Vollständigkeitsnachweis eines Beweissystems für Hennessy-Milner-Logik mit Rekursion

Diplomarbeit 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

Begonnen am 1. November 1991
Abgegeben am 31. März 1992

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 31. März 1992

Danksagung

Wir bedanken uns bei allen Mitgliedern des Dienstagsclubs, insbesondere bei unserem Betreuer Norbert Götz für die ständige Unterstützung und bei Terry Stroup für die Hinführung zu dieser Arbeit. Darüberhinaus möchten wir uns auch bei Colin Stirling, Wolfgang Degen sowie bei Michaela Huhn und Peter Niebert für ihre Ratschläge und Kritik bedanken.

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 $\mathbf{p}$,$\mathbf{q}$,$\mathbf{r}$... . 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=($\mathcal{P}$, $\{\xrightarrow{a}, a \in \mathsf{Act}\}$) 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* $T_1 = (\mathcal{P}_1, \{\stackrel{a}{\longrightarrow}, a \in \mathsf{Act}\})$ *und* $T_2 = (\mathcal{P}_2, \{\stackrel{a}{\longrightarrow}, a \in \mathsf{Act}\})$. *Eine binäre Relation* $R \subseteq \mathcal{P}_1 \times \mathcal{P}_2$ *ist eine starke Bisimulation, wenn für alle* $(\mathbf{p},\mathbf{q}) \in \mathrm{R}$ *gilt:*

1. *wenn* $\mathbf{p} \stackrel{a}{\longrightarrow} \mathbf{p}'$, *dann gibt es ein* $\mathbf{q}'$ *mit* $\mathbf{q} \stackrel{a}{\longrightarrow} \mathbf{q}'$ *und* $(\mathbf{p}', \mathbf{q}') \in R$,
2. *wenn* $\mathbf{q} \stackrel{a}{\longrightarrow} \mathbf{q}'$, *dann gibt es ein* $\mathbf{p}'$ *mit* $\mathbf{p} \stackrel{a}{\longrightarrow} \mathbf{p}'$ *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. An vergleichbaren Arbeiten gibt es eine Arbeit von Kozen [Koz83], die einen eingeschränkten Teil des propositionalen μ–Kalkls behandelt. Das System von [HN92], in dem die Einschränkung von Kozen abgeschwächt wird, dient als Grundlage einer Implementierung, die in die Concurrency Workbench [CPS89] eingebunden wird. Für den vollen propositionalen μ–Kalkül ist nur ein automatentheoretisches Entscheidungsverfahren bekannt [SE89].

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 unter der Einschränkung von trennenden Sequenzen. Der Beweis der Korrekheit und Vollständigkeit des Systems findet sich in Kapitel 5. Das 6.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 unendliches Prozeßverhalten 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 $T = (\mathcal{P},\{\xrightarrow{a}, a \in \mathsf{Act}\})$, 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 "$\mathbf{p}$ hat die Eigenschaft φ" oder "$\mathbf{p}$ erfüllt die Spezifikation φ", in Zeichen $\mathbf{p} \models \varphi$, wenn $\mathbf{p} \in [\![\, \varphi \,]\!]$. Weiterhin führen wir folgende Abkürzungen ein:*

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

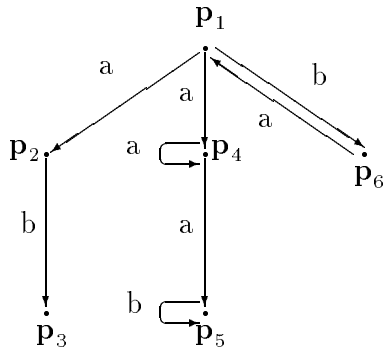


Abbildung 2.1: Transitionssystem für Beispiel 2.6

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* $\mathbf{p}$ *und* $\mathbf{q}$ *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*

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

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*

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

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

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

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

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

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

5. *es ist ein Übergang möglich*

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

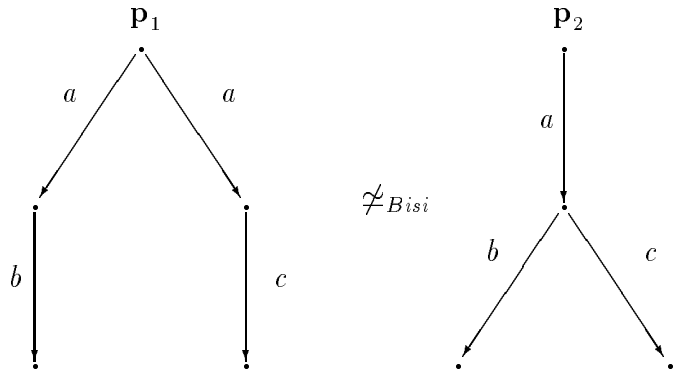
6. *nach allen Übergängen ist noch ein weiterer möglich*

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

7. *es ist höchstens ein Übergang möglich*

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

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



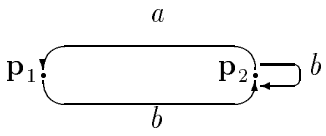
dann gilt z.B.:

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

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

Problematisch ist die Formulierung von unendlichem Verhalten.

Beispiel 2.7 *Sei das folgende Transitionssystem* $T = (\{\mathbf{p}_1, \mathbf{p}_2\}, \{\stackrel{a}{\longrightarrow}, \stackrel{b}{\longrightarrow}\})$ *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.

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.

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

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

Definition 2.8 (HML_{Var} -Syntax) *Sei* Var *eine Menge von propositionalen Variablen. Dann ist HML_{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 HML_{Var} definieren, das heißt, welche Prozeßmenge sie bezeichnen.

Definition 2.9 (HML_{Var} -Semantik) *Bei gegebener Variablenbelegung σ und gegebenem Transitionssytem $T = (\mathcal{P}, \{\stackrel{a}{\longrightarrow}, a \in \mathsf{Act}\})$ ist die durch eine Formel $\varphi \in HML_{\mathsf{Var}}$ 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 $\mathbf{p}$ *erfüllt φ gemäß σ, in Zeichen $\mathbf{p} \models_\sigma \varphi$, wenn $\mathbf{p} \in [\![\varphi]\!]\sigma$.*

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 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).

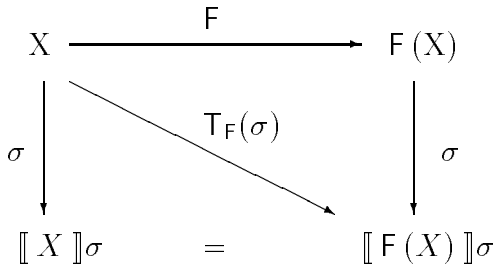


Abbildung 2.2: Transformation von σ durch F

Wir interessieren uns jetzt für diejenigen Variablenbelegungen σ, für die diese Semantiken übereinstimmen: $[\![X]\!]\sigma = [\![X]\!]\mathsf{T}_\mathsf{F}(\sigma)$, X $\in$ Var . Diese Bedingung, die man an eine Variablenbelegung stellt, werden wir im weiteren durch die modale Gleichung X $\equiv^T$ 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 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 $\equiv^T$ 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.

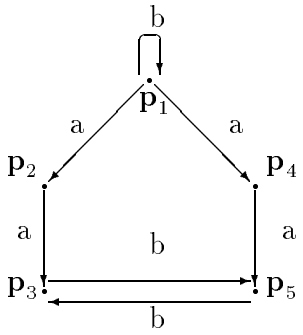


Abbildung 2.3: Transitionssystem für Beispiel 2.12

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 Präals 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

- $\sigma_\mu := \bigcap\{\sigma \mid \mathsf{T_F}(\sigma) \subseteq \sigma\}$ *ist die kleinste Lösung,*
- $\sigma_\nu := \bigcup\{\sigma \mid \sigma \subseteq \mathsf{T_F}(\sigma)\}$ *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*:*

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

und μ *ist das Supremum der folgenden Postfixpunkte:*

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

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]. Aus diesem Grund werden wir von nun an tt und ff nur noch als Abkürzung für die entsprechenden Fixpunktformeln verwenden. Jetzt können wir die vollständige formale Semantik von HML mit Rekursion angeben.

[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.

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 :*

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

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

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

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 $\nu^0 X.\varphi :=$ tt und $\nu^{i+1} X.\varphi := \varphi[X := \nu^i X.\varphi]$, wofür wir abkürzend φ^{i+1}(tt) schreiben; und sei $\mu^0 X.\varphi :=$ ff sowie $\mu^{i+1} X.\varphi := \varphi[X := \mu^i X.\varphi]$, wofür wir φ^{i+1}(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 φ^i(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

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

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".

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:

- $[\![\nu^i X]\!]\sigma_\mathcal{P} = [\![X]\!]\mathsf{T}^i_\varphi\,(\sigma_\mathcal{P})$
- $[\![\nu X.\varphi]\!] = [\![X]\!](\bigcap\limits_{i\in\omega} \mathsf{T}^i_\varphi\,(\sigma_\mathcal{P})) = \bigcap\limits_{i\in\omega} [\![X]\!]\mathsf{T}^i_\varphi\,(\sigma_\mathcal{P}) = \bigcap\limits_{i\in\omega} [\![\nu^i X.\varphi]\!] = [\![\bigwedge\limits_{i\in\omega} \nu^i X.\varphi]\!]$
- $[\![\mu^i X]\!]\sigma_\emptyset = [\![X]\!]\mathsf{T}^i_\varphi\,(\sigma_\emptyset)$
- $[\![\mu X.\varphi]\!] = [\![X]\!](\bigcup\limits_{i\in\omega} \mathsf{T}^i_\varphi\,(\sigma_\emptyset)) = \bigcup\limits_{i\in\omega} ([\![X]\!]\mathsf{T}^i_\varphi\,(\sigma_\emptyset)) = \bigcup\limits_{i\in\omega} [\![\mu^i X.\varphi]\!] = [\![\bigvee\limits_{i\in\omega} \mu^i X.\varphi]\!]$

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 Approximation 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 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.

[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.

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.

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

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*.

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)

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

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.

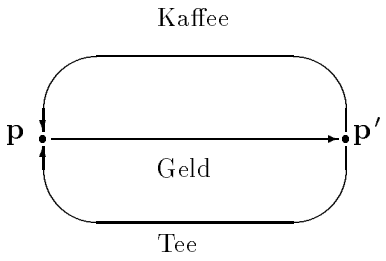


Abbildung 3.4: Kaffee - und Teeautomat

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 **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 $\langle Tee\rangle$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 $\langle Tee\rangle$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 μHML-Formel: "Es gilt φ_2 jetzt oder nach allen Übergängen gilt φ_2 irgendwann". Das legt folgende Formulierung nahe:

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

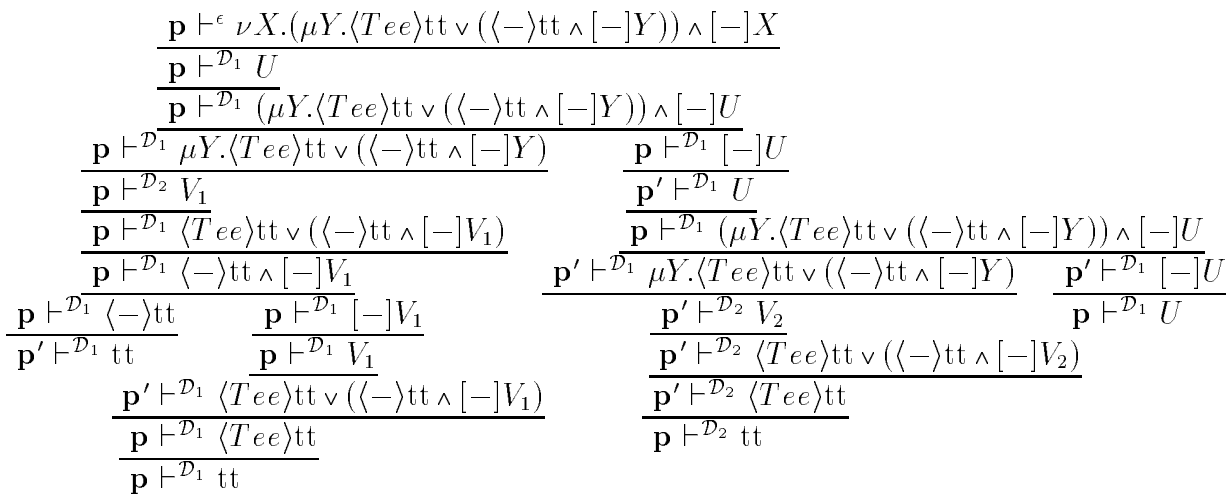


Abbildung 3.5: Beispielableitung

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 $\mathbf{p}$ 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.

