Interner Bericht

Eine Fixpunkt-Semantik für temporal stratifizierte Programme

Robi Malik Otto Mayer

28. November 1994
261/94

Fachbereich Informatik

Universität Kaiserslautern · Postfach 3049 · D-67653 Kaiserslautern

Eine Fixpunkt-Semantik für temporal stratifizierte Programme

Robi Malik Otto Mayer

29. November 1994
261/94

Zusammenfassung

In dieser Arbeit beschreiben wir einen Ansatz zur automatischen Synthese zustandsendlicher, reaktiver Systeme, ausgehend von einer rein deklarativen, logischen Spezifikation. Dazu verwenden wir *temporal stratifizierte Programme*, das sind spezielle Logik-Programme auf der Grundlage einer linearen, temporalen Aussagenlogik. Die Umgebung eines zu implementierenden Steuerungsprogrammes wird hier durch eine Menge von PROLOG-ähnlichen Programmklauseln beschrieben; zusätzlich wird eine Sicherheitsbedingung angegeben, die in dem System gelten soll. Wir zeigen, wie durch eine solche Spezifikation ein sie implementierender endlicher Automat definiert ist und geben einen Algorithmus zu seiner Berechnung auf der Grundlage einer Fixpunkt-Iteration an.

1 Einführung

Reaktive Systeme sind 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. Allerdings sind die Spezifikationen in den meisten Fällen unterbestimmt, so daß zur Modellierung in der Regel *nicht-deterministische* endliche Automaten benötigt werden. Auf diesen Grundlagen konnten inzwischen sehr effiziente Verifikationsverfahren entwickelt werden [BCLMD94].

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.

Dazu haben wir in der Arbeit [MaMa94] eine einfache, logische Programmiersprache zur Spezifikation zustandsendlicher, reaktiver Systeme vorgestellt: Die Sprache der *temporal stratifizierten Programme*. Als Grundlage dient hier eine lineare, temporale Aussagenlogik, für die eine PROLOG-ähnliche Klauselnotation angeben wird. Die Umgebung eines reaktiven Systems wird nun deklarativ durch eine Menge solcher Klauseln — die einem endlichen Automaten entspricht — beschrieben. Außerdem können die *Sicherheitsbedingungen*, die in dem System gelten sollen, in einer ebenfalls rein deklarativen Form als Beweisziel spezifiziert werden.

In dieser Arbeit befassen wir uns mit der Fragestellung, ob und wann ein Steuerungsprogramm existiert, das die Einhaltung der geforderten Sicherheitsbedingungen garantiert, und wie wir dieses ggf. automatisch generieren können. Es stellt sich heraus, daß es — unter bestimmten Voraussetzungen — ein *allgemeinstes* Steuerungsprogramm gibt, das das Problem löst. Diese Steuerung ist im allgemeinen hochgradig nicht-deterministisch, umfaßt aber alle möglichen Lösungen. Wir werden sehen, wie sich dieses Steuerungsprogramm als Ergebnis einer Fixpunkt-Iteration ergibt, die auch zu seiner effektiven Berechnung benutzt werden kann.

Dieser Bericht gliedert sich wie folgt: Zunächst gehen wir in Abschnitt 2 auf die Grundlagen der verwendeten Logik und die darauf aufbauende Gestalt der temporal stratifizierten Programme ein. In Abschnitt 3 gehen wir dann zunächst genauer auf das Generierungsproblem ein und erklären anschließend, wie es gelöst wird. Wir zeigen, wie das gesuchte Steuerungsprogramm als Ergebnis einer Fixpunkt-Iteration über dem Raum aller potentiellen Lösungskandidaten berechnet werden kann, und daß diese Vorgehensweise vollständig ist. Zum Abschluß folgt noch eine kurze Zusammenfassung und ein Ausblick auf zukünftige Forschungsaktivitäten.

2 F2-Programme

In diesem Abschnitt definieren wir die formalen, logischen Grundlagen unserer Arbeit. Als erstes führen wir eine einfache, temporale Logik namens F2 und darauf aufbauend eine Klasse von elementaren Logik-Programmen ein. Wir werden sehen, daß wir zur klaren Spezifikation reaktiver Systeme durch diese Programme eine *Normalform* benötigen, die wir in der Eigenschaft der *temporalen Stratifizierung* finden. Schließlich folgt eine Untersuchung der Semantik unserer Logik-Programme, die eng mit den Eigenschaften endlicher Automaten zusammenhängt. Die meisten Ergebnisse dieses Abschnittes entstammen der Arbeit [MaMa94], in der auch einige zusätzliche Aspekte untersucht werden.

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 $p \in \Sigma$.
- $\bar{I}_k(\text{START}) = 1$ 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 $k = 1, \ldots, m$.
- ▷ $\bar{I}'_k(\textsc{start}) = \bar{I}_k(\textsc{start}) = 1$ 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 $\bar{I}'_k(\circ\alpha) = \bar{I}_k(\circ\alpha) = \bar{I}_{k+1}(\alpha) = \bar{I}'_{k+1}(\alpha)$.

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

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

Definition 2.8 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* (gelesen „next-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 2.9 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.

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

[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.

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 2.11 Sei Σ ein endliches Alphabet. Ein *aussagenlogisches Literal* ist eine F2-Formel der Gestalt p oder $\neg p$ für ein $p \in \Sigma$. Ein *o-temporales Literal* ist eine F2-Formel der Gestalt $\circ^i L$ für ein aussagenlogisches Literal L. Ein *o-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 o-freies Literal L.

Definition 2.12 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 o-temporales Literal L und o-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 o-freie temporale Literale $L_1, \ldots, L_n$.

Definition 2.13 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 — die dabei entstehenden Programme werden durch einen *Normalisierungsalgorithmus* in Normalformen ähnlich der hier betrachteten Form transformiert [Lloyd84]. Eine Übertragung dieses Verfahren auf unsere F2-Programme ist möglich. Es ist sogar vorstellbar, 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 struk-

turieren und dann einige der Aussagenvariablen aus Σ als Ein- bzw. Ausgaben auszeichnen.

Definition 2.14 Sei P ein F2-Programm über Σ und $p \in \Sigma$. Dann erklären wir die *positive Definition* i-ter Stufe von p, $\textsc{Def}^{+}_{i}(p)$, als die Menge der Klauseln in P, in deren Kopf das o-temporale Literal $\circ^i p$ auftritt. Analog wird die *negative Definition* i-ter Stufe von p erklärt:

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

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

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

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

Definition 2.15 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 2.16 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 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. Die Zusammenhänge mit deterministischen und nicht-deterministischen endlichen Automaten werden in [MaMa94] genauer ausgearbeitet.

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

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

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 $I' = (\{a, b, Z^\circ\}, \ldots, \{a, b, Z^\circ\})$, 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 hierarchische 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 wird 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 nächsten Abschnitt einführen werden.

In diesem Abschnitt definieren wir erst einmal 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 2.17 (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 2.18 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. Wie üblich bezeichnen wir mit $|M|$ die Anzahl der Elemente der endlichen Menge M. Nun seien $p, q \in \Sigma$ mit $p \sqsubset^0 q$. 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 auch aus $p <^0 q$ die Aussage $N(p) > N(q)$ folgt. Seien also $p, q \in \Sigma$ mit $p <^0 q$. Dann gibt es aufgrund der Definition der Relation $<^0$ 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 n. Im Induktionsanfang für $n = 1$ 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 Behauptung sei für ein $n > 0$ bereits bewiesen, und betrachten den Fall $p = p_0 \sqsubset^0 \cdots \sqsubset^0 p_n \sqsubset^0 p_{n+1} = q$. Nach Induktionsannahme ist dann $N(p_0) > N(p_n)$, und wegen $p_n \sqsubset^0 p_{n+1}$ folgt aus der Voraussetzung $N(p_n) > N(p_{n+1})$. Damit haben wir $N(p) = N(p_0) > N(p_n) > N(p_{n+1}) = N(q)$.

(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:

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

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 $i < |\mathcal{N}_k|$, $j < |\mathcal{N}_l|$ und $k > l$. Wegen $|\mathcal{N}_l| = n_{l+1} - n_l$ folgt

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

Um noch zu zeigen, daß N' injektiv ist, betrachten wir $p, q \in \Sigma$ 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 $q = p_l^j$, wobei $k = l$. Dann aber folgt aus der Definition von N' auch $i = j$ und somit $p = q$, d.h. N' ist injektiv.

(H4) $\Longrightarrow$ *(H1).* Sei $N \colon \Sigma \to \mathbb{N}_0$ 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. □

2.4 Temporal stratifizierte Programme

Leider zeigt sich, daß die Eigenschaft der temporalen Hierarchie nicht ausreicht, um die als Hilfsvariablen 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 2.19 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 $\circ$-*abgestuft* (gelesen „next-abgestuft"), wenn C eine der beiden Gestalten

$$C \equiv L \leftarrow \textsc{Start}$$
$$C \equiv \circ L \leftarrow L_1 \wedge \ldots \wedge L_n$$

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

Definition 2.20 Ein F2-Ziel $\leftarrow \alpha$ heißt temporal flach, wenn es die Gestalt

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

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 $\mathcal{E}^\circ$, 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 $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$.

Definition 2.21 Sei $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$ eine Signatur. Ein *temporal stratifiziertes F2-Programm* über Σ ist ein F2-Programm über Σ, das den folgenden drei Bedingungen genügt:

- ▷ $\textsc{Def}(\mathcal{E}^\circ)$ ist o-abgestuft.
- ▷ $\textsc{Def}(\mathcal{A}) = \emptyset$.
- ▷ $\textsc{Def}(\mathcal{P})$ 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.

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 Ein/Ausgaben in $\mathcal{A}$ und den Werten der Zustands-Prädikatensymbole in $\mathcal{E}^\circ$ im vorausgegangenen Zustand berechnet werden können. Damit haben wir in den Prädikatensymbolen aus der Menge $\mathcal{E}^\circ$ genau die Hilfsvariablen gefunden, in denen die zustandsübergreifende Information gespeichert wird.

2.5 Modelle temporal stratifizierter Programme

Bei temporal stratifizierten Programmen ist es möglich, eine sinnvolle Definition von *Zulässigkeit* einzuführen, die ihre 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 — die Zusammenhänge mit der Erfüllbarkeit zeigen wir später auf.

Definition 2.22 Sei P ein F2-Programm über Σ, und sei $p \in \Sigma$. Dann definieren wir für jedes $i \in \mathbb{N}_0$ und jedes $\pi \in \{+, -\}$

$$\textsc{Bodies}^i_\pi(p) = \bigvee \{ W \mid L \leftarrow W \in \textsc{Def}^i_\pi(p) \}.$$

Die Formel $\textsc{Bodies}^i_\pi(p)$ beschreibt also die Disjunktion der Rümpfe aller Klauseln in $\textsc{Def}^i_\pi(p)$.

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

Definition 2.23 Sei $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$ 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

$$\neg\, \textsc{Bodies}^i_+(p) \vee \neg\, \textsc{Bodies}^i_-(p)$$

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

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

im Unendlichen gültig sind, und außerdem für jedes $p \in \mathcal{E}^\circ$ 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 $\textsc{Def}^1(\mathcal{E}^\circ)$ 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 $\textsc{Def}^0(\mathcal{E}^\circ)$ 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:

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

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

Dies entspricht in seiner Semantik ungefähr der vom logischen Programmieren bekannten *Negation durch Scheitern* [Lloyd84].

Es ist möglich, zu jedem zulässigen temporal stratfizierten Programm ein Modell zu konstruieren, d.h. zulässige temporal stratifizierte Programme sind stets erfüllbar im Unendlichen. Ist das Programm zusätzlich deterministisch, so ist dieses Modell sogar eindeutig bestimmt. Diese Eigenschaften werden in [MaMa94] formuliert und bewiesen. Wir wiederholen hier nur die Aussagen, die wir später noch benötigen. Zunächst beginnen wir mit einigen Hilfsaussagen über die Existenz und Eindeutigkeit geeigneter Zustände.

Lemma 2.5 Sei P ein zulässiges, temporal stratifiziertes F2-Programm über $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$, und sei $I^{\mathcal{E}}$ ein Zustand über $\mathcal{E}^\circ \cup \mathcal{A}$. 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 $p \in \mathcal{E}^\circ \cup \mathcal{A}$ 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 $j = 1, \ldots, k$. Für $k = 0$ ist nichts zu zeigen. Als 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 $\mathcal{E}^\circ \cup \mathcal{A} \cup \{p_1, \ldots, p_{k-1}\}$ 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 ist $I(W_i) = 1$, und wir unterscheiden: Falls $L \equiv p$ ein positives Literal ist, gilt $I(\textsc{Bodies}^0_+(p)) = I(W_1 \vee \cdots \vee W_i \vee \cdots \vee W_m) = 1$, und damit $I(p) = 1$ nach Konstruktion von I. Sonst ist $L \equiv \neg p$ ein negatives Literal, es 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 zulässiges, temporal stratifiziertes F2-Programm über $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$. 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 $\mathcal{E}^\circ \cup \mathcal{A}$ durch

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

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

Mit $I_1' \models \textsc{Def}^0(\mathcal{E}^\circ)$ folgt auch $I^+ \models \textsc{Def}^0(\mathcal{E}^\circ)$. Zu zeigen bleibt $I^+ \models \textsc{Def}^1(\mathcal{E}^\circ)$. Nun hat jeder Klauselkopf in $\textsc{Def}^1(\mathcal{E}^\circ)$ 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 $\mathcal{E}^\circ \cup \mathcal{A}$ durch

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

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

Um noch $I \models \textsc{Def}^1(\mathcal{E}^\circ)$ zu zeigen, stellen wir zunächst fest, daß die Klauseln in $\textsc{Def}^1(\mathcal{E}^\circ)$ 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 $n = 0$. □

Lemma 2.7 *P sei ein temporal stratifiziertes F2-Programm über $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$, und sei $I = (I_1, \ldots, I_k)$ ein endliches Modell für P. Weiter sei J ein Zustand über Σ mit $J|_{\mathcal{E}^\circ} = I_k|_{\mathcal{E}^\circ}$ und $J \models \textsc{Def}(\mathcal{P})$. Dann ist auch $I' = (I_1, \ldots, I_{k-1}, J)$ ein Modell für P.*

