Teil 2: ASM Semantik von Programmen

- Gurevich Abstract State Machines
- ASM Semantik von Programmen
- Zusammenhang zur natürlichen Semantik

Idee: Ein Cursor wandert in kleinen Schritten durch den abstrakten Syntaxbaum des Programms und führt dabei Aktionen aus.

- Der Cursor ist im Modus *up* (▲) oder im Modus *down* (▼).
- Wegen der rekursiven Prozeduren wird ein *Programmstack* benötigt.
- Variablen haben *Adressen* (wegen der Variablenparameter).
- Die Werte, die an den Adressen stehen, werden in einem *Speicher* abgespeichert.

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Ursprung der ASMs

Church-Turing These (1936):

Jeder Algorithmus kann auf einer Turing-Maschine berechnet werden.

ASM These (Gurevich 1985):

Jedes Computersystem kann auf seiner natürlichen Abstraktionsstufe *Schritt für Schritt* durch eine *Abstract State Maschine* simuliert werden.

ASM = Abstract State + Abstract Machine

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Was sind ASMs?

Zustände = (endliche oder unendliche) algebraische Strukturen

Abstrakter Speicher: $f(x + f(y))$ entspricht $\texttt{mem}[x + \texttt{mem}[y]]$

f ist eine *dynamische* Funktion, '+' ist eine *statische* Funktion.

Transitionsregeln:

parallele Ausführung
- **if** *cond* **then** $\underbrace{f(t_1, \ldots, t_n) := s}_{\text{Update}}$
- $\vdots$
- **if** ... **then** ...

Lauf einer Maschine: $\mathfrak{A}_0 \Longrightarrow \mathfrak{A}_1 \Longrightarrow \cdots \Longrightarrow \mathfrak{A}_n \Longrightarrow \cdots$

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Universen für die ASM Semantik von Programmen

Pos	[Positionen im Syntaxbaum]
$\mathit{Adr} = \mathbb{N}$	[Adressen]
$\mathit{Val} = \mathbb{Z}$	[Werte]
$\mathit{Store} = \mathit{Adr} \hookrightarrow \mathit{Val}$	[Speicher]
$\mathit{Env} = \mathit{Var} \hookrightarrow \mathit{Adr}$	[Lokale Umgebungen]
$\mathit{Frame} = \mathit{Pos} \times \mathit{Env}$	[Frames auf dem Stack]

Bemerkung: Falls $store \in \mathit{Store}$ und $env \in \mathit{Env}$ mit $\mathrm{ran}(env) \subseteq \mathrm{dom}(store)$, dann ist $store \circ env \in \mathit{State}$, wobei für $x \in \mathrm{dom}(env)$

$$(store \circ env)(x) = store(env(x)).$$

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Universen (Fortsetzung)

- In der lokalen Umgebung (*Environment*) werden die Adressen der lokalen Variablen festgehalten.
- Verschiedene Variablen haben verschiedene Adressen (kein *Aliasing*)
- Im Speicher (*Store*) werden die Werte, die an den Adressen abgespeichert sind, festgehalten.
- Ein Frame besteht aus der Position des Cursors im Syntaxbaum und der lokalen Umgebung.
- Bei jedem Prozeduraufruf wird ein neues Frame erzeugt.
- Bei Beendigung der Prozedur kehrt man zum alten Frame zurück.

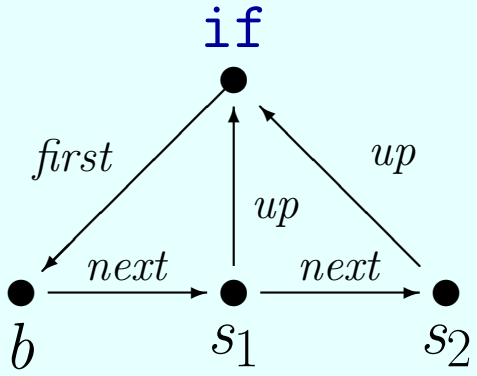
Zusätzliche Bedingung für Programme:

Im Bereich einer lokalen Variablen x (oder eines formalen Parameters) wird die Variable x nicht nochmals deklariert. Sonst muss die Variable umbenannt werden.

