

Abstract-State Machines

Eine Sammlung didaktischer Beispiele

Andreas Glausch

15. Februar 2003

Zusammenfassung

Diese Studienarbeit versucht durch eine Vielzahl von Beispielen den Begriff *Abstract-State Machine* und *sequenzieller Algorithmus* didaktisch sinnvoll darzustellen. Es werden dabei sowohl Beispiele als auch Gegenbeispiele angegeben, um Umfang und Grenzen dieser Begriffe aufzuzeigen.

Inhaltsverzeichnis

1	Begriffe	3
1.1	Grundlegendes aus der Algebra	3
1.2	Sequenzieller Algorithmus	5
1.3	Abstract-State Machine	7
2	Gegenbeispiele zum Algorithmusbegriff	10
2.1	Nicht-algorithmische Abläufe	10
2.2	Nicht-algorithmische Transformationssysteme	13
3	Ein syntaktisches Rechenmodell	16

1 Begriffe

1.1 Grundlegendes aus der Algebra

Definition 1 (Signatur). Eine *Signatur* Σ ist ein Tupel $(f_1, \dots, f_k, n_1, \dots, n_k)$, wobei $k \in \mathbb{N}^+$, $n_1, \dots, n_k \in \mathbb{N}$ und f_1, \dots, f_k Funktionssymbole sind.

Eine Signatur besteht also aus einer Menge von Funktionssymbolen, wobei jedem Symbol f_i eine Stelligkeit n_i zugeordnet ist.

Definition 2 (Σ -Algebra). Sei eine Signatur $\Sigma = (f_1, \dots, f_k, n_1, \dots, n_k)$ gegeben. Eine *Algebra* \mathfrak{A} über der Signatur Σ ist ein Tupel (U, g_1, \dots, g_k) , wobei:

- U eine nichtleere Menge, das *Universum* von \mathfrak{A} ist, und
- g_i eine n_i -stellige Funktion mit Werte- und Bildbereich in U für $i = 1, \dots, k$ ist. Die Funktion g_i wird auch mit $\llbracket f_i \rrbracket_{\mathfrak{A}}$, der *Interpretation* von f_i durch die Algebra \mathfrak{A} , bezeichnet.

Die Menge aller Σ -Algebren sei mit Alg_{Σ} bezeichnet.

Eine Σ -Algebra ist somit eine semantische Interpretation einer Signatur Σ .

Definition 3 (Homomorphismus, Isomorphismus). Sei eine Signatur $\Sigma = (f_1, \dots, f_k, n_1, \dots, n_k)$ gegeben. Ein *Homomorphismus* zwischen den Σ -Algebren $\mathfrak{A} = (U, g_1, \dots, g_k)$ und $\mathfrak{B} = (V, h_1, \dots, h_k)$ ist eine Funktion $\Phi : U \rightarrow V$ mit $\Phi(g_i(x_1, \dots, x_{n_i})) = h_i(\Phi(x_1), \dots, \Phi(x_{n_i}))$.

Ist Φ bijektiv, so ist Φ ein *Isomorphismus*. Zwei Σ -Algebren heißen *isomorph*, wenn es einen Isomorphismus Φ zwischen ihnen gibt.

Isomorphismus definiert eine Äquivalenzrelation für Algebren. Sind zwei Algebren isomorph, so kann, intuitiv gesprochen, jede Berechnung der einen Algebra analog auch in der anderen ausgeführt werden.

Definition 4 (Isomorpher Abschluss). Sei eine Signatur Σ und $S \subseteq \text{Alg}_{\Sigma}$ gegeben. Dann ist $[S] \subseteq \text{Alg}_{\Sigma}$ folgendermaßen definiert: $\mathfrak{A} \in [S]$ gdw. es ein $\mathfrak{A}' \in S$ gibt, so dass \mathfrak{A} und \mathfrak{A}' isomorph sind. $[S]$ ist der *isomorphe Abschluss* von S . Ist $[S] = S$, dann heißt S *abgeschlossen bezüglich Isomorphismus*.

Definition 5 (Σ -Term). Sei eine Signatur Σ gegeben. Wir definieren induktiv:

- Ein 0-stelliges Funktionssymbol aus Σ ist ein Σ -Term.
- Sind t_1, \dots, t_n Terme und ist f ein n -stelliges Funktionssymbol aus Σ , so ist $f(t_1, \dots, t_n)$ ein Σ -Term.

Die Menge aller Σ -Terme sei mit T_{Σ} bezeichnet.

Ein Σ -Term ist also eine rein syntaktische Struktur. Zusammen mit einer Σ -Algebra kann dieser Term dann aber semantisch folgendermaßen interpretiert werden:

Definition 6 ($\llbracket t \rrbracket_{\mathfrak{A}}$). Seien eine Signatur $\Sigma = (f_1, \dots, f_k, n_1, \dots, n_k)$, eine Σ -Algebra $\mathfrak{A} = (U, g_1, \dots, g_k)$ und ein Σ -Term t gegeben. $\llbracket t \rrbracket_{\mathfrak{A}}$ ist dann induktiv folgendermaßen definiert:

- Falls t ein 0-stelliges Funktionssymbol f_i ist, so ist $\llbracket t \rrbracket_{\mathfrak{A}} = g_i$
- Falls $t = f_i(t_1, \dots, t_{n_i})$, so ist $\llbracket t \rrbracket_{\mathfrak{A}} = g_i(\llbracket t_1 \rrbracket_{\mathfrak{A}}, \dots, \llbracket t_{n_i} \rrbracket_{\mathfrak{A}})$

Nicht jedes Element des Universums einer Σ -Algebra \mathfrak{A} muss zwangsläufig aus einem Term t durch $\llbracket t \rrbracket_{\mathfrak{A}}$ bestimmbar sein. Ist ein Element auf diese Weise bestimmbar, so bezeichnen wir es als *Termelement*. Wenn wir im folgenden in dieser Arbeit schreiben „Sei eine Σ -Algebra gegeben“, so wird, falls nicht näher angegeben, eine Signatur Σ als gegeben vorausgesetzt.

Definition 7 (Termelement, termerzeugte Algebra). Sei eine Σ -Algebra $\mathfrak{A} = (U, g_1, \dots, g_k)$ gegeben. Dann ist $u \in U$ ein *Termelement*, wenn es einen Σ -Term t gibt, so dass $\llbracket t \rrbracket_{\mathfrak{A}} = u$.

Ist jedes Element aus U ein Termelement, so ist \mathfrak{A} eine *termerzeugte Algebra*.

Die Menge aller Elemente aus U , die keine Termelemente sind, werden umgangssprachlich auch als *junk* bezeichnet. Termerzeugte Algebren werden auch *no-junk-Algebren* genannt.

