Interner Bericht

Spezifikation reaktiver Systeme durch temporal stratifizierte Programme

Robi Malik Otto Mayer

7. November 1994

259/94

Fachbereich Informatik

Universität Kaiserslautern · Postfach 3049 · D-67653 Kaiserslautern

Spezifikation reaktiver Systeme durch temporal stratifizierte Programme

Robi Malik Otto Mayer

7. November 1994

259/94

Zusammenfassung

Temporal stratifizierte Programme sind spezielle Logik-Programme auf der Grundlage einer linearen, temporalen Aussagenlogik, mit denen zustandsendliche reaktive Systeme spezifiziert werden können. Dabei wird die Umgebung eines zu implementierenden Steuerungsprogrammes durch eine Menge von PROLOG-ähnlichen Programmklauseln beschrieben; zusätzlich wird eine Sicherheitsbedingung angegeben, die in dem System gelten soll. Die Sprache ist so gestaltet, daß sie für resolutionsbasierte Verfahren zur Verifikation und Synthese von Steuerungsprogrammen geeignet ist. Wir zeigen, daß temporal stratifizierte Programme in ihrer Ausdrucksmächtigkeit endlichen Automaten gleichkommen.

1 Einführung

Unter *reaktiven Systemen* verstehen wir Computer-Programme, die eine ständige Interaktion mit ihrer Umgebung aufrechterhalten [BeBe91]. Solche Programme werden in vielen praktischen Anwendungen benötigt, zum Beispiel in industriellen Steuerungs- und Regelsystemen oder in Prozeßrechensystemen, aber auch in Betriebssystemen sowie bei modernen, grafischen Benutzerschnittstellen. Oft wird die Aufgabe noch dadurch erschwert, daß die Geschwindigkeit der Interaktion nicht vom Programm bestimmt werden kann, sondern vielmehr von der Umgebung diktiert wird.

In diesen Bereich fallen auch viele sicherheitskritische Anwendungen, bei denen Programmfehler mit erheblichen wirtschaftlichen Folgen verbunden sind oder sogar Menschenleben gefährden können. Nicht zuletzt deshalb wird der Ruf nach einer Unterstützung des äußerst schwierigen und kostenintensiven Software-Entwicklungsvorganges für reaktive Systeme durch Methoden der *formalen Spezifikation* und *Verifikation*, sowie einer ganz oder teilweise automatisierten *Programmgenerierung* immer lauter.

In der Literatur wurde eine Vielzahl von Methoden der formalen Spezifikation reaktiver Systeme untersucht. Die ältesten Ansätze beruhen auf endlichen Automaten oder auf Petri-Netzen; später wurden auch imperative, gleichungsorientierte und logische Beschreibungssprachen in Betracht gezogen. Bei den logischen Spezifikationsmethoden scheint Einigkeit darüber zu bestehen, daß reaktive Systeme mit einer *temporalen Logik* adäquat beschrieben werden können, während für die Implementierung im allgemeinen Zustandsdiagramme verwendet werden. Bei vielen Steuerungsproblemen genügt es, eine temporale Aussagenlogik und endliche Automaten zu verwenden. Auf dieser Grundlage konnten inzwischen sehr effiziente Verifikationsverfahren entwickelt werden [CES86].

Dagegen erweist sich das Problem der automatischen Programmgenerierung bisher als wesentlich schwieriger. Hier besteht die Zielvorstellung darin, reaktive Systeme durch abstrakte Beschreibungen zu spezifizieren, aus denen dann das System automatisch erzeugt werden kann. Dazu wurden in der Literatur einige Programmiersprachen beschrieben, mit denen reaktive Systeme in einer mehr oder weniger übersichtlichen Weise beschrieben und implementiert werden können. Viele dieser Sprachen orientieren sich aber noch am herkömmlichen Stil imperativer Programmierung. Einen wesentlichen Fortschritt sehen wir in zustandsbasierten Ansätzen und besonders in den sogenannten *synchronen Sprachen* [BeBe91] [Halb93], bei denen Zusammenhänge zwischen Ein- und Ausgabeströmen durch Funktionen und Taktbedingungen beschrieben werden. Wir wollen jetzt noch etwas weiter gehen und streben eine Programmgenerierung auf der Grundlage einer stärker *deklarativen* Sprache an, bei der der Anwender — so die Zielvorstellung — nur noch das gewünschte Verhalten seines reaktiven Systems spezifiziert, ohne die Belegung der Ausgaben explizit angeben zu müssen.

In diesem Bericht beschreiben wir eine einfache, logische Programmierspra-

che zur Spezifikation zustandsendlicher, reaktiver Systeme, die als Grundlage für weitere Untersuchungen auf diesem Gebiet dienen soll. Dabei gehen wir von einer linearen, temporalen Aussagenlogik aus, für die wir eine PROLOG-ähnliche Klauselnotation angeben. Mit ihr können Sicherheitsbedingungen, die in einem reaktiven System gelten sollen, in einer rein deklarativen Form spezifiziert werden. Die von uns verwendete Notation ähnelt der beim logischen Programmieren verwendeten Syntax [Lloyd84]. Dadurch werden die Spezifikationen für resolutionsbasierte Verfahren zur Programmverifikation und -synthese zugänglich.

In diesem Bericht beschränken wir uns auf die formale Beschreibung der verwendeten Programmiersprache, ihre Semantik und ihre Ausdrucksmächtigkeit. In Abschnitt 2 definieren wir zunächst die Klasse der *temporal stratifizierten Programme* und ihre Semantik. Danach stellen wir diese Programme in Abschnitt 3 der Klasse der endlichen Automaten gegenüber und zeigen, daß beide Konzepte hinsichtlich ihrer Ausdrucksmächtigkeit äquivalent sind. Zum Abschluß folgt eine kurze Zusammenfassung und ein Ausblick auf zukünftige Forschungsaktivitäten.

2 F2-Programme

2.1 Syntax und Semantik von F2

Wir setzen im folgenden die Existenz eines endlichen Alphabets Σ von *Aussagenvariablen* voraus und definieren zunächst eine Sprache der *linearen temporalen Logik* über dieser Menge, die wir F2 nennen.

Definition 2.1 (Syntax von F2) Sei Σ ein endliches Alphabet von Aussagenvariablen. Wir definieren die Menge $F2_\Sigma$ der *F2-Formeln über* Σ induktiv wie folgt:

- Jedes Symbol $p \in \Sigma$ ist in $F2_\Sigma$.
- $F2_\Sigma$ enthält ein ausgezeichnetes Symbol START, das nicht in Σ auftritt.
- Zu $\alpha, \beta \in F2_\Sigma$ sind auch $\neg\alpha \in F2_\Sigma$ und $\alpha \wedge \beta \in F2_\Sigma$.
- Zu $\alpha \in F2_\Sigma$ ist auch $\circ\alpha \in F2_\Sigma$ („Next"-Operator).

Aus diesen Operatoren lassen sich in der üblichen Weise alle bekannten aussagenlogischen Operatoren als Abkürzung definieren. Außerdem schreiben wir $\circ^i$ als Abkürzung für eine Folge von i $\circ$-Operatoren.

Als nächstes betrachten wir die Semantik der eben eingeführten Sprache. Es handelt sich um eine lineare temporale Logik, d.h. eine Interpretation ist eine Folge von Zuständen, in denen den Aussagenvariablen aus Σ Wahrheitswerte zugeordnet werden.

Definition 2.2 Sei Σ ein endliches Alphabet von Aussagenvariablen. Ein *temporaler Zustand* über Σ ist eine Abbildung $I\colon \Sigma \to \{0,1\}$.

Definition 2.3 Eine *endliche Interpretation* über Σ der Länge $n \in \mathbb{N}_0$ ist eine endliche Folge $I = (I_1, \ldots, I_n)$ von temporalen Zuständen über Σ. Eine *unendliche Interpretation* über Σ ist eine unendliche Folge $I = (I_1, I_2, \ldots)$ von temporalen Zuständen über Σ.

Wir bezeichnen mit $\mathcal{I}(\Sigma)$ die Menge der Interpretationen über Σ. Als Länge einer endlichen Interpretation $I = (I_1, \ldots, I_n)$ definieren wir die natürliche Zahl n. Die Länge einer unendlichen Interpretation definieren wir formal als ∞, und meinen damit ein Objekt, das größer ist als jede natürliche Zahl.

Definition 2.4 Sei Σ ein endliches Alphabet und $I = (I_1, I_2, \ldots)$ eine endliche oder unendliche Interpretation über Σ. Jede Folge $\bar{I} = (\bar{I}_1, \bar{I}_2, \ldots)$ der gleichen Länge wie I von Abbildungen $\bar{I}_k\colon F2_\Sigma \to \{0,1\}$ heißt *Fortsetzung* von I auf $F2_\Sigma$, wenn sie die folgenden Bedingungen erfüllt:

- $\bar{I}_k(p) = I_k(p)$ für jedes k und jedes $p \in \Sigma$.
- $\bar{I}_k(\textsc{start}) = 1$ gdw. $k = 1$.
- $\bar{I}_k(\neg\alpha) = 1 - \bar{I}_k(\alpha)$.
- $\bar{I}_k(\alpha \wedge \beta) = 1$ gdw. $\bar{I}_k(\alpha) = 1$ und $\bar{I}_k(\beta) = 1$.
- $\bar{I}_k(\circ\alpha) = \bar{I}_{k+1}(\alpha)$, falls k echt kleiner ist als die Länge von I.

Bei einer endlichen Interpretation I wird über die Interpretation von Formeln der Gestalt $\circ\alpha$ im letzten Zustand von I nichts ausgesagt. Dies ist die einzige Unbestimmtheit in der Definition: Sie tritt in der letzten Bedingung auf, wenn eine Formel im Endzustand einer endlichen Interpretation betrachtet wird. Die Erweiterung einer unendlichen Interpretation auf $F2_\Sigma$ ist dagegen stets eindeutig definiert, so daß wir in diesem Fall auch von *der* Erweiterung $\bar{I}$ von I auf $F2_\Sigma$ sprechen.

Wir identifizieren im folgenden $\bar{I}$ mit I, wenn aus dem Zusammenhang ersichtlich ist, was gemeint ist. Außerdem stellen wir fest, daß ein Zustand I über Σ eindeutig durch eine Teilmenge $M_I \subseteq \Sigma$ beschrieben werden kann, nämlich die Menge der $p \in \Sigma$ mit $I(p) = 1$. Deshalb bezeichnen wir im folgenden auch Teilmengen $I \subseteq \Sigma$ als Zustände und Folgen von solchen Teilmengen als Interpretationen über Σ.

Ebenfalls gebräuchlich ist die Notation von Zuständen als *Literalmengen*. Dabei werden alle in einem Zustand zu wahr interpretierten *Literale*, d.h. Formeln der Gestalt p oder $\neg p$ für ein $p \in \Sigma$, aufgezählt. Beispielsweise wird mit $I = \{p, \neg q, r\}$ der Zustand I über $\Sigma = \{p, q, r\}$ beschrieben, der p und r mit 1 und q mit 0 belegt.

Definition 2.5 Sei $\alpha \in F2_\Sigma$ eine F2-Formel und I eine Interpretation über Σ. Wir nennen I ein *Modell* für α, wenn es eine Erweiterung $\bar{I}$ von I auf $F2_\Sigma$ gibt, so daß für jeden Zustand I_k von I gilt $\bar{I}_k(\alpha) = 1$. In diesem Fall schreiben wir auch $I \models \alpha$.

Definition 2.6 Sei $M \subseteq F2_\Sigma$ eine Menge von F2-Formeln und I eine Interpretation über Σ. I heißt Modell für M, $I \models M$, wenn I Modell für jede Formel $\alpha \in M$ ist.

Definition 2.7 Sei $M \subseteq F2_\Sigma$ eine Menge von F2-Formeln. M heißt

- *erfüllbar in der Länge* $n \in \mathbb{N}_0$, wenn es eine Interpretation über Σ der Länge n gibt, die Modell für M ist.
- *erfüllbar im Unendlichen*, wenn es eine unendliche Interpretation über Σ gibt, die Modell für M ist.
- *gültig in der Länge* $n \in \mathbb{N}_0$, wenn jede Interpretation über Σ der Länge n ein Modell für M ist.
- *gültig im Unendlichen*, wenn jede unendliche Interpretation über Σ für M ist.

