldots\ldots \supseteq [\![\, \Gamma_n \,]\!] \ni \mathit{Impl}$$

Will man nun schrittweise verifizieren, daß eine Implementierung $\mathbf{p}$ die Spezifikation Γ_0 erfüllt, also $\mathbf{p} \in [\![\, \Gamma_0 \,]\!]$, so treten zwei Teilaufgaben auf:

1. die Korrektheitsbeweise der einzelnen Verfeinerungsschritte $\Gamma_i \rightsquigarrow \Gamma_{i+1}$, also die Beweise für die Inklusionsbeziehung der zugehörigen Modellklassen $[\![\Gamma_i]\!] \supseteq [\![\Gamma_{i+1}]\!]$ und

2. der Beweis für den letzten Verfeinerungsschritt $\Gamma_n \rightsquigarrow \mathbf{p}$, also der Nachweis, daß die Implementierung in der feinsten Modellklasse enthalten ist, also $\mathbf{p} \in [\![\Gamma_n]\!]$.

Der zweite Punkt wurde eingehend in den letzten Jahren unter anderem in [Cle90], [Lar88], [SW89] und [Win89] untersucht. Es wurden in diesem Zusammenhang zielgerichtete Beweissysteme konstruiert und implementiert. In Kapitel 3 erläutern wir stellvertretend eines der vorgeschlagenen Beweissysteme, die auf dem Prinzip der *Fixpunktinduktion* basieren. Die grundlegende Idee dieser sogenannten *Model-Checker* benutzen wir in Kapitel 4 für unsere eigentliche Arbeit: ein *zielgerichtetes Beweissystem* für die Verfeinerungsrelation zwischen beliebigen μHML-Spezifikationen. Der einzige uns bekannte Ansatz in der Literatur, der sich mit diesem Punkt beschäftigt, ist ein Hilbert-System für den modalen μ-Kalkül [Koz83], einer Logik, in der μHML enhalten ist.

Unsere Arbeit gliedert sich folgendermaßen: im zweiten Kapitel stellen wir Hennessy-Milner-Logik mit Rekursion vor. Für diese Sprache präsentieren wir im 3.Kapitel den Model-Checker, den wir als Anregung für unsere Arbeit benutzt haben. Den Kern dieser Arbeit erläutern wir in Kapitel 4: ein Beweissystem für die Implikation zwischen Hennessy-Milner-Formeln. Das 5.Kapitel beschäftigt sich mit möglichen Erweiterungen der vorliegenden Arbeit.

Kapitel 2

Hennessy-Milner-Logik mit Rekursion

2.1 Einleitung

In diesem Kapitel stellen wir zunächst die Hennessy-Milner-Logik ohne Rekursion [HM85] vor, die wir mit HML abkürzen. Sie bildet die Grundsprache unserer Spezifikationslogik. Anhand von Beispielen erläutern wir, wie man mit HML Eigenschaften von Prozessen beschreiben kann. Da die Ausdrucksstärke dieser Logik nicht ausreicht, um unendlichesProzeßverhalten zu spezifizieren, erweitert man HML um die Möglichkeit zur *rekursiven Formulierung* von HML -Formeln. Die dabei entstehende ausdrucksstarke temporale Logik bezeichnen wir mit μHML. Die Idee der Erweiterung von Logiken um Fixpunktoperatoren geht auf den μ-Kalkül von Scott und DeBakker zurück [SdB69]. Derartige Kalküle wurden in den siebziger Jahren unter anderem von Hitchcock und Park, DeRoever und DeBakker [HP73] [Roe74] [BR72] untersucht. Seit einigen Jahren werden sie für die Spezifikation verteilter Systeme verwendet. Am häufigsten wird dabei die oben erwähnte HML -Erweiterung μHML benutzt, die eine Unterlogik des modalen μ-Kalküls ist.

2.2 Hennessy-Milner-Logik

Wir wollen das Verhalten eines Prozesses durch einen Zustand in einem Transitionssystem und dem davon aus erreichbaren Transitionssystem beschreiben. Wir beschreiben also das Verhalten eines Prozesses $\mathbf{p}$ durch seine möglichen Übergänge $\mathbf{p} \xrightarrow{a} \mathbf{q}$ zu anderen Prozessen und dem Verhalten dieser Prozesse. Im weiteren nehmen wir eine Menge $\mathcal{P}$ von Prozessen und eine Menge Act von möglichen Aktionen als gegeben an. Wie bereits erwähnt, bezeichnen wir Prozesse mit $\mathbf{p}$,$\mathbf{q}$,$\mathbf{r}$... und Aktionen mit a,b,c... . Das Transitionssystem T = $(\mathcal{P},\{\xrightarrow{a}, \text{a} \in \mathsf{Act}\})$, das das Verhalten der Prozesse aus $\mathcal{P}$ beschreibt, sei gegeben.

Mit einer Spezifikation wählt man nun einige Prozesse aus $\mathcal{P}$ aus, die bestimmte gewünschte Eigenschaften haben. Eine Spezifikation besteht also aus einer Menge von Eigenschaften, und jeder Prozeß, der alle Eigenschaften besitzt, erfüllt die Spezifikation. Als Grundsprache zur Formulierung solcher Eigenschaften benutzen wir die von Hennessy und Milner vorgeschlagene Sprache HML , jedoch ohne expliziten $\neg$-Operator.

Definition 2.1 (HML Syntax) *HML ist die kleinste Menge von Formeln, die gemäß folgender Syntax erzeugt werden:*

$$\varphi \;:=\; \mathrm{tt} \mid \mathrm{ff} \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \langle a \rangle \varphi \mid [a]\varphi \qquad\qquad a \in \mathsf{Act}$$

Die intuitive Bedeutung der Formeln ist folgende: jeder Prozeß hat die Eigenschaft tt, keiner die Eigenschaft ff. Ein Prozeß hat die Eigenschaft $\varphi_1 \wedge \varphi_2$, wenn er sowohl φ_1 als auch φ_2 hat; bei $\varphi_1 \vee \varphi_2$ reicht eine der beiden Eigenschaften. Interessanter sind die Modalitäten: ein Prozeß $\mathbf{p}$ hat die Eigenschaft $\langle a \rangle \varphi$, wenn er durch die Ausführung einer a-Aktion $\mathbf{p} \xrightarrow{a} \mathbf{q}$ in einen Prozeß $\mathbf{q}$ übergehen kann, der die Eigenschaft φ hat. $[a]\varphi$ legt hingegen fest, das $\mathbf{p}$ keinen a-Übergang machen kann, ohne zu einem Prozeß $\mathbf{q}$ zu werden, der die Eigenschaft φ hat. Das heißt insbesondere, daß jeder Prozeß, der überhaupt keinen a-Übergang hat, diese Eigenschaft besitzt. Diese modale Logik wurde ursprünglich mit einem $\neg$-Operator definiert. Wir haben uns für diese Version von HML entschieden, bei der zu jedem Operator sein dualer hinzugenommen wurde (insbesondere $[a] \;=\; \neg\langle a \rangle \neg$), da sie für den Sequenzenkalkül, den wir in Kapitel 4 vorstellen werden, besser geeignet ist. Nach der informellen Erklärung der Bedeutung von HML-Formeln folgt nun deren formale Semantik.

Definition 2.2 (HML Semantik) *Die durch eine Formel $\varphi \in$ HML beschriebene Prozeßmenge ist durch die Abbildung $[\![\ ]\!] : \mathrm{HML} \longrightarrow \mathcal{P}$ wie folgt induktiv definiert:*

$$\begin{array}{rclcrcl}
[\![\, \mathrm{tt} \,]\!] & = & \mathcal{P} & & [\![\, \mathrm{ff} \,]\!] & = & \emptyset \\
[\![\, \varphi_1 \wedge \varphi_2 \,]\!] & = & [\![\, \varphi_1 \,]\!] \cap [\![\, \varphi_2 \,]\!] & & [\![\, \varphi_1 \vee \varphi_2 \,]\!] & = & [\![\, \varphi_1 \,]\!] \cup [\![\, \varphi_2 \,]\!] \\
[\![\, \langle a \rangle \varphi \,]\!] & = & \overline{\langle a \rangle}\, [\![\, \varphi \,]\!] & & [\![\, [a]\varphi \,]\!] & = & \overline{[a]}\, [\![\, \varphi \,]\!]
\end{array}$$

Die in der Literatur als agent transformer *bezeichneten Operatoren $\overline{\langle a \rangle}$ und $\overline{[a]}$ sind für $Q \subseteq \mathcal{P}$ folgendermaßen definiert:*

$$\begin{array}{rcl}
\overline{\langle a \rangle}\, Q & := & \{\mathbf{p} \in \mathcal{P} \mid \exists \mathbf{p}'.\, \mathbf{p} \xrightarrow{a} \mathbf{p}' \wedge \mathbf{p}' \in Q\} \\
\overline{[a]}\, Q & := & \{\mathbf{p} \in \mathcal{P} \mid \forall \mathbf{p}'.\, \mathbf{p} \xrightarrow{a} \mathbf{p}' \Rightarrow \mathbf{p}' \in Q\}
\end{array}$$

Notation 2.3 *Wir sagen "***p** *hat die Eigenschaft* φ*" oder "***p** *erfüllt die Spezifikation* φ*", in Zeichen* $\mathbf{p} \models \varphi$*, wenn* $\mathbf{p} \in [\![\varphi]\!]$*. Weiterhin führen wir folgende Abkürzungen ein:*

$$
\begin{array}{lcll}
[A]\varphi & := & [a_1]\varphi \wedge \ldots\ldots \wedge [a_n]\varphi & \\
\langle A\rangle\varphi & := & \langle a_1\rangle\varphi \vee \ldots\ldots \vee \langle a_n\rangle\varphi & \quad \textit{wobei} \quad A = \{a_1, \ldots, a_n\} \subseteq \mathsf{Act} \\
[-A]\varphi & := & [\mathsf{Act} - A]\varphi & \\
[-]\varphi & := & [\mathsf{Act}\,]\varphi &
\end{array}
$$

Dies ist die Logik, die Hennessy und Milner [HM85] für die Charakterisierung der Bisimulation benutzt haben. Bezeichne $\mathsf{TH}(\mathbf{p})$ die Theorie von $\mathbf{p}$, also die Menge aller HML -Formeln für die $\mathbf{p} \models \varphi$ gilt und $\simeq_{Bisi}$ die Bisimulations-Relation.

Satz 2.4 *Wenn* $\mathbf{p} \simeq_{Bisi} \mathbf{q}$*, dann gilt* $\mathsf{TH}(\mathbf{p}) = \mathsf{TH}(\mathbf{q})$.

Der Satz besagt, daß HML adäquat bezüglich $\simeq_{Bisi}$ ist, da man keine Eigenschaften in HML formulieren kann, die zwei bisimulare Prozesse unterscheidet. Die Umkehrung dieses Satzes gilt nur, wenn die Prozesse des betrachteten Transitionssystems nur *bildendlich* (image finite) sind; das heißt, daß die Menge $\{\ \mathbf{q} \mid \mathbf{p} \xrightarrow{a} \mathbf{q}\ \}$ für alle Prozesse $\mathbf{p} \in \mathcal{P}$ und alle Aktionen a $\in \mathsf{Act}$ endlich ist.

Satz 2.5 *Wenn* $\mathsf{TH}(\mathbf{p})$=$\mathsf{TH}(\mathbf{q})$ *und alle von* **p** *und* **q** *aus durch Ausführen von Transitionen erreichbaren Prozesse bildendlich sind, dann gilt* $\mathbf{p} \simeq_{Bisi} \mathbf{q}$.

Unter der oben genannten Einschränkung ist HML also auch expressiv bezüglich $\simeq_{Bisi}$, da man nicht-bisimulare Prozesse unterscheiden kann. Diese beiden Sätze nennt man *die modale Charakterisierung der Bisimulation.*

Mit dieser Sprache können wir jetzt Eigenschaften von Prozessen ausdrücken und somit Prozeßmengen beschreiben, die gerade die geforderten Eigenschaften haben.

Beispiel 2.6 *Sei* $\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_6\}$ *ein Prozeßsystem und* $\mathsf{Act} = \{a, b\}$ *die Menge von Aktionen, die diese Prozesse* $\mathbf{p}_i$ *ausführen können. Das Verhalten dieser Prozesse sei durch das Transitionssystem* $T = (\mathcal{P}, \{\xrightarrow{a}, \xrightarrow{b}\})$ *in Abbildung 2.1 beschrieben. Um zu verdeutlichen, wie man Eigenschaften der Prozesse formuliert, geben wir zunächst stets eine informelle Beschreibung der Eigenschaft an, dann die entsprechende Formulierung in HML direkt aus der Definition der Semantik von HML .*

1. *es gibt einen a-Übergang, nach dem ein b-Übergang möglich ist*

$$\varphi_1 = \langle a\rangle\langle b\rangle \mathrm{tt} \ \textit{mit}\ [\![\varphi_1]\!] = \{\mathbf{p}_1, \mathbf{p}_4, \mathbf{p}_6\};$$

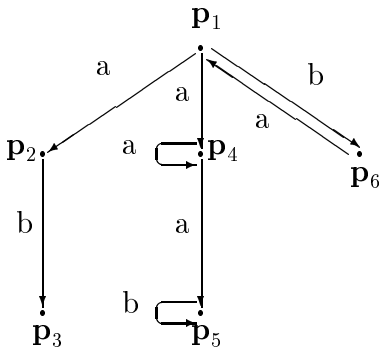


Abbildung 2.1: Transitionssystem für Beispiel 2.6

die duale Formel lautet $[a][b]\mathrm{ff}$ *und bezeichnet natürlich auch die duale Prozeßmenge, also alle Prozesse, die, falls sie einen a-Übergang machen können, danach nicht in der Lage sind, einen b-Übergang auszuführen.*

2. *nach allen a-Übergängen ist ein b möglich*

$$\varphi_2 = [a]\langle b\rangle \mathrm{tt} \text{ mit } [\![\varphi_2]\!] = \{\mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_5, \mathbf{p}_6\}$$

3. *nach allen a-Übergängen ist nur ein b möglich*

$$\varphi_3 = [a](\langle b\rangle \mathrm{tt} \wedge [-b]\mathrm{ff}) \text{ mit } [\![\varphi_3]\!] = \{\mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_5\}$$

4. *es gibt einen a-Übergang, nach dem nur ein b möglich ist*

$$\varphi_4 = \langle a\rangle(\langle b\rangle \mathrm{tt} \wedge [-b]\mathrm{ff}) \text{ mit } [\![\varphi_4]\!] = \{\mathbf{p}_1, \mathbf{p}_4\};$$

Man kann auch Eigenschaften formulieren, ohne sich direkt auf die Aktionen zu beziehen.

5. *es ist ein Übergang möglich*

$$\varphi_1 = \langle -\rangle \mathrm{tt} \text{ mit } [\![\varphi_1]\!] = \{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_4, \mathbf{p}_5, \mathbf{p}_6\}$$

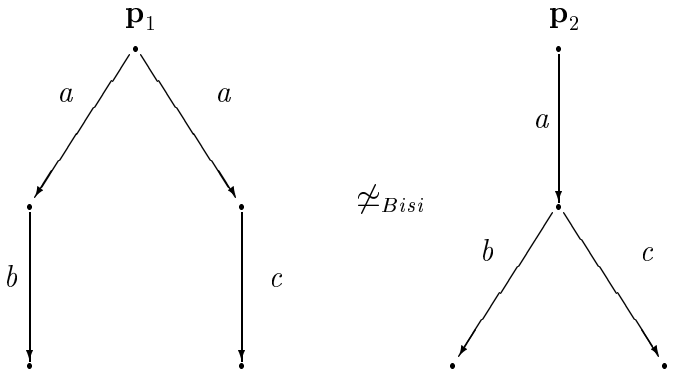
6. *nach allen Übergängen ist noch ein weiterer möglich*

$$\varphi_2 = [-]\langle -\rangle \mathrm{tt} \text{ mit } [\![\varphi_2]\!] = \{\mathbf{p}_1, \mathbf{p}_3, \mathbf{p}_4, \mathbf{p}_5, \mathbf{p}_6\}$$

7. *es ist höchstens ein Übergang möglich*

$$\varphi_3 = [-][-]\mathrm{ff} \text{ mit } [\![\varphi_3]\!] = \{\mathbf{p}_2, \mathbf{p}_3\}$$

Mit solchen HML -Formeln kann man nun nicht-bisimulare Prozesse unterscheiden:



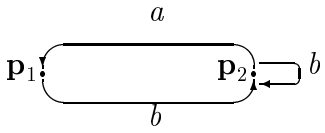
dann gilt z.B.:

$$\varphi_1 = \langle a \rangle (\langle b \rangle \mathrm{tt} \wedge [c]\mathrm{ff}) \in \mathsf{TH}(\mathbf{p}_1) \setminus \mathsf{TH}(\mathbf{p}_2)$$

$$\varphi_2 = \langle a \rangle (\langle b \rangle \mathrm{tt} \wedge \langle c \rangle \mathrm{tt}) \in \mathsf{TH}(\mathbf{p}_2) \setminus \mathsf{TH}(\mathbf{p}_1)$$

Problematisch ist die Formulierung von unendlichem Verhalten.

Beispiel 2.7 *Sei das folgende Transitionssystem* $T = (\{\mathbf{p}_1, \mathbf{p}_2\}, \{\xrightarrow{a}, \xrightarrow{b}\})$ *gegeben:*



Wie beschreibt man Eigenschaften wie zum Beispiel:

1. *es kann unendlich oft der Übergang "erst a, dann b" hintereinander ausgeführt werden,*

2. *es können niemals zwei a-Übergänge hintereinander gemacht werden (= Sicherheits-Eigenschaft),*

3. *es ist irgendwann wieder ein a möglich (= Lebendigkeits-Eigenschaft).*

Solche Eigenschaften kann man nicht durch endlich viele HML -Formeln beschreiben, da jede einzelne Formel nur einen *endlichen* Teil des potentiell unendlichen Prozeßverhaltens beschreibt. Nur durch *unendliche* Mengen von HML -Formeln kann man unendliches Verhalten beschreiben. Die Eigenschaft 1 wäre beschreibbar durch die unendliche Konjunktion:

$$\langle a\rangle\langle b\rangle \mathrm{tt} \wedge \langle a\rangle\langle b\rangle\langle a\rangle\langle b\rangle \mathrm{tt} \wedge \ldots = \bigwedge_{i\in\omega} (\langle a\rangle\langle b\rangle)^i \mathrm{tt}$$

Eigenschaft 2 wird durch die folgende Konjunktion ausgedrückt:

$$[a][a]\mathrm{ff} \wedge [-][a][a]\mathrm{ff} \wedge [-][-][a][a]\mathrm{ff} \wedge \ldots = \bigwedge_{i\in\omega} [-]^i[a][a]\mathrm{ff}$$

Die letzte Eigenschaft kann man durch eine Disjunktion beschreiben:

$$\langle a\rangle \mathrm{tt} \vee \langle -\rangle\langle a\rangle \mathrm{tt} \vee \langle -\rangle\langle -\rangle\langle a\rangle \mathrm{tt} \vee \ldots = \bigvee_{i\in\omega} \langle -\rangle^i\langle a\rangle \mathrm{tt}$$

Da jede HML -Formel jeweils nur einen endlichen Teil des Verhaltens eines Prozesses beschreibt, kann man also mit dieser Sprache keine *eventualities*[1] und keine *invariants*[2] ausdrücken. Dies sind aber gerade die interessanten Eigenschaften bei unendlichen Prozessen. Insbesondere die *Sicherheits-Eigenschaften*: "es passiert nie etwas Schlechtes", und die *Lebendigkeits-Eigenschaften*: "irgendwann passiert etwas Gutes", benötigt man für die Spezifikation von deadlock und livelock freien realen Systemen. Aus diesem Grund erweitert man HML um Fixpunkte. Deswegen wird HML um Fixpunkte erweitert. Im nächsten Kapitel beschreiben wir das Vorgehen bei dieser Erweiterung, wobei wir uns an die Artikel [Lar88] und [SW89] halten und untersuchen die neue Logik bezüglich ihrer Ausdrucksstärke und Eignung für die Spezifikation von verteilten Systemen.

[1] Es passiert etwas an einem unspezifizierten Zeitpunkt in der Zukunft.

[2] Es gilt etwas während des gesamten potentiell unendlichen Verhaltens eines Prozesses.

2.3 Erweiterung um Rekursion

Wie wir im letzten Abschnitt gesehen haben, läßt sich unendliches Verhalten von Prozessen nur durch unendliche Mengen von HML -Formeln beschreiben. Die Eigenschaft, daß ein Prozeß **p** unendlich oft ein gewisses Verhalten zeigt beschreibt man durch die unendliche Konjunktion:"**p** zeigt das Verhalten einmal, zweimal ...". Die grundlegende Idee der Erweiterung von HML besteht in der *rekursiven* Formulierung von Eigenschaften. Statt durch die unendliche Konjunktion von Eigenschaften beschreibt man **p** dann durch die rekursive Eigenschaft:"**p** zeigt das gewünschte Verhalten einmal und verhält sich dann wieder wie am Anfang". Dafür erweitert man HML so, daß man HML -Formeln rekursiv formulieren kann. Dazu fügt man HML im ersten Schritt eine Menge von propositionalen Variablen hinzu.

Definition 2.8 ($\mathrm{HML}_{\mathsf{Var}}$ -Syntax) *Sei* Var *eine Menge von propositionalen Variablen. Dann ist $\mathrm{HML}_{\mathsf{Var}}$ die kleinste Menge von Formeln, die gemäß folgender Syntax erzeugt werden:*

$$\varphi \ ::= \ \mathrm{tt} \mid \mathrm{ff} \mid X \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \langle a \rangle \varphi \mid [a]\varphi \qquad a \in \mathsf{Act} \ ; \ X \in \mathsf{Var}$$

Wenn man nun den Variablen durch eine *Variablenbelegung* $\sigma : \mathsf{Var} \longrightarrow 2^{\mathcal{P}}$ eine Prozeßmenge $\sigma(X)$ von einem beliebig, aber fest gewählten Transitionssystem T=($\mathcal{P}$,{$\stackrel{a}{\longrightarrow}$, a $\in$ Act }) zuordnet, kann man wie bei HML festlegen, was für Eigenschaften die Formeln von $\mathrm{HML}_{\mathsf{Var}}$ definieren, das heißt, welche Prozeßmenge sie bezeichnen.

Definition 2.9 ($\mathrm{HML}_{\mathsf{Var}}$ -Semantik) *Bei gegebener Variablenbelegung σ und gegebenem Transitionssytem $T = (\mathcal{P}, \{\stackrel{a}{\longrightarrow}, a \in \mathsf{Act}\})$ ist die durch eine Formel $\varphi \in \mathrm{HML}_{\mathsf{Var}}$ beschriebene Prozeßmenge folgendermaßen induktiv definiert:*

$$\begin{aligned}
[\![X]\!]\sigma &= \sigma(X) \\
[\![\mathrm{tt}]\!]\sigma &= \mathcal{P} & \qquad [\![\mathrm{ff}]\!]\sigma &= \emptyset \\
[\![\varphi_1 \wedge \varphi_2]\!]\sigma &= [\![\varphi_1]\!]\sigma \cap [\![\varphi_2]\!]\sigma & \qquad [\![\varphi_1 \vee \varphi_2]\!]\sigma &= [\![\varphi_1]\!]\sigma \cup [\![\varphi_2]\!]\sigma \\
[\![[a]\varphi]\!]\sigma &= \overline{[a]}\,[\![\varphi]\!]\sigma & \qquad [\![\langle a \rangle \varphi]\!]\sigma &= \overline{\langle a \rangle}\,[\![\varphi]\!]\sigma
\end{aligned}$$

Die Operatoren $\overline{\langle a \rangle}$,$\overline{[a]}$ sind wie in Definition 2.2 definiert.