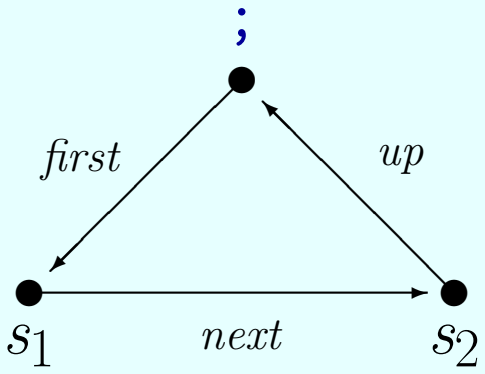
[Dies wird z.B. bei Java oder C# verlangt.]

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Syntaxbäume



`if` b `then` s_1 `else` s_2 `end`



s_1 `;` s_2

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Der dynamische Zustand eines Programms

Zustandsvariablen:

pos: *Pos*	[Position des Cursors]
mode: {*up*, *down*}	[Mode des Cursors]
env: *Env*	[Lokale Umgebung]
frames: *List*(*Frame*)	[Programmstack]
store: *Store*	[Speicher]

Anfangswerte:

$pos = \texttt{main()}$

$mode = down$

$env = \{\ \}$

$frames = [\ ]$

$store = \{\ \}$

Schreibweise:

Der Update $env(x) := \alpha$ steht für $env := env[x \mapsto \alpha]$

Anfangszustand: ▼`main()`

Endzustand: ▲`main()`

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Transitionsregeln der ASM-Semantik

Leere Anweisung:

$$\blacktriangledown \texttt{skip} \longrightarrow \blacktriangle \texttt{skip}$$

Zuweisung:

$$\blacktriangledown (x \texttt{ := } e) \longrightarrow \blacktriangle (x \texttt{ := } e)$$
$$\mathit{store}(\mathit{env}(x)) := \mathcal{A}[\![e]\!](\mathit{store} \circ \mathit{env})$$

Sequenzielle Komposition:

$$\blacktriangledown (s_1 \texttt{ ; } s_2) \longrightarrow \blacktriangledown s_1$$
$$(\blacktriangle s_1 \texttt{ ; } s_2) \longrightarrow \blacktriangledown s_2$$
$$(s_1 \texttt{ ; } \blacktriangle s_2) \longrightarrow \blacktriangle (s_1 \texttt{ ; } s_2)$$

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Transitionen (If-Then-Else, While)

If-Then-Else-Anweisung:

$$\blacktriangledown \texttt{if } b \texttt{ then } s_1 \texttt{ else } s_2 \texttt{ end} \longrightarrow \textbf{if } \mathcal{B}[\![b]\!](\mathit{store} \circ \mathit{env}) = 1 \textbf{ then } \blacktriangledown s_1 \textbf{ else } \blacktriangledown s_2$$

$$\texttt{if } b \texttt{ then } \blacktriangle s_1 \texttt{ else } s_2 \texttt{ end} \longrightarrow \blacktriangle \texttt{if } b \texttt{ then } s_1 \texttt{ else } s_2 \texttt{ end}$$

$$\texttt{if } b \texttt{ then } s_1 \texttt{ else } \blacktriangle s_2 \texttt{ end} \longrightarrow \blacktriangle \texttt{if } b \texttt{ then } s_1 \texttt{ else } s_2 \texttt{ end}$$

While-Anweisung:

$$\blacktriangledown \texttt{while } b \texttt{ do } s \texttt{ end} \longrightarrow \textbf{if } \mathcal{B}[\![b]\!](\mathit{store} \circ \mathit{env}) = 1 \textbf{ then } \blacktriangledown s \textbf{ else } \blacktriangle \texttt{while } b \texttt{ do } s \texttt{ end}$$

$$\texttt{while } b \texttt{ do } \blacktriangle s \texttt{ end} \longrightarrow \blacktriangledown \texttt{while } b \texttt{ do } s \texttt{ end}$$

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Transitionen (lokale Variablendeklaration, Prozeduraufruf)