Definition 8 (Unteralgebra). Seien eine Signatur $\Sigma = (f_1, \dots, f_k, n_1, \dots, n_k)$ und zwei Σ -Algebren $\mathfrak{A} = (U, g_1, \dots, g_k)$, $\mathfrak{A}' = (V, h_1, \dots, h_k)$ gegeben. \mathfrak{A}' ist eine *Unteralgebra von \mathfrak{A}'* , wenn gilt:

- $V \subseteq U$
- $h_i(a_1, \dots, a_{n_i}) = g_i(a_1, \dots, a_{n_i})$ für alle $a_1, \dots, a_{n_i} \in V$.

Es ist leicht zu zeigen, dass eine Σ -Algebra \mathfrak{A} und eine Unteralgebra von \mathfrak{A} alle Σ -Terme gleich interpretieren.

Definition 9 (Termerzeugte Unteralgebra). Seien die Σ -Algebren $\mathfrak{A} = (U, g_1, \dots, g_k)$ und $\mathfrak{A}' = (V, h_1, \dots, h_k)$ gegeben. \mathfrak{A}' ist eine *termerzeugte Unteralgebra von \mathfrak{A}* , wenn \mathfrak{A}' termerzeugt und eine Unteralgebra von \mathfrak{A} ist.

Man kann recht einfach zeigen, dass zu jeder Σ -Algebra \mathfrak{A} genau eine termerzeugte Unteralgebra existiert. Diese kann durch Entfernen aller Nicht-Termelemente aus dem Universum von \mathfrak{A} und durch ein entsprechendes Einschränken aller Funktionen von \mathfrak{A} konstruiert werden.

1.2 Sequenzieller Algorithmus

Definition 10 (Initialisiertes Transitionssystem). Seien eine Menge \mathcal{S} , eine Menge $\mathcal{I} \subseteq \mathcal{S}$ und $\tau : \mathcal{S} \rightarrow \mathcal{S}$ gegeben. Dann ist $M = (\mathcal{S}, \mathcal{I}, \tau)$ ein *initialisiertes Transitionssystem*. \mathcal{S} ist die *Zustandsmenge*, \mathcal{I} die *Anfangszustandsmenge* und τ die *Transformation* von M .

Definition 11 (Ablauf eines initialisierten Transitionssystems). Sei $M = (\mathcal{S}, \mathcal{I}, \tau)$ ein initialisiertes Transitionssystem. Eine Folge $(s_n)_{n \in \mathbb{N}}$ mit $s_n \in \mathcal{S}$ für alle $n \in \mathbb{N}$ ist ein *Ablauf von M* , wenn gilt:

- $s_0 \in \mathcal{I}$
- $s_{i+1} = \tau(s_i)$

Wir definieren nun eine besondere Klasse von Signaturen, die Klasse der ASM-Signaturen. Diese enthalten alle wichtigen booleschen Funktionssymbole, und das Funktionssymbol *undef*.

Definition 12 (ASM-Signatur). Eine Signatur der Form

$$\Sigma = (tt, ff, undef, \neg, \wedge, f_1, \dots, f_k, 0, 0, 0, 1, 2, n_1, \dots, n_k)$$

heißt *ASM-Signatur*.

Als Abkürzung schreiben wir $\Sigma = (f_1, \dots, f_k, n_1, \dots, n_k)_{ASM}$.

Nun definieren wir entsprechend zu den ASM-Signaturen die Klasse der ASM-Algebren.

Definition 13 (ASM-Algebra). Eine Algebra der Form

$$\mathfrak{A} = (U \cup \{true, false, undefined\}, true, false, undefined, NOT, AND, g_1, \dots, g_k)$$

$$\text{mit} \quad NOT(x) = \begin{cases} true & \text{falls } x = false \\ false & \text{sonst} \end{cases}$$

$$\text{und} \quad AND(x, y) = \begin{cases} true & \text{falls } x = y = true \\ false & \text{sonst} \end{cases}$$

heißt *ASM-Algebra*. Als Abkürzung schreiben wir $\mathfrak{A} = (g_1, \dots, g_k)_{ASM}$. Jede Algebra \mathfrak{A}' , die isomorph zu \mathfrak{A} ist, ist ebenfalls eine *ASM-Algebra*.

Zu jeder ASM-Algebra \mathfrak{A} gibt es eine ASM-Signatur Σ , so dass \mathfrak{A} eine Σ -Algebra ist. Durch die booleschen Funktionssymbole und ihre entsprechenden Interpretationen können in einer ASM-Algebra boolesche Terme gebildet und interpretiert werden. Des Weiteren enthalten sie das Element *undefined*, um partielle Funktionen in die ASM-Algebren einbetten zu können: Der Wert an nicht definierten Stellen einer Funktion ist dann *undefined*.

Definition 14 (Aktualisierung). Seien $\mathfrak{A} = (U, g_1, \dots, g_k)$ eine Σ -Algebra, f_i ein Funktionssymbol der Stelligkeit n_i aus Σ , $\bar{a} \in U^{n_i}$ und $b \in U$. Dann ist (f_i, \bar{a}, b) eine *Aktualisierung* von \mathfrak{A} .

Definition 15 (Inkonsistenz). Sei eine Σ -Algebra \mathfrak{A} und eine Menge von Aktualisierungen \mathcal{A} von \mathfrak{A} gegeben. \mathcal{A} ist *inkonsistent* genau dann, wenn es in Σ ein Funktionssymbol f , eine Stelle \bar{a} von $f_{\mathfrak{A}}$ und Elemente b, b' des Universums von \mathfrak{A} mit $b \neq b'$ gibt, so dass $(f, \bar{a}, b) \in \mathcal{A}$ und $(f, \bar{a}, b') \in \mathcal{A}$. \mathcal{A} ist *konsistent*, wenn \mathcal{A} nicht inkonsistent ist.

Definition 16 (Anwendung von Aktualisierungen). Seien eine Signatur $\Sigma = (f_1, \dots, f_k, n_1, \dots, n_k)$, eine Σ -Algebra $\mathfrak{A} = (U, g_1, \dots, g_k)$ und eine Aktualisierung (f_i, \bar{a}, b) von \mathfrak{A} gegeben. (f_i, \bar{a}, b) wird auf \mathfrak{A} *angewendet*, indem die Funktion g_i an der Stelle \bar{a} auf b gesetzt wird.

Eine konsistente Menge \mathcal{A} von Aktualisierungen wird auf einer Σ -Algebra \mathfrak{A} *angewendet*, indem gleichzeitig jede Aktualisierung aus \mathcal{A} auf \mathfrak{A} angewendet wird. Eine inkonsistente Menge \mathcal{A} wird auf \mathfrak{A} *angewendet*, indem \mathfrak{A} unverändert bleibt. Die resultierende Algebra wird als $\mathfrak{A} + \mathcal{A}$ geschrieben.