Eine wichtige Konsequenz dieser Erfüllbarkeitsdefinition besteht darin, daß sich die Existenz von längeren Modellen auf die Existenz von kürzeren Modellen überträgt. Dies gilt jedoch nur für die hier definierte einfache Logik F2.

Lemma 2.1 Sei $M \subseteq F2_\Sigma$ eine Menge von F2-Formeln. Wenn M erfüllbar ist in der Länge $n \in \mathbb{N}_0$ (bzw. in $n = \infty$), dann ist M auch erfüllbar in der Länge m für jedes $m \leq n$.

Beweis. Sei $I = (I_1, \ldots, I_n)$ das nach Voraussetzung existierende Modell für M. Dann gibt es eine Erweiterung $\bar{I} = (\bar{I}_1, \ldots, \bar{I}_n)$ von I auf $F2_\Sigma$. Wir betrachten die Interpretation $I' = (I_1, \ldots, I_m)$, die einem Anfangsstück von I der Länge $m \leq n$ entspricht, und behaupten, daß diese ebenfalls ein Modell für M ist. Dazu müssen wir lediglich zeigen, daß $\bar{I}' = (\bar{I}_1, \ldots, \bar{I}_m)$ eine Erweiterung von I' auf $F2_\Sigma$ ist. Also überprüfen wir die in Definition 2.4 aufgestellten Bedingungen:

- Sei $p \in \Sigma$. Dann folgt sofort $\bar{I}'_k(p) = \bar{I}_k(p) = I_k(p) = I'_k(p)$ für jedes $k = 1, \ldots, m$.
- $\bar{I}'_k(\text{START}) = \bar{I}_k(\text{START}) = 1$ gdw. $k = 1$.
- $\bar{I}'_k(\neg\alpha) = \bar{I}_k(\neg\alpha) = 1 - \bar{I}_k(\alpha) = 1 - \bar{I}'_k(\alpha)$.
- $\bar{I}'_k(\alpha \wedge \beta) = \bar{I}_k(\alpha \wedge \beta) = 1$ gdw. $1 = \bar{I}_k(\alpha) = \bar{I}'_k(\alpha)$ und $1 = \bar{I}_k(\beta) = \bar{I}'_k(\beta)$.

▷ Für $k < m$ ist wegen $k < n$ auch $I'_k(\circ\alpha) = I_k(\circ\alpha) = \bar{I}_{k+1}(\alpha) = \bar{I}'_{k+1}(\alpha)$.

Damit sind alle Bedingungen aus Definition 2.4 erfüllt, und es folgt die Behauptung. □

Definition 2.8 Seien $M_1, M_2 \subseteq F2_\Sigma$ zwei Mengen von F2-Formeln. M_2 heißt *logische Konsequenz* von M_1, $M_1 \models M_2$, wenn jedes Modell für M_1 auch Modell für M_2 ist. M_1 und M_2 heißen *logisch äquivalent*, wenn $M_1 \models M_2$ und $M_2 \models M_1$ gilt.

Da wir später auch Interpretationen betrachten müssen, die nur auf Teilmengen von Σ definiert sind, benötigen wir noch die folgende Definition.

Definition 2.9 Sei Σ ein endliches Alphabet, und sei $\Sigma' \subseteq \Sigma$. Ein Zustand I_k über Σ heißt *Erweiterung* eines Zustandes I'_k über Σ' auf Σ, $I_k \geq I'_k$, wenn $I_k(p) = I'_k(p)$ für alle $p \in \Sigma'$. Eine endliche oder unendliche Interpretation $I = (I_1, I_2, \ldots)$ über Σ heißt Erweiterung einer gleichlangen Interpretation $I' = (I'_1, I'_2, \ldots)$ über Σ' auf Σ, $I \geq I'$, wenn für jedem Zustand I_k von I gilt $I_k \geq I'_k$.

In diesem Zusammenhang verwenden wir auch den Begriff der *Einschränkung* $I|_{\Sigma'}$ eines Zustandes oder einer Interpretation I über Σ auf eine Teilmenge $\Sigma' \subseteq \Sigma$.

F2-Formeln, die keine $\circ$-Operatoren enthalten, bezeichnen wir als *$\circ$-frei*. Diese Formeln entsprechen im wesentlichen aussagenlogischen Formeln, die zusätzlich das Prädikatensymbol START enthalten können. Da wir derartige Eigenschaften später noch häufiger benötigen werden, definieren wir eine entsprechende Redeweise.

Definition 2.10 Sei α eine F2-Formel über Σ. α heißt *$\circ$-frei*, wenn α keinen $\circ$-Operator enthält. α heißt *temporal flach*, wenn in α außerdem das Symbol START nicht auftritt.

Lemma 2.2 Sei α eine $\circ$-freie F2-Formel über Σ. Genau dann ist α gültig im Unendlichen, wenn α gültig ist als aussagenlogische Formel über $\Sigma \cup \{\text{START}\}$.

Beweis. Sei zunächst α gültig im Unendlichen. Sei I eine beliebige aussagenlogische Interpretation über $\Sigma \cup \{\text{START}\}$. Damit bilden wir die unendliche Interpretation $I' = (I|_\Sigma, I|_\Sigma, \ldots)$ über Σ. Nach Voraussetzung ist I' ein Modell für α. Da I' eine unendliche Interpretation ist, gibt es nur eine Fortsetzung $\bar{I}'$ von I' auf $F2_\Sigma$. Im Falle $I(\text{START}) = 1$ folgt für jede Fortsetzung[1] $\bar{I}$ von I auf die Menge der $\circ$-freien F2-Formeln $\bar{I}(\alpha) = \bar{I}'_1(\alpha) = 1$. Anderfalls ist $I(\text{START}) = 0$, und es folgt stets $\bar{I}(\alpha) = \bar{I}'_2(\alpha) = 1$, da α ja keinen $\circ$-Operator enthält.

[1] Wir verwenden hier den Begriff der Fortsetzung einer aussagenlogischen Interpretation über $\Sigma \cup \{\text{START}\}$ auf $F2_\Sigma$, den wir nicht definiert haben. Dieser kann über $\circ$-freien Formeln — und nur dort — analog zu Definition 2.4 erklärt werden.

Nun sei umgekehrt α gültig als aussagenlogische Formel über $\Sigma \cup \{\text{START}\}$. Sei $I = (I_1, I_2, \ldots)$ eine beliebige unendliche Interpretation über Σ. Dann folgt aus der Voraussetzung

$$\begin{array}{lll} I_1 \cup \{\text{START}\} & \models_{\text{AL}} & \alpha, \\ I_j & \models_{\text{AL}} & \alpha, \quad \text{für alle } j \geq 2, \end{array}$$

wobei die linke Seite jeweils als aussagenlogische Interpretation über $\Sigma \cup \{\text{START}\}$ aufzufassen ist. Damit folgt $I \models \alpha$. □

Korollar 2.3 Sei α eine temporal flache F2-Formel über Σ. Genau dann ist α gültig im Unendlichen, wenn α gültig ist als aussagenlogische Formel über Σ.

Beweis. Da in α das Symbol START nicht auftritt, ist die Gültigkeit von α als aussagenlogische Formel über Σ äquivalent mit der Gültigkeit über $\Sigma \cup \{\text{START}\}$. Alles weitere folgt aus Lemma 2.2. □

Temporal flache F2-Formeln können also über Zuständen interpretiert werden, da in diesem Fall der Wahrheitswert in einem Zustand nicht von den anderen Zuständen einer Interpretation abhängt. Diese Beobachtung rechtfertigt die folgende Definition ebenso wie entsprechende Definitionen für Gültigkeit und Erfüllbarkeit.

Definition 2.11 Sei α eine temporal Flache F2-Formel über Σ, und $I \subseteq \Sigma$ ein Zustand über Σ. I heißt Modell für α, wenn $I(\alpha) = 1$.

2.2 Programme

In diesem Abschnitt definieren wir eine Klasse von zukunftsgerichteten temporallogischen Programmen in Analogie zu bekannten Logik-Programmen. Diese Programme bezeichnen wir als F2-Programme. Wir werden viele vom logischen Programmieren bekannte Begriffe an unseren Hintergrund anpassen.

Definition 2.12 Sei Σ ein endliches Alphabet. Ein *aussagenlogisches Literal* ist eine F2-Formel der Gestalt p oder $\neg p$ für ein $p \in \Sigma$. Ein *$\circ$-temporales Literal* ist eine F2-Formel der Gestalt $\circ^i L$ für ein aussagenlogisches Literal L. Ein *$\circ$-freies Literal* ist ein aussagenlogisches Literal oder eine F2-Formel der Gestalt START oder $\neg$ START. Ein *(allgemeines) temporales Literal* ist eine F2-Formel der Gestalt $\circ^i L$ für ein $\circ$-freies Literal L.

Definition 2.13 Sei Σ ein endliches Alphabet. Eine *F2-Klausel* ist eine F2-Formel der Gestalt $L \leftarrow L_1 \wedge \ldots \wedge L_n$ für ein $\circ$-temporales Literal L und $\circ$-freie Literale $L_1, \ldots, L_n$. Dabei nennen wir L den *Kopf* und $L_1 \wedge \ldots \wedge L_n$ den *Rumpf* der Klausel. Ein *F2-Ziel* ist eine F2-Formel der Gestalt $\leftarrow L_1 \wedge \ldots \wedge L_n$ für $\circ$-freie temporale Literale $L_1, \ldots, L_n$.

Definition 2.14 Sei Σ ein endliches Alphabet. Ein *F2-Programm* über Σ ist eine endliche Menge von F2-Klauseln über Σ.

Bei unserer Definition einer Klausel haben wir negierte Literale im Klauselkopf zugelassen. Dadurch entsteht bei der Zusammenfassung von Klauselmengen zu Programmen die Möglichkeit, Widersprüche zu erzeugen, etwa durch Aufnahme zweier Klauseln $p \leftarrow \alpha$ und $\neg p \leftarrow \beta$, wobei $\alpha \wedge \beta$ erfüllbar ist. Um so etwas zu verhindern, formulieren wir später zusätzliche Bedingungen.

Die Einschränkung der Klauselrümpfe auf Konjunktionen von Literalen kann ähnlich wie beim logischen Programmieren aufgeweicht werden. Dort kann man beliebige aussagenlogische Formeln als Klauselrümpfe und Ziele zulassen, und die so entstehenden Programme werden durch den *Normalisierungsalgorithmus* in Normalformen ähnlich der hier betrachteten Form transformiert [Lloyd84]. Eine Übertragung dieses Verfahren auf unsere F2-Programme ist möglich, was zumindest für die übrigen aussagenlogischen Operatoren leicht vorstellbar ist. Es ist sogar möglich, auf diese Weise komplexere temporale Operatoren einzuführen, wie z.B. SINCE. Obwohl wir diesen Aspekt in der vorliegenden Arbeit nicht weiter formalisieren, werden wir in einigen Beispielen um der besseren Lesbarkeit willen bereits komplexere Klauselrümpfe verwenden.

Zunächst aber wollen wir F2-Programmen unabhängig von der Existenz von Modellen eine Bedeutung geben. Wir verfolgen das Ziel, mit unseren F2-Programmen reaktive Systeme zu beschreiben. Das heißt, wir müssen durch die logischen Formeln eines F2-Programms bestimmte Folgen von Ein/Ausgabekombinationen an ein reaktives System als korrekt und andere als inkorrekt klassifizieren. Dazu werden wir zunächst die Klauseln eines Programms genauer strukturieren und dann einige der Aussagenvariablen aus Σ als Ein- bzw. Ausgaben auszeichnen.

Definition 2.15 Sei P ein F2-Programm über Σ und $p \in \Sigma$. Dann definieren wir die *positive Definition* i-ter Stufe von p, $\text{DEF}_i^+(p)$, als die Menge der Klauseln in P, in deren Kopf das $\circ$-temporale Literal $\circ^i p$ auftritt. Analog wird die *negative Definition* i-ter Stufe von p erklärt:

$$\begin{aligned} \text{DEF}_+^i(p) &= \{ C \in P \mid C \equiv \circ^i p \leftarrow \alpha \}; \\ \text{DEF}_-^i(p) &= \{ C \in P \mid C \equiv \circ^i \neg p \leftarrow \alpha \}. \end{aligned}$$

Die *Definition* von p ist die Vereinigung aller dieser Mengen:

$$\text{DEF}(p) = \bigcup_{i=0}^{\infty} (\text{DEF}_+^i(p) \cup \text{DEF}_-^i(p)).$$

Die Definition einer Menge von Prädikatensymbolen ist die Vereinigung der Definitionen aller enthaltenen Prädikatensymbole.

