18. Markierte Graphen und Prozesse

Systeme der Informatik berechnen selten eine einzelne Ausgabe auf der Basis einer konkreten Eingabe. Stattdessen bestehen solche Systeme aus einer Vielzahl von Komponenten, oft *Prozesse* genannt. Diese Komponenten sind größtenteils unabhängig voneinander, teilweise interagieren sie jedoch miteinander, um gemeinsam etwas zu erreichen. Man spricht hier von nebenläufigen (engl. *concurrent*) Prozessen, welche neben lokalen Aktivitäten auch Informationen untereinander austauschen. Beispiele sind allgegenwärtig: eine Datenbank von Google, das GPS Satellitennavigationssystem, ein gewöhnliches Betriebssystem, eine CPU im Zusammenspiel mit einem 3D-Graphikprozessor, ein Mobiltelefon im GSM Netz, ein Pulsmesser und die entsprechende Uhr mit Empfänger am Handgelenk. Man spricht zusammenfassend auch von reaktiven Systemen.

Die lokalen Aktivitäten der reaktiven Systeme sind zumeist prozedurale Berechnungen im bekannten Sinne, und können beispielsweise als Prozeduren in SML adäquat repräsentiert werden. Die Beschreibung und Analyse der Nebenläufigkeit und Interaktion geschieht in der Regel mit *Kantenmarkierten Graphen*. Diese stehen im Mittelpunkt dieses Kapitels.

18.1 Graphen mit Kantenmarkierungen

Wir betrachten zunächst eine beliebige Menge M von Kantenmarkierungen, und werden später sehen, wie wir dieser Menge mehr Struktur verleihen können. Mit dieser Menge M definieren wir einen markierten Graph wie folgt: Ein markierter Graph ist ein Tripel $G = (V, M, E)$, so dass $E \subseteq V \times M \times V$. Beachten Sie, dass E keine binäre, sondern eine ternäre, also dreistellige Relation ist. Anstelle der Notation $\langle v, m, w\rangle \in E$ (aus Kapitel 8), schreiben wir auch $(v, m, w) \in E$.

Abb. 18.1 enthält einige Beispiele für kantenmarkierte Graphen mit Knotenmenge $V = \{0, 1, 2, 3\}$ und Kantenmarkierungen aus $M = \{a, b, c\}$.

Aufgabe 18.1. Was ist der Unterschied zwischen einer ternären Relation $E \subseteq V \times M \times V$, einer Funktion $E \in V \times M \to V$, und einer Funktion $E \in V \times M \rightharpoonup V$? Geben Sie Beispiele, die den Unterschied deutlich machen.

Aufgabe 18.2. Zeichnen Sie die markierten Graphen (V, M, E) zu

– $V = \{A\}$, $M = \{a, b\}$, $E = \{(A, a, A), (A, b, A)\}$;

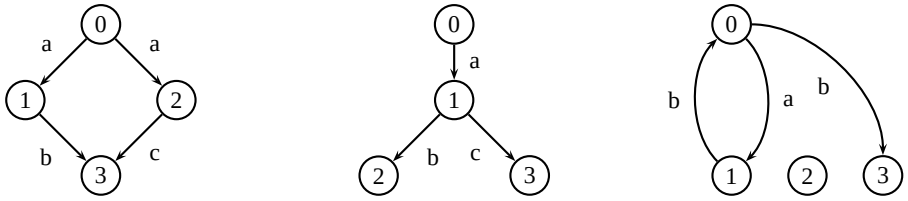


Abbildung 18.1: Einige markierte Graphen mit Knotenmenge $V = \{0, 1, 2, 3\}$ und Kantenmarkierungen aus $M = \{a, b, c\}$. Links sehen wir einen Graph mit $E = \{(0, a, 1), (0, a, 2), (1, b, 3), (2, c, 3)\}$. Daneben $E = \{(0, a, 1), (1, c, 3), (1, b, 2)\}$, und schliesslich $E = \{(0, a, 1), (1, b, 0), (0, b, 3), (1, b, 0)\}$.

– $V = \mathbb{N}$, $M = \mathbb{N}$, $E = \{(n, (n \mod 3), n + 3) \mid n \in \mathbb{N}\}$;

– $V = \{A, B, C\}$, $M = \{a, b, c\}$, $E = \{(A, a, B), (B, b, C), (B, c, C), (B, b, C)\}$.

Wir erweitern einige Definitionen für Graphen und Relationen aus Kapitel 8 auf markierte Graphen wie folgt:

Sei $G = (V, M, E)$ ein markierter Graph, und sei $G' = (V, E')$ der Graph, der durch Weglassen der Markierungen aus E entsteht, d.h. $E' = \{(v, w) \in V^2 \mid \exists m \in M : (v, m, w) \in E\}$.

– Ein Knoten w heißt *Nachfolger* eines Knotens v, wenn $(v, w) \in E'$. Er heißt m-*Nachfolger* von w, wenn $(v, m, w) \in E$

– Ein Knoten v heißt *Vorgänger* eines Knotens w, wenn $(v, w) \in E'$. Er heißt m-*Vorgänger* von w, wenn $(v, m, w) \in E$.