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 $\mu X.X$.

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

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

Kapitel 4

Beweissystem für μHML

4.1 Einleitung

In diesem Kapitel kommen wir nun zum Kern der Arbeit, dem Beweissystem für μHML. Wir erweitern ein Gentzen-System für den HML -Anteil der Sprache um Regeln zur Behandlung von Fixpunkten, wobei wir die Idee des vorgestellten Model-Checkers verwenden. Dieses erweiterte System erlaubt, im Gegensatz zu Hilbert-Systemen, ein zielgerichtetes Vorgehen bei der Beweisführung. Nach der Präsentation unseres Beweissystems zeigen wir dessen Korrektheit und Vollständigkeit für einen eingeschränkten Teil von μHML. An vergleichbaren Arbeiten gibt es eine Arbeit von Kozen [Koz83], die einen etwas stärker eingeschränkten Teil des propositionalen μ–Kalküls behandelt. Das System von [HN92], in der die Einschränkung von Kozen abgeschwächt und sprachtheoretisch beschrieben wird, dient als Grundlage einer Implementierung, die in die Concurrency Workbench [CPS89] eingebunden wird. Diese Einschränkung haben wir für unser System übernommen. Für den vollen propositionalen μ–Kalkül gibt es bisher nur ein automaten-theoretisches Entscheidungsverfahren [SE89].

Im nächsten Abschnitt stellen wir das eigentliche Beweissystem vor, also die Regeln sowie die Abbruch- und Erfolgsbedingungen. Das Kapitel schließt mit einem ausführlichen Beispiel, an dem wir die Arbeitsweise des Beweissystems demonstrieren.

4.2 Das Beweissystem

In diesem Abschnitt stellen wir den Kern unserer Arbeit vor, ein Beweissystem für μHML_t, den sogenannten *trennenden* Anteil von μHML. Dieses System ist ein Gentzen-artiger Sequenzenkalkül, der es erlaubt, zielgerichtete Beweise für die semantische Implikation zwischen Formeln 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, führen wir noch einige Begriffe und notationelle Vereinbarungen ein.

Aussgehend von der Definition der Erfülltheitsrelation $\models_T$ definieren wir zunächst den

Begriff der semantischen Folgerung, den wir ebenfalls durch $\models_T$ symbolisieren. 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.*

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

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

Die Implikationsrelation $\models_T$ (wie auch $\models_{\mathcal{T}}$) ist mit einer Menge von Transitionssystemen parametrisiert. Je nachdem, wie man diese Menge wählt, bekommt man unterschiedliche Relationen $\models_T$ und $\models_{\mathcal{T}}$. Wählt man zum Beispiel $\mathcal{T}_R$ als die Menge von Transitionssystemen, deren Übergangsrelation $\xrightarrow{a}$ reflexiv ist, so gilt beispielsweise $[a]\varphi \models_{\mathcal{T}_R} \varphi$, eine Implikation, die nicht gilt, wenn man beliebige Transitionssysteme betrachtet. Bei uns steht $\mathcal{T}$ immer für die Klasse aller Transitionssysteme über einem gegebenen Alphabet Act. Im folgenden schreiben wir oft $\models$ anstelle von $\models_T$.

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 $\bigwedge_{i=1}^{n} \gamma_i \models^{\mathcal{D}} \bigvee_{j=1}^{m} \delta_j$. Das bedeutet, Mengen auf der linken Seite des $\vdash^{\mathcal{D}}$ stehen für die Konjunktion ihrer Formeln, Mengen auf der rechten 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 $\bigvee_{j=1}^{m} \delta_j$.

Die syntaktische Identität von μHML-Formeln bezeichnen wir mit $\equiv$. Bei gegebener Deklarationsliste $\mathcal{D}$ für Fixpunktformeln steht $\mathcal{D}(\varphi)$ 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 ist $\mathcal{D}(\Delta)$ elementweise definiert. Die semantische Implikation bezüglich einer Konstantendeklaration $\Gamma \models^{\mathcal{D}} \Delta$ steht dann als Abkürzung für $\mathcal{D}(\Gamma) \models \mathcal{D}(\Delta)$. Ein wichtiger Begriff im Zusammenhang mit den Deklarationslisten ist die sog. *syntaktische Äquivalenz* von Formeln und Sequenzen.

Definition 4.2 (Syntaktische Äquivalenz)

- *Zwei Formeln φ_1 und φ_2 heißen* syntaktisch äquivalent *bezüglich einer Konstantendeklaration $\mathcal{D}$, wenn gilt:*

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

- *Zwei Formelmengen* $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ *und* $\Gamma' = \{\gamma'_1, \ldots, \gamma'_m\}$ *heißen* syntaktisch äquivalent *bezüglich einer Konstantendeklaration* $\mathcal{D}$*, wenn gilt:*

$$\Gamma \simeq_{\mathcal{D}} \Gamma, \quad :\Leftrightarrow \quad \mathcal{D}(\Gamma) = \mathcal{D}(\Gamma') \text{ , wobei } = \text{ die Mengengleichheit ist.}$$

- *Zwei Sequenzen* $\Gamma \vdash^{\mathcal{D}_1} \Delta$ *und* $\Gamma' \vdash^{\mathcal{D}_2} \Delta'$ *heißen* syntaktisch äquivalent*, wenn gilt:*

$$\Gamma \vdash^{\mathcal{D}_1} \Delta \simeq \Gamma' \vdash^{\mathcal{D}_2} \Delta' \quad :\Leftrightarrow \quad \mathcal{D}_1(\Gamma) = \mathcal{D}_2(\Gamma') \quad \text{und} \quad \mathcal{D}_1(\Delta) = \mathcal{D}_2(\Delta').$$

Man beachte, daß zwei syntaktisch äquivalente Formel bzw. Formelmengen auch semantisch übereinstimmen. Diese Definition der syntaktischen Äquivalenz werden wir später in der Beschreibung der Abbruchkriterien verwenden. Ebenfalls in den Abbruchkriterien benötigen wir eine Notation für die Negation einer Formel, da wir keinen expliziten nicht-Operator haben; wir werden dann $\overline{\varphi}$ für die duale Formel von φ schreiben.

Zum Abschluß müssen wir nun noch präzisieren, welche Teilsprache von μHML wir betrachten. Der sogenannte *trennende* Anteil, für den unser System korrekt und vollständig ist, wird in [HN92] folgendermaßen definiert.

Definition 4.3 (Aktive Variable) *Sei* φ *eine geschlossene Formel und* ψ *eine eine echte Unterformel von* φ*, mit* $\psi \prec \varphi$ *abgekürzt. Dann bezeichnen wir mit* V_φ *die Liste aller Fixpunktvariablen, die zwischen* ψ *und* φ *liegen:* $V_\varphi = (X_n, \ldots, X_1)$ *mit* $\varphi \succeq \sigma X_n.\chi_n \succ \ldots \succ \sigma X_1.\chi_1 \succ \psi$*. Eine Variable* X_i *aus* V_φ *heißt* aktiv *in* ψ*, falls* $\psi[X_1 := \sigma X_1.\chi_1] \ldots [X_{i-1} := \sigma X_{i-1}.\chi_{i-1}]$ *ein freies Vorkommen von* X_i *enthält. Dabei steht* σ *für den größten oder kleinsten Fixpunkt.*

Definition 4.4 (Erreichbarkeitssprache) *Sei* $\Sigma_{\langle\rangle} := \{\langle a\rangle | a \in \mathcal{A}\}$ *und* $\Sigma_{[\,]} := \{[a] | a \in \mathcal{A}\}$ *sowie* $\Sigma := \Sigma_{\langle\rangle} \cup \Sigma_{[\,]}$*. Die* Erreichbarkeitssprache *von* X *bezüglich einer Unterformel* $\psi \prec \varphi(X)$*,* $\mathcal{L}(X, \psi) \subseteq \Sigma^*$ *wird induktiv definiert durch:*

1. $\mathcal{L}(X, \psi) = \emptyset$*, falls* X *nicht aktiv in* ψ*.*
2. $\mathcal{L}(X, X) = \{\varepsilon\}$*, wobei* ε *das leere Wort ist.*
3. $\mathcal{L}(X, \psi_1 \vee \psi_2) = \mathcal{L}(X, \psi_1) \cup \mathcal{L}(X, \psi_2)$.
4. $\mathcal{L}(X, \psi_1 \wedge \psi_2) = \mathcal{L}(X, \psi_1) \cup \mathcal{L}(X, \psi_2)$.
5. $\mathcal{L}(X, \langle a\rangle\psi') = \langle a\rangle \circ \mathcal{L}(X, \psi') = \{\langle a\rangle w;\ w \in \mathcal{L}(X, \psi')\}$
6. $\mathcal{L}(X, [a]\psi') = [a] \circ \mathcal{L}(X, \psi') = \{[a]w;\ w \in \mathcal{L}(X, \psi')\}$
7. $\mathcal{L}(X, \sigma Y.\psi') = \mathcal{L}(Y, \psi')^* \circ \mathcal{L}(X, \psi')$.
8. $\mathcal{L}(X, Y) = \mathcal{L}(Y, \psi')^* \circ \mathcal{L}(X, \psi')$*, wobei* $\sigma Y.\psi' \prec \varphi$ *die zu* Y *gehörende Fixpunktformel ist.*

Definition 4.5 ($\mu\mathbf{HML}_t$)

- *Zwei Symbole σ_1 und σ_2 aus Σ heißen* linkstrennend, *wenn $\sigma_1 \neq \sigma_2$ oder $\sigma_1, \sigma_2 \in \Sigma_{\langle\rangle}$. Dual dazu heißen sie* rechtstrennend, *wenn $\sigma_1 \neq \sigma_2$ oder $\sigma_1, \sigma_2 \in \Sigma_{[]}$*

- *Zwei Wörter $v = v_1 v_2 \ldots v_k$ und $w = w_1 w_2 \ldots w_l$ aus Σ^* heißen* linkstrennend *(resp.* rechtstrennend*), wenn es ein $1 \leq i \leq min(k,l)$ gibt, bei dem v_i und w_i linkstrennend (resp. rechtstrennend) sind.*

- *Zwei Formeln φ und ψ heißen* bezüglich der Variablen X rechtstrennend *(resp.* linkstrennend*), wenn in ihren Erreichbarkeitssprachen $\mathcal{L}(X,\psi)$ und $\mathcal{L}(X,\varphi)$ jedes $w_1 \in \mathcal{L}(X,\psi)$ von jedem $w_2 \in \mathcal{L}(X,\varphi)$ rechtstrennend (resp. linkstrennend) sind.*

- *Eine Formel φ der Sequenz $\Gamma \vdash \Delta$ heißt* trennend, *wenn $\varphi \in \Gamma$ und alle Unterformeln der Form $\psi_1 \wedge \psi_2$ von φ linkstrennend bezüglich aller ihrer aktiven Variablen sind, oder wenn $\varphi \in \Gamma$ und alle Unterformeln der Form $\psi_1 \vee \psi_2$ von φ rechtstrennend bezüglich aller ihrer aktiven Variablen sind.*

- *Die Menge aller trennenden Sequenzen über μHML bezeichnen wir mit μHML_t*

Das besondere Verhalten von trennenden Formeln und Sequenzen bei der Generierung von Tableaus benutzen wir in der Konstruktion des Korrektheitsbeweises.