Definition 2.16 Sei Σ ein endliches Alphabet, und sei $\mathcal{A} \subseteq \Sigma$. Wir nennen P ein F2-Programm mit Ein/Ausgaben $\mathcal{A}$, wenn P ein F2-Programm über Σ ist mit $\textsc{Def}(\mathcal{A}) = \emptyset$.

Die Semantik von $P \cup \{\leftarrow \alpha\}$ besteht aus einer Menge von Ein/Ausgabefolgen, d.h. Interpretationen über $\mathcal{A}$, die sich zu Modellen für P und α erweitern lassen:

Definition 2.17 Sei P ein F2-Programm über Σ mit Ein/Ausgaben $\mathcal{A} \subseteq \Sigma$ und $\leftarrow \alpha$ ein F2-Ziel. Eine Interpretation $I^{\mathcal{A}}$ über $\mathcal{A}$ heißt

- ▷ *mögliche Ein/Ausgabefolge* für $P \cup \{\leftarrow \alpha\}$, wenn es eine Erweiterung I von $I^{\mathcal{A}}$ auf Σ gibt, die Modell für P und für α ist.
- ▷ *gültige Ein/Ausgabefolge* für $P \cup \{\leftarrow \alpha\}$, wenn jede Erweiterung I von $I^{\mathcal{A}}$ auf Σ, die Modell für P ist, auch Modell für α ist.

Wir bezeichnen die im folgenden die Menge der möglichen Ein/Ausgabefolgen für $P \cup \{\leftarrow \alpha\}$ mit $\mathcal{L}_m(P \cup \{\leftarrow \alpha\})$ und die Menge der gültigen Ein/Ausgabefolgen mit $\mathcal{L}_g(P \cup \{\leftarrow \alpha\})$. Im allgemeinen sind diese beiden Mengen nicht gleich, und es gilt

$$\mathcal{L}_g(P \cup \{\leftarrow \alpha\}) \subseteq \mathcal{L}_m(P \cup \{\leftarrow \alpha\}).$$

Der Unterschied zwischen den beiden Definitionen hängt eng mit der Eigenschaft des *Determinismus* bei endlichen Automaten zusammen. Eine mögliche Ein/Ausgabefolge liegt dann vor, wenn es eine Abarbeitungsfolge gibt, die diese Ein/Ausgaben liefert. Bei einer gültigen Ein/Ausgabefolge wird zusätzlich gefordert, daß dies für jede Abarbeitungsfolge gilt.

Beispiel 2.1 Gegeben sei das folgende F2-Programm P über $\Sigma = \{a, b, c, Z^*\}$ mit Ein/Ausgaben $\mathcal{A} = \{a, b, c\}$ und das zugehörige F2-Ziel $\leftarrow \alpha$:

$$\begin{array}{lrcl} P: & \circ Z^* & \leftarrow & \neg Z^* \wedge a \wedge \neg b \\ & \circ \neg Z^* & \leftarrow & \neg Z^* \wedge \neg a \wedge b \\ & \circ \neg Z^* & \leftarrow & \neg Z^* \wedge c \\ \leftarrow \alpha: & & \leftarrow & Z^* \rightarrow c \end{array}$$

Dazu betrachten wir die Ein/Ausgabefolge $I^{\mathcal{A}} = (\{a, b\}, \{a, b\}, \ldots, \{a, b\})$, d.h. von den Ein/Ausgaben werden immer nur a und b als wahr interpretiert. Dann ist $I^{\mathcal{A}}$ eine mögliche Ein/Ausgabefolge für $P \cup \{\leftarrow \alpha\}$, denn die Erweiterung I von $I^{\mathcal{A}}$ auf Σ, die Z^* stets als falsch interpretiert, ist ein Modell für P und für α. Andererseits ist $I^{\mathcal{A}}$ keine gültige Ein/Ausgabefolge, denn die Interpretation $I' = (\{a, b, Z^*\}, \ldots, \{a, b, Z^*\})$, die Z^* stets als wahr interpretiert und ebenfalls eine Erweiterung von $I^{\mathcal{A}}$ auf Σ ist, ist zwar ein Modell für P, verletzt aber die Bedingung α.

2.3 Temporal stratifizierte Programme

Wir interessieren uns für Möglichkeiten zur automatischen Synthese eines Steuerungsprogrammes, das die Erfüllung der durch ein F2-Programm und -Ziel gestellten Anforderungen sicherstellt. Ein solches Steuerungsprogramm wir im allgemeinen einige *Hilfsvariable* benötigen, in denen es Informationen über vergangene Zustände speichert. In jedem Zustand wird es aus diesen Variablen und den aktuellen Eingaben die benötigten Ausgaben und einen Folgezustand berechnen, der wieder in den Hilfsvariablen gespeichert wird.

Nun wäre es wünschenswert, wenn diese Hilfsvariablen einem Teil der in dem gegebenen Programm auftretenden Prädikatensymbolen entsprechen würden. Es zeigt sich jedoch, daß dies nicht immer der Fall ist — oft ist ein großer Teil der in diesen Prädikatensymbolen gespeicherten Information redundant, während andere, wichtige Informationen nicht aus ihnen entnommen werden können. Ein weiteres Problem sind *rekursive* Definitionen von Prädikatensymbolen, weil sie die Auswertung in einem Schritt stark erschweren.

Deshalb erstreben wir eine besondere Gestalt von Programmen, aus der die benötigten Hilfsvariablen und die Berechnungsvorschriften für ihre Werte direkt abgelesen werden können. Diese finden wir in der Normalform der sogenannten *temporal stratifizierten Programme*, die wir im folgenden einführen werden.

Zunächst aber definieren wir die Klasse der *temporal hierarchischen Programme*. Bei ihnen sind Rekursionen zwar prinzipiell zugelassen, aber es muß stets mindestens ein $\circ$-Operator auftreten, bevor bei der Abarbeitung eines rekursiv definierten Prädikatensymbols p wieder ein rekursiver Aufruf von p erfolgen kann. Dazu benötigen wir eine formale Beschreibung der Abhängigkeitsbeziehungen zwischen den Prädikatensymbolen in einem F2-Programm.

Definition 2.18 (Prädikat-Abhängigkeiten) Sei Σ ein endliches Alphabet, P ein F2-Programm über Σ, und seien $p, q \in \Sigma$. Wir schreiben

- $p \sqsubset^i q$ für ein $i \in \mathbb{N}_0$, wenn es in P eine Klausel C der Gestalt

$$C \equiv \circ^i p \leftarrow L_1 \wedge \ldots \wedge L_n$$

  gibt, wobei auf der rechten Seite ein Literal der Gestalt $L_k \equiv q$ oder $L_k \equiv \neg q$ auftritt.

- $p <^i q$ für ein $i \in \mathbb{N}_0$, wenn es ein $n > 0$ sowie $p_0, \ldots, p_n \in \Sigma$ und $i_1, \ldots, i_n \in \mathbb{N}_0$ gibt, so daß $p = p_0 \sqsubset^{i_1} p_1 \sqsubset^{i_2} \cdots \sqsubset^{i_n} p_n = q$ gilt, wobei $i = i_1 + \cdots + i_n$.

- $p < q$, wenn $p <^i q$ für ein $i \in \mathbb{N}_0$ gilt.

Es ist leicht einzusehen, daß die Relation $<$ transitiv ist. Wir interessieren uns für die sogenannten *hierarchischen* Programme, bei denen $<$ zusätzlich irreflexiv ist, denn dann treten in dem Programm keine rekursiven Definitionen von

Prädikatensymbolen auf. In unserem temporalen Kontext ist die Relation $<^0$ ebenfalls von großem Interesse. Sie gibt an, ob zwischen den Belegungen von zwei Prädikatensymbolen im gleichen Zustand eine Abhängigkeit bestehen kann. In dem wir fordern, daß $<^0$ irreflexiv ist, stellen wir sicher, daß zwischen allen rekursiven Abhängigkeiten mindestens ein $\circ$-Operator liegt.

Definition 2.19 Sei P ein F2-Programm über Σ. P heißt *hierarchisch*, falls es kein Prädikatensymbol $p \in \Sigma$ gibt mit $p < p$. P heißt *temporal hierarchisch*, falls es kein Prädikatensymbol $p \in \Sigma$ gibt mit $p <^0 p$.

Der folgende Satz enthält einige Eigenschaften temporal hierarchischer Programme, die sich als sehr nützliche Kriterien zum Nachweis der temporalen Hierarchie erweisen. Besonders interessant ist die Eigenschaft (H4), die besagt, daß es möglich ist, alle Prädikatensymbole eines temporal hierarchischen Programms in einer linearen Kette $p_1 \sqsubset^0 p_2 \sqsubset^0 \cdots \sqsubset^0 p_n$ anzuordnen. Eine analoge Aussage gilt übrigens auch bei hierarchischen Programmen mit der Ordnung $<$, aber wir zeigen hier nur den Beweis für den temporal hierarchischen Fall. In dem interessanten Fall, daß das Programm keine $\circ$-Operatoren enthält (*temporal flache* Programme, siehe unten), sind beide Begriffe übrigens äquivalent.

Satz 2.4 (Eigenschaften temporal hierarchischer Programme) Sei Σ ein endliches Alphabet, und sei P ein F2-Programm über Σ. Dann sind folgende Aussagen äquivalent:

(H1) P ist temporal hierarchisch.

(H2) Es gibt eine *Niveauabbildung* $N: \Sigma \to \mathbb{N}_0$, so daß für alle $p, q \in \Sigma$ mit $p \sqsubset^0 q$ gilt $N(p) > N(q)$.

(H3) Es gibt eine Niveauabbildung $N: \Sigma \to \mathbb{N}_0$, so daß für alle $p, q \in \Sigma$ mit $p <^0 q$ gilt $N(p) > N(q)$.

(H4) Es gibt eine *injektive* Niveauabbildung $N: \Sigma \to \mathbb{N}_0$, so daß für alle $p, q \in \Sigma$ mit $p <^0 q$ gilt $N(p) > N(q)$.

Beweis. Wir zeigen die Äquivalenz der vier Aussagen durch einen Ringschluß.

(H1) $\Longrightarrow$ (H2). Sei P ein temporal hierarchisches F2-Programm über Σ. Zu jedem Prädikatensymbol $p \in \Sigma$ erklären wir die Menge $\mathcal{N}(p)$ der Vorgängerprädikate durch

$$\mathcal{N}(p) = \{ q \in \Sigma \mid p <^0 q \}.$$

$\mathcal{N}(p)$ ist endlich als Teilmenge des endlichen Alphabets Σ, und damit ist folgende Funktion

$$N: \Sigma \to \mathbb{N}_0: p \mapsto |\mathcal{N}(p)|$$

eine wohldefinierte Niveauabbildung. Dabei bezeichnen wir mit $|M|$ die Anzahl der Elemente der endlichen Menge M. Sei nun $p \sqsubset^0 q$. Wir zeigen $N(p) > N(q)$.

Zunächst folgt aufgrund der Transitivität von $<^0$ für beliebiges $r \in \Sigma$ aus $q <^0 r$ stets auch $p <^0 r$. Somit ist $\mathcal{N}(p) \supseteq \mathcal{N}(q)$. Andererseits ist wegen $p <^0 q$ auch $q \in \mathcal{N}(p)$, aber da P hierarchisch ist, gilt $q \notin \mathcal{N}(q)$. Damit haben wir $\mathcal{N}(p) \neq \mathcal{N}(q)$. Es folgt $N(p) = |\mathcal{N}(p)| > |\mathcal{N}(q)| = N(q)$.