– Zwei Knoten heißen *benachbart* oder adjazent, wenn sie in G' benachbart sind.

– Ein endlicher *Pfad* ist ein nichtleeres Tupel $\langle v_1, m_1, \ldots, v_{n-1}, m_{n-1}, v_n\rangle \in (V \times (M \times V)^*)$, sodass für alle $i \in \{1, \ldots n - 1\}$, $(v_i, m_i, v_{i+1}) \in E$.

– Die Definitionen von Ausgangspunkt, Endpunkt, Länge eines Pfades, einfache Pfade und Zyklen übertragen sich vom unmarkierten Graphen G'.

– Ist $\langle v_1, m_1, \ldots, v_{n-1}, m_{n-1}, v_n\rangle$, ein Pfad, so ist $\langle m_1, \ldots, m_{n-1}\rangle$ die *Spur* vom Ausgangspunkt v_1 zum Endpunkt v_n in G.

– Ein Knoten w ist von einem Knoten v mit Spur $\mathcal{M} \in M^*$ in G *erreichbar*, wenn es einen Pfad mit Spur $\mathcal{M}$ vom Ausgangspunkt v zum Endpunkt w in G gibt.

– Ein Knoten v *hat die Spur* $\mathcal{M}$ in G, wenn von ihm ein Knoten w mit Spur $\mathcal{M}$ in G erreichbar ist.

– Die Definitionen von Wurzel, Quelle, initial, Senke, terminal und isoliert übertragen sich vom unmarkierten Graphen G'.

– Eine Spur heißt *vollständig*, wenn ihr Endpunkt terminal ist.

– Ein Graph ohne Zyklen heißt *azyklischer Graph*.
– Ein markierter Graph (V, M, E) heißt *Teilgraph* eines Graphen (V', M', E'), wenn $V \subseteq V', M \subseteq M', E \subseteq E'$ gilt.
– Für einen markierten Graphen (V, M, E) und Knoten $v \in V$ bezeichnet $Reach(G, v)$ den maximalen Teilgraph, der alle von v erreichbaren Knoten mit den zugehörigen Kanten umfasst.
– Die Umkehrrelation zu E ist $E^{-1} = \{(w, m, v) \mid (v, m, w) \in E\}$.

Gerne benutzen wir Pfeilsymbole für die Relation, und schreiben dann $v \xrightarrow{m} w$, falls $(v, m, w) \in \rightarrow$.

18.1.1 Prozesse

Ein *Prozess* ist ein Paar (G, v), wobei $G = (V, M, E)$ ein markierter Graph ist, und $v \in V$ einen Knoten als Anfangsknoten (oder *initialen* Knoten) auszeichnet. Für einen Prozess $P = (G, v)$ bezeichnen wir mit $Traces(P)$ die Menge aller Spuren, die v in G hat. Außerdem bezeichnen wir mit $Reach(P)$ den Prozess $(Reach(G, v), v)$. Beispielsweise ist $Traces(P) = \{\langle a\rangle, \langle a, b\rangle, \langle a, c\rangle\}$ für zwei der drei Prozesse in Abb. 18.1, wobei jeweils der Knoten 0 der Anfangsknoten sei.

Aufgabe 18.3. Sei $(G', v) = Reach(P)$. Zeigen Sie, dass G' zusammenhängend ist.

18.2 Äquivalenz von markierten Graphen und Prozessen

Wir wenden uns nun der Frage zu, wann zwei Graphen (oder Prozesse) äquivalent sind. Diese Frage ist für nebenläufige Systeme von ähnlicher Bedeutung wie die bereits mehrfach diskutierte Frage, wann zwei Programme äquivalent sind (Kapitel 2.9 und Kapitel 9.8). Allerdings ist die Antwort nicht ganz offensichtlich. Wir werden zunächst zwei verschiedene Definitionen kennen lernen, die wir später noch verfeinern werden. Die erste Definition basiert auf der Idee, dass zwei Prozesse dann gleich sind, wenn diese dieselben Spuren haben.

Definition 18.1. *Zwei Prozesse $P = (G, v)$ und $Q = (G', v')$ ($P \sim_{sp} Q$) sind spuräquivalent* $\overset{def}{\Longleftrightarrow} Traces(P) = Traces(Q)$.

Anschaulich gesprochen sind P und Q spuräquivalent, wenn jede Spur von P auch Spur von Q ist, und umgekehrt.

Aufgabe 18.4. Bestimmen Sie die Spuren der Prozesse in Abb. 18.1, wobei jeweils der Knoten 0 der Anfangsknoten sei. Welche Prozesse sind spuräquivalent?

Eine alternative Definition verwendet den Begriff der Isomorphie aus Kapitel 8.12.