Definition 17 (Algebraisches Transitionssystem). Sei $M = (\mathcal{S}, \mathcal{I}, \tau)$ ein initialisiertes Transitionssystem. M ist ein *algebraisches Transitionssystem* genau dann, wenn folgende Eigenschaften erfüllt sind:

1. Es gibt eine ASM-Signatur Σ , so dass $\mathcal{S} \subseteq \text{Alg}_{\Sigma}$. Σ heißt die *Signatur* von M .
2. Für jedes $\mathfrak{A} \in \mathcal{S}$ sind die Universen von \mathfrak{A} und $\tau_M(\mathfrak{A})$ gleich.
3. \mathcal{S} und \mathcal{I} sind abgeschlossen bezüglich Isomorphismus.
4. Seien $\mathfrak{A}, \mathfrak{B} \in \mathcal{S}$. Wenn Φ ein Isomorphismus zwischen \mathfrak{A} und \mathfrak{B} ist, so ist Φ ebenfalls ein Isomorphismus zwischen $\tau(\mathfrak{A})$ und $\tau(\mathfrak{B})$.

τ nennen wir auch eine *algebraische Transformation*. Abläufe eines algebraischen Transitionssystem werden *algebraische Abläufe* genannt.

Aus der Definition des initialisiertes Transitionssystem und aus Punkt 1. folgt, dass auch \mathcal{I} eine Teilmenge von Alg_{Σ} sein muss. Weiter ist obigen Definitionen ist in einem Ablauf eines algebraischen Transitionssystem die Signatur und das Universum konstant. Die einzige Änderung, die τ an einem Zustand \mathfrak{A} von \mathcal{A} vornehmen kann, ist also die Aktualisierung der Funktionen von \mathfrak{A} an bestimmten Stellen auf bestimmte Werte. Somit kann eine solche Transformation entweder direkt durch eine Funktion wie τ oder durch eine Menge von Aktualisierungen zu jedem Zustand angegeben werden:

Definition 18 (Aktualisierungsmenge Δ_M). Seien $M = (\mathcal{S}, \mathcal{I}, \tau)$ ein algebraisches Transitionssystem und $\mathfrak{A} \in \mathcal{S}(A)$. Dann ist $\Delta_M(\mathfrak{A})$ folgendermaßen definiert:

$$\Delta_M(\mathfrak{A}) = \{(f, (u_1, \dots, u_n), u_0) \mid \llbracket f \rrbracket_{\mathfrak{A}}(u_1, \dots, u_n) \neq \llbracket f \rrbracket_{\tau(\mathfrak{A})}(u_1, \dots, u_n) \\ \text{und } u_0 = \llbracket f \rrbracket_{\tau(\mathfrak{A})}(u_1, \dots, u_n)\}.$$

Es ist leicht zu sehen, dass $\tau(\mathfrak{A}) = \mathfrak{A} + \Delta_M(\mathfrak{A})$ gilt, da $\Delta_M(\mathfrak{A})$ gerade die Stellen von \mathfrak{A} aktualisiert, die \mathfrak{A} von $\tau(\mathfrak{A})$ unterscheidet. Damit ist klar, dass Δ_M die gesamte Dynamik des algebraischen Transitionssystems M charakterisiert. Wir definieren nun den Begriff des *Algorithmischen Transitionssystems*, der an die Funktion Δ_M eine zusätzliche Forderung stellt:

Definition 19 (Algorithmisches Transitionssystem). Sei $M = (\mathcal{S}, \mathcal{I}, \tau)$ ein algebraisches Transitionssystem. M ist ein *algorithmisches Transitionssystem* genau dann, wenn es eine endliche Menge von Σ -Termen T gibt, so dass gilt: Seien $\mathfrak{A}, \mathfrak{B} \in \mathcal{S}$. Wenn $\llbracket t \rrbracket_{\mathfrak{A}} = \llbracket t \rrbracket_{\mathfrak{B}}$ für alle $t \in T$, dann ist $\Delta_M(\mathfrak{A}) = \Delta_M(\mathfrak{B})$. T heißt dann *charakteristisch für M* .

τ wird dann auch *algorithmische Transformation* genannt. Abläufe eines algorithmischen Transitionssystems werden *algorithmische Abläufe* genannt.

Algorithmische Transitionssysteme sind also algebraische Transitionssysteme, die eine endliche, charakteristische Termmenge besitzen: Ergeben alle Terme der charakteristischen Termmenge auf zwei Algebren die gleichen Elemente, so muss die Funktion Δ_M für beide Algebren die gleichen Aktualisierungen erzeugen.

1.3 Abstract-State Machine

Wenn im folgenden der Begriff *Signatur* oder *Algebra* verwendet wird, dann sei damit implizit immer eine ASM-Signatur bzw. eine ASM-Algebra gemeint.

Definition 20 (Σ -Bedingung). Sei eine Signatur Σ gegeben. Wir definieren induktiv:

- Seien t und s Σ -Terme. Dann sind $t = s$ und $\neg(t = s)$ Σ -Bedingungen.
- Seien β_1 und β_2 Σ -Bedingungen. Dann ist $(\beta_1 \wedge \beta_2)$ eine Σ -Bedingung.

Σ -Bedingungen sind also rein syntaktische Objekte, die aber wiederum in einer Σ -Algebra interpretiert werden können:

Definition 21 (Interpretation einer Σ -Bedingung). Sei eine Signatur Σ , eine Σ -Algebra \mathfrak{A} und eine Σ -Bedingung β gegeben. Dann ist die *Interpretation* $\llbracket \beta \rrbracket_{\mathfrak{A}}$ von β durch \mathfrak{A} folgendermaßen definiert:

- Ist β eine Σ -Bedingung der Form $t = s$, dann ist

$$\llbracket \beta \rrbracket_{\mathfrak{A}} := \begin{cases} true & \text{falls } \llbracket t \rrbracket_{\mathfrak{A}} = \llbracket s \rrbracket_{\mathfrak{A}} \\ false & \text{sonst.} \end{cases}$$

- Ist β eine Σ -Bedingung der Form $\neg(t = s)$, dann ist

$$\llbracket \beta \rrbracket_{\mathfrak{A}} := NOT(\llbracket t = s \rrbracket_{\mathfrak{A}}).$$

- Ist β eine Σ -Bedingung der Form $(\beta_1 \wedge \beta_2)$, dann ist

$$\llbracket \beta \rrbracket_{\mathfrak{A}} := AND(\llbracket \beta_1 \rrbracket_{\mathfrak{A}}, \llbracket \beta_2 \rrbracket_{\mathfrak{A}}).$$

Σ -Bedingungen werden also in jeder Algebra als *true* oder als *false* interpretiert.

Definition 22 (Σ -Regel). Sei eine ASM-Signatur Σ gegeben. Wir definieren induktiv:

- Seien t_0, \dots, t_n Σ -Terme und sei f ein n -stelliges Funktionssymbol aus Σ , das kein konstantes Funktionssymbol ist. Dann ist $f(t_1, \dots, t_n) := t_0$ eine Σ -Regel. Regeln dieser Form heißen *Zuweisungsregeln*.
- Ist β eine Σ -Bedingung und ist R eine Zuweisungsregel, so ist **if β then R** eine Σ -Regel. Regeln dieser Form heißen *bedingte Zuweisungsregeln*.
- Sind R_1, \dots, R_n bedingte Zuweisungsregeln, so ist **par $R_1 \dots R_n$ endpar** eine Σ -Regel. Regeln dieser Form heißen *ASM-Programme*.

