Vorlesung

Programmschemata

Prof. Klaus Indermark

SS 2002

Inhaltsverzeichnis

Kapitel 1

Einleitung

Inhalt

- Berechnungsstärke von Kontrollstrukturen
- Schleifenstrukturen von Flussdiagrammen
- Iteration und Rekursion
- Beziehungen zwischen Kontroll- und Datenstrukturen
- Charakterisierung der Äquivalenz von Programmschemata durch formale Sprachen

Frage:

Haben Kontrollstrukturen unterschiedliche Berechnungsstärke?

Antwort:

- Berechnenbarkeitstheorie: nein
  denn WHILE ist bereits universell
  folgende Strukturen sind nicht berechnungsstärker:
  - Iteration (Flussdiagramm, GOTO)
  - Rekursion (rekursive Prozeduren)
  - Rekursion auf höherer Stufe (rekursive Proteduren mit Prozedur-Parametern, getypter λ-Kalkül)
  - Selbstanwendung (ungetypter λ-Kalkül)
- Compilerbau: ja

  denn: Rekursion = Iteration + Keller
  Rekursion auf höherer Stufe = Iteration + "Spaghetti-Stack"

Grund

Die klassische Berechenbarkeitstheorie bezieht sich auf $\langle \mathbb{N}, 0, suc \rangle \rightsquigarrow$ Kodierungsmöglichkeiten. Compiler hingegen ignorieren diese Kodierungsmöglichkeiten wegen ihrer Ineffizienz.
Hier gehen wir von einer Unabhängigkeit von Grundfunktionen, d.h. einer relativen Berechenbarkeit aus.
Programm = $\langle$Schema, Interpretation$\rangle$

Folie 0.4
Folie 0.5
Folie 0.6

Definition 1.1 *Seien S_1, S_2 Schemata und k_1, k_2 Schemaklassen.*

- $S_1 \sim S_2$ *(äquivalent)* $:\Leftrightarrow \forall$ *Interpreationen I gilt:* $[\![S_1, I]\!] = [\![S_2, I]\!]$
- $k_1 \rightsquigarrow k_2$ *(Klasse k_1 übersetzbar in Klasse k_2)* $:\Leftrightarrow \forall S \in k_1 \exists S' \in k_2 : S \sim S'$
- $k_1 \sim k_2$ *(äquivalent)* $:\Leftrightarrow k_1 \rightsquigarrow k_2 \wedge k_2 \rightsquigarrow k_1$

Resultate:

- WHILE $\rightsquigarrow$ ITER $\rightsquigarrow$ REC $\rightsquigarrow$ n-REC $\rightsquigarrow$ SELF
- Hierarchie dieser Schemaklassen
- ITER $\rightsquigarrow$ WHILE + "boolsche Variablen"
- REC $\rightsquigarrow$ ITER + "Keller"

Folie 0.7

Unabhängig von der Interpretation der Operationssymbole und des Prädikatensymbols P berechnen die entsprechenden Programme bei Eingabe auf X dieselben Werte auf Y und Z. Die Programme terminieren genau dann, wenn für die Eingabe X gilt:

- $p\big(g(X)\big) = true$,
- $p\big(f^m(X)\big) = false$ und
- $p\big(f^i(X)\big) = true$ $(i = 1, \ldots, m-1)$

Bei Termination ergibt sich auf Y der Wert $f^m(X)$ und auf Z der Wert $g^2(X)$. Somit gilt für die iterativen Schemata A und B: $A \sim B$.

Kapitel 2

IANOV-Schema

Es existieren verschiedene Möglichkeiten der Zerlegung eines Programms in Schema und Interpretation. Die Abstraktion von der Struktur des Zustandsraums liefert *einfache Schemata*, und zwar

- DIJKSTRA-Schemata (WHILE)
- KOSARAJU-Schemata (REPEAT–EXIT)
- IANOV-Schemata (GOTO)
- DE BAKKER-Schemata (REC)

Die Charakterisierung der Äquivalenz von Schemata erfolgt durch *formale Sprachen*.

Definition 2.1 (IANOV-Schema) *Seien Ω, Π und Q nicht-leere endliche Mengen und $q_1 \in Q$. Dann heißt eine Abbildung*

$$S : Q \longrightarrow (\Omega \times Q) \cup (\Pi \times Q \times Q) \cup \{\mathsf{stop}\}$$

eine IANOV-***Schema*** *über der Menge Ω der* ***Operationssymbole*** *und der Menge Π der* ***Prädikatssymbole****. Die Elemente $q \in Q$ heißen* ***(Sprung-)Marken****, q_1 heißt* ***Startmarke*** *und q mit $S(q) =$ stop* ***Stopmarke****. $Ian(\Omega, \Pi)$ bezeichnet die Klasse aller* IANOV*-Schemata über Ω und Π.*

Definition 2.2 (Interpretation) *Sei A eine Menge und α eine Abbildung mit*

- $\alpha(f) : A \longrightarrow A$ *für alle* $f \in \Omega$
- $\alpha(p) : A \longrightarrow \{T, F\}$ *für alle* $p \in \Pi$

Dann heißt $\mathfrak{A} = \langle A; \alpha \rangle$ eine ***Interpretation*** *von (Ω, Π). $Int(\Omega, \Pi)$ bezeichnet die Klasse aller Interpretationen von (Ω, Π).*

Schreibweise:

Wir schreiben statt $\alpha(f)$ auch $f_{\mathfrak{A}}$, bzw. $\underline{f}$, falls die Interpretation $\mathfrak{A}$ bereits festgelegt wurde.

Definition 2.3 (Programm, Semantik) *Wenn $S \in Ian(\Omega, \Pi)$ und $\mathfrak{A} \in Int(\Omega, \Pi)$, so heißt $P = \langle S, \mathfrak{A} \rangle$ ein* IANOV-***Programm****. Es berechnet eine partielle* ***Zustandstransformation***

$$[\![P]\!] : A \dashrightarrow A$$

Zunächst bestimmt P die ***Transition (Einzelschrittfunktion)***

$$\Delta_P : Q \times A \longrightarrow (Q \cup \{\mathsf{stop}\}) \times A$$

durch

$$\begin{aligned}
\Delta_P(q,a) &:= \quad (q', f_{\mathfrak{A}}(a)) \quad &, \textit{falls } S(q) = (f, q') \\
\Delta_P(q,a) &:= \quad (q_{p_{\mathfrak{A}}(a)}, a) \quad &, \textit{falls } S(q) = (p, q_T, q_F) \\
\Delta_P(q,a) &:= \quad (\mathsf{stop}, a) \quad &, \textit{falls } S(q) = \mathsf{stop}
\end{aligned}$$

Durch Iteration erhält man die Semantik von P:

$$[\![P]\!](a) = a' :\curvearrowright\curvearrowright \exists n \in \mathbb{N} : \Delta_P^n(q_1, a) = (\mathsf{stop}, a')$$

Beachte:

n ist eindeutig, falls es existiert; andernfalls liegt eine unendliche Berechnung vor: $[\![P]\!](a) = \perp$ (d. h. nicht definiert)

Schreibweisen:

$[\![S]\!]_{\mathfrak{A}}$	statt	$[\![S, \mathfrak{A}]\!]$
q: do f goto q'	statt	$S(q) = (f, q')$
q: if p then goto q' else goto q'' fi	statt	$S(q) = (p, q', q'')$
q: stop	statt	$S(q) = \mathsf{stop}$

siehe auch Folie 1.2

Beachte:

IANOV-Schemata entsprechen einfachen *iterativen* Schemata. Sie können auch als *Flussdiagramme*, *GOTO-Programme* oder *Maschinenprogramme* interpretiert werden. Der Nachteil solcher Schemata ist allerdings ihre Unübersichtlichkeit.

Definition 2.4 (Äquivalenz) *Für $S, S' \in Ian(\Omega, \Pi)$ definieren wir*

$$S \sim S' :\curvearrowright\curvearrowright \forall \mathfrak{A} \in Int(\Omega, \Pi) : [\![S]\!]_{\mathfrak{A}} = [\![S']\!]_{\mathfrak{A}}$$

Folie 1.4

Das Ziel dieses Kapitels ist unter anderem die Entscheidbarkeit der Äquivalenz zu untersuchen. Dazu charakterisieren wir die Äquivalenz durch *reguläre Sprachen.*

2.1 Freie Interpretationen

Definition 2.5 $\mathfrak{F} = \langle \Omega^*; \varphi \rangle \in Int(\Omega, \Pi)$ *heißt* ***frei*** $:\curvearrowright\curvearrowright$ *für jedes $f \in \Omega$ und $w \in \Omega^*$ gilt: $\varphi(f)(w) = wf$.*

Beachte:

Es werden also keine Bedingungen an die Prädikate gestellt. In diesem Sinne ist eine solche Interpretation *frei*.

Das Ziel dieses Abschnittes ist die Reduktion der Äquivalenz ($\sim$) auf *freie Interpretationen* ("Syntaktisierung").

$\mathfrak{A} \in Int(\Omega, \Pi)$ und $a \in A$ bestimmen eine Abbildung

$$h_{(\mathfrak{A},A)} : \Omega^* \longrightarrow A$$

und eine freie Interpretation $\mathfrak{F}_{(\mathfrak{A},A)} \in Int(\Omega, \Pi)$ durch

$$\begin{aligned} h_{(\mathfrak{A},a)}(\varepsilon) &:= a \\ h_{(\mathfrak{A},a)}(wf) &:= f_{\mathfrak{A}}\big(h_{(\mathfrak{A},a)}(w)\big) \\ \varphi(p)(w) &:= \alpha(p)\big(h_{(\mathfrak{A},a)}(w)\big) \end{aligned}$$

Satz 2.1 (Reduktion auf freie Interpretationen) *Für $S \in Ian(\Omega, \Pi)$, $\mathfrak{A} \in Int(\Omega, \Pi)$ und $a \in A$ gilt:*

$$[\![S]\!]_{\mathfrak{A}}(a) = a' \curvearrowright\!\!\curvearrowright [\![S]\!]_{\mathfrak{F}_{(\mathfrak{A},a)}}(\varepsilon) = w$$
$$\textit{und } h_{(\mathfrak{A},a)}(w) = a$$

Beweis:

Mit $h := h_{(\mathfrak{A},a)}$ und $\mathfrak{F} := \mathfrak{F}_{(\mathfrak{A},a)}$ gilt für die einzelnen Rechenschritte von $(S, \mathfrak{A})$ bei Eingabe a und $(S, \mathfrak{F})$ bei Eingabe von ε folgende Beziehung:

(1) Falls $S(q) = (f, q')$, so gilt:
$\Delta_{(S,\mathfrak{A})}\big(q, h(w)\big) = (q', a') \curvearrowright\!\!\curvearrowright \Delta_{(S,\mathfrak{F})}(q, w) = (q', wf)$ und $h(wf) = a'$

(2) Falls $S(q) = (p, q', q'')$, so gilt:
$\Delta_{(S,\mathfrak{A})}\big(q, h(w)\big) = \big(\widetilde{q}, h(w)\big) \curvearrowright\!\!\curvearrowright \Delta_{(S,\mathfrak{F})}(q, w) = (\widetilde{q}, w)$

Daraus folgt die Behauptung:

$$\begin{aligned} [\![S]\!]_{\mathfrak{A}}(a) = a' &\curvearrowright\!\!\curvearrowright \Delta^n_{(S,\mathfrak{A})}(q_1, a) = (\mathsf{stop}, a') \\ &\curvearrowright\!\!\curvearrowright \Delta^n_{(S,\mathfrak{F})}(q_1, \varepsilon) = (\mathsf{stop}, w) \text{ und } h(w) = a' \end{aligned}$$

q.e.d.

Korollar 2.1 *Für $S_1, S_2 \in Ian(\Omega, \Pi)$ gilt:*

$$S_1 \sim S_2 \curvearrowright\!\!\curvearrowright \textit{ für alle freien } \mathfrak{F} \in Int(\Omega, \Pi) : [\![S_1]\!]_{\mathfrak{F}}(\varepsilon) = [\![S_2]\!]_{\mathfrak{F}}(\varepsilon)$$

Beweis:

"$\curvearrowright$": gilt nach Definition der Äquivalenz ($\sim$)

"$\curvearrowleft$": Für $\mathfrak{A} \in Int(\Omega, \Pi)$ und $a \in A$ folgt:

$$\begin{aligned} & [\![S_1]\!]_{\mathfrak{A}}(a) = a' \\ \curvearrowright\!\!\curvearrowright\ & [\![S_1]\!]_{\mathfrak{F}_{(\mathfrak{A},a)}}(\varepsilon) = w \text{ und } a' = h_{(\mathfrak{A},a)}(w) \\ \curvearrowright\!\!\curvearrowright\ & [\![S_2]\!]_{\mathfrak{F}_{(\mathfrak{A},a)}}(\varepsilon) = w \text{ und } a' = h_{(\mathfrak{A},a)}(w) \\ \curvearrowright\!\!\curvearrowright\ & [\![S_2]\!]_{\mathfrak{A}}(a) = a' \end{aligned}$$

q.e.d.

Folie 1.5

2.2 Die Standardform eines IANOV-Schemas

Das Ziel dieses Abschnitts ist die Konstruktion einer *formalen Sprache* $L(S)$ mit $S_1 \sim S_2 \curvearrowright\!\!\!\curvearrowright L(S_1) = L(S_2)$

Sei

- $S : Q \longrightarrow (\Omega \times Q) \cup (\Pi \times Q^2) \cup \{\mathsf{stop}\} \in Ian(\Omega, \Pi)$
- $\Pi := \{p_1, \ldots, p_n\}$
- $B := \{T, F\}^n$ "*Bedingungen*"
- $\mathfrak{F} := \langle \Omega^*; \varphi \rangle \in Int(\Omega, \Pi)$ frei

φ legt für jedes $w \in \Omega^*$ Bedingungen fest:

$$\varphi_\Pi : \Omega^* \longrightarrow B$$

$$\varphi_\Pi(w) := \big(\varphi(p_1)(w), \ldots, \varphi(p_n)(w)\big)$$

Diese Bedingungen bestimmen die Berechnungen von $(S, \mathfrak{F})$. Für die mit (q, w) beginnenden Berechnungen gibt es drei Möglichkeiten:

(1) **Operationsschritt:**
$\Delta^m_{(S,\mathfrak{F})}(q, w) = (\widetilde{q}, w)$ und $\Delta_{(S,\mathfrak{F})}(\widetilde{q}, w) = (q', wf)$

(2) **Testschleife:**
$\Delta^m_{(S,\mathfrak{F})}(q, w) = (\widetilde{q}, w)$ und $\Delta^{m'}_{(S,\mathfrak{F})}(\widetilde{q}, w) = (\widetilde{q}, w)$

(3) **Stopschritt:**
$\Delta^m_{(S,\mathfrak{F})}(q, w) = (\widetilde{q}, w)$ und $\Delta_{(S,\mathfrak{F})}(\widetilde{q}, w) = (\mathsf{stop}, w)$

Beachte:

In jedem der drei Fälle existiert ein geeignetes $m < |Q|$. Das Vorliegen eines dieses Fälle ist bereits durch den boolschen Vektor $\varphi_\Pi(w) \in B$ bestimmt.

Mit den obigen drei Fällen können wir die **charakteristische Abbildung** von S definieren:

$$ch_S : Q \times B \longrightarrow (\Omega \times Q) \cup \{\mathsf{loop}, \mathsf{stop}\}$$

$$ch_S(q, \beta) := \begin{cases} (f, q') & : \text{ Operationsschritt (1)} \\ \mathsf{loop} & : \text{ Testschleife (2)} \\ \mathsf{stop} & : \text{ Stopschritt (3)} \end{cases} \quad \text{wobei } \varphi_\Pi(w) = \beta$$

Wir vereinfachen ch_S, indem wir Q auf die *durch Operationsschritte erreichbaren* Zustände einschränken. Die Menge Q_c der **charakteristischen Zustände** von S sei definiert durch:

- $q_1 \in Q_c$
- $q \in Q_c, \beta \in B, ch_S(q, \beta) = (f, q') \curvearrowright q' \in Q_c$

Damit erhalten wir die **vereinfachte charakteristische Abbildung** von S:

$$ch_S : Q_c \times B \longrightarrow (\Omega \times Q_c) \cup \{\mathsf{loop}, \mathsf{stop}\}$$

Beachte:

ch_S ist effektiv aus S konstruierbar.