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 $\vdash$ 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. Man zeigt das Unterziel, um dann daraus das abgeschwächte Ziel zu folgern.

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

Abbruch und Erfolg

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.

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

1. Gentzen-Abbrüche
   (a) $\Gamma, \varphi \vdash^{\mathcal{D}} \Delta, \varphi$
   (b) $\Gamma, \varphi, \overline{\varphi} \vdash^{\mathcal{D}} \Delta$
   (c) $\Gamma \vdash^{\mathcal{D}} \Delta, \varphi, \overline{\varphi}$
2. Keine weitere Regelanwendung möglich ($\emptyset \vdash^{\mathcal{D}} \emptyset$)
3. Erfolgreich Fixpunkt-Induktion
   (a) Man trifft auf eine Sequenz $\Gamma, U \vdash^{\mathcal{D}} \Delta$ mit $U = \mu X.\varphi \in \mathcal{D}$ und oberhalb im Baum ist bereits ein Knoten vorhanden, der mit $\Gamma', U \vdash^{\mathcal{D}'} \Delta'$ beschriftet ist, wobei Γ' syntaktisch äquivalent zu Γ und Δ' syntaktisch äquivalent zu Δ ist und U mindestens einmal abgewickelt wurde.
   (b) Man trifft auf eine Sequenz $\Gamma \vdash^{\mathcal{D}} U, \Delta$ mit $U = \nu X.\varphi \in \mathcal{D}$ und oberhalb im Baum ist bereits ein Knoten vorhanden, der mit $\Gamma', U \vdash^{\mathcal{D}'} \Delta'$ beschriftet ist, wobei Γ' syntaktisch äquivalent zu Γ und Δ' syntaktisch äquivalent zu Δ ist und U mindestens einmal abgewickelt wurde.

Abbildung 4.2: Abbruchbedingungen des Beweissystemes

In Fall 1 bricht man ab, da man gemäß der Aussagenlogik auf ein wahres Blatt gestoßen ist, da die Blätter für die gültigen Implikationen $\wedge\Gamma \wedge \varphi \models^{\mathcal{D}} \varphi \vee \vee\Delta$ bzw. $\wedge\Gamma \wedge \varphi \wedge \overline{\varphi} \models^{\mathcal{D}} \vee\Delta$ bzw. $\wedge\Gamma \models^{\mathcal{D}} \varphi \vee \overline{\varphi} \vee \vee\Delta$ stehen. In Fall 2 muß man notgedrungen abbrechen, da keine weitere Regel mehr anwendbar ist. Solch ein Blatt entspricht semantisch $\mathrm{tt} \models \mathrm{ff}$ und ist somit erfolglos.

3(a) und 3(b) sind die interessanten Fälle. Man ist bei der Ableitung auf eine semantische Situation gestoßen, die im Baum in äquivalenter Form schon einmal aufgetreten ist. Aufgrund der Art der Fixpunkte und ihrer Abwicklung hat man hier eine erfolgreich Fixpunkt-Induktion durchgeführt. 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. Ein weiterer wichtiger Unterschied zwischen unserem Beweissystem

und dem Model-Checker ist, daß wir bis auf den trivialen negativen Abbruch $\emptyset \vdash^{\mathcal{D}} \emptyset$ keine negativen Abbrüche haben, also insbesondere nicht festlegen, wann ein Fixpunktinduktion erfolglos durchgeführt wurde. Dadurch gibt es in unserem System auch unendliche Tableaus, also Tableaus mit wenigstens einem unendlichen Pfad. Somit erhalten wir auch eine etwas modifizierte Charakterisierung wann ein Tableau erfolgreich ist.

Definition 4.6 *Ein maximales Tableau heißt* erfolgreich*, wenn alle Pfade endlich sind und kein Blatt mit $\emptyset \vdash^{\mathcal{D}} \emptyset$ beschriftet ist.*

Die beweistheoretische Implikation ist dann wie folgt definiert.

Definition 4.7 (beweistheoretische Implikation)

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

Für dieses System werden wir zeigen, daß es korrekt und vollständig für $\mu\mathrm{HML}_t$ ist.

Theorem 4.8 (Korrektheit und Vollständigkeit) *Für beliebige trennende Sequenzen $\Gamma \vdash \Delta$:*

$$\Gamma \vdash \Delta \textit{ gdw. } \Gamma \models \Delta.$$

In den Abschnitten 5.1 und 5.2 beweisen wir dieses Theorem.

4.3 Beispiele

Die Arbeitsweise des Beweissystemes soll nun noch zum Abschluß anhand eines Beispieles gezeigt werden. Zu beweisen sei die folgende Aussage: *"Wenn es einen Pfad gibt, bei dem die Eigenschaft φ unendlich oft gilt, so existiert ein Pfad, auf dem es immer eine Fortsetzung gibt, in der irgendwann φ 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_1 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"? Intuitiv 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:

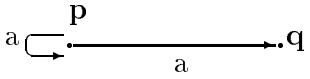
$$\nu X.\psi \wedge ([-]\mathrm{ff} \vee \langle - \rangle X)$$

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_1

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

Drückt dies nun die gewünschte Eigenschaft φ_1 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. Man hat folglich mit dieser Formel die Eigenschaft φ_2, nicht jedoch φ_1 beschrieben. Zum Beispiel erfüllt der Prozeß **p** in folgendem Transitionssystem diese Eigenschaft φ_2 (wobei **q** die Eigenschaft φ erfüllt, **p** hingegen nicht), aber es gibt offensichtlich von **p** ausgehend keinen Pfad, bei dem φ unendlich oft gilt.



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 *verschränkt* 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_1:

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

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. 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 $\mathcal{D}_1 = ((R = \nu X.\langle - \rangle(\mu Y.(\varphi \wedge X) \vee \langle - \rangle Y)))$ und $\mathcal{D}_2 = \mathcal{D}_1 \cdot (T = \nu X.((\mu Y.\varphi \vee \langle - \rangle Y) \wedge ([-]\mathrm{ff} \vee \langle - \rangle X)))$ Danach werden die Konstanten gemäß ihrer Deklaration wieder expandiert. Die ersten Ableitungsschritte lauten somit:

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

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

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\langle - \rangle(\mu Y.(\varphi \wedge R) \vee \langle - \rangle Y) \vdash^{\mathcal{D}_2} \mu Y.\varphi \vee \langle - \rangle Y}{\langle - \rangle(\mu Y.(\varphi \wedge R) \vee \langle - \rangle Y) \vdash^{\mathcal{D}_3} U_1}}{\langle - \rangle(\mu Y.(\varphi \wedge R) \vee \langle - \rangle Y) \vdash^{\mathcal{D}_3} \varphi \vee \langle - \rangle U_1}}{\langle - \rangle(\mu Y.(\varphi \wedge R) \vee \langle - \rangle Y) \vdash^{\mathcal{D}_3} \varphi, \langle - \rangle U_1}}{\langle - \rangle(\mu Y.(\varphi \wedge R) \vee \langle - \rangle Y) \vdash^{\mathcal{D}_3} \langle - \rangle U_1}$$

In $\mathcal{D}_3$ wurde die Konstante U_1 für die Eigenschaft "Es gibt einen Pfad, auf dem irgendwann einmal 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:

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

Wiederum spaltet sich der Beweis auf: Der eine Zweig $\varphi \wedge R \vdash^{\mathcal{D}_4} U_1$ endet nach drei Schritten erfolgreich mit der Sequenz $\varphi, R \vdash^{\mathcal{D}_4} \varphi, \langle - \rangle U_1$. Der zweite Zweig benötigt noch einige Ableitungsschritte bis eine Sequenz ($S_1 \vdash^{\mathcal{D}_4} U_1$) auftaucht, die es bereits einmal weiter oben gab. An dieser Stelle wird nach den Kriterien aus Abbildung 4.2 erfolgreich abgebrochen.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\langle - \rangle S_1 \vdash^{\mathcal{D}_4} U_1}{\langle - \rangle S_1 \vdash^{\mathcal{D}_4} \varphi \vee \langle - \rangle U_1}}{\langle - \rangle S_1 \vdash^{\mathcal{D}_4} \varphi, \langle - \rangle U_1}}{\langle - \rangle S_1 \vdash^{\mathcal{D}_4} \langle - \rangle U_1}}{S_1 \vdash^{\mathcal{D}_4} U_1}$$

In diesem Blatt ist auf der linken Seite die Konstante S_1 enthalten, die für einen kleinsten Fixpunkt steht. Darüberhinaus wurde diese Konstante zwischen dem erstmaligen Aufteten der Sequenz und diesem Blatt einmal expandiert. Damit ist das Blatt erfolgreich.

In Abbildung 4.3 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.

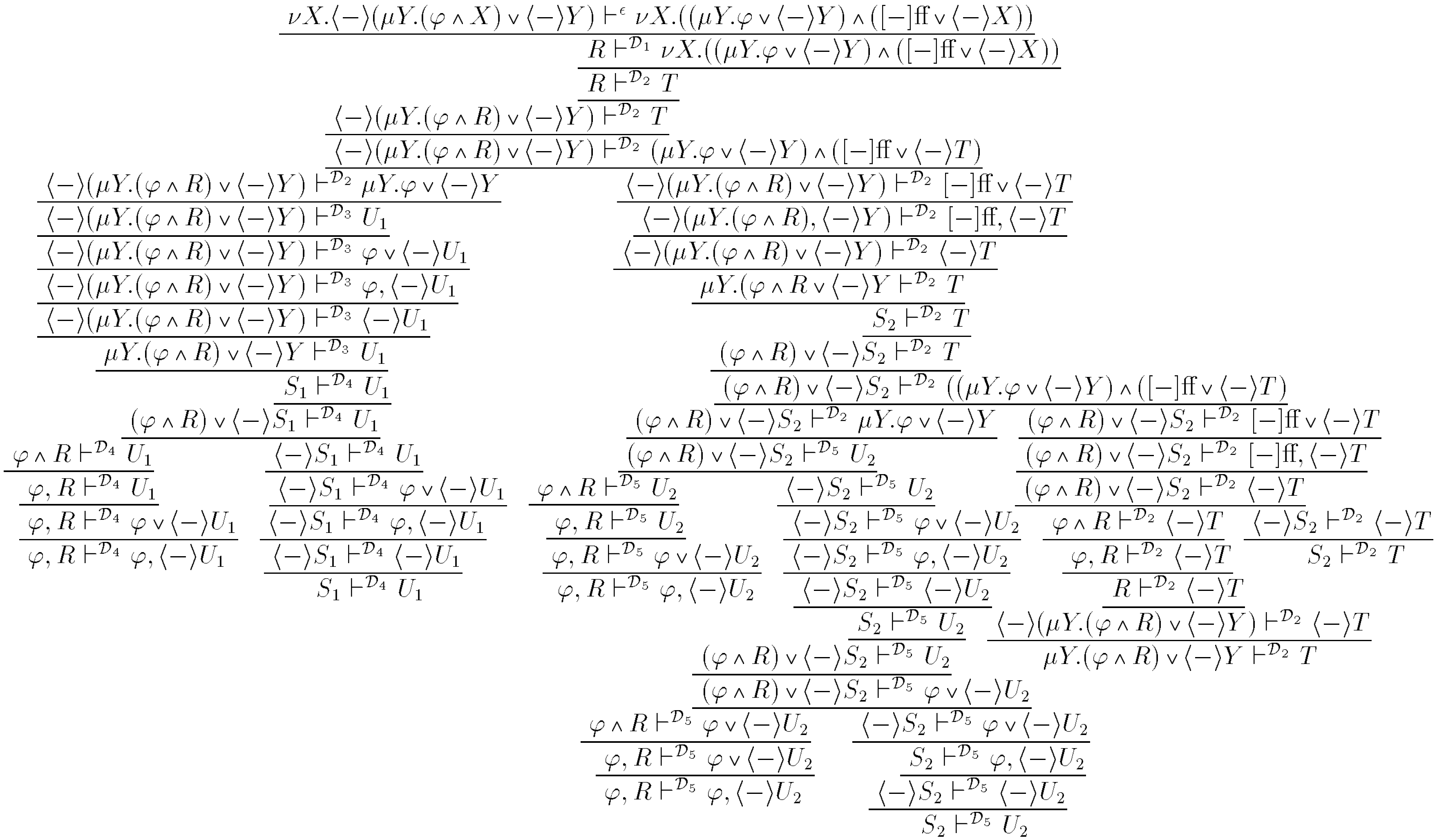


Abbildung 4.3: Beispielableitung

Kapitel 5

Die Korrektheit und Vollständigkeit

5.1 Die Korrektheit

In diesem Abschnitt zeigen wir die Korrektheit des Beweissystems für $\mu\mathrm{HML}_t$. Für die Konstruktion des Beweises benötigen wir einige Lemmata und Sätze, die wir jetzt erläutern.