Beweis. Offenbar ist $I' \models \textsc{Def}(\mathcal{P})$, da $\textsc{Def}(\mathcal{P})$ temporal flach ist und jeden Zustand von I' als Modell hat. Außerdem gilt für jede Interpretation I'' mit $I''|_{\mathcal{E}^\circ} = I|_{\mathcal{E}^\circ}$ auch $I'' \models \textsc{Def}^0(\mathcal{E}^\circ)$, da in $\textsc{Def}^0(\mathcal{E}^\circ)$ nur Prädikatensymbole aus $\mathcal{E}^\circ$ vorkommen. Also ist auch $I' \models \textsc{Def}^0(\mathcal{E}^\circ)$. Um auch $I \models \textsc{Def}^1(\mathcal{E}^\circ)$ zu begründen, sei $C \equiv \circ L \leftarrow L_1 \wedge \ldots \wedge L_n \in \textsc{Def}^1(\mathcal{E}^\circ)$. Da $I \models P$, folgt sofort $I_1(C) = \cdots =$

$I_{k-2}(C) = 1$. Auch finden wir leicht eine Erweiterung $\bar{J}$ von J auf $L2_\Sigma$ mit $J(C) = 1$. Wir müssen nur noch $I_{k-1}(C)$ betrachten. Sei also $I_{k-1}(L_1 \wedge \ldots \wedge L_n) = 1$. Da $I \models P$ und $J|_{\mathcal{E}^\circ} = I_k|_{\mathcal{E}^\circ}$ folgt $J(L) = I_k(L) = I_{k-1}(\circ L) = 1$, und somit gilt $I_{k-1}(C) = 1$. □

Satz 2.8 Sei P ein zulässiges und temporal stratifiziertes F2-Programm über $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$. 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 2.6. □

3 Die Fixpunkt-Konstruktion

3.1 Ein Modell für reaktive Systeme

Bisher haben wir immer temporal stratifizierte Programme über Signaturen $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$ betrachtet und ihnen als Bedeutung Sprachen über $2^{\mathcal{A}}$, den möglichen Belegungen der Ein/Ausgabe-Variablen, zugeordnet. Dabei haben wir die Mengen der möglichen und der gültigen Ein/Ausgabefolgen für $P \cup \{\leftarrow \alpha\}$ als denkbare Bedeutungen eines Programmes zusammen mit einem Ziel kennengelernt. Auf diese Weise ist es möglich, eine Beziehung zu endlichen Automaten herzustellen — dieser Aspekt wird in [MaMa94] genauer ausgearbeitet. Das folgende Beispiel zeigt allerdings, daß mit einer solchen Semantik die intuitive Vostellung nicht immer richtig erfaßt wird.

Beispiel 3.1 Gegeben sei die Signatur $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$ mit $\mathcal{E}^\circ = \{p^\circ\}$, $\mathcal{A} = \{a\}$ und $\mathcal{P} = \{\text{ANF}\}$. Dazu sei ein temporal stratifiziertes Programm und ein temporal flaches Ziel wie folgt gegeben:

$$\begin{array}{lll} P: & \text{ANF} \leftarrow & p^\circ \wedge a \\ & \text{ANF} \leftarrow & \neg p^\circ \wedge \neg a \\ \leftarrow \alpha: & \leftarrow \quad \text{ANF} & \end{array}$$

Hier wird offenbar gefordert, daß die Ein/Ausgabe-Leitung a stets den gleichen Wert führt wie das Zustandsbit p°. Wir wollen untersuchen, wie dies von unserer bisherigen Semantik repräsentiert wird.

Zunächst stellen wir fest, daß jede Interpretation $I^{\mathcal{A}}$ über $\mathcal{A}$ eine mögliche Ein/Ausgabefolge für $P \cup \{\leftarrow \alpha\}$ ist, denn die Erweiterung I von $I^{\mathcal{A}}$ auf Σ, die p° stets wie a und ANF stets zu wahr interpretiert, ist ein Modell für P und für α.

Andererseits gibt es keine nicht-triviale gültige Ein/Ausgabefolge für $P \cup \{\leftarrow \alpha\}$: Zu jeder nicht leeren Interpretation $I^{\mathcal{A}}$ über $\mathcal{A}$ gibt es eine Erweiterung I

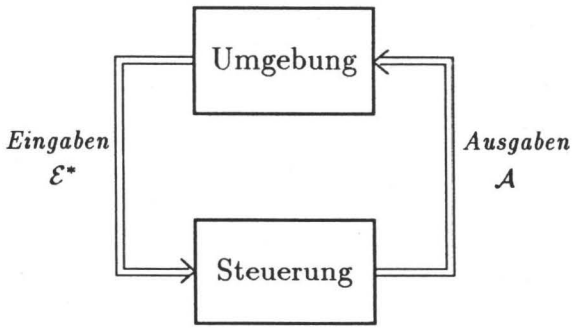

Abbildung 1: Modell eines reaktiven Systems.

auf Σ, die in allen Zuständen p° anders als a und ANF zu falsch interpretiert. Diese ist ein Modell für P, nicht aber für α.

Obwohl beide Konzepte für sich durchaus sinnvolle Deutungen enthalten mögen, entspricht keines von ihnen dem intuitiven Verständnis der durch $P \cup \{\leftarrow \alpha\}$ gegebenen Spezifikation. Das Problem besteht darin, daß in beiden Fällen nur Sprachen über $\mathcal{A} = \{a\}$ betrachtet werden. Dadurch bleibt der Zusammenhang unberücksichtigt, daß p° und a stets den gleichen Wert haben müssen.

Dieses Problem kann offenbar nur gelöst werden, wenn wir den Zustand in die Bedeutung von $P \cup \{\leftarrow \alpha\}$ einbeziehen, d.h. wenn wir an Stelle von Ein/Ausgabefolgen über $2^{\mathcal{A}}$ nunmehr Sprachen über $2^{\mathcal{E}^\circ \cup \mathcal{A}}$ betrachten. Dies führt uns zu einer etwas anderen Sichtweise temporal stratifizierter Programme, in der wir die Zustands-Prädikatensymbole in $\mathcal{E}^\circ$ als Eingaben mit Nebenbedingungen und die Symbole in $\mathcal{A}$ als reine Ausgaben auffassen.

Diese Zusammenhänge treten in einem neuen Modell für reaktive Systeme (Abb. 1) deutlich hervor. Ein reaktives System ist ein *Steuerungsprogramm*, das in ständiger Interaktion mit einer *Umgebung* steht. Die Umgebung liefert der Steuerung fortwährend Eingaben, kodiert in $\mathcal{E}^\circ$, die ihren Zustand beschreiben. Die Steuerung generiert daraufhin Ausgaben, kodiert in $\mathcal{A}$, die den Zustand der Umgebung verändern und damit die neuen Eingaben beeinflussen können. Die Regeln, nach denen die Umgebung ihren Zustand ändert, werden durch ein temporal stratifiziertes Programm beschrieben. Unser Ziel ist es dann, ein Steuerungsprogramm zu finden, das nur solche Ausgaben erzeugt, die garantieren, daß der Zustand der Umgebung stets den Anforderungen genügt.

Die Anforderungen werden durch ein temporal flaches Ziel, oder allgemeiner durch eine beliebige temporal flache Formel, beschrieben. Damit wird eine Menge von erlaubten Zuständen vorgegeben, in denen sich das System befinden darf; die zu generierende Steuerung muß sicherstellen, daß das System niemals einen Zustand außerhalb dieser Menge annimmt. Diese Art von Anforderungen wird auch als *Sicherheitsbedingung* bezeichnet.

Es fällt auf, daß alle bisher als Eingaben an das Steuerungsprogramm betrachteten Prädikatensymbole in dem neuen Modell als ein Teil des Zustandes aufgefaßt werden. Dies ist jedoch nicht weiter problematisch, weil wir davon ausgehen, daß das Steuerungsprogramm jederzeit den gesamten Zustand der Umgebung auslesen kann. Eingabe-Prädikatensymbole sind somit Zustands-Symbole ohne jegliche Übergangsbedinungen, so daß ihr Wert in jedem Zustand vollständig nicht-deterministisch ist. Im Kontext eines temporal stratifizierten Programmes wird dies dadurch ausgedrückt, daß die Definition dieser Prädikatensymbole in $\mathcal{E}^\circ$ leer ist.

Die anderen Prädikatensymbole in $\mathcal{E}^\circ$, deren Definition nicht leer ist, können je nach Sichtweise als Zustands-Symbole oder als Eingaben, die gewissen Nebenbedingungen genügen, verstanden werden. Bei der Sichtweise als Eingaben ist aber zu beachten, daß wir elementare Erfüllbarkeitsaussagen wie z.B. Satz 2.8 verlieren. Wir können nicht erwarten, daß sich jede Interpretation über der Menge $\mathcal{E}^\circ$ dieser Eingaben ohne weiteres zu einem Modell für ein temporal stratifiziertes Programm erweitern läßt. Dies wird nur noch möglich sein, wenn die Interpretation gewissen Nebenbedingungen genügt, die zudem noch von eventuell geleisteten Ausgaben abhängen können.

Ein weiteres konzeptuelles Problem besteht in der Annahme, daß das Steuerungsprogramm den Zustand der Umgebung jederzeit vollständig kennt. Diese Annahme ist in der Praxis nicht immer gerechtfertigt. In der Regel verfügt ein reaktives System über eine Anzahl von Sensoren, mit denen einige Meßwerte erfaßt werden, die zwar Rückschlüsse über den Zustand der Umgebung erlauben, ihn aber nicht unbedingt eindeutig bestimmen.

Wenn wir in dem Programm aus Beipiel 3.1 davon ausgehen können, daß der Wert des Zustandsbits p° der Steuerung jederzeit als Eingabe zur Verfügung steht, ist es nicht schwierig, eine Steuerung zu bauen, die den Anforderungen genügt — wir müssen lediglich dafür sorgen, daß die Ausgabe a stets den gleichen Wert wie die Eingabe p° annimmt. Wenn aber die Steuerung in dem Beispiel nicht auf den Wert von p° zugreifen kann, gibt es ohne zusätzliches Wissen über den Wert von p° keine Möglichkeit, das Generierungsproblem zu lösen — bei keiner Belegung der Ausgabe a wäre garantiert, daß diese den gleichen Wert wie p° hat.

Nun ist es in der Praxis häufig der Fall, daß auf den Wert bestimmter Zustandsbits zwar nicht direkt zugegriffen werden kann, diese aber gewissen Gesetzmäßigkeiten genügen, mit deren Hilfe wiederum Rückschlüsse über ihren Wert möglich sind. Im Kontext eines temporal stratifizierten Programmes würde dies dadurch ausgedrückt, daß die Definition der betreffenden Zustandsprädikatensymbole nicht leer ist, sondern die entsprechenden Bedingungen enthält.

Wenn diese Bedingungen deterministisch sind, d.h. den Folgezustand in Abhängigkeit von dem aktuellen Zustand und den Ausgaben eindeutig bestimmen, ist es möglich, diesen Zustand jederzeit zu berechnen und ihn dann wie eine ständig verfügbare Eingabe zu behandeln. Im allgemeinen Fall sind die betreffenden Regeln aber nicht vollständig deterministisch — dann sind manchmal

zusätzliche Vorkehrungen erforderlich, um sicherzustellen, daß der Wert der betreffenden Zustandsprädikatensymbole immer dann, wenn er benötigt wird, auch bekannt ist.

Beispiel 3.2 Nehmen wir an, daß das Programm aus Beispiel 3.1 um die beiden folgenden Klauseln erweitert wird:

$$\begin{array}{lll} P': & p^\circ & \leftarrow \textsc{Start} \\ & \circ p^\circ & \leftarrow a \end{array}$$

Nun ist es auch bei unbekanntem Wert von p° durchaus möglich, eine Steuerung zu bauen, die die Anforderungen erfüllt. Offenbar genügt es, als Ausgabe stets $a = 1$ zu setzen, denn dadurch erzwingen wir, daß stets auch $p^\circ = 1$ gilt.

3.2 Existenz von Modellen

Wir betrachten jetzt also temporal stratifizierte Programme über Signaturen $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$, wobei die Prädikatensymbole in $\mathcal{E}^\circ$ die Zustände und die Prädikatensymbole in $\mathcal{A}$ die Ausgaben bezeichnen. Wir wollen hier erörtern, wie die Belegung der Zustands-Prädikatensymbole aus $\mathcal{E}^\circ$ in einem Folgezustand von den Belegungen der Prädikatensymbole im aktuellen Zustand abhängt.

Wie bereits erwähnt, gelten die ursprünglich bewiesenen Erfüllbarkeitsaussagen nicht mehr, wenn wir die Eingaben eines temporal stratifizierten Programmes auf die Zustände erweitern. Vielmehr wird die Existenz von Modellen nur dann gesichert sein, wenn die Eingaben bestimmten durch das Programm beschriebenen Nebenbedingungen genügen. Um diese Zusammenhänge überhaupt formal erfassen zu können, definieren wir zunächst die Zustandsübergangsrelationen eines temporal stratifizierten Programmes. Aus ihnen werden sich die notwendigen Bedingungen für die Existenz von Modellen ergeben.

Definition 3.1 (Zustandsübergangsrelationen) Sei P ein temporal stratifiziertes Programm über $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$. Die Menge der *potentiellen Initialzustände* von P ist definiert durch

$$\begin{aligned} \textsc{Init}_P = \{\, & I^\circ \subseteq \mathcal{E}^\circ \mid \\ & \text{Für alle Klauseln } C \equiv L \leftarrow \textsc{Start} \in \textsc{Def}^0(\mathcal{E}^\circ) \text{ ist } I^\circ(L) = 1. \,\}. \end{aligned}$$

Zu $I^\mathcal{E} \subseteq \mathcal{E}^\circ \cup \mathcal{A}$ ist die Menge der *potentiellen Folgezustände* erklärt als

$$\begin{aligned} \textsc{Succ}_P(I^\mathcal{E}) = \{\, & I^\circ \subseteq \mathcal{E}^\circ \mid \\ & \text{Es gibt einen Zustand } I \text{ über } \Sigma \text{ mit } I \geq I^\mathcal{E},\ I \models \textsc{Def}(\mathcal{P}), \text{ so} \\ & \text{daß für alle Klauseln } C \equiv \circ L \leftarrow L_1 \wedge \ldots \wedge L_k \in \textsc{Def}^1(\mathcal{E}^\circ) \\ & \text{mit } I(L_1 \wedge \ldots \wedge L_k) = 1 \text{ gilt } I^\circ(L) = 1. \,\}. \end{aligned}$$

Alle Modelle eines temporal stratifizierten Programmes entsprechen seinen Zustandsübergangsrelationen, und umgekehrt sind wir bei zulässigen Programmen stets in der Lage, Zustände über $\mathcal{E}^\circ$, die den Zustandsübergangsrelationen genügen, zu Modellen zu erweitern. Die folgenden Hilfssätze präzisieren diese wichtigen Eigenschaften.