Folie 1.6
Folie 1.7
Folie 1.8

Definition 2.6 (freie Standardform) *Die Standardform ist* ***frei*** $\curvearrowright$ *jeder syntaktische Pfad ist auch ein Berechnungspfad. Wir sprechen von einer syntaktischen Charakterisierung der Äquivalenz ($\sim$).*

2.3 Der endliche Automat eines IANOV-Schemas

$S \in Ian(\Omega, \Pi)$ bestimmt durch seine *charakteristische Abbildung*

$$ch_S : Q_c \times B \longrightarrow (\Omega \times Q_c) \cup \{\mathsf{loop}, \mathsf{stop}\}$$

einen *endlichen Automaten* $\mathfrak{A}_S := \langle \overline{Q}, \Sigma, \delta, q_1, F\rangle$ mit

- $\overline{Q} := Q_c \cup \{q^\beta \mid q \in Q_c, \beta \in B\}$
- $\Sigma := \Omega \cup B$
- $F := \{q^\beta \mid ch_S(q, \beta) = \mathsf{stop}\}$
- $\delta : \overline{Q} \times \Sigma - \to \overline{Q}$
  - $\delta(q, \beta) := q^\beta$
  - $\delta(q^\beta, f) = q' :\curvearrowright ch_S(q, \beta) = (f, q')$

Folie 1.9

2.4 Die Sprache eines IANOV-Schemas

$S \in Ian(\Omega, \Pi)$ bestimmt über seinen *endlichen Automaten* $\mathfrak{A}_S$ die *reguläre Sprache* $L_S := L(\mathfrak{A}_S) \subseteq \Sigma^*$. Es folgt: $L_S \subseteq (B\Omega)^* B$ und $\beta_1 f_1 \dots \beta_r f_r \beta_{r+1} \in L_S$ mit $r \in \mathbb{N} :\curvearrowright \exists q_1, \dots, q_r \in Q_c : ch_S(q_i, \beta_i) = (f_i, q_{i+1})$ für $i = 1, \dots, r$ und $ch_S(q_{r+1}, \beta_{r+1}) = \mathsf{stop}$.

Folie 1.10

Satz 2.2 *Für $S_1, S_2 \in Ian(\Omega, \Pi)$ gilt: $S_1 \sim S_2 \curvearrowright L_{S_1} = L_{S_2}$.*

Beweis:

Zunächst zeigen wir eine Beziehung zwischen L_S und $\{[\![S]\!]_{\mathfrak{F}}(\varepsilon) \mid \mathfrak{F} \in Int(\Omega, \Pi) \text{ frei}\}$:

$(*)$ Sei $f_1 \dots f_r \in \Omega^*$ und $\mathfrak{F} := \langle \Omega^*; \varphi\rangle \in Int(\Omega, \Pi)$ mit $\varphi_\Pi(f_1 \dots f_i) = \beta_{i+1}$ für $i = 0, 1, \dots r$
Dann gilt:
$\beta_1 f_1 \dots \beta_r f_r \beta_{r+1} \in L_S \curvearrowright [\![S]\!]_{\mathfrak{F}}(\varepsilon) = f_1 \dots f_r$

Beweis von $(*)$:

$\beta_1 f_1 \dots \beta_r f_r \beta_{r+1} \in L_S$
$\curvearrowright \exists q_1, \dots, q_r \in Q_c : ch_S(q_i, \beta_i) = (f_i, q_{i+1})$ und $ch_S(q_{r+1}, \beta_{r+1}) = \mathsf{stop}$
$\curvearrowright \exists q_1, \dots, q_{r+1} \in Q_c$ und $\exists \widetilde{q_1}, \dots, \widetilde{q_{r+1}} \in Q : (q_1, \varepsilon) \vdash^* (\widetilde{q_1}, \varepsilon) \vdash (q_2, f_1) \vdash^*$
$(\widetilde{q_2}, f_1) \vdash \dots \vdash (q_{r+1}, f_1 \dots f_r) \vdash (\mathsf{stop}, f_1, \dots f_r)$.

Da nun $S_1 \sim S_2 \curvearrowright\!\!\curvearrowright [\![S_1]\!]_{\mathfrak{F}}(\varepsilon) = [\![S_2]\!]_{\mathfrak{F}}(\varepsilon)\ \forall$ freie $\mathfrak{F} \in Int(\Omega, \Pi)$ ergibt sich aus $(*)$ die Behauptung:

- "$\curvearrowright$": $\beta_1 f_1 \dots \beta_r f_r \beta_{r+1} \in L_{S_1}$
  Wähle freies $\mathfrak{F} = \langle \Omega^*; \varphi \rangle$ mit $\varphi_\Pi(f_1 \dots f_i) = \beta_{i+1}$ $(i = 0, \dots, r)$. Nach $(*)$ folgt: $[\![S_1]\!]_{\mathfrak{F}}(\varepsilon) = f_1 \dots f_r$. Da $S_1 \sim S_2$: $[\![S_2]\!]_{\mathfrak{F}}(\varepsilon) = f_1 \dots f_r$ und nach $(*)$ folgt: $\beta_1 f_1 \dots \beta_r f_r \beta_{r+1} \in L_{S_2}$
- "$\curvearrowleft$": z. z.: $[\![S_1]\!]_{\mathfrak{F}}(\varepsilon) = [\![S_2]\!]_{\mathfrak{F}}(\varepsilon)\ \forall$ freie $\mathfrak{F} \in Int(\Omega, \Pi)$

  Sei $[\![S_1]\!]_{\mathfrak{F}}(\varepsilon) = f_1 \dots f_r$. Mit $\beta_{i+1} := \varphi_\Pi(f_1 f_2 \dots f_i)$ $(0 \le i \le r) \overset{(*)}{\curvearrowright}$ $\beta_1 f_1 \dots \beta_r f_r \beta_{r+1} \in L_{S_1}$ und nach Voraussetzung auch $\in L_{S_2} \overset{(*)}{\curvearrowright} [\![S_2]\!]_{\mathfrak{F}}(\varepsilon) = f_1 \dots f_r \curvearrowright S_1 \sim S_2$

q.e.d.

Korollar 2.2 *Die Äquivalenz von* IANOV*-Schemata ist entscheidbar.*

Beweis:

$S_1 \sim S_2 \curvearrowright\!\!\curvearrowright \mathfrak{A}_{S_1} \sim \mathfrak{A}_{S_2}$

Die Äquivalenz von endlichen Automaten ist entscheidbar.

q.e.d.

Folie 1.11

Folie 1.12

Kapitel 3

DIJKSTRA-Schemata

Während es sich bei IANOV-Schemata um *Flussdiagramme* handelt, die *beliebige* Schleifenstrukturen haben können, geht es in diesem Kapitel um die *strukturierte* Programmierung. Diese geht zurück auf DIJKSRTA, HOARE und WIRTH. Ausgelöst wurde die Diskussion um diesen Ansatz durch die Arbeit von DIJKSTRA *"Goto statements considered harmful"* (1968), in der gefordert wird statt Sprungbefehlen (goto) zu verwenden, einen induktiven, d. h. kompositionellen Programmaufbau zu verfolgen. Ein Programm soll also nur aus *Sequenzen*, *Verzweigungen* und *while-Schleifen* bestehen. Durch diese Struktur wird beispielsweise die Programmverifikation erleichtert.

3.1 Syntax von DIJKSTRA-Schemata

Seien Ω, Π nicht-leere endlichen Mengen von *Operations- und Funktionssymbolen.* Wir definieren $BExp(\Pi)$ als die **Menge der boolschen Ausdrücke** über Π induktiv wie folgt:

- $\Pi \subseteq BExp(\Pi)$
- $b_1, b_2 \in BExp(\Pi) \curvearrowright (b_1 \wedge b_2), (b_1 \vee b_2), (\neg b_1) \in BExp(\Pi)$

Dann ist die Menge $Dij(\Omega, \Pi)$ der **Dijkstra-Schemata** über Ω und Π definiert durch:

- $\Omega \subseteq Dij(\Omega, \Pi)$
- **Sequenz:**
  $S_1, S_2 \in Dij(\Omega, \Pi) \curvearrowright (S_1; S_2) \in Dij(\Omega, \Pi)$
- **Verzweigung:**
  $S_1, S_2 \in Dij(\Omega, \Pi), b \in BExp(\Pi) \curvearrowright$ if b then S_1 else S_2 fi $\in Dij(\Omega, \Pi)$
- **Schleife:**
  $S \in Dij(\Omega, \Pi), b \in BExp(\Pi) \curvearrowright$ while b do S end $\in Dij(\Omega, \Pi)$

3.2 Semantik von DIJKSTRA-Schemata

Zunächst definieren wir die Semantik der boolschen Ausdrücke über Π. $b \in BExp(\Pi)$ bestimmt eine Funktion

$$[\![b]\!]_{\mathfrak{A}} : A \longrightarrow \{0, 1\}$$

durch

- $[\![p]\!]_{\mathfrak{A}} := \alpha(p)$, für $p \in \Pi$
- $[\![(b_1 \wedge b_2)]\!]_{\mathfrak{A}}(a) := [\![b_1]\!]_{\mathfrak{A}}(a) \wedge [\![b_2]\!]_{\mathfrak{A}}(a)$
- $[\![(b_1 \vee b_2)]\!]_{\mathfrak{A}}(a) := [\![b_1]\!]_{\mathfrak{A}}(a) \vee [\![b_2]\!]_{\mathfrak{A}}(a)$
- $[\![\neg b]\!]_{\mathfrak{A}}(a) := \neg [\![b]\!]_{\mathfrak{A}}(a)$

Sei $\mathfrak{A} = \langle A; \alpha \rangle \in Int(\Omega, \Pi)$ und $S \in Dij(\Omega, \Pi)$. Dann heißt $(S, \mathfrak{A})$ ein **Dijkstra-Programm** über Ω und Π. Es bestimmt eine **Speichertransformation**

$$[\![S]\!]_{\mathfrak{A}} : A \dashrightarrow A$$

die durch Induktion über dir Struktur von S definiert ist:

- $[\![f]\!]_{\mathfrak{A}} := \alpha(f)$, für $f \in \Omega$
- $[\![(S_1; S_2)]\!]_{\mathfrak{A}}(a) := [\![S_2]\!]_{\mathfrak{A}}\big([\![S_1]\!]_{\mathfrak{A}}(a)\big)$
- $[\![\mathsf{if}\ b\ \mathsf{then}\ S_2\ \mathsf{else}\ S_2\ \mathsf{fi}]\!]_{\mathfrak{A}}(a) := \begin{cases} [\![S_1]\!]_{\mathfrak{A}}(a) & : \quad [\![b]\!]_{\mathfrak{A}}(a) = 1 \\ [\![S_2]\!]_{\mathfrak{A}}(a) & : \quad [\![b]\!]_{\mathfrak{A}}(a) = 0 \end{cases}$
- $[\![\mathsf{while}\ b\ \mathsf{do}\ S\ \mathsf{end}]\!]_{\mathfrak{A}}(a) := \begin{cases} [\![S]\!]^k_{\mathfrak{A}}(a) & : \quad \forall i < k : [\![b]\!]_{\mathfrak{A}}\big([\![S]\!]^i_{\mathfrak{A}}(a)\big) = 1 \\ & \quad \text{und } [\![b]\!]_{\mathfrak{A}}\big([\![S]\!]^k_{\mathfrak{A}}(a)\big) = 0 \\ \bot & : \quad \text{sonst} \end{cases}$

Beachte:

(1) k ist eindeutig definiert

(2) $[\![\mathsf{while}\ b\ \mathsf{do}\ S\ \mathsf{end}]\!]_{\mathfrak{A}}(a) = \bot \curvearrowright$
   (a) $[\![b]\!]_{\mathfrak{A}}\big([\![S]\!]^i_{\mathfrak{A}}(a)\big) = 1 \forall i \in \mathbb{N}$, oder
   (b) $\exists i \in \mathbb{N} : [\![S]\!]^i_{\mathfrak{A}}(a) = \bot$ und $\forall j < i : [\![S]\!]^j_{\mathfrak{A}}(a) = 1$

Satz 3.1 *Zu jedem $S \in Dij(\Omega, \Pi)$ lässt sich ein äquivalentes $S' \in Ian(\Omega, \Pi)$ konstruieren, also:*

$$Dij(\Omega, \Pi) \rightsquigarrow Ian(\Omega, \Pi)$$

Beweis:

durch Induktion über die Struktur von S. Definiere zunächst ein *Testschema* S_b für $b \in BExp(\Pi)$.

$$S_b : Q \longrightarrow (\Pi \times Q^2) \cup \{true, false\}$$

mit $q_1 \in Q$ heißt **Testschema**.

$$[\![S_b]\!]_{\mathfrak{A}} : A \longrightarrow \{0, 1\} \text{ mit}$$

$$[\![S_b]\!]_{\mathfrak{A}}(a) := \begin{cases} 1 & : \quad (q_1, a) \vdash^* (true, a) \\ 0 & : \quad (q_1, a) \vdash^* (false, a) \\ \bot & : \quad (q_1, a) \vdash^* (\widetilde{q}, a) \vdash^+ (\widetilde{q}, a) \end{cases}$$

Zu jedem boolschen Ausdruck $b \in BExp(\Pi)$ gibt es ein äquivalentes Testschema S_b: Folie 2.1

Folie 2.2

q.e.d.

Korollar 3.1 *Die Äquivalenz von* DIJKSTRA*-Schemata ist entscheidbar.*

3.3 Schemasprachen über Ω und Π

Ziel dieses Abschnittes ist es zu zeigen, dass $Ian(\Omega,\Pi) \not\rightsquigarrow Dij(\Omega,\Pi)$ gilt.

Wir definieren die Mengen

- $\mathcal{L}_{Dij} := \{L_S \mid S \in Ian(\Omega,\Pi) \text{ und } \exists S' \in Dij(\Omega,\Pi) : S \sim S'\}$
- $\mathcal{L}_{Ian} := \{L_S \mid S \in Ian(\Omega,\Pi)\}$

Für die Sprache L_S eines IANOV-Schemas $S \in Ian(\Omega,\Pi)$ gilt: $L_S \subseteq (B\Omega)^*B$ mit $B = \{0,1\}^n, n = |\Pi|$. L_S besitzt außerdem die folgende **"deterministische" Eigenschaft**:

$$(D)\begin{cases} \text{Für alle } w,v,v' \in (B \cup \Omega)^* \text{ und } f \in \Omega \text{ gilt:} \\ wfv, wv' \in L_S \curvearrowright first(v') = f \end{cases}$$

Definition 3.1 (Schemasprache) *$L \subseteq \mathcal{L}(B\Omega)^*B$ mit der Eigenschaft (D) heißt **Schemasprache** über Ω und Π ($L \in \mathcal{L}(\Omega,\Pi)$).*

$L \in (\Omega,\Pi)$ kann als Pfadmenge eines Baumes aufgefasst werden, in dem Verzweigungen bei Bedingungsknoten auftreten. Beim Abwickeln der Standardform eines IANOV-Schemas erhält man einen *regulären* unendlichen Baum (vgl. Unifikation mit "*occur-check*").

Aus der Definition der Schemasprachen folgt:

- $T \subseteq B \curvearrowright T \in \mathcal{L}(\Omega,\Pi)$
- $f \in \Omega \curvearrowright BfB \in \mathcal{L}(\Omega,\Pi)$
- $L \in \mathcal{L}(\Omega,\Pi), L' \subseteq L \curvearrowright L' \in \mathcal{L}(\Omega,\Pi)$
- $\mathcal{L}_{Dij}(\Omega,\Pi) \subseteq \mathcal{L}_{Ian}(\Omega,\Pi) \subseteq \mathcal{L}(\Omega,\Pi)$

3.3.1 Operationen auf Schemasprachen

Zur Charakterisierung der Sprachen mit der Eigenschaft (D) führen wir Operationen auf $\mathcal{L}(\Omega,\Pi)$ ein, und zwar für *Sequenz*, *Verzweigung* und *bedingte Iterationen* (while-Schleifen).

Definition 3.2 (bedingte Operationen) *Seien $L, L_1, L_2 \subseteq (B\Omega)^*B$ und $T \subseteq B$, $\overline{T} := B \setminus T$. Dann heißt*

(1) ***bedingtes Produkt*** *von L_1 und L_2*
$L_1 \circ L_2 := \{w\beta v \mid w\beta \in L_1, \beta v \in L_2\}$

(2) ***bedingte Vereinigung*** *von L_1 und L_2 bezüglich T*
$L_1 \overset{T}{\cup} L_2 := (T \circ L_1) \cup (\overline{T} \circ L_2)$

(3) ***bedingter Stern*** *von L bezüglich T*
$L^{*,T} := \bigcup_{i=0}^{\infty} (T \circ L)^i \circ \overline{T}$

Dabei ist $L^0 := B$ und $L^{i+1} := L^i \circ L$.

Lemma 3.1 *$\mathcal{L}(\Omega,\Pi)$ ist abgeschlossen unter den bedingten Operationen mit beliebigem $T \subseteq B$.*

Beweis:

Die Eigenschaft (D) bleibt erhalten.

q.e.d.

Satz 3.2 *$\mathcal{L}_{Dij}(\Omega,\Pi)$ ist die kleinste Teilklasse von $\mathcal{L}(\Omega,\Pi)$, welche für jedes $f\in\Omega$ BfB enthält und abgeschlossen ist unter den bedingten Operationen.*

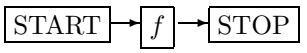
Beweis:

Durch Induktion über den Aufbau von $Dij(\Omega,\Pi)$ und mit Benutzung der entsprechenden IANOV-Schemata.