Definition 18.2. *Zwei Prozesse $((V, M, E), v)$ und $((V', M', E'), v')$ sind isomorph, wenn es eine Bijektion $\phi \in V \to V'$ gibt, sodass $(v_1, m, v_2) \in E$ genau dann, wenn $(\phi(v_1), m, \phi(v_2)) \in E'$, und $\phi(v) = v'$.*

Zwei Prozesse P, Q sind isomorph ($P \cong Q$), wenn ihre Graphen isomorphe Struktur besitzen, d.h. dass die Knoten bijektiv aufeinander abbildbar sind, und die Bijektion die initialen Zustände aufeinander abbildet. Beachten Sie, dass Isomorphie die Knoten von V auf diejenigen von V' abbildet, während die Kantenmarkierungen nicht abgebildet werden. Isomorphie erlaubt uns also, von den Identitäten der Knoten zu abstrahieren.

Wir betrachten dazu folgendes Beispiel: links der Prozess $P = (G, 3)$ mit Knotenmenge $\{0, 1, 2, 3\}$ und rechts der Prozess $Q = (G', A)$ mit Knotenmenge $\{A, B, C, D\}$. Der Anfangsknoten ist dabei jeweils mit einem Pfeil gekennzeichnet.

Beide Prozesse sind isomorph, da die Bijektion $\{(3, A), (2, B), (1, C), (0, D)\}$ die Bedingungen aus Definition 18.2 erfüllt.

Aufgabe 18.5. Welche der Prozesse aus Abb. 18.1 sind isomorph zu P, welche zu Q?

Proposition 18.1. *Zwei isomorphe Prozesse sind spuräquivalent.*

Beweis. Sei $P = ((V, M, E), v)$ und $Q = ((V', M', E'), v')$, und sei $\phi \in V \to V'$ eine Bijektion, sodass $(v_1, m, v_2) \in E$ genau dann, wenn $(\phi(v_1), m, \phi(v_2)) \in E'$, und $\phi(v) = v'$.

Wir zeigen nur $Traces(P) \subseteq Traces(Q)$, die andere Richtung geht analog mit ϕ^{-1}, der Umkehrfunktion von ϕ, die sicher existiert (warum?). Sei $\langle m_1, \ldots, m_{n-1}\rangle \in Traces(P)$ beliebig, also eine beliebige Spur von v. Wir müssen zeigen, dass diese auch eine Spur von v' ist.

Aufgrund der Definition von "*Spur*" gibt es Knoten $v_2, \ldots v_n \in V'$, sodass $\langle v, m_1, v_2, \ldots, v_{n-1}, m_{n-1}, v_n\rangle$, ein Pfad in E ist. Wenn wir zeigen können, dass dann $\langle \phi(v), m_1, \phi(v_2), \ldots, \phi(v_{n-1}), m_{n-1}, \phi(v_n)\rangle$, ein Pfad in E'ist, dann ist damit $\langle m_1, \ldots, m_{n-1}\rangle$ eine Spur von $\phi(v') = v$, und wir sind fertig.

Es bleibt also zu zeigen, dass wenn $\langle v_1, m_1, v_2, \ldots, v_{n-1}, m_{n-1}, v_n\rangle$ ein Pfad von v_1 ist (in E), dass auch $\langle \phi(v_1), m_1, \phi(v_2), \ldots, \phi(v_{n-1}), m_{n-1}, \phi(v_n)\rangle$ ein Pfad von $\phi(v_1)$ ist (in E'). Wir zeigen dies per Induktion über n.

$n = 1$: Mit $(v_1, m_1, v_2) \in E$ ist (nach Voraussetzung) $(\phi(v_1), m_1, \phi(v_2)) \in E'$, und damit $\langle \phi(v_1), m_1, \phi(v_2)\rangle$ ein Pfad.

$n \geq 1$: Mit $(v_{n-1}, m, v_n) \in E$ ist (nach Voraussetzung) $(\phi(v_{n-1}), m, \phi(v_n)) \in E'$. Da der Pfad $\langle v_1, m_1, v_2, \ldots, v_{n-2}, m_{n-2}, v_{n-1}\rangle$ kürzer ist, dürfen wir annehmen (Induktionsannahme), dass $\langle \phi(v), m_1, \phi(v_2), \ldots, \phi(v_{n-2}), m_{n-2}, \phi(v_{n-1})\rangle$ ein Pfad von $\phi(v_1)$ ist (in E'). Aus beiden folgt die Behauptung: $\langle \phi(v), m_1, \phi(v_2), \ldots, \phi(v_{n-1}), m_{n-1}, \phi(v_n)\rangle$ ist ein Pfad von $\phi(v)$ (in E').

Aufgabe 18.6. Gilt die Umkehrung der obigen Proposition?

Aufgabe 18.7. Zeigen Sie, dass Isomorphie auf Prozessen eine Äquivalenzrelation ist. Zeigen Sie dasselbe für Spuräquivalenz. (Erinnerung: Eine Äquivalenzrelation ist eine reflexive, transitive und symmetrische, binäre Relation. Eine Relation R ist symmetrisch, wenn $R^{-1} \subseteq R$ gilt.)