Σ -Regeln sind wie die Σ -Terme rein syntaktisch definiert. Durch eine Σ -Algebra können Σ -Regeln jedoch semantisch interpretiert werden.

Definition 23 (Aktualisierungsmenge Δ_R). Seien eine ASM-Signatur Σ , eine Σ -Algebra $\mathfrak{A} = (U, g_1, \dots, g_k)_{ASM}$ und eine Σ -Regel R gegeben.

- Falls $R = f(t_1, \dots, t_n) := t_0$, so ist

$$\Delta_R(\mathfrak{A}) := \{(f, (\llbracket t_1 \rrbracket_{\mathfrak{A}}, \dots, \llbracket t_n \rrbracket_{\mathfrak{A}}), \llbracket t_0 \rrbracket_{\mathfrak{A}})\}$$

- Falls $R = \text{if } \beta \text{ then } Z$, so ist

$$\Delta_R(\mathfrak{A}) := \begin{cases} \Delta_Z(\mathfrak{A}) & \text{falls } \llbracket \beta \rrbracket_{\mathfrak{A}} = true \\ \emptyset & \text{sonst} \end{cases}$$

- Falls $R = \text{par } R_1 \dots R_n \text{ endpar}$, so ist

$$\Delta_R(\mathfrak{A}) := \bigcup_{i=1}^n \Delta_{R_i}(\mathfrak{A})$$

Eine Σ -Regel R wird dann auf einer Σ -Algebra \mathfrak{A} *ausgeführt*, indem die zugehörige Aktualisierungsmenge bestimmt und auf \mathfrak{A} angewendet wird: $\mathfrak{A}' = \mathfrak{A} + \Delta_R(\mathfrak{A})$. Damit lässt sich die Transformation zu einer Σ -Regel definieren:

Definition 24 (Transformation τ_R). Seien eine ASM-Signatur Σ , eine Σ -Regel R und eine Σ -Algebra \mathfrak{A} gegeben. Dann ist $\tau_R(\mathfrak{A}) = \mathfrak{A} + \Delta_R(\mathfrak{A})$.

Es ist leicht zu zeigen, dass für eine beliebige Σ -Regel R die Transformation τ_R algorithmisch ist. Umgekehrt gilt das ASM-Theorem:

Satz 1 (ASM-Theorem). *Für jedes algorithmische Transitionssystem $M = (\mathcal{S}, \mathcal{I}, \tau)$ existiert ein ASM-Programm R , so dass $\tau(\mathfrak{A}) = \tau_R(\mathfrak{A})$ für alle $\mathfrak{A} \in \mathcal{S}$.*

Der erste Beweis des ASM-Theorems erfolgte in [1]. Einen auf die hier verwendeten Begriffe angepassten Beweis ist in [2] zu finden.

2 Gegenbeispiele zum Algorithmusbegriff

2.1 Nicht-algorithmische Abläufe

Wir haben bereits die Begriffe *Ablauf* und *algorithmischer Ablauf* definiert. Um nun den Begriff der algorithmischen Abläufe näher zu erläutern, werden im folgenden einige Beispiele von Abläufen beschrieben, die keine algorithmischen Abläufe sind. Wir betrachten dabei nur Abläufe von Transitionssystemen, die Algebren mit gleicher Signatur und gleichem Universum enthalten. Andere Abläufe sind trivialerweise nicht algorithmisch.

Beispiel 1. Seien $\Sigma = (a, b, 0, 0)_{ASM}$ und die Σ -Algebren $\mathfrak{A} = (\{0, 1\}, 0, 1)$ und $\mathfrak{B} = (\{0, 1\}, 1, 0)$ gegeben. Dann ist der Ablauf $(\mathfrak{A}, \mathfrak{B}, \mathfrak{B}, \dots)$ nicht algorithmisch.

Um sich dies intuitiv klar zu machen, beachte man, dass ein algorithmisches Transitionssystem auf den aktuellen Zustand nur über Σ -Terme zugreifen kann, es besitzt keinerlei weitere Informationen über das Universum. Es kann zwar den Wert des Symbols a bestimmen, kann aber nicht entscheiden, ob es sich um eine 0 oder um eine 1 handelt. Dazu würde man einen Term benötigen, der in beiden Algebren \mathfrak{A} und \mathfrak{B} als 0 oder als 1 interpretiert werden kann. Diesen gibt es aber nicht. Somit kann der Zustand \mathfrak{A} nicht von \mathfrak{B} unterscheiden werden, um für \mathfrak{A} eine Aktualisierung zu berechnen und für \mathfrak{B} keine.

Formal zeigt Beispiel 1 einen direkten Widerspruch zum Punkt 4 der Definition eines algebraischen Transitionssystems: $\Phi = \{0 \mapsto 1, 1 \mapsto 0\}$ ist ein Isomorphismus zwischen \mathfrak{A} und \mathfrak{B} , aber kein Isomorphismus zwischen $\tau(\mathfrak{A}) = \mathfrak{B}$ und $\tau(\mathfrak{B}) = \mathfrak{B}$. Dieser Ablauf ist also nicht nur nicht-algorithmisch, sondern auch nicht-algebraisch.

Nun geben wir ein Beispiel für einen Ablauf an, der zwar algebraisch ist, aber nicht algorithmisch:

Beispiel 2. Seien $\Sigma = (s, i, 0, 1)_{ASM}$ und die Σ -Algebren $\mathfrak{A} = (\mathbb{N}, 0, \oplus)$ und $\mathfrak{B} = (\mathbb{N}, 1, \oplus)$ mit $\oplus(x) = x + 1$ gegeben. Dann ist der Ablauf $(\mathfrak{A}, \mathfrak{B}, \mathfrak{B}, \dots)$ nicht algorithmisch.

Die Intuition hinter diesem Gegenbeispiel ist ganz ähnlich zu der in Beispiel 1: Es kann zwar der Wert des Symbols i bestimmt werden, es kann aber nicht anhand von Σ -Termen entschieden werden, ob es sich dabei um eine 0 oder um eine 1 handelt. Somit kann ein algorithmisches Transitionssystem nicht für \mathfrak{A} eine Aktualisierung berechnen und für \mathfrak{B} keine. Für einen formalen Nachweis beweisen wir zunächst zwei Lemmata:

Lemma 2. Seien ein algebraisches Transitionssystem $M = (\mathcal{S}, \mathcal{I}, \tau)$, zwei Algebren $\mathfrak{A}, \mathfrak{B} \in \mathcal{S}$ und ein Isomorphismus Φ zwischen \mathfrak{A} und \mathfrak{B} gegeben.