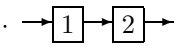
Zunächst wird $b\in BExp(\Pi)$ eine Menge $B(b)\subseteq B$ der Bedingungen zugeordnet, unter denen b wahr ist:

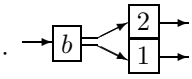
- $B(p_i) := \{\beta\in B \mid proj_i(\beta)=1\}$
- $B(b_1\wedge b_2) := B(b_1)\cap B(b_2)$
- $B(b_1\vee b_2) := B(b_1)\cup B(b_2)$
- $B(\neg b) := B\setminus B(b)$

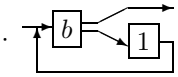
Damit erhalten wir:

- 

hat die Sprache BfB.
- Wenn 

die Schemasprache L_i hat, so hat
  - 

die Sprache $L_1\circ L_2$,
  - 

die Sprache $L_1 \overset{B(b)}{\cup} L_2$,
  - 

die Sprache $L_1^{*,B(b)}$.

q.e.d.

Zum Nachweis von $\mathcal{L}_{Dij}(\Omega,\Pi)\subset\mathcal{L}_{Ian}(\Omega,\Pi)$ benötigen wir zunächst noch den Begriff der *Ableitung einer Schemasprache.*

Definition 3.3 (Ableitung) *Sei $L\in\mathcal{L}(\Omega,\Pi)$ und $w\in(B\Omega)^*B$. Dann heißt*

$$d_w(L) := \{v\in(B\Omega)^*B \mid \exists f\in\Omega : wfv\in L\}$$

*die **Ableitung** von L nach w und*

$$D(L) := \{d_w(L) \mid w\in(B\Omega)^*B\}$$

*ist die **Menge aller Ableitungen** von L.*

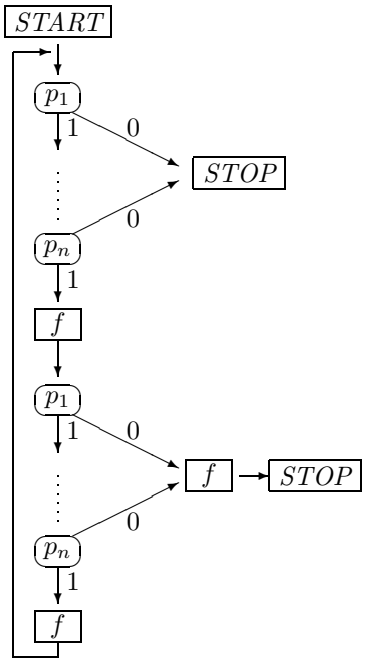
Offensichtlich gilt $d_w(L)\in\mathcal{L}(\Omega,\Pi)$. Das folgende Lemma folgt sofort aus den obigen Definitionen.

Lemma 3.2 *Sei $L,L_1,L_2\in\mathcal{L}(\Omega,\Pi)$, $T\subseteq B$ und $w\in(B\Omega)^*B$. Dann gilt:*

- $d_w(L_1\circ L_2)\in\{d_v(L_2), d_w(L_1)\circ L_2 \mid v\in(B\Omega)^*B\}$
- $d_w(L_1 \overset{T}{\cup} L_2)\in\{d_w(L_1), d_w(L_2)\}$
- $d_w(L^{*,T})\in\{d_v(L)\circ L^{*,T} \mid v\in(B\Omega)^*B\}$

Satz 3.3 *Sei ein* IANOV*-Schema gegeben durch*

$\widehat{S}$:



mit $\tau := \{1\}^n$ *und* $\overline{\tau} := B \setminus \tau$
sowie $\widehat{L} := L_{\widehat{S}} = [\![(\tau f \tau f)^*(\overline{\tau} \vee \tau f \overline{\tau} f B)]\!]$

Dann gilt: $\widehat{L} \in \mathcal{L}_{Ian}(\Omega, \Pi) \setminus \mathcal{L}_{Dij}(\Omega, \Pi)$

Beweis:

Es ist klar, dass $\widehat{L} \in \mathcal{L}_{Ian}(\Omega, \Pi)$. Zu zeigen bleibt also $\widehat{L} \notin \mathcal{L}_{Dij}(\Omega, \Pi)$. Dazu weisen wir nach:

$(*)$ $\left\{ \begin{array}{l} \text{Für jedes } L \in \mathcal{L}_{Dij}(\Omega, \Pi) \text{ gilt:} \\ \widehat{L} \notin \{L\} \cup D(L) \end{array} \right.$

Beweis von $(*)$ durch *strukturelle Induktion* über $L \in \mathcal{L}_{Dij}(\Omega, \Pi)$:

(1) $L = BgB$ mit $g \in \Omega$
$(*)$ gilt für L, weil L und alle $d_w(L)$ im Gegensatz zu $\widehat{L}$ endliche Mengen sind.

(2) $L = L_1 \circ L_2$, Annahme: $(*)$ gelte nicht für L

(a) $\widehat{L} = L_1 \circ L_2$
$[\![\overline{\tau}]\!] \subseteq \widehat{L} \curvearrowright [\![\overline{\tau}]\!] \subseteq L_1 \cap L_2$ $(**)$
Aus $[\![\tau f \overline{\tau} f B]\!] \subseteq \widehat{L}$ folgt, dass gilt: $\tau \in L_1$ oder $\tau \in L_2$, weil die Zerlegung bei $\overline{\tau}$ im Widerspruch zu $(**)$ stünde. Also muss entweder $B \subseteq L_1$ oder $B \subseteq L_2$ sein, und somit $B = L_1$ oder $B = L_2$ gelten. Daraus folgt $\widehat{L} = L_1$ oder $\widehat{L} = L_2 \rightsquigarrow$ Widerspruch zur I.V.

(b) $\widehat{L} = d_w(L_1 \circ L_2)$ mit $w \in (B\Omega)^*B$

(i) $\widehat{L} = d_v(L_2)$ mit $v \in (B\Omega)^*B \rightsquigarrow$ Widerspruch zur I.V.

(ii) $\widehat{L} = d_w(L_1) \circ L_2 \rightsquigarrow$ Widerspruch (i)

(3) $L = L_1 \overset{T}{\cup} L_2$ mit $T \subseteq B$ und $(*)$ gelte für L_i.

(a) $L = L_1 \overset{T}{\cup} L_2$
Da $\widehat{L} = d_{\tau f \tau}(\widehat{L})$ folgt $\widehat{L} \in \{d_{\tau f \tau}(L_i) \mid i = 1, 2\}$ (Lemma 3.2) $\rightsquigarrow$ Widerspruch zur I.V.

(b) $\widehat{L} = d_w(L_1 \overset{T}{\cup} L_2) \in \{d_w(L_i) \mid i = 1, 2\} \rightsquigarrow$ Widerspruch zur I.V.

(4) $L = L_1^{*,T}$

(a) $\widehat{L} = L_1^{*,T}$ mit $T \subseteq B$
$\tau f \overline{\tau} f B \subseteq \widehat{L} \curvearrowright \overline{T} = B \curvearrowright T = \emptyset \curvearrowright \widehat{L} = B \rightsquigarrow$ Widerspruch zur Definition von $\widehat{L}$

(b) $\widehat{L} = d_w(L_1^{*,T}) = d_v(L_1) \circ L_1^{*,T} \rightsquigarrow$ Widerspruch (2).(b).(ii)

q.e.d.

3.4 Übersetzbarkeit von IANOV-Schemata in DIJKSTRA-Schemata

Um ein DIJKSTRA-Schema in ein IANOV-Schema übersetzen zu können, benötigt man zusätzlich *boolsche Variablen*, die beliebige Sprünge simulieren können (BÖHM, JAKOBINI).

Satz 3.4 $Ian(\Omega, \Pi) \sim \bigcup_{m=1}^{\infty} Dij(\Omega, \Pi, X_m)$

Beweis:

Folie 2.6–2.10

$\underline{Ian(\Omega, \Pi) \rightsquigarrow Dij(\Omega, \Pi, X_m)}$:
Wähle $m := |Q|$. O.B.d.A sei $Q := \{q_1, \ldots, q_m\}$ mit Anfangszustand q_1 und *einzigem* Stopzustand q_m.

$$S_i := \begin{cases} x_i \leftarrow \mathsf{false}; f; x_j \leftarrow \mathsf{true} & : \quad S(q_i) = (f, q_j) \\ x_i \leftarrow \mathsf{false}\ ; \mathsf{if}\ p\ \mathsf{then}\ x_j \leftarrow \mathsf{true}\ \mathsf{else}\ x_k \leftarrow \mathsf{true} & : \quad S(q_i) = (p, q_j, q_k) \end{cases}$$

für $i = 1, \ldots, m-1$

```
S' :=  x_1 ← true; x_2 ← false; ...; x_m ← false;
       while ¬x_m do
           if x_1 then S_1 else
               if x_2 then S_2 else
                   ⋮
                   if x_{m-1} then S_{m-1} else f fi   // else-Fall tritt nie auf
               ⋮
               fi
           fi
       end
```

Offensichtlich gilt $S \sim S'$.

$\underline{Dij(\Omega, \Pi, X_m) \rightsquigarrow Ian(\Omega, \Pi)}$:
Zunächst erweitern wir $Ian(\Omega, \Pi)$ zu $Ian(\Omega, \Pi, X_m)$ (analog zu $Dij(\Omega, \Pi, X_m)$) um das Setzen und Testen der boolschen Variablen.
$\curvearrowright Dij(\Omega, \Pi, X_m) \rightsquigarrow Ian(\Omega, \Pi, X_m)$
Durch Simulation der boolschen Variablen in der "endlichen Kontrolle" kann man nun zeigen, dass $Ian(\Omega, \Pi, X_m) \rightsquigarrow Ian(\Omega, \Pi)$ gilt:

Setze $Q' := Q \times \{0,1\}^m$.

[siehe Aufgabe 5.12]

q.e.d.

Kapitel 4

KOSARAJU-Schemata

Was wir bisher gesehen haben war, dass IANOV-Schemata Flussdiagrammen entsprechen; ferner stellen DIJKSTRA-Schemata Flussdiagramme mit eingeschränkter Schleifenstruktur dar, d. h. man kann eine Schleife nur an einer Stelle verlassen. KNUTH gab 1974 mit seiner Arbeit *"structured programming with goto statements"* eine Anwort auf DIJKSTRA. KOSARAJU zeigte später, dass man die Aussagekraft von IANOV-Schemata erhält, indem man die DIJKSTRA-Schemata um ein Konstrukt erweitert, das es erlaubt, eine Schleife an mehr als einem Ausgang zu verlassen (*"analysis of structured programs"*, 1974). Die Idee dabei ist, dass man Schleifen in Flussdiagrammen soweit abwickelt, bis es nur noch Sprünge enthält, die zu Vorgängerknoten gehen. Man nennt das dann entstehende Flussdiagramm *Blocknormalform* (ENGLER).

4.1 Syntax von KOSARAJU-Schemata

Sei Ω eine Menge von einstelligen *Operationssymbolen* und Π eine Menge von einstelligen *Prädikatssymbolen*. Seien ferner $S_1 \in Kos_{n_1}(\Omega, \Pi)$, $S_2 \in Kos_{n_2}(\Omega, \Pi)$, $S \in Kos_{n+1}(\Omega, \Pi)$, $n_{max} := max\{n_1, n_2\}$ und $b \in BExp(\Pi)$. Dann ist die Menge der KOSARAJU-**Schemata**

$$Kos(\Omega, \Pi) = \bigcup_{n=0}^{\infty} Kos_n(\Omega, \Pi)$$

induktiv definiert durch

- $\Omega \subseteq Kos_0(\Omega, \Pi)$
- exit n $\in Kos_n(\Omega, \Pi) \quad (n > 0)$
- $(S_1; S_2) \in Kos_{n_{max}}(\Omega, \Pi)$
- if b then S_1 else S_2 fi $\in Kos_{n_{max}}(\Omega, \Pi)$
- repeat S end $\in Kos_n(\Omega, \Pi)$

4.2 Semantik von KOSARAJU-Schemata

Sei $\mathfrak{A} := \langle A; \alpha \rangle \in Int(\Omega, \Pi)$, $S \in Kos_n(\Omega, \Pi)$.
Dann ist $\langle S, \mathfrak{A} \rangle$ ein KOSARAJU-**Programm** mit $n + 1$ Ausgängen.

Sei $f \in \Omega$, $a \in A$, $S_1 \in Kos_{n_1}(\Omega,\Pi)$, $S_2 \in Kos_{n_2}(\Omega,\Pi)$, $S \in Kos_{n+1}(\Omega,\Pi)$ und $b \in BExp(\Pi)$. Dann ist die Abbildung

$$[\![S]\!]_{\mathfrak{A}} : A \dashrightarrow A \times \{0,1,\ldots n\}$$

induktiv definiert durch

- $[\![f]\!]_{\mathfrak{A}}(a) := \big(\alpha(f)(a), 0\big)$
- $[\![\mathsf{exit\ n}]\!]_{\mathfrak{A}}(a) := (a,n)$
- $[\![(S_1;S_2)]\!]_{\mathfrak{A}}(a) := \begin{cases} [\![S_2]\!]_{\mathfrak{A}}(a') & : \quad [\![S_1]\!]_{\mathfrak{A}}(a) = (a',0) \\ [\![S_1]\!]_{\mathfrak{A}}(a) & : \quad \text{sonst} \end{cases}$
- $[\![\mathsf{if}\ b\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2\ \mathsf{fi}]\!]_{\mathfrak{A}}(a) := \begin{cases} [\![S_1]\!]_{\mathfrak{A}}(a) & : \quad [\![b]\!]_{\mathfrak{A}}(a) = 1 \\ [\![S_2]\!]_{\mathfrak{A}}(a) & : \quad [\![b]\!]_{\mathfrak{A}}(a) = 0 \end{cases}$
- $[\![\mathsf{repeat}\ S\ \mathsf{end}]\!]_{\mathfrak{A}} : A \dashrightarrow A \times \{0,1,\ldots n-1\}$

  $[\![\mathsf{repeat}\ S\ \mathsf{end}]\!]_{\mathfrak{A}}(a) := \begin{cases} (t^k(a), m-1) & : \quad e\big(t^i(a)\big) = 0 \text{ für } 0 \le i \le k-2 \\ & \quad \text{und } e\big(t^{k-1}(a)\big) = m > 0 \\ \bot & : \quad \text{sonst} \end{cases}$

  wobei
  - $t := first \circ [\![S]\!]_{\mathfrak{A}}$ (Transformation)
  - $e := second \circ [\![S]\!]_{\mathfrak{A}}$ (Exit-Funktion)

Im weiteren Verlauf verfolgen wir nun das Ziel nachzuweisen, dass $Ian(\Omega,\Pi) \sim Kos_0(\Omega,\Pi)$ gilt. Dazu erweitern wir die IANOV-Schemata um meherere Ausgänge: $Ian_n(\Omega,\Pi)$ mit $(n+1)$ Stopbefehlen $\mathsf{stop\ 0}$, …, $\mathsf{stop\ n}$

Für $S \in Ian_n(\Omega,\Pi)$ und $\mathfrak{A} \in Int(\Omega,\Pi)$ sei dann:
$[\![S]\!]_{\mathfrak{A}} : A \dashrightarrow A \times \{0,1,\ldots,n\}$

Satz 4.1 $Kos_n(\Omega,\Pi) \rightsquigarrow Ian_n(\Omega,\Pi) \quad \forall n \in \mathbb{N}$

Beweis:

durch Konstruktion äquivalenter Flussdiagramme: Folie 3.2

q.e.d.

4.3 Blocknormalform (BNF) von Flussdiagrammen

Die Übersetzung von $Kos_n(\Omega,\Pi)$ nach $Ian_n(\Omega,\Pi)$ bestimmt eine induktiv aufgebaute Teilklasse: die entstehenden IANOV-Schemata sind in der *Blocknormalform (BNF)*, d. h. Sprünge gehen nur noch zu Vorgängerknoten. Umgekehrt gilt aber auch, dass jedes Flussdiagramm durch Abwickeln von Schleifen in BNF transformierbar ist. Formal bedeutet dies, dass die Übersetzung von $Ian_n(\Omega,\Pi)$ nach $Kos_n(\Omega,\Pi)$ als das Auflösen eines Funktionsgleichungssystems zu verstehen ist.

4.3.1 Fortsetzungssemantik von IANOV-Schemata

$S : Q \longrightarrow (\Omega \times Q) \cup (\Pi \times Q \times Q) \cup \{\mathsf{stop}\}$
$\mathfrak{A} = \langle A; \alpha \rangle \in Int(\Omega, \Pi)$
$Q = \{q_1, \ldots, q_r\}$ mit Startzustand q_1

Sei $S_i \in Ian(\Omega, \pi)$ wie S, aber mit Startzustand q_i. Dann gilt:

(1) $S(q_i) = (f, q_j) \curvearrowright [\![S_i]\!]_{\mathfrak{A}} = [\![S_j]\!]_{\mathfrak{A}} \circ f_{\mathfrak{A}}$

(2) $S(q_i) = (p, q_j, q_k) \curvearrowright [\![S_i]\!]_{\mathfrak{A}} = \mathsf{if}\ p_{\mathfrak{A}}\ \mathsf{then}\ [\![S_j]\!]_{\mathfrak{A}}\ \mathsf{else}\ [\![S_k]\!]_{\mathfrak{A}}\ \mathsf{fi}$

(3) $S(q_i) = \mathsf{stop} \curvearrowright [\![S_i]\!]_{\mathfrak{A}} = id_A$

Folge:

$[\![S]\!]_{\mathfrak{A}}$ ist die Lösung eines *Funktionsgleichungssystems*. $[\![S]\!]_{\mathfrak{A}}$ ist die kleinste Lösung im folgenden Sinne:

$$[A - \rightharpoonup A] := \{g \mid g : A - \rightharpoonup A\}$$

ist eine Halbordung bezüglich $g_1 \leq g_2 :\curvearrowright graph(g_1) \subseteq graph(g_2)$. $[A - \rightharpoonup A]$

- ist **(ketten-) vollständig**, d. h. eine Folge $g_1 \leq g_2 \leq \ldots$ besitzt eine kleinste obere Schranke g mit $graph(g) = \bigcup_{i=1}^{\infty} graph(g_i)$.
- besitzt ein *kleinstes Element* $g_\emptyset$ mit $graph(g_\emptyset) = \emptyset$.