18.3 Eine Sprache für Prozesse

Uns interessiert nun die Frage, wie wir Prozesse erzeugen können. Wir haben zuvor (Kapitel 7) gesehen, wie sich reine und markierte Bäume in einer Sprache (als SML Datenstrukturen) darstellen lassen. Nun machen wir in gewissen Sinne das umgekehrte. Wir geben eine einfache Sprache für Graphen, bzw. Prozesse. Die konkreten Knotenbezeichner sind dabei für uns unwesentlich. Wir versuchen, stattdessen, Prozesse bis auf Isomorphie erzeugen zu können, d.h. es ist nicht von Bedeutung, wie die Knoten im Graph des Prozesses heißen. Der erste Schritt ist eine Sprache für (kantenmarkierte) azyklische Graphen.

18.3.1 Abstrakte Syntax

Die abstrakte Syntax einer Sprache beschreibt die Bauform der Sprache. Üblicherweise definiert man die abstrakte Syntax einer Sprache mithilfe einer schematischen Darstellung, die als abstrakte Grammatik bezeichnet wird.

Abb. 18.2 zeigt eine abstrakte Grammatik für unsere erste Sprache L_0, wobei wir eine beliebige Menge M von Markierungen annehmen. Wir finden darin drei Operatoren: einen nullstelligen Operator '0', einen zweistelligen Operator '$.$', sowie einen zweistelligen Operator '$+$'. Lassen Sie sich nicht von den verwendeten Symbolen '0' und '$+$' verwirren, bevor wir uns die Bedeutung dieser Sprache bewusst gemacht haben. Wir lesen die Grammatik in dieser Abbildung wie folgt:

– 0 ist in der Sprache L_0.

– Wenn P in der Sprache L_0 ist, und $a \in M$, dann ist $a.P$ in der Sprache L_0.

– Wenn P_1 in der Sprache L_0 ist, und P_2 ebenfalls, dann ist $P_1 + P_2$ in der Sprache L_0.

Wir nennen die Elemente von L_0 *Terme* der Sprache L_0. Die einzelnen Konstrukte sollen dabei die folgende intuitive Bedeutung haben:

$$
\begin{array}{lcl}
a \in M & & \\
P \in L_0 & = & 0 \quad | \quad a.P \quad | \quad P + P
\end{array}
$$

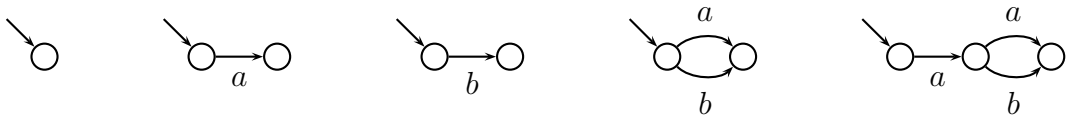
Abbildung 18.2: Abstrakte Grammatik von L_0.

– 0 ist der Term, der keinerlei Nachfolger besitzt, entspricht also einem terminalen Knoten.

– Der Term $a.P$ besitzt eine Kante, die zum Term P führt und mit a markiert ist.

– $P_1 + P_2$ stellt einen Term dar, der alle die ausgehenden Kanten besitzt, die die Terme P_1 oder P_2 besitzen.

Aufgrund dieser Intuition heißt der Operator '.' auch Präfix-Operator, der Operator '$+$' heißt Auswahl-Operator, und '0' wird gemeinhin schlicht *Stop*(-Operator) genannt. Hier sind einige sehr einfache Beispiele, die die intuitive Bedeutung der Terme in Form von Prozessen erhellen. Aufgrund der Tatsache, dass Knotenbeschriftungen für uns nicht relevant sein sollen, sind diese in diesen Beispielen nicht aufgeführt. Allein die Struktur der Graphen und die Kantenbeschriftungen sind für uns von Bedeutung.



0 $\quad a.0$ $\quad b.0$ $\quad (a.0) + (b.0)$ $\quad a.((a.0) + (b.0))$

Wir postulieren die folgenden Klammersparregeln:

$P + Q + R$	$\rightsquigarrow$	$(P + Q) + R$	+ klammert links
$a.b.P$	$\rightsquigarrow$	$a.(b.P)$	. klammert rechts
$a.P + Q$	$\rightsquigarrow$	$(a.P) + Q$	Punkt vor Strich

Aufgabe 18.8. Fügen Sie die fehlenden Klammern in die folgenden Terme ein: $a.P$, $a.(P{+}a.Q)$, $a.P + a.Q$, $a.a.a.a.a.0$, $a.P + (b.Q)$, und $a.P + a.b.Q$.

18.3.2 Semantik von L_0

Die Semantik einer Sprache legt die Regeln für die Ausführung von Elementen der Sprache, also den Termen von L_0, fest. Typischerweise unterscheidet man zwischen der *statischen* und der *dynamischen* Semantik[1], wobei die *statische* Semantik die Bedingungen formuliert, die semantisch zulässige Terme erfüllen müssen. Das heißt, im Allgemeinen gibt es Terme, die syntaktisch

[1] Siehe dazu auch Kapitel 2.8 und Kapitel 12

zulässig sind, die aber semantisch keinen Sinn ergeben. Wie in Kapitel 2.6 und 2.8 bereits diskutiert, geschieht dies in SML zum Beispiel, indem geprüft wird, ob eine Phrase *wohlgetypt* ist. Dies ist aufgrund der Einfachheit unserer Sprache (keine Datentypen) nicht sehr erhellend, wir überspringen diesen Schritt daher zunächst.