Dann gilt

$$(f, (u_1, \dots, u_n), u_0) \in \Delta_M(\mathfrak{A})$$

$$\text{gdw. } (f, (\Phi(u_1), \dots, \Phi(u_n)), \Phi(u_0)) \in \Delta_M(\mathfrak{B}).$$

Kürzer gefasst bedeutet dies, dass für isomorphe Algebren eines algebraischen Transitionssystems isomorphe Aktualisierungsmengen berechnet werden.

Beweis.

$$(f, (u_1, \dots, u_n), u_0) \in \Delta_A(\mathfrak{A})$$

$$\text{gdw. } f_{\mathfrak{A}}(u_1, \dots, u_n) \neq f_{\tau(\mathfrak{A})}(u_1, \dots, u_n) = u_0$$

$$\text{gdw. } f_{\mathfrak{B}}(\Phi(u_1), \dots, \Phi(u_n)) \neq f_{\tau(\mathfrak{B})}(\Phi(u_1), \dots, \Phi(u_n)) = \Phi(u_0)$$

$$\text{(nach Punkt 4 der Def. algebraisches Transitionssystem)}$$

$$\text{gdw. } (f, (\Phi(u_1), \dots, \Phi(u_n)), \Phi(u_0)) \in \Delta_A(\mathfrak{B})$$

□

Lemma 3. *Sei ein algorithmisches Transitionssystem $M = (\mathcal{S}, \mathcal{I}, \tau)$ gegeben. Seien außerdem zwei Algebren $\mathfrak{A}, \mathfrak{B} \in \mathcal{S}$ gegeben und seien \mathfrak{A}' und \mathfrak{B}' die termerzeugten Unteralgebren zu \mathfrak{A} und \mathfrak{B} . Ist Φ ein Isomorphismus zwischen \mathfrak{A}' und \mathfrak{B}' , dann gilt*

$$(f, (u_1, \dots, u_n), u_0) \in \Delta_M(\mathfrak{A})$$

$$\text{gdw. } (f, (\Phi(u_1), \dots, \Phi(u_n)), \Phi(u_0)) \in \Delta_M(\mathfrak{B}).$$

Die Voraussetzung an die Algebren \mathfrak{A} und \mathfrak{B} ist in diesem Lemma schwächer als in Lemma 2: Als Voraussetzung wird nicht verlangt, dass \mathfrak{A} und \mathfrak{B} isomorph sind, es genügt, einen Isomorphismus auf ihren termerzeugten Unteralgebren anzugeben. Dieses Lemma läßt sich dafür nicht mehr wie Lemma 2 für algebraische Transitionssysteme beweisen, man benötigt die zusätzlichen Eigenschaften eines algorithmischen Transitionssystems.

Beweis. Es folgt direkt aus der Definition der termerzeugten Unteralgebra, dass \mathfrak{A} und \mathfrak{A}' alle Terme gleich interpretieren. Damit werden nach Definition des algorithmischen Transitionssystems für \mathfrak{A} und \mathfrak{A}' die gleichen Aktualisierungsmengen berechnet, analog gilt dies für \mathfrak{B} und \mathfrak{B}' . Damit lässt sich schreiben:

$$(f, (u_1, \dots, u_n), u_0) \in \Delta_A(\mathfrak{A})$$

$$\text{gdw. } (f, (u_1, \dots, u_n), u_0) \in \Delta_A(\mathfrak{A}')$$

$$\text{gdw. } (f, (\Phi(u_1), \dots, \Phi(u_n)), \Phi(u_0)) \in \Delta_A(\mathfrak{B}') \text{ (nach Lemma 1)}$$

$$\text{gdw. } (f, (\Phi(u_1), \dots, \Phi(u_n)), \Phi(u_0)) \in \Delta_A(\mathfrak{B})$$

□

Betrachten wir nun wieder Beispiel 2: Seien $\mathfrak{A} = (\mathbb{N}, 0, \oplus)$ und $\mathfrak{B} = (\mathbb{N}, 1, \oplus)$ mit $\oplus(x) = x + 1$ gegeben. Wir nehmen nun an, es gäbe ein algorithmisches Transitionssystem A , der den Ablauf $(\mathfrak{A}, \mathfrak{B}, \mathfrak{B}, \dots)$ berechnet. Die termerzeugte Unteralgebra zu \mathfrak{A} ist \mathfrak{A} selbst, zu \mathfrak{B} ist die termerzeugte Unteralgebra $\mathfrak{B}' = (\mathbb{N} \setminus \{1\}, 1, \oplus')$ mit $\oplus'(x) = x + 1$. \mathfrak{A} und \mathfrak{B}' sind mit $\Phi(x) = x + 1$ isomorph, nach Lemma 3 können die Aktualisierungen $\Delta_M(\mathfrak{A})$ durch Φ auf $\Delta_M(\mathfrak{B})$ abgebildet werden. Der obige Ablauf dagegen bestimmt für \mathfrak{A} eine Aktualisierung, für \mathfrak{B} aber keine, wodurch sich ein Widerspruch ergibt.

Beispiel 3. Seien \mathfrak{A} und \mathfrak{B} wie in Beispiel 2 gegeben. Dann ist der Ablauf $(\mathfrak{B}, \mathfrak{A}, \dots)$ nicht algorithmisch.

Intuitiv besteht das Problem darin, dass Elemente des Universums, die nicht durch einen Term bestimmt werden können, auch nach einem Transformationsschritt nicht mehr bestimmt werden können: In \mathfrak{B} ist 0 kein Termelement, in \mathfrak{A} dagegen schon.

Für einen formalen Nachweis beweisen wir erneut zwei Lemmata. Zunächst aber noch ein neuer Begriff:

Definition 25 (kritisches Element). Seien ein algorithmisches Transitionssystem $M = (\mathcal{S}, \mathcal{I}, \tau)$ und eine Algebra $\mathfrak{A} \in \mathcal{S}$ gegeben. Nach Definition existiert für M dann eine charakteristische Termmenge T . Dann ist $\llbracket t \rrbracket_{\mathfrak{A}}$ für $t \in T$ ein *kritisches Element von M* .

Lemma 4. Seien ein algorithmisches Transitionssystem $M = (\mathcal{S}, \mathcal{I}, \tau)$ und eine Algebra $\mathfrak{A} \in \mathcal{S}$ gegeben. Sei weiter $\delta = (f, (u_1, \dots, u_n), u_0) \in \Delta_M(\mathfrak{A})$ gegeben. Dann sind u_0, \dots, u_n und $f(u_1, \dots, u_n)$ *kritische Elemente von M* .

Beweis. Wir nehmen nun an, dass es ein $i \in \{0, \dots, n\}$ gibt, so dass u_i kein kritisches Element von M ist. Weiter betrachten wir die Algebra \mathfrak{A}' , die aus \mathfrak{A} entsteht, indem das Element u_i durch das Element $\diamond \notin U$ ersetzt wird. \mathfrak{A} und \mathfrak{A}' sind isomorph mit $\Phi(x) = x$ für $x \neq u_i$ und $\Phi(u_i) = \diamond$ und interpretieren alle charakteristischen Terme gleich. Damit werden für \mathfrak{A} und \mathfrak{A}' die gleichen Aktualisierungsmengen berechnet, es gilt also $(f, (u_1, \dots, u_n), u_0) \in \Delta_{\mathfrak{A}}(\mathfrak{A}')$. u_i ist aber kein Element von $\mathfrak{A}' \rightarrow$ Widerspruch.