(H2) $\Longrightarrow$ *(H3)*. Sei $N: \Sigma \to \mathbb{N}_0$ eine Niveauabbildung, so daß für alle $p, q \in \Sigma$ mit $p \sqsubset^0 q$ gilt $N(p) > N(q)$. Wir zeigen, daß dann bereits aus $p <^0 q$ die Aussage $N(p) > N(q)$ folgt. Seien also $p, q \in \Sigma$ mit $p <^0 q$. Dann gibt es ein $n > 0$ sowie $p_0, \ldots, p_n \in \Sigma$, so daß $p = p_0 \sqsubset^0 p_1 \sqsubset^0 \cdots \sqsubset^0 p_n = q$ gilt. Wir zeigen die Behauptung durch vollständige Induktion über k. Im Induktionsanfang für $k = 1$ haben wir $p = p_0 \sqsubset^0 p_1 = q$, d.h. es ist $p \sqsubset^0 q$, und die Behauptung folgt direkt aus der Voraussetzung. Nun nehmen wir als Induktionsannahme an, die Bahauptung sei für ein $k > 0$ bereits beweisen, und betrachten den Fall $p = p_0 \sqsubset^0 \cdots \sqsubset^0 p_k \sqsubset^0 p_{k+1} = q$. Nach Induktionsannahme ist dann $N(p_0) > N(p_k)$, und wegen $p_k \sqsubset^0 p_{k+1}$ folgt aus der Voraussetzung $N(p_k) > N(p_{k+1})$. Damit haben wir $N(p) = N(p_0) > N(p_k) > N(p_{k+1}) = N(q)$.

(H3) $\Longrightarrow$ *(H4)*. Sei eine beliebige, nicht notwendig injektive Niveauabbildung $N: \Sigma \to \mathbb{N}_0$ gegeben. Dann erklären wir die *Niveaus* zu N durch

$$\mathcal{N}_k = \{\, p \in \Sigma \mid N(p) = k \,\}.$$

Im folgenden nehmen wir an, $\mathcal{N}_k$ habe die Gestalt

$$\mathcal{N}_k = \{p_k^0, \ldots, p_k^{j_k}\}.$$

Weiter bezeichnen wir mit

$$n_k = |\bigcup_{i=0}^{k-1} \mathcal{N}_i|$$

die Anzahl der Prädikatensymbole in Niveaus unterhalb von $\mathcal{N}_k$. Offenbar gehört jedes $p \in \Sigma$ einem Niveau an, ist also ein $p_k^j \in \mathcal{N}_k$. Damit definieren wir die folgende Niveauabbildung:

$$N': \Sigma \to \mathbb{N}_0;\ p_k^j \mapsto n_k + j.$$

Seien $p, q \in \Sigma$ mit $p <^0 q$. Dann ist nach Voraussetzung $N(p) > N(q)$. Wir zeigen zunächst, daß dann auch $N'(p) > N'(q)$ folgt. Nun bedeutet $N(p) > N(q)$ stets $p = p_k^i$ und $q = p_l^j$ für natürliche Zahlen $k > l$. Außerdem ist hier $j < |\mathcal{N}_l| = n_{l+1} - n_l$, und folglich gilt

$$N'(p) = n_k + i \geq n_k \geq n_{l+1} > n_l + j = N'(q).$$

Um noch zu zeigen, daß N' injektiv ist, seien $p, q \in \Sigma$ mit $N'(p) = N'(q)$. Dann ist $N(p) = N(q)$, denn wäre $N(p) \neq N(q)$, etwa $N(p) > N(q)$, dann würde wie oben folgen $N'(p) > N'(q)$. Also ist $p = p_k^i$ und $q = p_k^j$ für das gleiche k. Dann aber folgt aus der Definition von N' auch $i = j$ und somit $p = q$, d.h. N' ist injektiv.

(II4) $\Longrightarrow$ *(II1)*. Sei $N: \Sigma \to \mathbb{N}_0$ eine injektive Niveauabbildung. Wäre P nicht hierarchisch, so gäbe es ein $p \in \Sigma$ mit $p <^0 p$. Für dieses p würde gelten $N(p) > N(p)$, was aber nicht möglich ist. □

Leider zeigt sich, daß die Eigenschaft der temporalen Hierarchie nicht ausreicht, um die relevanten rekursiv definierten Prädikatensymbole zu isolieren. Wir benötigen eine weitere Verschärfung, bei der die betroffenen Prädikatensymbole exakt ausgegrenzt werden können. Dies führt uns zur Klasse der *temporal stratifizierten Programme*.

Die folgenden Definitionen sollen sichern, daß alle mehrfachen Auftreten von $\circ$-Operatoren in einer Klausel entfernt oder durch zusätzliche Hilfsprädikate ersetzt wurden, so daß im Klauselrumpf entweder nur noch Variable aus dem Gegenwartszustand oder nur noch Variable aus dem unmittelbar vorausgegangenen Zustand auftreten. Dann können die Klauseln mit der zustandsübergreifenden Information getrennt von den übrigen behandelt werden.

Definition 2.20 Sei C eine F2-Klausel. C heißt *temporal flach*, wenn C keine temporalen Operatoren enthält, d.h. wenn sie die Gestalt

$$C \equiv L \leftarrow L_1 \wedge \ldots \wedge L_n$$

für aussagenlogische Literale $L, L_1, \ldots, L_n$ hat. C heißt *temporal abgestuft* oder $\circ$-*abgestuft*, wenn C eine der beiden Gestalten

$$C \equiv L \leftarrow \text{START}$$
$$C \equiv \circ L \leftarrow L_1 \wedge \ldots \wedge L_n$$

für aussagenlogische Literale $L, L_1, \ldots, L_n$ hat. Wir nennen eine Klauselmenge temporal flach bzw. $\circ$-abgestuft, wenn jedes Element diese Eigenschaft besitzt.

Definition 2.21 Ein F2-Ziel $\leftarrow \alpha$ heißt temporal flach, wenn es die Gestalt

$$\leftarrow \alpha \equiv \leftarrow L_1 \wedge \ldots \wedge L_n$$

hat für aussagenlogische Literale $L_1, \ldots, L_n$.

Im folgenden verwenden wir eine feinere Strukturierung der Menge der Aussagenvariablen. Wir gehen davon aus, daß deren Grundmenge Σ drei Typen von Prädikatensymbolen enthält, nämlich die *Zustandsprädikate* in $\mathcal{E}^\star$, die *Ein/Ausgabe-Prädikatensymbole* in $\mathcal{A}$ und die *Hilfsprädikate* in $\mathcal{P}$. Wir nennen die disjunkte Vereinigung dieser drei Mengen von Prädikatensymbolen eine *Signatur* und schreiben dafür auch $\Sigma = (\mathcal{E}^\star, \mathcal{A}, \mathcal{P})$.

Definition 2.22 Sei $\Sigma = (\mathcal{E}^\star, \mathcal{A}, \mathcal{P})$ eine Signatur. Ein *temporal stratifiziertes F2-Programm* über Σ ist ein F2-Programm über Σ, das den folgenden drei Bedingungen genügt:

- ▷ $\text{DEF}(\mathcal{E}^\star)$ ist $\circ$-abgestuft.
- ▷ $\text{DEF}(\mathcal{A}) = \emptyset$.
- ▷ $\text{DEF}(\mathcal{P})$ ist temporal flach und hierarchisch.

Wie wir leicht sehen, ist jedes temporal stratifizierte F2-Programm auch temporal hierararchisch. Die Umkehrung dieser Aussage gilt nicht, aber es gibt ähnlich wie bei der Normalisierung logischer Programme einen Algorithmus, der bei temporal hierarchischen Programmen die temporale Stratifizierung herstellt.

Bei einem temporal stratifizierten Programm über $\Sigma = (\mathcal{E}^\star, \mathcal{A}, \mathcal{P})$ können wir die Prädikatensymbole in $\mathcal{E}^\star$ als Kodierung des Zustandes eines endlichen Automaten auffassen, eine Eigenschaft, die wir später noch genauer untersuchen werden.

Eine weitere wichtige Eigenschaft von temporal stratifizierten F2-Programmen besteht darin, daß bei Modellen die Werte der verwendeten Prädikatensymbole in einem beliebigen Zustand stets aus den Werten der Eingaben in $\mathcal{A}$ und den Werten der Zustands-Prädikatensymbole in $\mathcal{E}^\star$ im vorausgegangenen Zustand berechnet werden können.

2.4 Modelle temporal stratifizierter Programme

Bei temporal stratifizierten Programmen ist es möglich, eine sinnvolle Definition von *Zulässigkeit* einzuführen, die die Erfüllbarkeit impliziert. Außerdem gibt es eine einfache syntaktische Eigenschaft, die *Determinismus*, d.h. eine Eindeutigkeit der Modelle, zur Folge hat. Diese Kriterien führen wir durch die folgenden Definitionen ein und zeigen später die Zusammenhänge mit der Erfüllbarkeit auf.

Definition 2.23 Sei P ein F2-Programm über Σ, und sei $p \in \Sigma$. Dann definieren wir für jedes $i \in \mathbb{N}_0$ und jedes $\pi \in \{+, -\}$

$$\text{BODIES}^i_\pi(p) = \bigvee \{ W \mid L \leftarrow W \in \text{DEF}^i_\pi(p) \}.$$

Die Formel $\text{BODIES}^i_\pi(p)$ beschreibt also die Disjunktion der Rümpfe aller Klauseln in $\text{DEF}^i_\pi(p)$.

Um widerspruchsvolle Programme auszuschließen, definieren wir die Klasse der zulässigen Programme, welche sich dadurch auszeichnet, daß $\text{BODIES}^i_+(p)$ und $\text{BODIES}^i_-(p)$ zusammen nie erfüllbar sind.

Definition 2.24 Sei $\Sigma = (\mathcal{E}^\star, \mathcal{A}, \mathcal{P})$ eine Signatur. Ein temporal stratifiziertes F2-Programm P über Σ heißt *zulässig*, wenn für jedes Prädikatensymbol $p \in \Sigma$ und jedes $i \in \{0, 1\}$ der Ausdruck

$$\neg\, \text{BODIES}^i_+(p) \vee \neg\, \text{BODIES}^i_-(p)$$

im Unendlichen gültig ist. P heißt *deterministisch*, wenn zusätzlich die Ausdrücke

$$\begin{array}{ll} \textsc{Bodies}^0_+(p) \vee \textsc{Bodies}^0_-(p) & \text{für } p \in \mathcal{P}, \\ \textsc{Bodies}^1_+(p) \vee \textsc{Bodies}^1_-(p) & \text{für } p \in \mathcal{E}^* \end{array}$$

im Unendlichen gültig sind, und außerdem für jedes $p \in \mathcal{E}^*$ die Definition 0-ter Stufe nicht leer ist, d.h. $\textsc{Def}^0(p) \neq \emptyset$.

Wir müssen die Erfüllbarkeit bzw. Gültigkeit im Unendlichen betrachten, um auch den Fall, daß in einem Klauselrumpf das Symbol START auftritt, abdecken zu können. Da andererseits in den Klauselrümpfen von $\textsc{Def}(\mathcal{P})$ und $\textsc{Def}^1(\mathcal{E}^*)$ weder START- noch $\circ$-Operatoren zugelassen sind, können die obigen Kriterien nach Korollar 2.3 auf aussagenlogische Erfüllbarkeit zurückgeführt werden. Weiterhin erlaubt die extrem einfache Form von $\textsc{Def}^0(\mathcal{E}^*)$ eine direkte Überprüfung des Determinismus. Auf diese Weise erhalten wir ein effektives Entscheidungsverfahren, um F2-Programme auf ihre Zulässigkeit und Determiniertheit zu testen.

Eine andere Möglichkeit besteht darin, die beiden genannten Eigenschaften zu erzwingen, in dem wir uns auf eine Teilklasse von Programmen beschränken, die stets zulässig und deterministisch sind. Eine solche Klasse erhalten wir zum Beispiel, wenn wir für jedes Prädikatensymbol p nur eine Klausel der Gestalt $p \leftrightarrow \alpha$ oder $\circ p \leftrightarrow \alpha$ erlauben, wobei diese Formeln implizit wie folgt interpretiert werden:

$$p \leftrightarrow \alpha \quad : \quad \begin{cases} p & \leftarrow \alpha; \\ \neg p & \leftarrow \neg\alpha. \end{cases}$$

$$\circ p \leftrightarrow \alpha \quad : \quad \begin{cases} \neg p & \leftarrow \textsc{Start}; \\ \circ p & \leftarrow \alpha; \\ \circ \neg p & \leftarrow \neg\alpha. \end{cases}$$

Dies entspricht in seiner Semantik ungefähr der *Negation durch Scheitern*.