Lemma 3.1 Sei P ein temporal stratifiziertes Programm über $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$, und sei $I = (I_1, I_2, \ldots)$ ein (endliches oder unendliches) Modell für P. Dann gilt:

- $I_1|_{\mathcal{E}^\circ} \in \textsc{Init}_P$, sofern $I \neq \epsilon$.
- Für alle $k \in \mathbb{N}_0$ ist $I_{k+1}|_{\mathcal{E}^\circ} \in \textsc{Succ}_P(I_k|_{\mathcal{E}^\circ \cup \mathcal{A}})$, sofern I_k und I_{k+1} als Zustände von I existieren.

Beweis. Erstens sei I_1 der erste Zustand von $I \neq \epsilon$. Dann ist $\bar{I}_1(\textsc{Start}) = 1$. Da $I \models P$ folgt für jede Klausel $C \equiv L \leftarrow \textsc{Start} \in \textsc{Def}^0(\mathcal{E}^\circ)$ sofort $I_1(L) = 1$. Da aber das Atom von L in $\mathcal{E}^\circ$ ist, folgt auch $I_1|_{\mathcal{E}^\circ}(L) = 1$, und da dies für jede solche Klausel gilt, folgt $I_1|_{\mathcal{E}^\circ} \in \textsc{Init}_P$.

Zweitens seien I_k und I_{k+1} zwei aufeinander folgende Zustände von I. Offensichtlich ist I_k eine Erweiterung von $I_k|_{\mathcal{E}^\circ \cup \mathcal{A}}$ auf Σ mit $I_k \models \textsc{Def}(\mathcal{P})$. Sei nun $C \equiv \circ L \leftarrow L_1 \wedge \ldots \wedge L_n$ eine Klausel in $\textsc{Def}^1(\mathcal{E}^\circ)$ mit $I_k(L_1 \wedge \ldots \wedge L_n) = 1$. Da I ein Modell für P ist, gilt dann $\bar{I}_{k+1}(L) = \bar{I}_k(\circ L) = 1$ und folglich auch $I_{k+1}|_{\mathcal{E}^\circ}(L) = 1$. Somit ist $I_{k+1}|_{\mathcal{E}^\circ} \in \textsc{Succ}_P(I_k|_{\mathcal{E}^\circ \cup \mathcal{A}})$. □

Lemma 3.2 Sei P ein zulässiges, temporal stratifiziertes Programm über $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$, und sei $I_1^\circ \in \textsc{Init}_P$. Dann gibt es zu jedem Zustand $I_1^{\mathcal{E}} \geq I_1^\circ$ über $\mathcal{E} = \mathcal{E}^\circ \cup \mathcal{A}$ einen Zustand $I_1 \geq I_1^{\mathcal{E}}$ über Σ, so daß $I = (I_1)$ ein Modell für P ist.

Beweis. Nach Lemma 2.5 existiert zu $I_1^{\mathcal{E}} \subseteq \mathcal{E}$ ein Zustand $I_1 \geq I_1^{\mathcal{E}}$ über Σ, so daß $I_1 \models \textsc{Def}(\mathcal{P})$. Außerdem gilt mit $I_1^{\mathcal{E}}|_{\mathcal{E}^\circ} = I_1^\circ \in \textsc{Init}_P$ für jede Klausel $C \equiv L \leftarrow \textsc{Start} \in \textsc{Def}^0(\mathcal{E}^\circ)$ auch $I_1(L) = I_1^\circ(L) = 1$, also $I = (I_1) \models \textsc{Def}^0(\mathcal{E}^\circ)$. Und $I \models \textsc{Def}^1(\mathcal{E}^\circ)$ gilt ohnehin, da die Klauselköpfe in $\textsc{Def}^1(\mathcal{E}^\circ)$ aufgrund ihrer besonderen Gestalt $\circ L$ von jeder Fortsetzung $\bar{I}_1$ auf $F2_\Sigma$ noch beliebig interpretiert werden dürfen. □

Lemma 3.3 Sei P ein zulässiges, temporal stratifiziertes Programm über $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$, und sei $\mathcal{E} = \mathcal{E}^\circ \cup \mathcal{A}$. Weiter sei $I = (I_1, \ldots, I_n)$ ein nicht-leeres Modell für P. Dann gibt es zu jedem Zustand $I_{n+1}^{\mathcal{E}}$ über $\mathcal{E}$ mit $I_{n+1}^{\mathcal{E}}|_{\mathcal{E}^\circ} \in \textsc{Succ}_P(I_n|_{\mathcal{E}})$ auch Zustände $I_n' \geq I_n|_{\mathcal{E}}$ und $I_{n+1}' \geq I_{n+1}^{\mathcal{E}}$ über Σ, so daß $I'' = (I_1, \ldots, I_{n-1}, I_n', I_{n+1}')$ ein Modell für P ist.

Beweis. Zunächst existiert wegen $I_{n+1}^\circ = I_{n+1}^{\mathcal{E}}|_{\mathcal{E}^\circ} \in \textsc{Succ}_P(I_n|_{\mathcal{E}})$ nach Definition 3.1 ein Zustand $I_n' \geq I_n^{\mathcal{E}} = I_n|_{\mathcal{E}}$ mit $I_n' \models \textsc{Def}(\mathcal{P})$, so daß für jede Klausel $C \equiv \circ L \leftarrow L_1 \wedge \ldots \wedge L_k \in \textsc{Def}^1(\mathcal{E}^\circ)$ mit $I_n'(L_1 \wedge \ldots \wedge L_k) = 1$ gilt $I_{n+1}^\circ(L) = 1$. Nach Lemma 2.7 folgt $I' = (I_1, \ldots, I_{n-1}, I_n') \models P$.

Nach Lemma 2.5 existiert zu $I^{\mathcal{E}}_{n+1} \subseteq \mathcal{E}$ ein Zustand $I'_{n+1} \geq I^{\mathcal{E}}_{n+1}$, so daß $I'_{n+1} \models \textsc{Def}(\mathcal{P})$. Damit erklären wir $I'' = (I_1, \ldots, I_{n-1}, I'_n, I'_{n+1})$ und behaupten, daß $I'' \models P$.

Erstens gilt $I'' \models \textsc{Def}(\mathcal{P})$, da $\textsc{Def}(\mathcal{P})$ temporal flach ist, und da $I' \models P$ und $I'_{n+1} \models \textsc{Def}(\mathcal{P})$.

Zweitens folgt $I'' \models \textsc{Def}^0(\mathcal{E}^\circ)$ wegen $I \models P$, und da die Klauseln von $\textsc{Def}^0(\mathcal{E}^\circ)$ in $\bar{I}'_{n+1}$ mit $n \geq 1$ wegen $\bar{I}'_{n+1}(\textsc{Start}) = 0$ ohnehin zu wahr interpretiert werden.

Drittens werden die Klauseln von $\textsc{Def}^1(\mathcal{E}^\circ)$ wegen $I \models P$ in $I_1, \ldots, I_{n-1}$ stets zu 1 interpretiert, während für den letzten Zustand I'_{n+1} von I'' nichts zu zeigen ist. Also betrachten wir noch ihren Wert unter I'_n. Dazu sei $C \equiv \circ L \leftarrow L_1 \wedge \ldots \wedge L_k \in \textsc{Def}^1(\mathcal{E}^\circ)$ mit $I'_n(L_1 \wedge \ldots \wedge L_k) = 1$. Da hier L ein Literal über $\mathcal{E}^\circ$ ist, gilt $\bar{I}'_n(\circ L) = I'_{n+1}(L) = I^{\mathcal{E}}_{n+1}(L) = I^\circ_{n+1}(L) = 1$, da ja $I^\circ_{n+1} \in \textsc{Succ}_P(I_n|_\mathcal{E}) = \textsc{Succ}_P(I'_n|_\mathcal{E})$. □

Natürlich existiert zu jedem Zustand I° über $\mathcal{E}^\circ$ eine Erweiterung auf $\mathcal{E}$, die die Voraussetzungen der beiden obigen Lemmas erfüllt. Es genügt ja, die noch undefinierten Prädikatensymbole in $\mathcal{A}$ mit 0 zu belegen. Damit erhalten wir die folgenden nützlichen Abschwächungen der beiden Lemmas.

Korollar 3.4 Sei P ein zulässiges, temporal stratifiziertes Programm über $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$, und sei $I^\circ_1 \in \textsc{Init}_P$. Dann gibt es einen Zustand $I_1 \geq I^\circ_1$ über Σ, so daß $I = (I_1)$ ein Modell für P ist.

Korollar 3.5 Sei P ein zulässiges, temporal stratifiziertes Programm über $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$, und sei $\mathcal{E} = \mathcal{E}^\circ \cup \mathcal{A}$. Weiter sei $I = (I_1, \ldots, I_n)$ ein nicht-leeres Modell für P. Dann gibt es zu jedem Zustand $I^\circ_{n+1} \in \textsc{Succ}_P(I_n|_\mathcal{E})$ auch Zustände $I'_n \geq I_n|_\mathcal{E}$ und $I'_{n+1} \geq I^\circ_{n+1}$ über Σ, so daß $I' = (I_1, \ldots, I_{n-1}, I'_n, I'_{n+1})$ ein Modell für P ist.

Weiter benötigen wir das Konzept der Menge der erreichbaren Zustände eines temporal stratifizierten Programmes, das sich aus der induktiven Fortsetzung der Zustandsübergangsrelationen ergibt. Ein erreichbarer Zustand ist ein Zustand über $\mathcal{E}^\circ$, zu dem es eine Folge von Ein/Ausgaben gibt, unter denen eine Abarbeitung des Programmes in diesen Zustand führen kann.

Definition 3.2 (Erreichbare Zustände) Es sei P ein temporal stratifiziertes Programm über $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$. Dann wird die Menge $\textsc{States}_P$ der *erreichbaren Zustände* von P induktiv definiert vermöge

$$\begin{aligned}
\textsc{States}^1_P &= \textsc{Init}_P; \\
\textsc{States}^{k+1}_P &= \bigcup \{ \textsc{Succ}_P(I^\mathcal{E}) \mid I^\mathcal{E} \subseteq \mathcal{E}^\circ \cup \mathcal{A},\ I^\mathcal{E}|_{\mathcal{E}^\circ} \in \textsc{States}^k_P \}; \\
\textsc{States}_P &= \bigcup_{k=1}^{\infty} \textsc{States}^k_P .
\end{aligned}$$

Satz 3.6 Sei P ein zulässiges und temporal stratifiziertes Programm über $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$. Dann gilt folgende Charakterisierung der Menge der erreichbaren Zustände von P:

$$\begin{aligned}\textsc{States}_P = \{\, I^\circ \subseteq \mathcal{E}^\circ \mid \\ \text{Es gibt ein } k \in \mathbb{N}_0 \text{ und ein Modell } I = (I_1, \ldots, I_k) \text{ für } P, \\ \text{so daß } I_k \geq I^\circ. \,\}.\end{aligned}$$

Beweis. Wir zeigen die beiden Inklusionen der Gleichung

$$\textsc{States}_P^k = \{\, I^\circ \subseteq \mathcal{E}^\circ \mid \text{Es gibt ein Modell } I = (I_1, \ldots, I_k) \text{ für } P \text{ mit } I_k \geq I^\circ. \,\},$$

jeweils durch Induktion über k. Zusammen mit der Definition von $\textsc{States}_P$ folgt daraus direkt die Behauptung.

Zunächst sei $I = (I_1, \ldots, I_k) \models P$. Wir zeigen $I_k|_{\mathcal{E}^\circ} \in \textsc{States}_P^k$. Im Induktionsanfang ist $I = (I_1)$, und nach Lemma 3.1 folgt $I_1|_{\mathcal{E}^\circ} \in \textsc{Init}_P = \textsc{States}_P^1$. Als Induktionsvoraussetzung sei nun $I_k|_{\mathcal{E}^\circ} \in \textsc{States}_P^k$. Da $I \models P$, folgt wiederum nach Lemma 3.1 auch $I_{k+1}|_{\mathcal{E}^\circ} \in \textsc{Succ}_P(I_k|_{\mathcal{E}})$, wobei $\mathcal{E} = \mathcal{E}^\circ \cup \mathcal{A}$. Dann folgt mit der Induktionsvoraussetzung unmittelbar $I_{k+1}|_{\mathcal{E}^\circ} \in \textsc{States}_P^{k+1}$.

Nun sei umgekehrt $I_k^\circ \in \textsc{States}_P^k$. Wir zeigen die Existenz eines Modells $I = (I_1, \ldots, I_k)$ für P mit $I_k \geq I_k^\circ$. Im Induktionsanfang sei $I_1^\circ \in \textsc{States}_P^1 = \textsc{Init}_P$. Dazu sichert Korollar 3.4 die Existenz des gewünschten Zustandes $I_1 \geq I_1^\circ$, so daß $I = (I_1) \models P$. Als Induktionsvoraussetzung existiere nun für jedes $I_k^\circ \in \textsc{States}_P^k$ ein Modell $I = (I_1, \ldots, I_k) \models P$. Wir betrachten $I_{k+1}^\circ \in \textsc{States}_P^{k+1}$. Nach Definition gibt es einen Zustand $I_k^\mathcal{E}$ über $\mathcal{E} = \mathcal{E}^\circ \cup \mathcal{A}$ mit $I_{k+1}^\circ \in \textsc{Succ}_P(I_k^\mathcal{E})$ für ein $I_k^\mathcal{E}|_{\mathcal{E}^\circ} \in \textsc{States}_P^k$. Nach Induktionsvoraussetzung existiert das Modell $I = (I_1, \ldots, I_k) \models P$ mit $I_k \geq I_k^\mathcal{E}|_{\mathcal{E}^\circ}$. Nach Lemma 2.5 gibt es dann einen Zustand $I_k' \geq I_k^\mathcal{E}$ über Σ, so daß $I_k' \models \textsc{Def}(\mathcal{P})$. Wegen $I_k'|_{\mathcal{E}^\circ} = I_k|_{\mathcal{E}^\circ}$ gilt nach Lemma 2.7 auch $I' = (I_1, \ldots, I_{k-1}, I_k') \models P$. Schließlich existieren nach Korollar 3.5 zu I' und $I_{k+1}^\circ \in \textsc{Succ}_P(I_k'|_\mathcal{E})$ Zustände $I_k'' \geq I_k'|_\mathcal{E} = I_k^\mathcal{E}$ und $I_{k+1}'' \geq I_{k+1}^\circ$, so daß $I'' = (I_1, \ldots, I_{k-1}, I_k'', I_{k+1}'')$ das gewünschte Modell für P ist. □

3.3 Ein/Ausgabe-Zuordnungen

Jetzt betrachten wir ein temporal stratifiziertes Programm P über $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$ und dazu ein temporal flaches Ziel $\leftarrow \alpha$. Wir interessieren uns dafür, ob es eine Zuordnung gibt, die jedem Zustand I° über $\mathcal{E}^\circ$ einen Zustand $I^\mathcal{A}$ über $\mathcal{A}$ zuordnet, so daß jede Erweiterung des resultierenden Zustandes auf Σ, die Modell für P ist, die durch $\leftarrow \alpha$ spezifizierten Anforderungen erfüllt, ebenso wie jeder potentielle Folgezustand. Allgemeiner können wir nach allen Zuständen über $\mathcal{A}$ fragen, die diese Bedingungen erfüllen.