Wir nehmen nun an, dass $f(u_1, \dots, u_n)$ kein kritisches Element ist. Dann betrachten wir eine Algebra \mathfrak{A}' die aus der Algebra \mathfrak{A} entsteht, indem die Funktion f durch eine Funktion f' ersetzt wird, die an der Stelle (u_1, \dots, u_n) auf u_0 gesetzt wird und sonst wie f abbildet. \mathfrak{A} und \mathfrak{A}' interpretieren alle charakteristischen Terme gleich, damit ist also die Aktualisierung $(f, (u_1, \dots, u_n), u_0)$ in $\Delta_M(\mathfrak{A}')$ enthalten. Dies ist jedoch ein Widerspruch dazu, dass dann $f(u_1, \dots, u_n) \neq u_0$ ist. \square

Korollar 5. Seien ein algorithmisches Transitionssystem $M = (\mathcal{S}, \mathcal{I}, \tau)$ und eine Algebra $\mathfrak{A} \in \mathcal{S}$ gegeben. Sei weiter $\delta = (f, (u_1, \dots, u_n), u_0) \in \Delta_M(\mathfrak{A})$ gegeben. Dann sind u_0, \dots, u_n *Termlemente von \mathfrak{A}* .

Lemma 6. *Seien ein algorithmisches Transitionssystem $M = (\mathcal{S}, \mathcal{I}, \tau)$, eine Algebra $\mathfrak{A} \in \mathcal{S}$ und ein Term t gegeben. Dann gibt es einen Term t' , so dass $\llbracket t \rrbracket_{\tau(\mathfrak{A})} = \llbracket t' \rrbracket_{\mathfrak{A}}$.*

Dies bedeutet, dass jeder Wert, der in $\tau(\mathfrak{A})$ aus einen Term interpretierbar ist, auch schon in \mathfrak{A} aus einen Term interpretiert werden kann.

Beweis. Wir beweisen die Aussage nun per Induktion über Teilterme:

Induktionsanfang: $t = f$ für ein 0-stelliges Funktionssymbol f

Fall 1: $\llbracket t \rrbracket_{\tau(\mathfrak{A})} = \llbracket t \rrbracket_{\mathfrak{A}}$
Dann ist $t' = f$.

Fall 2: $\llbracket t \rrbracket_{\tau(\mathfrak{A})} \neq \llbracket t \rrbracket_{\mathfrak{A}}$
Dann enthält $\Delta_M(\mathfrak{A})$ eine Aktualisierung $(f, (), \llbracket t \rrbracket_{\tau(\mathfrak{A})})$. Nach Lemma 3 ist damit $\llbracket t \rrbracket_{\tau(\mathfrak{A})}$ ein Termelement von \mathfrak{A} .

Induktionsschritt: Sei $t = g(t_1, \dots, t_n)$ für ein n -stelliges Funktionssymbol g und es gelte die Behauptung für die Teilterme t_1, \dots, t_n . Es gibt also Terme t'_1, \dots, t'_n , so dass $\llbracket t_i \rrbracket_{\tau(\mathfrak{A})} = \llbracket t'_i \rrbracket_{\mathfrak{A}}$ für $i = 1, \dots, n$.

Fall 1: $\llbracket g(t_1, \dots, t_n) \rrbracket_{\tau(\mathfrak{A})} = \llbracket g(t'_1, \dots, t'_n) \rrbracket_{\mathfrak{A}}$
Dann ist $t' = g(t'_1, \dots, t'_n)$.

Fall 2: $\llbracket g(t_1, \dots, t_n) \rrbracket_{\tau(\mathfrak{A})} \neq \llbracket g(t'_1, \dots, t'_n) \rrbracket_{\mathfrak{A}}$
Dann enthält $\Delta_M(\mathfrak{A})$ eine Aktualisierung $(f, (u_1, \dots, u_n), \llbracket g(t_1, \dots, t_n) \rrbracket_{\tau(\mathfrak{A})})$. Nach Lemma 3 ist $\llbracket g(t_1, \dots, t_n) \rrbracket_{\tau(\mathfrak{A})}$ damit ein Termelement von \mathfrak{A} .

□

Aus Lemma 5 lässt sich nun direkt das folgende Korollar folgern:

Korollar 7. *Seien ein algorithmisches Transitionssystem $M = (\mathcal{S}, \mathcal{I}, \tau)$ und eine Algebra $\mathfrak{A} \in \mathcal{S}$ mit dem Universum U gegeben. Falls $u \in U$ in $\tau(\mathfrak{A})$ ein Termelement ist, so ist u in \mathfrak{A} ebenfalls ein Termelement.*

Kommen wir nun wieder zu Beispiel 4: Seien $\mathfrak{A} = (\mathbb{N}, 0, \oplus)$ und $\mathfrak{B} = (\mathbb{N}, 1, \oplus)$ mit $\oplus(x) = x + 1$ gegeben. Die 0 ist in $\tau(\mathfrak{B}) = \mathfrak{A}$ ein Termelement, aber nicht in \mathfrak{B} , damit kann nach dem Korollar $(\mathfrak{B}, \mathfrak{A}, \dots)$ nicht algorithmisch sein.

2.2 Nicht-algorithmische Transformationssysteme

Betrachten wir nun noch einmal das Beispiel 2 aus dem vorherigen Abschnitt: Seien $\mathfrak{A} = (\mathbb{N}, 0, \oplus)$ und $\mathfrak{B} = (\mathbb{N}, 1, \oplus)$ mit $\oplus(x) = x + 1$ gegeben. Sei $M = (\{\{\mathfrak{A}, \mathfrak{B}\}\}, [\{\mathfrak{A}\}], \tau)$ ein initialisiertes Transitionssystem, mit $\tau(\mathfrak{A}) = \mathfrak{B}$

und $\tau(\mathfrak{B}) = \mathfrak{B}$. Zur Erinnerung: $[\{\mathfrak{A}, \mathfrak{B}\}]$ bezeichnet den isomorphen Abschluss der Menge von Algebren $\{\mathfrak{A}, \mathfrak{B}\}$. Damit erfüllt M die Forderungen an ein algebraisches Transitionssystem und erzeugt den Ablauf $(\mathfrak{A}, \mathfrak{B}, \mathfrak{B}, \dots)$. Das besondere an diesem Beispiel ist also, dass es alle Forderungen an ein algorithmisches Transitionssystem bis auf die nach der Existenz einer charakteristischen Termmenge erfüllt.