Wir beweisen im folgenden Aussagen darüber, daß wir für jedes zulässige temporal stratifizierte Programm ein Modell konstruieren können, und daß dieses Modell für deterministische temporal stratifizierte Programme immer eindeutig ist. Wir beginnen aber zunächst mit einigen Hilfsaussagen über die Existenz und Eindeutigkeit geeigneter Zustände.

Lemma 2.5 Sei P ein zulässiges, temporal stratifiziertes F2-Programm über $\Sigma = (\mathcal{E}^*, \mathcal{A}, \mathcal{P})$, und sei $I^{\mathcal{E}}$ ein Zustand über $\mathcal{E}^* \cup \mathcal{A}$. Dann gibt es einen Zustand $I \geq I^{\mathcal{E}}$ über Σ, der Modell ist für $\textsc{Def}(\mathcal{P})$.

Beweis. Da $\textsc{Def}(\mathcal{P})$ hierarchisch ist, können wir nach Satz 2.4 die Prädikatensymbole in $\mathcal{P}$ anordnen als $p_1, \ldots, p_n$, wobei aus $p_i <^0 p_j$ stets $i > j$ folgt. Dann definieren wir den Zustand I über Σ wie folgt: Für $p \in \mathcal{E}^* \cup \mathcal{A}$ sei $I(p) = I^{\mathcal{E}}(p)$, und für $p \in \mathcal{P}$ erklären wir

$$I(p) = \begin{cases} 1, & \text{falls } I(\textsc{Bodies}^0_+(p)) = 1; \\ 0, & \text{sonst.} \end{cases}$$

Offensichtlich gilt $I \geq I^{\mathcal{E}}$. Wir müssen aber zeigen, daß $I(p)$ für jedes $p \in \mathcal{P}$ einen eindeutigen Wert hat. Dazu zeigen wir durch Induktion über k für $k = 0, \ldots, n$, daß $I(p_j)$ wohldefiniert ist für jedes $r = 1, \ldots, k$. Für $k = 0$ ist nichts zu zeigen. Nach Induktionsannahme seien $I(p_1), \ldots, I(p_{k-1})$ wohldefiniert. $I(p_k)$ ist wohldefiniert, wenn $I(\textsc{Bodies}^0_+(p_k))$ wohldefiniert ist. Nun treten wegen der Hierarchieeigenschaft in den Klauselrümpfen von $\textsc{Def}(p_k)$ nur Prädikatensymbole aus $\mathcal{E}^* \cup \mathcal{A} \cup \{p_1, \ldots, p_{k-1}\}$ auf. Sie alle aber sind wohldefiniert nach Induktionsannahme.

Zu zeigen bleibt, daß I ein Modell ist für $\textsc{Def}(\mathcal{P})$. Dazu sei $p \in \mathcal{P}$ und $C \equiv L \leftarrow W_i \in \textsc{Def}(p)$. Ist $I(W_i) = 0$, so folgt sofort $I(C) = 1$. Sonst unterscheiden wir: Falls $L \equiv p$ ein positives Literal ist, so ist $I(\textsc{Bodies}^0_+(p)) = I(W_1 \vee \cdots \vee W_i \vee \cdots \vee W_m) = 1$, und damit $I(p) = 1$ nach Konstruktion von I. Ist aber $L \equiv \neg p$ ein negatives Literal, so folgt $I(\textsc{Bodies}^0_-(p)) = 1$. Wäre nun $I(\textsc{Bodies}^0_+(p)) = 1$, so würde auch gelten $I(\neg\, \textsc{Bodies}^0_+(p) \vee \neg\, \textsc{Bodies}^0_-(p)) = 0$, im Widerspruch zur Zulässigkeit von P. Also ist $I(\textsc{Bodies}^0_+(p)) = 0$ und damit $I(p) = 0$. Es folgt $I(\neg p) = 1$ und $I(C) = 1$. □

Lemma 2.6 Sei P ein deterministisches, temporal stratifiziertes F2-Programm über $\Sigma = (\mathcal{E}^*, \mathcal{A}, \mathcal{P})$. Weiter seien I und J zwei Zustände über Σ, die beide Modelle für $\textsc{Def}(\mathcal{P})$ sind, mit $I|_{\mathcal{E}^* \cup \mathcal{A}} = J|_{\mathcal{E}^* \cup \mathcal{A}}$. Dann gilt $I = J$.

Beweis. Wir müssen nur noch zeigen, daß für alle $p \in \mathcal{P}$ gilt $I(p) = J(p)$. Da $\textsc{Def}(\mathcal{P})$ hierarchisch ist, können wir nach Satz 2.4 die Prädikatensymbole in $\mathcal{P}$ anordnen als $p_1, \ldots, p_n$, wobei aus $p_i <^0 p_j$ stets $i > j$ folgt. Wir zeigen durch Induktion über k, für $k = 0, \ldots, n$, daß für jedes $r = 1, \ldots, k$ gilt $I(p_r) = J(p_r)$. Für $k = 0$ ist nichts zu zeigen. Also gelte als Induktionsannahme $I(p_1) = J(p_1), \ldots, I(p_{k-1}) = J(p_{k-1})$. Da P deterministisch ist, ist der Ausdruck $\textsc{Bodies}^0_+(p_k) \vee \textsc{Bodies}^0_-(p_k)$ gültig im Unendlichen. Also gilt mit Korollar 2.3 $I(\textsc{Bodies}^0_\pi(p_k)) = 1$ für ein $\pi \in \{+, -\}$. Falls $I(\textsc{Bodies}^0_+(p_k)) = 1$, so gibt es eine Klausel $p_k \leftarrow W \in \textsc{Def}(p_k)$ mit $I(W) = 1$. Da I Modell ist für $\textsc{Def}(\mathcal{P})$, folgt daraus $I(p_k) = 1$. Andererseits enthält W aufgrund der Hierarchiebedingung nur Symbole aus $\mathcal{E}^* \cup \mathcal{A} \cup \{p_1, \ldots, p_{k-1}\}$, also folgt aus der Voraussetzung und der Induktionsannahme $J(W) = I(W) = 1$, und damit — da auch J ein Modell für $\textsc{Def}(\mathcal{P})$ ist — $J(p_k) = 1 = I(p_k)$. Im Falle $I(\textsc{Bodies}^0_-(p_k)) = 1$ folgt analog $I(p_k) = J(p_k) = 0$. □

Lemma 2.7 Sei P ein zulässiges, temporal stratifiziertes F2-Programm über $\Sigma = (\mathcal{E}^*, \mathcal{A}, \mathcal{P})$. Weiter seien $I = (I_1, \ldots, I_n)$ ein endliches Modell für P und $I^{\mathcal{A}}$ ein Zustand über $\mathcal{A}$. Dann gibt es einen Zustand $I_{n+1} \geq I^{\mathcal{A}}$ über Σ, derart daß $I^+ = (I_1, \ldots, I_{n+1})$ ein Modell für P ist.

Beweis. Wir unterscheiden zwei Fälle:

Fall 1. Zunächst sei $n = 0$, d.h. $I = \epsilon$ ist die leere Interpretation. Wir definieren einen Zustand I_1' über $\mathcal{E}^* \cup \mathcal{A}$ durch

$$I_1'(p) = \begin{cases} I^{\mathcal{A}}(p), & \text{falls } p \in \mathcal{A}; \\ 1, & \text{falls } p \in \mathcal{E}^* \text{ und } p \leftarrow \text{START} \in \text{DEF}_+^0(p); \\ 0. & \text{falls } p \in \mathcal{E}^* \text{ und } p \leftarrow \text{START} \notin \text{DEF}_+^0(p). \end{cases}$$

Bei dieser Konstruktion gilt wegen der Zulässigkeit von P stets $I_1' \models \text{DEF}^0(\mathcal{E}^*)$. Nach Lemma 2.5 gibt es einen Zustand $I_1 \geq I_1'$ über Σ mit $I_1 \models \text{DEF}(\mathcal{P})$. Wir definieren $I^+ = (I_1)$ und erhalten damit auch $I^+ \models \text{DEF}(\mathcal{P})$.

Mit $I_1' \models \text{DEF}^0(\mathcal{E}^*)$ folgt auch $I^+ \models \text{DEF}^0(\mathcal{E}^*)$. Zu zeigen bleibt $I^+ \models \text{DEF}^1(\mathcal{E}^*)$. Nun hat jeder Klauselkopf in $\text{DEF}^1(\mathcal{E}^*)$ die Gestalt $C \equiv \circ L \leftarrow W$ für ein aussagenlogisches Literal L. Bei einer Erweiterung $\bar{I}^+$ von I^+ mit $\bar{I}_1^+(\circ L) = 1$ für jedes aussagenlogische Literal L — dies ist laut Definition 2.4 eine Erweiterung von I^+ auf $F2_\Sigma$ — muß für jede der genannten Klauseln gelten $\bar{I}_1^+(C) = 1$. Also wählen wir diese Erweiterung und erhalten damit I^+ als das gesuchte Modell.

Fall 2. Nun sei $n > 0$, und $I = (I_1, \ldots, I_n)$ sei nach Voraussetzung gegeben. Dann definieren wir einen Zustand I_{n+1}' über $\mathcal{E}^* \cup \mathcal{A}$ durch

$$I_{n+1}'(p) = \begin{cases} I^{\mathcal{A}}(p), & \text{falls } p \in \mathcal{A}; \\ 1, & \text{falls } p \in \mathcal{E}^* \text{ und } I_n(\text{BODIES}_+^1(p)) = 1; \\ 0, & \text{falls } p \in \mathcal{E}^* \text{ und } I_n(\text{BODIES}_+^1(p)) = 0. \end{cases}$$

Nach Lemma 2.5 gibt es einen Zustand $I_{n+1} \geq I_{n+1}'$ über Σ mit $I_{n+1} \models \text{DEF}(\mathcal{P})$; wir erklären damit $I^+ = (I_1, \ldots, I_n, I_{n+1})$. Mit $I \models P$ und $\bar{I}_{n+1}(\text{START}) = 0$ folgt sofort $I^+ \models \text{DEF}^0(\mathcal{E}^*) \dot{\cup} \text{DEF}(\mathcal{P})$.

Um noch $I \models \text{DEF}^1(\mathcal{E}^*)$ zu zeigen, stellen wir zunächst fest, daß die Klauseln in $\text{DEF}^1(\mathcal{E}^*)$ aufgrund ihrer besonderen Form von $I_1, \ldots, I_{n-1}$ stets zu 1 interpretiert werden, wenn nur I ein Modell für P ist. Bei der obigen Konstruktion von I_{n+1}' werden diese Klauseln wegen der Zulässigkeit von P auch von I_n zu 1 interpretiert. Um zu zeigen, daß diese Klauseln von I_{n+1} zu 1 interpetiert werden, verwenden wir das gleiche Argument wie oben im Fall $n = 0$. □

Korollar 2.8 Sei P ein zulässiges, temporal stratifiziertes F2-Programm über $\Sigma = (\mathcal{E}^*, \mathcal{A}, \mathcal{P})$. Dann gibt es zu jedem endlichen Modell $I = (I_1, \ldots, I_n)$ für P einen Zustand I_{n+1} über Σ, derart daß $I^+ = (I_1, \ldots, I_{n+1})$ ein Modell ist für P.

Beweis. Dies ist eine direkte Folgerung aus Lemma 2.7. Wir müssen für I_{n+1} lediglich eine Belegung $I^{\mathcal{A}}$ der Ein/Ausgaben vorgeben, z.B. $I^{\mathcal{A}} = \emptyset$. □

Satz 2.9 Sei P ein zulässiges und temporal stratifiziertes F2-Programm über $\Sigma = (\mathcal{E}^*, \mathcal{A}, \mathcal{P})$. Weiter sei $I^{\mathcal{A}} = (I_1^{\mathcal{A}}, I_2^{\mathcal{A}}, \ldots)$ eine (endliche oder unendliche) Interpretation über $\mathcal{A}$. Dann gibt es eine (gleichlange) Interpretation $I \geq I^{\mathcal{A}}$ über Σ, derart daß I ein Modell ist für P.