Wir sagen $[A - \rightharpoonup A]$ ist eine **vollständige Halbordnung**. Das Funktionsgleichungssystem von $\langle S, \mathfrak{A} \rangle$ bestimmt eine stetige Transformation

$$T_{\langle S, \mathfrak{A} \rangle} : [A - \rightharpoonup A]^r \longrightarrow [A - \rightharpoonup A]^r$$

Die erste Komponente des kleinesten Fixpunktes ist die Semantik:

$$[\![S]\!]_{\mathfrak{A}} = first\big(fix(T_{\langle S, \mathfrak{A} \rangle})\big)$$

4.3.2 Reguläre Schemata

Reguläre Sprachen stellen eine Zwischenstufe zur Auflösung der Funktionsgleichungssysteme von IANOV-Schemata dar.

Syntax von regulären Schemata

Sei $Reg(\Omega, \Pi)$ die Menge der regulären Schemata. $Reg(\Omega, \Pi) := \bigcup_{n=0}^{\infty} Reg_n(\Omega, \Pi)$ ist induktiv definiert durch:

- $x_i \in Reg_n(\Omega, \Pi)$, falls $1 \leq i \leq n$
- $id \in Reg_n(\Omega, \Pi)$ für alle $n \in \mathbb{N}$
- $f(t) \in Reg_n(\Omega, \Pi)$, falls $f \in \Omega$ und $t \in Reg_n(\Omega, \Pi)$
- $p(t_1, t_2) \in Reg_n(\Omega, \Pi)$, falls $p \in \Pi$ und $t_1, t_2 \in Reg_n(\Omega, \Pi)$
- $\mathsf{iter}(t) \in Reg_n(\Omega, \Pi)$, falls $t \in Reg_{n+1}(\Omega, \Pi)$

Semantik von regulären Schemata

Sei $\mathfrak{A} \in Int(\Omega, \Pi)$ und $t \in Reg_n(\Omega, \Pi)$. Dann ist die Semantik der regulären Schemata

$$[\![t]\!]_{\mathfrak{A}} : [A - \rightarrow A]^n \longrightarrow [A - \rightarrow A]$$

induktiv definiert durch:

- $[\![x_i]\!]_{\mathfrak{A}}(g_1, \ldots, g_n) := g_i$
- $[\![id]\!]_{\mathfrak{A}}(g_1, \ldots, g_n) := id_A$
- $[\![f(t)]\!]_{\mathfrak{A}}(g_1, \ldots, g_n) := [\![t]\!]_{\mathfrak{A}}(g_1, \ldots, g_n) \circ f_{\mathfrak{A}}$[1]
- $[\![p(t_1, t_2)]\!]_{\mathfrak{A}}(g_1, \ldots, g_n) :=$ if $p_{\mathfrak{A}}$ then $[\![t_1]\!]_{\mathfrak{A}}(g_1, \ldots, g_n)$ else $[\![t_2]\!]_{\mathfrak{A}}(g_1, \ldots, g_n)$ fi
- $[\![\mathsf{iter}(t)]\!]_{\mathfrak{A}}(g_1, \ldots, g_n) := fix([\![t]\!]_{\mathfrak{A}})(g_1, \ldots, g_n)$

  Dabei ist $[\![t]\!]_{\mathfrak{A}} : [A - \rightarrow A]^{n+1} \longrightarrow [A - \rightarrow A]$
  und $fix([\![t]\!]_{\mathfrak{A}}) : [A - \rightarrow A]^n \longrightarrow [A - \rightarrow A]$
  mit $fix([\![t]\!]_{\mathfrak{A}})(g_1, \ldots, g_n) := fp\big(g_{n+1} \mapsto [\![t]\!]_{\mathfrak{A}}(g_1, \ldots, g_{n+1})\big)$

  Also gilt: $[\![\mathsf{iter}(t)]\!]_{\mathfrak{A}}(g_1 \ldots, g_n)$ ist kleinste Lösung von

  $$F := [\![t]\!]_{\mathfrak{A}}(g_1, \ldots, g_n, F)$$

Satz 4.2 $Ian(\Omega, \Pi) \rightsquigarrow Reg_0(\Omega, \Pi)$

Beweis:

$S : Q \longrightarrow (\Omega \times Q) \cup (\Pi \times Q \times Q) \cup \{\mathsf{stop}\}$ mit $Q := \{q_1, \ldots, q_r\}$

$$(G_r) \left\{ \begin{array}{ccc} x_1 & = & t_1 \\ & \vdots & \\ x_r & = & t_r \end{array} \right. \quad \text{mit } t_i = \left\{ \begin{array}{ccc} f(x_j) & : & S(q_i) = (f, q_j) \\ p(x_j, x_k) & : & S(q_i) = (p, q_j, q_k) \\ id & : & S(q_i) = \mathsf{stop} \end{array} \right.$$

Es gilt: $t_i \in Reg_r(\Omega, \Pi)$

(G_r) lässt sich in $Reg(\Omega, \Pi)$ durch Iteration nach x_1 auflösen:

1. Schritt:

$$(G_{r-1}) \left\{ \begin{array}{ccc} x_1 & = & t'_1 \\ & \vdots & \\ x_{r-1} & = & t'_{r-1} \end{array} \right. \quad \text{mit } t'_i = t_i[x_r / iter(t_r)] \in Reg_{r-1}(\Omega, \Pi)$$

Nach r Schritten ergibt sich $t \in Reg_0(\Omega, \Pi)$. Es gilt dann $S \sim t$, wegen der funktionalen Semantik von $\langle S, \mathfrak{A} \rangle$ und der Semantik von $Reg(\Omega, \Pi)$ bezüglich $\mathfrak{A}$.

q.e.d.

[1] Man beachte hierbei die "vertauschte" Reihenfolge

4.4 Fortsetzungssemantik von KOSARAJU-Schemata

Die Idee der Fortsetzungssemantik von KOSARAJU-Schemata ist es, die Nebenausgänge $1, \ldots, n$ als Anschlüsse für die Fortsetzungen zu benutzen. Eine weitere Fortsetzung wird durch den Hauptausgang 0 reserviert, wobei am Hauptausgang die Iteration erfolgt.

Sei $S \in Kos_n(\Omega, \Pi)$, $\mathfrak{A} \in Int(\Omega, \Pi)$. Wir definieren die *Fortsetzungssemantik von* KOSARAJU-*Schemata*

$$[\![S]\!]^c_{\mathfrak{A}} : [A \dashrightarrow A]^{n+1} \longrightarrow [A \dashrightarrow A]$$

induktiv durch:

- $[\![f]\!]^c_{\mathfrak{A}}(g_0) := g_0 \circ f_{\mathfrak{A}}$
- $[\![\mathsf{exit\ n}]\!]^c_{\mathfrak{A}}(g_0, \ldots, g_n) := g_n$
- $[\![(S_1; S_2)]\!]^c_{\mathfrak{A}}(g_0, \ldots, g_{n_{max}}) := [\![S_1]\!]^c_{\mathfrak{A}}\big([\![S_2]\!]^c_{\mathfrak{A}}(g_0, \ldots, g_{n_2}), g_1, \ldots, g_{n_1}\big)$
- $[\![\mathsf{if}\ b\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2\ \mathsf{fi}]\!]^c_{\mathfrak{A}}(g_0 \ldots, g_{n_{max}})$
  $:= \mathsf{if}\ [\![b]\!]^c_{\mathfrak{A}} = 1\ \mathsf{then}\ [\![S_1]\!]^c_{\mathfrak{A}}(g_0, \ldots, g_{n_1})\ \mathsf{else}\ [\![S_2]\!]^c_{\mathfrak{A}}(g_0, \ldots, g_{n_2})\ \mathsf{fi}$
- $[\![\mathsf{repeat}\ S\ \mathsf{end}]\!]^c_{\mathfrak{A}}(g_0, \ldots, g_n) := fix\big(g \mapsto [\![S]\!]^c_{\mathfrak{A}}(g, g_0, \ldots, g_n)\big)$

Lemma 4.1 *Für $S \in Kos(\Omega, \Pi)$, $\mathfrak{A} \in Int(\Omega, \Pi)$ gilt: $[\![S]\!]^c_{\mathfrak{A}}(g_0, \ldots, g_n)(a) = g_i(a')$, falls $[\![S]\!]_{\mathfrak{A}}(a) = (a', i)$.*

Satz 4.3 $Reg_0(\Omega, \Pi) \rightsquigarrow Kos_0(\Omega, \Pi)$

Beweis:

Übersetzung von $Reg_n(\Omega, \Pi)$ nach $Kos_n(\Omega, \Pi)$:

$$trans : Reg(\Omega, \Pi) \longrightarrow Kos(\Omega, \Pi)$$

x_i	$\rightsquigarrow$	$trans(x_i) :=$ exit n-i
id	$\rightsquigarrow$	$trans(id) :=$ exit n
$f(t)$	$\rightsquigarrow$	$trans\big(f(t)\big) := \big(f; trans(t)\big)$
$p(t_1, t_2)$	$\rightsquigarrow$	$trans\big(p(t_1, t_2)\big) :=$ if p then $trans(t_1)$ else $trans(t_2)$ fi
iter(t)	$\rightsquigarrow$	$trans\big(\mathsf{iter}(t)\big) :=$ repeat $trans(t)$ end

exit 0 wird am Ende ersetzt durch repeat exit 1 end.

q.e.d.

Korollar 4.1 $Ian(\Omega, \Pi) \sim Kos_0(\Omega, \Pi)$

KOSARAJU-**Hierarchie:**

Seien $Kos_0^{(k)}(\Omega, \Pi)$ die KOSARAJU-Schemata mit 0 Ausgängen und höchstem exit-Level k. Dann gilt:

$$Kos_0^{(k)}(\Omega, \Pi) \rightsquigarrow Kos_0^{(k+1)}(\Omega, \Pi)$$

$$Kos_0^{(k+1)}(\Omega, \Pi) \not\rightsquigarrow Kos_0^{(k)}(\Omega, \Pi)$$

Dieses Ergebnis hängt mit der *Sternhöhenhierarchie* zusammen.

Kapitel 5

De Bakker-Scott-Schemata

Wir betrachten in diesem Kapitel die *Rekursion* als Kontrollstruktur, wie man sie von rekursiven Prozeduren kennt. Ein Problem der Rekursion ist die ineffiziente Implementierung, daher wurde sie beispielsweise in FORTRAN nicht implementiert. Knuth vertrat die Meinung, dass man Rekursion nicht benötigt, da man sie ohne weiteres in *Iteration* übersetzten kann. Wir werden jedoch sehen, dass die Rekursion mächtiger als die Iteration ist, dass man also einen *Keller-Speicher* benötigt, um die Rekursion zu simulieren. Im Folgenden werden wir vom Zustandsraum abstrahieren und uns auf einfache rekursive Schemata konzentrieren.

5.1 Syntax von De Bakker-Scott-Schemata

Seien Ω und Π wie bisher.
$F_1, F_2, \ldots$ seien **Funktionsvariablen** und x eine **Argumentvariable**

Die Menge T_r der Terme über Ω, Π und $F_1, \ldots, F_r$ ist induktiv definiert durch:

- $x \in T_r$
- $f(t) \in T_r$, falls $f \in \Omega$, $t \in T_r$
- $\mathsf{if}\ p(x)\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2\ \mathsf{fi} \in T_r$, falls $p \in \Pi$, $t_1, t_2 \in T_r$
- $F_i(t) \in T_r$, falls $1 \leq i \leq r$, $t \in T_r$

De Bakker-Scott-**Schemata** $S \in dBS(\Omega, \Pi)$ bestehen aus Gleichungsschemata der Form

$$S \left\{ \begin{array}{lcll} F_1(x) & = & t_1 & \\ & \vdots & & \text{mit } t_i \in T_r \\ F_r(x) & = & t_r & \end{array} \right.$$

5.2 Fixpunktsematik von De Bakker-Scott-Schemata

Sei $S = (F_i(x) = t_i \mid 1 \leq i \leq r) \in dBS(\Omega, \Pi)$, $\mathfrak{A} = \langle A; \alpha \rangle \in Int(\Omega, \Pi)$

Das De Bakker-Scott-Programm $\langle S, \mathfrak{A} \rangle$ bestimmt

$$[\![S]\!]_{\mathfrak{A}} : A \dashrightarrow A$$

Zu ihrer Definition erklären wir zunächst die *Termsemantik*

$$[\![t]\!]_{\mathfrak{A}} : [A \dashrightarrow A]^r \times A \dashrightarrow A$$

durch

- $[\![x]\!]_{\mathfrak{A}}(g_1, \ldots, g_r, a) := a$
- $[\![f(t)]\!]_{\mathfrak{A}}(g_1, \ldots, g_r, a) := \alpha(f)\big([\![t]\!]_{\mathfrak{A}}(g_1, \ldots, g_r, a)\big)$
- $[\![\mathsf{if}\ p(x)\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2\ \mathsf{fi}]\!]_{\mathfrak{A}}(g_1, \ldots, g_r, a) := \begin{cases} [\![t_1]\!]_{\mathfrak{A}}(g_1, \ldots, g_r, a) : \alpha(p)(a) = 1 \\ [\![t_2]\!]_{\mathfrak{A}}(g_1, \ldots, g_r, a) : \alpha(p)(a) = 0 \end{cases}$
- $[\![F_i(t)]\!]_{\mathfrak{A}}(g_1, \ldots, g_r, a) := g_i\big([\![t]\!]_{\mathfrak{A}}(g_1, \ldots, g_r, a)\big)$

Über die Termsemantik der rechten Seiten $t_1, \ldots, t_r$ erhalten wir die *Transformation*

$$T_{(S,\mathfrak{A})} : [A \dashrightarrow A]^r \longrightarrow [A \dashrightarrow A]^r$$

$$T_{(S,\mathfrak{A})}(\overline{g}) := \Big(\big(a \mapsto [\![t_1]\!]_{\mathfrak{A}}(\overline{g}, a)\big), \ldots, \big(a \mapsto [\![t_r]\!]_{\mathfrak{A}}(\overline{g}, a)\big)\Big)$$

Es folgt:

$\overline{g}$ ist eine Lösung von $\langle S, \mathfrak{A}\rangle$
$\curvearrowright\!\!\curvearrowright$ $g_i(a) = [\![t_i]\!]_{\mathfrak{A}}(\overline{g}, a)$
$\curvearrowright\!\!\curvearrowright$ $\overline{g} = T_{(S,\mathfrak{A})}(\overline{g})$

Da $T_{(S,\mathfrak{A})}$ "stetig" ist, besitzt $\langle S, \mathfrak{A}\rangle$ eine kleinste Lösung:

$$fix(T_{(S,\mathfrak{A})}) := \bigsqcup_{i=0}^{\infty} T^i_{(S,\mathfrak{A})}(\underbrace{g_\emptyset, \ldots, g_\emptyset}_{r-mal})$$

Wir bezeichnen die Lösungsfunktionen durch $[\![S]\!]^{(i)}_{\mathfrak{A}}$, also:

$$fix(T_{(S,\mathfrak{A})}) := \big([\![S]\!]^{(1)}_{\mathfrak{A}}, \ldots, [\![S]\!]^{(r)}_{\mathfrak{A}}\big)$$

und erklären die erste Komponente als Semantik von $\langle S, \mathfrak{A}\rangle$:

$$[\![S]\!]_{\mathfrak{A}} := [\![S]\!]^{(i)}_{\mathfrak{A}}$$

5.3 Reduktionssemantik von De Bakker-Scott-Schemata

Sei $S = (F_i(x) = t_i \;\big|\; 1 \le i \le r) \in dBS(\Omega, \Pi)$, $\mathfrak{A} = \langle A; \alpha\rangle \in Int(\Omega, \Pi)$

$Comp(S, \mathfrak{A})$ sei die Menge der **Berechnungsterme** über $\langle S, \mathfrak{A}\rangle$:

- $A \subseteq Comp(S, \mathfrak{A})$
- $f(u) \in Comp(S, \mathfrak{A})$, falls $f \in \Omega$, $u \in Comp(S, \mathfrak{A})$
- $\mathsf{if}\ p(u_1)\ \mathsf{then}\ u_2\ \mathsf{else}\ u_3\ \mathsf{fi} \in Comp(S, \mathfrak{A})$, falls $p \in \Pi$, $u_1, u_2, u_3 \in Comp(S, \mathfrak{A})$
- $F_i(u) \in Comp(S, \mathfrak{A})$, falls $1 \le i \le r$, $u \in Comp(S, \mathfrak{A})$

Berechnungsregeln:

- *Konstantenreduktion*: $f(a) \to f_{\mathfrak{A}}(a)$
- $\mathsf{if}\ p(a)\ \mathsf{then}\ u_1\ \mathsf{else}\ u_0\ \mathsf{fi} \to u_{p_{\mathfrak{A}}(a)}$
- *Funktionsaufruf*: $F_i(u) \to t_i[x/u]$

Reduktionsrelation:

$\Rightarrow \subseteq Comp(S,\mathfrak{A})^2$ ist induktiv über den Aufbau der Berechnungsterme festgelegt:

- $u \Rightarrow u'$, falls $u \to u'$
- $u \Rightarrow u$, falls $u \in Comp(S,\mathfrak{A})$
- $f(u) \Rightarrow f(u')$, falls $u \Rightarrow u'$
- $\mathsf{if}\ p(u_1)\ \mathsf{then}\ u_2\ \mathsf{else}\ u_3\ \mathsf{fi} \Rightarrow \mathsf{if}\ p(u_1')\ \mathsf{then}\ u_2'\ \mathsf{else}\ u_3'\ \mathsf{fi}$, falls $u_1 \Rightarrow u_1'$, $u_2 \Rightarrow u_2'$, $u_3 \Rightarrow u_3'$
- $F_i(u) \Rightarrow F_i(u')$, falls $u \Rightarrow u'$

Ziel: $[\![S]\!]_{\mathfrak{A}}(a) = a' \curvearrowright F_1(a) \Rightarrow^* a'$

Beobachtungen:

(1) $\Rightarrow$ ist nicht-deterministisch

(2) $\Rightarrow$ erlaubt parallele Ersetzungen in einem Schritt.
Insbesondere gilt: $t \in T_r, u \Rightarrow u' \curvearrowright t[x/u] \Rightarrow t[x/u']$

5.3.1 Die Church-Rosser-Eigenschaft von $\Rightarrow$

Wir zeigen:

$$F_1(a) \Rightarrow^* a_1 \text{ und } F_1(a) \Rightarrow^* a_2 \curvearrowright a_1 = a_2$$

Dazu werden wir zunächst die Fixpunktsemantik auf Berechnungsterme ausdehnen und danach zeigen, dass diese unter Reduktion invariant ist.

Fixpunktsematik für $Comp(S,\mathfrak{A})$

Sei $u \in Comp(S,\Omega)$, $[\![S]\!]_{\mathfrak{A}}^{(i)}$ die i-te Lösungsfunktion von $(S,\mathfrak{A})$

$[\![u]\!]_{\mathfrak{A}}$ ist induktiv definiert durch:

- $[\![a]\!]_{\mathfrak{A}} := a$
- $[\![f(u)]\!]_{\mathfrak{A}} := f_{\mathfrak{A}}\big([\![u]\!]_{\mathfrak{A}}\big)$
- $[\![\mathsf{if}\ p(u)\ \mathsf{then}\ u_1\ \mathsf{else}\ u_0\ \mathsf{fi}]\!]_{\mathfrak{A}} := [\![u_{p_{\mathfrak{A}}([\![u]\!]_{\mathfrak{A}})}]\!]_{\mathfrak{A}}$
- $[\![F_i(u)]\!]_{\mathfrak{A}} := [\![S]\!]_{\mathfrak{A}}^{(i)}\big([\![u]\!]_{\mathfrak{A}}\big)$

Lemma 5.1 *Für* $S = (F_i(x) = t_i \mid 1 \le i \le n) \in dBS(\Omega,\Pi)$, $\mathfrak{A} = \langle A;\alpha\rangle \in Int(\Omega,\Pi)$, $t \in T_r$ *und* $u \in Comp(S,\mathfrak{A})$ *gilt:*

$$[\![t[x/u]]\!]_{\mathfrak{A}} = [\![t]\!]_{\mathfrak{A}}\big([\![S]\!]_{\mathfrak{A}}^{(1)}, \ldots, [\![S]\!]_{\mathfrak{A}}^{(r)}, [\![u]\!]_{\mathfrak{A}}\big)$$

Lemma 5.2 *Für $u, u' \in Comp(S, \mathfrak{A})$ gilt:*

$$u \to u' \curvearrowright [\![u]\!]_{\mathfrak{A}} = [\![u']\!]_{\mathfrak{A}}$$

Beweis:

durch Induktion über den Aufbau der Reduktionsrelation, Korrektheit der Funktionsaufrufregel $F_i(u) \Rightarrow t_i[x/u]$

$$\begin{aligned} [\![F_i(u)]\!]_{\mathfrak{A}} &= [\![S]\!]^{(i)}_{\mathfrak{A}}\big([\![u]\!]_{\mathfrak{A}}\big) \\ [\![t_i[x/u]]\!]_{\mathfrak{A}} &= [\![t_i]\!]_{\mathfrak{A}}\big([\![S]\!]^{(1)}_{\mathfrak{A}}, \ldots, [\![S]\!]^{(r)}_{\mathfrak{A}}, [\![u]\!]_{\mathfrak{A}}\big) \\ &= [\![S]\!]^{(i)}_{\mathfrak{A}}\big([\![u]\!]_{\mathfrak{A}}\big) \end{aligned}$$

q.e.d.

Korollar 5.1 *Für $u \in Comp(S, \mathfrak{A})$ gilt:*

$$u \Rightarrow^* a_1 \text{ und } u \Rightarrow^* a_2 \curvearrowright a_1 = a_2$$

Definition 5.1 (Reduktionssematik) *Für ein dBS-Programm $\langle S, \mathfrak{A}\rangle$ definieren wir:*

$$[\![S]\!]^{red}_{\mathfrak{A}}(a) = a' :\curvearrowright\curvearrowright F_1(a) \Rightarrow^* a'$$

5.3.2 Deterministische Reduktionsstrategien

Die call-by-name-Reduktionsrelation

Alternative Bezeichnungen hierfür sind:

- *left-most-outer-most*
- *outside-in*

$\Rightarrow_o \subseteq \Rightarrow$ ist definiert durch

- $u \Rightarrow_o u'$, falls $u \to u'$
- $f(u) \Rightarrow_o f(u')$, falls $u \Rightarrow_o u'$
- $\mathsf{if}\ p(u)\ \mathsf{then}\ u_1\ \mathsf{else}\ u_0\ \mathsf{fi} \Rightarrow_o \mathsf{if}\ p(u')\ \mathsf{then}\ u_1\ \mathsf{else}\ u_0\ \mathsf{fi}$, falls $u \Rightarrow_o u'$

Folgerung:

$\Rightarrow_o$ ist deterministisch, d. h.:

$$u \Rightarrow_o u_1 \text{ und } u \Rightarrow_o u_2 \curvearrowright u_1 = u_2$$

Satz 5.1 (Standardisierungssatz) *Für $u \in Comp(S, \mathfrak{A})$ und $a \in A$ gilt:*

$$u \Rightarrow^* a \curvearrowright u \Rightarrow^*_o a$$

Beweis:

durch Induktion über die Länge n der Reduktionsschritte:

I.A.: $n = 0$: $u \Rightarrow^0 a \curvearrowright u = a \curvearrowright u \Rightarrow_o^0 a$

I.V.: $u \Rightarrow^n a \curvearrowright u \Rightarrow_o^* a$

I.S.: Sei $u \Rightarrow^{n+1} a$

(1) $u = a' \curvearrowright a' = a \curvearrowright u \Rightarrow_o^0 a$

(2) $u = f(u') \curvearrowright f(u') \Rightarrow^n f(a') \Rightarrow a \curvearrowright u' \Rightarrow^n a \overset{I.V.}{\curvearrowright} u' \Rightarrow_o^* a'$
$u = f(u') \Rightarrow_o^* f(a') \Rightarrow^0 a$

(3) $u = \mathsf{if}\ p(u')\ \mathsf{then}\ u_1\ \mathsf{else}\ u_0\ \mathsf{fi} \Rightarrow^{n+1} a$
$\curvearrowright u \Rightarrow^l \mathsf{if}\ p(a')\ \mathsf{then}\ u_1'\ \mathsf{else}\ u_0'\ \mathsf{fi} \Rightarrow u'_{p_{\mathfrak{A}}(a')} \Rightarrow^k a$
$\overset{I.V.}{\curvearrowright} u' \Rightarrow^l a'$ und $u'_{p_{\mathfrak{A}}(a')} \Rightarrow_o^* a$
$\curvearrowright u \Rightarrow_o^* \mathsf{if}\ p(a')\ \mathsf{then}\ u_1\ \mathsf{else}\ u_0\ \mathsf{fi} \Rightarrow_o u_{p_{\mathfrak{A}}(a')} \Rightarrow^l u'_{p_{\mathfrak{A}}(a')} \Rightarrow^k a$
$\overset{I.V.}{\curvearrowright} u \Rightarrow_o^* u_{p_{\mathfrak{A}}(a')} \Rightarrow_o^* a \quad (l + k = n)$

(4) $u = F_i(u') \Rightarrow^{n+1} a$
$\curvearrowright F_i(u') \Rightarrow^k F_i(u'') \Rightarrow t_i[x/u''] \Rightarrow^l a$
Da $u' \Rightarrow^k u''$ gilt: $t_i[x/u'] \Rightarrow^k t_i[x/u'']$
$\curvearrowright F_i(u') \Rightarrow_o t_i[x/u'] \Rightarrow^k t_i[x/u''] \Rightarrow^l a$

q.e.d.

Die call-by-value-Reduktionsrelation

Alternative Bezeichnungen hierfür sind:

- *right-most-inner-most*
- *inside-out*

$\Rightarrow_i \subseteq \Rightarrow$ ist definiert durch

- $\rightarrow_i$ wie $\rightarrow$ bis auf den eingeschränkten Funktionsaufruf:
$F_i(a) \rightarrow_i t_i[x/a]$
- $u \Rightarrow_i u'$, falls $u \rightarrow_i u'$
- $f(u) \Rightarrow_i f(u')$, falls $u \Rightarrow_i u'$
- $\mathsf{if}\ p(u)\ \mathsf{then}\ u_1\ \mathsf{else}\ u_0\ \mathsf{fi} \Rightarrow_i \mathsf{if}\ p(u')\ \mathsf{then}\ u_1\ \mathsf{else}\ u_0\ \mathsf{fi}$, falls $u \Rightarrow_i u'$
- $F_i(u) \Rightarrow_i F_i(u')$, falls $u \Rightarrow_i u'$

Folgerung:

$\Rightarrow_i$ ist deterministisch, d. h.:

$$u \Rightarrow_i u_1 \text{ und } u \Rightarrow_i u_2 \curvearrowright u_1 = u_2$$

Unser Ziel ist es nun, die Vollständigkeit der *call-by-value*-Strategie zu zeigen.

Lemma 5.3 *Für $t \in T_r$, $u \in Comp(S, \mathfrak{A})$ und $a \in A$ gilt:*

$$t[x/u] \Rightarrow^n a \curvearrowright \exists a' \in A : \exists n_1, n_2 \in \mathbb{N} :$$
$$u \Rightarrow^{n_1} a' \text{ und } t[x/a'] \Rightarrow^{n_2} a \text{ und } n_1 + n_2 = n$$

Satz 5.2 *Für $u \in Comp(S, \mathfrak{A})$ und $a \in A$ gilt:*

$$u \Rightarrow^* a \curvearrowright u \Rightarrow_i^* a$$

Beweis:

analog wie für $\Rightarrow_o$. Einziger Unterschied ist der letzte Induktionsschritt:

(4) $u = F_i(u') \Rightarrow^{n+1} a$
$F_i(u') \Rightarrow^k F_i(u'') \Rightarrow t_i[x/u''] \Rightarrow^l a$ mit $k + l = n$
Wegen Lemma 5.3: $u'' \Rightarrow^{l_1} a'$ und $t_i[x/a'] \Rightarrow^{l_2} a$ mit $l_1 + l_2 = l$
$t = F_i(u') \Rightarrow_i^{k+l_1} F_i(a') \Rightarrow_i t_i[x/a'] \Rightarrow_i^{l_2} a$

q.e.d.

Korollar 5.2 *Für $n \in Comp(S, \mathfrak{A})$ und $a \in A$ gilt:*

$$u \Rightarrow^* a \curvearrowright\!\!\curvearrowright u \Rightarrow_o^* a \curvearrowright\!\!\curvearrowright u \Rightarrow_i^* a$$

5.4 Äquivalenz von Fixpunkt- und Reduktionssemantik

$S = (F_i(x) = t_i \mid 1 \leq i \leq r) \in dBS(\Omega, \Pi)$ $t_i \in T_r$
$\mathfrak{A} = \langle A; \alpha \rangle \in Int(\Omega, \Pi)$
$T_{(S,\mathfrak{A})} : [A - \!\!\rightarrow A]^r \longrightarrow [A - \!\!\rightarrow A]^r$

Approximationsfunktion:

$$\big(g_1^{(n)}, \ldots, g_r^{(n)}\big) := T_{(S,\mathfrak{A})}^n(\underbrace{g_\emptyset, \ldots, g_\emptyset}_{r-mal})$$

$$g_i^{(n+1)}(a) := [\![t_i]\!]_{\mathfrak{A}}\big(g_1^{(n)}, \ldots, g_r^{(n)}, a\big)$$

$$\big([\![S]\!]_{\mathfrak{A}}^{(1)}, \ldots, [\![S]\!]_{\mathfrak{A}}^{(r)}\big) := fix(T_{(S,\mathfrak{A})})$$

$$= \bigsqcup_{n=0}^{\infty} T_{(S,\mathfrak{A})}^n(\underbrace{g_\emptyset, \ldots, g_\emptyset}_{r-mal})$$

$$= \bigsqcup_{n=0}^{\infty} \big(g_1^{(n)}, \ldots, g_r^{(n)}\big)$$

$$[\![S]\!]_{\mathfrak{A}} := [\![S]\!]_{\mathfrak{A}}^{(1)}$$

Lemma 5.4 *Für $u, u' \in Comp(S, \mathfrak{A})$ gilt:*

$$u \Rightarrow u' \curvearrowright [\![u]\!]_{\mathfrak{A}} = [\![u']\!]_{\mathfrak{A}}$$

Korollar 5.3 *Die Reduktionssematik ist korrekt, d. h.*

$$F_1(a) \Rightarrow^* a' \curvearrowright [\![S]\!]_{\mathfrak{A}}(a) = a'$$

Lemma 5.5 *Wenn $[\![t]\!]_{\mathfrak{A}}\big([\![S]\!]_{\mathfrak{A}}^{(1)}, \ldots, [\![S]\!]_{\mathfrak{A}}^{(r)}, a\big) = a'$, dann existiert ein $n \in \mathbb{N}$ mit $[\![t]\!]_{\mathfrak{A}}\big(g_1^{(n)}, \ldots, g_r^{(n)}, a\big) = a'$*

Lemma 5.6 *Für alle $n \in \mathbb{N}$, für alle $t \in T_r$ und für alle $a, a' \in A$ gilt:*

$$[\![t]\!]_{\mathfrak{A}}\big(g_1^{(n)}, \ldots, g_r^{(n)}, a\big) = a' \curvearrowright t[x/a] \Rightarrow^* a'$$

Beweis:

durch Induktion über n

I.A.: $n = 0$: Induktion über t:

- $t = x$: $a = a' \curvearrowright a \Rightarrow^* a'$
- $t = f(t')$: $[\![f(t')]\!]_{\mathfrak{A}}(g_\emptyset, \ldots, g_\emptyset, a) = a'$, also
  $[\![t']\!]_{\mathfrak{A}}(g_\emptyset, \ldots, g_\emptyset, a) = a''$ und $f_{\mathfrak{A}}(a'') = a'$
  $\curvearrowright t[x/a] \Rightarrow f(t')[x/a] = f(t'[x/a]) \overset{I.V.}{\Rightarrow^*} f(a'') = a'$
- $t = \mathsf{if}\ p(x)\ \mathsf{then}\ t_1\ \mathsf{else}\ t_0\ \mathsf{fi}$: analog
- $t = F_i(t')$: trivial erfüllt, weil kein (a, a') existiert

I.S.: $n > 0$: Induktion über t:

- $t = x$, $t = f(t')$ und $t = \mathsf{if}\ p(x)\ \mathsf{then}\ t_1\ \mathsf{else}\ t_0\ \mathsf{fi}$: wie oben
- $t = F_i(t')$: $[\![F_i(t')]\!]_{\mathfrak{A}}\big(g_1^{(n+1)}, \ldots, g_r^{(n+1)}, a\big) = a'$
  $g_i^{(n+1)}\big([\![t']\!]_{\mathfrak{A}}\big(g_1^{(n+1)}, \ldots, g_r^{(n+1)}, a\big)\big) = a'$
  $[\![t_i]\!]_{\mathfrak{A}}\big(g_1^{(n)}, \ldots, g_r^{(n)}, [\![t']\!]_{\mathfrak{A}}\big(g_1^{(n+1)}, \ldots, g_r^{(n+1)}, a\big)\big) = a'$

  $\underbrace{t'[x/a] \Rightarrow^* a''}_{(1)}$ und $\underbrace{t_i[a/a''] \Rightarrow^* a'}_{(2)}$

  $t[x/a] = F_i(t')[x/a] = F_i(t'[x/a]) \overset{(1)}{\Rightarrow^*} F_i(a'') \Rightarrow t_i[x/a''] \overset{(2)}{\Rightarrow^*} a'$

q.e.d.