Einen solchen Zusammenhang könnten wir durch eine Relation $X \subseteq 2^{\mathcal{E}^\circ} \times 2^\mathcal{A}$ beschreiben, oder aber mit $\mathcal{E} = \mathcal{E}^\circ \cup \mathcal{A}$ äquivalent durch eine Teilmenge $X \subseteq 2^\mathcal{E}$. Eine solche Teilmenge nennen wir eine *Ein/Ausgabe-Zuordnung.*

Definition 3.3 Sei Σ eine Menge von Prädikatensymbolen. Eine *Ein/Ausgabe-Zuordnung* über Σ ist eine Relation $X \subseteq 2^{\Sigma}$.

Damit ist eine Ein/Ausgabe-Zuordnung X über Σ eine Menge von Zuständen $I \subseteq \Sigma$, d.h. wir können X auffassen als Teilmenge $X \subseteq 2^{\Sigma}$ — denn 2^{Σ} ist die Menge aller Zustände über Σ — oder aber äquivalent als Element $X \in 2^{2^{\Sigma}}$.

Eine Ein/Ausgabe-Zuordnung für ein temporal stratifiziertes Programm über $\Sigma = (\mathcal{E}^{\circ}, \mathcal{A}, \mathcal{P})$ ist eine Ein/Ausgabe-Zuordnung über $\mathcal{E} = \mathcal{E}^{\circ} \cup \mathcal{A}$, etwa $X \in 2^{2^{\mathcal{E}}}$. Wir interessieren uns besonders für solche Ein/Ausgabe-Zuordnungen, die die gewünschte Eigenschaft haben, daß alle von ihnen zugelassenen Ausgaben die Erfüllung der Anforderungen garantieren. Diese Ein/Ausgabe-Zuordnungen nennen wir *sicher* und formalisieren die von ihnen geforderten Eigenschaften in der folgenden Definition.

Definition 3.4 (Sichere Ein/Ausgabe-Zuordnung) Sei P ein temporal stratifiziertes Programm über $\Sigma = (\mathcal{E}^{\circ}, \mathcal{A}, \mathcal{P})$, und sei $\mathcal{E} = \mathcal{E}^{\circ} \cup \mathcal{A}$. Weiter sei $\leftarrow \alpha$ ein temporal flaches Ziel. Eine Ein/Ausgabe-Zuordnung $H \subseteq 2^{\mathcal{E}}$ für P heißt *sichere Ein/Ausgabe-Zuordnung* für $P \cup \{\leftarrow \alpha\}$, wenn für alle Zustände $I^{\mathcal{E}} \in H$ die folgenden beiden Bedingungen erfüllt sind:

(SI1) Für alle Zustände $I \geq I^{\mathcal{E}}$ über Σ mit $I \models \textsc{Def}(\mathcal{P})$ gilt auch $I \models \alpha$.

(SI2) $\textsc{Succ}_P(I^{\mathcal{E}}) \subseteq H|_{\mathcal{E}^{\circ}}$[2].

Bedingung (SI1) garantiert, daß alle Zustände einer sicheren Ein/Ausgabe-Zuordnung H den durch das Ziel $\leftarrow \alpha$ gestellten Anforderungen genügen. Die zweite Bedingung (SI2) stellt zudem sicher, daß es keine potentiellen Folgezustände eines Zustandes in H gibt, die außerhalb von H liegen und damit die Anforderungen verletzten könnten. Offenbar entspricht jede Lösung des durch $P \cup \{\leftarrow \alpha\}$ spezifizierten Generierungsproblems einer sicheren Ein/Ausgabe-Zuordnung.

Beispiel 3.3 Jedes Paar $P \cup \{\leftarrow \alpha\}$ aus Programm und Ziel hat die leere Menge als sichere Ein/Ausgabe-Zuordnung, denn sie erfüllt trivialerweise beide Bedingungen der Definition.

Beispiel 3.4 Wir betrachten nochmals das Programm P und Ziel $\leftarrow \alpha$ aus Beispiel 3.1. Hier haben wir neben der leeren Menge die sichere Ein/Ausgabe-Zuordnung $H = \{ \{p^{\circ}, a\}, \{\neg p^{\circ}, \neg a\} \}$. Offensichtlich gilt erstens für jede Erweiterung I eines der aufgelisteten Zustände von H, die Modell für P ist, auch $I(\textsc{Anf}) = 1$ und damit $I \models \alpha$; da zweitens $H|_{\mathcal{E}^{\circ}}$ alle Zustände über $\mathcal{E}^{\circ} = \{p^{\circ}\}$ enthält, kann auch jeder potentielle Folgezustand eines Zustandes in H zu einem Zustand in H erweitert werden.

[2]Dabei erklären wir $H|_{\mathcal{E}^{\circ}} = \{ I^{\mathcal{E}}|_{\mathcal{E}^{\circ}} \mid I^{\mathcal{E}} \in H \}$.

Andererseits ist $H' = \{\{p^\circ, a\}\}$ keine sichere Ein/Ausgabe-Zuordnung für $P \cup \{\leftarrow \alpha\}$, da der potentielle Folgezustand $\{\neg p^\circ\} \in \textsc{Succ}_P(\{p^\circ, a\})$ nicht zu einem Zustand in H' erweitert werden kann.

Die Tatsache, daß die leere Menge immer eine sichere Ein/Ausgabe-Zuordnung für $P \cup \{\leftarrow \alpha\}$ ist, zeigt, daß nicht jede sichere Ein/Ausgabe-Zuordnung interessant ist als Lösung für das durch $P \cup \{\leftarrow \alpha\}$ beschriebene Generierungsproblem. Vielmehr benötigen wir ein Konzept, das so etwas wie die „größte" sichere Ein/Ausgabe-Zuordnung beschreibt.

Es ist möglich, sichere Ein/Ausgabe-Zuordnungen durch eine Fixpunkteigenschaft zu charakterisieren, aus der sich eine geeignete Charakterisierung der Lösungen des Generierungsproblems ergibt. Wir definieren zunächst das betreffende Funktional und zeigen, daß es genau die sicheren Ein/Ausgabe-Zuordnungen als Fixpunkte hat.

Definition 3.5 Sei P ein temporal stratifiziertes Programm über $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$, und sei $\mathcal{E} = \mathcal{E}^\circ \cup \mathcal{A}$. Dann wird das P *zugeordnete Funktional* $T_P: 2^{2^{\mathcal{E}}} \to 2^{2^{\mathcal{E}}}$ definiert durch

$$T_P(X) = \{ I^{\mathcal{E}} \in X \mid \textsc{Succ}_P(I^{\mathcal{E}}) \subseteq X|_{\mathcal{E}^\circ} \}.$$

Zu einer Ein/Ausgabe-Zuordnung X beschreibt $T_P(X)$ die Menge der Zustände in X, deren potentielle Folgezustände alle wieder zu Zuständen in X erweitert werden können. Deshalb nennen wir $T_P(X)$ auch die *Menge der sicheren Zustände* bezüglich X. In gewisser Weise entspricht dies der Eigenschaft (SI2) aus Definition 3.4, die ja fordert, daß die potentiellen Folgezustände aller Zustände in einer sicheren Ein/Ausgabe-Zuordnung H wieder zu Zuständen in H erweitert werden können. Allerdings ist es möglich, daß $T_P(X)$ Zustände enthält, deren Folgezustände zwar in X enthalten sind, nicht aber in $T_P(X)$. Daher müssen wir die Anwendung des Operators T_P iterieren, bis wir zu einem Fixpunkt gelangen.

Zuvor müssen wir aber noch eine zusätzliche Bedingung betrachten, um sicherzustellen, daß die in Betracht zu ziehenden Fixpunkte von T_P neben der Bedingung (SI2) auch die Bedingung (SI1) aus Definition 3.4 erfüllen.

Definition 3.6 Sei P ein temporal stratifiziertes Programm über $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$ und $\mathcal{E} = \mathcal{E}^\circ \cup \mathcal{A}$. Weiter sei $\leftarrow \alpha$ ein temporal flaches Ziel. Dann definieren wir die Menge der *gültigen Zustände* für $P \cup \{\leftarrow \alpha\}$ als

$$H_{P \cup \{\leftarrow \alpha\}} = \{ I^{\mathcal{E}} \subseteq \mathcal{E} \mid \text{Für alle } I \geq I^{\mathcal{E}} \text{ mit } I \models \textsc{Def}(\mathcal{P}) \text{ gilt } I \models \alpha. \}.$$

Offensichtlich gilt für das Funktional T_P stets $T_P(X) \subseteq X$, so daß wir es auch als Abbildung

$$T_P: 2^{H_{P \cup \{\leftarrow \alpha\}}} \to 2^{H_{P \cup \{\leftarrow \alpha\}}}$$

auffassen können. Für diese Einschränkung interessieren wir uns besonders, weil sie die entscheidende Fixpunkt-Eigenschaft besitzt.

Definition 3.7 Sei P ein temporal stratifiziertes Programm über $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$ und $\mathcal{E} = \mathcal{E}^\circ \cup \mathcal{A}$. Weiter sei $\leftarrow \alpha$ ein temporal flaches Ziel. Dann nennen wir die Einschränkung von T_P auf die Teilmengen von $H_{P\cup\{\leftarrow\alpha\}}$,

$$T_{P\cup\{\leftarrow\alpha\}} = T_P|_{\mathcal{H}}, \text{ wobei } \mathcal{H} = 2^{H_{P\cup\{\leftarrow\alpha\}}},$$

das $P \cup \{\leftarrow \alpha\}$ *zugeordnete Funktional.*

Nun zeigt sich, daß sichere Ein/Ausgabe-Zuordnungen für $P \cup \{\leftarrow \alpha\}$ stets Fixpunkte von $T_{P\cup\{\leftarrow\alpha\}}$ sind. Damit erhalten wir eine mathematische Klassifikation der sicheren Ein/Ausgabe-Zuordnungen, die als Grundlage für die folgenden Betrachtungen dient.

Satz 3.7 Sei P ein temporal stratifiziertes Programm über $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$, und $\mathcal{E} = \mathcal{E}^\circ \cup \mathcal{A}$. Weiter sei $\leftarrow \alpha$ ein temporal flaches Ziel. Genau dann ist $X \subseteq H_{P\cup\{\leftarrow\alpha\}}$ eine sichere Ein/Ausgabe-Zuordnung für $P \cup \{\leftarrow \alpha\}$, wenn X ein Fixpunkt von T_P ist, d.h. wenn $T_P(X) = X$.

Beweis. Sei $X \subseteq H_{P\cup\{\leftarrow\alpha\}}$ eine sichere Ein/Ausgabe-Zuordnung für $P \cup \{\leftarrow \alpha\}$. Da trivialerweise $T_P(X) \subseteq X$ gilt, genügt es, $X \subseteq T_P(X)$ zu beweisen. Nun gilt für alle $I^{\mathcal{E}} \in X$ nach Eigenschaft (SI2) aus Definition 3.4 auch $\textsc{Succ}_P(I^{\mathcal{E}}) \subseteq X|_{\mathcal{E}^\circ}$, und damit $I^{\mathcal{E}} \in T_P(X)$. Das aber bedeutet $X \subseteq T_P(X)$.

Sei nun umgekehrt $X \subseteq H_{P\cup\{\leftarrow\alpha\}}$ ein Fixpunkt von T_P, und sei $I^{\mathcal{E}} \in X$. Wir zeigen, daß $I^{\mathcal{E}}$ die beiden Bedingungen aus Definition 3.4 erfüllt. Da $I^{\mathcal{E}} \in H_{P\cup\{\leftarrow\alpha\}}$, gilt erstens für alle $I \geq I^{\mathcal{E}}$ mit $I \models \textsc{Def}(\mathcal{P})$ auch $I \models \alpha$. Zweitens ist $\textsc{Succ}_P(I^{\mathcal{E}}) \subseteq X|_{\mathcal{E}^\circ}$, da $I^{\mathcal{E}} \in X = T_P(X) = \{ I^{\mathcal{E}} \in X \mid \textsc{Succ}_P(I^{\mathcal{E}}) \subseteq X|_{\mathcal{E}^\circ} \}$. □

Diese Erkenntnis allein sagt jedoch noch nicht viel über die Existenz oder Berechenbarkeit von sicheren Ein/Ausgabe-Zuordnungen aus. Wir stellen aber außerdem fest, daß T_P monoton[3] ist, so daß wir die bekannten Fixpunktsätze anwenden können.

Lemma 3.8 Sei P ein temporal stratifiziertes Programm über $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$, und sei $\mathcal{E} = \mathcal{E}^\circ \cup \mathcal{A}$. Dann ist die Abbildung $T_P: 2^{2^{\mathcal{E}}} \to 2^{2^{\mathcal{E}}}$ monoton.