Lokale Variablendeklaration:

$$\blacktriangledown\texttt{var } x \texttt{ := } e \texttt{ in } s \texttt{ end} \longrightarrow \blacktriangledown s$$
$$\quad \textbf{let } \alpha = \textbf{new}(\mathit{Adr}) \textbf{ in}$$
$$\quad\quad \mathit{store}(\alpha) := \mathcal{A}[\![e]\!](\mathit{store} \circ \mathit{env})$$
$$\quad\quad \mathit{env}(x) := \alpha$$

$$\texttt{var } x \texttt{ := } e \texttt{ in } \blacktriangle s \texttt{ end} \longrightarrow \blacktriangle\texttt{var } x \texttt{ := } e \texttt{ in } s \texttt{ end}$$

Prozeduraufruf:

$$\blacktriangledown p\texttt{(}\vec{e}\texttt{;}\vec{z}\texttt{)} \longrightarrow \texttt{procedure } p\texttt{(}\vec{x}\texttt{;}\vec{y}\texttt{) begin } \blacktriangledown s \texttt{ end}$$
$$\quad \textbf{let } \vec{v} = \mathcal{A}[\![\vec{e}]\!](\mathit{store} \circ \mathit{env}) \textbf{ and } \vec{\alpha} = \textbf{new}(\mathit{Adr}) \textbf{ in}$$
$$\quad\quad \mathit{frames} := \mathit{push}(\mathit{frames}, \langle \mathit{pos}, \mathit{env} \rangle)$$
$$\quad\quad \mathit{store}(\vec{\alpha}) := \vec{v}$$
$$\quad\quad \mathit{env} := \{\vec{x} \mapsto \vec{\alpha},\ \vec{y} \mapsto \mathit{env}(\vec{z})\}$$

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Transitionen (Return)

Rückkehr von einem Prozeduraufruf:

$\texttt{procedure}\ p(\vec{x};\vec{y})\ \texttt{begin}\ \blacktriangle s\ \texttt{end} \longrightarrow$

let $\langle oldpos, oldenv\rangle = top(\mathit{frames})$ **in**

$\mathit{frames} := pop(\mathit{frames})$

$\mathit{pos} := oldpos$

$\mathit{env} := oldenv$

$\mathit{mode} := \mathit{up}$

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Erklärungen

▼`var` x `:=` e `in` s `end`

- Es wird eine neue Adresse α für die lokale Variable x alloziert.
- Im Speicher wird an der Adresse α der Wert des Ausdrucks e abgespeichert.
- In der lokalen Umgebung wird die neue Adresse von x festgehalten.

▼$p(\vec{e};\vec{z})$ mit der Deklaration `procedure` $p(\vec{x};\vec{y})$ `begin` s `end`

- Das aktuelle Frame wird auf den Stack gelegt.
- Für die Werteparameter $\vec{x}$ werden neue Addressen $\vec{\alpha}$ alloziert.
- Die Werte der Argumente $\vec{e}$ werden an die Adressen $\vec{\alpha}$ kopiert.
- Die Variablenparameter $\vec{y}$ erhalten die Adressen von $\vec{z}$.
- Der Rumpf der Prozedur wird in der neuen Umgebung ausgeführt.

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Erklärungen (Fortsetzung)

$\texttt{procedure}\ p(\vec{x};\vec{y})\ \texttt{begin}\ {}^{\blacktriangle}s\ \texttt{end}$

- Bei der Rückkehr von einem Prozeduraufruf wird das oberste Frame vom Stack zum aktuellen Frame.
- Die Ausführung fährt bei der alten Position in der alten Umgebung weiter.
- Der Modus des Cursors wird geändert von ${}^{\blacktriangledown}p(\vec{e};\vec{z})$ zu ${}^{\blacktriangle}p(\vec{e};\vec{z})$.

Die Variablen in *env* haben verschiedene Adressen.

Zur Laufzeit eines Programms gilt:

$$x, y \in \mathrm{dom}(\mathit{env}),\ x \neq y \Longrightarrow \mathit{env}(x) \neq \mathit{env}(y)$$

Grund: Bei einem Prozeduraufruf $p(\vec{e};\vec{z})$ muss $\vec{z}$ aus verschiedenen Variablen bestehen.