Lemma 5.1 (Backward-Soundness) *Für jede Regel aus Abbildung 4.1 folgt die Erfülltheit des Zieles aus der Erfülltheit der entsprechenden Unterziele der Regel.*

Beweis zu 5.1
Für die Regeln ($\wedge \vdash$), ($\vdash \vee$), ($weak_1$) und ($weak_2$) gilt die Aussage trivialerweise bzw. ist der entsprechende Beweis sehr einfach.
Die Beweise für die Regeln ($\vdash \wedge$), ($\vee \vdash$), ($\langle a \rangle \vdash$) und ($\vdash [a]$) sind Schlußfolgerungen der propositionalen Logik zusammen mit Eigenschaften der Erfüllbarkeitsrelation $\models_T^{\mathcal{D}}$.

($\vdash \wedge$) Per Annahme gilt $\wedge\Gamma \models^{\mathcal{D}} \varphi_1 \vee \vee\Delta$ und $\wedge\Gamma \models^{\mathcal{D}} \varphi_2 \vee \vee\Delta$. Zu zeigen ist, daß dann auch $\wedge\Gamma \models^{\mathcal{D}} (\varphi_1 \wedge \varphi_2) \vee \Delta$ gilt. Es gilt per Annahme für alle $T \in \mathcal{T}$ und alle $\mathbf{p} \in \mathcal{P}_T$:

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

Somit gilt dann also auch $\wedge\Gamma \models^{\mathcal{D}} (\varphi_1 \wedge \varphi_2) \vee \Delta$.

($\vdash \vee$) Der Beweis ist analog.

($\langle a \rangle \vdash$) Per Annahme gilt $\varphi \wedge \wedge\Gamma \models^{\mathcal{D}} \vee\Delta$. Wir haben zu zeigen, daß dann auch $\langle a \rangle \varphi \wedge \wedge\{[a]\gamma; \gamma \in \Gamma\} \models^{\mathcal{D}} \vee\{\langle a \rangle \delta; \delta \in \Delta\}$ gilt. Ausgeschrieben heißt das:

$\forall T \in \mathcal{T}.\forall \mathbf{p} \in \mathcal{P}_T$ .wenn $\mathbf{p} \models^{\mathcal{D}}_T \langle a\rangle\varphi \wedge \wedge\{[a]\gamma; \gamma \in \Gamma\}$ dann $\mathbf{p} \models^{\mathcal{D}}_T \bigvee_{\delta\in\Delta}\langle a\rangle\delta$. Es gelte also für beliebiges $T \in \mathcal{T}$ und beliebiges $\mathbf{p} \in \mathcal{P}_T$:

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

was zu zeigen war.

$(\vdash [a])$ Der Beweis ist dual zu obigem Beweis.

Bei den letzten vier Regeln $(\sigma \vdash)$, $(\vdash \sigma)$, $(Konst \vdash)$ und $(\vdash Konst)$ folgt die Backwardsoundness wieder unmittelbar. Bei Regel $(\sigma \vdash)$ und $(\vdash \sigma)$ ist 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 $[\![\, \nu X.\varphi(X) \,]\!] = [\![\, \varphi(\nu X.\varphi(X)) \,]\!]$.
□

Die somit bewiesene Backward-Soundness ist ein wichtiger Ansatzpunkt in der Konstruktion des Korrektheitsbeweises, genauso wie die nun folgenden Lemmata 5.3 und 5.4. Für deren Beweis verwenden wir die sog. *finite-model*-Eigenschaft von μHML.

Satz 5.2 (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.

Eine wichtige Eigenschaft von Logiken mit der finite-model-Eigenschaft ist die Tatsache, daß einen Formel bereits gültig ist, wenn sie in allen endlichen Modellen gültig ist. Aufgrund dieser Tatsache können wir die nächsten beiden Lemmata zeigen.

Lemma 5.3 *Für beliebige endliche Formelmengen $\Gamma, \Delta \subseteq \mu HML$ und eine beliebige Formel $\varphi \in \mu HML$ gilt:*

$$
\begin{array}{l}
\textit{Wenn} \wedge\Gamma \not\models^{\mathcal{D}} \nu X.\varphi \vee \vee\Delta \textit{ dann existiert ein } n \in \omega \textit{ mit} \\
\wedge\Gamma \models^{\mathcal{D}} \nu^n X.\varphi \vee \vee\Delta \textit{ und } \wedge\Gamma \not\models^{\mathcal{D}} \nu^{n+1} X.\varphi \vee \vee\Delta.
\end{array}
$$

Beweis zu 5.3

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

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

Da $\nu^0 X.\varphi = \mathrm{tt}$, gilt auf jeden Fall $\wedge\Gamma \models \nu^0 X.\varphi \vee \vee\Delta$ und per Induktion somit $\forall n \in \omega.\ \wedge\Gamma \models \nu^n X.\varphi \vee \vee\Delta$. Ausgeschrieben bedeutet das:

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

Als Widerspruch zu Voraussetzung wollen wir jetzt folgern, daß

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

was gleichbedeutend ist mit $\wedge\Gamma \models^{\mathcal{D}} \nu X.\varphi \vee \vee\Delta$. Sei jetzt $T \in \mathcal{T}, \mathbf{p} \in \mathcal{P}_T$ und aufgrund der Annahme (1) gilt: wenn $\mathbf{p} \models^{\mathcal{D}}_T \wedge\Gamma$ dann $\forall n \in \omega. \mathbf{p} \models^{\mathcal{D}}_T \nu^n X.\varphi \vee \vee\Delta$. Wir machen nun eine Fallunterscheidung, ob $\mathbf{p}$ endlich oder unendlich ist.

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

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

Es gibt also zumindest keinen *endlichen* Prozeß, der ein Gegenmodell zu $\wedge\Gamma \models^{\mathcal{D}} \nu X.\varphi \vee \vee\Delta$ ist.

Fall *b*)**:** $\mathbf{p}$ ist ein unendlicher Prozeß. Nun kann man, da φ nicht notwendigerweise stetig ist, nicht direkt schließen, daß

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

Nimmt man für $\mathbf{p}$ das Gegenteil an, nämlich

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

Das bedeutet, der unendliche Prozeß $\mathbf{p}$ erfüllt die Formel $\wedge\Gamma \wedge \overline{(\nu X.\varphi \vee \vee\Delta)}$. Wegen Satz 5.2 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. Somit haben wir gezeigt, daß aus der Annahme (1) folgt: $\wedge\Gamma \models^{\mathcal{D}} \nu X.\varphi \vee \vee\Delta$. Dies widerspricht jedoch der Voraussetzung des Lemmas. □

Lemma 5.4 *Für beliebige endliche Formelmengen $\Gamma, \Delta \subseteq \mu HML$ und eine beliebige Formel $\varphi \in \mu HML$ gilt:*

$$\begin{array}{l} \textit{Wenn } \wedge\Gamma \wedge \mu X.\varphi \not\models^{\mathcal{D}} \vee\Delta \textit{ dann existiert ein } n \in \omega \textit{ mit} \\ \wedge\Gamma \wedge \mu^n X.\varphi \models^{\mathcal{D}} \vee\Delta \textit{ und } \wedge\Gamma \wedge \mu^n X.\varphi \not\models^{\mathcal{D}} \vee\Delta. \end{array}$$

Beweis zu 5.4
Sei also $\wedge\Gamma \wedge \mu X.\varphi \not\models^{\mathcal{D}} \vee\Delta$ und gelte als Widerspruchsannahme

$$\forall n \in \omega.\ \text{wenn}\ \wedge\Gamma \wedge \mu^n X.\varphi \models \vee\Delta\ \text{dann}\ \wedge\Gamma \wedge \mu^{n+1} X.\varphi \models \vee\Delta. \qquad (2)$$

Da $\mu^0 X.\varphi = \mathrm{ff}$, gilt auf jeden Fall $\wedge\Gamma \wedge \mu^0 X.\varphi \models \vee\Delta$ und per Induktion somit $\forall n \in \omega.\ \wedge\Gamma \wedge \mu^n X.\varphi \models \vee\Delta$. Ausgeschrieben bedeutet das:

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

Als Widerspruch zur Voraussetzung wollen wir wiederum schließen, daß

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

was gleichbedeutend mit $\wedge\Gamma \wedge \mu X.\varphi \models^{\mathcal{D}} \vee\Delta$ ist. Sei wiederum $T \in \mathcal{T}, \mathbf{p} \in \mathcal{P}_T$ und aufgrund der Annahme (2) gilt: wenn $\exists n \in \omega. \mathbf{p} \models^{\mathcal{D}}_T \wedge\Gamma \wedge \mu^n X.\varphi$ dann $\mathbf{p} \models^{\mathcal{D}}_T \vee\Delta$. Wie beim vorhergehenden Lemma machen wir wieder eine Fallunterscheidung, ob $\mathbf{p}$ endlich oder unendlich ist.

Fall *a*)**:** $\mathbf{p}$ ist ein endlicher Prozeß. Dann ist φ stetig und es gilt nach Satz 2.26 und dem analogen Schluß wie im vorhergehenden Lemma, daß

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

Fall *b*)**:** $\mathbf{p}$ ist ein unendlicher Prozeß. In gleicher Weise wie im vorhergehenden Lemma bekommt man aus der Widerspruchsannahme; daß $\mathbf{p} \models^{\mathcal{D}}_T \wedge\Gamma \wedge \mu X.\varphi \wedge \overline{\vee\Delta}$. Wegen Satz 5.2 gibt es dann auch wiederum ein endliches Modell $\overline{\mathbf{p}}$ 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 *a* steht. Somit haben wir gezeigt, daß aus der Annahme (2) folgt:$\wedge\Gamma \wedge \mu X.\varphi \models^{\mathcal{D}} \vee\Delta$, was wiederum der Voraussetzung des Lemmas widerspricht.

□

Damit ist man nun in der Lage, die Korrektheit (die eine Hälfte von Theorem 4.8) zu zeigen. Die grundlegende Beweisidee, bei der wir uns an die Konstruktion im Korrektheitsbeweis des Model-Checkers in [SW89] angelehnt haben, ist folgende: Wir nehmen an, daß wir für eine Sequenz ein erfolgreiches Tableau konstruiert haben, ohne daß die der Sequenz entsprechende semantische Implikation gültig ist. Aufgrund der bewiesenen

Backward-Soundness der Regeln muß dann bereits ein semantisch falsches Blatt existieren. Mittels eines Induktionsargumentes sowie mit Hilfe der Lemmata 5.3 und 5.4 zeigen wir, daß es dann unendlich viele falsche Blätter geben muß, was im Widerspruch zu Endlichkeit des erfolgreichen Tableaus steht.