Notation 2.10 **p** *erfüllt φ gemäß σ, in Zeichen* **p** $\models_\sigma \varphi$, *wenn* **p**$\in [\![\varphi]\!]\sigma$.

Durch die Variablenbelegung σ wird den propositionalen Variablen eine Bedeutung in Form einer Prozeßmenge gegeben. Wir werden jetzt mit Hilfe sogenannter *modaler Gleichungen* X $\equiv^T \varphi$(X) Variablenbelegungen charakterisieren, die den Variablen interessante Prozeßmengen zuweisen. Dazu beschreiben wir zunächst, was

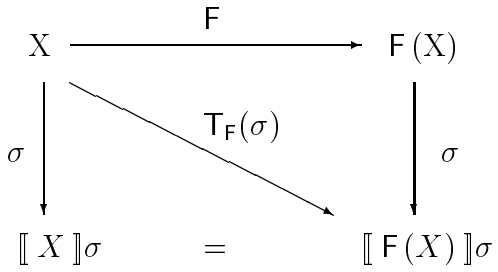


Abbildung 2.2: Transformation von σ durch F

eine modale Gleichung bedeutet und welche Variablenbelegung solch eine Gleichung löst. Wie wir sehen werden, ist diese Art der Charakterisierung nicht eindeutig; deswegen benutzt man Fixpunktformulierungen, um gewisse Lösungen auszuzeichnen.

Für die rekursive Formulierung von Eigenschaften geben wir den Variablen durch eine *Formelzuweisung* F noch eine weitere Bedeutung.

Definition 2.11 (Formelzuweisung) *Eine Formelzuweisung* F *ist eine Funktion* $\mathsf{F}: \mathsf{Var} \longrightarrow HML_{\mathsf{Var}}$ *, die jeder Variablen* $X \in \mathsf{Var}$ *eine beliebige* HML_{Var} *-Formel zuweist.*

Bei gegebener Variablenbelegung σ und Formelzuweisung F steht jede Variable nun für zwei verschiedene Prozeßmengen: $[\![X]\!]\sigma$ und $[\![X]\!]\mathsf{T}_{\mathsf{F}}(\sigma) := [\![\mathsf{F}(X)]\!]\sigma$. Solch eine Formelzuweisung F kann also als Transformation für Variablenbelegungen verwendet werden. Dabei weist man jeder Variablen X statt der durch σ angegebenen Prozeßmenge σ(X) die zu F(X) gehörige Prozeßmenge zu. Das heißt, daß jedes F ein gegebenes σ in eine Belegung $\mathsf{T}_{\mathsf{F}}(\sigma)$:X$\longrightarrow [\![\mathsf{F}(X)]\!]\sigma$ transformiert (siehe Abbildung 2.2).

Wir interessieren uns jetzt für diejenigen Variablenbelegungen σ, für die diese Semantiken übereinstimmen: $[\![X]\!]\sigma = [\![X]\!]\mathsf{T}_{\mathsf{F}}(\sigma)$, X $\in \mathsf{Var}$. Diese Bedingung, die man an eine Variablenbelegung stellt, werden wir im weiteren durch die modale Gleichung X $\equiv^T \mathsf{F}$(X) abkürzen, wobei T ein beliebiges Transitionssystem ist. Intuitiv soll die Formel X dieselbe Eigenschaft beschreiben wie die Formel F(X). Diese Eigenschaft X hieße dann intuitiv: die Prozesse, die X erfüllen, können das von F(X) verlangte Verhalten zeigen und sich dann wieder wie X verhalten. Dies ist genau die rekursive Art der Formulierung von Eigenschaften, die wir in der Einleitung als das Ziel der Erweiterung angegeben haben.

Beispiel 2.12 *In Beispiel 2.7 könnte man die Eigenschaft 1 anstelle der unendlichen Konjunktion durch die rekursive Gleichung* $X \equiv^T \langle a\rangle\langle b\rangle X$ *beschreiben. Eine*

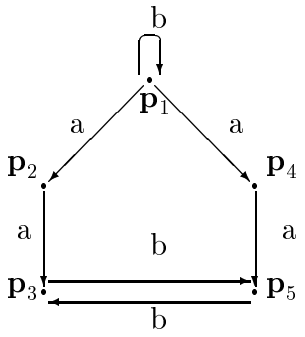


Abbildung 2.3: Transitionssystem für Beispiel 2.12

Variablenbelegung, die auf beiden Seiten der Gleichung zu der identischen Prozeßmenge führt, also die modale Gleichung löst, ist zum Beispiel $\sigma(X) = \{\mathbf{p}_2\}$, *da* $\overline{\langle a \rangle}\,\overline{\langle b \rangle}\,\{\mathbf{p}_2\} = \{\mathbf{p}_2\}$; *aber die Variablenbelegung* $\sigma(X) = \emptyset$ *ist ebenfalls eine Lösung der Gleichung.*

Die Mehrdeutigkeit der Charakterisierung von Variablenbelegungen durch modale Gleichungen wird an folgendem Beispiel noch deutlicher (siehe Abbildung 2.3): Mögliche Lösungen der Gleichung $X \equiv^T \langle b \rangle X$ *sind* $\sigma_1(X) = \emptyset$, $\sigma_2(X) = \{\mathbf{p}_1\}$, $\sigma_3(X) = \{\mathbf{p}_3, \mathbf{p}_5\}$ *und* $\sigma_4(X) = \{\mathbf{p}_1, \mathbf{p}_3, \mathbf{p}_5\}$.

Wie man am Beispiel erkennt, müssen wir präzisieren, welche Variablenbelegung σ wir durch die modale Gleichung $X \equiv^T \mathsf{F}(X)$ kennzeichnen wollen, also welche Prozeßmenge wir X zuweisen wollen. Wir werden nachher sehen, daß insbesondere zwei Lösungen der modalen Gleichung interessante Prozesse beschreiben, nämlich diejenigen, die X die kleinste, beziehungsweise die größte Prozeßmenge zuweisen, so daß die Gleichung erfüllt ist. Diese kleinste bzw. größte Lösung der modalen Gleichung erhält man als kleinsten bzw. größten Fixpunkt der Variablenbelegungs-Transformation $\mathsf{F}{:}\sigma \longrightarrow \mathsf{T}_{\mathsf{F}}(\sigma)$. Für diese Charakterisierung definieren wir zunächst den Verband der Variablenbelegungen und zeigen, daß die Transformation F monoton bezüglich dieser Belegungen ist. Nach dem Satz von Knaster-Tarski existieren somit eindeutige kleinste und größte Lösungen.

Definition 2.13 *Seien* σ_1, σ_2 *beliebige Variablenbelegungen und I eine Indexmenge.*

1. $\sigma_1 \subseteq \sigma_2$ *gdw.* $\sigma_1(X) \subseteq \sigma_2(X)$ *für alle* $X \in \mathsf{Var}$,
2. $\bigcup_{i \in I} \sigma_i$ *ist gegeben durch :* $(\bigcup_{i \in I} \sigma_i)(X) := \bigcup_{i \in I}(\sigma_i(X))$,
3. $\bigcap_{i \in I} \sigma_i$ *ist gegeben durch :* $(\bigcap_{i \in I} \sigma_i)(X) := \bigcap_{i \in I}(\sigma_i(X))$.

Zusammen mit dieser neuen Relation $\subseteq$ bildet die Menge aller Variablenbelegungen einen vollständigen Verband.

Definition 2.14 (Prä- und Post-Fixpunkt) *Eine Variablenbelegung σ heißt* Prä-Fixpunkt *einer Formelzuweisung F, wenn für jede Variable $X \in \mathsf{Var}$ gilt:*

$$[\![\mathsf{F}(X)]\!]\sigma \subseteq [\![X]\!]\sigma \ .$$

Eine Variablenbelegung σ heißt Post-Fixpunkt *von F, wenn gilt:*

$$[\![X]\!]\sigma \subseteq [\![\mathsf{F}(X)]\!]\sigma \ .$$

Da wir im weiteren besonders an Variablenbelegungen interessiert sind, die sowohl Prä- als auch Post-Fixpunkte von F sind, führen wir hierfür auch noch eine eigene Bezeichnung ein.

Definition 2.15 (F-Fixpunkt) *Eine Variablenbelegung σ heißt* F-Fixpunkt, *wenn sie sowohl Prä-Fixpunkt als auch Post-Fixpunkt von F ist.*

Als nächstes stellen wir den Zusammenhang zwischen den beiden Bedeutungen einer Variablen her und erklären, wie man die gesuchten kleinsten und größten Lösungen von modalen Gleichungen aus der zugrundeliegenden Formelzuweisung entwickeln kann.

Wenn man eine Formelzuweisung F als Variablenbelegungs-Transformation betrachtet, ist eine Variablenbelegung σ genau dann ein Prä- (bzw. Post-) Fixpunkt von F, wenn $\mathsf{T}_\mathsf{F}(\sigma) \subseteq \sigma$ (bzw. $\sigma \subseteq \mathsf{T}_\mathsf{F}(\sigma)$), und σ ist genau dann F-Fixpunkt, wenn σ ein Fixpunkt von $\mathsf{F}{:}\sigma \longrightarrow \mathsf{T}_\mathsf{F}(\sigma)$ ist. Das bedeutet, daß die Fixpunkte von F gerade die Lösungen der modalen Gleichung $\mathrm{X} \equiv^T \mathsf{F}(\mathrm{X})$ sind.

Wenn die Transformation T_F monoton bezüglich σ ist, existieren nach dem Fixpunktsatz von Knaster-Tarski eindeutige kleinste und größte Fixpunkte von T_F.

Satz 2.16 (Knaster-Tarski) *Sei V die Trägermenge eines vollständigen Verbands mit der Ordnungsrelation $\preceq$ und $f : V \longrightarrow V$ eine monotone Funktion auf V. Dann hat f:*

1. *einen kleinsten Fixpunkt, gegeben durch $INF\{X \preceq V |\ f(X) \preceq X\}$,*
2. *einen größten Fixpunkt, gegeben durch $SUP\{X \preceq V |\ X \preceq f(X)\}$.*

Um diesen Satz auf T_F anwenden zu können, muß T_F monoton bezüglich Variablenbelegungen sein; also wenn $\sigma_1 \subseteq \sigma_2$ gilt, so muß für alle $X \in \mathsf{Var}$ gelten:

$$\mathsf{T}_\mathsf{F}(\sigma_1)(X) = [\![\mathsf{F}(X)]\!]\sigma_1 \subseteq [\![\mathsf{F}(X)]\!]\sigma_2 = \mathsf{T}_\mathsf{F}(X)(\sigma_2)$$

Das bedeutet, die Monotonie und die Stetigkeit von T_F ist identisch mit der Monotonie und der Stetigkeit von $\mathrm{HML}_\mathsf{Var}$ -Formeln bezüglich Variablenbelegungen. Dafür müssen wir zeigen, daß jede Formel $\varphi \in \mathrm{HML}_\mathsf{Var}$ monoton bezüglich σ ist.

Lemma 2.17 (Monotonie von $\mathrm{HML}_{\mathsf{Var}}$) *Sei $\varphi \in HML_{\mathsf{Var}}$ gegeben.*

$$\textit{Wenn } \sigma_1 \subseteq \sigma_2, \textit{ dann gilt } [\![\varphi]\!]\sigma_1 \subseteq [\![\varphi]\!]\sigma_2.$$

Der Beweis ergibt sich per Induktion über den Formelaufbau direkt aus der Semantik-Definition von $\mathrm{HML}_{\mathsf{Var}}$. Verwendet man HML mit $\neg$-Operator, gilt die Monotonie nur für solche Formeln, bei der die freien Variablen im Geltungsbereich einer geraden Anzahl von Negationen auftauchen. Wenn das betrachtete Transitionssystem bildendlich ist, ist jedes φ sogar stetig und antistetig und somit auch T_F, was wir im nächsten Lemma festhalten.

Lemma 2.18 (Stetigkeit und Antistetigkeit von $\mathrm{HML}_{\mathsf{Var}}$) *Gegeben sei die Formel $\varphi \in HML_{\mathsf{Var}}$ und ein bildendliches Transitionssystem. Dann gilt:*

$$\textit{Stetigkeit: } \textit{Wenn } \sigma_1 \subseteq \sigma_2 \subseteq \sigma_3 \ldots, \textit{ dann gilt } [\![\varphi]\!](\bigcup_{i\in\omega} \sigma_i) = \bigcup_{i\in\omega} ([\![\varphi]\!]\sigma_i).$$
$$\textit{Antistetigkeit: } \textit{Wenn } \sigma_1 \supseteq \sigma_2 \supseteq \sigma_3 \ldots, \textit{ dann gilt } [\![\varphi]\!](\bigcap_{i\in\omega} \sigma_i) = \bigcap_{i\in\omega} ([\![\varphi]\!]\sigma_i).$$

Korollar 2.19 T_F *ist für beliebige Formelbelegungen* F *monoton bezüglich Variablenbelegungen und sogar stetig, wenn das zugrundeliegende Transitionssystem bildendlich ist.*

Mit der gezeigten Monotonie von T_F können wir nun den Fixpunktsatz von Knaster-Tarski anwenden.

Korollar 2.20 *Sei eine beliebige Formelbelegung* F *gegeben. Dann existieren für die modale Gleichung* $X \equiv^T \mathsf{F}(X)$ *eine eindeutige kleinste und eindeutige größte Lösung.*

Wegen der herausragenden Bedeutung der beiden speziellen Lösungen bekommen diese eine eigene Bezeichnung.

Notation 2.21

- $\sigma_\mu := \bigcap\{\sigma |\ \mathsf{T}_\mathsf{F}(\sigma) \subseteq \sigma\}$ *ist die kleinste Lösung,*
- $\sigma_\nu := \bigcup\{\sigma |\ \sigma \subseteq \mathsf{T}_\mathsf{F}(\sigma)\}$ *ist die größte Lösung*

Eine sehr nützliche alternative Charakterisierung, mit der man diese beiden Lösungen tatsächlich berechnen kann, ist die *Kleene'sche Approximation* von Fixpunkten.

Satz 2.22 (Kleene-Approximation für $\mathsf{T_F}$) *Wenn $\mathsf{T_F}$ stetig und antistetig ist, so ist ν das Infimum der folgenden Approximationskette aus Präfixpunkten von F:*

$$\sigma_p \supseteq \mathsf{T_F}(\sigma_p) \supseteq \mathsf{T_F}^2(\sigma_p) \supseteq \mathsf{T_F}^3(\sigma_p) \ldots \ ,$$

und μ ist das Supremum der folgenden Postfixpunkte:

$$\sigma_\emptyset \subseteq \mathsf{T_F}(\sigma_\emptyset) \subseteq \mathsf{T_F}^2(\sigma_\emptyset) \subseteq \mathsf{T_F}^3(\sigma_\emptyset) \ldots \ .$$

Dabei ist $\sigma_p(X) = \mathcal{P}$ und $\sigma_\emptyset(X) = \emptyset$ für alle X$\in$Var .

Notation 2.23 *Die Prozeßmenge, die einer beliebigen Variablen X gemäß der größten Lösung ν bzw. der kleinsten Lösung μ der modalen Gleichung $X \equiv^T \mathsf{F}\,(X)$ zugewiesen wird, bezeichnen wir im weiteren durch die folgenden Formeln:*

$$[\![\ \nu X.\mathsf{F}\,(X)\ ]\!] = \sigma_\nu(X) \text{ bzw. } [\![\ \mu X.\mathsf{F}\,(X)\ ]\!] = \sigma_\mu(X).$$

Mit der oben genannten Notation erhalten wir die endgültige Syntax unserer Spezifikations-Logik μHML.

Definition 2.24 (μHML-Syntax) *μHML ist die kleinste Menge von Formeln, die sich gemäß folgender Syntax bilden lassen:*

$$\varphi \ := \ X \ \mid \ \varphi_1 \wedge \varphi_2 \ \mid \ \varphi_1 \vee \varphi_2 \ \mid \ \langle a \rangle \varphi \ \mid \ [a]\varphi \ \mid \ \nu X.\varphi \ \mid \ \mu X.\varphi \qquad a \in \mathsf{Act}\ ;\ X \in \mathsf{Var}$$

Auf die propositionalen Konstanten tt und ff können wir jetzt verzichten, weil tt der neuen Formel $\nu X.X$ entspricht und ff durch die Formel $\mu X.X$ beschrieben werden kann[3]. Jetzt können wir die vollständige formale Semantik von HML mit Rekursion angeben.

Definition 2.25 (μHML-Semantik) *Bei gegebener Variablenbelegeng σ und gegebenem Transitionssystem $T = (\mathcal{P}, \{\xrightarrow{a}, a \in \mathsf{Act}\})$ ist die durch eine Formel $\varphi \in \mu$HML beschriebene Prozeßmenge wie folgt induktiv definiert :*

$$\begin{aligned}
[\![\ X\ ]\!]\sigma &= \sigma(X) \\
[\![\ \varphi_1 \wedge \varphi_2\ ]\!]\sigma &= [\![\ \varphi_1\ ]\!]\sigma \ \cap \ [\![\ \varphi_2\ ]\!]\sigma \qquad & [\![\ \varphi_1 \vee \varphi_2\ ]\!]\sigma &= [\![\ \varphi_1\ ]\!]\sigma \ \cup \ [\![\ \varphi_2\ ]\!]\sigma \\
[\![\ [a]\varphi\ ]\!]\sigma &= \overline{[a]}\,[\![\ \varphi\ ]\!]\sigma & [\![\ \langle a\rangle\varphi\ ]\!]\sigma &= \overline{\langle a\rangle}\,[\![\ \varphi\ ]\!]\sigma \\
[\![\ \nu X.\varphi\ ]\!]\sigma &= \sigma_\nu(X) & [\![\ \mu X.\varphi\ ]\!]\sigma &= \sigma_\mu(X)
\end{aligned}$$

Dabei sind die Variablenbelegungen σ_ν und σ_μ wie in Notation 2.21 definiert:

$$\sigma_\mu = \bigcap\{\sigma \subseteq \sigma_\mathcal{P} | \mathsf{T}_\varphi\ (\sigma) \subseteq \sigma\}$$
$$\sigma_\nu = \bigcup\{\sigma \subseteq \sigma_\mathcal{P} | \sigma \subseteq \mathsf{T}_\varphi\ (\sigma)\}$$

[3]die größte Prozeßmenge, die man X zuweisen kann, um die Gleichung X $\equiv^T$ X zu erfüllen, ist die gesamte Prozeßmenge, während die kleinste die leere Menge ist.

Wir werden uns für die Beschreibung von temporalen Eigenschaften von Prozessen auf geschlossene Formeln beschränken, in denen also jede Variable X $\in$ Var durch einen Variablenbinder $\nu X.$ oder $\mu X.$ gebunden ist. Somit können wir im weiteren auf Variablenbelegungen verzichten, da für geschlossene Formeln $[\![.]\!]=[\![.]\!]\sigma$ gilt. Neben der *semantischen Approximation* von Fixpunkten durch die Kleene'sche Approximation gibt es auch eine *syntaktische Approximation* dieser Fixpunkte. Dies sind unendliche HML -Formelmengen, die die gleiche Prozeßmenge beschreiben, wie die approximierte Fixpunktformel. Im folgenden nehmen wir an, daß das betrachtete Transitionssystem höchstens abzählbar viele Zustände hat [4]. Der Ausdruck $\varphi[X := \psi]$ steht für die Formel φ', in der jedes freie Vorkommen von X in φ durch die Formel ψ ersetzt wurde.

Satz 2.26 (syntaktische Approximation) *Sei* $\nu^0 X.\varphi := \mathrm{tt}$ *und* $\nu^{i+1}X.\varphi := \varphi[X := \nu^i X.\varphi]$, *wofür wir abkürzend* $\varphi^{i+1}(\mathrm{tt})$ *schreiben; und sei* $\mu^0 X.\varphi := \mathrm{ff}$ *sowie* $\mu^{i+1}X.\varphi := \varphi[X := \mu^i X.\varphi]$, *wofür wir* $\varphi^{i+1}(\mathrm{ff})$ *schreiben. Ist* φ *stetig und antistetig, so gelten die beiden folgenden Äquivalenzen für beliebige Prozesse* **p**:

1. $\mathbf{p} \models \nu X.\varphi \iff \mathbf{p} \models \bigwedge_{i\in\omega} \nu^i X.\varphi$

2. $\mathbf{p} \models \mu X.\varphi \iff \mathbf{p} \models \bigvee_{i\in\omega} \mu^i X.\varphi$

Mit den Fixpunktformeln haben wir also eine sehr kompakte Schreibweise für solche unendlichen HML -Formelmengen gefunden. Außerdem ermöglicht diese Äquivalenz, die Bedeutung von Fixpunktformeln viel intuitiver als bisher zu erklären. So beschreibt νX.φ diejenigen Prozesse, die für alle $i \in \omega$ die Formel $\varphi^i(\mathrm{tt})$ erfüllen, also unendlich oft das von φ verlangte Verhalten zeigen können. Entsprechend der Charakterisierung von Sicherheitseigenschaften ("Es passiert nie etwas Schlechtes") kann man die größten Fixpunke zur Spezifikation von Sicherheits-Eigenschaften verwenden. Die Formel $\mu X.\varphi$ hingegen beschreibt Prozesse, die eine der Formeln $\varphi^i(\mathrm{ff})$ erfüllen, was aber bedeutet, daß diese Prozesse nur endlich oft das Verhalten φ zeigen, da kein Prozeß ff erfüllen kann. Diese Fixpunktart kann man demnach für die Spezifikation von Lebendigkeits-Eigenschaften benutzen, wenn man Lebendigkeit allgemein auffaßt als "irgendwann passiert etwas Gutes".