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Natürlichen Semantik $\Longrightarrow$ ASM-Semantik

Theorem. Falls in der natürlichen Semantik $\sigma \multimap\!\![s]\!\!\rightarrowtail \sigma'$ herleitbar ist und $\mathfrak{A}$ ein ASM-Zustand ist mit

- $\mathfrak{A}(\mathit{pos}) = {}^{\blacktriangledown}s$
- $\sigma \subseteq \mathfrak{A}(\mathit{store}) \circ \mathfrak{A}(\mathit{env})$

dann erreicht die ASM-Semantik nach endlich vielen Schritten einen Zustand $\mathfrak{B}$ so, dass

- $\mathfrak{B}(\mathit{pos}) = {}^{\blacktriangle}s$
- $\sigma' \subseteq \mathfrak{B}(\mathit{store}) \circ \mathfrak{B}(\mathit{env})$
- $\mathfrak{A}(\mathit{env}) \subseteq \mathfrak{B}(\mathit{env})$
- $\mathfrak{A}(\mathit{frames}) = \mathfrak{B}(\mathit{frames})$

und für alle Adressen $\alpha \in \mathrm{dom}(\mathfrak{A}(\mathit{store})) \setminus \mathrm{ran}(\mathfrak{A}(\mathit{env}))$ gilt

$$\mathfrak{B}(\mathit{store})(\alpha) = \mathfrak{A}(\mathit{store})(\alpha) \text{ und } \alpha \notin \mathrm{ran}(\mathfrak{B}(\mathit{env}))$$

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Beweis (leere Anweisung) *

Mit $\sigma \subseteq \mathit{store} \circ \mathit{env}$ drückt man aus, dass für alle $x \in \mathrm{dom}(\sigma)$ gilt

$$\sigma(x) = \mathit{store}(\mathit{env}(x)).$$

Die Behauptung wird mit Induktion nach der Länge der Herleitung von $\sigma \multimap\!\![s]\!\!\rightarrowtail \sigma'$ bewiesen.

Fall 1: $\dfrac{}{\sigma \multimap\!\![\mathtt{skip}]\!\!\rightarrowtail \sigma}$

Falls $\mathfrak{A}(\mathit{pos}) = {}^{\blacktriangledown}\mathtt{skip}$, dann gilt im Nachfolgezustand $\mathfrak{B}$ von $\mathfrak{A}$, dass $\mathfrak{B}(\mathit{pos}) = {}^{\blacktriangle}\mathtt{skip}$.

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Beweis (Zuweisung) *

Fall 2: $\dfrac{}{\sigma \rightarrow[x := e]\rightarrow \sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma]}$

Falls $\mathfrak{A}(\mathit{pos}) = {}^{\blacktriangledown}(x := e)$, dann gilt im Nachfolgezustand $\mathfrak{B}$, dass

- $\mathfrak{B}(\mathit{pos}) = {}^{\blacktriangle}(x := e)$
- $\mathfrak{B}(\mathit{store})(\mathfrak{B}(\mathit{env})(x)) = \mathcal{A}[\![e]\!](\mathfrak{A}(\mathit{store}) \circ \mathfrak{A}(\mathit{env})) = v$

Falls $\sigma \subseteq \mathfrak{A}(\mathit{store}) \circ \mathfrak{A}(\mathit{env})$, dann ist $\mathcal{A}[\![e]\!]\sigma = v$.

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Beweis (Sequenzielle Komposition) *

Fall 3:

$$\frac{\sigma \,-[s_1]\rightarrow\, \sigma' \qquad \sigma' \,-[s_2]\rightarrow\, \sigma''}{\sigma \,-[s_1 \;;\; s_2]\rightarrow\, \sigma''}$$

Falls $\mathfrak{A}(\mathit{pos}) = {}^{\blacktriangledown}(s_1 \;;\; s_2)$, dann gilt im Nachfolgezustand $\mathfrak{A}'$, dass $\mathfrak{A}'(\mathit{pos}) = {}^{\blacktriangledown}s_1$.