Beweis zu 4.8($\Rightarrow$)
Gegeben sei $\Gamma_0 \vdash \Delta_0$, das heißt es gibt ein erfolgreiches Tableau τ mit der Wurzel $\Gamma_0 \vdash^{\epsilon} \Delta_0$ und nehmen an, daß Δ_0 keine semantische Folgerung von Γ_0 ist, also $\Gamma_0 \not\models \Delta_0$ gilt. Aufgrund der Backward-Soundess muß es dann bereits ein semantisch falsches Blatt $\Gamma' \vdash^{\mathcal{D}'} \Delta'$ geben.

Dieses Blatt kann nicht von der Form $\Gamma, \varphi \vdash^{\mathcal{D}} \varphi, \Delta$, $\Gamma \vdash^{\mathcal{D}} \overline{\varphi}, \varphi, \Delta$ oder $\Gamma, \varphi, \overline{\varphi} \vdash^{\mathcal{D}} \Delta$ sein, da für solche Blätter die semantische Implikation trivialerweise immer gilt. Folglich muß es von der Form $\Gamma \vdash^{\mathcal{D}} U, \Delta$ bzw. $\Gamma, U \vdash^{\mathcal{D}} \Delta$ sein, wobei die Konstante U und die Formelmengen Γ, Δ die Erfolgsbedingungen aus Abbildung 4.2 erfüllen.

Unter allen semantisch falschen Blättern wählt man nun eines, bei dem die Konstante U die folgende Bedingung erfüllt: Oberhalb der Einführung von U wurde keine andere Konstante U' eingeführt, die ebenfalls zu eimen semantisch falschen Blatt führt. Wir haben somit ein falsches Blatt und eine zugehörige Konstante ausgewählt. Je nachdem, ob diese Konstante links oder rechts vom $\vdash^{\mathcal{D}'}$ steht, unterscheiden wir zwei Fälle:

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

Dieser Teilbaum τ' kann nun mehrere falsche Blätter der Form $\widehat{\Gamma} \vdash^{\hat{\mathcal{D}}} U, \widehat{\Delta}$ mit der selben Konstanten U enthalten. Für jedes dieser Blätter gilt also $\widehat{\Gamma} \not\models^{\hat{\mathcal{D}}} U, \widehat{\Delta}$. Damit gibt es nach Lemma 5.3 für jedes dieser Blätter ein $n \in \omega$ mit

$$\widehat{\Gamma} \models^{\hat{\mathcal{D}}} \nu^n X.\varphi \vee \bigvee \widehat{\Delta} \text{ und } \widehat{\Gamma} \not\models^{\hat{\mathcal{D}}} \nu^{n+1} X.\varphi \vee \bigvee \widehat{\Delta}.$$

Aus all diesen Blättern des Teilbaumes τ' wählt man nun ein Blatt $\Gamma \vdash^{\mathcal{D}} U, \Delta$ mit minimalem n. Nun transformiert man τ' so, daß die Kontstante U an jeder Stelle durch die endliche Approximation $\nu^n X.\varphi$ ersetzt wird. Den neuen Baum nennen wir τ'^*. Da die Sequenzen im Tableau trennend sind, enthalten die Formelmengen Γ und Δ nicht die Konstante U, so daß deren Semantik durch die Transformation nicht verändert wird. Dies ist in Abbildung 5.1 dargestellt.

In τ'^* gilt entsprechend der Wahl des Blattes $\Gamma \models^{\mathcal{D}} \nu^n X.\varphi, \Delta$, das heißt durch die Transformation ist aus dem falschen Blatt $\Gamma \vdash^{\mathcal{D}} U, \Delta$ in τ' das wahre Blatt $\Gamma \vdash^{\mathcal{D}} \nu^n X.\varphi, \Delta$ in τ'^* geworden. Ebenso ist jetzt auch der Knoten mit der Sequenz $\Gamma \vdash^{\mathcal{D}''} \nu^n X.\varphi, \Delta$ in τ'^* semantisch gültig. Der Nachfolgerknoten von $\Gamma \vdash^{\mathcal{D}''} \nu^n X.\varphi\Delta$ ist allerdings mit $\Gamma \vdash^{\mathcal{D}''} \varphi[X := \nu^n X.\varphi], \Delta$ beschriftet und, da $\varphi[X := \nu^n X.\varphi]$ für $\nu^{n+1} X.\varphi$ steht, gemäß der Wahl von n falsch. Aufgrund der Backward-Soundness muß τ'^* also mindestens ein weiteres falsches Blatt besitzen. Dieses semantisch falsche Blatt stammt nicht von einem

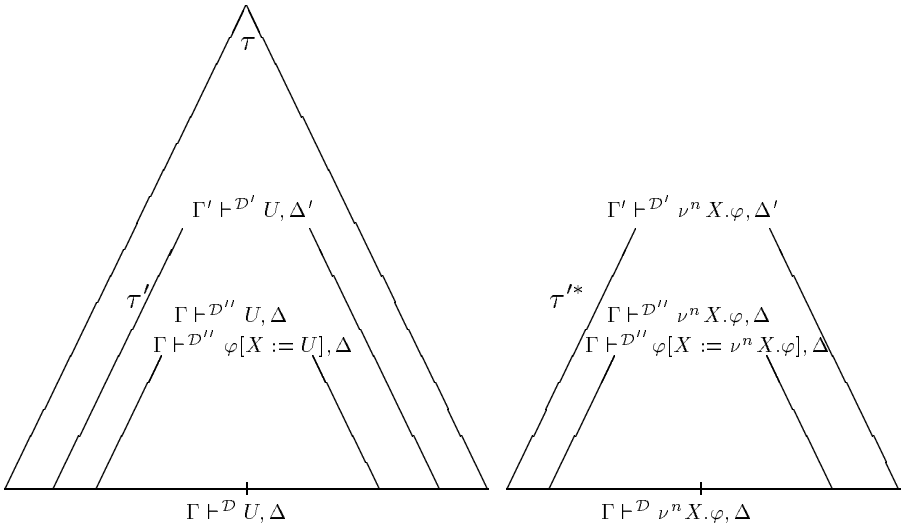


Abbildung 5.1: Skizze zum Korrektheitsbeweis

Fixpunktabbruch mit der Konstante U, da alle diese Blätter durch die Transformation und die Wahl von n im neuen Baum gültig sind. Also muß es ein anderes Blatt $\widehat{\Gamma} \vdash^{\mathcal{D}} V, \widehat{\Delta}$ bzw. $\widehat{\Gamma}, V \vdash^{\mathcal{D}} \widehat{\Delta}$ mit einer von U verschiedenen Konstante V geben, welches semantisch falsch ist. Dieses V muß, aufgrund der Wahl von U, im Tableau unterhalb von U eingeführt worden sein. Zusammen mit Fall 2 kann man nun die gesamte Konstruktion mit diesem Blatt im Baum τ'^* wiederholen. Das bedeutet jedoch, daß das ursprüngliche Tableau unendlich viele verschiedene Konstanten enthält, was im Widerspruch zur Endlichkeit des Tableaus steht.

Fall 2:
Dieser Fall ist zum Fall 1 dual. Anstelle des größten Fixpunktes auf der rechten Seite muß man nun einen kleinsten Fixpunkt auf der linken Seite des $\vdash^{\mathcal{D}}$ behandeln. Man wählt das kleinste n, für das gilt: $\wedge\Gamma \wedge \mu^n X.\varphi \models^{\mathcal{D}} \vee\Delta$ und $\wedge\Gamma \wedge \mu^{n+1} X.\varphi \not\models^{\mathcal{D}} \vee\Delta$. 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. Somit kann man die gesamte Konstruktion in τ'^* wiederholen. Da Blätter, die in τ'^* erfolglos sind auch insbesondere in τ falsch sind, muß es also im ursprünglichen Tableau unendlich viele verschiedene Konstanten gegeben haben (die insbesondere zu semantisch falschen Blättern geführt haben). Dies steht aber im Widerspruch zur Endlichkeit des Tableaus. Folglich sind auch die Blätter der Form $\Gamma \vdash^{\mathcal{D}} U, \Delta$ bzw. $\Gamma, U \vdash^{\mathcal{D}} \Delta$ gültig und somit wegen der Backward-Soundness auch die Sequenz an der Wurzel des Tableaus. □

5.2 Die Vollständigkeit

Nachdem wir die Korrektheit des Kalküls für trennende Sequenzen gezeigt haben, beweisen wir nun die entsprechende Vollständigkeit. Im gesamten Abschnitt werden wir uns auf Grund der Symmetrie der Regeln (siehe Beobachtung 5.5) auf die Betrachtung der rechten Seite des $\vdash$ beschränken.

Der klassische Beweisansatz um die Vollständigkeit einer Logik zu zeigen, ist i.A. folgender: man zeigt, daß jede konsistente[1] Formelmenge ein Modell hat, was äquivalent ist zur Aussage: ist eine Formel nicht herleitbar, so ist sie auch nicht gültig, und es gibt somit ein Gegenmodell für die Formel.

Um aus der Konsistenz einer Formel ein Modell für diese Formel zu konstruieren, haben wir uns eine Konstruktion ausgedacht, die auf eine *Fixpunkt-Elimination* hinausläuft. Die Beweisskizze sieht wie folgt aus: Eine konsistente Formel φ ist gegeben, ein Modell für φ gesucht.

1. Sukzessive Ersetzung der μ-Fixpunkte in φ durch je eine geeignet gewählte, endliche Approximation. Die neue Formel φ' ist weiterhin endlich und konsistent, und enthält höchstens noch ν-Fixpunkte (Lemma 5.7).

2. Wir ersetzen die Formel φ' durch die Formelmenge φ'^{∞}, die anstelle der ν-Fixpunkte jeweils alle endlichen Approximationen dieser Fixpunkte enthält. Diese Menge ist i.A. unendlich und ist weiterhin konsistent (Lemma 5.11).

3. Diese Menge φ'^{∞}, die wir endlich erzeugt nennen, besteht nur aus HML -Formeln. Da unser Beweissystem für diesen Sprachanteil stark vollständig ist (Theorem 5.12), hat diese unendliche Menge auf Grund ihrer Konsistenz auch ein Modell.

[1] Eine Formelmenge Δ heißt $\vdash$ -*konsistent*, gdw. es keine endliche Teilmenge $\{\varphi_1, \varphi_2, \ldots, \varphi_n\} \subseteq \Delta$ gibt, so daß $\varphi_1 \vee \ldots \vee \varphi_n \vdash \mathrm{ff}$.

4. Mittels *Filtration* zeigen wir, daß endlich erzeugte HML -Formelmengen die sogenannte *small-model Eigenschaft* (Korollar 5.19) besitzen. Somit bekommen wir ein endliches Modell T für φ'^{∞}.

5. Auf endlichen Modellen sind ν-Fixpunkte und die Menge ihrer Approximationen aber semantisch äquivalent. Somit ist T also auch ein Modell für φ', und dann natürlich auch für φ, da T für jeden μ-Fixpunkt jeweils eine endliche Approximation erfüllt. Folglich ist T das gesuchte Modell für φ.

In der folgenden Beobachtung begründen wir, warum wir uns auf die Betrachtung der rechten Seite des $\vdash$ beschränken können, und trotzdem eine Aussage für das ganze System erhalten.

Beobachtung 5.5 *Für alle Formelmengen $\Gamma, \Delta \subseteq \mu HML$ gilt folgende Äquivalenz:*

$$\Gamma \vdash \Delta \; gdw. \; \vdash \overline{\Gamma} \vee \Delta$$

Diese Eigenschaft, in der $\overline{\Gamma}$ die duale Formel zu Γ bezeichnet, besitzen alle Gentzen-Systeme mit $\neg$-Regel und auch unser System, da zu jeder Regel auch die duale Regel in unserem System enthalten ist; d.h., unsere Regeln sind unter Dualität abgeschlossen und gleiches gilt für die Abbruch- und Erfolgs-Bedingungen.