[4] Die gewöhnliche Fixpunktinduktion verallgemeinert sich im überabzählbaren Falle zu transfiniter Induktion

Den Zusammenhang der beiden Approximationen für eine Formel $\nu X.\varphi$ stellt die nachfolgende Tabelle dar.

syntaktische Approximation	semantische Approximation
$\nu^0 X.\varphi : \varphi^0(\mathrm{tt}) = \mathrm{tt}$	$[\![X]\!]\mathsf{T}^0_\varphi(\sigma_\mathcal{P}) = [\![X]\!]\sigma_\mathcal{P}$
$\nu^1 X.\varphi : \varphi^1(\mathrm{tt}) = \varphi[X := \varphi^0(\mathrm{tt})] = \varphi[X := \mathrm{tt}]$	$[\![X]\!]\mathsf{T}^1_\varphi(\sigma_\mathcal{P}) = [\![\varphi]\!]\sigma_\mathcal{P}$
$\nu^2 X.\varphi : \varphi^2(\mathrm{tt}) = \varphi[X := \varphi^1(\mathrm{tt})] =$	$[\![X]\!]\mathsf{T}^2_\varphi(\sigma_\mathcal{P}) = [\![\varphi]\!]\mathsf{T}^1_\varphi(\sigma_\mathcal{P}) =$
$= \varphi[X := \varphi[X := \mathrm{tt}]]$	$= [\![\varphi[X := \varphi]]\!]\sigma_\mathcal{P}$
$\vdots$	$\vdots$

Wie man sieht, gilt stets:

- $[\![\nu^i X]\!]\sigma_\mathcal{P} = [\![X]\!]\mathsf{T}^i_\varphi(\sigma_\mathcal{P})$
- $[\![\nu X.\varphi]\!] = [\![X]\!](\bigcap_{i\in\omega} \mathsf{T}^i_\varphi(\sigma_\mathcal{P})) = \bigcap_{i\in\omega} [\![X]\!]\mathsf{T}^i_\varphi(\sigma_\mathcal{P}) = \bigcap_{i\in\omega} [\![\nu^i X.\varphi]\!] = [\![\bigwedge_{i\in\omega} \nu^i X.\varphi]\!]$
- $[\![\mu^i X]\!]\sigma_\emptyset = [\![X]\!]\mathsf{T}^i_\varphi(\sigma_\emptyset)$
- $[\![\mu X.\varphi]\!] = [\![X]\!](\bigcup_{i\in\omega} \mathsf{T}^i_\varphi(\sigma_\emptyset)) = \bigcup_{i\in\omega} ([\![X]\!]\mathsf{T}^i_\varphi(\sigma_\emptyset)) = \bigcup_{i\in\omega} [\![\mu^i X.\varphi]\!] = [\![\bigvee_{i\in\omega} \mu^i X.\varphi]\!]$

Die beiden unteren Aussagen erhält man aus der entsprechenden Gegenüberstellung der beiden Approximationen für die Formel $\mu X.\varphi$:

syntaktische Approximation	semantische Approximation
$\mu^0 X.\varphi : \varphi^0(\mathrm{ff}) = \mathrm{ff}$	$[\![X]\!]\mathsf{T}^0_\varphi(\sigma_\emptyset) = [\![X]\!]\sigma_\emptyset$
$\mu^1 X.\varphi : \varphi^1(\mathrm{ff}) = \varphi[X := \varphi^0(\mathrm{ff})] = \varphi[X := \mathrm{ff}]$	$[\![X]\!]\mathsf{T}^1_\varphi(\sigma_\emptyset) = [\![\varphi]\!]\sigma_\emptyset$
$\mu^2 X.\varphi : \varphi^2(\mathrm{ff}) = \varphi[X := \varphi^1(\mathrm{ff})] =$	$[\![X]\!]\mathsf{T}^2_\varphi(\sigma_\emptyset) = [\![\varphi]\!]\mathsf{T}^1_\varphi(\sigma_\emptyset) =$
$= \varphi[X := \varphi[X := \mathrm{ff}]]$	$= [\![\varphi[X := \varphi]]\!]\sigma_\emptyset$
$\vdots$	$\vdots$

Durch die syntaktische Approximation erhält man sehr intuitive Beschreibungen von Prozeßmengen, die durch Fixpunktformeln beschrieben werden.

Beispiel 2.27 (Approximationen) *Sei ein beliebiges Transsitionssystem gegeben. Welche Prozeßmenge beschreibt die Formel $\nu X.\langle a\rangle X$? Die syntaktische Approxima-*

tion liefert:

$\nu^0 X.\langle a\rangle X = \mathrm{tt}$
beschreibt alle Prozesse,
$\nu^1 X.\langle a\rangle X = \langle a\rangle \mathrm{tt}$
alle Prozesse, die ein a ausführen können,
$\nu^2 X.\langle a\rangle X = \langle a\rangle\langle a\rangle \mathrm{tt}$
alle Prozesse, die zwei a's ausführen können.
$\vdots$

Die Konjunktion dieser Approximationen, und somit auch die Fixpunktformel, beschreibt also alle Prozesse, die einen unendlichen a-Pfad haben. Die Approximation der Formel $\mu X.[a]X$ hingegen liefert folgendes:

$\mu^0 X.[a]X = \mathrm{ff}$
beschreibt die leere Prozeßmenge,
$\nu^1 X.[a]X = [a]\mathrm{ff}$
alle Prozesse, die kein a ausführen können,
$\nu^2 X.[a]X = [a][a]\mathrm{ff}$
alle Prozesse, die höchstens ein a ausführen können,
$\vdots$

Die Disjunktion dieser Formeln und somit $\mu X.[a]X$ beschreibt alle Prozesse, die nur endlich oft hintereinander a ausführen können. Von besonderem Interesse ist die Bedeutung der Formel $\nu X.\varphi \wedge [-]X$

$\nu^0 X.\varphi \wedge [-]X = \mathrm{tt}$
beschreibt wieder alle Prozesse,
$\nu^1 X.\varphi \wedge [-]X = \varphi \wedge [-]\mathrm{tt}$
alle Prozesse, die die Eigenschaft φ haben,
$\nu^2 X.\varphi \wedge [-]X = \varphi \wedge [-](\varphi \wedge [-]\mathrm{tt})$
alle Prozesse, die jetzt φ erfüllen und nach dem Ausführen einer beliebigen Aktion ebenfalls wieder,
$\vdots$

Diese Fixpunktformel beschreibt somit alle Prozesse, für die φ stets gilt, das heißt, auf allen Pfaden und zu jedem Zeitpunkt.

Diese letzte Formel entspricht in ihrer Bedeutung dem Standard-Branching-Time-Operator $\forall G\varphi$. Ebenso kann man μHML-Makros angeben, die die restlichen Standard-Operatoren gemaß Pnueli [Pnu85] definieren.

Beobachtung 2.28

1. $\exists F\varphi := \mu X.\varphi \vee \langle - \rangle X$;
   es gibt einen Pfad, auf dem irgendwann mal φ gilt.

2. $\exists G\varphi := \nu X.\varphi \wedge ([-]\mathrm{ff} \vee \langle - \rangle X)$;
   es gibt einen Pfad, auf dem stets φ gilt.

3. $\forall F\varphi := \mu X.\varphi \vee (\langle - \rangle \mathrm{tt} \wedge [-]X)$;
   auf allen Pfaden gilt irgendwann mal φ.

4. $\varphi U_{st} \psi := \mu X.\psi \vee (\varphi \wedge \langle - \rangle \mathrm{tt} \wedge [-]X)$;
   φ gilt auf allen Pfaden solange, bis irgendwann mal ψ, gilt und ψ gilt irgendwann in der Zukunft.

5. $\varphi U_{we} \psi := \nu X.\psi \vee (\varphi \wedge [-]X)$;
   φ gilt auf allen Pfaden solange, bis irgendwann mal ψ gilt, oder φ gilt immer.

Wie wir an den Beispielen gesehen haben, kann man mit der erweiterten Logik auch Aussagen über unendliches Verhalten machen. Doch wie sieht es mit der generellen Eignung von μHML für die Spezifikation von verteilten Systemen aus? In Kapitel 1 haben wir Kriterien für die Beurteilung der Eignung einer Logik als Spezifikationssprache angeführt:

1. Ausdrucksstärke

2. Verträglichkeit mit der verwendeten Verhaltensäquivalenz

3. Verifizierbarkeit von Verfeinerungsschritten.

Auf diese drei Kriterien gehen wir jetzt näher ein.

Die Erweiterung der modalen Logik HML um die beiden Fixpunktoperatoren liefert eine sehr ausdrucksstarke temporale Logik. Es sind nicht nur alle Standard-Operatoren der (pure branching time [5]) temporalen Logik definierbar, sondern auch z.B. schwache und starke Until-Operatoren [Lar88](siehe Beobachtung 2.28). Wie man sieht, kann man aus den wenigen Grundbausteinen, die μHML benutzt, mächtige Makros definieren, die dann insbesondere auch die Lesbarkeit von μHML-Spezifikationen verbessern.

Da bei der Erweiterung von HML zu μHML eigentlich nur syntaktische Abkürzungen für unendliche HML -Formelmengen eingeführt wurden, hat sich die

[5] das bedeutet, daß es nur Zustandsformeln und keine Pfadformeln wie in der "full branching time" temporalen Logik gibt. Die verwendeten Operatoren quantifizieren also stets über alle Pfade, die durch einen Zustand einer Kripke-Struktur führen.

logische Äquivalenz, die μHML auf Prozessen induziert, nicht verändert. Das bedeutet, daß auch die erweiterte HML -Version mit der Bisimulationsäquivalenz verträglich ist, diesmal sogar ohne die Einschränkung der Bildendlichkeit. Im folgenden Satz bezeichnet $\mathsf{TH}(\mathbf{p}) \subseteq \mu$HML wiederum die Theorie von $\mathbf{p}$ und $\simeq_{Bisi}$ die Bisimulations-Relation.

Satz 2.29 (μHML und Bisimulation [SW90])

$$\textit{Es gilt } \mathbf{p} \simeq_{Bisi} \mathbf{q} \textit{ gdw. } \mathsf{TH}(\mathbf{p}) = \mathsf{TH}(\mathbf{q}).$$

Als letztes Kriterium bleibt die Verifizierbarkeit von Verfeinerungsschritten. Wie in der Einleitung der Studienarbeit erwähnt, bedeutet das bei unserem Vorgehen: wie verifiziert man, daß ein gegebener Prozeß eine spezifizierte Eigenschaft hat, und wie zeigt man, daß eine verfeinerte Spezifikation korrekt in Bezug auf die Ausgangsspezifikation ist? Der erste Fall ist unter anderem von Stirling, Larsen, Winskel und Cleaveland untersucht worden. In den Artikeln [SW89] [Lar88] [Win89] [Cle90] werden korrekte, vollständige und implementierbare Beweissysteme vorgestellt. Diese tableaubasierten Systeme werden als *Model-Checker* bezeichnet und benutzen alle dieselbe Beweistechnik : *Fixpunktinduktion*.

An vergleichbaren Untersuchungen für die zweite Fragestellung, wann eine μHML-Spezifikation eine andere korrekt verfeinert, ist uns nur ein Hilbert-System von Kozen für einen Teil des minimalen modalen μ-Kalküls [Koz83] bekannt. Da diese Systeme sehr unkomfortabel sind und kaum effizient implementierbar, haben wir mit der vorliegenden Arbeit versucht, diese Lücke zu schließen. Dafür haben wir ein tableaubasiertes Beweissystem für die semantische Implikation zwischen μHML-Formeln konstruiert, welches ebenfalls Fixpunktinduktion benutzt. Um die zugrundeliegende Idee und die Arbeitsweise unseres Systems zu verdeutlichen, stellen wir im nächsten Kapitel die oben angesprochenen Systeme am Beispiel eines Model-Checkers von Stirling vor.

Kapitel 3

Model-Checking

3.1 Einleitung

In diesem Kapitel wird nun das Problem des Model-Checkings behandelt, also die Frage, wann ein Zustand eines Transitionssystems **p** eine bestimmte Eigenschaft, ausgedrückt durch eine μHML-Formel, besitzt. Lösungen dieses Problems sind seit ein paar Jahren bekannt. Die erste Arbeit in diesem Zusammenhang stammt von Larsen [Lar88]. Sein System behandelt jedoch eine stark eingeschränkte Klasse von Formeln aus μHML, in denen entweder nur kleinste oder nur größte Fixpunkte vorkommen dürfen.

Ein System für beliebig geschachtelte Fixpunkte, welche gerade die Ausdrucksstärke von μHML ausmachen, stammt von Stirling und Walker [SW89]. Andere Model-Checker, die in ähnlicher Weise arbeiten finden sich bei Cleaveland [Cle90] und Winskel [Win89]. In der Version von Cleaveland wurde ein derartiges System im Rahmen der Concurrency Workbench [CPS89] implementiert.

Allen diesen Tableaumethoden ist gemeinsam, daß sie *zielgerichtete* Beweise ermöglichen und daß sie als als wichtiges Beweisprinzip zum Nachweis von Fixpunkteigenschaften *Fixpunktinduktion* benutzen. Einen konkreten Model-Checker werden wir in diesem Kapitel vorstellen. Bis auf geringfügige Unterschiede in der Darstellung halten wir uns an [SW89]. Die drei Bestandteile dieses Model-Checkers, nämlich *Regeln*, *Abbruchkriterien* und *Erfolgsbedingungen* stellen wir im nächsten Abschnitt dar. Besondere Sorgfalt erfordert die Behandlung der Fixpunkte. Auf die damit verbundenen Probleme gehen wir in einem eigenen Unterabschnitt ein. Anhand eines Beispiels soll in Abschnitt 3.3 die Arbeitsweise des Model-Checkers verdeutlicht werden.

3.2 Der Model-Checker

In diesem Abschnitt stellen wir einen konkreten Model-Checker vor. Bis auf geringfügige Unterschiede halten wir uns in der Darstellung an [SW89]. Bei dem Model-Checker handelt es sich um ein zielgerichtetes Beweissystem, welches mit Fixpunktinduktion arbeitet. Die drei Komponenten des Systems werden nun der Reihe nach vorgestellt.

Die Regeln

Die Regeln bestehen aus einem Ziel sowie aus einem oder mehreren Unterzielen. Mit ihnen lassen sich, entsprechen der zielgerichteten Vorgehensweise, aus einem Ziel ein oder mehrere Unterziele generieren. Wenn man dann die Unterziele zeigen kann, so ist damit auch das ursprüngliche Ziel gezeigt. Durch die Generierung von immer weiteren Unterzielen läßt sich auf diese Weise ein sogenanntes *Tableau* oder ein *Ableitungsbaum* für eine zu zeigende Eigenschaft eines endlichen Prozesses $\mathbf{p}$ erzeugen. Die Regeln sind in Abbildung 3.1 zusammengefaßt.

$(\wedge)$	$\dfrac{\mathbf{p} \vdash^{\mathcal{D}} \varphi_1 \wedge \varphi_2}{\mathbf{p} \vdash^{\mathcal{D}} \varphi_1 \ , \ \mathbf{p} \vdash^{\mathcal{D}} \varphi_2}$	
$(\vee_1)$	$\dfrac{\mathbf{p} \vdash^{\mathcal{D}} \varphi_1 \vee \varphi_2}{\mathbf{p} \vdash^{\mathcal{D}} \varphi_1}$	
$(\vee_2)$	$\dfrac{\mathbf{p} \vdash^{\mathcal{D}} \varphi_1 \vee \varphi_2}{\mathbf{p} \vdash^{\mathcal{D}} \varphi_2}$	
$([a])$	$\dfrac{\mathbf{p} \vdash^{\mathcal{D}} [a]\varphi}{\mathbf{p}_1 \vdash^{\mathcal{D}} \varphi \ \ldots \ \mathbf{p}_n \vdash^{\mathcal{D}} \varphi}$	$a \in \mathsf{Act}\ , \{\mathbf{p}_1, \ldots, \mathbf{p}_n\} = \{\mathbf{p}' \mid \mathbf{p} \xrightarrow{a} \mathbf{p}'\}$
$(\langle a \rangle)$	$\dfrac{\mathbf{p} \vdash^{\mathcal{D}} \langle a \rangle \varphi}{\mathbf{p}' \vdash^{\mathcal{D}} \varphi}$	$a \in \mathsf{Act}\ , \mathbf{p}' \in \{\mathbf{q} \mid \mathbf{p} \xrightarrow{a} \mathbf{q}\}$
(μ)	$\dfrac{\mathbf{p} \vdash^{\mathcal{D}} \mu X.\varphi}{\mathbf{p}' \vdash^{\mathcal{D}'} U}$	$\mathcal{D}' = \mathcal{D} \cdot (U = \mu X.\varphi)$, U neue Konstante
(ν)	$\dfrac{\mathbf{p} \vdash^{\mathcal{D}} \nu X.\varphi}{\mathbf{p}' \vdash^{\mathcal{D}'} U}$	$\mathcal{D}' = \mathcal{D} \cdot (U = \nu X.\varphi)$, U neue Konstante
$(Konst)$	$\dfrac{\mathbf{p} \vdash^{\mathcal{D}} U}{\mathbf{p}' \vdash^{\mathcal{D}'} \varphi[X := U]}$	$(U = \nu X.\varphi) \in \mathcal{D}$ oder $(U = \nu X.\varphi) \in \mathcal{D}$

Abbildung 3.1: Regeln des Model-Checkers

Die Regeln für die Konjunktion und die Disjunkion funktionieren wie erwartet: Soll man von einem Prozeß die Konjunktion zweier Eigenschaften zeigen, so muß

man zeigen, daß er sowohl die eine als auch die andere Eigenschaft besitzt. Dies sind die Unterziele in Regel ($\wedge$). Ist die zu beweisende Eigenschaft eine Disjunktion zweier Formeln, so genügt es zu zeigen, daß der Prozeß die eine der beiden Formeln erfüllt (Regeln ($\vee_1$) und ($\vee_2$)).

Die Unterziele bei den beiden Modaloperatoren $[a]$ und $\langle a \rangle$ beziehen sich nicht auf den ursprünglichen Zustand $\mathbf{p}$, sondern auf Zustände, die von $\mathbf{p}$ aus über passende Transitionen erreichbar sind. Entsprechend der Semantik der Modaloperatorn muß man bei $[a]\varphi$ die Eigenschaft φ von *allen* von $\mathbf{p}$ über eine a-Transition erreichbaren Zuständen zeigen, während bei $\langle a \rangle \varphi$ *ein* solcher Zustand genügt.

Etwas komplizierter sind die restlichen drei Regeln, die sich mit der Behandlung der Fixpunkte befassen. Um eine eindeutige Kennzeichnung von Fixpunktformeln zu erreichen, benutzt man *Konstanten.* (Im weiteren stehen Großbuchstaben U, V, W, U_1, U_2 ... immer für Konstanten). Regeln (μ) und (ν) dienen der Konstanteneinführung. Trifft man auf eine Formel mit einem Fixpunkt als äußerstem Konstrukt, so führt man eine neue Konstante als Abkürzung für die Formel ein. $\mathcal{D}$ und $\mathcal{D}'$ sind dabei sogenannte *Deklarationslisten* in denen festgehalten wird, welche Konstanten im Laufe der bisherigen Ableitung für welche Fixpunktformeln eingeführt wurden. In Regel (μ) ist $\mathcal{D}' = \mathcal{D} \cdot (U = \mu X.\varphi)$ dann die Deklarationsliste, die aus $\mathcal{D}$ durch Hinzufügen der Deklaration der Konstanten U als $\mu X.\varphi$ entsteht. Man beachte , daß die Deklarationslisten niemals verkürzt werden. Zu Beginn der Ableitung sind noch keine Konstanten deklariert, sodaß die Wurzel mit dem zu zeigenden Ziel $\mathbf{p} \models \varphi$ bezüglich der leeren Deklarationsliste ϵ beschriftet ist, also mit $\mathbf{p} \vdash^{\epsilon} \varphi$.

In Regel $(Konst)$ wird eine Konstante, die ja für eine Fixpunktformel steht, gemäß ihrer Deklaration in $\mathcal{D}$ durch die Expansion der entsprechenden Fixpunktformel ersetzt, was einer einmaligen Abwicklung des Fixpunktes entspricht.

Damit hat man sämtliche Regeln, mit denen neue Unterziele erzeugt werden können, beisammen.

Die Abbrüche

Zusätzlich zu den Regeln muß man jetzt noch angeben, wann die Anwendung dieser Regeln zum Abbruch kommt, das heißt, wann die Generierung von Unterzielen endet. Beim Abbruch unterscheidet man drei Fälle, die in Abbildung 3.2 zusammengestellt sind.

In den Fällen 1 und 2 bleibt nichts anderes übrig, als abzubrechen, da keine Regel mehr anwendbar ist. Fall 3 ist interessanter. Dadurch, daß weiter oberhalb die gleiche Situation bereits einmal aufgetreten ist, unter Umständen mit einer kürzeren Deklarationsliste, kann man an dieser Stelle mit der Ableitung aufhören.

Ein Ableitungsbaum, bei der an jedem Blatt abgebrochen wurde, die man also nicht mehr fortsetzen kann, nennen wir *maximal.*

1. $\mathbf{p} \vdash^{\mathcal{D}} [a]\varphi$ und es gibt kein $\mathbf{p}'$ mit $\mathbf{p} \xrightarrow{a} \mathbf{p}'$
2. $\mathbf{p} \vdash^{\mathcal{D}} \langle a \rangle \varphi$ und es gibt kein $\mathbf{p}'$ mit $\mathbf{p} \xrightarrow{a} \mathbf{p}'$
3. $\mathbf{p} \vdash^{\mathcal{D}} U$ und es gibt einen Knoten oberhalb im Ableitungsbaum, der mit $\mathbf{p} \vdash^{\mathcal{D}'} U$ beschriftet ist.

Abbildung 3.2: Abbruchbedingungen des Model-Checkers

Der Erfolg

Hat man nun eine maximale Ableitung, so muß man noch entscheiden, ob man diese als erfolgreich, das heißt als tatsächlichen Beweis für das ursprüngliche Ziel, ansehen will. Die Kriterien für den Erfolg eines Blattes stehen in Abbildung 3.3.

Ein Blatt eines vollständigen Ableitungsbaumes heißt *erfolgreich*, wenn einer der folgenden Fälle auftritt:

1. Das Blatt ist von der Form $\mathbf{p} \vdash^{\mathcal{D}} [a]\varphi$.
2. Das Blatt ist von der Form $\mathbf{p} \vdash^{\mathcal{D}} U$, wobei $(U = \nu X.\varphi) \in \mathcal{D}$.

Abbildung 3.3: Erfolgsbedingungen für Blätter

Definition 3.1 (erfolgreiches Tableau) *Ein Tableau heißt* erfolgreich*, wenn alle seine Blätter erfolgreich sind.*

In Fall 1 gibt es nach Abbildung 3.2 kein $\mathbf{p}'$ mit $\mathbf{p} \xrightarrow{a} \mathbf{p}'$. Damit gilt $\mathbf{p} \models^{\mathcal{D}} [a]\varphi$ und das Blatt ist gemäß der Semantik von $[a]\varphi$ erfolgreich.

Im zweiten Fall liegt eine erfolgreiche Fixpunktinduktion vor. Oberhalb des Blattes gibt es nach Abbruchbedingung 3 aus Abbildung 3.2 einen Knoten, der mit $\mathbf{p} \vdash^{\mathcal{D}'} U$ beschriftet ist. Der eindeutige Nachfolgerknoten davon ist $\mathbf{p} \vdash^{\mathcal{D}'} \varphi[X := U]$. Also gibt es in dem Tableau einen Ast von $\mathbf{p} \vdash^{\mathcal{D}'} \varphi[X := U]$ zu $\mathbf{p} \vdash^{\mathcal{D}} U$. Von unten nach oben gelesen bedeutet dies, daß man einen Beweis dafür hat, daß aus $\mathbf{p} \models^{\mathcal{D}} U$ $\mathbf{p} \models^{\mathcal{D}'} \varphi[X := U]$ folgt, wenn man im Augenblick davon absieht, daß sich die Ableitung auch verzweigen kann. Die Verallgemeinerung auf den verzweigten Fall stellt keine wesentliche Komplizierung dar. Stellt man sich U nicht als Abkürzung für die Fixpunktformel $\nu X.\varphi$ sondern für deren n-te Approximation vor, so hat man ebenfalls eine Beweis dafür, daß aus der Annahme, daß $\mathbf{p}$ diese n-te Approximation erfüllt ($\mathbf{p} \vdash^{\mathcal{D}} U$), folgt, daß $\mathbf{p}$ ebenfalls die (n+1)-te Approximation erfüllt ($\mathbf{p} \vdash^{\mathcal{D}'} \varphi[X := U]$). Da $\mathbf{p}$ trivialerweise $\nu^0 X.\varphi = \mathrm{tt}$ erfüllt, so folgt daraus, daß

$\forall n \in \omega.\mathbf{p} \models \nu^n X.\varphi$. Da φ aufgrund der Endlichkeit von $\mathbf{p}$ stetig ist, so ist dies gleichbedeutend mit $\mathbf{p} \models \nu X.\varphi$.