Satz 5.3 $Ian(\Omega, \Pi) \rightsquigarrow dBS(\Omega, \Pi)$

Beweis:

Die Fortsetzungssemantik von $S \in Ian(\Omega, \Pi)$ liefert ein äquivalentes $S' \in dBS(\Omega, \Pi)$.
$S : Q \longrightarrow (\Omega, Q) \cup (\Pi \times Q^2) \cup \{\mathsf{stop}\}$
$Q = \{q_1, \ldots, q_n\}$ mit Startzustand q_1
Konstruktion von S': $q_i \mapsto F_i$

$$\begin{aligned} S(q_i) = (f, q_j) &: F_i(x) = F_j(f(x)) \\ S(q_i) = (p, q_j, q_k) &: F_i(x) = \mathsf{if}\ p(x)\ \mathsf{then}\ F_j(x)\ \mathsf{else}\ F_k(x)\ \mathsf{fi} \\ S(q_i) = \mathsf{stop} &: F_i(x) = x \end{aligned}$$

Korrektheit dieser Transformation:
$\mathfrak{A} := \langle A; \alpha \rangle \in Int(\Omega, \Pi)$, $a \in A$
Dann gilt:

$$(q_1, a)(q_{i_1}, a_1) \cdots (q_{i_s}, a_s)(\mathsf{stop}, a_s)$$

ist genau dann eine Berechnung von $(S, \mathfrak{A})$ mit Eingabe a, wenn

$$F_1(a) \Rightarrow_i^2 F_{i_1}(a_1) \Rightarrow_i^2 \ldots \Rightarrow_i^2 F_{i_s}(a_s) \Rightarrow_i a_s$$

eine *cbv*-Berechnung von $(S', \mathfrak{A})$ mit Eingabe a.
Grund:

$$(q_i, a) \vdash (q_j, a') \curvearrowright\!\!\curvearrowright F_i(a) \Rightarrow_i^2 F_j(a')$$

q.e.d.

Diese Übersetzung liefert dBS-Schemata mit einer speziellen Termstruktur, in der nur Funktionsaufrufe an äußerster Stelle vorkommen. Solche Schemata heißen auch *endrekursiv*.

Satz 5.4 $dBS(\Omega,\Pi) \not\curvearrowright Ian(\Omega,\Pi)$

Beweis:

Für $S \in Ian(\Omega,\Pi) \cup dBS(\Omega,\Pi)$ definieren wir die **Wertsprache** $W(S) \subseteq \Omega^*$:

$$W(S) := \{[\![S]\!]_{\mathfrak{F}}(\varepsilon) \mid \mathfrak{F} \in Int(\Omega,\Pi) \text{ frei}\}$$

Offensichtlich gilt: $S \sim S' \curvearrowright W(S) = W(S')$[1]
Ferner gilt: $S \in Ian(\Omega,\Pi) \curvearrowright W(S)$ regulär.

Sei $S_0 \in dBS(\Omega,\Pi)$ definiert durch

$$F(x) = \mathsf{if}\ p(x)\ \mathsf{then}\ f(F(g(x)))\ \mathsf{else}\ x\ \mathsf{fi}$$

Die Wertsprache von S_0 ist dann: $W(S_0) = \{g^n f^n \mid n \in \mathbb{N}\}$, denn:

$F(\varepsilon) \Rightarrow^3 f(F(\varepsilon \circ g)) = f(F(g)) \Rightarrow_i f(f(F(g(g)))) \Rightarrow_i f(f(F(gg))) \Rightarrow_i \ldots \Rightarrow_i f^n(F(g^n)) \Rightarrow f^n(g^n) \Rightarrow g^n f^n$

$W(S_0)$ ist also kontextfrei.

q.e.d.

5.5 IANOV-Schemata mit Markenkeller

Eine iterative Berechnung rekursiver Prozeduren erfolgt im Compilerbau mit Hilfe eines *Prozedurkellers* mit mehreren Komponenten. Hier genügt ein Keller mit der *return address*, also ein Keller, auf dem die Rücksprungadressen (*return stack*) gespeichert werden.

5.5.1 Syntax

Wir definieren die Klasse der **Ianov-Schemata mit *return stack*** $Ian(\Omega,\Pi,RS)$ durch Modifikation der Klasse der IANOV-Schemata, indem wir die **Kellerbefehle**

- $(\mathsf{push}(q), p)$
- pop

hinzunehmen. Wir erhalten also $S \in Ian(\Omega,\Pi,RS)$ als eine Abbildung:

$$S : Q \longrightarrow (\Omega \cup \{\mathsf{push}(Q)\} \times Q) \cup (\Pi \times Q^2) \cup \{\mathsf{stop}, \mathsf{pop}\}$$

[1] Die Umkehrung gilt i. A. nicht (vgl. Schemasprachen)

5.5.2 Semantik

$\mathfrak{A} := \langle A; \alpha \rangle \in Int(\Omega, \Pi)$

$Conf(S, \mathfrak{A}) := (Q \cup \{\mathsf{stop}\}) \times Q^* \times A$

Die zweite Komponente einer Konfiguration ist der Keller mit Spitze links.

$$\begin{array}{rll} (q, w, a) \vdash (q', q''w, a) & \text{falls} & S(q) = (\mathsf{push}(q''), q') \\ (q, q'w, a) \vdash (q', w, a) & \text{falls} & S(q) = \mathsf{pop} \end{array}$$

Bei leeren Keller ist pop nicht definiert.

stop-Normierung

Es gibt zwei Möglichkeiten für den Abbruch einer Berechnung:

(1) Man trifft auf eine stop-Befehl
(2) Man trifft bei leerem Keller auf einen pop-Befehl

Als Konvention schließen wir stop-Befehle aus und brechen eine Berechnung nur für den Fall (2) ab:

$$[\![S]\!]_{\mathfrak{A}}(a) = a' \curvearrowright (q_1, \varepsilon, a) \vdash^* (q, \varepsilon, a') \text{ mit } S(q) = \mathsf{pop}$$

Unser Ziel in diesem Abschnitt ist es zu zeigen, dass IANOV-Schemata mit Markenkeller zu DE BAKKER-SCOTT-Schemata äquivalent sind, also $Ian(\Omega, \Pi, RS) \sim dBS(\Omega, \Pi)$. Dazu zeigen wir zunächst, dass zu jedem Ian-Schema ein äquivalentes dBS-Schema existiert; danach die umgekehrte Richtung.

Satz 5.5 $Ian(\Omega, \Pi, RS) \rightsquigarrow dBS(\Omega, \Pi)$

Beweis:

$S : Q \longrightarrow (\Omega \cup \{\mathsf{push}(Q)\} \times Q) \cup (\Pi \times Q^2) \cup \{\mathsf{stop}, \mathsf{pop}\}$

O.B.d.A. sei $Q = \{q_1, \ldots, q_{r+1}\}$ und $S(q_{r+1}) = \mathsf{pop}$, $S(q) \neq \mathsf{pop}\ \forall i \neq r+1$

Übersetzung in ein äquivalentes dBS-Schema S': $q_i \mapsto F_i$

$$\begin{array}{rcl} S(q_i) = (f, q_j) & : & F_i(x) = F_j(f(x)) \\ S(q_i) = (p, q_j, q_k) & : & F_i(x) = \mathsf{if}\ p(x)\ \mathsf{then}\ F_j(x)\ \mathsf{else}\ F_k(x)\ \mathsf{fi} \\ S(q_i) = (\mathsf{push}(q_j), q_k) & : & F_i(x) = F_j(F_k(x)) \\ S(q_{r+1}) = \mathsf{pop} & : & F_{r+1}(x) = x \end{array}$$

Zu zeigen: $[\![S]\!]_{\mathfrak{A}} = [\![S']\!]_{\mathfrak{A}}$

Dazu betten wir der Menge $Conf(S, \mathfrak{A})$ in die Menge $Comp(S', \mathfrak{A})$ ein, so dass sich die Berechnungen entsprechen:

$$\beta : Q \times Q^* \times A \longrightarrow Comp(S', \mathfrak{A})$$

$$\beta(q_i, \underbrace{q_{i_1} \ldots q_{i_s}}_{=:q_w}, a) = \underbrace{F_{i_s}(\ldots F_{i_1}}_{=:F_w}(F_i(a)) \ldots)$$

Dann gilt: β ist injektiv und für alle Konfigurationen γ, γ':

$(*)$ $\gamma \vdash \gamma' \curvearrowright \beta(\gamma) \Rightarrow_i^* \beta(\gamma')$

(1) $\gamma = (q_i, q_w, a)$ und $S(q_i) = (f, q_j)$
$\curvearrowright \gamma' = (q_j, q_w, f_{\mathfrak{A}}(a))$
$\beta(\gamma) = F_w(F_i(a)) \Rightarrow_i F_w(F_j(f(a))) \Rightarrow_i F_w(F_j(f_{\mathfrak{A}}(a))) = \beta(\gamma')$

(2) $\gamma = (q_i, q_w, a)$ und $S(q_i) = (p, q_{j_1}, q_{j_0})$
$\curvearrowright \gamma' = (q_{j_{p_{\mathfrak{A}}(a)}}, q_w, a)$
$\beta(\gamma) = F_w(F_i(a)) \Rightarrow_i F_w(\mathsf{if}\ p(a)\ \mathsf{then}\ F_{j_1}(a)\ \mathsf{else}\ F_{j_0}(a)\ \mathsf{fi}) \Rightarrow_i F_w(F_{j_{p_{\mathfrak{A}}}}(a)) = \beta(\gamma')$

(3) $\gamma = (q_i, q_w, a)$ und $S(q_i) = (\mathsf{push}(q_j), q_k)$
$\curvearrowright \gamma' = (q_k, q_j q_w, a)$
$\beta(\gamma) = F_w(F_i(a)) \Rightarrow_i F_w(F_j(F_k(a))) \Rightarrow_i F_{jw}(F_k(a)) = \beta(\gamma')$

(4) $\gamma = (q_i, q_w, a)$ und $S(q_i) = \mathsf{pop}$
$\curvearrowright \gamma' = (q_j, q_w, a)$
$\beta(\gamma) = F_w(F_j(F_i(a))) \Rightarrow_i F_w(F_j(a)) = \beta(\gamma')$

Aus $(*)$ folgt:
$[\![S]\!]_{\mathfrak{A}}(a) = a' \curvearrowright (q_1, \varepsilon, a) \vdash^* (q_{r+1}, \varepsilon, a') \curvearrowright F_1(a) \Rightarrow_i^* F_{r+1}(a') \Rightarrow_i a'$
$\curvearrowright [\![S]\!]_{\mathfrak{A}} \leq [\![S']\!]_{\mathfrak{A}}$

$[\![S]\!]_{\mathfrak{A}}(a) = \perp$
$\curvearrowright$ unendliche Berechnung von $\langle S, \mathfrak{A}\rangle$
$\overset{(*)}{\curvearrowright}$ unendliche Berechnung von $\langle S', \mathfrak{A}\rangle$
$\curvearrowright [\![S']\!]_{\mathfrak{A}}(a) = \perp$

q.e.d.

Satz 5.6 $dBS(\Omega, \Pi) \rightsquigarrow Ian(\Omega, \Pi, RS)$

Beweis:

Umformung durch Entschachtelung rechter Regelseiten in die Form eines übersetzten IANOV-Schemas mit *return stack*, d. h. die rechten Regelseiten sind von der Form

- $F(f(x))$,
- $\mathsf{if}\ p(x)\ \mathsf{then}\ F_j(x)\ \mathsf{else}\ F_k(x)\ \mathsf{fi}$,
- $F_j(F_k(x))$ oder
- x.

Sei $S = (F_i(x) = t_i \mid 1 \leq i \leq r) \in dBS(\Omega, \Pi)$

Wir transformieren S in die oben angegebene Normalform mit Hilfe folgender Regeln:

(1) Hinzufügen von $F_{r+1}(x) = x$

(2) Entschachteln von Verzweigungen:
Ersetze $F_i(x) = \mathsf{if}\ p(x)\ \mathsf{then}\ u_1\ \mathsf{else}\ u_2\ \mathsf{fi}$ mit $u_i \neq F_i(x)$ für ein $i = 1, 2$
durch $\begin{cases} \mathsf{if}\ p(x)\ \mathsf{then}\ G_1(x)\ \mathsf{else}\ G_2(x)\ \mathsf{fi} \\ G_1(x) = u_1 \\ G_2(x) = u_2 \end{cases}$

(3) Entschachteln von Funktionsapplikationen:
Ersetze $F_i(x) = f(u)$ durch $F_i(x) = F_{r+1}(f(u))$

(4) Entschachtelung von F-Applikationen:
Ersetze $F_i(x) = F_j(f(u))$ mit $u \neq x$
durch $\begin{cases} F_i(x) = G(u) \\ G(x) = F_j(f(x)) \end{cases}$
Ersetze $F_i(x) = F_j(F_k(u))$ mit $u \neq x$
durch $\begin{cases} F_i(x) = G(u) \\ G(x) = F_j(F_k(x)) \end{cases}$

Durch sukkzessive Anwendung der Regeln (2)–(4) erhält man ein Gleichungssystem mit rechten Regelseiten von obiger Form.

Ersetzt man zusätzlich

- $F_i(x) = x$ mit $1 \leq i \leq r$ durch $F_i(x) = F_{r+1}(F_{r+1}(x))$ und
- $F_i(x) = F_j(x)$ durch $F_i(x) = F_{r+1}(F_j(x))$,

so entfallen rechte Regelseiten der Form x und $F_j(x)$, d. h. es ergibt sich die Form eines übersetzten IANOV-Schemas mit Markenkeller. Die im Beweis von Satz 5.5 angegebene Transformation lässt sich umkehren:

$$\begin{aligned} F_i(x) = F_j(f(x)) &\quad : \quad S(q_i) = (f, q_j) \\ F_i(x) = F_j(F_k(x)) &\quad : \quad S(q_i) = (\mathsf{push}(q_j), q_k) \\ F_i(x) = \mathsf{if}\ p(x)\ \mathsf{then}\ F_j(x)\ \mathsf{else}\ F_k(x)\ \mathsf{fi} &\quad : \quad S(q_i) = (p, q_j, q_k) \\ F_{r+1}(x) = x &\quad : \quad S(q_{r+1}) = \mathsf{pop} \end{aligned}$$

q.e.d.

Folie 4.8

Korollar 5.4 $dBS(\Omega, \Pi) \sim Ian(\Omega, \Pi, RS)$

Kapitel 6

Entscheidbare Eigenschaften von de Bakker-Scott-Schemata

Das Ziel dieses Kapitels ist die Reduktion der Konvergenz, Divergenz und der Äquivalenz von Schemata auf entscheidbare Eigenschaften kontextfreier Sprachen.

6.1 Konvergenz und Divergenz

Lemma 6.1 *Für jede kontextfreie Grammatik $G \in CFG$ ist entscheidbar, ob*

(1) alle (Rechts-) Ableitungen endlich sind,

(2) alle (Rechts-) Ableitungen unendlich sind.

Beweis:

Idee:

(1) alle Ableitungen endlich $\curvearrowright\!\!\curvearrowright$ keines der erreichbaren Nichtterminale ist rekursiv

(2) alle Ableitungen unendlich $\curvearrowright\!\!\curvearrowright$ das Startsymbol ist nicht produktiv

q.e.d.

Satz 6.1 *Für $S \in dBS(\Omega, \Pi)$ ist entscheidbar, ob*

(1) S konvergiert,

(2) S divergiert.

Beweis:

Sei $\Pi = \{p_1, \ldots, p_n\}$ mit $n \geq 1$ und $S = (F_i(x) = t_i \mid 1 \leq i \leq r) \in dBS(\Omega, \Pi)$

O.B.d.A. sei jedes t_i verzweigungsfrei oder von der Form
$t_i = \mathsf{if}\ p(x)\ \mathsf{then}\ u_1\ \mathsf{else}\ u_2\ \mathsf{fi}$, wobei u_1 und u_2 verzweigungsfrei sind.

Zu S konstruieren wir $G_S \in CFG$ mit

$(*)$ { Jede Berechnung von S mit *cbv*-Reduktion unter einer freien Interpretation entspricht einer Rechtsableitung von G_S und umgekehrt.