Beweis. Wir konstruieren die Zustände von $I = (I_1, I_2, \ldots)$ induktiv mit Hilfe von Lemma 2.7. □

Korollar 2.10 Sei P ein zulässiges, temporal stratifiziertes F2-Programm über $\Sigma = (\mathcal{E}^*, \mathcal{A}, \mathcal{P})$ und $\leftarrow \alpha$ ein F2-Ziel. Dann ist jede gültige Ein/Ausgabefolge für $P \cup \{\leftarrow \alpha\}$ auch eine mögliche Ein/Ausgabefolge für $P \cup \{\leftarrow \alpha\}$.

Beweis. Sei $I^{\mathcal{A}}$ eine gültige Ein/Ausgabefolge für $P \cup \{\leftarrow \alpha\}$. Nach Satz 2.9 existiert eine gleichlange Interpretation $I \geq I^{\mathcal{A}}$, so daß $I \models P$. Da $I^{\mathcal{A}}$ gültige Ein/Ausgabefolge ist, folgt $I \models \alpha$, und damit ist $I^{\mathcal{A}}$ eine mögliche Ein/Ausgabefolge für $P \cup \{\leftarrow \alpha\}$. □

Satz 2.11 Sei P ein deterministisches und temporal stratifiziertes Programm über $\Sigma = (\mathcal{E}^*, \mathcal{A}, \mathcal{P})$. Weiter seien $I = (I_1, I_2, \ldots)$ und $J = (J_1, J_2, \ldots)$ zwei Modelle gleicher Länge für P mit $I|_{\mathcal{A}} = J|_{\mathcal{A}}$. Dann folgt $I = J$.

Beweis. Für die Werte von $p \in \mathcal{A}$ gilt die Behauptung ohnehin. Wir folgern daraus durch Induktion über k, daß für alle $k = 1, 2, \ldots$ gilt $I_k = J_k$.

Sei zunächst $k = 1$. Für jedes $p \in \mathcal{E}^*$ existiert — da P deterministisch ist — eine Klausel $p \leftarrow \text{START}$ oder $\neg p \leftarrow \text{START}$ in P. Falls $C \equiv p \leftarrow \text{START} \in P$, folgt wegen $\bar{I}_1(\text{START}) = \bar{J}_1(\text{START}) = 1$ und wegen $I_1(C) = J_1(C) = 1$, da I und J Modelle sind für P, auch $I_1(p) = 1 = J_1(p)$. Sonst ist $C' \equiv \neg p \leftarrow \text{START} \in P$, und es folgt analog $I_1(p) = 0 = J_1(p)$. Also gilt $I_1|_{\mathcal{E}^* \cup \mathcal{A}} = J_1|_{\mathcal{E}^* \cup \mathcal{A}}$. Mit Lemma 2.6 folgt $I_1 = J_1$.

Nun gelte $k > 1$, und als Induktionsannnahme sei $I_k = J_k$. Zunächst betrachten wir ein beliebiges $p \in \mathcal{E}^*$. Da P deterministisch ist, ist die Formel $\text{BODIES}^1_+(p) \vee \text{BODIES}^1_-(p)$ gültig im Unendlichen. Also gibt es ein $\pi \in \{+, -\}$ mit $\bar{I}_k(\text{BODIES}^1_\pi(p)) = 1$. Falls $\bar{I}_k(\text{BODIES}^1_+(p)) = 1$, gibt es eine Klausel $\circ p \leftarrow W \in \text{DEF}(p)$ mit $\bar{I}_k(W) = 1$. Nach Induktionsannahme folgt $\bar{J}_k(W) = \bar{I}_k(W) = 1$, und schließlich erhalten wir, da I und J Modelle für P sind, $I_{k+1}(p) = 1 = J_{k+1}(p)$ Anderfalls folgt analog $I_{k+1}(p) = 0 = J_{k+1}(p)$. Also ist $I_{k+1}|_{\mathcal{E}^* \cup \mathcal{A}} = J_{k+1}|_{\mathcal{E}^* \cup \mathcal{A}}$. Mit Lemma 2.6 folgt $I_{k+1} = J_{k+1}$. □

Eine wichtige Eigenschaft deterministischer Programme ist die Äquivalenz möglicher und gültiger Ein/Ausgabefolgen, die wir jetzt leicht beweisen können. Aus Korollar 2.10 und dem nachfolgenden Korollar 2.12 folgt nämlich sofort $\mathcal{L}_m(P \cup \{\leftarrow \alpha\}) = \mathcal{L}_g(P \cup \{\leftarrow \alpha\})$ für determinischtische, temporal stratifizierte Programme.

Korollar 2.12 Sei P ein deterministisches und temporal stratifiziertes F2-Programm über $\Sigma = (\mathcal{E}^*, \mathcal{A}, \mathcal{P})$, und sei $\leftarrow \alpha$ ein F2-Ziel. Dann ist jede mögliche Ein/Ausgabefolge für $P \cup \{\leftarrow \alpha\}$ eine gültige Ein/Ausgabefolge für $P \cup \{\leftarrow \alpha\}$.

Beweis. Sei $I^{\mathcal{A}} = (I^{\mathcal{A}}_1, \ldots, I^{\mathcal{A}}_n)$ eine mögliche Ein/Ausgabefolge für $P \cup \{\leftarrow \alpha\}$. Dann gibt es eine Erweiterung I gleicher Länge von $I^{\mathcal{A}}$ auf Σ, so daß $I \models P$

und $I \models \alpha$. Um zu zeigen, daß $I^{\mathcal{A}}$ eine gültige Ein/Ausgabefolge ist, sei $J = (J_1, \ldots, J_n)$ eine beliebige Erweiterung von $I^{\mathcal{A}}$ auf Σ, die Modell für P ist. Dann sind I und J Modelle für P mit $I|_{\mathcal{A}} = J|_{\mathcal{A}}$, d.h. aus Satz 2.11 folgt $I = J$. Damit gilt $J \models \alpha$, und da J beliebig gewählt war, ist $I^{\mathcal{A}}$ eine gültige Ein/Ausgabefolge für $P \cup \{\leftarrow \alpha\}$. □

3 Vergleich mit endlichen Automaten

Endliche Automaten spielten bereits in den Anfängen der Informatik und im Bereich des Compilerbaus eine wichtige Rolle. Sie wurden in der Theorie der *formalen Sprachen* intensiv betrachtet und auf ihre berechenbarkeitstheoretischen Eigenschaften hin untersucht [Harr78]. Auch reaktive Systeme können durch endliche Automaten beschrieben werden, und wir werden in diesem Abschnitt zeigen, daß diese Beschreibungsmethode äquivalent zur Sprache der temporal stratifizierten Programme ist.

3.1 Endliche Automaten

Definition 3.1 Sei A eine endliche Menge von *Aktionen*. Ein *endlicher Automat* über A ist ein 3-Tupel $S = (Q, T, Q_0)$. Dabei ist Q eine endliche Menge von *Zuständen*, $T \subseteq Q \times A \times Q$ bezeichnet die sogenannte *Übergangsrelation*, und $Q_0 \subseteq Q$ bezeichnet eine Menge von sogenannten *Anfangszuständen*.

Für jede Aktion $a \in A$ erklären wir die *Transitionsrelation* T_a entweder als binäre Relation auf Q

$$T_a = \{ (p, q) \mid (p, a, q) \in T \}$$

oder als Abbildung $T_a : Q \to 2^Q$, wobei

$$T_a(p) = \{ q \in Q \mid (p, a, q) \in T \}.$$

Definition 3.2 Sei $S = (Q, T, Q_0)$ ein endlicher Automat über A. S heißt *deterministisch*, wenn $|Q_0| = 1$ gilt und wenn aus $(p, a, q_1) \in T$ und $(p, a, q_2) \in T$ stets folgt $q_1 = q_2$. Sonst heißt S *nicht-deterministisch*.

Im Falle eines deterministischen endlichen Automaten ist $T_a(p)$ stets leer oder einelementig, so daß wir T_a auch als Abbildung $T_a : Q \to Q \cup \{\bot\}$ betrachten können. Hier ist $\bot$ ein neues Symbol, das wir verwenden, um den Fall, daß es zu einem Zustand p und einer Eingabe a keinen Übergang gibt, behandeln zu können. Auch ist es ein bekanntes Ergebnis, daß zu jedem nicht-deterministischen endlichen Automaten ein äquivalenter deterministischer endlicher Automat existiert, so daß wir uns bei Bedarf auf den deterministischen Fall beschränken können.

Definition 3.3 Sei $S = (Q, T, Q_0)$ ein endlicher Automat über A. Ein *Pfad* in S ist eine endliche, alternierende Folge

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n$$

von Zuständen $q_i \in Q$ und Aktionen $a_i \in A$, die in einem Anfangszustand $q_0 \in Q_0$ beginnt, und wobei für jedes $i = 1, \ldots, n$ gilt $(q_{i-1}, a_i, q_i) \in T$.

Ein *unendlicher Pfad* kann entsprechend definiert werden. Wir schreiben für $\hat{a} = (a_1, \ldots, a_n)$ statt $q_0 \xrightarrow{a_1} \cdots \xrightarrow{a_n} q_n$ auch $q_0 \xrightarrow{\hat{a}} q_n$. Im Falle eines deterministischen Automaten ist ein Pfad bereits durch die Folge $(a_1, a_2, \ldots)$ der Aktionen eindeutig definiert, so daß wir auch diese Schreibweise verwenden.

Definition 3.4 Sei $S = (Q, T, Q_0)$ ein endlicher Automat über A. Eine (endliche oder unendliche) Folge $\hat{a} = (a_1, a_2, \ldots)$ heißt von S *akzeptierte Aktionsfolge*, falls es einen Pfad

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots$$

in S gibt. Die Menge der von S akzeptierten Aktionsfolgen bezeichnen wir mit $\mathcal{L}(S)$.

Definition 3.5 Zwei endliche Automaten S und S' über A heißen *äquivalent*, wenn sie die gleichen Aktionsfolgen akzeptieren, d.h. wenn $\mathcal{L}(S) = \mathcal{L}(S')$.

3.2 Äquivalenz von Automaten mit F2-Programmen

In diesem Abschnitt wollen wir einen semantischen Zusammenhang zwischen endlichen Automaten über A und F2-Programmen mit Ein/Ausgaben $\mathcal{A}$ herstellen. Dazu wählen wir als Aktionen $a \in A$ gerade die Zustände über $\mathcal{A}$, d.h. wir vergleichen die korrekten Ein/Ausgabefolgen eines F2-Programms mit Ein/Ausgaben $\mathcal{A}$ mit den akzeptierten Aktionsfolgen eines endlichen Automaten über $2^{\mathcal{A}}$. Wir erinnern daran, daß man einen Zustand über $\mathcal{A}$ auch als Teilmenge von $\mathcal{A}$ oder als Element von $2^{\mathcal{A}}$ auffassen kann.

Definition 3.6 Sei P ein F2-Programm über Σ mit Ein/Ausgaben $\mathcal{A}$, $\leftarrow \alpha$ ein F2-Ziel, und sei S ein endlicher Automat über $A = 2^{\mathcal{A}}$. Wir sagen

- S ist *korrekt* bezüglich $P \cup \{\leftarrow \alpha\}$, wenn jede von S akzeptierte Aktionsfolge eine für $P \cup \{\leftarrow \alpha\}$ mögliche Ein/Ausgabefolge ist.
- S ist *vollständig* bezüglich $P \cup \{\leftarrow \alpha\}$, wenn jede für $P \cup \{\leftarrow \alpha\}$ mögliche Ein/Ausgabefolge eine von S akzeptierte Aktionsfolge ist.
- S ist *äquivalent* zu $P \cup \{\leftarrow \alpha\}$, wenn S korrekt und vollständig bezüglich $P \cup \{\leftarrow \alpha\}$ ist.

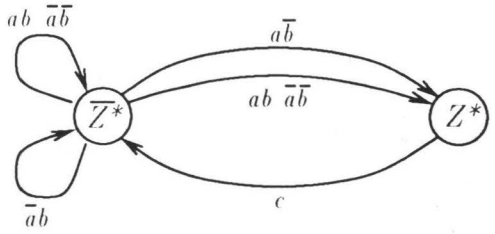

Abbildung 1: Der endliche Automat zu Beispiel 3.1.