Durch Regeln, Abbruchbedingungen und die Definition des Erfolges ist der Model-Checker vollständig beschrieben. Als Notation für die beweistheoretische Implikation führen wir das übliche $\vdash$ ein, das heißt, $\mathbf{p} \vdash \varphi$ wenn es ein erfolgreiches Tableau mit Wurzel $\mathbf{p} \vdash^{\epsilon} \varphi$ gibt.

Theorem 3.2 (Korrektheit und Vollständigkeit)

$$\mathbf{p} \models \varphi \ \textit{genau dann wenn} \ \mathbf{p} \vdash \varphi.$$

In [SW89] findet sich der Beweis für dieses Theorem.

Die Behandlung der Fixpunkte

Nachdem wir nun den Model-Checker vorgestellt haben, wollen wir uns noch etwas ausführlicher mit der Behandlung der Fixpunkte beschäftigen, insbesondere mit der Frage, welchem Zweck die Konstanten dienen. Die Regeln aus Abbildung 3.1 schreiben vor, daß man für eine Fixpunktformel eine Konstante einführt (Regeln (μ) und (ν)) über die dann die Expansion des Fixpunktes erfolgt (Regel ($Konst$)).

Die naheliegende Behandlung von Fixpunktformeln wäre, daß man als Unterziel einer zu zeigenden Fixpunkteigenschaft, zum Beispiel $\mathbf{p} \vdash \mu X.\varphi$, zu beweisen versucht, daß $\mathbf{p}$ dann die Abwicklung der Fixpunktformel erfüllt. Anstelle von Regel (μ) aus Abbildung 3.1 hätte man dann folgende einfachere Regel, die ohne Konstanten auskommt:

$$\frac{\mathbf{p} \vdash \mu X.\varphi}{\mathbf{p} \vdash \varphi[X := \mu X.\varphi]}$$

Im wesentlichen werden Fixpunkte ja auch auf diese Art behandelt, nämlich abgewickelt. Würde man aber tatsächlich auf diese einfache Weise verfahren, wie oben angedeutet, so würde das System inkorrekt und unvollständig [SW89]. Dieser Mangel tritt erst auf, wenn die Fixpunkte in der Formel mindestens bis zur Tiefe zwei geschachtelt auftreten. Liegt zum Beispiel folgende Situation vor: das zu zeigende Ziel sei $\mathbf{p} \vdash \psi$ wobei $\psi = \mu X.\nu Y.\varphi(X, Y)$. Als nächste zwei Unterziele bekäme man zunächst $\mathbf{p} \vdash \nu Y.\varphi(\psi, Y)$ und danach $\mathbf{p} \vdash \varphi(\psi, \nu Y.\varphi(\psi, Y))$. Es kann nun passieren, daß im im weiteren Verlauf der Ableitung $\mathbf{q} \vdash \psi$ als Ziel auftritt. Somit hat man dieselbe Eigenschaft wie oben *erneut* zu zeigen, nur diesmal nicht für $\mathbf{p}$ sondern für $\mathbf{q}$. Das Problem nun ist, daß dann im folgenden die Situation $\mathbf{p} \vdash \nu Y.\varphi(\psi, Y)$ wiederum auftreten kann und man an dieser Stelle abbricht, da man den Induktionsschritt der Fixpunktinduktion für $\mathbf{p}$ gezeigt zu haben scheint. Dabei ist schiefgelaufen, daß man einen Zwischenschritt im Beweis für $\mathbf{q} \vdash \psi$ für das zweite Auftreten von $\mathbf{p} \vdash \nu Y.\varphi(\psi, Y)$ gehalten hat, da zwischenzeitlich ein neuer Beweis (für die gleiche Eigenschaft ψ zwar aber für einen anderen Zustand) begonnen hat. Dies ist zu vermeiden.

Es gibt unterschiedliche Wege, dieses Problem zu lösen. Eine Möglichkeit wäre, explizit eine Liste von "Hypothesen" einzuführen, in der festgehalten wird, welche Fixpunktformeln zusammen mit welchen Zuständen aktuell für die Fixpunktinduktion herangezogen werden dürfen. In dem obigen Beispiel würde die bedeuten, daß man beim Schritt von $\mathbf{q} \vdash \psi$ nach $\mathbf{q} \vdash \nu Y.\varphi(\psi, Y)$ sich gewissermaßen dieses $\mathbf{q} \vdash \nu Y.\varphi(\psi, Y)$ als neue Hypothese merkt und, was das Entscheidende ist, aus der Menge der Hypothesen $\mathbf{p} \vdash \nu Y.\varphi(\psi, Y)$ entfernt. Genauer gesagt wird eine Formel ψ immer dann aus der Liste der Hypothese gestrichen, wenn sie als echte Unterformel in einer neu aufgenommenen Hypothese, in diesem Fall $\nu Y.\varphi(\psi, Y)$ enthalten ist. Der Model-Checker von Cleaveland [Cle90] funktioniert auf diese Weise und bildet in dieser Form auch die Grundlage des Model-Checkers in der Concurrency Workbench.

Eine andere Möglichkeit besteht darin, Konstanten als Abkürzungen für Fixpunktformeln einzuführen und zwar bei jedem Auftreten einer solchen Formel eine neue. Dies verhindert, daß die angedeuteten Verwechselungen auftreten. Der Model-Checker basierend auf dieser Idee stammt von Stirling und Walker [SW89] und in analoger Form haben wir ihn hier auch vorgestellt. Das Beweissystem, welches wir in Kapitel 4 vorstellen werden, benutzt Konstanten als Abkürzungen für Fixpunkformeln in gleicher Weise.

3.3 Beispiele

Zum Abschluß des Kapitels wollen wir anhand eines Beispiels die Arbeitsweise des Model-Checkers dargestellen. Es soll eine Eigenschaft eines einfachen Transitionssystems nachgewiesen werden. Der Prozeß, dargestellt durch das Transitionssystem aus Abbildung 3.4 mit Anfangszustand $\mathbf{p}$, soll das Verhalten eines Kaffee- und Teeautomaten modellieren. Von diesem Automaten soll folgende Eigenschaft nachgewiesen werden:

"In allen möglichen Abläufen bietet der Automat unendlich oft Tee an".

Zunächst muß diese Eigenschaft in μHML ausgedrückt werden. "Der Automat bietet Tee an" heißt, es gibt einen mit "Tee" beschrifteten Übergang wofür in μHML die Formel $\langle Tee\rangle \mathrm{tt}$ steht. Etwas komplizierter wird schon die Eigenschaft, daß auf allen Pfaden diese Eigenschaft unendlich oft auftritt. Intuitiv heißt "unendlich oft" dasselbe wie "immer wieder φ" oder, anders ausgedrückt "es ist *immer* der Fall, daß *irgendwann* φ zutrifft", wobei φ für $\langle Tee\rangle \mathrm{tt}$ steht.

Zunächst wenden wir uns dem "für alle Pfade gilt immer φ_1" zu. Das bedeutet, jetzt gilt φ_1 und nach allen Übergängen gilt φ_1 wiederum und nach allen weiteren Übergängen erneut usw. Als rekursive Formel heißt dies:

$$\nu X.\varphi_1 \wedge [-]X$$

Nun zu "Auf allen Pfaden gilt irgendwann φ_2" oder etwas näher an der rekursiven

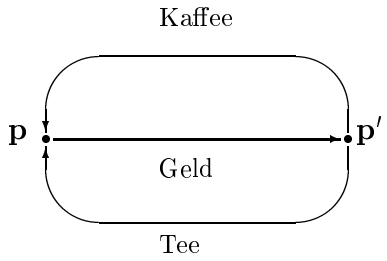


Abbildung 3.4: Kaffee - und Teeautomat

μHML-Formel: "Es gilt φ_2 jetzt oder nach allen Übergängen gilt φ_2 irgendwann". Das legt folgende Formulierung nahe:

$$\mu X.\varphi_2 \vee [-]X.$$

Dies ist jedoch nicht ganz das Gewünschte. Zum Beispiel erfüllt ein Transitionssystem mit nur einem Zustand und ohne Übergänge diese Formel, auch wenn der einzige Zustand die Eigenschaft φ_2 nicht besitzt. Der Grund liegt darin, daß ein Zustand die Formel $[-]\psi$ insbesondere auch dann erfüllt, wenn er keine Übergänge besitzt. Was also tatsächlich formuliert wurde, ist "Auf allen Pfaden gilt irgenwann, daß φ_2 zutrifft oder daß keine Fortsetzung mehr möglich ist". Dies läßt sich leicht dadurch beheben, daß man mittels $\langle - \rangle \mathrm{tt}$ explizit die Existenz eines Überganges fordert. Damit bekommt man:

$$\mu X.\varphi_2 \vee (\langle - \rangle \mathrm{tt} \wedge [-]X)$$

Setzt man beide Formel zusammen, so erhält man die gewünschte Formel:

$$\nu X.(\mu Y.\langle Tee \rangle \mathrm{tt} \vee (\langle - \rangle \mathrm{tt} \wedge [-]Y)) \wedge [-]X$$

Diese Eigenschaft soll vom Prozeß **p** des Transitionssystems aus Abbildung 3.4 nachgewiesen werden. Mit Hilfe der Regeln aus Abbildung 3.1 generiert man nun das Tableau. An dessen Wurzel steht das zu zeigende Ziel. Diese Ableitung ist in Abbildung 3.5 dargestellt.

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathbf{p} \vdash^{\epsilon} \nu X.(\mu Y.\langle Tee\rangle \mathrm{tt} \vee (\langle - \rangle \mathrm{tt} \wedge [-]Y)) \wedge [-]X}{\mathbf{p} \vdash^{\mathcal{D}_1} U}}{\mathbf{p} \vdash^{\mathcal{D}_1} (\mu Y.\langle Tee\rangle \mathrm{tt} \vee (\langle - \rangle \mathrm{tt} \wedge [-]Y)) \wedge [-]U}}{\dfrac{\mathbf{p} \vdash^{\mathcal{D}_1} \mu Y.\langle Tee\rangle \mathrm{tt} \vee (\langle - \rangle \mathrm{tt} \wedge [-]Y)}{\mathbf{p} \vdash^{\mathcal{D}_2} V_1}}}{\dfrac{\mathbf{p} \vdash^{\mathcal{D}_1} \langle Tee\rangle \mathrm{tt} \vee (\langle - \rangle \mathrm{tt} \wedge [-]V_1)}{\mathbf{p} \vdash^{\mathcal{D}_1} \langle - \rangle \mathrm{tt} \wedge [-]V_1}}$$

$$\dfrac{\dfrac{\mathbf{p} \vdash^{\mathcal{D}_1} \langle - \rangle \mathrm{tt}}{\mathbf{p}' \vdash^{\mathcal{D}_1} \mathrm{tt}} \qquad \dfrac{\mathbf{p} \vdash^{\mathcal{D}_1} [-]V_1}{\mathbf{p} \vdash^{\mathcal{D}_1} V_1}}{\dfrac{\mathbf{p}' \vdash^{\mathcal{D}_1} \langle Tee\rangle \mathrm{tt} \vee (\langle - \rangle \mathrm{tt} \wedge [-]V_1)}{\mathbf{p} \vdash^{\mathcal{D}_1} \langle Tee\rangle \mathrm{tt}}}$$

$$\dfrac{\mathbf{p} \vdash^{\mathcal{D}_1} \mathrm{tt}}{}$$

$$\dfrac{\dfrac{\dfrac{\mathbf{p} \vdash^{\mathcal{D}_1} [-]U}{\mathbf{p}' \vdash^{\mathcal{D}_1} U}}{\mathbf{p} \vdash^{\mathcal{D}_1} (\mu Y.\langle Tee\rangle \mathrm{tt} \vee (\langle - \rangle \mathrm{tt} \wedge [-]Y)) \wedge [-]U}}{\dfrac{\mathbf{p}' \vdash^{\mathcal{D}_1} \mu Y.\langle Tee\rangle \mathrm{tt} \vee (\langle - \rangle \mathrm{tt} \wedge [-]Y)}{\mathbf{p}' \vdash^{\mathcal{D}_2} V_2}}$$

$$\dfrac{\dfrac{\mathbf{p}' \vdash^{\mathcal{D}_1} [-]U}{\mathbf{p} \vdash^{\mathcal{D}_1} U}}{\mathbf{p}' \vdash^{\mathcal{D}_2} \langle Tee\rangle \mathrm{tt} \vee (\langle - \rangle \mathrm{tt} \wedge [-]V_2)}$$

$$\dfrac{\mathbf{p}' \vdash^{\mathcal{D}_2} \langle Tee\rangle \mathrm{tt}}{\mathbf{p} \vdash^{\mathcal{D}_2} \mathrm{tt}}$$

Abbildung 3.5: Beispielableitung

$$\dfrac{\dfrac{\mathbf{q} \vdash^{\mathcal{D}} \nu X.X}{\mathbf{q} \vdash^{\mathcal{D}'} U}}{\mathbf{q} \vdash^{\mathcal{D}'} U}$$

Abbildung 3.6: Eine Ableitung für $\mathbf{q} \vdash \mathrm{tt}$

Eine kleine Ungenauigkeit bleibt noch. In den Formeln haben wir die Proposition tt verwendet, obwohl μHML nach Definition 2.24 diese gar nicht enthält. Vereinbarungsgemäß steht tt für die μHML-Formel $\mu X.X$, jedoch muß noch nachgewiesen werden, daß man, wie es in der Ableitung geschehen ist, tatsächlich an Stellen $\mathbf{q} \vdash \mathrm{tt}$ abbrechen darf, genauergesagt, daß man an diesen Stellen die Ableitung erfolgreich fortsetzen könnte. Dies ist aus Abbildung 3.6 ersichtlich. Dabei steht $\mathcal{D}'$ für $\mathcal{D} \cdot (U = \nu X.X)$. Damit ist das Blatt nach den Kriterien aus Abbildung 3.3 erfolgreich, unabhängig von der Wahl von $\mathbf{q}$. Entsprechend läßt sich leicht zeigen, daß für $\mathbf{q} \vdash \mathrm{ff}$ kein erfolgreicher Ableitungsbaum existiert, unabhängig von der Wahl von $\mathbf{q}$. Dabei steht ff als Abkürzung für die Fixpunktformel $\mu X.X$.

Kapitel 4

Beweissystem für μHML

4.1 Einleitung

In diesem Kapitel kommen wir nun zum Kern der Arbeit, dem Beweissystem für μHML. Besondere Sorgfalt erfordert dabei, wie zu erwarten, wiederum die Behandlung der Fixpunkte. Wir werden sie in ähnlicher Weise wie im Model-Checker behandeln. Damit bekommen wir ein Gentzen-Sytem erweitert um die Behandlung der Fixpunkte. Es erlaubt, im Gegensatz zu Hilbert-Systemen, zielgerichtetes Vorgehen bei der Beweisführung. Für den propositionalen μ-Kalkül (mit geringfügiger Einschränkung) ist bisher nur ein Hilbert-System bekannt [Koz83].

Im nächsten Abschnitt stellen wir zunächst das Beweissystem selbst dar, also die Regeln, die Abbruchkriterien sowie die Erfolgsbedingungen. In Abschnitt 4.3 zeigen wir dann die Endlichkeit des Beweissystemes, bevor wir in Abschnitt 4.4 die Korrektheit zeigen. In Abschnit 4.5 stellen wir ein äquivalentes, aber effizienteres Beweissystem vor. Das Kapitel schließt mit einem etwas ausführlicheren Beispiel, an dem wir die Arbeitsweise des Beweissystems demonstrieren.

4.2 Das Beweissystem

Aussgehend von der Definition der Erfülltheitsrelation $\models_\tau$ soll nun der Begriff der semantischen Folgerung (ebenfalls durch $\models_\tau$ symbolisiert) eingeführt werden. Zwei Formalisierungen der semantischen Folgerungen bezüglich Transitionssystemen sind üblich.

Definition 4.1 *T stehe für ein einzelnes Transitionssystem und $\mathcal{T}$ stehe für eine Menge von Transitionssystemen.*

$$
\begin{array}{lcll}
T \models \varphi & :\Leftrightarrow & \forall \mathbf{p} \in \mathcal{P}_T \,.\, \mathbf{p} \models_T \varphi & \\
\varphi \models_{\mathcal{T}} \psi & :\Leftrightarrow & \forall T \in \mathcal{T}. & \textit{wenn } T \models \varphi \textit{ dann } T \models \psi \\
\varphi \models_\tau \psi & :\Leftrightarrow & \forall T \in \mathcal{T} . \forall \mathbf{p} \in \mathcal{P}_T \,. & \textit{wenn } \mathbf{p} \models_T \varphi \textit{ dann } \mathbf{p} \models_T \psi
\end{array}
$$

Entsprechend sei $\Gamma \models_{\mathcal{T}} \psi$ bzw. $\Gamma \models_{\tau} \psi$ definiert, wobei Γ für eine Menge von μHML- Formeln steht. $\models_{\mathcal{T}}$ wird als *globale*, $\models_{\tau}$ als *lokale* semantische Implikation bezeichnet. Offensichtlich gilt, wenn $\varphi \models_{\tau} \psi$ dann $\varphi \models_{\mathcal{T}} \psi$. Wir interessieren uns im folgenden nur für den lokalen Folgerungsbegriff, da durch ihn logische Folgerungen bezüglich einzelner Zustände der Transitionssysteme, die wir als Prozesse ansehen, ausgedrückt werden.

Die Implikationsrelation $\models_{\tau}$ (wie auch $\models_{\mathcal{T}}$) ist mit einer Menge von Transitionssystemen parametrisiert. Je nach Art von Bedingungen, mit denen man $\mathcal{T}$ einschränkt, bekommt man unterschiedliche Relationen $\models_{\tau}$ und $\models_{\mathcal{T}}$. Wählt man zum Beispiel $\mathcal{T}_R$ als eine Menge von Transitionssystemen, deren Übergangsrelation $\xrightarrow{a}$ reflexiv ist, so gilt $[a]\varphi \models_{\tau_R} \varphi$, eine Implikation, die nicht für beliebige Transitionssysteme gilt. Wir nehmen im weiteren, sofern nicht eigens betont, keine Einschränkungen der Form der Transitionssysteme an, so daß $\mathcal{T}$ immer für die Klasse aller Transitionssysteme (über einem Alphabet Act) steht. Insbesondere schreiben wir, wenn $\mathcal{T}$ aus dem Kontext klar ist, oft $\models$ anstelle von $\models_{\tau}$.

In diesem Abschnitt nun werden wir den Kern unserer Arbeit vorstellen, ein Beweissystem für μHML. Dieses System ist ein Gentzen-artiger Sequenzenkalkül, der es erlaubt, zielgerichtete Beweise für die lokale semantische Implikation zu führen. Er basiert, wie die im Kapitel 3 behandelten Model-Checker auf dem Prinzip der Fixpunktinduktion, um die Fixpunktformeln zu behandeln.

Bevor wir das System selbst vorstellen, noch einige notationelle Vereinbarungen. Das Beweissystem benutzt Sequenzen der Form $\Gamma \vdash^{\mathcal{D}} \Delta$, wobei $\mathcal{D}$, wie im Model-Checker, für eine Liste von Konstantendeklarationen steht. Γ und Δ bezeichnen endliche Mengen von geschlossenen μHML-Formeln, die Konstanten enthalten dürfen, die in $\mathcal{D}$ definiert sind. Wenn $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_n\}$ und $\Delta = \{\delta_1, \delta_2, \ldots, \delta_m\}$, so steht die Sequenz $\Gamma \vdash^{\mathcal{D}} \Delta$ für das zu zeigende Ziel $\bigwedge_{i=1}^{n}\gamma_i \models^{\mathcal{D}} \bigvee_{j=1}^{m}\delta_i$, oder anders ausgedrückt für $[\![\bigwedge_{i=1}^{n}\gamma_i]\!] \subseteq [\![\bigvee_{j=1}^{m}\delta_i]\!]$. Das bedeutet, Mengen auf der linken Seite des $\vdash^{\mathcal{D}}$ stehen für die Konjunktion iher Formeln, Mengen auf der linken Seite für ihre Disjunktion. Dabei sei auf zwei Sonderfälle hingewiesen. Ist Γ beziehungsweise Δ leer, so steht Γ für die leere Konjunktion, also für tt, respektive Δ für die leere Disjunktion, also für ff. Als weitere Abkürzung verwenden wir $\wedge\Gamma$ für $\bigwedge_{i=1}^{n}\gamma_i$ und $\vee\Delta$ für $\bigvee_{j=1}^{m}\delta_i$.

$\equiv$ stehe für die syntaktische Identität von μHML-Formeln (incl. Konstanten). Ferner sei $\mathcal{D}$ eine Deklarationsliste für Fixpunktformeln. $\mathcal{D}(\varphi)$ stehe dann für die Formel, bei der alle Konstanten durch die Fixpunktformeln entsprechend ihrer Deklaration in $\mathcal{D}$ ersetzt werden, bis die Formel keine Konstanten mehr enthält. Für Formelmengen Δ sei $\mathcal{D}(\Delta)$ elementweise definiert. Ferner stehe die semantische Implikation bezüglich einer Konstantendeklaration $\Gamma \models^{\mathcal{D}} \Delta$ als Abkürzung für $\mathcal{D}(\Gamma) \models \mathcal{D}(\Delta)$.

Die Regeln

Mit Hilfe der Regeln des Beweissystems, die in Abbildung 4.1 zusammengefaßt sind, generiert man Ableitungsbäume oder Tableaus, deren Wurzel mit der zu beweisenden Implikation beschriftet ist.

Die ersten vier Regeln sind die üblichen Regeln zur Behandlung der logischen Konnektive $\wedge$ und $\vee$ in Gentzensystemen. Die Regeln $(\wedge \vdash)$ und $(\vdash \vee)$ sind dabei nur syntaktische Umformungen, da eine Formelmenge auf der linken Seite des $\vdash$ ohnehin für die Konjunktion und auf der rechten Seite für die Disjunktion ihrer Formeln steht. Die Behandlung der Disjunktion in Regel $(\vdash \vee)$ vergleiche man insbesondere mit der Regel für das $\vee$ im Modelchecker aus 3.1.

Die Regeln $(\vdash \wedge)$ und $(\vee \vdash)$ spalten ein Ziel in je zwei Unterziele auf. Zum Beispiel werden aus dem Ziel, '$\wedge\Gamma$ impliziert $\vee\Delta \vee (\varphi_1 \wedge \varphi_2)$' , die zwei Unterziele '$\wedge\Gamma$ impliziert $\vee\Delta \vee \varphi_1$' und '$\wedge\Gamma$ impliziert $\vee\Delta \vee \varphi_2$'.