Sei dazu $G_S := \langle N, \Omega, P, A\rangle$ mit $N := \{A\} \cup B \times \{F_1, \ldots, F_r\} \times B$

P wird so konstruiert, dass gilt:

$$(*)\begin{cases} [\beta_2, F_i, \beta_1] \Rightarrow_r^* w \in \Omega^* \\ \curvearrowright\!\curvearrowright \exists \mathfrak{F} = \langle \Omega^*; \varphi\rangle \in Int(\Omega, \Pi) \text{ frei} \\ \text{mit } F_i(\varepsilon) \Rightarrow_i^* w^R, \varphi(\Pi)(\varepsilon) = \beta_1, \varphi(\Pi)(w^R) = \beta_2 \end{cases}$$

Dazu:

(1) $A \to [\beta_2, F_1, \beta_1] \in P$ für alle $\beta_1, \beta_2 \in B$

(2) Zu $[\beta_2, F_i, \beta_1] \in N$ sei $\alpha_m(\cdots(\alpha_1(x))\cdots)$ mit $m \geq 0$ und $\alpha_j \in \Omega \cup \{F_1, \ldots, F_r\}$ der durch β_1 bestimmte Teilterm von t_i, d. h. t_i falls t_i verzweigungsfrei ist und u_1, falls $t_i =$ if $p_j(x)$ then u_1 else u_2 fi und $\beta_1 = (b_1, \ldots, b_{j-1}, 1, b_{j+1}, \ldots, b_n)$[1]
   - Falls $m \geq 1$:
     $[\beta_2, F_i, \beta_1] \to [\beta_2, \alpha_m, \gamma_{m-1}][\gamma_{m-1}, \alpha_{m-1}, \gamma_{m-2}] \cdots [\gamma_1, \alpha_1, \beta_1] \in P$
     für alle $\gamma_{m-1}, \ldots, \gamma_1 \in B$, wobei $[\ldots, \alpha, \ldots]$ ($\alpha \in \Omega$) durch α ersetzt wird
   - Falls $m = 0$ und $\beta_1 = \beta_2$:
     $[\beta_2, F_i, \beta_1] \to \varepsilon \in P$

(3) Transformiere G_S in eine reduzierte kontextfreie Grammatik. Dann erfüllt G_S offensichtlich $(*)$ und es gilt:
   S konvergiert (divergiert)
   $\curvearrowright\!\curvearrowright \forall \mathfrak{A} = \langle A; \alpha\rangle \in Int(\Omega, \Pi)$, $a \in A$: $[\![S]\!]_{\mathfrak{A}}(a)$ ist (un-)definiert
   $\curvearrowright\!\curvearrowright \forall \mathfrak{F} = \langle \Omega^*; \varphi\rangle \in Int(\Omega, \Pi)$ frei: $[\![S]\!]_{\mathfrak{F}}(\varepsilon)$ ist (un-)definiert
   $\curvearrowright\!\curvearrowright \forall \mathfrak{F} = \langle \Omega^*; \varphi\rangle \in Int(\Omega, \Pi)$ frei: $\exists(\nexists) w \in \Omega^*$: $F_1(\varepsilon) \Rightarrow_i^* w^R$
   $\overset{(*)}{\curvearrowright\!\curvearrowright}$ alle (Rechts-) Ableitungen von G_S sind (un-)endlich.

q.e.d.

Folie 5.1
Folie 5.2

Korollar 6.1 *Die Wertsprachen von* DE BAKKER-SCOTT-*Schemata sind genau die kontextfreien Sprachen.*

Beweis:

"$\supseteq$": *Übung 7, Aufgabe 20*
"$\subseteq$": $Val(S) = \{w^R \mid w \in L(G_S)\}$

q.e.d.

6.2 Exkurs: Deterministische Kontextfreie Sprachen

Das Ziel dieses Exkurses ist die Reduktion der Äquivalenz von DE BAKKER-SCOTT-Schemata auf die Äquivalenz *deterministischer Kellerautomaten.*

Definition 6.1 (Kellerautomat) *Ein* ***(nicht-deterministischer) Kellerautomat*** *ist ein Tupel der Form* $\mathcal{M} := \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0\rangle$, *wobei*

[1] entsprechend für u_2

- Q: *endliche Zustandsmenge*
- Σ: *Eingabealphabet*
- Γ: *Kelleralphabet*
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \longrightarrow \mathcal{P}_f(Q \times \Gamma^*)$ *Transformationsfunktion*
- $q_0 \in Q$: *Startzustand*
- $Z_0 \in \Gamma$: *Kellerstartsymbol*

Dabei gilt

- $\mathcal{M}$ *heißt* ***deterministisch****, falls für alle* $q \in Q$, $a \in \Sigma$ *und* $Z \in \Gamma$ *gilt:* $|\delta(q, a, Z)| + |\delta(q, \varepsilon, Z)| \leq 1$.
- $K = Q \times \Sigma^* \times \Gamma^*$ *ist die* ***Konfigurationsmenge****.*
- *Die* ***Übergangsrelation*** $\vdash \subseteq K \times K$ *ist gegeben durch:*
  $(q, xw, Z\gamma) \vdash (q', w, \alpha\gamma)$ *für alle* $q \in Q$, $x \in \Sigma \cup \{\varepsilon\}$, $Z \in \Gamma$, $\gamma \in \Gamma^*$ *und* $(q', \alpha) \in \delta(q, x, Z)$
- *Die durch* $\mathcal{M}$ *erkannte* ***Sprache*** *ist*
  $L(\mathcal{M}) = \{w \in \Sigma^* \mid \exists q \in Q : (q_0, w, Z_0) \vdash^* (q, \varepsilon, \varepsilon)\}$

Satz 6.2 *Eine Sprache ist genau dann kontextfrei, wenn sie von einem (nichtdeterminisischen) Kellerautomaten erkannt wird.*

Definition 6.2 (determinitisch-kontextfreie Sprache) *Eine Sprache* $L \in CFL$ *heißt* ***deterministisch*** *(*$L \in DCFL$*), wenn sie durch einen deterministischen Kellerautomaten erkannt wird.*

Satz 6.3 $DCFL \subset CFL$

Beweis:

Idee: Die Sprachen in $DCFL$ sind unter Komplement abgeschlossen.

q.e.d.

Satz 6.4 *Es ist entscheidbar, ob zwei deterministische Kellerautomaten* $\mathcal{M}_1$, $\mathcal{M}_2$ *äquivalent sind, d. h. ob* $L(\mathcal{M}_1) = L(\mathcal{M}_1)$ *gilt.*

Beweis:

Ursprünglich stammt der Beweis von G. Snizerguers: *Decidebility Results from Complete Formal Systems*, TCS 251, S. 1–166, 2000.

Der Beweis wurde vereinfacht durch C. Stirling: *Decidebility of DPDA Equivalence*, TCS 255, S. 1–31, 2001

Idee: Anwendnung von Methoden aus der Theorie nebenläufiger Systeme (Bisimulation)

q.e.d.

6.3 Das Äquivalenzproblem für DE BAKKER-SCOTT-Schemata

Das Ziel dieses Abschnittes ist die Konstruktion deterministischer Kellerautomaten $\mathcal{M}_1$ und $\mathcal{M}_2$ zu $S_1, S_2 \in dBS(\Omega, \Pi)$ mit

$$L(\mathcal{M}_1) = L(\mathcal{M}_1) \curvearrowright\!\!\!\!\curvearrowright S_1 \sim S_2$$

Satz 6.5 *Zu $S_1, S_2 \in dBS(\Omega, \Pi)$ ist entscheidbar, ob $S_1 \sim S_2$ gilt.*

Beweis:

Ähnlich wie im Beweis von Satz 6.1 gehen wir davon aus, dass beide Schemata S_1 und S_2 eine gewisse Normalform haben. Jede Gleichung habe also die Form: $F(x) =$ if $p(x)$ then $\alpha_1(x)$ else $\alpha_2(x)$ fi, wobei $\alpha_i \in \{\varepsilon, f, F_j, F_j f, F_j F_k\}$. Diese Normalform lässt sich durch die Einführung geeigneter Hilfsfunktionen erstellen (siehe Beispiel unten).

Einem Schema $S = (F_i(x) = t_i \mid 1 \leq i \leq r) \in dBS(\Omega, \Pi)$ lässt sich dann bezüglich einer Auswertbedingung der Prädikate $b_0 \in B$ wie folgt ein *deterministischer Kellerautomat* $\mathcal{M}_{S,b_0} := \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0 \rangle$ zuordnen:

- $Q := B$
- $\Sigma := B \times \Omega \times B$
- $\Gamma := \{F_1, \ldots, F_r\}$
- für jede Gleichung $F_i(x) =$ if $p_j(x)$ then $\alpha_1(x)$ else $\alpha_2(x)$ fi:
  - $\delta(b, bfb', F_i) = (b', \alpha')$, $\begin{cases} \text{falls } b = 1 \text{ und } \alpha_1^R = f\alpha' \\ \text{oder } b = 0 \text{ und } \alpha_2^R = f\alpha' \end{cases}$
  - $\delta(b, \varepsilon, F_i) = (b, \alpha')$, $\begin{cases} \text{falls } b = 1, \alpha_1 \in \{F_1, \ldots, F_r\}^* \text{ und } \alpha_1^R = \alpha' \\ \text{oder } b = 0, \alpha_2 \in \{F_1, \ldots, F_r\}^* \text{ und } \alpha_2^R = \alpha' \end{cases}$
- $q_0 := b_0$
- $Z_0 := F_1$

Da jedem F_i eindeutig eine Gleichung zugeordnet ist, ist $\mathcal{M}_{S,b_0}$ deterministisch. Ferner gilt für $S_1, S_2 \in dBS(\Omega, \Pi)$ und für alle freien Interpretationen $\mathfrak{F} = \langle \Omega^*; \varphi \rangle$ mit $\varphi(\Pi)(\varepsilon) = b_0$:

$$[\![S_1]\!]_{\mathfrak{F}} = [\![S_2]\!]_{\mathfrak{F}} \curvearrowright\!\!\!\!\curvearrowright L(\mathcal{M}_{S_1,b_0}) = L(\mathcal{M}_{S_2,b_0})$$

Begründung:

Einem *cbv*-Reduktionsschritt ($\alpha \in \{F_1, \ldots, F_r\}^*, w \in \Omega^*, f \in \Omega, i, j, k \in \{1, \ldots, r\}$) entspricht ein Transformationsschritt ($b = \varphi(\Pi)(w)$, $b' = \varphi(\Pi)(wf)$ und $v \in \Sigma^*$):

$$\begin{array}{llll} \alpha F_i w & \Rightarrow_i & \alpha w & : \quad (b, v, F_i\alpha) \vdash (b, v, \alpha) \\ \alpha F_i w & \Rightarrow_i & \alpha w f & : \quad (b, bfb'v, F_i\alpha) \vdash (b', v, \alpha) \\ \alpha F_i w & \Rightarrow_i & \alpha F_j w & : \quad (b, v, F_i\alpha) \vdash (b, v, F_j\alpha) \\ \alpha F_i w & \Rightarrow_i & \alpha F_j w f & : \quad (b, bfb'v, F_i\alpha) \vdash (b', v, F_j\alpha) \\ \alpha F_i w & \Rightarrow_i & \alpha F_j F_k w & : \quad (b, v, F_i\alpha) \vdash (b, v, F_k F_j\alpha) \end{array}$$

q.e.d.

Beispiel:

$F(x) = \text{if } p(x) \text{ then } f(F(g(x))) \text{ else } x \text{ fi}$

$$\rightsquigarrow \begin{cases} F(x) = \text{if } p(x) \text{ then } H(G(x)) \text{ else } x \text{ fi} \\ G(x) = \text{if } p(x) \text{ then } F(g(x)) \text{ else } F(g(x)) \text{ fi} \\ H(x) = \text{if } p(x) \text{ then } f(x) \text{ else } f(x) \text{ fi} \end{cases}$$

Hierbei sind G und H "neue" Symbole.

$\mathcal{M}_{S,b_0} := \langle Q, \Sigma, \Gamma, \delta, q_o, Z_0 \rangle$ mit

- $Q := \{0,1\}$
- $\Sigma := \{0,1\} \times \{f,g\} \times \{0,1\}$
- $\Gamma := \{F,G,H\}$
- $q_0 := b_0 \in \{1,0\}$
- $Z_0 := F$

δ:

q	x	Z	q'	α
0	ε	F	0	ε
1	ε	F	1	GH
0	$0g0$	G	0	F
0	$0g1$	G	1	F
1	$1g0$	G	0	F
1	$1g1$	G	1	F
0	$0f0$	H	0	ε
0	$0f1$	H	1	ε
1	$1f0$	H	0	ε
1	$1f1$	H	1	ε

Beispielrechnung ($q_0 = 1$):

$q \in Q$	$w \in \Sigma^*$	$\gamma \in \Gamma^*$
1	$1g11g00f11f1$	F
1	$1g11g00f11f1$	GH
1	$1g00f11f1$	FH
1	$1g00f11f1$	GHH
0	$0f11f1$	FHH
0	$0f11f1$	HH
1	$1f1$	H
1	ε	ε

$\rightsquigarrow$ akzeptiert

Kapitel 7

Luckham-Park-Paterson-Schemata

Bisher haben wir nur "einfache" Schemata kennengelernt, die nur einen *monolithischen Zustandsraum* und *monadische Grundoperationen* zuließen. In diesem Kapitel werden wir ein Schema definieren, welches einen erweiterten Zustandsraum besitzt. Dies könnte man erreichen, indem man Rekursion auf höheren Stufen zulassen würde, doch wir werden hier einen *strukturierten Zustandsraum* einführen. Dabei setzen wir folgende Annahmen voraus:

- Der Speicher besitzt endlich viele Komponenten gleichen Typs.
- Wir haben mehrere Eingabekomponenten und eine Ausgabekomponente.

Folie 6.1

7.1 Syntax von Luckham-Park-Paterson-Schemata

Wir definieren:

- **Operationssymbole**: $\Omega := \bigcup_{k\in\mathbb{N}} \Omega^{(k)}$
- **Prädikatssymbole**: $\Pi := \bigcup_{k\in\mathbb{N}} \Pi^{(k)}$
- **Kontrollmarken**: $Q := \{0, \ldots, r\} \quad (r \geq 1)$
- **Eingabevariablen**: $X := \{x_1, \ldots, x_n\} \quad (n \geq 0)$
- **Programmvariablen**: $Y := \{y_1, \ldots, y_p\} \quad (p \geq 0)$
- **Ausgabevariable**: z

Die **Menge der Befehle** $\mathfrak{C}$:

- **Startbefehle**: $(y_1, \ldots, y_p) \leftarrow (t_1, \ldots, t_p)$; goto 1
  wobei $t_i \in T_\Omega(X)$ (Terme über X)
- **Zuweisungen**: $(y_1, \ldots, y_p) \leftarrow (t_1, \ldots, t_p)$; goto q
  wobei $t_i \in T_\Omega(X)$ und $1 \leq q \leq r$
- **Verzweigungen**: if $p(t_1, \ldots, t_k)$ then goto q_1 else goto q_0 fi
  wobei $p \in \Pi^{(k)}$, $t_i \in T_\Omega(V)$, $q_0, q_1 \in \{1, \ldots, r\}$

- **Stopbefehle**: $z \leftarrow t$; stop
  wobei $t \in T_\Omega(V)$ (Terme über V)

$S \in LPP^{(n,p)}(\Omega,\Pi) :\curvearrowright\curvearrowright S : Q \longrightarrow \mathfrak{C}$ für ein passendes Q, wobei $S(q_0)$ der Startbefehl ist.

Wir definieren die Klasse aller Luckham-Park-Paterson-Schemata:

$$LPP(\Omega,\Pi) := \bigcup_{n,p\geq 0} LPP^{(n,p)}(\Omega,\Pi)$$

7.2 Semantik von Luckham-Park-Paterson-Schemata

Sei A eine Menge und α eine Abbildung mit

- $\alpha(f) : A^k \longrightarrow A$ für alle $f \in \Omega^{(k)}$
- $\alpha(p) : A^k \longrightarrow \{0,1\}$ für alle $p \in \Pi^{(k)}$

Dann ist $\mathfrak{A} := \langle A;\alpha\rangle \in Int(\Omega,\Pi)$ eine **Interpretation** über Ω und Π.
$\mathfrak{F} := \langle T_\Omega(A);\varphi\rangle \in Int(\Omega,\Pi)$ heißt **frei** $:\curvearrowright\curvearrowright \forall f \in \Omega^{(k)}, t_1,\ldots,t_k \in T_\Omega(X) :$ $\varphi(f)(t_1,\ldots,t_k) = f(t_1,\ldots,t_k)$.

Jede Variablenbelegung $\beta : V \longrightarrow A$ lässt sich eindeutig fortsetzen zu:

$$h_\beta : T_\Omega(V) \longrightarrow A$$

$$h_\beta\big(f(t_1,\ldots,t_k)\big) = \alpha(f)\big(h_\beta(t_1),\ldots,h_\beta(t_k)\big)$$

Für $S \in LPP^{(n,p)}(\Omega,\Pi)$ und $\mathfrak{A} \in Int(\Omega,\Pi)$ heißt $\langle S,\mathfrak{A}\rangle$ Luckham-Park-Paterson-**Programm**.