Es ist möglich, zu jedem temporal stratifizierten F2-Programm P über einer Signatur $\Sigma = (\mathcal{E}^*, \mathcal{A}, \mathcal{P})$ und jedem $\circ$-freien Ziel $\leftarrow \alpha$ einen ihm entsprechenden endlichen Automaten zu konstruieren, indem man die Belegungen der rekursiven Hilfsprädikate in $\mathcal{E}^*$ als Zustände auffaßt. Diese Konstruktion führen wir jetzt zunächst ein und zeigen anschließend, daß der so konstruierte Automat äquivalent zu $P \cup \{\leftarrow \alpha\}$ ist.

Definition 3.7 Sei P ein temporal stratifiziertes F2-Programm über der Signatur $\Sigma = (\mathcal{E}^*, \mathcal{A}, \mathcal{P})$, und sei $\leftarrow \alpha$ ein temporal flaches F2-Ziel. Dann definieren wir den von $P \cup \{\leftarrow \alpha\}$ *erzeugten endlichen Automat* über $2^{\mathcal{A}}$ durch

$$S_{P,\alpha} = (Q, T, Q_0)$$

mit

- $Q = 2^{\mathcal{E}^*}$, die Menge der Zustände über $\mathcal{E}^*$.
- $T = \{ (I_1^*, I^{\mathcal{A}}, I_2^*) \in 2^{\mathcal{E}^*} \times 2^{\mathcal{A}} \times 2^{\mathcal{E}^*} \mid$
  Es gibt einen Zustand I über Σ mit $I \geq I_1^*$, $I \geq I^{\mathcal{A}}$, $I \models \alpha$, $I \models \textsc{Def}(\mathcal{P})$, und für jede Klausel $C \equiv \circ L \leftarrow L_1 \wedge \ldots \wedge L_n \in \textsc{Def}^1(\mathcal{E}^*)$ mit $\bar{I}(L_1 \wedge \ldots \wedge L_n) = 1$ gilt auch $I_2^*(L) = 1$. $\}$
- $Q_0 = \{ I^{**} \subseteq \mathcal{E}^* \mid$
  Für alle Klauseln $L \leftarrow \textsc{Start} \in \textsc{Def}^0(\mathcal{E}^*)$ gilt $I^{**}(L) = 1$. $\}$

Beispiel 3.1 Wir betrachten nochmals das temporal stratifizierte Programm P und das $\circ$-freie Ziel $\leftarrow \alpha$ aus Beispiel 2.1:

$$\begin{array}{lrcl} P: & \circ Z^* & \leftarrow & \neg Z^* \wedge a \wedge \neg b \\ & \circ \neg Z^* & \leftarrow & \neg Z^* \wedge \neg a \wedge b \\ & \circ \neg Z^* & \leftarrow & \neg Z^* \wedge c \\ \leftarrow \alpha: & \leftarrow & Z^* \rightarrow c & \end{array}$$

Der von $P \cup \{\leftarrow \alpha\}$ erzeugte endliche Automat ist in Abb. 1 dargestellt. Er ist gegeben durch $S_{P,\alpha} = (Q, T, Q)$, wobei die Zustandsmenge Q aus den Zuständen

$Z^\star$ und $\bar{Z}^\star$ besteht, in denen das Prädikatensymbol $Z^\star \in \mathcal{E}^\star$ als wahr bzw. falsch interpretiert wird. Die Übergangsrelation T wird durch die Pfeile im Bild angegeben, dabei bedeutet z.B. die mit $a\bar{b}$ beschriftete Kante von $\bar{Z}^\star$ nach $Z^\star$, daß ein Übergang von $\bar{Z}^\star$ nach $Z^\star$ bei jeder Interpretation über $\{a, b\}$, die a als wahr und b als falsch interpretiert, möglich ist. Wir sehen hier auch, daß dieser endliche Automat nicht-deterministisch ist, da zwei der von $\bar{Z}^\star$ ausgehenden Kanten die gleichen Beschriftungen tragen.

Satz 3.1 Sei P ein zulässiges, temporal stratifiziertes F2-Programm über $\Sigma = (\mathcal{E}^\star, \mathcal{A}, \mathcal{P})$, und sei α ein temporal flaches F2-Ziel. Dann ist der von $P \cup \{\leftarrow \alpha\}$ erzeugte endliche Automat $S_{P,\alpha}$ äquivalent zu $P \cup \{\leftarrow \alpha\}$.

Beweis. Um die Äquivalenz zu beweisen, müssen wir zeigen, daß jede von $S_{P,\alpha}$ akzeptierte Aktionsfolge eine für $P \cup \{\leftarrow \alpha\}$ mögliche Ein/Ausgabefolge ist und umgekehrt.

Sei zunächst $I^{\mathcal{A}} = (I_1^{\mathcal{A}}, I_2^{\mathcal{A}}, \ldots)$ eine mögliche Ein/Ausgabefolge für $P \cup \{\leftarrow \alpha\}$. Dann existiert eine Erweiterung $I = (I_1, I_2, \ldots)$ von $I^{\mathcal{A}}$ auf Σ, die Modell für P und für α ist. Falls $I^{\mathcal{A}} = (I_1^{\mathcal{A}}, \ldots, I_n^{\mathcal{A}})$ endlich ist, existiert nach Korollar 2.8 auch ein Zustand I_{n+1}, so daß $I = (I_1, \ldots, I_n, I_{n+1})$ ein Modell für P ist. Wir bilden $I^\star = (I_1^\star, I_2^\star, \ldots)$ mit $I^\star = I|_{\mathcal{E}^\star}$ und behaupten, daß in $S_{P,\alpha}$ ein Pfad

$$I_1^\star \xrightarrow{I_1^{\mathcal{A}}} I_2^\star \xrightarrow{I_2^{\mathcal{A}}} I_3^\star \xrightarrow{I_3^{\mathcal{A}}} \cdots$$

existiert, der im endlichen Fall in $I_{n+1}^\star$ endet.

Dies zeigen wir durch vollständige Induktion über die Länge des Pfades. Im Induktionsanfang müssen wir zeigen, daß $I_1^\star \in Q_0$. Sei $L \leftarrow \textsc{Start} \in \textsc{Def}^0(\mathcal{E}^\star)$. Da $\bar{I}_1(\textsc{Start}) = 1$ für jede Erweiterung von I auf $F2_\Sigma$, und da $I \models P$, folgt $I_1^\star(L) = I_1(L) = 1$. Nach Konstruktion von Q_0 gilt dann $I_1^\star \in Q_0$.

Nehmen wir nun an, daß für ein $k \in \mathbb{N}_0$ der Pfad

$$I_1^\star \xrightarrow{I_1^{\mathcal{A}}} \cdots \xrightarrow{I_{k-1}^{\mathcal{A}}} I_k^\star$$

bereits konstruiert ist. Im endlichen Fall können wir $k \leq n$ annehmen. Hier gilt $I_k \geq I_k^\star$, $I_k \geq I_k^{\mathcal{A}}$ und $I_k \models \alpha$. Außerdem folgt mit $I \models P$, daß für jede Klausel

$$C \equiv \circ L \leftarrow L_1 \wedge \ldots \wedge L_m \in \textsc{Def}^1(\mathcal{E}^\star)$$

mit $\bar{I}_k(L_1 \wedge \ldots \wedge L_m) = 1$ gilt $\bar{I}_k(\circ L) = 1$, und damit $\bar{I}_{k+1}(L) = 1$. Wegen $I^\star = I|_{\mathcal{E}^\star}$ ist dann auch $I_{k+1}^\star(L) = 1$. Daraus aber folgt $(I_k^\star, I_k^{\mathcal{A}}, I_{k+1}^\star) \in T$.

Sei nun umgekehrt $I^{\mathcal{A}} = (I_1^{\mathcal{A}}, I_2^{\mathcal{A}}, \ldots)$ eine von $S_{P,\alpha}$ akzeptierte Aktionsfolge. Das heißt, es gibt einen Pfad

$$q_1 \xrightarrow{I_1^{\mathcal{A}}} q_2 \xrightarrow{I_2^{\mathcal{A}}} \cdots$$

in $S_{P,\alpha}$. Im endlichen Fall ist $I^{\mathcal{A}} = (I_1^{\mathcal{A}}, \ldots, I_n^{\mathcal{A}})$ für ein $n \in \mathbb{N}_0$, und der Pfad endet in in q_{n+1}. Wir zeigen durch Induktion über k für $k = 0, 1, \ldots$ die folgende Aussage:

> „*Es existiert eine Interpretation* $I = (I_1, \ldots, I_k)$ *über* Σ *mit* $I \geq (q_1, \ldots, q_k)$, $I \geq (I_1^{\mathcal{A}}, \ldots, I_k^{\mathcal{A}})$, $I \models P$, $I \models \alpha$. *Im Falle* $k = 0$ *gilt für jede Klausel* $L \leftarrow \textsc{Start} \in \textsc{Def}^0(\mathcal{E}^\star)$ *immer* $q_1(L) = 1$. *Sonst, im Falle* $k > 0$, *gilt für jede Klausel* $\circ L \leftarrow L_1 \wedge \ldots \wedge L_m \in \textsc{Def}^1(\mathcal{E}^\star)$ *mit* $\bar{I}_k(L_1 \wedge \ldots \wedge L_m) = 1$ *immer* $q_{k+1}(L) = 1$."

Dann ist I nämlich eine Erweiterung von $I^{\mathcal{A}}$ auf Σ, die Modell für P und für α ist, d.h. $I^{\mathcal{A}}$ ist eine mögliche Ein/Ausgabefolge für $P \cup \{\leftarrow \alpha\}$.

Zunächst stellen wir im Induktionsanfang für $k = 0$ fest, daß für die leere Interpretation $I = \epsilon$ alle geforderten Bedingungen gelten. Die Aussage $q_1(L) = 1$ für $L \leftarrow \textsc{Start} \in \textsc{Def}^0(\mathcal{E}^\star)$ folgt direkt aus der Definition von Q_0.

Nehmen wir also an, daß die Existenz von $I = (I_1, \ldots, I_k)$ wie in der Induktionsvoraussetzung für ein $k \in \mathbb{N}_0$ bereits gesichert wurde. Im endlichen Fall gilt dabei $k < n$. Die Existenz des Überganges

$$q_{k+1} \xrightarrow{I_{k+1}^{\mathcal{A}}} q_{k+2}$$

bedeutet, daß es einen Zustand I_{k+1} über Σ gibt mit $I_{k+1} \geq q_{k+1}$, $I_{k+1} \geq I_{k+1}^{\mathcal{A}}$, $I_{k+1} \models \textsc{Def}(\mathcal{P})$, $I_{k+1} \models \alpha$, so daß für alle Klauseln $\circ L \leftarrow L_1 \wedge \ldots \wedge L_m \in \textsc{Def}(\mathcal{E}^\star)$ mit $\bar{I}'_{k+1}(L_1 \wedge \ldots \wedge L_m) = 1$ auch $q_{k+2}(L) = 1$ gilt. Dann betrachten wir die Interpretation $I' = (I_1, \ldots, I_k, I_{k+1})$ über Σ. Aufgrund der genannten Eigenschaften von I_{k+1} und der Induktionsvoraussetzung folgt sofort $I' \geq (q_1, \ldots, q_{k+1})$, $I' \geq (I_1^{\mathcal{A}}, \ldots, I_{k+1}^{\mathcal{A}})$ und $I' \models \alpha$. Um auch $I' \models P$ zu begründen, verwenden wir $I \models P$, $I_{k+1} \models \textsc{Def}(\mathcal{P})$, $I_{k+1} \geq q_{k+1}$ und die Eigenschaften von q_{k+1} aus der Induktionsvoraussetzung. Die entsprechenden Eigenschaften für q_{k+2} haben wir ebenfalls bereits gesichert, und damit sind alle Bedingungen der Induktionsbehauptung für den Fall $k + 1$ bewiesen. Somit folgt die Behauptung. □

Theorem 3.2 *Sei* P *ein zulässiges, temporal stratifiziertes F2-Programm über* $\Sigma = (\mathcal{E}^\star, \mathcal{A}, \mathcal{P})$, *wobei* $|\mathcal{E}^\star| = K$, *und sei* α *ein temporal flaches F2-Ziel. Dann gibt es einen zu* $P \cup \{\leftarrow \alpha\}$ *äquivalenten endlichen Automat* $S = (Q, T, Q_0)$ *mit* $|Q| \leq 2^K$ *Zuständen.*

Beweis. Dies ist eine direkte Folgerung aus Satz 3.1. Der von $P \cup \{\leftarrow \alpha\}$ erzeugte endliche Automat $S_{P,\alpha}$ hat die Zustandsmenge $Q = 2^{\mathcal{E}^\star}$, und diese enthält gerade $2^{|\mathcal{E}^\star|} = 2^K$ Zustände. □

Da $S_{P,\alpha}$ nicht notwendig deterministisch ist, bezieht sich die Zahl 2^K der Zustände in Theorem 3.2 auf die Anzahl der Zustände eines nicht-determistischen Automaten. Dann gibt es bekanntlich einen äquivalenten deterministischen Automat mit nicht mehr als 2^{2^K} Zuständen.