Die Regeln $(weak_1)$ und $(weak_2)$ sind Abschwächungsregeln für die linke und die rechte Seite. Anstelle des Zieles ist es ausreichend, die abgeschwächten Unterziele zu zeigen.

Die Regeln $(\langle a \rangle \vdash)$ und $(\vdash [a])$ sind dual zueinander und befassen sich mit der Behandlung der Modaloperatoren $[a]$ und $\langle a \rangle$. Mit der Regel $(\langle a \rangle \vdash)$ zum Beispiel will man aus der Existenz eines a-Übergangs, nach dem φ gilt, sowie der Tatsache, daß nach allen a-Übergängen die Formeln aus Γ gelten, folgern, daß es einen a-Übergang gibt, nach dem eine der Formeln aus Δ gilt. Aus der Vorraussetzung weiß man, daß es einen a-Übergang gibt, nach dem sowohl φ als auch alle γ aus Γ gelten. Kann man dann das Unterziel $\Gamma, \varphi \vdash^{\mathcal{D}} \Delta$ zeigen, so gilt nach diesem a-Übergang eine der Formeln δ aus Δ. Damit ist das ursprüngliche Ziel gezeigt.

Die letzten vier Regeln behandeln die Fixpunktoperatoren μ und ν und zwar in ähnlicher Weise, wie dies beim Model-Checker in Kapitel 3 geschehen ist. Beim Auftreten von Fixpunktformeln auf der linken oder auf der rechten Seite einer Sequenz werden neue Konstanten eingeführt (Regeln $(\sigma \vdash)$ und $(\vdash \sigma)$), deren Deklaration in $\mathcal{D}'$ festgehalten wird. Die Konstanten können dann nach den Regeln $(Konst \vdash)$ und $(\vdash Konst)$ gemäß ihrer Deklaration expandiert werden, was der Abwicklung der Fixpunke entspricht. Kleinste und größte Fixpunkte werden von den Regeln dabei auf beiden Seiten des $\vdash$ völlig gleichbehandelt. Der Unterschied zwischen den Fixpunkten auf der linken und der rechten Seite der Sequenz tritt später bei der Beurteilung des Erfolges einer Ableitung auf.

$$(\wedge \vdash) \quad \frac{\Gamma, \varphi_1 \wedge \varphi_2 \vdash^{\mathcal{D}} \Delta}{\Gamma, \varphi_1, \varphi_2 \vdash^{\mathcal{D}} \Delta}$$

$$(\vdash \vee) \quad \frac{\Gamma \vdash^{\mathcal{D}} \Delta, \varphi_1 \vee \varphi_2}{\Gamma \vdash^{\mathcal{D}} \Delta, \varphi_1, \varphi_2}$$

$$(\vdash \wedge) \quad \frac{\Gamma \vdash^{\mathcal{D}} \Delta, \varphi_1 \wedge \varphi_2}{\Gamma \vdash^{\mathcal{D}} \Delta, \varphi_1 \quad \Gamma \vdash^{\mathcal{D}} \Delta, \varphi_2}$$

$$(\vee \vdash) \quad \frac{\Gamma, \varphi_1 \vee \varphi_2 \vdash^{\mathcal{D}} \Delta}{\Gamma, \varphi_1 \vdash^{\mathcal{D}} \Delta \quad \Gamma, \varphi_2 \vdash^{\mathcal{D}} \Delta}$$

$$(weak_1) \quad \frac{\Gamma, \varphi \vdash^{\mathcal{D}} \Delta}{\Gamma \vdash^{\mathcal{D}} \Delta}$$

$$(weak_2) \quad \frac{\Gamma \vdash^{\mathcal{D}} \Delta, \varphi}{\Gamma \vdash^{\mathcal{D}} \Delta}$$

$$(\langle a \rangle \vdash) \quad \frac{\langle a \rangle \varphi, \{[a]\gamma; \gamma \in \Gamma\} \vdash^{\mathcal{D}} \{\langle a \rangle \delta; \delta \in \Delta\}}{\varphi, \Gamma \vdash^{\mathcal{D}} \Delta} \qquad a \in \mathsf{Act}$$

$$(\vdash [a]) \quad \frac{\{[a]\gamma; \gamma \in \Gamma\} \vdash^{\mathcal{D}} [a]\varphi, \{\langle a \rangle \delta; \delta \in \Delta\}}{\Gamma \vdash^{\mathcal{D}} \varphi, \Delta} \qquad a \in \mathsf{Act}$$

$$(\sigma \vdash) \quad \frac{\Gamma, \sigma X.\varphi \vdash^{\mathcal{D}} \Delta}{\Gamma, U \vdash^{\mathcal{D}'} \Delta}$$

$\sigma X.\varphi \in \{\nu X.\varphi, \mu X.\varphi\}$
$\mathcal{D}' = \mathcal{D} \cdot (U = \sigma X.\varphi)$
Uneue Konstante

$$(\vdash \sigma) \quad \frac{\Gamma \vdash^{\mathcal{D}} \Delta, \sigma X.\varphi}{\Gamma \vdash^{\mathcal{D}} \Delta, U}$$

$\sigma X.\varphi \in \{\nu X.\varphi, \mu X.\varphi\}$
$\mathcal{D}' = \mathcal{D} \cdot (U = \sigma X.\varphi)$
Uneue Konstante

$$(Konst \vdash) \quad \frac{\Gamma, U \vdash^{\mathcal{D}} \Delta}{\Gamma, \varphi[X := U] \vdash^{\mathcal{D}} \Delta}$$

$\sigma X.\varphi \in \{\nu X.\varphi, \mu X.\varphi\}$
$(U = \sigma X.\varphi) \in \mathcal{D}$

$$(\vdash Konst) \quad \frac{\Gamma \vdash^{\mathcal{D}} U, \Delta}{\Gamma \vdash^{\mathcal{D}} \varphi[X := U], \Delta}$$

$\sigma X.\varphi \in \{\nu X.\varphi, \mu X.\varphi\}$
$(U = \sigma X.\varphi) \in \mathcal{D}$

Abbildung 4.1: Regeln des Beweissystems

Die Abbrüche

Um das Beweissystem zu vervollständigen, muß man, wie beim Model-Checker, noch angeben, wann man mit der Anwendung der Regeln, das heißt der Generierung von neuen Unterzielen, abbricht und wann man ein maximales Tableau als erfolgreich betrachtet.

Beim Abbruch der Regelanwendung unterscheidet man vier Fälle (siehe Abbildung 4.2):

1. $\Gamma, \varphi \vdash^{\mathcal{D}} \Delta, \varphi$
2. $\emptyset \vdash^{\mathcal{D}} \emptyset$
3. Wiederholung einer Sequenz
   (a) Man trifft auf eine Sequenz $\Gamma, U \vdash^{\mathcal{D}} \Delta$ und oberhalb im Baum ist bereits ein Knoten vorhanden, der mit $\Gamma, U \vdash^{\mathcal{D}'} \Delta$ beschriftet ist.
   (b) Man trifft auf eine Sequenz $\Gamma \vdash^{\mathcal{D}} U, \Delta$ und oberhalb im Baum ist bereits ein Knoten vorhanden, der mit $\Gamma \vdash^{\mathcal{D}'} U, \Delta$ beschriftet ist.
4. Abbruch vor der unnützen Einführung einer neuen Konstante
   (a) Man trifft auf einen Knoten der Form $\Gamma, \sigma X.\varphi \vdash^{\mathcal{D}} \Delta$ und oberhalb im Tableau existiert ein Knoten der Form $\Gamma', \sigma X.\varphi \vdash^{\mathcal{D}'} \Delta'$ wobei $\sigma X.\varphi \in \{\nu X.\varphi, \mu X.\varphi\}$, $\mathcal{D}(\Gamma) = \mathcal{D}'(\Gamma')$ und $\mathcal{D}(\Delta) = \mathcal{D}'(\Delta')$.
   (b) Man trifft auf einen Knoten der Form $\Gamma \vdash^{\mathcal{D}} \sigma X.\varphi, \Delta$ und oberhalb im Tableau existiert ein Knoten der Form $\Gamma' \vdash^{\mathcal{D}'} \sigma X.\varphi, \Delta'$ wobei $\sigma X.\varphi \in \{\nu X.\varphi, \mu X.\varphi\}$, $\mathcal{D}(\Gamma) = \mathcal{D}'(\Gamma')$ und $\mathcal{D}(\Delta) = \mathcal{D}'(\Delta')$.

Abbildung 4.2: Abbruchbedingungen des Beweissystemes

In Fall 1 bricht man ab, da man auf ein wahres Blatt gestoßen ist. Bei Fall 2 muß man notgedrungen abbrechen, da keine weitere Regel mehr anwendbar ist. Ein solches Blatt entspricht semantisch $\mathrm{tt} \models \mathrm{ff}$, das heißt das Blatt ist erfolglos.

3(a) und 3(b) sind die interessanten Fälle. Man ist bei der Ableitung auf eine Situation gestoßen, die im Baum schon einmal aufgetreten ist. Da man durch Fortsetzung der Ableitung keine weiteren Informationen mehr erhält, die man nicht bereits nach dem ersten Auftreten der Sequenz hätte bekommen können, bricht man hier ab. Ob ein solches Blatt erfolgreich ist oder nicht, hängt von der Art des Fixpunktes und von der Ableitung selber ab.

Im Fall 4(a) (oder analog 4(b)) steht man vor der Einführung einer neuen Konstante nach Regel $(\sigma \vdash)$ oder $(\vdash \sigma)$ aus Abbildung 4.1. Diese Neueinführung ist jedoch nicht sinnvoll, da die Konstante, die man nach dem Knoten $\Gamma, \sigma X.\varphi \vdash^{\mathcal{D}} \Delta$ für $\sigma X.\varphi$ einführen kann, in einem "Kontext" Γ und Δ eingeführt würde, der zu dem "Kontext" Γ', Δ' identisch ist. Γ und Γ' unterscheiden sich nur durch eventuell andere Konstanten als Abkürzungen für Fixpunktformeln. Das gleiche gilt auch für Δ und Δ'. Syntaktisch unterscheiden sich die Mengen jedoch und nach den Regeln wäre man gezwungen, eine neue Konstante für $\sigma X.\varphi$ einzuführen. Wenn man dies zuläßt, kann es passieren, daß auf beiden Seiten des $\vdash$ wechselseitig immer neue Konstanten eingeführt werden, ohne daß der Beweis Fortschritte macht. Ohne diesen Abbruch wären also unendliche Ableitungsbäume möglich.

Der Erfolg

In Abbildung 4.3 sind die Erfolgskriterien zusammengefaßt.

Ein Blatt eines maximalen Tableaus heißt *erfolgreich*, wenn einer der folgenden Fälle auftritt. Das Blatt ist von der Form:

1. $\Gamma, \varphi \vdash^{\mathcal{D}} \varphi, \Delta$
2. Erfolgreiche Fixpunktinduktion
   (a) $\Gamma, U \vdash^{\mathcal{D}} \Delta$ wobei $(U = \mu X.\varphi) \in \mathcal{D}$ und zwischen dem erstmaligen Auftreten dieser Sequenz $\Gamma, U \vdash^{\mathcal{D}'} \Delta$ und dem Blatt $\Gamma, U \vdash^{\mathcal{D}} \Delta$ wurde für U Regel 11 aus Abbildung 4.1 mindestens einmal angewendet.
   (b) $\Gamma \vdash^{\mathcal{D}} U, \Delta$ wobei $(U = \nu X.\varphi) \in \mathcal{D}$ und zwischen dem erstmaligen Auftreten dieser Sequenz $\Gamma \vdash^{\mathcal{D}'} U, \Delta$ und dem Blatt $\Gamma \vdash^{\mathcal{D}} U, \Delta$ wurde für U Regel 12 aus Abbildung 4.1 mindestens einmal angewendet.

Abbildung 4.3: Erfolgsbedingungen für Blätter

Blätter der Form 1 sind immer erfolgreich, da sie für die Implikation $\wedge\Gamma \wedge \varphi \models^{\mathcal{D}} \varphi \vee \vee\Delta$ stehen. Die Blätter der Fälle 2(a) und 2(b) sind diejenigen, bei denen der Erfolg durch Fixpunktinduktion gewährleistet ist. Die Nebenbedingung, daß der Fixpunkt zwischen dem ersten und dem zweiten Auftreten der Sequenz einmal abgewickelt wurde, ist nötig, da diese Abwicklung dem Induktionsschritt bei der Fixpunktinduktion entspricht. Im einfacheren und in der Anwendung der Regeln determinitischeren Model-Checker ist diese Bedingung automatisch erfüllt.

Definition 4.2 (Erfolgreiches Tableau) *Ein maximales Tableau heißt* erfolgreich*, wenn alle seine Blätter erfolgreich sind.*

Definition 4.3 (beweistheoretische Implikation)

$$\Gamma \vdash \Delta \quad :\Leftrightarrow \quad \textit{Es gibt ein erfolgreiches Tableau mit } \Gamma \vdash^{\epsilon} \Delta \textit{an der Wurzel.}$$

Damit können wir das Hauptergebnis unserer Arbeit formulieren, die Korrektheit des Beweissystems:

Theorem 4.4 (Korrektheit)

$$\textit{Wenn } \Gamma \vdash \Delta \textit{ dann } \Gamma \models \Delta.$$

In Abschnitt 4.4 beweisen wir dieses Theorem. Dazu zeigen wir zunächst, daß alle Tableaus endlich sind.

4.3 Die Endlichkeit

Das Ziel dieses Abschnittes ist es, zu zeigen, daß es nur endliche Tableaus gibt, erfolgreiche wie erfolglose. Das bedeutet, daß auf allen Zweigen früher oder später eines der angegebenen Abbruchkriterien die Fortsetzung der Ableitung verhindert. Die Skizze des Beweises sieht wie folgt aus: Jede Formel generiert im Laufe einer Ableitung nur endlich viele semantisch unterschiedliche Unterformeln. Damit werden für einen Fixpunkt aufgrund der Abbruchkriterien aus Abbildung 4.2 unter Punkt 4 nur endlich viele verschiedene Konstanten eingeführt. Somit entstehen aus einer Formel im Laufe einer Ableitung nur endlich viele syntaktisch unterschiedliche Objekte und es wird irgendwann gemäß Abbruch 3 aus Abbildung 4.2 abgebrochen.

Definition 4.5 (syntaktisch äquivalent) *Zwei Formeln* φ_1 *und* φ_2 *heißen* syntaktisch äquivalent *bezüglich einer Konstantendeklaration* $\mathcal{D}$*, wenn folgendes gilt:*

$$\varphi_1 \simeq_{\mathcal{D}} \varphi_2 \quad :\Leftrightarrow \quad \mathcal{D}(\varphi_1) \equiv \mathcal{D}(\varphi_2).$$

Zwei Sequenzen $\Gamma_1 \vdash^{\mathcal{D}_1} \Delta_1$ *und* $\Gamma_2 \vdash^{\mathcal{D}_2} \Delta_2$ *heißen* syntaktisch äquivalent*, wenn gilt:*

$$\Gamma_1 \vdash^{\mathcal{D}_1} \Delta_1 \equiv \Gamma_2 \vdash^{\mathcal{D}_2} \Delta_2 \quad :\Leftrightarrow \quad \mathcal{D}_1(\Gamma_1) \equiv \mathcal{D}_2(\Gamma_2) \textit{ und } \mathcal{D}_1(\Delta_1) \equiv \mathcal{D}_2(\Delta_2).$$

Man beachte, daß zwei syntaktisch äquivalente Formel auch semantisch übereinstimmen.

Im Laufe einer Ableitung tritt eine bestimmte μHML-Formel durch seine Unterformeln auf. Die Menge aller dieser Unterformeln ist endlich und wird als *Fischer-Ladner-Abschluß* [FL79] der Formel bezeichnet.

Definition 4.6 (Fischer-Ladner-Abschluß) *Der* Fischer-Ladner-Abschluß *einer geschlossenen μHML-Formel ist die kleinste Menge von Formeln $FL(\varphi)$, die folgende Bedingungen erfüllt:*

$$
\begin{array}{llllll}
 & \varphi & \in FL(\varphi) & & & \\
\textit{wenn} & \varphi_1 \wedge \varphi_2 & \in FL(\varphi) & \textit{dann} & \varphi_1 \in FL(\varphi) & \textit{und}\ \ \varphi_2 \in FL(\varphi) \\
\textit{wenn} & \varphi_1 \vee \varphi_2 & \in FL(\varphi) & \textit{dann} & \varphi_1 \in FL(\varphi) & \textit{und}\ \ \varphi_2 \in FL(\varphi) \\
\textit{wenn} & [a]\varphi & \in FL(\varphi) & \textit{dann} & \varphi \in FL(\varphi) & \\
\textit{wenn} & \langle a\rangle\varphi & \in FL(\varphi) & \textit{dann} & \varphi \in FL(\varphi) & \\
\textit{wenn} & \nu X.\psi(X) & \in FL(\varphi) & \textit{dann} & \psi(\nu X.\psi(X)) \in FL(\varphi) & \\
\textit{wenn} & \mu X.\psi(X) & \in FL(\varphi) & \textit{dann} & \psi(\mu X.\psi(X)) \in FL(\varphi) &
\end{array}
$$

Der Fischer-Ladner-Abschluß einer Formelmenge wird wiederum elementweise definiert. Mit Hilfe dieser Definition läßt sich folgendes Lemma beweisen.

Lemma 4.7 *Für jede Fixpunktformel der Wurzel eines Tableaus werden nur endlich viele verschiedene Konstanten eingeführt.*

Beweis zu 4.7

Jeder Knoten im Baum mit der Wurzel $\Gamma \vdash^{\epsilon} \Delta$ ist von der Form $\Gamma' \vdash^{\mathcal{D}'} \Delta'$ wobei es für jede Formel $\delta' \in \Delta'$ ein $\delta \in FL(\Delta)$ gibt mit $\delta' \simeq_{\mathcal{D}} \delta$. Das bedeutet, die Formeln aus Δ' stammen bis auf beliebige Konstanten anstelle von Fixpunkt-Unterformeln aus dem Fischer-Ladner-Abschluß von Δ. Analog gilt dies für Formeln $\gamma \in \Gamma$. Daher besteht Δ' ebenso wie Γ' immer nur aus endlich vielen, syntaktisch nicht-äquivalenten Formeln. Betrachtet man die Abbrüche 4(a) und 4(b) aus Abbildung 4.2, so bedeutet dies, daß in einem Ableitungsbaum für eine bestimmte Fixpunktformel nicht unendlich viele verschiedene Konstanten eingeführt werden können. □

Satz 4.8 (Endlichkeit der Tableaus) *Jedes Tableau ist endlich.*

Beweis zu 4.8

Angenommen, es gibt ein unendliches Tableau. Da alle Ableitungsbäume endlich verzweigt sind, bedeutet dies nach König's Lemma, daß es in dem Baum einen unendlichen Pfad gibt. Da alle Regeln bis auf diejenigen, die sich mit der Behandlung von Fixpunkten befassen, echt verkürzend wirken, treten auf diesem Pfad unendlich oft Sequenzen auf, bei denen mindestens eine Konstante U "blank", das heißt nicht echt innerhalb einer μHML-Formel, auftaucht, also Sequenzen der Form $\Gamma \vdash^{\mathcal{D}} U, \Delta$. (Ohne Beschränkung der Allgemeinheit stehe die Konstante auf der rechten Seite

von $\vdash^{\mathcal{D}}$). Wegen Lemma 4.7 wissen wir, daß die Anzahl der Konstanten im Tableau insgesamt nur endlich ist. Somit muß auf dem unendlichen Pfad mindestens eine bestimmte Konstante V unendlich oft blank aufgetaucht sein. Das heißt, auf diesem Pfad gibt es unendlich viele Sequenzen der Form $\Gamma_1 \vdash^{\mathcal{D}_1} V, \Delta_1, \Gamma_2 \vdash^{\mathcal{D}_2} V, \Delta_2, \Gamma_3 \vdash^{\mathcal{D}_3} V, \Delta_3 \ldots$. Für alle $\gamma' \in \Gamma_i$ gibt es ein $\gamma \in FL(\Gamma)$ mit $\gamma \simeq_{\mathcal{D}_i} \gamma'$. Ebenso gilt dies für die Δ_i. Die Mengen Γ_i und Δ_i enthalten also Formeln aus dem Fischer-Ladner-Abschluß der entsprechenden Formelmengen an der Wurzel des Baumes modulo beliebiger Konstanten anstelle von Fixpunktformeln. Da die Anzahl der Konstanten endlich ist, kann es nur endlich viele Mengen dieser Form geben. Damit muß sich eine der Sequenzen $\Gamma_i \vdash^{\mathcal{D}_i} V, \Delta_i$ wiederholen und an dieser Stelle wird gemäß Fall 3(b) aus Abbildung 4.2 abgebrochen. Dies steht im Widerspruch zur Unendlichkeit des Pfades. □

4.4 Die Korrektheit

Mit Hilfe der im Abschnitt 4.3 gezeigten Endlichkeit der Tableaus werden wir nun die Korrektheit des Beweissystems zeigen. Dazu benötigen wir zunächst die sogenannte *Backward-Soundness* der Regeln.

Satz 4.9 (Backward-Soundness) *Für jede Regel aus Abbildung 4.1 folgt die Erfülltheit des Zieles aus der Erfülltheit der entsprechenden Unterziele der Regel.*

Beweis zu 4.9

Für die Regeln ($\wedge \vdash$) und ($\vdash \vee$) gilt die Aussage trivialerweise, da im ersten Fall das Komma für ein $\wedge$ steht und im zweiten für ein $\vee$. Ebenso einfach ist dies für die Regeln $(weak_1)$ und $(weak_2)$: Gilt $\wedge\Gamma \models^{\mathcal{D}} \vee\Delta$ so auch $\wedge\Gamma \wedge \varphi \models^{\mathcal{D}} \vee\Delta$.
Für Regel ($\vdash \wedge$) zeigt man es folgendermaßen: Gelte $\wedge\Gamma \models^{\mathcal{D}} \varphi_1 \vee \vee\Delta$ und $\wedge\Gamma \models^{\mathcal{D}} \varphi_2 \vee \vee\Delta$. Dann gilt für alle T aus $\mathcal{T}$ und alle $\mathbf{p}$ aus $\mathcal{P}_T$:

$$
\begin{array}{ll}
 & (\text{ wenn } \mathbf{p} \models^{\mathcal{D}}_T \wedge\Gamma \text{ dann } \mathbf{p} \models^{\mathcal{D}}_T \varphi_1 \vee \vee\Delta) \text{ und} \\
 & (\text{ wenn } \mathbf{p} \models^{\mathcal{D}}_T \wedge\Gamma \text{ dann } \mathbf{p} \models^{\mathcal{D}}_T \varphi_2 \vee \vee\Delta) \\
\Rightarrow & (\mathbf{p} \not\models^{\mathcal{D}}_T \wedge\Gamma \text{ oder } \mathbf{p} \models^{\mathcal{D}}_T \varphi_1 \vee \vee\Delta) \text{ und } (\mathbf{p} \not\models^{\mathcal{D}}_T \wedge\Gamma \text{ oder } \mathbf{p} \models^{\mathcal{D}}_T \varphi_2 \vee \vee\Delta) \\
\Rightarrow & (\mathbf{p} \not\models^{\mathcal{D}}_T \wedge\Gamma) \text{ oder } (\mathbf{p} \models^{\mathcal{D}}_T \varphi_1 \vee \vee\Delta \text{ und } \mathbf{p} \models^{\mathcal{D}}_T \varphi_2 \vee \vee\Delta) \\
\Rightarrow & (\mathbf{p} \not\models^{\mathcal{D}}_T \wedge\Gamma) \text{ oder } (\mathbf{p} \models^{\mathcal{D}}_T (\varphi_1 \vee \vee\Delta) \wedge (\varphi_2 \vee \vee\Delta)) \\
\Rightarrow & (\mathbf{p} \not\models^{\mathcal{D}}_T \wedge\Gamma) \text{ oder } (\mathbf{p} \models^{\mathcal{D}}_T (\varphi_1 \wedge \varphi_2) \vee \vee\Delta) \\
\Rightarrow & \text{ wenn } \mathbf{p} \models^{\mathcal{D}}_T \wedge\Gamma \text{ dann } \mathbf{p} \models^{\mathcal{D}}_T (\varphi_1 \wedge \varphi_2) \vee \vee\Delta.
\end{array}
$$

Der Beweis für Regel ($\vee \vdash$) erfolgt analog.