Die *dynamische* Semantik beschreibt, welche Ergebnisse die Ausführung zulässiger Terme liefern soll. Etwas formaler kann man sagen, dass wir festlegen müssen, welches mathematische Objekt mit einem Term assoziiert werden soll. In unserem Kontext soll das Ergebnis der Ausführung (die Semantik) eines Termes P ein Prozess (G, v) sein (wobei G azyklisch ist). Dafür haben wir festzulegen, was $G = (V, M, \rightarrow)$ ist und was $v \in V$ ist. Um diese Frage zu beantworten, bedienen wir uns zunächst eines – eleganten – technischen Kniffes. Wir identifizieren V mit L_0: die Knoten eines Prozesses sind alle erlaubten Terme der Sprache! Das heißt, die Kanten des Prozesses verbinden Terme der Sprache, und daher ist $\rightarrow$ eine Teilmenge von $(L_0 \times M \times L_0)$.

Damit ist intuitiv klar, dass der Anfangsknoten der Semantik von P, der Knoten P selbst ist. Nun bleibt es, festzulegen, was die Relation $\rightarrow$ sein könnte. Wir werden die ternäre Relation $\rightarrow$ mithilfe so genannter Inferenzregeln definieren, die wir in Kapitel 2.6 bereits kennen gelernt haben.

Allgemein hat eine Inferenzregel die Form

$$\frac{A_1 \ldots A_n}{A} \qquad \text{wobei } n \geq 0$$

Die Aussagen $A_1 \ldots A_n$ über dem Strich werden als *Prämissen* bezeichnet und die Aussage A unter dem Strich als *Konklusion*. Wir sagen, dass die Aussage A mit der Regel aus den Aussagen $A_1 \ldots A_n$ abgeleitet werden kann. Hat eine Regel keine Vorbedingung, so ist sie ohne jede Vorbedingung gültig, man spricht dann trefflich von einem *Axiom* (vgl. dazu die Nomenklatur in Kapitel 8.4).

Abb. 18.3 zeigt die definierenden Inferenzregeln für die Menge $\rightarrow$, wobei $\rightarrow$ genau die Tripel enthalten soll, die mit den Regeln abgeleitet werden können. Für jede Regel ist links ein Name angegeben. Die Regel prefix ist ein Axiom. Als konkretes Beispiel betrachten wir die Inferenzregel choice_l:

$$\frac{P \xrightarrow{a} P'}{P + Q \xrightarrow{a} P'}$$

Die Regel besagt, dass die Kante $P + Q \xrightarrow{a} P'$ (also das Tripel $\langle P + Q, a, P' \rangle$) dann in $\rightarrow$ enthalten ist, wenn die Kante $P \xrightarrow{a} P'$ darin enthalten ist. Insgesamt besagen die Regeln, dass fur jedes $P, Q \in L_0$ und jedes $a \in M$ gilt,

– $(a.P, a, P) \in \rightarrow$;
– $(P + Q, a, P') \in \rightarrow$, wenn $(P, a, P') \in \rightarrow$;
– $(P + Q, a, Q') \in \rightarrow$, wenn $(Q, a, Q') \in \rightarrow$;

$$\text{prefix}\ \frac{}{a.P \xrightarrow{a} P} \qquad \text{choice\_l}\ \frac{P \xrightarrow{a} P'}{P + Q \xrightarrow{a} P'} \qquad \text{choice\_r}\ \frac{Q \xrightarrow{a} Q'}{P + Q \xrightarrow{a} Q'}$$

Abbildung 18.3: Semantik von L_0

– nichts sonst ist Element von $\rightarrow$.

Beachten Sie, dass wir festgelegt haben, dass in $\rightarrow$ nur die Tripel liegen, die durch die Inferenzregeln abgeleitet werden können, und sonst keine. Anders gesagt ist $\rightarrow$ die kleinste Relation (bezüglich der Inklusionsordnung IO), die den obigen Regeln genügt. Jetzt wollen wir den Begriff der Semantik eines Terms $P \in L_0$ formal definieren. Sei

$$\mathcal{G}_{L_0} = \{(L_0, M, E) \mid E \subseteq L_0 \times M \times L_0\}$$

die Menge aller Graphen über M mit Knotenmenge L_0. Wir nennen die Funktion

$$\begin{aligned} [\![\_]\!] &\in L_0 \rightarrow \mathcal{G}_{L_0} \times L_0 \\ \text{mit } [\![P]\!] &= ((L_0, M, \rightarrow), P) \end{aligned}$$

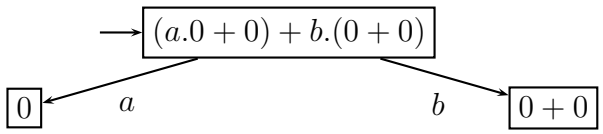
die Semantik der Terme von L_0.