Wir definieren die Menge der **Konfigurationen**:

$$Conf(S,\mathfrak{A}) := Q \times A^n \times A^p$$

Eingabeabbildung:

$$\mathsf{input} : A^n \longrightarrow Conf(S,\mathfrak{A})$$

ist definiert durch den Startbefehl $(y_1,\ldots,y_p) \leftarrow (t_1,\ldots,t_p)$; goto 1
mit $\mathsf{input}(a_1,\ldots,a_n) := (1,a_1,\ldots,a_n,b_1,\ldots,b_p)$,
wobei $b_j := h_\beta(t_j)$ mit $\beta(x_i) = a_i$

Transformationsfunktion:

$$\vdash \subseteq Conf(S,\mathfrak{A})^2$$

mit $(q,a_1,\ldots,a_n,b_1,\ldots,b_p) \vdash (q',a_1,\ldots,a_n,b'_1,\ldots,b'_p)$, falls

(1) $S(q) = (y_1,\ldots,y_p) \leftarrow (t_1,\ldots,t_p)$; goto q' und $b_j := h_\beta(t_j)$ mit $\beta(x_i) = a_i$ und

(2) $S(q) =$ if $p(t_1,\ldots,t_k)$ then goto q_1 else goto q_0 fi, $b'_j = b_j$ und $q' = q_{\alpha(p)}\big(h_\beta(t_1),\ldots,h_\beta(t_k)\big)$

Ausgabeabbildung:

$$\mathsf{output} : Conf(S, \mathfrak{A}) \dashrightarrow A$$

mit $\mathsf{output}(q, a_1, \ldots, a_n, b_1, \ldots, b_p) = c \curvearrowright S(q) = z \leftarrow t; \mathsf{stop}$ und $c = h_\beta(t)$

Semantik von $\langle S, \mathfrak{A}\rangle$:

$$[\![S]\!]_{\mathfrak{A}} : A^n \dashrightarrow A$$
$$[\![S]\!]_{\mathfrak{A}}(a_1, \ldots, a_n) = c$$

$\curvearrowright \exists \gamma \in Conf(S, \mathfrak{A}) : \mathsf{input}(a_q, \ldots, a_n) \vdash^* \gamma$ und $\mathsf{output}(\gamma) = c$

Die Äquivalenz zweier Luckham-Park-Paterson-Schemata sei definert wie bisher, wobei man beachten muss, dass äquivalente Schemata gleichviele Eingabevariablen haben müssen, nicht aber gleichviele Programmvariablen.

Beachte:

Monadische Luckham-Park-Paterson-*Schemata* ($\Omega = \Omega^{(1)}, \Pi = \Pi^{(1)}, n = 1$) sind nicht äquivalent zu Ianov-Schemata:

- Zurücksetzen von Variablen (z. B.: $y_1 \leftarrow x_1$)
- erweiterte Tests, Hilfsspeicher (z. B.: if $p(f(x)) \ldots$)

7.3 Reduktion auf freie Interpreationen

Eine Interpretation $\mathfrak{A} = \langle A; \alpha\rangle$ und ein $\overline{a} = (a_1, \ldots, a_n) \in A^n$ bestimmen die freie Interpretation $\mathfrak{F}_{(\mathfrak{A},\overline{a})} := \langle T_\Omega(X); \varphi\rangle$ mit

$$\varphi(p)(t_1, \ldots, t_k) = \alpha(p)\big(h_\beta(t_1), \ldots, h_\beta(t_k)\big) \text{ für alle } p \in \Pi^{(k)}, \beta(x_i) = a_i$$

Satz 7.1 *Für $S \in LPP^{(n,p)}(\Omega, \Pi)$, $\mathfrak{A} \in Int(\Omega, \Pi)$ und $\overline{a} \in A^n$ gilt:*

$$[\![S]\!]_{\mathfrak{A}}(\overline{a}) = h_\beta\big([\![S]\!]_{\mathfrak{F}_{(\mathfrak{A},\overline{a})}}(x_1, \ldots, x_n)\big) \text{ mit } \beta(x_i) = a_i$$

Beweis:

Dieser Satz ist eine direkte Verallgemeinerung des Satzes 2.1 für Ianov-Schemata.

q.e.d.

Korollar 7.1 *Für $S_1, S_2 \in LPP^{(n,p)}(\Omega, \Pi)$ gilt $S_1 \sim S_2 \curvearrowright [\![S_1]\!]_{\mathfrak{F}} = [\![S_2]\!]_{\mathfrak{F}}$ für jede freie Interpreation $\mathfrak{F}$.*

7.4 Entscheidbare Eigenschaften von LUCKHAM-PARK-PATERSON-Schemata

Das Ziel dieses letzten Abschnitts ist es, die *Unentscheidbarkeit* der Konvergenz, Divergenz und Äquivalenz für LUCKHAM-PARK-PATERSON-Schemata zu zeigen. Unser Hilfsmittel dabei ist die Simulation von TURING-Maschinen durch *endliche 2-Band-Automaten*. Da die Unentscheidbarkeit schon für eine einfache Teilklasse der LUCKHAM-PARK-PATERSON-Schemata gilt, wollen wir uns im Folgenden auf diese konzentrieren.

Die Menge der **einfachen** LUCKHAM-PARK-PATERSON**-Schemata** $\mathfrak{S} \subseteq LPP^{(n,p)}(\Omega, \Pi)$ ist definiert durch:

- $\Omega := \Omega^{(1)} := \{f\}$
- $\Pi := \Pi^{(1)} := \{p\}$
- $n = 1$: eine Eingabevariable x
- $k = 2$: zwei Programmvariablen y_1, y_2
- **Befehlsmakros**:
  - **Startbefehl**: $(y_1, y_2) \leftarrow (x, x)$; goto 1
  - **Zuweisung & Testen**: $y_j \leftarrow f(y_j)$; if $p(y_j)$ then goto q_1 else goto q_0 fi wobei $1 \leq j \leq 2$
  - **Schleife**: loop
  - **Stopbefehl**: $z \leftarrow x$; stop (d. h. Ausgabewert = Eingabewert)

Folie 6.2

Um die Unentscheidbarkeit obiger Eigenschaften für diese einfachen LUCKHAM-PARK-PATERSON-Schemata zu zeigen, genügt es, freie Interpretationen

$$\mathfrak{F} := \langle T_\Omega(X); \varphi \rangle \in Int(\Omega, \Pi)$$

zu betrachten, wobei wegen $\Omega = \{f\}$ gilt:

$$T_\Omega(X) = \{x, f(x), f(f(x)), \ldots\} = \{f^n(x) \mid n \in \mathbb{N}\}$$

Somit ist φ bestimmt durch

$$\varphi(p)(x), \quad \varphi(p)(f(x)), \quad \varphi(p)(f(f(x))), \quad \ldots$$

d. h. eine *Folge von Wahrheitswerten*, die wir durch

$$w \in \{0,1\}^\infty \text{ mit } w_i := w(i) := \varphi(p)(f^i(x))$$

beschreiben können. Eine solche Folge w bestimmt also eindeutig eine freie Interpretation $\mathfrak{F}_w$.

Automateninterpretation:

Diese ist gegeben durch: $\mathfrak{T} = \langle \{0,1\}^\infty; \alpha \rangle \in Int(\Omega, \Pi)$ mit

- $\alpha(f) =$ tail
- $\alpha(p) =$ head

wobei

- $\mathsf{tail} : w_0 w_1 w_2 \ldots \mapsto w_1 w_2 \ldots$
- $\mathsf{head} : w_0 w_1 w_2 \ldots \mapsto w_0$

Für $S \in \mathfrak{S}$ heißt dann $\mathcal{A} := \langle S, \mathfrak{T} \rangle$ ein **endlicher 2-Band-Automat** ($\mathcal{A} \in EZA$)

Folie 6.3

Arbeitsweise eines 2-Band-Automaten:

Start:	Eingabefolge $w \in \{0,1\}^\infty$ wird auf die beiden Arbeitsbänder y_1, y_2 kopiert.
Arbeitsschritt:	Auf einem der Arbeitsbänder wird das erste Symbol gelöscht und dann das nächste Symbol gelesen.
Stop:	Die Eingabefolge wird ausgegeben.

Es gilt also: $[\![S]\!]_{\mathfrak{T}} : \{0,1\}^\infty \dashrightarrow \{0,1\}^\infty$ (partielle Identität)

Zur Simulation von Berechnungen einer TURING-Maschine ordnen wir einem $\mathcal{A} := \langle S, \mathfrak{T} \rangle \in EZA$ eine Zerlegung von $\{0,1\}^\infty$ in drei Teilmengen zu:

- $Acc(\mathcal{A}) := \{w \in \{0,1\}^\infty \mid [\![S]\!]_{\mathfrak{T}}(w) = w\}$
  (d. h. bei Eingabe von w wird der stop-Befehl erreicht)
- $Rej(\mathcal{A}) := \{w \in \{0,1\}^\infty \mid [\![S]\!]_{\mathfrak{T}}(w) = \perp\}$
  (d. h. bei Eingabe von w wird ein loop-Befehl erreicht)
- $Div(\mathcal{A}) := \{w \in \{0,1\}^\infty \mid [\![S]\!]_{\mathfrak{T}}(w) = \perp\}$
  (d. h. bei Eingabe von w erfolgt eine unendliche Berechnung)

Zusammenhang zwischen freien und Automateninterpretation

$S \in \mathfrak{S}$ verhält sich bei der *freien Interpretation* $\mathfrak{F}_w$ mit Eingabe x genauso wie bei der *Automateninterpretation* $\mathfrak{T}$ mit Eingabe w.

Definition 7.1 *Für $w \in \{0,1\}^\infty$ sei $\rho_w : T_\Omega(X) \longrightarrow \{0,1\}^\infty$ gegeben durch $\rho_{w_0 w_1 w_2 \ldots}\big(f^n(x)\big) := w_n w_{n+1} w_{n+2} \ldots$*

Lemma 7.1 *Für $S \in \mathfrak{S}$ und $w \in \{0,1\}^\infty$ gilt:*

$$(q, x, t_1, t_2) \vdash_{\mathfrak{F}_w} (q', x, t_1', t_2') \curvearrowright\!\!\curvearrowright (q, x, \rho_w(t_1), \rho_w(t_2)) \vdash_{\mathfrak{T}} (q', x, \rho_w(t_1'), \rho_w(t_2'))$$

Beweis:

Fallunterscheidung bzgl. des ausgeführten Befehls.

Sei $S(q) = y_1 \leftarrow f(y_1)$; if $p(y_1)$ then q' else q_0 fi
$\curvearrowright$ für $t_1 = f^m(x)$: $t_1' = f^{m+1}(x)$, $\varphi(p)\big(f^{m+1}(x)\big) = 1 = w_{m+1}$, $t_2' = t_2$

Berechnung bzgl. $\mathfrak{T}$:

$\rho(t_1) = w_m w_{m+1} \ldots$

Bei Ausführung von $S(q) = y_1 \leftarrow \mathsf{tail}(y_1)$; if $\mathsf{head}(y_1)$ then q' else q_0 fi
wird zunächst w_m gelöscht und dann w_{m+1} getestet. Daraus folgt die Behauptung.

q.e.d.

Korollar 7.2 *$\widetilde{S} \in \mathfrak{S}$ entstehe aus $S \in \mathfrak{S}$ durch Ersetzen der* loop*-Befehle durch* stop*-Befehle. Dann gilt:*

(1) S divergiert $\curvearrowright$ $Acc(S, \mathfrak{T}) = \emptyset$

(2) $\widetilde{S}$ terminiert nicht für ein $\mathfrak{A} \in Int(\Omega, \Pi)$ und $a \in A$ $\curvearrowright$ $Div(S, \mathfrak{T}) \neq \emptyset$

Beweis:

(1) S divergiert

$\curvearrowright \forall \mathfrak{A} = \langle A; \alpha \rangle \in Int(\Omega, \Pi)$: $\forall a \in A$: $\forall k \in \mathbb{N}$: $\exists q \in Q$: $\exists b_1, b_2 \in A$:
$(1, a, a, a) \vdash^k_{\mathfrak{A}} (q, a, b_1, b_2)$
$\curvearrowright \forall w \in \{0,1\}^\infty$: $\forall k \in \mathbb{N}$: $\exists q \in Q$: $\exists t_1, t_2 \in T_\Omega(X)$:
$(q, x, x, x) \vdash^k_{\mathfrak{F}_w} (q, x, t_1, t_2)$
$\overset{L7.1}{\curvearrowright} \forall w \in \{0,1\}^\infty$: $\forall k \in \mathbb{N}$: $\exists q \in Q$: $\exists t_1, t_2 \in T_\Omega(X)$:
$(1, w, w, w) \vdash^k_{\mathfrak{T}} (q, w, \rho_w(t_1), \rho_w(t_2))$
$\curvearrowright \forall w \in \{0,1\}^\infty$: $w \notin Acc(S, \mathfrak{T})$
$\curvearrowright Acc(S, \mathfrak{T}) = \emptyset$

(2) $\widetilde{S}$ terminiert nicht für ein $\mathfrak{A} \in Int(\Omega, \Pi)$ und $a \in A$

$\curvearrowright \exists w \in \{0,1\}^\infty$: $\forall k \in \mathbb{N}$: $\exists q \in Q$: $\exists t_1, t_2 \in T_\Omega(X)$:
$(1, x, x, x) \vdash^k_{\mathfrak{F}_w} (q, x, t_1, t_2)$
$\overset{L7.1}{\curvearrowright} \exists w \in \{0,1\}^\infty$: $\forall k \in \mathbb{N}$: $\exists q \in Q$: $\exists t_1, t_2 \in T_\Omega(X)$:
$(1, w, w, w) \vdash^k_{\mathfrak{T}} (q, w, \rho_w(t_1), \rho_w(t_2))$
$\curvearrowright \exists w \in \{0,1\}^\infty$: $w \in Div(\widetilde{S}, \mathfrak{T}) = Div(S, \mathfrak{T})$

q.e.d.

Satz 7.2 *Es ist nicht entscheidbar, ob für ein gegebenes $S \in \mathfrak{S}$ gilt:*

(1) $Acc(S, \mathfrak{T}) = \emptyset$

(2) $Div(S, \mathfrak{T}) \neq \emptyset$

Beweis:

Idee:

Simulation einer TURING-Maschinen-Berechnung auf einem endlichen 2-Band-Automaten:

Zu einer deterministischen TURING-Maschine M mit Eingabe v lässt sich ein $\mathcal{A}_{M,v} \in EZA$ konstruieren, der bei Eingabe von $w \in \{0,1\}^\infty$ prüft, ob w die Berechnung von M auf v beschreibt:

- $w \in Acc(\mathcal{A}_{M,v})$
  $\curvearrowright$ w beschreibt eine abbrechende Berechnung von M auf v.
- $w \in Div(\mathcal{A}_{M,v})$
  $\curvearrowright$ w beschreibt eine nicht-abbrechende Berechnung von M auf v.
- $w \in Rej(\mathcal{A}_{M,v})$
  $\curvearrowright$ w beschreibt keine Berechnung von M auf v.

Konstruktion von $\mathcal{A}_{M,v}$:

Sei $w := w_0 w_1 w_2 \ldots$ mit $w_i := u_1 q c u_2$ ist die Konfiguration von M nach dem i-ten Berechnungsschritt. Überprüfe, ob w_{i+1} die Folgekonfiguration von w_i ist (auf den Bändern y_1 und y_2). *[siehe* MANNA*]*

$Acc(\mathcal{A}_{M,v}) = \emptyset$
$\curvearrowleft\curvearrowright$ M hält nicht auf Eingabe v
$\curvearrowleft\curvearrowright$ $Div(\mathcal{A}_{M,v}) \neq \emptyset$

Daraus folgt, dass die Entscheidbarkeit von (1) oder (2) die Entscheidbarkeit des *Halteproblems* für TURING-Maschinen impliziert.

q.e.d.

Korollar 7.3 *Es ist nicht entscheidbar, ob für $S_1, S_2 \in \mathfrak{S}$ gilt:*

(1) S_1 *konvergiert*

(2) S_1 *divergiert*

(3) $S_1 \sim S_2$

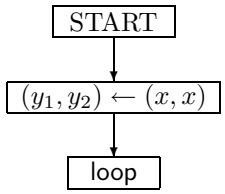
Beweis:

(1) Annahme: Konvergenz von S_1 sei entscheidbar.
$\curvearrowleft\curvearrowright$ Es ist entscheidbar, ob $\underline{S_1}$ nicht konvergiert.
$\curvearrowleft\curvearrowright$ Es ist entscheidbar, ob $\widetilde{S_1}$ nicht konvergiert.
$\curvearrowleft\curvearrowright$ Es ist entscheidbar, ob $Div(S_1, \mathfrak{T}) \neq \emptyset$

(2) S_1 divergiert $\curvearrowleft\curvearrowright$ $Acc(S_1, \mathfrak{T}) = \emptyset$

(3) Wäre die Äquivalenz entscheidbar, so wäre auch die Divergenz entscheidbar. Z. B. ist jedes divergente LPP-Schema zu folgendem LPP-Schema äquivalent:

S_2:



q.e.d.

Index