Nun zu Regel ($\langle a \rangle \vdash$):
Es gelte $\varphi \wedge \wedge\Gamma \models^{\mathcal{D}} \vee\Delta$.

Zu zeigen ist: $\langle a\rangle\varphi \wedge \wedge\{[a]\gamma; \gamma \in \Gamma\} \models^{\mathcal{D}} \vee\{\langle a\rangle\delta; \delta \in \Delta\}$ d.h.

$$\forall T \in \mathcal{T}.\forall \mathbf{p} \in \mathcal{P}_T \;.\quad \begin{array}{l}\text{wenn } \mathbf{p} \models^{\mathcal{D}}_T \langle a\rangle\varphi \wedge \wedge\{[a]\gamma; \gamma \in \Gamma\} \\ \text{dann } \mathbf{p} \models^{\mathcal{D}}_T \bigvee_{\delta\in\Delta} \langle a\rangle\delta.\end{array}$$

Sei also (für beliebiges $T \in \mathcal{T}$ und beliebiges $\mathbf{p} \in \mathcal{P}_T$)

$$\begin{array}{ll} & \mathbf{p} \models^{\mathcal{D}}_T \langle a\rangle\varphi \wedge \wedge\{[a]\gamma; \gamma \in \Gamma\} \\ \Rightarrow & \mathbf{p} \models^{\mathcal{D}}_T \langle a\rangle\varphi \wedge [a]\bigwedge_{\gamma\in\Gamma}\gamma \\ \Rightarrow & \exists \mathbf{p}' \in \mathcal{P}_T \;.\mathbf{p}\xrightarrow{a}\mathbf{p}' \wedge \mathbf{p}' \models^{\mathcal{D}}_T \varphi \text{ und } \forall \mathbf{p}'' \in \mathcal{P}_T \text{ . wenn } \mathbf{p}\xrightarrow{a}\mathbf{p}'' \text{ dann } \mathbf{p}'' \models^{\mathcal{D}}_T \bigwedge_{\gamma\in\Gamma}\gamma \\ \Rightarrow & \exists \mathbf{p}' \in \mathcal{P}_T \;.\mathbf{p}\xrightarrow{a}\mathbf{p}' \text{ und } \mathbf{p}' \models^{\mathcal{D}}_T \varphi \wedge \bigwedge_{\gamma\in\Gamma}\gamma \\ \Rightarrow & \exists \mathbf{p}' \in \mathcal{P}_T \;.\mathbf{p}\xrightarrow{a}\mathbf{p}' \text{ und } \mathbf{p}' \models^{\mathcal{D}}_T \vee\Delta \\ \Rightarrow & \mathbf{p} \models^{\mathcal{D}}_T \langle a\rangle\vee\Delta \\ \Rightarrow & \mathbf{p} \models^{\mathcal{D}}_T \bigvee_{\delta\in\Delta} \langle a\rangle\delta. \end{array}$$

Der Beweis für die Regel ($\vdash [a]$) ist hierzu dual.

Bei Regel ($\sigma \vdash$) und ($\vdash \sigma$) ist wiederum nichts zu zeigen. Der Fixpunkt wird jeweils durch die entsprechende Konstante ersetzt, an der Semantik ändert sich dadurch nichts.
In Regel ($Konst \vdash$) und analog in Regel ($\vdash Konst$) wird ein Fixpunkt abgewickelt; das heißt, es wird die Formel $\nu X.\varphi(X)$, für die U steht, durch $\varphi(\nu X.\varphi(X))$ ersetzt, was $\varphi[X := U]$ entspricht. Da $\nu X.\varphi(X)$ einen Fixpunkt darstellt, gilt $[\![\, \nu X.\varphi(X) \,]\!] = [\![\, \varphi(\nu X.\varphi(X)) \,]\!]$. □

Satz 4.10 (finite - model - Eigenschaft) *Sei φ eine μHML-Formel. Ist φ erfüllbar, so bereits in einem Transitionssystem mit endlich vielen Zuständen.*

Der Beweis für diesen Satz findet sich in [SE89]. Dort wird diese Eigenschaft für den modalen μ-Kalkül gezeigt. Damit gilt sie auch für μHML, da μHML eine Unterlogik des Propositionale μ-Kalküls ist mit tt und ff als einzigen Propositionen.

Wir wollen uns nun überlegen, daß das Nichterfülltsein einer Fixpunktformel seine Ursache bereits im Nichterfülltsein einer endlichen Approximation dieser Formel findet.

Lemma 4.11

1. *Sei* $\wedge\Gamma \not\models^{\mathcal{D}} \nu X.\varphi \vee \vee\Delta$.
   Dann existiert ein $n \in \omega$ *mit*
   $\wedge\Gamma \models^{\mathcal{D}} \varphi^n(\mathrm{tt}) \vee \vee\Delta$ *und* $\wedge\Gamma \not\models^{\mathcal{D}} \varphi^{n+1}(\mathrm{tt}) \vee \vee\Delta$.

2. *Sei* $\wedge\Gamma \wedge \mu X.\varphi \not\models^{\mathcal{D}} \vee\Delta$.
   Dann existiert ein $n \in \omega$ *mit*
   $\wedge\Gamma \wedge \varphi^n(\mathrm{ff}) \models^{\mathcal{D}} \vee\Delta$ *und* $\wedge\Gamma \wedge \varphi^{n+1}(\mathrm{ff}) \not\models^{\mathcal{D}} \vee\Delta$.

Beweis zu 4.11

Zu 1):
Sei also $\wedge\Gamma \not\models^{\mathcal{D}} \nu X.\varphi \vee \vee\Delta$ und gelte als Widerspruchsannahme

$$\forall n \in \omega.\ \text{wenn}\ \ \wedge\Gamma \models \varphi^n(\mathrm{tt}) \vee \vee\Delta\ \ \text{dann}\ \ \wedge\Gamma \models \varphi^{n+1}(\mathrm{tt}) \vee \vee\Delta.$$

Da $\varphi^0(\mathrm{tt}) = \mathrm{tt}$, gilt auf jeden Fall $\wedge\Gamma \models \varphi^0(\mathrm{tt}) \vee \vee\Delta$ und per Induktion somit $\forall n \in \omega.\ \wedge\Gamma \models \varphi^n(\mathrm{tt}) \vee \vee\Delta$. Damit folgt im weiteren:

$$\begin{array}{l}\forall n \in \omega. \forall T \in \mathcal{T}. \forall \mathbf{p} \in \mathcal{P}_T\ .\ \text{wenn}\ \mathbf{p} \models^{\mathcal{D}}_T \wedge\Gamma\ \text{dann}\ \mathbf{p} \models^{\mathcal{D}}_T \varphi^n(\mathrm{tt}) \vee \vee\Delta \quad \Rightarrow \\ \forall T \in \mathcal{T}. \forall \mathbf{p} \in \mathcal{P}_T\ .\ \text{wenn}\ \mathbf{p} \models^{\mathcal{D}}_T \wedge\Gamma\ \text{dann}\ \forall n \in \omega. \mathbf{p} \models^{\mathcal{D}}_T \varphi^n(\mathrm{tt}) \vee \vee\Delta.\end{array}$$

Man möchte nun schließen können:

$$\forall T \in \mathcal{T}. \forall \mathbf{p} \in \mathcal{P}_T\ .\ \text{wenn}\ \mathbf{p} \models^{\mathcal{D}}_T \wedge\Gamma\ \text{dann}\ \mathbf{p} \models^{\mathcal{D}}_T \nu X.\varphi \vee \vee\Delta.$$

Sei T also beliebig aus $\mathcal{T}$, $\mathbf{p}$ beliebig aus $\mathcal{P}_T$ und gelte:
wenn $\mathbf{p} \models^{\mathcal{D}}_T \wedge\Gamma$ dann $\forall n \in \omega. \mathbf{p} \models^{\mathcal{D}}_T \varphi^n(\mathrm{tt}) \vee \vee\Delta$.

Fall *a*): $\mathbf{p}$ ist ein endlicher Prozeß.
Damit ist φ stetig und nach Satz 2.26:
$\mathbf{p} \models^{\mathcal{D}}_T \nu X.\varphi \vee \vee\Delta$ genau dann wenn $\forall n \in \omega. \mathbf{p} \models^{\mathcal{D}}_T \varphi^n(\mathrm{tt}) \vee \vee\Delta$.
Also gilt:

$$\forall T \in \mathcal{T}. \forall \mathbf{p} \in \mathcal{P}_T\ .\ \text{wenn}\ \mathbf{p} \models^{\mathcal{D}}_T \wedge\Gamma\ \text{und}\ \mathbf{p}\ \text{endlich}\ \ \text{dann}\ \mathbf{p} \models^{\mathcal{D}}_T \nu X.\varphi \vee \vee\Delta.$$

Fall *b*): $\mathbf{p}$ ist ein unendlicher Prozeß.
Nun kann man, da φ nicht notwendigerweise stetig sein muß, nicht direkt das Gewünschte schließen. Nimmt man das Gegenteil an, nämlich

$$\begin{array}{l}\mathbf{p} \models^{\mathcal{D}}_T \wedge\Gamma\ \text{und}\ \mathbf{p} \not\models^{\mathcal{D}}_T \nu X.\varphi \vee \vee\Delta\ \ \text{so folgt} \\ \mathbf{p} \models^{\mathcal{D}}_T \wedge\Gamma\ \text{und}\ \mathbf{p} \models^{\mathcal{D}}_T \neg(\nu X.\varphi \vee \vee\Delta)\ \text{und weiter} \\ \mathbf{p} \models^{\mathcal{D}}_T \wedge\Gamma \wedge \neg(\nu X.\varphi \vee \vee\Delta).\end{array}$$

Das bedeutet, der unendliche Prozeß $\mathbf{p}$ erfüllt die Formel $\wedge\Gamma \wedge \neg(\nu X.\varphi \vee \vee\Delta)$. Wegen Satz 4.10 gibt es nun auch einen endlichen Prozeß $\overline{\mathbf{p}}$, der dieselbe Formel erfüllt. Das heißt, $\overline{\mathbf{p}} \models^{\mathcal{D}}_T \wedge\Gamma$ und $\overline{\mathbf{p}} \not\models^{\mathcal{D}}_T \nu X.\varphi \vee \vee\Delta$ was im Widerspruch zu Fall *a*) steht.

Zu 2):
Sei $\wedge\Gamma \wedge \mu X.\varphi \not\models^{\mathcal{D}} \vee\Delta$ und gelte als Widerspruchsannahme

$$\forall n \in \omega.\ \text{wenn}\ \ \wedge\Gamma \wedge \varphi^n(\mathrm{ff}) \models \vee\Delta\ \ \text{dann}\ \ \wedge\Gamma \wedge \varphi^{n+1}(\mathrm{ff}) \models \vee\Delta.$$

Da $\varphi^0(\mathrm{ff}) = \mathrm{ff}$, gilt auf jeden Fall $\wedge\Gamma \wedge \varphi^0(\mathrm{ff}) \models \vee\Delta$ und per Induktion $\forall n \in \omega.(\ \wedge\Gamma \wedge \varphi^n(\mathrm{ff}) \models \vee\Delta)$. Daraus ergibt sich im weiteren:

$$\forall n \in \omega.\forall T \in \mathcal{T}.\forall \mathbf{p} \in \mathcal{P}_T \text{ . wenn } \mathbf{p} \models^{\mathcal{D}}_T \wedge\Gamma \wedge \varphi^n(\mathrm{ff}) \text{ dann } \mathbf{p} \models^{\mathcal{D}}_T \vee\Delta \quad \Rightarrow$$
$$\forall T \in \mathcal{T}.\forall \mathbf{p} \in \mathcal{P}_T \text{ . wenn } (\exists n \in \omega.\mathbf{p} \models^{\mathcal{D}}_T \wedge\Gamma \wedge \varphi^n(\mathrm{ff})) \text{ dann } \mathbf{p} \models^{\mathcal{D}}_T \vee\Delta.$$

Man möchte wiederum schließen können:

$$\forall T \in \mathcal{T}.\forall \mathbf{p} \in \mathcal{P}_T \text{ . wenn } \mathbf{p} \models^{\mathcal{D}}_T \wedge\Gamma \wedge \mu X.\varphi \text{ dann } \mathbf{p} \models^{\mathcal{D}}_T \vee\Delta.$$

Sei wiederum T beliebig aus $\mathcal{T}$, $\mathbf{p}$ beliebig aus $\mathcal{P}_T$ und gelte:
wenn $(\exists n \in \omega.\mathbf{p} \models^{\mathcal{D}}_T \wedge\Gamma \wedge \varphi^n(\mathrm{ff}))$ dann $\mathbf{p} \models^{\mathcal{D}}_T \vee\Delta$.

Fall a): $\mathbf{p}$ ist ein endlicher Prozeß.
Dann ist φ stetig und man erhält sofort wenn $\mathbf{p} \models^{\mathcal{D}}_T \wedge\Gamma \wedge \mu X.\varphi$ dann $\mathbf{p} \models^{\mathcal{D}}_T \vee\Delta$.

Fall b): $\mathbf{p}$ ist ein unendlicher Prozeß.
In gleicher Weise wie in Fall 1.b bekommt man aus der Widerspruchsannahme, daß $\mathbf{p} \models^{\mathcal{D}}_T \wedge\Gamma \wedge \mu X.\varphi \wedge \neg\vee\Delta$. Wegen Satz 4.10 gibt es dann auch ein endliches $\overline{\mathbf{p}}$ mit $\overline{\mathbf{p}} \models^{\mathcal{D}}_T \wedge\Gamma \wedge \mu X.\varphi$ und $\overline{\mathbf{p}} \not\models^{\mathcal{D}}_T \vee\Delta$, was im Widerspruch zu Fall 2.a steht.

□

Damit ist man nun in der Lage, die Korrektheit (Theorem 4.4) zu zeigen.

Beweis zu 4.4

Gegeben sei $\Gamma_0 \vdash \Delta_0$, das heißt es gibt ein erfolgreiches Tableau τ mit der Wurzel $\Gamma_0 \vdash^{\epsilon} \Delta_0$. Um zu zeigen, daß dann auch $\Gamma_0 \models \Delta_0$ gilt, genügt es $\Gamma' \models^{\mathcal{D}'} \Delta'$ für alle Blätter $\Gamma' \vdash^{\mathcal{D}'} \Delta'$ des Baumes zu zeigen. Nach Satz 4.9, der Backward-Soundness, gilt dann die semantische Implikation auch für die Wurzel.

Nun nimmt man an, es gebe in τ ein semantisch falsches (aber erfolgreiches) Blatt. Dies kann nicht von der Form $\Gamma, \varphi \vdash^{\mathcal{D}} \varphi, \Delta$ sein, da für solche Blätter die sematische Implikation trivialerweise immer gilt. Folglich muß es von der Form gemäß 2(a) oder 2(b) aus Abbildung 4.3 sein. Eine Konstante U (bzw. V) eines erfolgreichen Blattes $\Gamma' \vdash^{\mathcal{D}} \Delta'$ dieser Form bezeichnen wir als *entscheidend*, wenn $U \in \Gamma'$ für einen kleinsten (bzw. $V \in \Delta'$ für einen größten) Fixpunkt steht und wenn zwischen dem zugehörigem ersten Auftreten der Sequenz diese Konstante einmal nach Regel ($Konst \vdash$) bzw. Regel ($\vdash Konst$) aus Abbildung 4.1 abgewickelt wurde. Die entscheidenden Konstanten eines solchen Blattes sind also diejenigen, welche den Erfolg des Blattes bewirkt haben könnten und davon kann es mehrere geben. Unter allen diesen Blättern wählt man nun eines so, daß eine seiner entscheidenden Konstanten U folgende Bedingung erfüllt: Oberhalb der Einführung von U kommt im Tableau kein weitere Konstante U$'$ vor, die ebenfalls entscheidend für ein falsches Blatt ist. Wir haben somit ein falsches Blatt und eine zugehörige entscheidende Konstante ausgewählt. Je nachdem, ob diese Konstante links oder rechts vom $\vdash^{\mathcal{D}'}$ steht, unterscheiden wir zwei Fälle:

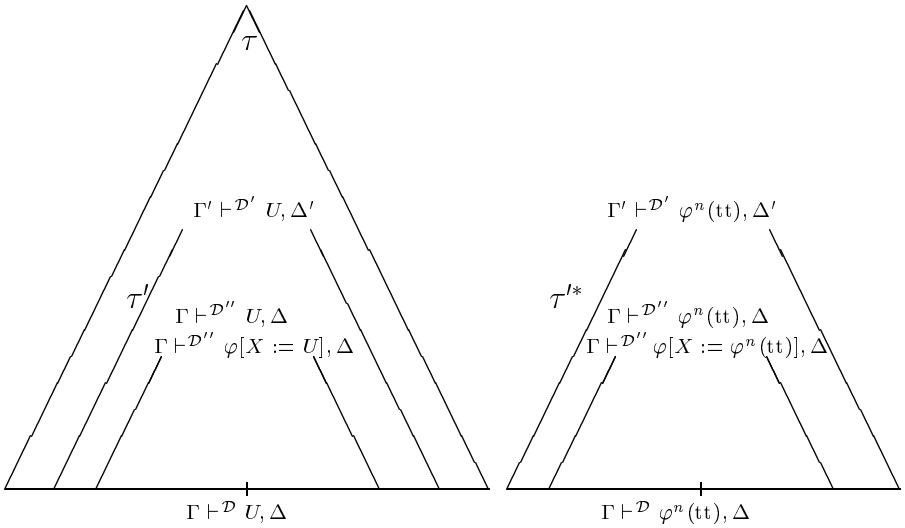


Abbildung 4.4: Skizze zum Korrektheitsbeweis

Fall 1:
Das ausgewählte Blatt ist von der Form $\Gamma \vdash^{\mathcal{D}} U, \Delta$ mit U als entscheidender Konstante , U steht also für einen größten Fixpunkt. Die Sequenz, bei der das U zum ersten mal auftaucht, sei $\Gamma' \vdash^{\mathcal{D}'} U, \Delta'$. Der Teilbaum mit dieser Sequenz als Wurzel sei mit τ' bezeichnet.

Dieser Teilbaum τ' kann nun mehrere falsche Blätter der Form $\overline{\Gamma} \vdash^{\overline{\mathcal{D}}} U, \overline{\Delta}$ enthalten. Für jedes dieser Blätter gilt also $\overline{\Gamma} \not\models^{\overline{\mathcal{D}}} U, \overline{\Delta}$. Damit gilt nach Lemma 4.11

$$\exists n \in \omega.(\overline{\Gamma} \models^{\overline{\mathcal{D}}} \varphi^n(\mathrm{tt}) \vee \vee\overline{\Delta} \text{ und } \overline{\Gamma} \not\models^{\overline{\mathcal{D}}} \varphi^{n+1}(\mathrm{tt}) \vee \vee\overline{\Delta}).$$

Unter allen genannten Blättern des Teilbaumes τ' mit U als entscheidender Konstante sei das Blatt $\Gamma \vdash^{\mathcal{D}} U, \Delta$ eines mit minimalem n. $\Gamma \vdash^{\mathcal{D}} U, \Delta$. τ' wird nun so transformiert, daß die Kontstante U an jeder Stelle durch die endliche Approximation $\varphi^n(\mathrm{tt})$ ersetzt wird. Dies ist in Abbildung 4.4 dargestellt.

In τ'^* gilt entsprechend der Wahl des Blattes $\Gamma \models^{\mathcal{D}} \varphi^n(\mathrm{tt}), \Delta$, das heißt durch die Transformation ist aus dem falschen Blatt $\Gamma \vdash^{\mathcal{D}} U, \Delta$ in τ' das wahre Blatt $\Gamma \vdash^{\mathcal{D}} \varphi^n(\mathrm{tt}), \Delta$ in τ'^* geworden. Ebenso ist natürlich auch der zugehörige Knoten mit der Sequenz $\Gamma \vdash^{\mathcal{D}''} U, \Delta$ in τ'^* nunmehr richtig. Der Nachfolgerknoten von $\Gamma \vdash^{\mathcal{D}''} \varphi^n(\mathrm{tt}), \Delta$ ist allerdings mit $\Gamma \vdash^{\mathcal{D}''} \varphi[X := \varphi^n(\mathrm{tt})], \Delta$ beschriftet und, da $\varphi[X := \varphi^n(\mathrm{tt})]$ für $\varphi^{n+1}(\mathrm{tt})$ steht, gemäß der Wahl von n falsch. Aufgrund der Backward-Soundness (Satz 4.9) muß τ'^* also falsche Blätter besitzen. Dies können nicht die Blätter mit U als entscheidender Konstante sein. Also muß eine andere Konstante für die fälschliche Entscheidung für den Erfolg der anderen Blätter verantwortlich sein. Dieses V muß, aufgrund der Wahl von U, im Baum unterhalb

von U eingeführt worden sein. Dies erlaubt es zusammen mit Fall 2, das gesamt Argument auf den Teilbaum τ'^* anzuwenden.

Fall 2:
Dieser Fall ist zum Fall 1 dual. Anstelle des größten Fixpunktes auf der rechten Seiten muß man nun einen kleinsten Fixpunkt auf der linken Seite des $\vdash^{\mathcal{D}}$ behandeln. Man wählt nun das kleinste m, für das gilt: $\wedge\Gamma \wedge \varphi^n(\mathrm{ff}) \models^{\mathcal{D}} \vee\Delta$ und $\wedge\Gamma \wedge \varphi^{n+1}(\mathrm{ff}) \not\models^{\mathcal{D}} \vee\Delta$. Ansonsten funktionieren die Überlegungen genauso und man kann wiederum folgern, daß es ein weiteres falsches Blatt mit einer anderen Konstante geben muß, die nach U eingeführt wurde.