Der nächste Satz bildet die Grundlage für die Ersetzung von μ-Fixpunkten in endlichen konsistenten Formeln durch eine entsprechende Approximation. Dabei benutzen wir $\nvdash\varphi$ als Abkürzung dafür, daß es kein erfolgreiches Tableau mit $\emptyset \vdash^{\epsilon} \varphi$ an der Wurzel gibt.

Satz 5.6 *Für jede Formel $\psi(\nu X.\varphi) \in \mu HML$ gilt:*

$$wenn \; \nvdash\psi(\nu X.\varphi) \;\; dann \;\; \exists n \in \omega.\nvdash\psi(\nu^n X.\varphi)$$

Beweis zu 5.6
Es gilt $\nvdash\psi(\nu X.\varphi)$, was heißt, bei allen Tableaus für $\psi(\nu X.\varphi)$ gibt es mindestens ein nichterfolgreiches Blatt $\emptyset \vdash \emptyset$ oder einen unendlichen Pfad. Sei nun also das n größer als $\mid 2^{FL(\psi(\nu X.\varphi)} \mid$ und man versuche, ein erfolgreiches Tableau τ_1 für die Sequenz $\emptyset \vdash \psi(\nu^n X.\varphi)$ zu entwickeln. Zugleich wende man jede Regel bei der Entwicklung von τ_1 in gleicher Weise auch auf einen hierbei entstehenden Ableitungsbaum τ_2 für $\emptyset \vdash \psi(\nu X.\varphi)$ an. Dieser gleicht somit τ_1 bis auf die Tatsache, daß der Fixpunkt $\nu X.\varphi$ in ersterem durch die Approximation $\nu^n X.\varphi$ repräsentiert wird und somit in letzterem an geeigneten Stellen die Regeln für die Konstanteneinführung ($\vdash \sigma$) bzw. Konstantenexpansion ($\vdash Konst$) angewandt werden, während dies in τ_1 nicht der Fall ist. Nun ist die Behauptung: in jedem der teilabgeleiteten Tableaus τ_1 gibt es ein Blatt, welches nicht erfolgreich ist. Dazu betrachte man eine Sequenz in τ_2, welche, fortgesetzt, zum Mißerfolg führen wird oder bereits selbst ein nichterfolgreiches Blatt darstellt. Laut Annahme muß es eine solche geben. Die entsprechende Sequenz in τ_1 kann dann kein erfolgreiches Blatt sein. Als Gegenteilsannahme sei diese

Sequenz ein erfolgreiches Blatt und muß mithin von einer der folgenden Formen sein:
• $\vdash \varphi, \overline{\varphi}, \Gamma$. Die entsprechende Sequenz in τ_2 ist dann auch von dieser Form und ebenfalls erfolgreich, ein Widerspruch.
• $\vdash U, \Gamma$ und erfolgreich aufgrund einer Wiederholung des Fixpunktes. Die Wiederholung betrifft also nicht den "Fixpunkt" $\nu^n X.\varphi$, die entsprechende Sequenz in τ_2 weist diese Wiederholung in derselben Form auf und ist damit gleichfalls erfolgreich.
• $\vdash U, \Gamma$, wobei das U als Konstante für den Fixpunkt $\nu X.X = \mathrm{tt}$ steht, der durch die n-fache Abwicklung des $\nu^n X.\varphi$ freigelegt wurde, also dem $\nu^0 X.\varphi$ entspricht. Dies jedoch bedeutet, daß der Fixpunkt $\nu X.\varphi$ in τ_2 mehr als $\mid 2^{FL(\psi(\nu X.\varphi))} \mid$ mal abgewickelt worden sein muß. Somit muß es vor der Sequenz in τ_2 bereits eine erfolgreiche Wiederholung und damit einen erfolgreichen Abbruch gegeben haben. □

Korollar 5.7 (Abschluß für kleinste Fixpunkte) *In einer konsistenten Formel sind alle kleinsten Fixpunkte durch geeignete Approximationen ersetzbar, ohne daß die Konsistenz verloren geht.*

Beweis zu 5.7
Eine einfache Folgerung aus Satz 5.6. Kontrapositiv besagt dieser, daß aus der Konsistenz von $\psi(\mu X.\varphi)$ gefolgert werden kann, daß es ein endliches n gibt, sodaß $\psi(\mu^n X.\varphi)$ ebenfalls konsistent ist. Damit lassen sich in einer Formel die kleinsten Fixpunkte sukzessive z.B. von außen nach innen ersetzen. Bei jeder einzelnen Ersetzung können weiter innenliegenden Fixpunkte vervielfältigt werden, durch die Endlichkeit der Approximationen endet der Prozeß jedoch nach endlicher Zeit. □

Als nächsten Konstruktionschritt im Vollständigkeitsbeweis wollen wir die ν-Fixpunkte eliminieren. Dazu zeigen wir, daß man zu einer konsistenten Formelmenge Δ, die nur ν-Fixpunkte enthält, beliebige Approximationen der Fixpunktformeln in die Menge mit aufnehmen kann, ohne die Konsistenz der Menge zu verlieren. Für den Beweis definieren wir uns zunächst die Approximation einer Fixpunktformel und benutzen dann die folgenden zwei Lemmata.

Definition 5.8 (Diagonale Approximierung) *Sei φ eine Formel aus μHML, wobei o.B.d.A. alle Rekursionsvariablen unterschiedlich seien. $Exp(\varphi)$ sei dann diejenige Formel, bei der alle Fixpunkte auf der obersten Ebene in φ genau einmal expandiert wurden. $App^n(\varphi)$ ist nun in folgender Weise induktiv definiert:*

$$
\begin{array}{rcl}
App^n(\mathrm{tt}) & := & \mathrm{tt} \\
App^n(\mathrm{ff}) & := & \mathrm{ff} \\
App^n(\varphi_1 \wedge \varphi_2) & := & App^n(\varphi_1) \wedge App^n(\varphi_1) \\
App^n(\varphi_1 \vee \varphi_2) & := & App^n(\varphi_1) \vee App^n(\varphi_1) \\
App^n([a]\varphi) & := & [a] App^n(\varphi) \\
App^n(\langle a \rangle \varphi) & := & \langle a \rangle App^n(\varphi) \\
App^0(\nu X.\varphi) & := & \mathrm{tt} \\
App^{n+1}(\nu X.\varphi) & := & App^n(Exp(\varphi))
\end{array}
$$

Lemma 5.9 (Kompositionalität von $\vdash$) *Für beliebige Formeln der Gestalt $\chi = \varphi(\psi_1, \psi_2, \ldots \psi_n)$ und $\chi' = \varphi(\psi'_1, \psi'_2, \ldots \psi'_n)$ aus μHML, wobei φ ein rein modaler Kontext ist, gilt: Wenn für alle $i \in \{1, \ldots n\}$ gilt: $\psi_i \vdash \psi'_i$, dann $\varphi(\psi_1, \psi_2, \ldots \psi_n) \vdash \varphi(\psi'_1, \psi'_2, \ldots \psi'_n)$.*

Beweis zu 5.9
Induktion über den Formelaufbau von φ.

1. $\varphi = \emptyset$, dann ist nicht zu zeigen;

2. $\varphi = \varphi_1 \vee \varphi_2$:

$$\dfrac{\dfrac{\varphi_1 \vee \varphi_2 \ \vdash \varphi'_1 \vee \varphi'_2}{\varphi_1 \vdash \varphi'_1 \vee \varphi'_2}}{\dfrac{\varphi_1 \vdash \varphi'_1, \varphi'_2}{\varphi_1 \vdash \varphi'_1}} \qquad \dfrac{\dfrac{\varphi_1 \vee \varphi_2 \ \vdash \varphi'_1 \vee \varphi'_2}{\varphi_2 \vdash \varphi'_1 \vee \varphi'_2}}{\dfrac{\varphi_2 \vdash \varphi'_1, \varphi'_2}{\varphi_2 \vdash \varphi'_2}}$$

   wobei die beiden Unterziele per Induktionsannahme ableitbar sind.

3. $\varphi = \varphi_1 \wedge \varphi_2$

$$\dfrac{\dfrac{\varphi_1 \wedge \varphi_2 \ \vdash \varphi'_1 \wedge \varphi'_2}{\varphi_1 \wedge \varphi_2 \vdash \varphi'_1}}{\dfrac{\varphi_1, \varphi_2 \vdash \varphi'_1}{\varphi_1 \vdash \varphi'_1}} \qquad \dfrac{\dfrac{\varphi_1 \wedge \varphi_2 \ \vdash \varphi'_1 \wedge \varphi'_2}{\varphi_1 \wedge \varphi_2 \vdash \varphi'_2}}{\dfrac{\varphi_1, \varphi_2 \vdash \varphi'_2}{\varphi_2 \vdash \varphi'_2}}$$

   wobei die beiden Unterziele per Induktionsannahme ableitbar sind.

4. $\varphi = \langle a \rangle \varphi_1$

$$\dfrac{\langle a \rangle \varphi_1 \vdash \langle a \rangle \varphi'_1}{\varphi_1 \vdash \varphi'_1}$$

   wobei das Unterziel per Induktionsannahme ableitbar ist.

5. $\varphi = [a] \varphi_1$

$$\dfrac{[a] \varphi_1 \vdash [a] \varphi'_1}{\varphi_1 \vdash \varphi'_1}$$

   wobei das Unterziel per Induktionsannahme ableitbar ist.

6. $\varphi = \nu X.\varphi_1$ braucht man nicht zu betrachten, da dieser Fall laut Annahme ausgeschlossen ist.

□

Dieses Lemma benötigen wir im Beweis der folgenden Aussage.

Lemma 5.10 *Für alle Formeln $\varphi \in \mu$HML gilt:*

$$\forall n \in \omega . \varphi \vdash App^n(\varphi)$$

Beweis zu 5.10
Sei φ von der Gestalt $\psi(\nu X_1.\psi_1, \ldots, \nu X_m.\psi_m)$, wobei die $\nu X_i.\psi_i$ sämtliche Fixpunkte auf der obersten Ebene innerhalb von φ darstellen und somit ψ rein modal ist. Wegen Lemma 5.9 reicht es zu zeigen, daß für jedes n sowie für alle $i \in \{1, 2, \ldots, m\}$ gilt: $\nu X_i.\psi_i \vdash App^n(\nu X_i.\psi_i)$. Dieses zeigen wir mittels Induktion über n.

$n = 0$: $\nu X_i.\psi_i \vdash \mathrm{tt}$ gilt offensichtlich.

$n > 0$: Zu zeigen ist also die Ableitbarkeit von $\nu X_i.\psi_i \vdash App^{n+1}(\nu X_i.\psi_i)$, d.h. von $\nu X_i.\psi_i \vdash App^n(Exp(\nu X_i.\psi_i))$. Dabei ist $\nu X_i.\psi_i$ von der Form $\nu X_i.\psi_i'(\nu Y_1.\chi_1, \ldots, \nu Y_p.\chi_p)$ wobei die $\nu Y_j.\chi_j$ alle an oberster Stufe innerhalb von $\nu X_i.\psi_i$ stehenden Fixpunktformeln sind und damit ψ_i' rein modal ist. $App^n(Exp(\nu X_i.\psi_i))$ ist damit gleich $\psi_i'(App^n(\nu Y_1.\chi_1[X_i := \nu X_i.\psi_i]), \ldots, App^n(\nu Y_p.\chi_p[X_i := \nu X_i.\psi_i]))$. Per Induktionsannahme gilt für alle $j \in \{1, \ldots, p\}$: $\nu Y_j.\chi_j[X_i := \nu X_i.\psi_i] \vdash App^n(\nu Y_i.\chi_j[X_i := \nu X_i.\psi_j])$ und mittels Lemma 5.9 die Behauptung für $n + 1$.

□

Mit Hilfe dieser Vorarbeiten können wir nun den zweiten wichtigen Satz für die Fixpunktelimination zeigen.