Bei der grafischen Veranschaulichung der Semantik eines Terms P beschränken wir uns auf die Darstellung von $Reach([\![P]\!])$, denn dies ist das Fragment der Semantik, welches uns interessiert.

Die obigen Inferenzregeln können dazu verwendet werden, um in kompakter Form die Semantik eines Termes P abzuleiten. Man erzeugt dafür für jede Kante in $Reach([\![P]\!])$ einen sogenannten *Beweisbaum*, welcher die Existenz dieser Kante in $Reach([\![P]\!])$ beweist. Wir betrachten als Beispiel den Term $(a.0 + 0) + b.(0 + 0)$. Wir erhalten die folgenden Beweisbäume, welche für jede Kante die Prämissen "aufstapeln", bis ein Axiom erreicht ist.

$$\dfrac{\dfrac{\dfrac{}{a.0 \xrightarrow{a} 0}\ \text{mit prefix}}{a.0 + 0 \xrightarrow{a} 0}\ \text{mit choice\_l}}{(a.0 + 0) + b.(0 + 0) \xrightarrow{a} 0}\ \text{mit choice\_l} \qquad \dfrac{\dfrac{}{b.(0 + 0) \xrightarrow{b} 0 + 0}\ \text{mit prefix}}{(a.0 + 0) + b.(0 + 0) \xrightarrow{b} 0 + 0}\ \text{mit choice\_r}$$

Da jede der Regeln von L_0 höchstens eine Prämisse besitzt, verzweigen sich die Beweisbäume nicht. Wir werden später auch Verzweigungen der Beweisbäume kennenlernen. Aufgrund der obigen Inferenzen besitzt der Prozeß $[\![(a.0 + 0) + b.(0 + 0)]\!]$ die folgende grafische Darstellung:



Aufgabe 18.9. Leiten Sie mit den Inferenzregeln die Kanten für folgende Prozesse her: 0, $0+0$, $a.0$, $a.b.0+a.c.0$, $a.(b.0+c.0)$, $a.b.(0+0)+a.b.0$. Zeichnen Sie die jeweiligen Prozesse.

Wir beantworten nun die Frage, inwiefern wir jeden endlichen azyklischen Prozess über M durch einen Term von L_0 – bis auf Isomorphie – darstellen können.

Proposition 18.2. *Für jeden endlichen azyklischen Prozess (G,v) über M gibt es einen Term $P \in L_0$, so daß, $Reach((G,v))$ zu $Reach([\![\ P\ ]\!])$ isomorph ist.*

Wir verzichten auf den Beweis dieser Proposition, da der Beweis etwas umständlich ist. Die folgende Aufgabe dient dazu, in den nötigen Umstand ein wenig einzuführen.

Aufgabe 18.10. Geben Sie einen Term aus L_0 an, dessen Semantik isomorph zu dem Prozess $(G,0)$ ist, wobei $G = (\{0,1,2,3,4,5,6\}, M, E)$ mit $E = \{(0,a,1),(0,a,2),(0,a,3),(1,b,4),(2,b,5),(3,b,6)\}$.

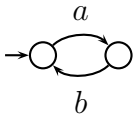
Aufgabe 18.11. Beweisen Sie, dass für beliebige $P,Q \in L_0$ der Prozess $Reach([\![\ P+Q\ ]\!])$ isomorph zu $Reach([\![\ Q+P\ ]\!])$ ist. Ist auch $Reach([\![\ a.(P+Q)\ ]\!])$ isomorph zu $Reach([\![\ a.(Q+P)\ ]\!])$? Ist $Reach([\![\ a.Q\ ]\!])$ isomorph zu $Reach([\![\ a.Q+a.Q\ ]\!])$? Wie steht es mit $Reach([\![\ a.(P+Q)+a.(P+Q)\ ]\!])$ und $Reach([\![\ a.(P+Q)+a.(Q+P)\ ]\!])$?

18.3.3 *L*: Syntax für beliebige Graphen

Um beliebige Graphen beschreiben zu können, fehlen uns noch Ausdrucksmittel zur Darstellung von Zyklen. Dazu bedienen wir uns *rekursiver Gleichungen*, wie wir sie bereits mehrfach gesehen haben. Konkret werden die Gleichungen ganz ähnlich zu den verschränkt rekursiven Definitionen von Funktionen, die wir in Kapitel 8.2 kennengelernt haben, verwendet.

Wir führen dazu eine Menge Var von *Rekursionsvariablen* ein, welche als linke Seiten in definierenden Gleichungen fungieren, und uns damit Mittel geben, Knoten, die auf Zyklen liegen, syntaktisch auszuzeichnen, und Kanten dorthin zurückweisen zu lassen. Hier ist dazu ein erstes Beispiel. Mit der Variablen $X \in Var$ können wir einen einfachen Zyklus wie folgt definieren:

$$X = a.b.X$$

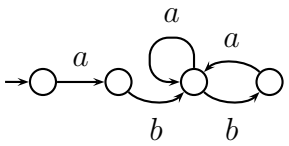


Der von X erzeugte Prozess ist rechts angedeutet. Im allgemeinen Fall verwenden wir mehrere Gleichungen, die oft verschränkt rekursiv definiert sind, wie etwa in folgendem Fall:

$$
\begin{array}{l}
a \in M \\
X \in Var \\
P \in L \;=\; 0 \;\mid\; a.P \;\mid\; P+P \;\mid\; X
\end{array}
$$

Abbildung 18.4: Abstrakte Grammatik von L.