Schwächt man die Forderung nach einer charakteristischen Termmenge ab, indem man auch unendliche Termmengen als charakteristisch zulässt, so erfüllt M auch diese Forderung nicht. Zu M existiert überhaupt keine charakteristische Termmenge, weder endlich noch unendlich. Nun betrachten wir ein Beispiel, in dem es eine solche unendliche charakteristische Termmenge gibt, aber keine endliche.

Beispiel 4. Sei $\Sigma = (i, s, 0, 1)$. Sei $\mathfrak{A}_n = (\mathbb{N}, 0, \oplus_n)$ mit

$$\oplus_n(x) = \begin{cases} x + 1 & \text{falls } x < n \\ 0 & \text{sonst.} \end{cases}$$

Sei weiterhin $\mathfrak{B}_n = (\mathbb{N}, n, \oplus)$ für $n \in \mathbb{N}$ mit $\oplus(x) = x + 1$. Sei nun $\mathcal{S} = [\{\mathfrak{A}_n, \mathfrak{B}_n\}_{n \in \mathbb{N}}]$. Nun sei $\tau : \mathcal{S} \rightarrow \mathcal{S}$ so, dass $\tau(\mathfrak{A}_n) = \mathfrak{A}_n$ für $n \in \mathbb{N}$ und $\tau(\mathfrak{B}_n) = \mathfrak{B}_{n+1}$. Dann ist $M = (\mathcal{S}, \emptyset, \tau)$ kein algorithmisches Transitionssystem.

Der Algorithmus scheint zunächst sehr einfach: Für \mathfrak{A}_n wird keine Aktualisierung berechnet, für \mathfrak{B}_n wird nur eine Aktualisierung berechnet, die den Wert für das Funktionssymbol i auf $n + 1$ setzt. Dass M trotzdem nicht algorithmisch ist, liegt daran, dass M die Algebren $\{\mathfrak{A}_n\}_{n \in \mathbb{N}}$ von der Algebra \mathfrak{B}_0 nur durch unendlich viele Terme unterscheiden kann. Eine charakteristische Termmenge für M wäre $T = \{s^n(i)\}_{n \in \mathbb{N} \setminus \{0\}}$: Wird einer der Terme aus T zu 0 interpretiert, so handelt es sich um eine Algebra \mathfrak{A}_n und es wird keine Aktualisierung berechnet. In allen anderen Fällen wird die Aktualisierungsregel $i := i + 1$ angewendet und genau eine Aktualisierung berechnet.

Wir zeigen nun, dass es keine endliche charakteristische Termmenge für M geben kann: Angenommen es gibt eine endliche charakteristische Termmenge T . Aus dieser Menge bestimmen wir das größte $m \in \mathbb{N}$, so dass $s^m(i)$ als Teilterm in einem Term aus T enthalten ist. Dies ist möglich, da T endlich ist. Wir betrachten nun die Algebra \mathfrak{A}_m . Alle Teilterme aus Termen von T der Form $s^n(i)$, $n \in \mathbb{N}$, werden in \mathfrak{B}_0 und \mathfrak{A}_m gleich interpretiert. Dies kann man mit dem Argument $n \leq m$, da m maximal gewählt wurde, und per Induktion über n nachweisen. Teilterme aus T , deren äußeres Funktionssymbol s ist, und nicht die Form $s^n(i)$ haben, werden in \mathfrak{B}_0 und \mathfrak{A}_m zu *undefined* interpretiert. Per Induktion über die Teilterme läßt sich dadurch zeigen, dass alle anderen Teilterme aus T in \mathfrak{B}_0 und \mathfrak{A}_m gleich interpretiert werden. Damit werden also alle Terme in T gleich interpretiert und

für \mathfrak{B}_0 und \mathfrak{A}_m werden die gleichen Aktualisierungsmengen berechnet, da T charakteristisch für M war. Damit ergibt sich ein Widerspruch.

Dieses Beispiel hat auch gezeigt, dass eine unendliche charakteristische Menge nötig sein kann, obwohl die Aktualisierungsmenge in jedem Fall endlich ist.

3 Ein syntaktisches Rechenmodell

Um das Verständnis für die Mächtigkeit Algorithmischer Transitionssysteme zu erhöhen, werden wir in diesem Abschnitt versuchen, sie in die Berechenbarkeitstheorie einzuordnen.

Laut dem ASM-Theorem (Satz 1) existiert zu jedem algorithmischen Transitionssystem $M = (\mathcal{S}, \mathcal{I}, \tau)$ ein ASM-Programm R , so dass für jeden Zustand \mathfrak{A} $\Delta_M(\mathfrak{A}) = \Delta_R(\mathfrak{A})$ ist. Dies bedeutet also, dass die gesamte Dynamik von M bereits durch das endliche, syntaktische ASM-Programm R charakterisiert wird. Das Programm R könnte also die Eingabe einer berechenbaren Funktion sein, die dann die Dynamik von M berechnet. Eine Schwierigkeit von ASM's dabei ist jedoch, dass sie auf semantischen Strukturen, den Algebren operieren. Im Allgemeinen sind diese Strukturen nicht in einer berechenbaren Umgebung repräsentierbar.

Wir können nun die Frage stellen, ob wir das ASM-Programm R eines algorithmischen Transitionssystem M dazu verwenden können, die Dynamik von M zu berechnen, ohne dabei semantische Strukturen ändern zu müssen. Im folgenden werden wir ein solches Rechenmodell für algorithmische Transitionssysteme vorstellen. Den Ansatz dazu liefert uns folgende Definition:

Definition 26 (Urterm). Sei ein algorithmisches Transitionssystem $M = (\mathcal{S}, \mathcal{I}, \tau)$ und ein Zustand \mathfrak{A} von M gegeben. Ein Term t' ist ein *Urterm* zu einem Term t , wenn $\llbracket t \rrbracket_{\tau(\mathfrak{A})} = \llbracket t' \rrbracket_{\mathfrak{A}}$.

Das solche Urterme immer existieren, sichert uns Lemma 6 zu. Wir werden nachweisen, dass die Berechnung eines solchen Urtermes ohne Veränderung der Semantik von \mathfrak{A} möglich ist. Genauer gesagt werden wir zeigen, dass es eine berechenbare Funktion \mathcal{E} gibt, die zu einem Zustand \mathfrak{A} Terme auf ihre Urterme abbildet. Diese Funktion kann dann verwendet werden, die Dynamik eines algorithmischen Transitionssystem nachvollziehen: Um den Term t in der Algebra $\tau(\mathfrak{A})$ zu interpretieren, müssen wir $\tau(\mathfrak{A})$ nicht explizit gegeben haben. Wir rechnen den Term t ohne Veränderung der Semantik von \mathfrak{A} in einen Urterm t' um, der uns dann in \mathfrak{A} die gewünschte Interpretation liefert.