Satz 5.11 (Abschluß für größte Fixpunkte) *Für alle konsistenten Formelmengen* $\Delta \cup \{\varphi\}$ *ist auch* $\Delta \cup \{\varphi\} \cup \bigcup_{n\in\omega}\{App^n(\varphi)\}$ *konsistent.*

Beweis zu 5.11
Sei im Widerspruch dazu die Menge $\Delta \cup \{\varphi\} \cup \bigcup_{n\in\omega}\{App^n(\varphi)\}$ inkonsistent. Dann gibt es ein $m \in \omega$, für das die Menge $\Delta \cup \{\varphi\} \cup \bigcup_{n<m}\{App^n(\varphi)\}$ noch konsistent ist, $\Delta \cup \{\varphi\} \cup \bigcup_{n\leq m}\{App^n(\varphi)\}$ jedoch inkonsistent. Aus dieser Tatsache und der propositionalen Vollständigkeit des Systems folgt, daß dann $\Delta \cup \{\varphi\} \cup \bigcup_{n<m}\{App^n(\varphi)\} \cup \{\overline{App^m(\varphi)}\}$ konsistent sein muß und damit auch $\{\varphi, \overline{App^m(\varphi)}\}$. Dies steht im Widerspruch zu Lemma 5.10.

□

Mittels der Lemmata 5.7 und 5.11 sind wir nun in der Lage, eine konsistente Formel durch eine i.A. unendliche aber ebenfalls konsistente Menge von Approximationen zu ersetzen. Dabei ist die Reihenfolge der Ersetzung – zuerst werden alle kleinsten Fixpunkte durch jeweils *eine* passende endliche Approximation ersetzt und danach die größten Fixpunkte durch eine passende *unendliche* Menge von Approximationen – von Bedeutung, da die Existenz eines endlichen n für die kleinsten Fixpunkte in Satz 5.6 davon abhängt, daß die Formel bzw. die Formelmenge, in der ersetzt wird, endlich ist.

Mit der Elimination sowohl der kleinsten als auch der größten Fixpunkte sind wir bei einer konsistenten HML – Formelmenge angekommen. Für diesen Teil der Logik können wir dabei auf ein bekanntes Ergebnis zurückgreifen.

Theorem 5.12 *Das Beweissystem ist stark korrekt und vollständig für HML* [2].

Dieses Ergebnis kann man in [Sti92] nachlesen.

Aufgrund der starken Korrektheit und Vollständigkeit des Beweissystems für HML wissen wir, daß eine konsistente Formelmenge auch ein Modell besitzt. Da sich aber im Verlauf der bisherigen Konstruktion die Semantik beim Übergang von der ursprünglichen μHML–Formel zu der endlich erzeugten Formelmenge geändert hat, reicht die Existenz eines Modells für die Menge der Approximationen alleine noch nicht aus. Damit die Semantik eines größten Fixpunktes mit der Semantik der Menge aller seiner Approximationen übereinstimmt, benötigt man stetige Fixpunktoperatoren. Bei endlichen Transitionssystemen sind alle Fixpunktoperatoren stetig. Was wir also noch brauchen, ist die *finite–model Eigenschaft* von HML$^\infty$. Als HML$^\infty$ bezeichnen wir die Menge der Formelmengen φ^∞, die man erhält, wenn man in einer Formel φ ohne kleinste Fixpunkte die größten Fixpunkte durch alle ihre Approximationen ersetzt. HML$^\infty$ ist damit die Menge aller *endlich erzeugten* Formelmengen. Die finite–model Eigenschaft gilt für den modalen μ–Kalkül [SE89], was aber an dieser Stelle nicht von Nutzen ist, da hieraus nicht auch die finite–model Eigenschaft für HML$^\infty$ folgt. Wir beweisen die etwas stärkere *small–model–Eigenschaft* für HML$^\infty$, die nicht nur die Existenz eines endlichen Modells fordert, sondern darüberhinaus einen funktionalen Zusammenhang zwischen der Länge der zu erfüllenden Formel und der Größe dieses Modells verlangt. Diese Eigenschaft von HML$^\infty$ läßt sich nachweisen, indem man zeigt, daß sich aus jedem Transitionssystem, welches eine Formelmenge φ^∞ erfüllt, durch Äquivalenzklassenbildung auf den Zuständen, *Filtrierung* genannt, eines gewinnen läßt, welches φ^∞ ebenfalls erfüllt und dessen endliche Größe von der Länge der Formel φ abhängt[3].

Bevor wir die Filtrierung eines Transitionssystems definieren können, benötigen wir noch den passenden Äquivalenzbegriff auf den Zuständen. Dazu dienen die drei folgenden Definitionen.

Definition 5.13 (Fischer-Ladner-Abschluß) *Der* Fischer-Ladner-Abschluß *einer geschlossenen μHML-Formel ist die kleinste Menge von Formeln $FL(\varphi)$, die folgende Bedingungen erfüllt:*

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

[2] Die starke Vollständigkeit impliziert die Kompaktheit der Logik. Diese hat zur Folge, daß man auch bei unendlichen Formelmengen aus der Konsistenz auf die Existenz eines Modells schließen kann.

[3] Es sei darauf hingewiesen, daß, obwohl der modale μ–Kalkül, also μHML, die small–model Eigenschaft besitzt [SE89], dies sich jedoch nicht mittels Filtrierung beweisen läßt . Dies ist auf die Anwesenheit von kleinsten Fixpunkten zurückzuführen, deren Erfülltheit im filtrierten, endlichen Transitionssystem verloren geht.

Der Fischer-Ladner-Abschluß einer Formelmenge wird elementweise definiert.

Definition 5.14 (Syntax von νHML) *νHML ist die kleinste Menge von Formeln, die gemäß folgender Syntax erzeugt werden:*

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

Definition 5.15 (Fischer–Ladner–Äquivalenz) *Sei φ eine beliebige νHML– Formel, $\mathrm{T} = (\mathcal{P}, \{\stackrel{a}{\longrightarrow}, a \in \mathsf{Act}\})$ ein Transitionssystem. Die durch den Fischer–Ladner–Abschluß der Formel φ induzierte Äquivalenzrelation $\equiv_{FL(\varphi)} \subseteq \mathcal{P} \times \mathcal{P}$ ist definiert durch:*

$$\mathbf{p}_1 \equiv_{FL(\varphi)} \mathbf{p}_2 \;:\Leftrightarrow\; \forall \psi \in FL(\varphi).\mathbf{p}_1 \models \psi \textit{ gdw. } \mathbf{p}_2 \models \psi$$

Wir definieren das *filtrierte Transitionssystem* aus T mittels Äquivalenzklassenbildung auf den Zuständen von T bezüglich dieser Äquivalenzrelation, die durch den Fischer-Ladner-Abschluß 5.13 von φ induziert wird.

Definition 5.16 (Filtrierung eines Transitionssystems) *Sei φ eine beliebige νHML–Formel und $\mathrm{T} = (\mathcal{P}, \{\stackrel{a}{\longrightarrow}, a \in \mathsf{Act}\})$ ein Transitionssystem. Das filtrierte Transitionssystem ist dann wie folgt definiert:*

$$\begin{array}{rcl} T_{/\equiv_{FL(\varphi)}} & := & (\mathcal{P}_{/\equiv_{FL(\varphi)}}, \{\stackrel{a}{\longrightarrow}, a \in \mathsf{Act}\}) \\ \mathcal{P}_{/\equiv_{FL(\varphi)}} & := & \{[\mathbf{p}]_{\equiv_{FL(\varphi)}}; \mathbf{p} \in \mathcal{P}\} \\ [\mathbf{p}]_{\equiv_{FL(\varphi)}} \stackrel{a}{\longrightarrow} [\mathbf{q}]_{\equiv_{FL(\varphi)}} & :\Leftrightarrow & \exists \mathbf{p}' \in [\mathbf{p}]_{\equiv_{FL(\varphi)}}.\exists \mathbf{q}' \in [\mathbf{q}]_{\equiv_{FL(\varphi)}}.\mathbf{p}' \stackrel{a}{\longrightarrow} \mathbf{q}' \end{array}$$

Es bleibt zu zeigen, daß das filtrierte Transitionssystem die semantischen Eigenschaften bezüglich der Approximationen der Formeln in $FL(\varphi)$ vom ursprünglichen Transitionssystem erbt. Diesem Nachweis dienen die folgenden zwei Lemmata.

Lemma 5.17 *Sei $\varphi \in \nu HML$, $\psi \in FL(\varphi)$ mit $\psi = \psi'(\psi_1, \ldots, \psi_m)$, wobei ψ' rein modal sei. Außerdem seien die Transitionssysteme T und $T_{/\equiv_{FL(\varphi)}}$ gegeben. Dann gilt für alle $\mathbf{p} \in \mathcal{P}$:*

wenn $\mathbf{p} \models \psi^\infty$
und wenn für alle $\mathbf{q} \in \mathcal{P}$ *und alle* $i \in \{1, \ldots, m\}$ *gilt wenn* $\mathbf{q} \models {\psi_i}^\infty$ *dann* $[\mathbf{q}]_{\equiv_{FL(\varphi)}} \models {\psi_i}^\infty$
dann $[\mathbf{p}]_{\equiv_{FL(\varphi)}} \models \psi^\infty$

Beweis zu 5.17
Mittels Induktion über den Formelaufbau von ψ'.

- Falls ψ' leer ist, ist nichts zu zeigen.

- $\psi' = \mathrm{tt}$: trivial.
- $\psi' = \mathrm{ff}$: ebenso.
- $\psi' = \chi_1 \wedge \chi_2$:

$$
\begin{array}{lll}
 & \mathbf{p} \models (\chi_1 \wedge \chi_2)^\infty & \\
\Rightarrow & \mathbf{p} \models \chi_1^\infty \wedge \chi_2^\infty & \\
\Rightarrow & \mathbf{p} \models \chi_1^\infty \text{ und } \mathbf{p} \models \chi_2^\infty & \\
\Rightarrow & [\mathbf{p}]_{\equiv_{FL(\varphi)}} \models \chi_1^\infty \text{ und } [\mathbf{p}]_{\equiv_{FL(\varphi)}} \models \chi_2^\infty & \text{wegen der Induktionsannahme und da } \chi_1, \chi_2 \in FL(\varphi) \\
\Rightarrow & [\mathbf{p}]_{\equiv_{FL(\varphi)}} \models \chi_1^\infty \wedge \chi_2^\infty & \\
\Rightarrow & [\mathbf{p}]_{\equiv_{FL(\varphi)}} \models (\chi_1 \wedge \chi_2)^\infty &
\end{array}
$$

- $\psi' = \chi_1 \vee \chi_2$:

$$
\begin{array}{lll}
 & \mathbf{p} \models (\chi_1 \vee \chi_2)^\infty & \\
\Rightarrow & \mathbf{p} \models \chi_1^\infty \vee \chi_2^\infty & \\
\Rightarrow & \mathbf{p} \models \chi_1^\infty \text{ oder } \mathbf{p} \models \chi_2^\infty & \\
\Rightarrow & [\mathbf{p}]_{\equiv_{FL(\varphi)}} \models \chi_1^\infty \text{ oder } [\mathbf{p}]_{\equiv_{FL(\varphi)}} \models \chi_2^\infty & \text{wegen der Induktionsannahme und da } \chi_1, \chi_2 \in FL(\varphi) \\
\Rightarrow & [\mathbf{p}]_{\equiv_{FL(\varphi)}} \models \chi_1^\infty \vee \chi_2^\infty & \\
\Rightarrow & [\mathbf{p}]_{\equiv_{FL(\varphi)}} \models (\chi_1 \vee \chi_2)^\infty &
\end{array}
$$