Beweis. Seien $X, Y \subseteq 2^{\mathcal{E}}$ mit $X \subseteq Y$, und sei $I^{\mathcal{E}} \in T_P(X) = \{ I^{\mathcal{E}} \in X \mid \textsc{Succ}_P(I^{\mathcal{E}}) \subseteq X|_{\mathcal{E}^\circ} \}$. Dann folgt sofort $I^{\mathcal{E}} \in X \subseteq Y$. Außerdem haben wir $\textsc{Succ}_P(I^{\mathcal{E}}) \subseteq X|_{\mathcal{E}^\circ} \subseteq Y|_{\mathcal{E}^\circ}$. Somit folgt $I^{\mathcal{E}} \in T_P(Y)$. □

Nach dem klassischen Ergebnis von Tarski [Tars55] folgt nun, daß das Funktional $T_{P\cup\{\leftarrow\alpha\}}$ einen *größten* Fixpunkt besitzt, der damit auch die größte sichere Ein/Ausgabe-Zuordnung liefert. Da $H_{P\cup\{\leftarrow\alpha\}}$ endlich ist, folgt zudem noch, daß dieser größte Fixpunkt durch eine endliche Iteration mittels T_P berechnet werden kann.

[3] $F : 2^A \to 2^B$ heißt *monoton*, wenn für alle $x, y \subseteq A$ mit $x \subseteq y$ stets $F(x) \subseteq F(y)$ gilt.

Theorem 3.9 *Sei P ein temporal stratifiziertes Programm über $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$ und $\mathcal{E} = \mathcal{E}^\circ \cup \mathcal{A}$. Weiter sei $\leftarrow \alpha$ ein temporal flaches Ziel. Dann besitzt die Abbildung $T_{P\cup\{\leftarrow\alpha\}}: H_{P\cup\{\leftarrow\alpha\}} \to H_{P\cup\{\leftarrow\alpha\}}$ einen größten Fixpunkt $\hat{H}_{P\cup\{\leftarrow\alpha\}}$, der wie folgt charakterisiert werden kann:*

$$\begin{aligned}\hat{H}_{P\cup\{\leftarrow\alpha\}} &= \mathrm{gfp}(T_{P\cup\{\leftarrow\alpha\}}) \\ &= \mathrm{lub}\{\, I^{\mathcal{E}} \subseteq \mathcal{E} \mid \\ &\qquad I^{\mathcal{E}} \text{ ist sichere Ein/Ausgabe-Zuordnung für } P \cup \{\leftarrow \alpha\}.\,\}\end{aligned}$$

Beweis. Folgt nach den Ergebnissen aus [Tars55] direkt aus Satz 3.7 und Lemma 3.8. □

$\hat{H}_{P\cup\{\leftarrow\alpha\}}$, der größte Fixpunkt von $T_{P\cup\{\leftarrow\alpha\}}$, ist also die Vereinigung aller sicheren Ein/Ausgabe-Zuordnungen für $P \cup \{\leftarrow \alpha\}$ und bezeichnet daher die *allgemeinste* sichere Ein/Ausgabe-Zuordnung für $P \cup \{\leftarrow \alpha\}$. Aus diesem Grund können wir diesen Fixpunkt $\hat{H}_{P\cup\{\leftarrow\alpha\}}$ als die *Bedeutung* des Paares $P \cup \{\leftarrow \alpha\}$ betrachten — die Tragweite dieses Konzepts entspricht ungefähr der des kleinsten Herbrand-Modells beim logischen Programmieren [Lloyd84].

Jede Lösung des durch $P \cup \{\leftarrow \alpha\}$ spezifizierten Generierungsproblems ist in $\hat{H}_{P\cup\{\leftarrow\alpha\}}$ enthalten. Natürlich ist die durch $\hat{H}_{P\cup\{\leftarrow\alpha\}}$ beschriebene allgemeinste Lösung in der Regel hochgradig nicht-deterministisch. Um eine deterministische Lösung zu erhalten, genügt es jedoch, in allen Fällen, bei denen $\hat{H}_{P\cup\{\leftarrow\alpha\}}$ im gleichen Zustand mehrere verschiedene Ausgaben zuläßt, eine beliebige dieser Ausgaben auszuwählen.

Theorem 3.10 *Sei P ein temporal stratifiziertes Programm über $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$ und $\mathcal{E} = \mathcal{E}^\circ \cup \mathcal{A}$. Weiter sei $\leftarrow \alpha$ ein temporal flaches Ziel. Dann gibt es eine natürliche Zahl $n \in \mathbb{N}_0$, so daß der größte Fixpunkt $\hat{H}_{P\cup\{\leftarrow\alpha\}}$ von T_P ausgedrückt werden kann als*

$$\hat{H}_{P\cup\{\leftarrow\alpha\}} = T_P^n(H_{P\cup\{\leftarrow\alpha\}}).$$

Beweis. Folgt direkt aus Theorem 3.9, da $2^{H_{P\cup\{\leftarrow\alpha\}}}$ endlich ist. □

Diese Aussage ist die Grundlage für alle möglichen Implementierungen des Generierungsverfahrens, weil sie sicherstellt, daß die allgemeinste Lösung $\hat{H}_{P\cup\{\leftarrow\alpha\}}$ des Generierungsproblems durch eine endliche Iteration des Operators T_P effektiv berechnet werden kann.

3.4 Auflösbarkeit

Unser eigentliches Ziel besteht ja darin, zu einem temporal stratifizierten Programm über $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$ und einem temporal flachen Ziel ein Steuerungsprogramm zu konstruieren, das zu jeder Kombination von Eingaben über $\mathcal{E}^\circ$ eine Kombination von Ausgaben über $\mathcal{A}$ berechnet, so daß die Anforderungen immer erfüllt werden. Im Kontext unserer Logik-Programme liegt es nahe, diese

Steuerung durch eine Klauselmenge zu beschreiben. Dies veranlaßt uns zu der folgenden Definition.

Definition 3.8 Sei Σ ein Alphabet von Prädikatensymbolen, und seien $\Sigma_1, \Sigma_2 \subseteq \Sigma$. Eine F2-Programm R über $\Sigma_1 \cup \Sigma_2$ heißt *zulässige Regelmenge vom Typ* $\Sigma_1 \leftarrow \Sigma_2$, wenn es die folgenden Bedingungen erfüllt:

- R ist zulässig.
- R ist temporal flach.
- R ist hierarchisch.
- Alle in Klauselköpfen aus R auftretenden Atome sind in $\mathcal{A}$.

Eine solche zulässige Regelmenge R vom Typ $\mathcal{A} \leftarrow \mathcal{E}$ hat die Eigenschaft, daß es leicht möglich ist, durch Auswertung der Regeln zu einer gegebenen Belegung der Prädikatensymbole in $\mathcal{E}$ eine Belegung der Prädikatensymbole in $\mathcal{A}$ zu berechnen, so daß ein Modell für R entsteht. Da gleichzeitig Nicht-Determinismus zugelassen ist, kann sie auch allgemeinere Relationen beschreiben.

Wenn wir zu einem temporal stratifizierten Programm über $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$ eine Regelmenge vom Typ $\mathcal{A} \leftarrow \mathcal{E}^\circ$ hinzunehmen, entsteht ein neues temporal stratifiziertes Programm über einer veränderten Signatur Σ', in der alle Ein/Ausgabe-Prädikatensymbole aus der Menge $\mathcal{A}$ der ursprünglichen Signater Σ jetzt als Hilfsprädikate in der Menge $\mathcal{P}$ der neuen Signatur Σ' auftreten. Später ist es dann unser Ziel, besondere Regelmengen zu finden, die sicherstellen, daß dieses neue Programm bestimmte Anforderungen erfüllt.

Lemma 3.11 Sei P ein zulässiges, temporal stratifiziertes Programm über $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$, und sei R eine zulässige Regelmenge vom Typ $\mathcal{A} \leftarrow \mathcal{E}^\circ$. Dann ist $P \cup R$ ein zulässiges, temporal stratifiziertes Programm über $\Sigma' = (\mathcal{E}^\circ, \emptyset, \mathcal{A} \cup \mathcal{P})$.

Beweis. Wir sehen sofort, daß $\textsc{Def}(\mathcal{E}^\circ)$ $\circ$-abgestuft ist, und daß $\textsc{Def}(\mathcal{A} \cup \mathcal{P})$ temporal flach ist. Trivialerweise ist $\textsc{Def}(\emptyset) = \emptyset$. Und da P und R zulässig sind, gilt dies auch für $P \cup R$.

Zu zeigen bleibt lediglich, daß $\textsc{Def}(\mathcal{A} \cup \mathcal{P}) = \textsc{Def}(\mathcal{P}) \cup R$ hierarchisch ist. Da in diesen Klauseln keine $\circ$-Operatoren auftreten, ist dies gleichbedeutend mit der Eigenschaft der temporalen Hierachie. Dazu zeigen wir die Eigenschaft (H2) aus Satz 2.4. Da nach Voraussetzung $\textsc{Def}(\mathcal{P})$ und R hierarchisch sind, existieren Niveauabbildungen

$$N_{\mathcal{P}}: \quad \Sigma \to \mathbb{N}_0 \quad \text{zu } \textsc{Def}(\mathcal{P}),$$
$$N_R: \quad \Sigma \to \mathbb{N}_0 \quad \text{zu } R.$$

Dann setzen wir

$$m = \max\{ N_R(p) \mid p \in \mathcal{A} \}$$

und erklären damit eine neue Niveauabbildung für $\text{DEF}(\mathcal{P}) \cup R$ vermöge

$$N\colon \Sigma \to \mathbb{N}_0;\ p \mapsto \begin{cases} 0, & \text{falls } p \in \mathcal{E}^\circ, \\ N_R(p) + 1, & \text{falls } p \in \mathcal{A}, \\ N_{\mathcal{P}}(p) + m + 2, & \text{falls } p \in \mathcal{P}. \end{cases}$$

Zu zeigen ist, daß für beliebige $p, q \in \Sigma$ mit $p \sqsubset^0 q$ stets $N(p) > N(q)$ gilt. Seien also $p, q \in \Sigma$ mit $p \sqsubset^0 q$. Zunächst einmal folgern wir $p \notin \mathcal{E}^\circ$, da die Definition von $\mathcal{E}^\circ$ in $\text{DEF}(\mathcal{P}) \cup R$ leer ist und daher zu $p \in \mathcal{E}^\circ$ kein $q \in \Sigma$ mit $p \sqsubset^0 q$ existieren kann. Dann unterscheiden wir die folgenden drei Fälle:

Fall 1: $q \in \mathcal{E}^\circ$. Da $p \notin \mathcal{E}^\circ$ folgt sofort $N(p) > 0 = N(q)$.

Fall 2: $q \in \mathcal{A}$. Falls $p \in \mathcal{A}$, so haben wir $N(p) = N_R(p) + 1 > N_R(q) + 1 = N(q)$, da N_R eine Niveauabbildung für R ist. Sonst ist $p \in \mathcal{P}$, und wir erhalten $N(p) = N_{\mathcal{P}}(p) + m + 2 \geq N_{\mathcal{P}}(p) + N_R(q) + 1 + 1 = N_{\mathcal{P}}(p) + N(q) + 1 > N(q)$.

Fall 3: $q \in \mathcal{P}$. Dann ist zunächst $p \notin \mathcal{A}$, da die Klauseln der Definition von $\mathcal{A}$ alle in der zulässigen Regelmenge R vom Typ $\mathcal{A} \leftarrow \mathcal{E}^\circ$ sind und somit nur Prädikatensymbole aus $\mathcal{E}^\circ \cup \mathcal{A}$ enthalten. Also ist $p \in \mathcal{P}$, und es folgt $N(p) = N_{\mathcal{P}}(p) + m + 2 > N_{\mathcal{P}}(q) + m + 2 = N(q)$. □

Definition 3.9 (Auflösbarkeit) Sei P ein temporal stratifiziertes Programm über $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$, und sei $\leftarrow \alpha$ ein temporal flaches Ziel. Das Paar $P \cup \{\leftarrow \alpha\}$ heißt *auflösbar*, wenn es eine zulässige Regelmenge R vom Typ $\mathcal{A} \leftarrow \mathcal{E}^\circ$ gibt, so daß jedes Modell für $P \cup R$ auch ein Modell für α ist.

Beispiel 3.5 Das Programm und Ziel in Beispiel 3.1 erweist sich als auflösbar — eine geeignete Regelmenge hat die folgende Gestalt:

$$\begin{array}{rrcl} R\colon & a & \leftarrow & p^\circ \\ & \neg a & \leftarrow & \neg p^\circ \end{array}$$

Bisher haben wir die Bedeutung des Paares $P \cup \{\leftarrow \alpha\}$ als Relation $\hat{H}_{P \cup \{\leftarrow \alpha\}} \subseteq 2^{\mathcal{E}^\circ \cup \mathcal{A}}$ definiert, aber nun interessieren wir uns für Regelmengen, die das Verhältnis zwischen Ein- und Ausgaben konstruktiv beschreiben. Deshalb suchen wir eine Möglichkeit, Relationen und Regelmengen zueinander in Beziehung zu setzen. Diese finden wir in dem folgenden Satz, der sogar eine Möglichkeit zeigt, wie eine Regelmenge aus einer Relation konstruiert werden kann.

Satz 3.12 Sei Σ ein endliches Alphabet, das sich als disjunkte Vereinigung $\Sigma = \mathcal{A} \cup \mathcal{E}$ ergibt, und sei $X \subseteq 2^\Sigma$ eine Ein/Ausgabe-Zuordnung über Σ. Dann gibt es eine zulässige Regelmenge R_X vom Typ $\mathcal{A} \leftarrow \mathcal{E}$, so daß für alle Zustände $I \subseteq \Sigma$ mit $I|_{\mathcal{E}} \in X|_{\mathcal{E}}$ die folgende Äquivalenz gilt.

$$I \models R_X \iff I \in X \tag{1}$$

Diese Regelmenge R_X kann effektiv konstruiert werden.

Bei diesem Satz ist besonders auf die Nebenbedingung $I|_{\mathcal{E}} \in X|_{\mathcal{E}}$ zu achten. Durch Regeln, die die Belegungen der Prädikatensymbole in $\mathcal{A}$ festlegen, können nur solche Interpretationen über $\mathcal{E}^\circ$ zu sicheren Ein/Ausgabe-Zuordnungen erweitert werden, die die Prädikatensymbole in $\mathcal{E}^\circ$ bereits in einer geeigneten Weise interpretieren. Diese Bedingung muß später durch zusätzliche Vorkehrungen — in unserem Falle die Fixpunktiteration — gesichert werden.