$X = a.b.Y$
$Y = b.Z + a.Y$
$Z = a.Y$



Der von X erzeugte Prozess ist wiederum rechts angedeutet.

Wir nehmen an, dass eine Menge solcher Gleichungen disjunkt, aber nicht unbedingt erschöpfend ist (vgl. Kapitel 4.6). Auf der linken Seite der Gleichungen finden wir Rekursionsvariablen aus Var, auf der rechten Seite finden wir Terme. Diese Terme entsprechen im Wesentlichen den Termen von L_0, mit dem Unterschied, dass auch Rekursionsvariablen aus Var in den Termen vorkommen können. Die abstrakte Grammatik dieser erweiterten Sprache L ist in Abb. 18.4 angegeben. Man sieht, dass bis auf Rekursionsvariablen $X \in Var$ die Grammatik mit der Grammatik von L_0 aus Abb. 18.2 übereinstimmt.

Mit diesen Festlegungen definiert uns eine Menge solcher Gleichungen offensichtlich eine partielle Funktion $\Gamma \in Var \rightharpoonup L$. In Mengenschreibweise ergeben die obigen Beispiele die folgenden partiellen Funktionen:

$$\Gamma = \{(X, a.b.X)\} \quad \text{bzw.} \quad \Gamma = \{(X, a.b.Y), (Y, a.Z + b.Y), (Z, a.Y)\}$$

Wir nennen $\Gamma \in Var \rightharpoonup L$ eine *Menge von rekursiven Gleichungen über* L. Wenn $(X, P) \in \Gamma$, also $\Gamma(X) = P$ ist, können wir sagen, dass X den gleichen Prozeß wie P repräsentiert, nur dass jedesmal, wenn eine Variable Y erreicht wird, mit der rechten Seite $\Gamma(Y)$ der definierenden Gleichung von Y fortgesetzt wird.

Ein jedes Γ beschreibt, wie wir sofort sehen werden, einen Graph – mit oder ohne Zyklen. Den Zusammenhang zwischen Γ und Graph werden wir wiederum durch eine semantische Abbildung beschreiben. Für ein gegebenes Γ liefert uns die Semantik einen Prozeß, indem wir einen Term aus L als Anfangszustand festlegen. In den beiden obigen Beispielen haben wir jeweils X als Anfangszustand festgelegt.

18.3.4 Semantik von *L*

Wie zuvor wird die Semantik von L eine Abbildung jedes Termes von L auf einen Prozess sein. Diese Semantik ist natürlich von der Festlegung einer bestimmten Menge rekursiver Gleichungen Γ abhängig.

$$\textsf{prefix}\ \frac{}{a.P \xrightarrow{a} P} \qquad \textsf{choice\_l}\ \frac{P \xrightarrow{a} P'}{P + Q \xrightarrow{a} P'} \qquad \textsf{choice\_r}\ \frac{Q \xrightarrow{a} Q'}{P + Q \xrightarrow{a} Q'} \qquad \textsf{rec}\ \frac{\Gamma(X) = P \quad P \xrightarrow{a} P'}{X \xrightarrow{a} P'}$$

Abbildung 18.5: Semantik von L für $\Gamma \in Var \rightharpoonup L$

Für eine gegebene Menge rekursiver Gleichungen Γ über L ergibt sich die Semantik von L durch das Anfügen einer einzigen Inferenzregel an die Regeln aus Abb. 18.3:

$$\textsf{rec}\ \frac{\Gamma(X) = P \qquad P \xrightarrow{a} P'}{X \xrightarrow{a} P'}$$

Diese Regel besagt, dass die linke Seite X einer Gleichung von Γ einen a-Nachfolger P' hat, wenn auch die rechte Seite P einen solchen hat. Die sonstigen Regeln bleiben unverändert; alle Inferenzregeln sind noch einmal in Abb. 18.5 zusammengefasst. Nun wollen wir wiederum den Begriff der Semantik eines Terms $P \in L$ formal definieren. Zunächst definieren wir

$$\mathcal{G}_L = \{(L, M, E) \mid E \subseteq L \times M \times L\}$$

als die Menge aller Graphen über M, bei denen die Knotenmenge die Menge L ist. Für eine partielle Funktion $\Gamma : Var \rightharpoonup L$ sei $\rightarrow_\Gamma \subseteq L \times M \times L$ die kleinste ternäre Relation, die den Inferenzregeln aus Abb. 18.5 genügt. Dies ist also die Menge, die für jedes $P, Q \in L$ und jedes $a \in M$, die folgenden Elemente enthält:

– $(a.P, a, P) \in \rightarrow_\Gamma$;
– $(P + Q, a, P') \in \rightarrow_\Gamma$, wenn $(P, a, P') \in \rightarrow_\Gamma$;
– $(P + Q, a, Q') \in \rightarrow_\Gamma$, wenn $(Q, a, Q') \in \rightarrow_\Gamma$;
– $(X, a, P') \in \rightarrow_\Gamma$, wenn $\Gamma(X) = P$ und $P \xrightarrow{a} P'$;
– nichts sonst ist Element von $\rightarrow_\Gamma$.