- $\psi' = [a]\chi$

$$
\begin{array}{lll}
\text{Sei} & [\mathbf{q}]_{\equiv_{FL(\varphi)}} \in \mathcal{P}_{/\equiv_{FL(\varphi)}} & \text{mit } [\mathbf{p}]_{\equiv_{FL(\varphi)}} \xrightarrow{a} [\mathbf{q}]_{\equiv_{FL(\varphi)}} \text{ beliebig ,} \\
\text{d.h.,} & \exists \mathbf{p}' \in [\mathbf{p}]_{\equiv_{FL(\varphi)}}.\exists \mathbf{q}' \in [\mathbf{q}]_{\equiv_{FL(\varphi)}}.\mathbf{p}' \xrightarrow{a} \mathbf{q}'. & \\
 & \mathbf{p}' \models ([a]\chi)^\infty \ (\text{wegen } \mathbf{p} \ \equiv_{FL(\varphi)} \mathbf{p}') & \\
\Rightarrow & \mathbf{p}' \models [a](\chi^\infty) & \\
\Rightarrow & \mathbf{q}' \models \chi^\infty & \\
\Rightarrow & [\mathbf{q}']_{\equiv_{FL(\varphi)}} \models \chi^\infty & \text{wegen der Induktionsannahme und da } \chi \in FL(\varphi) \\
\Rightarrow & [\mathbf{p}']_{\equiv_{FL(\varphi)}} \models [a](\chi^\infty) & \\
\Rightarrow & [\mathbf{p}']_{\equiv_{FL(\varphi)}} \models ([a]\chi)^\infty &
\end{array}
$$

- $\psi' = \langle a \rangle \chi$:

$$
\begin{array}{lll}
 & \mathbf{p} \models (\langle a \rangle \chi)^\infty & \\
 & \mathbf{p} \models \langle a \rangle (\chi^\infty) & \\
\Rightarrow & \exists \mathbf{q} \in \mathcal{P}.\mathbf{p} \xrightarrow{a} \mathbf{q} \text{ und } \mathbf{q} \models \chi^\infty & \\
\Rightarrow & [\mathbf{p}]_{\equiv_{FL(\varphi)}} \xrightarrow{a} [\mathbf{q}]_{\equiv_{FL(\varphi)}} \text{ und } [\mathbf{q}]_{\equiv_{FL(\varphi)}} \models \chi^\infty & \text{wegen der Induktionsannahme und da } \chi \in FL(\varphi) \\
\Rightarrow & [\mathbf{p}]_{\equiv_{FL(\varphi)}} \models \langle a \rangle (\chi^\infty) & \\
\Rightarrow & [\mathbf{p}]_{\equiv_{FL(\varphi)}} \models (\langle a \rangle \chi)^\infty &
\end{array}
$$

□

Lemma 5.18 *Sei $\varphi \in \nu HML$, $\varphi' \in FL(\varphi)$. Außerdem seien wiederum die Transitionssysteme T und $T_{/\equiv_{FL(\varphi)}}$ gegeben. Dann gilt für alle $\mathbf{p} \in \mathcal{P}$:*

$$\textit{wenn } \mathbf{p} \models \psi^{\infty} \textit{ dann } [\mathbf{p}]_{\equiv_{FL(\varphi)}} \models \psi^{\infty}$$

Beweis zu 5.18
Wir zeigen dies mittels Induktion über n und Induktion über sämtliche größte Fixpunkte gleichzeitig und benutzen dazu $App^n(\varphi')$ aus Definition 5.8.

$$\forall \mathbf{p} \in \mathcal{P}.\forall n \in \omega.\ \text{wenn } \mathbf{p} \models App^n(\varphi') \text{ dann } [\mathbf{p}]_{\equiv_{FL(\varphi)}} \models App^n(\varphi')$$

Sei φ' von der Gestalt $\psi(\nu X_1.\psi_1, \ldots, \nu X_m.\psi_m)$, wobei die $\nu X_i.\psi_i$ sämtliche Fixpunkte auf der obersten Ebene innerhalb von φ' darstellen und somit ψ rein modal ist. Wegen Lemma 5.17 reicht es zu zeigen, daß für jedes n sowie für alle $i \in \{1, 2, \ldots, m\}$ gilt: wenn $\mathbf{p} \models App^n(\nu X_i.\psi_i)$ dann $[\mathbf{p}]_{\equiv_{FL(\varphi)}} \models App^n(\nu X_i.\psi_i)$. Dieses zeigen wir mittels Induktion über n.

- $n = 0$: trivial.
- $n > 0$: Zu zeigen ist also die Folgerung:

$$\text{wenn } \mathbf{p} \models App^{n+1}(\nu X_i.\psi_i) \text{ dann } [\mathbf{p}]_{\equiv_{FL(\varphi)}} \models App^{n+1}(\nu X_i.\psi_i)$$

$$\begin{array}{rl} & \mathbf{p} \models App^{n+1}(\nu X_i.\psi_i) \\ d.h. & \mathbf{p} \models App^n(Exp(\nu X_i.\psi_i)) \\ \Rightarrow & \mathbf{p} \models App^n(\psi_i[X_i := \nu X_i.\psi_i]) \end{array}$$

Diese Formel $\psi[X_i := \nu X_i.\psi_i]$ ist nun von folgender Form $\psi_i'(\nu Y_1.\chi_1[X_i := \nu X_i.\psi_i], \ldots, \nu Y_p.\chi_p[X_i := \nu X_i.\psi_i])$, wobei die $\nu Y_j.\chi_j[X_i := \nu X_i.\psi_i]$ wiederum alle Fixpunkte der obersten Ebene sind. Damit gilt:

$$\begin{array}{rl} & \mathbf{p} \models App^n(\psi_i'(\nu Y_1.\chi_1[X_i := \nu X_i.\psi_i], \ldots, \nu Y_p.\chi_p[X_i := \nu X_i.\psi_i])) \\ \Rightarrow & \mathbf{p} \models \psi_i'(App^n(\nu Y_1.\chi_1[X_i := \nu X_i.\psi_i]), \ldots, App^n(\nu Y_p.\chi_p[X_i := \nu X_i.\psi_i])) \end{array}$$

Da für sämtliche $j \in \{1, \ldots, p\}$ die Formeln $\nu Y_j.\chi_j[X_i := \nu X_i.\psi_i]$ aus dem Fischer–Ladner–Abschluß von φ sind, gilt aufgrund der Induktionsannahme und Lemma 5.17 sofort:

$$[\mathbf{p}]_{\equiv_{FL(\varphi)}} \models App^{n+1}(\varphi)$$

Damit gilt die Induktionsaussage für alle n und daher können wir schließen:

$$\begin{array}{rl} & \mathbf{p} \models \varphi'^{\infty} \\ \Rightarrow & \forall n \in \omega.\mathbf{p} \models App^n(\varphi') \\ \Rightarrow & \forall n \in \omega.[\mathbf{p}]_{\equiv_{FL(\varphi)}} \models App^n(\varphi') \\ \Rightarrow & [\mathbf{p}]_{\equiv_{FL(\varphi)}} \models \varphi'^{\infty} \end{array}$$

Der letzte Schritt ist durch die Endlichkeit des filtrierten Modells gerechtfertigt. □

Korollar 5.19 (small–model Eigenschaft von HML$^\infty$) [4] *Sei φ^∞ die von $\varphi \in \nu HML$ endlich erzeugte Formelmenge. Ist φ^∞ erfüllbar, so ist sie auch erfüllbar in einem Transitionssystem der Größe $2^{|FL(\varphi)|}$.*

Beweis zu 5.19
Eine einfache Folgerung aus Lemma 5.17 und Lemma 5.18. Sei φ^∞ mit $\varphi \in \nu$HML erfüllbar, d.h., es gibt ein Transitionssytem $\mathrm{T} = (\mathcal{P}, \{\xrightarrow{a}, a \in \mathsf{Act}\})$ und $\mathbf{p} \in \mathcal{P}$ mit $\mathbf{p} \models \varphi^\infty$. Dann gilt $[\mathbf{p}]_{\equiv_{FL(\varphi)}} \models \varphi^\infty$, wobei $[\mathbf{p}]_{\equiv_{FL(\varphi)}}$ ein Zustand aus dem *endlichen*, filtrierten Transitionssystem $T_{/\equiv_{FL(\varphi)}}$ der Größe $2^{|FL(\varphi)|}$ ist. □

Theorem 5.20 (Vollständigkeit)

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

Beweis zu 5.20
Ausgehend von einer konsistenten Formel φ kann man nach Korollar 5.7 durch Ersetzung aller kleinsten Fixpunkte zu einer konsistenten Formel φ' aus νHML gelangen. Satz 5.11 liefert eine konsistente Formelmenge $\{\varphi'\} \cup \bigcup_{n\in\omega}\{App^n(\varphi')\}$. Da $\bigcup_{n\in\omega}\{App^n(\varphi')\}$ nur aus HML -Formeln besteht und konsistent ist, besitzt diese Formelmenge ein Modell. Dieses ist ebenfalls ein Modell für φ'^∞. Aufgrund der small–model–Eigenschaft von HML$^\infty$ aus Korollar 5.19 ist dies auch ein Modell für φ' und somit auch für φ, was den Beweis der Vollständigkeit abschließt. □

[4] Die gleiche Aussage gilt auch für νHML, da man die Filtrierung nach $\equiv_{FL(\varphi)}$ auch hierfür durchführen kann, d.h., die Lemmata 5.17 und 5.18 gelten auch für νHML analog, wenn auch nicht für μHML.

Kapitel 6

Ausblick

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

Mögliche Erweiterungen der Arbeit sind einerseits der Ausbau der theoretischen Grundlagen und zum anderen Arbeiten, die die Benutzbarkeit des Systems verbessern.

Was die theoretischen Grundlagen betrifft, steht die Modifikation der Beweisregeln im Vordergrund, um ein Beweissystem für die gesamte Logik zu bekommen. Ein genaue Charakterisierung, wann ein Pfad in einem Tableau erfolglos ist, also die Beschreibung von negativen Abbrüchen wäre interessant im Hinblick auf die Endlichkeit der erzeugten Tableaus.

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 $SP'_1 \ \mathbf{op} \ SP'_2 \models SP$ 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 untersuchen, 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 die Veränderung des Operatorensatzes $\langle a \rangle, [a]$ von μHML. Zum einen könnte man in den Modalitäten kompliziertere Beobachtungen zulassen, wie zum Beispiel Kausalitäten und echte Parallelität, wodurch dann natürlich auch eine andere logische Äquivalenz induziert würde. Ein Beweissystem für diese neue Äquivalenz bekäme man, indem man eine Axiomatisierung der angestrebten Verhaltensäquivalenz zu den bestehenden Regeln des Beweissystems hinzunimmt, um damit die Beobachtungen in den Modalitäten verändern zu können. Eine andere Modifikation des Operatorensatzes wäre die Hinzunahme von z.B. relativierten Next- oder Vergangenheits-Operatoren , um in dieser modifizierten Logik dann auch linear-time bzw. past-time Logiken kodieren zu können. Für diese neue Logik erhält man dann ein Beweissystem, indem man Regeln für die neuen Modalitäten hinzufügt.

Da die Beweise schnell lang und unübersichtlich werden, wie man in Abbildung 4.3 sieht, ist eine Rechner-Unterstützung bei der Erstellung der Beweise wünschenswert. Unser System war der Ausgangspunkt für die Implementierung eines Beweissystems, mit dem man vollautomatisch Verfeinerungschritte verifizieren kann. Diese Implementierung wird jetzt an größeren Beispielen, wie z.B dem Entwurf von asynchronen Schaltungen, auf seine Praxistauglichkeit hin überprüft [Rei92].

Literaturverzeichnis

[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.

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

[HN92] Michaela Huhn and Peter Niebert. *Realisierung eines Beweissystems für Hennessy-Milner-Logik mit Rekursion.* Diplomarbeit, Universität Erlangen, 1992.

[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.

[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.

[Rei92] Tilman Reinhardt. *Spezifikation und Verifikation delay-insensitiver Schaltwerke.* Diplomarbeit, Universität Erlangen, 1992.

[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.

[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 Jos C. M. Baeten and Jan-Willem Klop, editors, *Proceedings of CONCUR '90*, volume 458 of *Lecture Notes in Computer Science*, pages 481–501. Springer-Verlag, 1990.