Beweis. Nehmen wir an, $\mathcal{A}$ habe die Gestalt $\mathcal{A} = \{a_1, \ldots, a_n\}$. Wir erklären $\Sigma^k = \mathcal{E} \cup \{a_1, \ldots, a_k\}$ und bezeichnen für $I \subseteq \Sigma^k$ mit $I \uplus a_{k+1}$ den Zustand $J = I \cup \{a_{k+1}\}$ über Σ^{k+1}. Entsprechend bezeichnet $I \uplus \neg a_{k+1}$ den gleichen Zustand wie I, aber über Σ^{k+1}. Weiter definieren wir zu $I \subseteq \Sigma^k$ den Klauselrumpf zu I,

$$\mathrm{cl}(I) = \bigwedge_{p \in I} p \wedge \bigwedge_{q \in \Sigma^k \setminus I} \neg q.$$

Hier ist zu beachten, daß stets gilt $I(\mathrm{cl}(I)) = 1$, und für alle Zustände $J \subseteq \Sigma$ mit $J(\mathrm{cl}(I)) = 1$ folgt immer $J \geq I$.

Damit definieren wir die Regelmenge R_X:

$$\begin{aligned} R_X^k &= \{\, a_k \leftarrow \mathrm{cl}(I) \mid I \subseteq \Sigma^{k-1}, I \uplus a_k \in X|_{\Sigma^k}, I \uplus \neg a_k \notin X|_{\Sigma^k} \,\} \\ &\cup \{\, \neg a_k \leftarrow \mathrm{cl}(I) \mid I \subseteq \Sigma^{k-1}, I \uplus \neg a_k \in X|_{\Sigma^k}, I \uplus a_k \notin X|_{\Sigma^k} \,\} \\ R_X &= \bigcup_{k=1}^{n} R_X^k \end{aligned}$$

Offenbar ist R_X ein zulässiges, temporal flaches, hierarchisches Programm über $\mathcal{A} \cup \mathcal{E}$, dessen Klauselköpfe alle in $\mathcal{A}$ sind. Somit ist R_X eine zulässige Regelmenge vom Typ $\mathcal{A} \leftarrow \mathcal{E}$.

Sei nun $I \subseteq \Sigma$ mit $I|_{\mathcal{E}} \in X|_{\mathcal{E}}$.

Zunächst gelte $I \models R_X$. Wir zeigen durch Induktion über k, daß stets gilt

$$I|_{\Sigma^k} \in X|_{\Sigma^k}.$$

Im Induktionsanfang für $k = 0$ haben wir $\Sigma^0 = \mathcal{E}$, und mit der Voraussetzung $I|_{\mathcal{E}} \in X|_{\mathcal{E}}$ folgt $I|_{\Sigma^0} = I|_{\mathcal{E}} \in X|_{\mathcal{E}} = X|_{\Sigma^0}$. Nehmen wir nun an, die Behauptung gelte für ein $k \in \mathbb{N}_0$, aber es wäre $I|_{\Sigma^{k+1}} \notin X|_{\Sigma^{k+1}}$. Da nach Induktionsvoraussetzung $I|_{\Sigma^k} \in X|_{\Sigma^k}$ gilt, existiert ein Zustand $J \subseteq \Sigma^{k+1}$ mit $J \geq I|_{\Sigma^k}$ und $J \in X|_{\Sigma^{k+1}}$. Wegen der Widerspruchsannahme ist dann $J \neq I|_{\Sigma^{k+1}}$, also $J(a_{k+1}) \neq I(a_{k+1})$. Nun unterscheiden wir zwei Fälle:

Fall 1. $I(a_{k+1}) = 0$, d.h. $J(a_{k+1}) = 1$. Damit haben wir einen Zustand $J' = J|_{\Sigma^k} = I|_{\Sigma^k} \subseteq \Sigma^k$ mit $J' \uplus a_{k+1} = J \in X|_{\Sigma^{k+1}}$ und $J' \uplus \neg a_{k+1} = I|_{\Sigma^{k+1}} \notin X|_{\Sigma^{k+1}}$. Nach Konstruktion von R_X gibt es dann eine Klausel $a_{k+1} \leftarrow \mathrm{cl}(J') \in R_X$. Nun ist aber $I(\mathrm{cl}(J')) = I|_{\Sigma^k}(\mathrm{cl}(I|_{\Sigma^k}) = 1$. Wegen $I \models R_X$ folgt jetzt $I(a_{k+1}) = 1$ im Widerspruch zur Voraussetzung dieses Falles.

Fall 2. $I(a_{k+1}) = 1$, d.h. $J(a_{k+1}) = 0$. Damit haben wir analog zum ersten Fall einen Zustand $J' = J|_{\Sigma^k} = I|_{\Sigma^k} \subseteq \Sigma^k$ mit $J' \uplus \neg a_{k+1} = J \in X|_{\Sigma^{k+1}}$ und

$J' \uplus a_{k+1} = I|_{\Sigma^{k+1}} \notin X|_{\Sigma^{k+1}}$. Nach Konstruktion von R_X gibt es dann eine Klausel $\neg a_{k+1} \leftarrow \mathrm{cl}(J') \in R_X$. Nun ist aber $I(\mathrm{cl}(J')) = I|_{\Sigma^k}(\mathrm{cl}(I|_{\Sigma^k}) = 1$. Wegen $I \models R_X$ folgt jetzt $I(a_{k+1}) = 0$ im Widerspruch zur Voraussetzung dieses Falles.

Nun sei umgekehrt $I \in X$. Zu zeigen ist $I \models R_X$. Sei also C eine Klausel aus R_X, etwa $C \equiv L \leftarrow \mathrm{cl}(J)$ für einen entsprechenden Zustand J. Falls $I(\mathrm{cl}(J)) = 0$, ist ohnehin $I(C) = 1$. Sei also $I(\mathrm{cl}(J)) = 1$. Dann folgt aufgrund der Konstruktion von $\mathrm{cl}(J)$ zunächst $I \geq J$. Nun unterscheiden wir wieder zwei Fälle:

Fall 1. L ist ein positives Literal, d.h. $L \equiv a_k$ und $C \equiv a_k \leftarrow \mathrm{cl}(J)$ für ein $J \subseteq \Sigma^{k-1}$ mit $J \uplus a_k \in X|_{\Sigma^k}$ und $J \uplus \neg a_k \notin X|_{\Sigma^k}$. Wäre $I(a_k) = 0$, so wäre wegen $I \geq J$ auch $I|_{\Sigma^k} = J \uplus \neg a_k \notin X|_{\Sigma^k}$, im Widerspruch zur Voraussetzung $I \in X$.

Fall 2. L ist ein negatives Literal, d.h. $L \equiv \neg a_k$ und $C \equiv \neg a_k \leftarrow \mathrm{cl}(J)$ für ein $J \subseteq \Sigma^{k-1}$ mit $J \uplus \neg a_k \in X|_{\Sigma^k}$ und $J \uplus a_k \notin X|_{\Sigma^k}$. Wäre $I(\neg a_k) = 0$, d.h. $I(a_k) = 1$, so wäre wegen $I \geq J$ auch $I|_{\Sigma^k} = J \uplus a_k \notin X|_{\Sigma^k}$, ebenfalls im Widerspruch zur Voraussetzung $I \in X$. □

Nun sind wir in der Lage, eine Charakterisierung der Auflösbarkeit eines temporal stratifizierten Programms P und eines dazugehörigen Zieles $\leftarrow \alpha$ mit Hilfe der Fixpunktsemantik des vorausgegangenen Abschnittes zu formulieren und zu beweisen. Dieser Beweis enthält auch deutliche Hinweise darauf, wie ein Algorithmus zur Entscheidung, ob $P \cup \{\leftarrow \alpha\}$ auflösbar ist, und zur Berechnung der gesuchten Regelmenge im Falle der Auflösbarkeit aussehen könnte.

Theorem 3.13 *Sei P ein zulässiges, temporal stratifiziertes Programm über $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$. Weiter sei $\leftarrow \alpha$ ein temporal flaches Ziel. Dann sind die folgenden beiden Aussagen äquivalent:*

(AU1) $P \cup \{\leftarrow \alpha\}$ *ist auflösbar.*

(AU2) $\textsc{Init}_P \subseteq \hat{H}|_{\mathcal{E}^\circ}$.

Dabei bezeichnen wir mit $\hat{H} = \hat{H}_{P \cup \{\leftarrow \alpha\}}$ den größten Fixpunkt des $P \cup \{\leftarrow \alpha\}$ zugeordneten Funktionals $T_{P \cup \{\leftarrow \alpha\}}$.

Beweis. *(AU1) $\Longrightarrow$ (AU2).* Sei $P \cup \{\leftarrow \alpha\}$ auflösbar. Dann gibt es nach Definition eine zulässige Regelmenge R vom Typ $\mathcal{A} \leftarrow \mathcal{E}^\circ$, so daß jede Interpretation I über Σ, die Modell ist für $P \cup R$, auch Modell für α ist. Dazu definieren wir die Relation $X \subseteq 2^{\mathcal{E}}$ durch

$$X = \{\, I^{\mathcal{E}} \subseteq \mathcal{E} \mid I^{\mathcal{E}}|_{\mathcal{E}^\circ} \in \textsc{States}_{P \cup R}, I^{\mathcal{E}} \models R \,\}.$$

Mit Lemma 3.11 fassen wir hier $P \cup R$ auf als temporal stratifiziertes Programm über $\Sigma' = (\mathcal{E}^\circ, \emptyset, \mathcal{A} \cup \mathcal{P})$. Zunächst zeigen wir, daß X eine sichere Ein/Ausgabe-Zuordnung für $P \cup \{\leftarrow \alpha\}$ ist. Dazu beweisen wir für $I^{\mathcal{E}} \in X$ die Eigenschaften (SI1) und (SI2) aus Definition 3.4:

Zu (SI1). Sei J ein Zustand über Σ mit $J \geq I^{\mathcal{E}}$ und $J \models \text{DEF}(\mathcal{P})$. Zu zeigen ist $J(\alpha) = 1$. Zunächst gilt wegen $J \geq I^{\mathcal{E}}$ und $I^{\mathcal{E}} \models R$ auch $J \models \text{DEF}(\mathcal{P}) \cup R$. Da außerdem $I^{\mathcal{E}}|_{\mathcal{E}^\circ} \in \text{STATES}_{P \cup R}$, existiert nach Satz 3.6 eine Interpretation $I = (I_1, \ldots, I_k)$ mit $I \models P \cup R$ und $I_k \geq I^{\mathcal{E}}|_{\mathcal{E}^\circ}$. Wegen $J|_{\mathcal{E}^\circ} = I_k|_{\mathcal{E}^\circ}$ und $J \models \text{DEF}(\mathcal{P}) \cup R$, ist nach Lemma 2.7 auch $I' = (I_1, \ldots, I_{k-1}, J) \models P \cup R$. Aufgrund der Wahl von R folgt $I' \models \alpha$ und somit $J(\alpha) = 1$.

Zu (SI2). Sei $J^\circ \in \text{SUCC}_P(I^{\mathcal{E}})$. Zu zeigen ist $J^\circ \in X|_{\mathcal{E}^\circ}$. Da nach Konstruktion von X gilt $I^{\mathcal{E}} \models R$, ist auch $J^\circ \in \text{SUCC}_{P \cup R}(I^{\mathcal{E}}|_{\mathcal{E}^\circ})$. Da $I^{\mathcal{E}}|_{\mathcal{E}^\circ} \in \text{STATES}_{P \cup R}$, ist $I^{\mathcal{E}}|_{\mathcal{E}^\circ} \in \text{STATES}^k_{P \cup R}$ für ein $k \in \mathbb{N}_0$, und nach Definition von $\text{STATES}_{P \cup R}$ folgt $J^\circ \in \text{STATES}^{k+1}_{P \cup R} \subseteq \text{STATES}_{P \cup R}$. Nach Lemma 2.5 konstruieren wir einen Zustand $J \geq J^\circ$ über Σ mit $J \models \text{DEF}(\mathcal{P}) \cup R$. Dann gilt $J|_{\mathcal{E}} \models R$ und somit $J^\circ = J|_{\mathcal{E}^\circ} \in X|_{\mathcal{E}^\circ}$.

Also ist X eine sichere Ein/Ausgabe-Zuordnung für $P \cup \{\leftarrow \alpha\}$, und damit $X \subseteq \hat{H}$. Sei nun $I^\circ \in \text{INIT}_P = \text{INIT}_{P \cup R} = \text{STATES}^1_{P \cup R} \subseteq \text{STATES}_{P \cup R}$. Nach Lemma 2.5 konstruieren wir einen Zustand $I \geq I^\circ$ über Σ mit $I \models \text{DEF}(\mathcal{P}) \cup R$. Dann gilt $I|_{\mathcal{E}} \models R$ und damit $I^\circ = I|_{\mathcal{E}^\circ} \in X|_{\mathcal{E}^\circ} \subseteq \hat{H}|_{\mathcal{E}^\circ}$. Somit folgt die Behauptung $\text{INIT}_P \subseteq \hat{H}|_{\mathcal{E}^\circ}$.

(AU2) $\Longrightarrow$ *(AU1).* Sei $\text{INIT}_P \subseteq \hat{H}|_{\mathcal{E}^\circ}$. Nach Satz 3.12 konstruieren wir zu $\hat{H}$ eine Regelmenge $\hat{R}_H$ vom Typ $\mathcal{A} \leftarrow \mathcal{E}^\circ$, so daß für alle $I^{\mathcal{E}} \subseteq \mathcal{E} = \mathcal{E}^\circ \cup \mathcal{A}$ mit $I^{\mathcal{E}}|_{\mathcal{E}^\circ} \in \hat{H}|_{\mathcal{E}^\circ}$ die folgende Äquivalenz gilt:

$$I^{\mathcal{E}} \models \hat{R}_H \iff I^{\mathcal{E}} \in \hat{H}. \tag{2}$$

Nun sei I ein Modell für $P \cup \hat{R}_H$, etwa $I = (I_1, I_2, \ldots)$. Wir zeigen durch Induktion über k, daß für jedes $k = 1, 2, \ldots$ gilt

$$I_k|_{\mathcal{E}} \in \hat{H}.$$

Da $\hat{H}$ eine gültige Ein/Ausgabe-Zuordnung für $P \cup \{\leftarrow \alpha\}$ ist, folgt dann wegen $I \models P$ und damit $I_k \models \text{DEF}(\mathcal{P})$ auch $I_k \models \alpha$ für alle k, und das bedeutet, daß $\hat{R}_H$ die in der Definition der Auflösbarkeit geforderte Regelmenge ist.