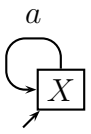
Die Semantik der Terme von L bezüglich Γ beschreiben wir durch eine kaskadierte Funktion, welche als erstes Argument die rekursiven Gleichungen gemäß Γ erhält, und als zweites Argument den Term $P \in L$, welcher als Anfangsknoten fungieren soll.

$$\begin{aligned} &[\![\ \_\ ]\!] \in\ (Var \rightharpoonup L) \to L \to \mathcal{G}_L \times L \\ &[\![\ \_\ ]\!]\ \Gamma\ P = ((L, M, \rightarrow_\Gamma), P) \end{aligned}$$

Für $[\![\ \_\ ]\!]\,\Gamma$ schreiben wir kurz auch $[\![\ \_\ ]\!]_\Gamma$.

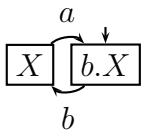
Wir wollen uns die Funktionsweise dieser Regeln anhand der Gleichung $X = a.X$, also $\Gamma = \{(X, a.X)\}$ veranschaulichen. Eine Kante von X existiert gemäß der Definition genau dann, wenn sie mit den Inferenzregeln abgeleitet werden kann. Davon gibt es nur eine:

$$\dfrac{\dfrac{\Gamma(X) = a.X \qquad \overline{a.X \xrightarrow{a} X}\ \text{mit prefix}}{X \xrightarrow{a} X}\ \text{mit rec}}$$



Der für $[\![\ X\ ]\!]_\Gamma$ resultierende Prozess ist rechts angedeutet, wobei wir uns wieder auf den erreichbaren Teilgraphen $Reach([\![\ X\ ]\!]_\Gamma)$ beschränken. Als zweites Beispiel betrachten wir $\Gamma = \{(X, a.b.X)\}$, welches uns schon bekannt ist, und möchten $[\![\ b.X\ ]\!]_\Gamma$ bestimmen. Wir erhalten den rechts angedeuteten Prozess aufgrund der folgenden Beweise:

$$\dfrac{\overline{b.X \xrightarrow{b} X}\ \text{mit prefix}}{} \qquad \dfrac{\dfrac{\Gamma(X) = a.b.X \qquad \overline{a.b.X \xrightarrow{a} b.X}\ \text{mit prefix}}{X \xrightarrow{a} b.X}\ \text{mit rec}}{}$$



Aufgabe 18.12. Überzeugen Sie sich davon, dass für das obige Beispiel keine weiteren Transitionen für $Reach([\![\ b.X\ ]\!]_\Gamma)$ beweisbar sind. Zeichnen Sie $Reach([\![\ a.(c.X + c.b.X)\ ]\!]_\Gamma)$.

Aufgabe 18.13. Erzeugen Sie durch Anwendung der semantischen Regeln die Prozesse $[\![\ X\ ]\!]_\Gamma$, wobei Γ jeweils gegeben ist durch:

1. $X = a.b.0 + a.c.0$
2. $X = a.b.X + a.c.0$
3. $X = a.a.a.a.X$
4. $X = a.Z$
5. $X = X$
6. $X = a.b.Y$
   $Y = b.Z + a.Y$
   $Z = a.Y$
7. $X = Y$
   $Y = a.Z$
8. $X = in.Y$
   $Y = out.X + in.out.Y$
9. $X = in.in.out.out.X$

Wir haben nun mit L eine Sprache, mit der wir beliebige endliche Prozesse bis auf Isomorphie syntaktisch notieren können. Dies ist auf den ersten Blick offensichtlich, allerdings ist der Beweis wiederum etwas umständlich. Wir verzichten daher zunächst darauf.

Aufgabe 18.14. Betrachten Sie die Prozesse $X = a.a.a.X$, $X = a.a.X$ und $X = a.(a.X + a.X)$ an. Welche dieser Prozesse sind isomorph, welche spuräquivalent?

Bemerkungen

Der in diesem Kapitel aufgezeigte Weg, die Semantik einer Sprache zu definieren, geht auf Hennessy und Plotkin (1979/81) zurück. Er wird gemeinhin als strukturelle, operationelle Semantik (SOS) bezeichnet. Das Wort *strukturell* bezieht sich dabei auf die Tatsache, dass die Semantik eines komplexen Terms aus der Struktur des Termes und der Semantik seiner Teilterme ergibt. *Operationell* bedeutet, dass die Semantik in sich ein ausführbares Modell mit Zuständen und Zustandsübergängen konstituiert.