In beiden Fällen gibt es also in τ'^* ein weiteres falsches Blatt mit einer Konstanten, die erst in τ'^* eingeführt wurde. Blätter, die in τ'^* falsch sind, sind insbesondere in τ falsch, und damit kann man das Argument wiederholt auf τ'^* anwenden, obwohl dies kein gültiger Ableitungsbaum ist. Das bedeutet jedoch, daß das Tableau unendlich viele verschiedene Konstanten enthält, was im Widerspruch zu Lemma 4.7 steht. □

4.5 Ein effizienteres System

Wir haben nun also das Beweissystem vorgestellt und dessen Korrektheit bewiesen. Die Fixpunktinduktion stellt dabei den wesentlichen Bestandteil des Beweisverfahrens dar. Die zugehörigen Abbrüche 3(a) und 3(b) aus Abbildung 4.2 wurden dadurch gerechtfertigt, daß eine weitere Ableitung überflüssig ist, wenn man zum zweiten mal auf eine Sequenz der Form $\Gamma, U \vdash^{\mathcal{D}} \Delta$ beziehungsweise $\Gamma \vdash^{\mathcal{D}} U, \Delta$ trifft. Allerdings kann man argumentieren, daß man nicht mehr weiter ableiten braucht, wenn man zum zweiten mal auf eine Sequenz von *beliebiger* Form trifft. Das so modifizierte System hat den Vorteil, daß die Ableitungsbäume wesentlich kleiner werden. Allerdings läßt sich der Korrektheitsbeweis für dieses System nicht so direkt führen, wie dies für das System aus Abschnitt 4.2 möglich war. Insbesondere ist dann das Lemma 4.11 nicht anwendbar. Dieser Abschnitt ist nun dem modifizierten System gewidmet und wir zeigen, daß beide Systeme äquivalent sind.

Die Regeln in diesem neuen System sind unverändert die Regeln aus Abbildung 4.1. Was sich ändert, sind die Abbruchkriterien. Auch die Definition des Erfolgs eines Tableaus muß entsprechend angepaßt werden. Abbildung 4.5 faßt die vier Fälle, die zum Abbruch der Regelanwendungen führen, zusammen.

Auch die Kriterien für den Erfolg eines Blattes haben sich geändert (siehe Abbildung 4.6). Ein maximales Tableau ist in dem neuen Sytem erfolgreich, wenn alle seine Blätter die neuen Erfolgsbedingungen erfüllen Man beachte insbesondere in den Fällen 2(a) beziehungsweise 2(b) aus Abbildung 4.6 die Bedingung, daß $(U = \mu X.\varphi)$ in $\mathcal{D}'$ beziehungsweise $(U = \nu X.\varphi)$ in $\mathcal{D}'$ enthalten ist. Das bedeutet, daß die Konstante U bereits vor $\Gamma \vdash^{\mathcal{D}'} \Delta$ eingeführt worden sein muß.

1. $\Gamma, \varphi \vdash^{\mathcal{D}} \Delta, \varphi$
2. $\emptyset \vdash^{\mathcal{D}} \emptyset$
3. Wiederholung einer Sequenz:
   $\Gamma \vdash^{\mathcal{D}} \Delta$ und oberhalb im Baum ist bereits ein Knoten mit $\Gamma \vdash^{\mathcal{D}'} \Delta$ beschriftet.
4. Abbruch vor der unnützen Einführung einer neuen Konstante
   (a) Man trifft auf einen Knoten der Form $\Gamma, \sigma X.\varphi \vdash^{\mathcal{D}} \Delta$ und oberhalb im Tableau existiert ein Knoten der Form $\Gamma', \sigma X.\varphi \vdash^{\mathcal{D}'} \Delta'$ wobei $\sigma X.\varphi \in \{\nu X.\varphi, \mu X.\varphi\}$, $\mathcal{D}(\Gamma) = \mathcal{D}'(\Gamma')$ und $\mathcal{D}(\Delta) = \mathcal{D}'(\Delta')$.
   (b) Man trifft auf einen Knoten der Form $\Gamma \vdash^{\mathcal{D}} \sigma X.\varphi, \Delta$ und oberhalb im Tableau existiert ein Knoten der Form $\Gamma' \vdash^{\mathcal{D}'} \sigma X.\varphi, \Delta'$ wobei $\sigma X.\varphi \in \{\nu X.\varphi, \mu X.\varphi\}$, $\mathcal{D}(\Gamma) = \mathcal{D}'(\Gamma')$ und $\mathcal{D}(\Delta) = \mathcal{D}'(\Delta')$.

Abbildung 4.5: Abbruchbedingungen des effizienteren Beweissystems

Damit hat man ein weiteres Beweissystem mit dem Vorteil, daß die Ableitungen durch die Verschärfung der Abbruchbedingungen zum Teil deutlich kürzer, in keinem Fall jedoch länger werden. Das alte System bezeichnen wir im folgenden der Kürze halber mit S_1, das neue mit S_2. $\vdash_1$ und $\vdash_2$ stehen für den beweistheoretischen Folgerungsbegriff in S_1 beziehungsweise S_2.

Um nachzuweisen, daß die beiden Systeme äquivalent sind, machen wir zunächst folgendes Beobachtung.

Beobachtung 4.12 *Gegeben seien auf einem Pfad eines Tableaus (in S_1 oder S_2) die Sequenzen $\Gamma_1 \vdash^{\mathcal{D}_1} \Delta_1$, $\Gamma_2 \vdash^{\mathcal{D}_2} \Delta_2$ und $\Gamma_3 \vdash^{\mathcal{D}_3} \Delta_3$ in dieser Reihenfolge. Außerdem sei U eine Konstante , die syntaktisch in $\Gamma_2 \cup \Delta_2$ und $\Gamma_3 \cup \Delta_3$ vorkommt und die in $\mathcal{D}_1$ noch nicht deklariert ist. Dann enthalten alle Sequenzen zwischen $\Gamma_2 \vdash^{\mathcal{D}_2} \Delta_2$ und $\Gamma_3 \vdash^{\mathcal{D}_3} \Delta_3$ die Konstante U oder eine weitere Konstante V, die ebenfalls in $\mathcal{D}_1$ noch nicht deklariert ist.*

Beweis zu 4.12

Da U in $\Gamma_2 \vdash^{\mathcal{D}_2} \Delta_2$ und $\Gamma_3 \vdash^{\mathcal{D}_3} \Delta_3$ enthalten ist, muß in allen Sequenzen dazwischen U selbst oder eine Konstante, sagen wir V, auftreten, aus der das U, möglicherweise indirekt über weitere Konstanten, durch Expansion wieder generiert werden kann. Wäre dies nicht der Fall, so könnte $\Gamma_3 \vdash^{\mathcal{D}_3} \Delta_3$ die Konstante U nicht enthalten, da bei jeder Konstanteneinführung neue Konstantennamen benutzt werden. Diese

Ein Blatt eines maximalen Tableaus heißt *erfolgreich*, wenn einer der folgenden Fälle auftritt.

1. Das Blatt ist von der Form: $\Gamma, \varphi \vdash^{\mathcal{D}} \varphi, \Delta$
2. Erfolgreiche Fixpunktinduktion : das Blatt ist von der Form $\Gamma \vdash^{\mathcal{D}} \Delta$, wobei:
   (a) Erfolg durch einen kleinsten Fixpunkt links.
   - Es gibt eine Sequenz $\Gamma \vdash^{\mathcal{D}'} \Delta$ und zwischen dieser und dem Blatt eine Sequenz Γ' , $U \vdash^{\mathcal{D}''} \Delta'$.
   - $(U = \mu X.\varphi) \in \mathcal{D}'$
   - Zwischen der Sequenz $\Gamma \vdash^{\mathcal{D}'} \Delta$ und dem Blatt $\Gamma \vdash^{\mathcal{D}} \Delta$ wurde auf U Regel $(Konst \vdash)$ aus Abbildung 4.1 mindestens einmal angewendet.

   (b) Erfolg durch einen größten Fixpunkt rechts.
   - Es gibt eine Sequenz $\Gamma \vdash^{\mathcal{D}'} \Delta$ und zwischen dieser und dem Blatt eine Sequenz $\Gamma' \vdash^{\mathcal{D}''} U$, Δ'.
   - $(U = \nu X.\varphi) \in \mathcal{D}'$
   - Zwischen der Sequenz $\Gamma \vdash^{\mathcal{D}'} \Delta$ und dem Blatt $\Gamma \vdash^{\mathcal{D}} \Delta$ wurde auf U Regel $(\vdash Konst)$ aus Abbildung 4.1 mindestens einmal angewendet.

Abbildung 4.6: neue Erfolgsbedingungen für die Blätter

Konstante V muß, da das U direkt oder indirekt aus ihr gewonnen werden kann, nach U eingeführt worden sein. Damit kann V, genauso wie U, nicht bereits in $\mathcal{D}_1$ definiert worden sein. □

Nun können wir die Äquivalenz der beiden Systeme formulieren.

Satz 4.13 (Äquivalenz von S_1 und S_2) $\Gamma \vdash_1 \Delta$ *gdw.* $\Gamma \vdash_2 \Delta$

Beweis zu 4.13

Es sind zwei Richtungen zu zeigen: für jede erfolgreiche Ableitung in S_1 gibt es eine erfolgreiche Ableitung in S_2 und umgekehrt. Im ersten Fall muß man zeigen, daß jede erfolgreiche Ableitung in S_1 durch entsprechende Verkürzung in eine erfolgreiche Ableitung in S_2 transformiert werden kann. Im zweiten Fall ist nachzuweisen, daß erfolgreiche Blätter eines Ableitungsbaums in S_2, die nicht zugleich erfolgreich in S_1 sind, in jedem Fall im System S_1 erfolgreich fortgesetzt werden können.

$\Rightarrow$: Sei ein erfolgreiches Tableau τ in S_1 gegeben. Zunächst transformiere man τ so, daß immer, wenn eine Konstante für einen kleinsten Fixpunkt auf der linken Seite beziehungsweise für einen größten Fixpunkt auf der rechten Seite nach Regel 11 beziehungsweise Regel 12 aus 4.1 expandiert werden kann, dies auch sofort geschieht. Ist die Expansion bei mehreren passenden Konstanten gleichzeitig möglich, dann natürlich hintereinander. Das so erhaltene Tableau sei mit τ' bezeichnet. Da sich die Expansion einer Konstanten in einer Sequenz, sagen wir $\Gamma \vdash^{\mathcal{D}} U\ ,\ \Delta$ auf die Formelmengen Γ und Δ nicht auswirkt, sondern nur U selbst betrifft, bleibt die Verzweigungsstruktur von τ' gegenüber τ unverändert. Das bedeutet: die Pfade von τ' ensprechen genau denen von τ, nur daß in den Pfaden unter Umständen manche Konstanten eher expandiert werden als in denen von τ.

In τ' betrachte man einen vollständigen Pfad von der Wurzel bis zu einem Blatt. Dieser Pfad ist nun nicht notwendigerweise auch ein Pfad (erfolgreich oder erfolglos) eines Ableitungsbaumes in S_2, da er unter Umständen nach den schärferen Abbruchregeln oder durch die Umstrukturierung beim Vorziehen von Konstantenexpansionen schon früher hätte enden müssen. Kürzt man den Pfad, so daß er den Abbruchregeln in S_2 genügt, so muß der dadurch enstandene verkürzte Pfad nicht unbedingt erfolgreich sein. Das Verkürzen des Pfades kann nur durch Regel 3 aus Abbildung 4.5 hervorgerufen worden sein, also durch das Auftreten von zwei Sequenzen $\Gamma_2 \vdash^{\mathcal{D}_2'} \Delta_2$ und $\Gamma_2 \vdash^{\mathcal{D}_2} \Delta_2$, da in allen anderen Abbrüchen die beiden Systeme übereinstimmen. Folgende Fälle müssen unterschieden werden:

Fall 1: Der ungekürzte Pfad endet mit dem Blatt $\Gamma\ ,\ \varphi \vdash^{\mathcal{D}} \varphi\ ,\ \Delta$ und die Sequenzen $\Gamma_2 \vdash^{\mathcal{D}_2'} \Delta_2$ und $\Gamma_2 \vdash^{\mathcal{D}_2} \Delta_2$ treten damit beide oberhalb des Blattes auf. Ist der Pfad bis $\Gamma_2 \vdash^{\mathcal{D}_2} \Delta_2$ nicht ohnehin erfolgreich in S_2, so ist der Ableitungsbaum um das Stück zwischen $\Gamma_2 \vdash^{\mathcal{D}_2'} \Delta_2$ und $\Gamma_2 \vdash^{\mathcal{D}_2} \Delta_2$ zu kürzen.

Fall 2: Der ungekürzte Pfad endet in s und oberhalb befindet sich eine Sequenz s', wobei s und s' abkürzend für $\Gamma\ ,\ U \vdash^{\mathcal{D}} \Delta$ und $\Gamma\ ,\ U \vdash^{\mathcal{D}'} \Delta$ beziehungsweise $\Gamma \vdash^{\mathcal{D}} U\ ,\ \Delta$ und $\Gamma \vdash^{\mathcal{D}'} U\ ,\ \Delta$ stehen. Ist der verkürzte Pfad nicht auch gleich in S_2 erfolgreich, so kann das folgende Gründe haben:

Fall 2.1: $\Gamma_2 \vdash^{\mathcal{D}_2'} \Delta_2$ und $\Gamma_2 \vdash^{\mathcal{D}_2} \Delta_2$ treten beide oberhalb von s' auf. Wie im Fall 1 läßt sich der Baum um das Stück dazwischen kürzen.

Fall 2.2: $\Gamma_2 \vdash^{\mathcal{D}_2'} \Delta_2$ und $\Gamma_2 \vdash^{\mathcal{D}_2} \Delta_2$ treten beide zwischen s' und s auf. Wiederum kürzt man den Baum um das entsprechende Stück.

Fall 2.3: $\Gamma_2 \vdash^{\mathcal{D}_2'} \Delta_2$ befindet sich vor s' und $\Gamma_2 \vdash^{\mathcal{D}_2} \Delta_2$ zwischen s' und s. Damit tritt zwischen $\Gamma_2 \vdash^{\mathcal{D}_2'} \Delta_2$ und $\Gamma_2 \vdash^{\mathcal{D}_2} \Delta_2$ eine Sequenz, nämlich s', mit einer Konstanten für einen kleinsten Fixpunkt links oder für eine größten Fixpunkt rechts auf. Nach Konstruktion von τ' wurde eine der passenden Konstanten, sagen wir U, auch tatsächlich abgewickelt. Als letzte Bedingung für den Erfolg in S_2 in Abbildung 4.6 gilt auch $(U = \mu X.\varphi) \in \mathcal{D}_2'$ oder $(U = \nu X.\varphi) \in \mathcal{D}_2'$, da sonst nach Lemma 4.12 entweder U oder eine neue Konstante, die es bei $\Gamma_2 \vdash^{\mathcal{D}_2'} \Delta_2$ noch nicht gab,

in $\Gamma_2 \vdash^{\mathcal{D}_2} \Delta_2$ syntaktisch vorkommen; beides ist ein Widerspruch. Damit ist der verkürzte Pfad auch in S_2 erfolgreich.

Diese Fallunterscheidung zeigt, daß man in den Fällen, bei denen τ' nicht zugleich auch ein erfolgreiches Tableau in S_2 ist, durch wiederholtes Verkürzen des Baumes einen erfolgreichen Ableitungsbaum in S_2 erhalten kann.

$\Leftarrow$: Gegeben sei ein erfolgreiches Tableau τ in S_2. Wir zeigen, wie man diesen Ableitungsbaum gegebenenfalls so verlängern kann, daß man eine erfolgreiche Ableitung auch in S_1 erhält. Der in S_2 erfolgreiche und damit maximale Ableitungsbaum τ ist auch im System S_1 ein Ableitungsbaum, wenn auch nicht unbedingt maximal. Der Grund hierfür ist, daß er Blätter der Form $\Gamma \vdash^{\mathcal{D}} \Delta$ enthalten kann, die keine erfolgreichen Blätter in S_1 darstellen. Solche Blätter, die in S_2 aber nicht in S_1 erfolgreich sind, bezeichnen wir als S_2-Blätter. Blätter anderer Form sind für den Beweis uninteressant, da Abbruch und Erfolg für sie in beiden Systemen übereinstimmen. τ habe nun eine endliche Anzahl n von S_2-Blättern $\Gamma_1 \vdash^{\mathcal{D}_1} \Delta_1$, ... , $\Gamma_n \vdash^{\mathcal{D}_n} \Delta_n$. Für jede dieser Sequenzen $\Gamma_i \vdash^{\mathcal{D}_i} \Delta_i$ gibt es den zugehörigen Vorgänger $\Gamma_i \vdash^{\mathcal{D}'_i} \Delta_i$ und zwischen beiden liegt wegen Bedingung 2(a) (beziehungsweise 2(b)) je ein Knoten der Form Γ'_i , $U \vdash^{\mathcal{D}''_i} \Delta'_i$ (beziehungsweise $\Gamma'_i \vdash^{\mathcal{D}''_i} V$, Δ'_i). Mit τ_i sei derjenige Teilbaum von τ bezeichnet, der $\Gamma_i \vdash^{\mathcal{D}'_i} \Delta_i$ als Wurzel hat und bei dem nach Γ'_i , $U \vdash^{\mathcal{D}''_i} \Delta'_i$ (beziehungsweise $\Gamma'_i \vdash^{\mathcal{D}''_i} V$, Δ'_i) abgebrochen wird. Da $\Gamma_i \vdash^{\mathcal{D}_i} \Delta_i$ als Blatt in τ_i nicht mehr auftaucht, hat τ_i höchstens $n-1$ S_2-Blätter. Setzt man nun an allen Blättern $\Gamma_i \vdash^{\mathcal{D}_i} \Delta_i$ aus τ mit den jeweiligen τ_i fort, so bekommt man einen Ableitungsbaum mit höchstens $n*(n-1)$ Blättern der Form (*). Man beachte dabei, daß in diesem Baum die neuen Blätter Γ_i , $U \vdash^{\mathcal{D}''} \Delta_i$ beziehungsweise $\Gamma_i \vdash^{\mathcal{D}''} V$, Δ_i nach den Kriterien aus S_1 erfolgreich sind. Dieses Verfahren setzt man so lange fort, bis alle unerwünschten Blätter verschwunden sind und man erhält so einen Ableitungsbaum im System S_1. □

4.6 Beispiele

Die Arbeitsweise des Beweissystemes soll nun noch zum Abschluß anhand eines Beispieles gezeigt werden. Zu beweisen sei die folgende Aussage: *"Wenn es einen unendlichen Pfad gibt, bei dem irgendwann φ immer gilt, so existiert ein Pfad, bei dem die Eigenschaft φ unendlich oft gilt".*

Erstere Eigenschaft sei mit φ_1, letztere mit φ_2 abgekürzt. Zunächst müssen die beiden Eigenschaften in μHML übersetzt werden. Im Falle von φ_2 soll dies etwas ausführlicher geschehen. Dabei ist es vorteilhaft, das Vorgehen hier mit der Herleitung der μHML-Formel aus Abschnitt 3.3 zu vergleichen. Obwohl die Eigenschaft dort ("Auf *allen* Pfaden gilt unendlich oft") sehr ähnlich klingt, macht die Formel φ_2 jetzt deutlich mehr Schwierigkeiten. Wir werden sehen, warum. Wie formuliert man also "Es gibt einen Pfad, bei dem die Eigenschaft φ unendlich oft gilt"? Intui-

tiv kann die Eigenschaft "unendlich oft φ" wieder durch "immer ist es der Fall, daß irgendwann φ gilt" ausgedrückt werden.

Also versuchen wir zunächst zu formulieren, daß es einen Pfad gibt, bei dem eine Eigenschaft ψ immer gilt. Das bedeutet dasselbe, wie "ψ gilt jetzt und es gibt einen direkten Nachfolger, von dem ausgehend es einen Pfad gibt, bei dem ψ immer gilt". Das legt folgende rekursive μHML-Formel nahe:

$$\nu X.\psi \wedge \langle - \rangle X$$

Leider drückt diese Formel eine etwas stärkere Eigenschaft aus, nämlich "Es gibt einen unendlichen Pfad, bei dem ψ immer gilt". Präziser in Worten ausgedrückt lautet die gewünschte Eigenschaft nämlich: "Jetzt gilt ψ und wenn es einen Nachfolger gibt, so ist einer darunter, von dem ausgehend es einen Pfad gibt, bei dem ψ immer gilt". In Formeln:

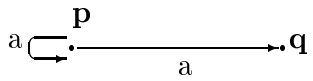
$$\nu X.\psi \wedge ([-]\mathrm{ff} \vee \langle - \rangle X)$$

Dies entspricht dem Branching-Time-Operator $\exists \mathrm{G}\psi$, der bereits in Kapitel 2 erwähnt wurde.

Einfacher zu formulieren ist die Eigenschaft, daß es einen Pfad gibt, bei dem irgendwann einmal φ gilt, nämlich durch $\mu X.\varphi \vee \langle - \rangle X$. Setzt man nun beide Formeln zusammen, so bekommt man für φ_2

$$\nu X.(\mu Y.\varphi \vee \langle - \rangle Y) \wedge ([-]\mathrm{ff} \vee \langle - \rangle X)$$

Drückt dies nun die gewünschte Eigenschaft φ_2 aus? Leider nein. Man hat im Grunde in falscher Form über die Pfade quantifiziert, genauer gesagt, hat man zwar formuliert, daß es einen Pfad gibt, bei dem immer gilt, daß es eine Fortsetzung gibt, auf der irgendwann φ gilt, jedoch nicht, daß es immer noch *derselbe* Pfad ist. Zum Beispiel erfüllt der Prozeß **p** in folgendem Transitionssystem diese Eigenschaft (wobei **q** die Eigenschaft φ erfüllt, **p** hingegen nicht).



Die Existenzquantifizierung in diesem Beispiel wird duch den Modaloperator $\langle - \rangle$ erreicht; um allerdings auszudrücken, daß das "immer" und das "irgendwann" sich auf die Existenz desselben Pfades beziehen, darf man beide Fixpunkte, den größten für das "immer" und den kleinsten für das "irgendwann" nicht voneinander trennen, sondern beide müssen *geschachtelt* auftreten.

Das Problem soll nun etwas anders formuliert werden. "Es gibt einen Pfad, bei dem unendlich oft φ gilt" heißt, "Es gibt einen Pfad, bei dem irgendwann φ gilt und von dieser Stelle aus gibt es wiederum einen Pfad, bei dem irgendwann φ gilt" und dies unendlich oft.

"Es gibt einen Pfad, bei dem irgendwann φ und X gilt", können wir bereits hinschreiben:

$$\theta(X) = \mu Y.(\varphi \wedge X) \vee \langle - \rangle Y$$

Das durch θ geforderte Verhalten soll nun unendlich oft hintereinander möglich sein. Dies drückt man durch einen größten Fixpunkt aus:

$$\nu X.\theta(X) = \nu X.\mu Y.(\varphi \wedge X) \vee \langle - \rangle Y$$

Dies ist allerdings immer noch nicht ganz richtig. Die Bedeutung dieser Formel ist zwar in der Tat, daß irgendwann φ gilt und von dort wieder irgendwann und unendlich oft so weiter. Der Fehler liegt diesmal bei dem "Irgendwann" denn es bedeutet "*Jetzt* oder irgendwann später". Damit besitzt zum Beispiel ein Prozeß, der aus nur einem Zustand besteht, für den die Eigenschaft φ erfüllt ist, und der keine Übergänge hat, diese Eigenschaft $\nu X.\theta(X)$. Wenn man nun noch das "irgendwann" durch "irgenwann außer jetzt" ersetzt, hat man nun endlich und tatsächlich die Eigenschaft φ_2:

$$\varphi_2 = \nu X.\langle - \rangle \theta(X) = \nu X.\langle - \rangle(\mu Y.(\varphi \wedge X) \vee \langle - \rangle Y)$$

Dieses ausführliche Beispiel zeigt zum einen, daß aufgrund der Ausdrucksstärke der Logik die Formulierung von Eigenschaften durchaus subtil sein kann. Insbesondere muß man, da es sich bei μHML um eine Logik der verzweigten Zeit handelt, sorgfältig beachten, über welche Pfade man jeweils quantifizieren will. Zum anderen scheint es, daß es interessante Eigenschaften gibt, bei denen man auf eine echte Verschränkung von Fixpunkten nicht verzichten kann.