Im Induktionsanfang für $k = 1$ folgt aus $I \models P$ zunächst $I_1|_{\mathcal{E}^\circ} \in \text{INIT}_P \subseteq \hat{H}|_{\mathcal{E}^\circ}$ wegen Lemma 3.1 und der Voraussetzung $\text{INIT}_P \subseteq \hat{H}|_{\mathcal{E}^\circ}$. Mit $I \models \hat{R}_H$ und folglich $I_1|_{\mathcal{E}} \models \hat{R}_H$ gilt dann nach obiger Eigenschaft (2) auch $I_1|_{\mathcal{E}} \in \hat{H}$.

Als Induktionsvoraussetzung gelte nun $I_k|_{\mathcal{E}} \in \hat{H}$ für ein $k \in \mathbb{N}_0$. Da $I \models P$, ist nach Lemma 3.1 auch $I_{k+1}|_{\mathcal{E}^\circ} \in \text{SUCC}_P(I_k|_{\mathcal{E}})$, und da $\hat{H}$ eine sichere Ein/Ausgabe-Zuordnung für $P \cup \{\leftarrow \alpha\}$ ist, folgt $I_{k+1}|_{\mathcal{E}^\circ} \in \hat{H}|_{\mathcal{E}^\circ}$. Da andererseits $I \models \hat{R}_H$ und damit $I_{k+1}|_{\mathcal{E}} \models \hat{R}_H$, folgt aus obiger Eigenschaft (2) auch $I_{k+1}|_{\mathcal{E}} \in \hat{H}$. □

Um für ein gegebenes temporal stratifiziertes Program P und ein temporal flaches Ziel $\leftarrow \alpha$ das Generierungsproblem zu lösen, genügt es offenbar, den größten Fixpunkt $\hat{H}_{P \cup \{\leftarrow \alpha\}}$ des zugeordneten Funktionals zu berechnen. Dann kann direkt getestet werden, ob $\text{INIT}_P \subseteq \hat{H}_{P \cup \{\leftarrow \alpha\}}$ gilt, d.h. ob überhaupt eine Lösung existiert, und wenn dies der Fall ist, können wir nach Satz 3.12 eine Regelmenge $\hat{R}$

konstruieren, die sofort eine Implementierung des gesuchten Steuerungsprogrammes liefert.

Ein naheliegender Ansatz zur Realisierung dieses Verfahrens besteht darin, die Relationen $H_{P\cup\{\leftarrow\alpha\}}$, $\textsc{Init}_P$ und $\textsc{Succ}_P$ explizit aufzuzählen und die Fixpunkt-Iteration über diesen Mengen durchzuführen. Ein solches Vorgehen wird jedoch aufgrund des bekannten Problems der Zustands-Explosion schnell an die Grenzen der praktischen Realisierbarkeit stoßen.

Ein alternativer und wesentlich erfolgversprechenderer Ansatz ergibt sich aus einer *symbolischen* Repräsentation der betroffenen Relationen [BCLMD94]. Dabei werden nicht mehr alle in einer Ein/Ausgabe-Zuordnung enthaltenen Zustände aufgezählt, sondern sie werden vielmehr durch eine aussagenlogische Formel beschrieben. Dann kann die Fixpunktiteration über diesen Formeln durchgeführt werden. Diese Vorgehensweise führt erfahrungsgemäß zu erheblichen Effizienzgewinnen, da die Ein/Ausgabe-Zuordnungen auf diese Weise wesentlich kompakter repräsentiert werden können. In diesem Bereich konnten durch die Verwendung von OBDDs [Brya86] besonders eindrucksvolle Ergebnisse erzielt werden.

3.5 Stromverarbeitende Funktionen

Bei den Betrachtungen des vorangegangenen Abschnittes stellt sich die Frage, ob die Eigenschaft der Auflösbarkeit eines temporal stratifizierten Programmes über $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$ und ein temporal flachen Zieles immer dann gegeben ist, wenn ein entsprechendes Steuerungsprogramm existiert. Schließlich kann das durch eine zulässige Regelmenge beschriebene Programm nur auf die in den Prädikatensymbolen aus $\mathcal{E}^\circ$ enthaltene Zustandsinformation zugreifen, und es wäre ja denkbar, daß die Vorgeschichte, d.h. die früheren Zustände, weitere relevante Information enthalten. Deshalb soll in diesem Abschnitt bewiesen werden, daß dies nicht der Fall ist.

In einem sehr viel allgemeineren Rahmen werden reaktive Systeme nicht als temporal stratifizierte Programme oder endliche Automaten aufgefaßt, sondern als Funktionen über *Strömen*, die einer Folge von Eingaben, einem sogenannten Strom, eine Folge von Ausgaben zuordnen. In diesem Rahmen können Prozesse aller Art modelliert und in einer Algebra von sogenannten stromverarbeitenden Funktionen analysiert werden [Broy93]. Wir interessieren uns hier lediglich für den Zusammenhang zwischen unserem Modell der temporal stratifizierten Programme und dem Modell der stromverarbeitenden Funktionen.

In unserem Kontext ist es sinnvoll, an Stelle von Strömen über beliebigen Alphabeten nur Ströme über Zuständen, d.h. Belegungen von Prädikatensymbolen zu betrachten. Dann kann jede Interpretation in unserem Sinne als Strom aufgefaßt werden. Wie in Abschnitt 2 bezeichnen wir daher mit $\mathcal{I}(\Sigma)$ die Menge der endlichen oder unendlichen Interpretationen über Σ. Mit $I \frown J$ bezeichnen wir das Ergebnis der Verkettung der beiden Interpretationen I und J. Dabei nehmen wir für unendliches I an, daß $I \frown J = I$. So erhalten wir in natürlicher Weise

eine Präfix-Ordnung über Interpretationen.

Definition 3.10 (Präfix-Ordnung) Seien $I, J \in \mathcal{I}(\Sigma)$. Wir bezeichnen I als ein *Präfix* von J, in Zeichen $I \sqsubseteq J$, wenn es eine Iterpretation $I' \in \mathcal{I}(\Sigma)$ gibt, so daß $I \frown I' = J$:

$$I \sqsubseteq J \iff \exists I' \in \mathcal{I}(\Sigma) : I \frown I' = J.$$

Das Verhalten eines reaktiven Systemen mit Eingabe-Kanälen Σ_1 und Ausgabe-Kanälen Σ_2 kann beschrieben werden durch eine Funktion

$$F{:}\mathcal{I}(\Sigma_1) \to \mathcal{I}(\Sigma_2),$$

die wir als *stromverarbeitende Funktion* vom Typ $\Sigma_1 \to \Sigma_2$ bezeichnen.

Eine im Hinblick auf die Implementierung sehr wichtige Forderung an solche stromverarbeitende Funktionen ist die der *Kausalität* [Halb93]. Diese besagt, daß die Werte von Ausgaben niemals von Eingaben abhängen dürfen, die erst eintreffen, nachdem die betreffenden Ausgaben bereits getätigt wurden. Diese Eigenschaft wird im Kontext der stromverarbeitenden Funktionen durch die *Präfix-Monotonie* gewährleistet.

Definition 3.11 (Präfix-Monotonie) Sei $F{:}\mathcal{I}(\Sigma_1) \to \mathcal{I}(\Sigma_2)$ eine stromverarbeitende Funktion. F heißt *präfix-monoton*, wenn für alle $I, J \in \mathcal{I}(\Sigma_1)$ die folgende Implikation gilt:

$$I \sqsubseteq J \Longrightarrow F(I) \sqsubseteq F(J).$$

Natürlich stellt sich hier genau wie bei dem Problem der Auflösbarkeit die Frage, ob und wann zu einer gegebenen Spezifikation eine präfix-monotone stromverarbeitende Funktion existiert, die den Anforderungen der Spezifikation genügt. Wir wollen diese Frage für den Fall untersuchen, daß die Spezifikation durch ein temporal stratifiziertes Programm gegeben ist.

Definition 3.12 (Lösbarkeit durch Funktionen) Sei P ein temporal stratifiziertes Programm über $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$ und $\mathcal{E} = \mathcal{E}^\circ \cup \mathcal{A}$. Weiter sei $\leftarrow \alpha$ ein temporal flaches Ziel. Wir nennen das Paar $P \cup \{\leftarrow \alpha\}$ *lösbar durch Funktionen*, wenn es eine stromverarbeitende Funktion

$$F{:}\mathcal{I}(\mathcal{E}^\circ) \to \mathcal{I}(\mathcal{E})$$

gibt, die die folgenden Eigenschaften besitzt:

(LF1) F ist präfix-monoton.

(LF2) Für alle Interpretationen I° über $\mathcal{E}^\circ$ gilt $F(I^\circ) \geq I^\circ$.

(LF3) Für alle Interpretationen I° über $\mathcal{E}^\circ$ und alle Interpretationen $I \geq F(I^\circ)$ über Σ mit $I \models P$ gilt auch $I \models \alpha$.

Diese Forderungen erscheinen in der Tat wesentlich schwächer als die im vorausgegangenen Abschnitt definierte Eigenschaft der Auflösbarkeit. Denn während dort eine Zuordnung gefordert wird, die die neuen Ausgaben allein aus dem aktuellen Zustand und den aktuellen Eingaben berechnet, kann die hier erwähnte stromverarbeitende Funktion zusätzlich auf alle Zustände und Eingaben der Vergangenheit zugreifen.

Beispiel 3.6 Das Programm und Ziel in Beispiel 3.1 erweist sich als lösbar durch Funktionen — die folgende stromverarbeitende Funktion $F : \mathcal{I}(\mathcal{E}^\circ) \to \mathcal{I}(\mathcal{E})$ erfüllt alle Bedingungen der Definition:

$$F(I_1^\circ, I_2^\circ, \ldots) = (I_1^\mathcal{E}, I_2^\mathcal{E}, \ldots), \text{ wobei } I_k^\mathcal{E}(a) = I_k^\circ(p^\circ) \text{ für alle } k \in \mathbb{N}_0.$$

Beispiel 3.7 Wir betrachten das folgende temporal stratifizierte Programm und temporal flache Ziel über $\Sigma = (\{a^\circ, e^\circ\}, \{a\}, \{\text{ANF}\})$:

$$\begin{array}{lllll} P: & e^\circ & \leftarrow & \text{START} \\ & a^\circ & \leftarrow & \text{START} \\ & \circ\, a^\circ & \leftarrow & a \\ & \circ\, \neg a^\circ & \leftarrow & \neg a \\ & \text{ANF} & \leftarrow & a^\circ \wedge e^\circ \\ & \text{ANF} & \leftarrow & \neg a^\circ \wedge \neg e^\circ \\ \leftarrow \alpha: & \leftarrow & \text{ANF} & \end{array}$$

Hier wird gefordert, daß die Ausgabeleitung a stets den gleichen Wert führt wie das Zustandsbit (bzw. die Eingabeleitung) e° im folgenden Zustand — unter dem Aspekt der Kausalität offenbar ein unlösbares Problem. Man kann auch zeigen, daß $P \cup \{\leftarrow \alpha\}$ weder auflösbar noch lösbar durch Funktionen ist.

Wir betrachten die folgende stromverarbeitende Funktion $F{:}\mathcal{I}(\mathcal{E}^\circ) \to \mathcal{I}(\mathcal{E})$ mit $F(I_1^\circ, \ldots, I_n^\circ) = (I_1^\mathcal{E}, \ldots, I_n^\mathcal{E})$, wobei

$$I_k^\mathcal{E}(p) = \begin{cases} I^\circ(p), & \text{falls } p \in \mathcal{E}^\circ; \\ I_{k+1}^\circ(e^\circ), & \text{falls } p \equiv a \text{ und } k < n; \\ 0, & \text{falls } p \equiv a \text{ und } k = n. \end{cases}$$

Offenbar genügt F aufgrund seiner Konstruktion direkt der Bedingung (LF2). Auch sieht man ein, daß F die Bedingung (LF3) erfüllt, denn bei jeder Interpretation $I \geq F(I^\circ)$ mit $I \models P$ gilt für alle $k \in \mathbb{N}_0$ offenbar $I_k(a^\circ) = I_k(e^\circ)$, und folglich auch $I(\text{ANF}) = 1$ sowie $I_k \models \alpha$.

Allerdings ist F nicht präfix-monoton. Um dies einzusehen, betrachten wir etwa $I^\circ = (\{a^\circ, e^\circ\})$ und $J^\circ = I^\circ \frown (\{a^\circ, e^\circ\})$. Hier gilt nämlich $I^\circ \sqsubseteq J^\circ$, aber wir haben

$$F(I^\circ) = (\{a^\circ, e^\circ, \neg a\}) \not\sqsubseteq (\{a^\circ, e^\circ, a\}, \{a^\circ, e^\circ, \neg a\}) = F(J^\circ).$$

Das wichtigste Ergebnis dieses Abschnittes ist das folgende Theorem, das besagt, daß für temporal stratifizierte Programme Lösbarkeit durch Funktionen und Auflösbarkeit äquivalent sind. Dies ist ein wichtiges Vollständigkeitsresultat, denn es bedeutet, daß wir die Menge der Zustands-Prädikatensymbole $\mathcal{E}^\circ$ bei der temporalen Stratifizierung ausreichend umfangreich gewählt haben, so daß sie alle nötige Information enthält. Damit genügt es, Spezifikationen reaktiver Systeme in Form von temporal stratifizierten Programmen und ihre Lösungen durch zulässige Regelmengen zu betrachten.