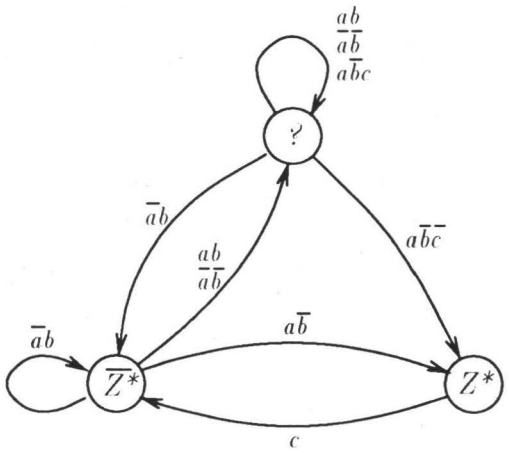

Abbildung 2: Der deterministische endliche Automat zu Beispiel 3.1.

Beispiel 3.2 In Beispiel 3.1 war $|\mathcal{E}^\star| = 1$, und der nicht-deterministische endliche Automat hatte $2 \leq 2^1$ Zustände. Abb. 2 zeigt den äquivalenten deterministischen Automat, der durch die bekannte Konstruktion entstanden ist. Hier erscheint ein zusätzlicher Zustand „?", der immer dann angenommen wird, wenn nicht feststeht, in welchem der beiden Zustände sich der nicht-deterministische Automat befinden würde.

Für uns ist an dieser Stelle vor allem die Existenz einer oberen Grenze für die Zahl der Zustände wichtig, wobei der Wert dieses Grenzwertes aus syntaktischen Eigenschaften des Programmes gewonnen werden kann. Das bedeutet nämlich, daß wir uns beim Generieren eines reaktiven Systems auf eine nur noch endliche Menge von möglichen Lösungskandidaten (endlichen Automaten) beschränken können.

Natürlich ist es wünschenswert, die Zahl der möglichen Lösungen klein zu halten. Eine Möglichkeit dazu bietet die Beschränkung auf deterministische Programme, denn in diesem Fall sind die erzeugten Automaten deterministisch. Dadurch entfällt ggf. die Konstruktion des deterministischen Automaten aus einem nicht-deterministischen Automaten, was zu einer drastischen Reduktion der Zahl der Zustände führen kann. Die Grundlage dafür ist die folgende Aussage.

Satz 3.3 Sei P ein deterministisches, temporal stratifiziertes Programm, und sei α ein $\circ$-freies Ziel. Dann ist der von $P \cup \{\leftarrow \alpha\}$ erzeugte endliche Automat $S_{P,\alpha}$ deterministisch.

Beweis. Wir müssen lediglich zeigen, daß die Mengen Q_0 und $\textsc{Post}(I^\star, I^{\mathcal{A}}) = \{\, I_1^\star \subseteq \mathcal{E}^\star \mid (I^\star, I^{\mathcal{A}}, I_1^\star) \in T \,\}$ stets einelementig sind.

Seien $I_1^\star, I_2^\star \in Q_0$. Wir zeigen, daß für jedes $p \in \mathcal{E}^\star$ gilt $I_1^\star(p) = I_2^\star(p)$. Da P deterministisch ist, existiert zu jedem $p \in \mathcal{E}^\star$ eine Klausel $L \leftarrow \textsc{Start} \in \textsc{Def}^0(p)$ mit $I_1^\star(L) = 1 = I_2^\star(L)$. Da aber p das Atom von L ist, folgt $I_1^\star(p) = I_2^\star(p)$.

Seien nun $I_1^\star, I_2^\star \in \textsc{Post}(I^\star, I^\mathcal{A})$ für beliebige Zustände $I^\star \subseteq \mathcal{E}^\star$ und $I^\mathcal{A} \subseteq \mathcal{A}$. Da nach der Definition von $\textsc{Post}(I^\star, I^\mathcal{A})$ für $i \in \{1,2\}$ stets gilt $(I^\star, I^\mathcal{A}, I_i^\star) \in T$, gibt es Zustände I_1 und I_2 über Σ mit $I_i \geq I^\star$, $I_i \geq I^\mathcal{A}$, so daß für jede Klausel $C \equiv \circ L \leftarrow W \in \textsc{Def}^1(\mathcal{E}^\star)$ mit $I_i(W) = 1$ auch $I_i^\star(L) = 1$ gilt, und zwar für jedes $i \in \{1,2\}$. Mit Lemma 2.6 folgt zunächst $I_1 = I_2$. Sei nun $p \in \mathcal{E}^\star$. Da P deterministisch ist, ist die Formel $\textsc{Bodies}_+^1(p) \vee \textsc{Bodies}_-^1(p)$ gültig im Unendlichen. Nach Korollar 2.3 gilt jetzt $I_i(\textsc{Bodies}_\pi^1(p)) = 1$ für ein $\pi \in \{+,-\}$. Somit gibt es eine Klausel $\circ L \leftarrow W \in \textsc{Def}^1(\mathcal{E}^\star)$ mit $I_i(W) = 1$, für jedes $i \in \{1,2\}$. Dann gilt aber $I_1^\star(L) = 1 = I_2^\star(L)$, und da p das Atom von L ist, folgt daraus $I_1^\star(p) = I_2^\star(p)$. □

4 Ausblick

In dieser Arbeit haben wir eine logische Programmiersprache zur Spezifikation zustandsendlicher reaktiver Systeme vorgestellt. Diese Sprache zeichnet sich durch eine rein deklarative Beschreibung des Systems und der an es gestellten Anforderungen aus. Es wird eine strenge Klauselnotation verwendent, um die Bearbeitung durch resolutionsbasierte Verfahren zu ermöglichen. Ähnliche logische Programmierprachen wurden bereits mehrfach vorgeschlagen (beispielsweise in [AbMa89], [Gabb87]) — wir interessieren uns besonders für die Anwendbarkeit zur Verifikation und zur Generierung reaktiver Systeme.

Deshalb haben wir die Normalform der temporal stratifizierten Programme so gewählt, daß eine direkte Korrespondenz zu endlichen Automaten, mit denen reaktive Systeme implementiert werden, hergestellt werden kann. Im Prinzip ist es sicherlich möglich, solche Automaten aufgrund ihrer Definition effektiv zu konstruieren und das Generierungsproblem auf diese Weise zu lösen. Allerdings ist Durchführbarkeit dieser Methode unter Effizienzaspekten doch fragwürdig, und daher werden wir versuchen, die Lösungen mit Hilfe von Resolution zu ermitteln.

Die Klauselnotation der temporal stratifizierten Programme wurde so gewählt, daß es möglich ist, einen Beweis der Gültigkeit der als temporal flaches Ziel gestellten Sicherbedingung durch ein der SLD-Resolution ähnliches Verfahren durchzuführen. Falls dieser Beweis scheitert, können der aus dem Beweisversuch hervorgehenden Information Regeln entnommen werden, durch deren Hinzunahme zu dem ursprünglichen Programm eine erfolgreiche Vollendung des Beweises möglich wird. Damit wäre das Problem der Generierung eines nachweisbar korrekten Steuerungsprogramms gelöst. An den entsprechenden Algorithmen arbeiten wir zur Zeit.

Ein weiteres Ziel ist eine Erweiterung der bisher sehr einfach gehaltenen Sprache um zusätzliche Konstrukte, um eine bessere Anwendernähe zu erreichen. Bei-

spielsweise wäre es denkbar, im Rumpf von Klauseln an Stelle von Konjunktionen von Literalen beliebige Konstrukte zuzulassen, die auch vergangenheitsorientierte temporale Operatoren enthalten können. Solche erweiterte Programme könnten vor der Bearbeitung durch Resolution in die Normalform eines temporal stratifizierten Programms transformiert werden.

Literatur

[AbMa89] Abadi, M., Manna, Z.: *Temporal logic programming.* J. Symbolic Computation **8**, 277–295, 1989.

[BeBe91] Benveniste, A., Berry, G.: *The synchronous approach to reactive and real-time systems.* Proc. IEEE **79** (9), 1270–1282, 1991.

[BeGo92] Berry, G., Gonthier, G.: *The ESTEREL synchronous programming language: design, semantics, implementation.* Science of Computer Programming **19**, 87–152, 1992.

[CES86] Clarke, E. M., Emerson, E. A., Sistla, A. P.: *Automatic verification on finite-state concurrent systems using temporal logic specifications.* ACM Transactions on Programming Languages and Systems **8** (2), 244–263, 1986.

[CGH94] Cremers, A. B., Griefahn, U., Hinze, R.: *Deduktive Datenbanken.* Vieweg, 1994.

[CPHP87] Caspi, P., Pilaud, D., Halbwachs, N., Plaice, J. A.: *LUSTRE: a declarative language for programming synchronous systems.* Proc. 14th ACM Symp. on Principles of Programming Languages, 178–188, München, 1987.

[Gabb87] Gabbay, D.: *The declarative past and imperative future.* Proc. Temporal Logic in Specifications, 409–448, 1987.

[Halb93] Halbwachs, N.: *Synchronous programming of reactive systems.* Kluwer Academic Publishers, Dordrecht-Boston-London, 1993.

[Harr78] Harrison, M. A.: *Introduction to formal language theorie.* Addison-Wesley, 1978.

[HPOG89] Halbwachs, N., Pilaud, D., Ouabdesselam, F., Glory, A.-C.: *Specifying, Programming and Verifying Real-Time Systems Using a Synchronous Declarative Language.* Proc. Workshop Automatic Verification of Finite State Systems, 213–231, Grenoble, 1989.

[Krög86] Kröger, F.: *Temporal Logic of Programs.* Springer, 1986.

[LGLL91] LeGuernic, P., Gautier, T., LeBorgne, M., LeMaire, C.: *Programming real-time applications with SIGNAL.* Proc. IEEE **79** (9), 1321–1336, 1991.

[Lloyd84] Lloyd, J. W.: *Foundations of logic programming.* Springer, 1984.

[Ostr89] Ostroff, J.: *Temporal logic for real-time systems.* Research Studies Press Ltd., Advanced Software Development Series, Taunton, Somerset, 1989.

[PiHa88] Pilaud, D., Halbwachs, N.: *From a synchronous declarative language to a temporal logic dealing with multiform time.* Proc. Symp. Formal Techniques in Real Time and Fault Tolerant Systems, Warrick, 1988.

[Redd86] Reddy, U. S.: *Logic languages based on functions: semantics and implementation.* Ph.D. thesis, University of Utah, 1986.

[RHR91] Ratel, C., Halbwachs, N., Raymond, P.: *programming and verifying critical systems by means of the synchronous data-flow language Lustre.* Software Engineering Notes **16** (5), 112–119, 1991.

[Sifa82] Sifakis, J.: *A unified approach for studying the properties of transition systems.* Theoretical Computer Science **18** (3), 227–258, 1982.

[SpMa92] Spies, K. und Mayer, O.: *EITeL — ein intervallbasierter temporallogischer Ansatz.* Interner Bericht, FB Informatik, Universität Kaiserslautern, 1992.

[Wagn91] Wagner, G.: *Logic programming with strong negation and inexact predicates.* J. Logic and Computation **1** (6), 835–859, 1991.