Zurück zum Beispiel. Die noch fehlende Eigenschaft φ_1 ist deutlich einfacher zu formulieren, was in diesem Fall an der fehlenden Verschränkung der Fixpunke liegt.

$$\varphi_1 = \mu X.(\nu Y.\varphi \wedge \langle - \rangle Y) \vee \langle - \rangle X$$

Nun soll, wie bereits zu Beginn des Abschnittes angekündigt, gezeigt werden: φ_1 impliziert φ_2.

Man beginnt also ein Tableau zu generieren, dessen Wurzel mit der zu zeigenden Implikation beschriftet ist. Die ersten vier Schritte behandeln die äußeren Fixpunkte, für die die Konstanten R und T eingeführt werden. Deren Deklaration wird in $\mathcal{D}_1$ und $\mathcal{D}_2$ festgehalten, das heißt $\mathcal{D}_1 = ((R = \mu X.(\nu Y.\varphi \wedge \langle - \rangle Y) \vee \langle - \rangle X))$ und $\mathcal{D}_2 = \mathcal{D}_1 \cdot (T = \nu X.\langle - \rangle(\mu Y.(\varphi \wedge X) \vee \langle - \rangle Y)$ Danach werden die Konstanten gemäß

ihrer Deklaration wieder expandiert. Die ersten Ableitungschritte lauten somit:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mu X.(\nu Y.\varphi \wedge \langle - \rangle Y) \vee \langle - \rangle X \vdash^{\epsilon} \nu X.\langle - \rangle(\mu Y.(\varphi \wedge X) \vee \langle - \rangle Y)}{R \vdash^{\mathcal{D}_1} \nu X.\langle - \rangle(\mu Y.(\varphi \wedge X) \vee \langle - \rangle Y)}}{R \vdash^{\mathcal{D}_2} T}}{R \vdash^{\mathcal{D}_2} \langle - \rangle(\mu Y.(\varphi \wedge T) \vee \langle - \rangle Y)}}{(\nu Y.\varphi \wedge \langle - \rangle Y) \vee \langle - \rangle R \vdash^{\mathcal{D}_2} \langle - \rangle(\mu Y.(\varphi \wedge T) \vee \langle - \rangle Y)}$$

Das $\vee$ auf der linken Seite bewirkt nun eine Aufspaltung in zwei Unterziele, von denen nur das erste, nämlich $\nu Y.\varphi \wedge \langle - \rangle Y \vdash^{\mathcal{D}_2} \langle - \rangle(\mu Y.(\varphi \wedge T) \vee \langle - \rangle Y)$ weiterverfolgt werden soll. Durch passende Anwendung der Regeln für die Behandlung der Fixpunkte sowie für das $\wedge$ auf der linken Seite und abschließender Abschwächung der linken Seite bekommt man:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(\nu Y.\varphi \wedge \langle - \rangle Y) \vdash^{\mathcal{D}_2} \langle - \rangle(\mu Y.(\varphi \wedge T) \vee \langle - \rangle Y)}{S_1 \vdash^{\mathcal{D}_3} \langle - \rangle(\mu Y.(\varphi \wedge T) \vee \langle - \rangle Y)}}{\varphi \wedge \langle - \rangle S_1 \vdash^{\mathcal{D}_3} \langle - \rangle(\mu Y.(\varphi \wedge T) \vee \langle - \rangle Y)}}{\varphi \ , \ \langle - \rangle S_1 \vdash^{\mathcal{D}_3} \langle - \rangle(\mu Y.(\varphi \wedge T) \vee \langle - \rangle Y)}}{\langle - \rangle S_1 \vdash^{\mathcal{D}_3} \langle - \rangle(\mu Y.(\varphi \wedge T) \vee \langle - \rangle Y)}$$

In $\mathcal{D}_3$ wurde die Konstante S_1 für die Eigenschaft "Es gibt einen unendlichen Pfad, bei dem immer φ gilt" eingeführt. Nun steht man zum ersten Mal im Verlaufe des Beweises vor einem wirklichen Schritt in dem Sinne, daß nicht nur aus Buchhaltungsgründen Konstanten eingeführt, Fixpunkte abgewickelt werden oder das Ziel entsprechend logischer Konnektive in Unterziele aufgespalten wird. Damit setzt sich die Ableitung wie folgt fort:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\langle - \rangle S_1 \vdash^{\mathcal{D}_3} \langle - \rangle(\mu Y.(\varphi \wedge T) \vee \langle - \rangle Y)}{S_1 \vdash^{\mathcal{D}_3} \mu Y.(\varphi \wedge T) \vee \langle - \rangle Y}}{S_1 \vdash^{\mathcal{D}_4} U_1}}{S_1 \vdash^{\mathcal{D}_4} (\varphi \wedge T) \vee \langle - \rangle U_1}}{S_1 \vdash^{\mathcal{D}_4} (\varphi \wedge T) \ , \ \langle - \rangle U_1}$$

Bis jetzt funktionierte die Ableitung relativ geradlinig. Die einzige Wahlmöglichkeit bestand darin, die Konstanteneinführungen und -abwicklungen in anderer Reihenfolge vorzunehmen. Von Zeit zu Zeit schadet es jedoch nichts, sich auch Gedanken darüber zu machen, was man eigentlich zu beweisen versucht. Beim augenblicklichen Stand der Dinge versucht man gerade aus "Es gibt einen unendlichen Pfad, bei dem immer φ gilt" , nämlich aus S_1, zu folgern, daß mindestens eine der beiden folgenden Eigenschaften zutrifft: "Im Augenblick gilt φ und es gibt einen Pfad, bei dem φ unendlich oft gilt", nämlich $\varphi \wedge T$, oder "Es gibt einen Nachfolger, von dem aus es einen Pfad gibt, bei dem irgendwann φ gilt und von dort aus unendlich oft φ". Beides sind natürlich nur Umformulierungen von φ_2 und S_1 impliziert jede davon. Erstere scheint jedoch naheriegender zu sein, da S_1 auf der linken Seite bereits

impliziert, daß T schon jetzt gilt und nicht erst irgendwann. Also werfen wir $\langle - \rangle U_1$ mit Hilfe der Abschwächregel ($weak_2$) fort. Würde man stattdessen $\varphi \wedge T$ loswerden, funktionierte der Beweis ebenfalls, es würde nur länger dauern.

$$\dfrac{\dfrac{S_1 \vdash^{\mathcal{D}_4} (\varphi \wedge T)\ ,\ \langle - \rangle U_1}{S_1 \vdash^{\mathcal{D}_4} \varphi \wedge T}}{\dfrac{\varphi \wedge \langle - \rangle S_1 \vdash^{\mathcal{D}_4} \varphi \wedge T}{\varphi\ ,\ \langle - \rangle S_1 \vdash^{\mathcal{D}_4} \varphi \wedge T}}$$

Wiederum spaltet sich der Beweis auf: Der eine Zweig $\varphi\ ,\ \langle - \rangle S_1 \vdash^{\mathcal{D}_4} \varphi$ endet sofort, und zwar, gemäß den Kriterien aus Abbildung 4.3 beziehungsweise Abbildung 4.6 erfolgreich. Der zweite Zweig benötigt noch einen Ableitungsschritt, die Expansion von T, bis eine Sequenz $\varphi\ ,\ \langle - \rangle S_1 \vdash^{\mathcal{D}_4} \langle - \rangle(\mu Y.(\varphi \wedge T) \vee \langle - \rangle Y)$ auftaucht, die es bereits einmal weiter oben gab. An dieser Stelle wird nach den Kriterien des effizienteren Systems nach Abbildung 4.5 abgebrochen.

$$\varphi\ ,\ \langle - \rangle S_1 \dfrac{\varphi\ ,\ \langle - \rangle S_1 \vdash^{\mathcal{D}_4} T}{\vdash^{\mathcal{D}_4} \langle - \rangle(\mu Y.(\varphi \wedge T) \vee \langle - \rangle Y)}$$

In diesem Blatt ist auf der rechte Seite die Konstante T enthalten, die für einen größten Fixpunkt steht. Darüberhinaus wurde diese Konstante zwischen den erstmaligen Aufteten der Sequenz und diesem Blatt einmal expandiert, nämlich unmittelbar vor dem Blatt. Damit ist das Blatt nach Definition 4.6 erfolgreich. Wollte man auch im alten System ein erfolgreiches Tableau erreichen, müßte man die Ableitung noch um einige Schritte fortsetzen, bis T dann zum zweiten mal auf der rechten Seite blank auftaucht.

In Abbildung 4.7 ist der gesamte Beweis mit den noch fehlenden Zweigen dargestellt. Dabei verzichten wir auf die explixite Angabe der Deklarationslisten $\mathcal{D}_i$. An den jeweiligen Stellen ergeben sich diese durch die Hinzufügung der Deklaration der entsprechenden neuen Konstanten.

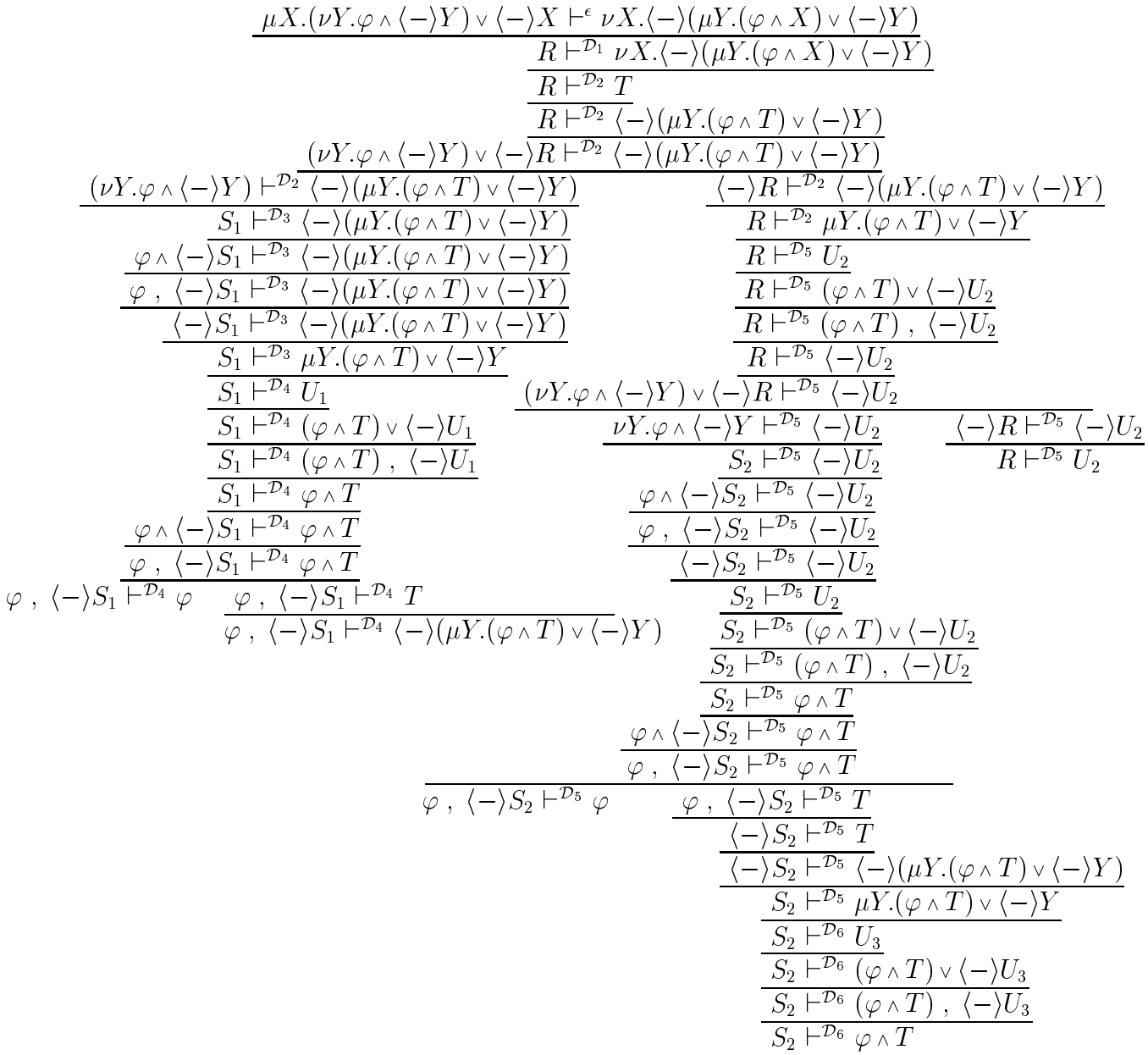


Abbildung 4.7: Beispielableitung

Kapitel 5

Ausblick

In den vorangehenden Kapiteln haben wir ein korrektes Beweissystem für Hennessy-Milner-Logik mit Rekursion vorgestellt, mit dem man Verfeinerungsschritte beim Programmentwurf in dieser Logik auf ihre Korrektheit überprüfen kann.

Für die Zukunft haben wir Erweiterungspläne für diese Arbeit, die in zwei Richtungen gehen, einerseits der Ausbau der theoretischen Grundlagen und zum anderen Erweiterungen, die die Benutzbarkeit des Systems verbessern.

Was die theoretischen Grundlagen betrifft, steht der Vollständigkeitsbeweis des Systems im Vordergrund, an dem wir zur Zeit bereits arbeiten.

Ein anderer wichtiger Erweiterungspunkt, um insbesondere die notwendigen Beweise bei der Verifikation eines Programms kurz und übersichtlich zu halten, betrifft das Vorgehen bei der Konstruktion und somit auch bei der Verifikation von Verfeinerungsschritten. Um den Programmentwurf überschaubar zu machen, muß man nicht nur schrittweise, sonder auch *modular* vorgehen. Dies bedeutet, daß die Programmentwicklung nicht so linear vorgenommen wird wie von uns vorgestellt, sondern daß es die Möglichkeit gibt, eine Spezifikation in mehrere unabhängige Teilspezifikationen aufzuspalten. Zum Beispiel könnte man sich im Laufe der Programmentwicklung entscheiden, das gewünschte Verhalten durch zwei parallelgeschaltete Prozesse zu implementieren. Dafür würde man die Spezifikation in zwei Teilspezifikationen aufspalten und durch einen Paralleloperator miteinander verknüpfen. In dieser Vorgehensweise benutzt man also die Parallelschaltung als Modularisierungsoperator für Spezifikationen. Andere aus der Prozeßtheorie bekannte Operatoren kann man in gleicher Weise verwenden. Um ein solches Vorgehen bei der Verfeinerung der Verifikation zugänglich zu machen, muß man nun auch die Modularisierungs-Operatoren mit in das Beweissystem aufnehmen, da man jetzt nicht mehr $SP' \models SP$, sondern SP'_1 **op** $SP'_2 \models SP$ zeigen muß. Das Ziel besteht also darin, für einen geeigneten Satz von Modularisierungsoperatoren ein *kompositionelles Beweissystem* zu entwickeln. Logische Äquivalente zu den bekannten Modularisierungs-Operatoren zu finden, ist eine ebenso schwierige wie lohnende Arbeit. Insbesondere ist zu unter-

suchen, inwieweit kompositionelle Ansätze, die es im Bereich des Model-Checking [LX90] [Sti85] [Win90] bereits gibt, übertragbar sind. Weiterhin wäre es interessant zu untersuchen, wo genau μHML bezüglich der Ausdrucksstärke in der Hierachie von modalen und temporalen Logiken [Sti92] steht. Ausdrucksschwächere Logiken, wie z.B. CTL^* [EH86], ließen sich dann mit Hilfe von Makrodefinitionen wie in Beispiel 2.28 in μHML codieren. Das entwickelte Beweissystem könnte man dann auch für die semantische Implikation in diesen Logiken benutzen, indem man die zu prüfende Implikation gemäß der Makrodefinitionen in eine μHML Implikation übersezt.

Eine andere Erweiterungsmöglichkeit wäre, zu den vorhandenen Operatoren von μHML weitere Modalitäten wie z.B. relativierte Next- oder Vergangenheits-Operatoren hinzuzunehmen, um in dieser modifizierten Logik dann auch linear-time bzw. past-time Logiken kodieren zu können. Ein Beweissystem für diese neue Logik erhält man aus dem alten System, indem man Regeln für die neuen Modalitäten hinzufügt. Somit bekäme man ein Beweissystem für weitere interessante Programmlogiken.

Da die Beweise schnell lang und unübersichtlich werden, wie man in Abbildung 4.7 sieht, ist eine Rechner-Unterstützung bei der Erstellung der Beweise wünschenswert. In unserer Arbeitsgruppe wird bereits über eine prototypische Implementierung im Beweisprüfer LEGO [LPT89] nachgedacht und erste Versuche in diese Richtung unternommen. LEGO ist eine auf Typentheorie basierende interaktive Umgebung für maschinenunterstütztes Beweisen.

Literaturverzeichnis

[AW91] Henrik Reif Andersen and Glynn Winskel. Compositional checking of satisfaction. In *Sixth Annual Symposium on Logic in Computer Science (LICS) (Amsterdam, The Netherlands)*. IEEE, Computer Society Press, June 1991.

[BK90] Jos C. M. Baeten and Jan-Willem Klop, editors. *Theories of Concurrency: Unification and Extension (CONCUR '90, Amsterdam, The Netherlands)*, volume 458 of *Lecture Notes in Computer Science*. Springer-Verlag, 1990.

[BR72] Jacobus de Bakker and Willem de Roever. A calculus for recursive program schemes. In *Automata, Languages and Programming (ICALP)*, pages 167–196. North-Holland, 1972.

[Cle90] Rance Cleaveland. Tableau-based model checking in the propositional mu-calculus. *Acta Informatica*, 27:725–747, 1990.

[CPS89] Rance Cleaveland, Joachim Parrow, and Bernhard Steffen. The Concurrency Workbench. Technical report, Laboratory for Foundations of Computer Science, University of Edinburgh, January 1989.

[EH86] E.A. Emerson and J.Y. Halpern. "sometimes" and "not never" revisited: on branching versus linear time temporal logic. *Journal of the ACM*, 33:151–178, 1986.

[FL79] M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer Science*, 18:194–211, 1979.

[GS86] Susanne Graf and Joseph Sifakis. A modal characterization of observational congruence on finite terms of ccs. *Information and Control*, 68:125–145, 1986.

[GS87] Susanne Graf and Joseph Sifakis. An expressive logic for a process algebra with silent actions. In B. Banieqbal, H. Barringer, and A. Pnueli, editors, *Temporal Logics in Specification*, volume 398 of *Lecture Notes in Computer Science*, pages 44–61. Springer-Verlag, 1987.

[HM85] Matthew Hennessy and Robin Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, 1985.

[Hol89] Sören Holmström. A refinement calculus for specifications in Hennessy-Milner Logic with recursion. *Formal Aspects of Computing*, 1(3):242–272, 1989.

[HP73] P. Hitchcock and D. M. R. Park. Induction rules and termination proofs. In *Automata, Languages and Programming (ICALP)*, pages 225–251. North-Holland, 1973.

[Koz83] Dexter Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[Lar88] Kim Larsen. Proof systems for hennessy-milner logic with recursion. In M. Dauchet and M. Nivat, editors, *Trees in Algebra and Programming (CAAP '88)*, volume 299 of *Lecture Notes in Computer Science*, pages 215–230. Springer-Verlag, 1988.

[Lar90] Kim Larsen. Ideal specification formalism = expressivity + compositionality + decidability + testability + ... In Baeten and Klop [BK90], pages 33–56.

[LPT89] Zhaohui Luo, Randy Pollack, and Paul Taylor. How to use LEGO (a preliminary user's manual). Technical Report 27, Laboratory for Foundations of Computer Science, University of Edinburgh, October 1989.

[LX90] Kim Larsen and Liu Xinxin. Compositionality through an operational semantics of contexts. In M[ichael] S. Paterson, editor, *Proceedings of ICALP '90*, volume 443 of *Lecture Notes in Computer Science*, pages 526–539. Springer-Verlag, 1990.

[Par81] D. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Fifth GI Conference on Theoretical Computer Science*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer-Verlag, 1981.

[Plo81] Gordon Plotkin. A structural approach to operational semantics. Report DAIMI FN-19, Computer Science Department, Aarhus University, September 1981.

[Pnu85] Amir Pnueli. Linear and branching structures in the semantics of reactive systems. In W. Brauer, editor, *Twelfth Colloquium on Automata, Languages and Programming (ICALP) (Nafplion, Greece)*, volume 194 of *Lecture Notes in Computer Science*, pages 15–32. Springer-Verlag, 1985.

[Roe74] Willem de Roever. *Recursive Program schemes: Semantics and proof theory.* PhD thesis, Free University, Amsterdam, 1974.

[SdB69] Dana Scott and Jacobus de Bakker. A theory of programs. unpublished manuscript, IBM, Vienna, 1969.

[SE89] Robert S. Streett and E. Allan Emerson. An automata theoretic decision procedure for the propositional mu-calculus. *Information and Computation*, 81(3):249–264, 1989.

[Sti85] Colin Stirling. A complete modal proof system for a subset of SCCS. In H. Ehrig, C. Floyd, M. Nivat, and J. Thatcher, editors, *Mathematical Foundations of Software Development, Volume1: Colloquium on Trees in Algebra and Programming (CAAP '85)*, volume 185 of *Lecture Notes in Computer Science*. Springer-Verlag, 1985.

[Sti91] Colin Stirling. An introduction to modal and temporal logics for ccs. In A[kinori] Yonezawa and T[akayasu] Ito, editors, *Concurrency: Theory, Languages, and Architecture (UK/Japan Workshop 1989)*, volume 491 of *Lecture Notes in Computer Science*, pages 2–20. Springer-Verlag, 1991.

[Sti92] Colin Stirling. Modal and temporal logics. In Samson Abramsky, Dov Gabbay, and Thomas Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2: Computational Structures. Oxford University Press, 1992.

[SW89] Colin Stirling and David Walker. Local model checking in the modal mu-calculus. Technical Report ECS-LFCS-89-78, Laboratory for Foundations of Computer Science, University of Edinburgh, May 1989.

[SW90] Colin Stirling and David Walker. A general tableau technique for verifying temporal properties of concurrent programs (extended abstract). In *Semantics for Concurrency*, pages 1–15. Springer, 1990.

[Win89] Glynn Winskel. A note on model checking the modal ν-calculus. In G. Ausiello, M. Dezani-Ciancaglini, and S. Ronchi Della Rocca, editors, *Sixteeth Colloquium on Automata, Languages and Programming (ICALP) (Stresa, Italy)*, volume 372 of *Lecture Notes in Computer Science*, pages 761–772. Springer-Verlag, 1989.

[Win90] Glynn Winskel. On the compositional checking of validity. In Baeten and Klop [BK90], pages 481–501.