Wenn wir eine Funktion \mathcal{E} konstruieren möchten, die zu jedem Term der Algebra $\tau(\mathfrak{A})$ einen Urterm der Algebra \mathfrak{A} berechnet, so benötigen wir dazu die Information der Gleichheit zweier Terme in der Algebra \mathfrak{A} . Diese Gleichheit ist eine Äquivalenzrelation über allen Σ -Termen:

Definition 27 ($\sim_{\mathfrak{A}}$). Sei eine Σ -Algebra \mathfrak{A} und eine Menge von Σ -Termen T gegeben. Dann ist

$$\sim_{\mathfrak{A}} := \{(t, u) \in T_{\Sigma} \times T_{\Sigma} \text{ und } \llbracket t \rrbracket_{\mathfrak{A}} = \llbracket u \rrbracket_{\mathfrak{A}}\}.$$

Ein Problem dabei ist, dass die Algebra \mathfrak{A} auch unberechenbare Funktionen enthalten kann und die Termgleichheit damit unentscheidbar wäre. Im

allgemeinen müssen wir also davon ausgehen, dass dieses Entscheidungsproblem nicht unbedingt entscheidbar, also berechenbar sein muss. Trotzdem soll unserer berechenbaren Funktion \mathcal{E} die Information der Gleichheit zweier Terme in \mathfrak{A} zur Verfügung stehen. Wir erreichen dies, indem wir der Funktion \mathcal{E} als zweiten Parameter die Äquivalenzrelation $\sim_{\mathfrak{A}}$ zur Verfügung stellen. Der erste Parameter ist das ASM-Programm des algorithmischen Transitionssystems. Der letzte Parameter schließlich ist der Term, zu dem der Urterm bestimmt werden soll. Wir suchen also eine Funktion \mathcal{E} , die Urterme bestimmt und berechenbar ist bis auf das Problem der Termgleichheit von \mathfrak{A} . Das eine solche Funktion existiert, sichert folgender Satz zu:

Satz 8. *Sei eine Signatur Σ gegeben. Sei Π_{Σ} die Menge aller ASM-Programme über Σ . Dann existiert eine berechenbare Funktion $\mathcal{E} : \Pi_{\Sigma} \times (T_{\Sigma} \times T_{\Sigma}) \times T_{\Sigma} \rightarrow T_{\Sigma}$, so dass für jedes algorithmische Transitionssystem $M = (\mathcal{S}, \mathcal{I}, \tau)$ der Signatur Σ gilt: Sei R ein ASM-Programm von M und $\mathfrak{A} \in \mathcal{S}$ beliebig, dann gilt*

$$\llbracket t \rrbracket_{\tau(\mathfrak{A})} = \llbracket \mathcal{E}(R, \sim_{\mathfrak{A}}, t) \rrbracket_{\mathfrak{A}}.$$

Der Satz sagt also aus, dass es eine universelle, berechenbare Funktion \mathcal{E} gibt, die bei Kenntniss des ASM-Programms R eines algorithmischen Transitionssystems M und bei Kenntniss der Termgleichheit im Zustand \mathfrak{A} zu jedem Term einen Urterm bestimmt.

Beweis (unvollständig). Sei R das ASM-Programm **par** $R_1 \dots R_n$ **endpar**. Zunächst sein eine Funktion *Erfüllt* definiert, die bei Eingabe der Termrelation $\sim_{\mathfrak{A}}$ und einer Σ -Bedingung β feststellt, ob β im Zustand \mathfrak{A} als *true* interpretiert wird:

Wenn β Bedingung der Form $t = s$ ist:
 Wenn $t \sim_{\mathfrak{A}} s$:
 gib *true* zurück
 sonst
 gib *false* zurück
Wenn β Bedingung der Form $\neg(t = s)$ ist:
 Wenn $t \sim_{\mathfrak{A}} s$:
 gib *false* zurück
 sonst
 gib *true* zurück
Wenn β Bedingung der Form $(\beta_1 \wedge \beta_2)$ ist:
 Wenn $\text{Erfüllt}(\sim_{\mathfrak{A}}, \beta_1) = \text{Erfüllt}(\sim_{\mathfrak{A}}, \beta_2) = \text{true}$:
 gib *true* zurück
 sonst:
 gib *false* zurück

Dann sei die Funktion \mathcal{E} an der Stelle $(R, \sim_{\mathfrak{A}}, t)$ folgendermaßen definiert:

Berechnung der anzuwendenden Aktualisierungsregeln

$U = \emptyset$

Für jede bedingte Zuweisung R_i aus R :

Sei **if** β **then** Z die Regel R_i

Wenn $\text{Erfüllt}(\sim_{\mathfrak{A}}, \beta) = \text{true}$:

$U = U \cup \{Z\}$

Test auf Konsistenz

Falls es Regeln $f(t_1, \dots, t_n) := t_0$ und $f(t'_1, \dots, t'_n) := t'_0$ in U gibt

mit $t_i \sim_{\mathfrak{A}} t'_i$ und für $i = 1, \dots, n$ und $t_0 \not\sim_{\mathfrak{A}} t'_0$:

U ist inkonsistent, gib $f(\mathcal{E}(R, \sim_{\mathfrak{A}}, t_1), \dots, \mathcal{E}(R, \sim_{\mathfrak{A}}, t_n))$ zurück

Anwendung der Aktualisierung

Sei $f(t_1, \dots, t_n)$ der Term t

Falls es eine Regel $f(v_1, \dots, v_n) := v_0$ in U gibt,

mit $\mathcal{E}(R, \sim_{\mathfrak{A}}, t_i) \sim_{\mathfrak{A}} v_i$ für $i = 1, \dots, n$:

gib v_0 zurück

sonst:

gib $f(\mathcal{E}(R, \sim_{\mathfrak{A}}, t_1), \dots, \mathcal{E}(R, \sim_{\mathfrak{A}}, t_n))$ zurück

Anhand der Konstruktion erkennt man sofort die Berechenbarkeit von \mathcal{E} . Der formale und recht technische Beweis, dass diese Konstruktion von \mathcal{E} auch wirklich Urterme berechnet, wird an dieser Stelle nicht geführt, um den zeitlichen Rahmen der Studienarbeit nicht zu sprengen. \square

Anhand der Konstruktion von \mathcal{E} im Beweis erkennt man schnell die Gültigkeit folgenden Korollars:

Korollar 9. *Schränkt man den zweiten Parameter von \mathcal{E} auf Relationen $\sim_{\mathfrak{A}}$ ein, die klassisch berechenbar sind, dann ist auch \mathcal{E} klassisch berechenbar. Schränkt man den zweiten Parameter von \mathcal{E} auf Relationen $\sim_{\mathfrak{A}}$ ein, deren Algebra \mathfrak{A} nur Funktionen enthält, die klassisch berechenbar sind, dann ist auch \mathcal{E} klassisch berechenbar.*

Literatur

- [1] Y.Gurevich: Sequential Abstract-State Machines Capture Sequential Algorithms. *ACM Transactions on Computational Logic*, Vol. 1, No. 1, July 2000, pp. 77-111
- [2] W.Reisig: On Gurevich's Theorem on Sequential Algorithms *Actas Informatica 23* (2003), to appear