Theorem 3.14 *Sei P ein zulässiges, temporal stratifiziertes Programm über $\Sigma = (\mathcal{E}^\circ, \mathcal{A}, \mathcal{P})$, und sei $\leftarrow \alpha$ ein temporal flaches Ziel. Genau dann ist $P \cup \{\leftarrow \alpha\}$ lösbar durch Funktionen, wenn $P \cup \{\leftarrow \alpha\}$ auflösbar ist.*

Beweis. Wie üblich schreiben wir $\mathcal{E} = \mathcal{E}^\circ \cup \mathcal{A}$. Zuerst zeigen wir, daß Lösbarkeit durch Funktionen stets Auflösbarkeit zur Folge hat. Sei also $P \cup \{\leftarrow \alpha\}$ lösbar durch Funktionen, und sei $F: \mathcal{I}(\mathcal{E}^\circ) \to \mathcal{I}(\mathcal{E})$ die stromverarbeitende Funktion, die den Eigenschaften (LF1) bis (LF3) aus der Definition genügt. Dann definieren wir die folgende Ein/Ausgabe-Zuordung $H = H(F, P) \subseteq 2^{\mathcal{E}}$ für P:

$$H = \{\, I_k^{\mathcal{E}} \subseteq \mathcal{E} \mid \text{Es gibt eine Interpretation } I^\circ \text{ über } \mathcal{E}^\circ \text{ mit } F(I^\circ) = (I_1^{\mathcal{E}}, \ldots, I_k^{\mathcal{E}}), \text{ und es gibt eine Interpretation } I \geq F(I^\circ) \text{ über } \Sigma \text{ mit } I \models P. \,\}.$$

Wir wollen Theorem 3.13 anwenden. Deshalb zeigen wir zunächst, daß H eine sichere Ein/Ausgabe-Zuordnung für $P \cup \{\leftarrow \alpha\}$ ist, d.h. daß H die beiden Eigenschaften (SI1) und (SI2) aus Definition 3.4 erfüllt. Sei also $I_k^{\mathcal{E}} \in H$. Dann existiert zunächst eine Interpretation $I^\circ = (I_1^\circ, \ldots, I_k^\circ)$ über $\mathcal{E}^\circ$, so daß $F(I^\circ) = I^{\mathcal{E}} = (I_1^{\mathcal{E}}, \ldots, I_k^{\mathcal{E}})$. Außerdem gibt es eine Erweiterung $I = (I_1, \ldots, I_k) \geq I^{\mathcal{E}}$ auf Σ mit $I \models P$.

Zu (SI1). Sei $I_k' \geq I_k^{\mathcal{E}}$ ein Zustand über Σ mit $I_k' \models \text{DEF}(\mathcal{P})$. Zu zeigen ist $I_k' \models \alpha$. Da $I = (I_1, \ldots, I_k) \models P$ und $I_k'|_{\mathcal{E}} = I_k^{\mathcal{E}} = I_k|_{\mathcal{E}}$, ist nach Lemma 2.7 auch $I' = (I_1, \ldots, I_{k-1}, I_k') \models P$. Dann folgt wegen $I' \geq I'|_{\mathcal{E}} = I^{\mathcal{E}} = F(I^\circ)$ aus der Eigenschaft (LF3) auch $I' \models \alpha$ und somit $I_k' \models \alpha$.

Zu (SI2). Sei $I_{k+1}^\circ \in \text{SUCC}_P(I_k^{\mathcal{E}})$. Zu zeigen ist $I_{k+1}^\circ \in H|_{\mathcal{E}^\circ}$. Da F nach Eigenschaft (LF1) präfix-monoton ist, folgt zunächst $F(I_1^\circ, \ldots, I_{k+1}^\circ) = I^{\mathcal{E}} \frown (I_{k+1}^{\mathcal{E}}) = (I_1^{\mathcal{E}}, \ldots, I_k^{\mathcal{E}}, I_{k+1}^{\mathcal{E}})$, wobei wegen (LF2) auch $I_{k+1}^{\mathcal{E}} \geq I_{k+1}^\circ$ gilt. Nach Lemma 3.3 existieren zu $I = (I_1, \ldots, I_k) \geq (I_1^{\mathcal{E}}, \ldots, I_k^{\mathcal{E}})$ und $I_{k+1}^{\mathcal{E}}$ mit $I_{k+1}^{\mathcal{E}}|_{\mathcal{E}^\circ} \in \text{SUCC}_P(I_k^{\mathcal{E}})$ wegen $I \models P$ zwei Zustände $I_\kappa' \geq I_k^{\mathcal{E}}$ und $I_{k+1}' \geq I_{k+1}^{\mathcal{E}}$ über Σ, so daß $I' = (I_1, \ldots, I_{k-1}, I_k', I_{k+1}') \models P$. Außerdem gilt hier offenbar $I' \geq (I_1^{\mathcal{E}}, \ldots, I_{k+1}^{\mathcal{E}}) = F(I_1^\circ, \ldots, I_{k+1}^\circ)$, es folgt $I_{k+1}^{\mathcal{E}} \in H$ und damit $I_{k+1}^\circ = I_{k+1}^{\mathcal{E}}|_{\mathcal{E}^\circ} \in H|_{\mathcal{E}^\circ}$.

Somit ist H eine sichere Ein/Ausgabe-Zuordnung für $P \cup \{\leftarrow \alpha\}$, d.h. es ist $H \subseteq \hat{H}_{P \cup \{\leftarrow \alpha\}}$. Nun sei $I_1^\circ \in \text{INIT}_P$, und sei $F(I_1^\circ) = (I_1^{\mathcal{E}})$ mit $I_1^{\mathcal{E}} \geq I_1^\circ$. Nach Lemma 3.2 gibt es einen Zustand $I_1 \geq I_1^{\mathcal{E}}$ über Σ, so daß $I = (I_1) \models P$. Damit

ist $I_1^{\mathcal{E}} \in H$ und $I_1^\circ \in H|_{\mathcal{E}^\circ}$. Da dies für alle $I_1^\circ \in \text{INIT}_P$ gilt, folgt $\text{INIT}_P \subseteq H|_{\mathcal{E}^\circ} \subseteq \hat{H}_{P\cup\{\leftarrow\alpha\}}|_{\mathcal{E}^\circ}$. Daraus aber folgt mit Theorem 3.13, daß $P \cup \{\leftarrow \alpha\}$ auflösbar ist.

Umgekehrt sei nun $P\cup\{\leftarrow \alpha\}$ auflösbar. Nach Theorem 3.13 ist dann $\text{INIT}_P \subseteq \hat{H}_{P\cup\{\leftarrow\alpha\}}$. Nun existiert zu jedem Zustand $I_0^\circ \in \hat{H}_{P\cup\{\leftarrow\alpha\}}|_{\mathcal{E}^\circ}$ eine Erweiterung $I_0^{\mathcal{E}} \geq I_0^\circ$ mit $I_0^{\mathcal{E}} \in \hat{H}_{P\cup\{\leftarrow\alpha\}}$. Wir wählen zu jedem Zustand I_0° eine solche Erweiterung $f(I_0^\circ) = I_0^{\mathcal{E}}$ und erklären damit

$$F: 2^{\mathcal{E}^\circ} \to 2^{\mathcal{E}};\ I_0^\circ \mapsto \begin{cases} f(I_0^\circ), & \text{falls } I_0^\circ \in \hat{H}_{P\cup\{\leftarrow\alpha\}}|_{\mathcal{E}^\circ}; \\ I_0^\circ, & \text{sonst.} \end{cases}$$

Im zweiten Fall ist mit I_0° der Zustand $I_0^{\mathcal{E}} \geq I_0^\circ$ über $\mathcal{E}$ gemeint, der die Prädikatensymbole in $\mathcal{A}$ alle als 0 interpretiert. Diese Abbildung von Zuständen erweitern wir in kanonischer Weise zu einer stromverarbeitenden Funktion

$$F: \mathcal{I}(\mathcal{E}^\circ) \to \mathcal{I}(\mathcal{E});\ (I_1^\circ, I_2^\circ, \ldots) \mapsto (F(I_1^\circ), F(I_2^\circ), \ldots).$$

Wir behaupten, daß F die Eigenschaften (LF1) bis (LF3) besitzt. Natürlich ist F präfix-monoton, da F als Abbildung über Zuständen definiert wurde; damit gilt (LF1). Auch die Eigenschaft (LF2) folgt direkt aus der Konstruktion.

Um auch Eigenschaft (LF3) zu begründen, sei $I^\circ = (I_1^\circ, I_2^\circ, \ldots)$ eine Interpretation über $\mathcal{E}^\circ$ mit $F(I^\circ) = I^{\mathcal{E}} = (I_1^{\mathcal{E}}, I_2^{\mathcal{E}}, \ldots) \geq I^\circ$. Außerdem sei $I = (I_1, I_2, \ldots) \geq I^{\mathcal{E}}$ eine Interpretation über Σ, so daß $I \models P$. Zu zeigen ist $I \models \alpha$. Wir zeigen durch Induktion über k

$$I_k^{\mathcal{E}} \in \hat{H}_{P\cup\{\leftarrow\alpha\}}.$$

Dann folgt jeweils für $I_k \geq I_k^{\mathcal{E}}$ mit $I_k \models \text{DEF}(\mathcal{P})$ auch $I_k \models \alpha$, da $\hat{H}_{P\cup\{\leftarrow\alpha\}}$ eine sichere Ein/Ausgabe-Zuordnung ist.

Im Induktionsanfang für $k = 1$ betrachten wir $I = (I_1) \models P$. Nach Lemma 3.1 gilt $I_1^\circ = I_1|_{\mathcal{E}^\circ} \in \text{INIT}_P \subseteq \hat{H}_{P\cup\{\leftarrow\alpha\}}|_{\mathcal{E}^\circ}$. Dann ist aufgrund der Konstruktion von F auch $I_1^{\mathcal{E}} = F(I_1|_{\mathcal{E}^\circ}) \in \hat{H}_{P\cup\{\leftarrow\alpha\}}$.

Als Induktionsvoraussetzung sei nun $I_k^{\mathcal{E}} \in \hat{H}_{P\cup\{\leftarrow\alpha\}}$. Da $I \models P$, folgt mit Lemma 3.1 auch $I_{k+1}^\circ = I_{k+1}|_{\mathcal{E}^\circ} \in \text{SUCC}_P(I_k|_{\mathcal{E}}) = \text{SUCC}_P(I_k^{\mathcal{E}})$. Da $\hat{H}_{P\cup\{\leftarrow\alpha\}}$ eine sichere Ein/Ausgabe-Zuordnung ist, folgt mit der Induktionsvoraussetzung auch $I_{k+1}^\circ \in \hat{H}_{P\cup\{\leftarrow\alpha\}}|_{\mathcal{E}^\circ}$, und aus der Konstruktion von F folgt schließlich $I_{k+1}^{\mathcal{E}} = F(I_{k+1}^\circ) \in \hat{H}_{P\cup\{\leftarrow\alpha\}}$. □

4 Schlußfolgerung

In dieser Arbeit haben wir einen Ansatz zur Spezifikation und automatischen Synthese zustandsendlicher, reaktiver Systeme auf der Grundlage des logischen Programmierens vorgeschlagen. Reaktive Systeme werden rein deklarativ spezifiziert, in dem wir erstens das Verhalten der Umgebung, mit der das System

interagieren soll, durch ein sogenanntes temporal stratifiziertes Programm, d.h. eine Menge von PROLOG-ähnlichen Programmklauseln, beschreiben und zweitens die Sicherheitsbedingungen, die von dem System erfüllt werden müssen, als Beweisziel in der Gestalt einer temporallogischen Formel vorgeben.

Ausgehend von dieser Spezifikationssprache haben wir gezeigt, unter welchen Bedingungen ein Steuerungsprogramm existiert, das die Einhaltung dieser Sicherheitsbedingungen in der gegebenen Umgebung garantiert. Falls eine solche Lösung existiert, ergibt sie sich direkt als Ergebnis einer Fixpunkt-Berechnung über der Menge aller möglichen Steuerungsprogramme. Wir haben gezeigt, wie dieses Verfahren im Prinzip durchgeführt werden kann, und seine Vollständigkeit bewiesen. Eine symbolische Realisierung auf der Grundlage von OBDDs [Brya86] dürfte sehr effiziente Algorithmen zur Lösung dieses Generierungsproblems liefern.

Ein anderer Lösungsansatz besteht in der Verwendung von resolutionsbasierten Verfahren und von Methoden aus dem Bereich der logischen Programmierung. Die Sprache der temporal stratifizierten Programme wurde so gewählt, daß sie für solche Verfahren besonders gut zugänglich ist.

Die hier verwendete Spezifikationssprache wurde bewußt sehr einfach gehalten, um eine sinnvolle Formulierung und Analyse des vorgeschlagenen Synthese-Verfahrens und der Vollständigkeitsaussagen zu ermöglichen. Später wird es notwendig sein, diese Sprache um zusätzliche Konstrukte zu erweitern, um eine bessere Anwendernähe zu erreichen. Beispielsweise wäre es denkbar, im Rumpf von Klauseln an Stelle von Konjunktionen von Literalen beliebige Konstrukte zuzulassen, die auch vergangenheitsorientierte temporale Operatoren enthalten können. Aber auch die Einführung von noch komplizierteren Sprachelementen ist denkbar, wie sie beispielsweise in [SpMa92] oder [LGLL91] vorgeschlagen werden. Solche erweiterte Programme könnten vor der Bearbeitung durch Resolution in die Normalform eines temporal stratifizierten Programms transformiert werden.

Ein offenes Problem besteht in der Behandlung von Anforderungen, die sich auf die *Lebendigkeit* beziehen. Durch temporal flache Ziele können nur *Sicherheitsbedingungen* beschrieben werden. Obwohl damit viele praxisrelevante Fälle erfaßt werden können — insbesondere alle Arten von Echtzeitbedingungen — erscheint eine Erweiterung auf echte Lebendigkeitsanforderungen wünschenswert. Da diese in der vorliegenden Logik noch nicht formuliert werden können, wird es dazu allerdings nötig sein, die Logik bezüglich ihrer Ausdrucksmächtigkeit zu erweitern.

Literatur

[BCLMD94] Burch, J. R., Clarke, E. M., Long, D. E., McMillan, K. E., Dill, D. L.: *Symbolic model checking for sequential circuit verifi-*

cation. IEEE Transactions on computer-aided design of integrated circuits and systems **13** (4), 1994.

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

[Broy93] Broy, M.: *Functional specifications of time-sensitive communicating systems.* ACM Transactions on Software Engineering and Methodology, **2** (1), 1–46, 1993.

[Brya86] Bryant, R. E.: *Graph-based algorithms for boolean function manipulation.* IEEE Trans. Comp. **35** (8), 677–691, 1986.

[Brya92] Bryant, R. E.: *Symbolic boolean manipulation with ordered binary-decision diagrams.* ACM Computing Surveys **24** (3), 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.

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

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

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

[MaMa94] Malik, R., Mayer, O.: *Spezifikation reaktiver Systeme durch temporal stratifizierte Programme.* Interner Bericht, FB Informatik, Universität Kaiserslautern, 1994.

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

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

[Tars55] Tarski, A.: *A lattice-theoretical fixpoint theorem and its applications.* Pacific J. Math. **5**, 285–309, 1955.

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