Mit der Induktionsvoraussetzung für $\sigma \,-[s_1]\rightarrow\, \sigma'$ erreicht man von $\mathfrak{A}'$ aus einen Zustand $\mathfrak{B}$ mit $\mathfrak{B}(\mathit{pos}) = {}^{\blacktriangle}s_1$ und $\sigma' \subseteq \mathfrak{B}(\mathit{store}) \circ \mathfrak{B}(\mathit{env})$.

Im Nachfolgezustand $\mathfrak{B}'$ von $\mathfrak{B}$ gilt $\mathfrak{B}'(\mathit{pos}) = {}^{\blacktriangledown}s_2$.

Mit der Induktionsvoraussetzung für $\sigma' \,-[s_2]\rightarrow\, \sigma''$ erreicht man von $\mathfrak{B}'$ aus einen Zustand $\mathfrak{C}$ mit $\mathfrak{C}(\mathit{pos}) = {}^{\blacktriangle}s_2$ und $\sigma'' \subseteq \mathfrak{C}(\mathit{store}) \circ \mathfrak{C}(\mathit{env})$.

Im Nachfolgezustand $\mathfrak{C}'$ von $\mathfrak{C}$ gilt $\mathfrak{C}'(\mathit{pos}) = {}^{\blacktriangle}(s_1 \;;\; s_2)$.

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Beweis (If-Then-Else) $*$

Fall 4: $$\frac{\sigma -[s_1]\rightarrow \sigma'}{\sigma -[\texttt{if } b \texttt{ then } s_1 \texttt{ else } s_2 \texttt{ end}]\rightarrow \sigma'} \quad \text{falls } \mathcal{B}[\![b]\!]\sigma = 1$$

Falls $\mathcal{B}[\![b]\!]\sigma = 1$ und $\sigma \subseteq \mathfrak{A}(\mathit{store}) \circ \mathfrak{A}(\mathit{env})$, dann ist $\mathcal{B}[\![b]\!](\mathfrak{A}(\mathit{store}) \circ \mathfrak{A}(\mathit{env})) = 1$.

Also gilt im Nachfolgezustand $\mathfrak{A}'$ von $\mathfrak{A}$, dass $\mathfrak{A}'(\mathit{pos}) = {}^{\blacktriangledown}s_1$.

Mit der Induktionsvoraussetzung für $\sigma -[s_1]\rightarrow \sigma'$ erreicht man von $\mathfrak{A}'$ aus einen Zustand $\mathfrak{B}$ mit $\mathfrak{B}(\mathit{pos}) = {}^{\blacktriangle}s_1$ und $\sigma' \subseteq \mathfrak{B}(\mathit{store}) \circ \mathfrak{B}(\mathit{env})$.

Im Nachfolgezustand $\mathfrak{B}'$ von $\mathfrak{B}$ gilt $\mathfrak{B}'(\mathit{pos}) = {}^{\blacktriangle}\texttt{if } b \texttt{ then } s_1 \texttt{ else } s_2 \texttt{ end}$.

Fall 5: $$\frac{\sigma -[s_2]\rightarrow \sigma'}{\sigma -[\texttt{if } b \texttt{ then } s_1 \texttt{ else } s_2 \texttt{ end}]\rightarrow \sigma'} \quad \text{falls } \mathcal{B}[\![b]\!]\sigma = 0$$

Ähnlich wie der Fall 4.

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Beweis (While) *

Fall 6:

$$\frac{\sigma \text{ -}[s]\rightarrow \sigma' \qquad \sigma' \text{ -}[\texttt{while } b \texttt{ do } s \texttt{ end}]\rightarrow \sigma''}{\sigma \text{ -}[\texttt{while } b \texttt{ do } s \texttt{ end}]\rightarrow \sigma''} \quad \text{falls } \mathcal{B}[\![b]\!]\sigma = 1$$

Sei $\mathfrak{A}(\mathit{pos}) = {}^{\blacktriangledown}\texttt{while } b \texttt{ do } s \texttt{ end}$.

Falls $\mathcal{B}[\![b]\!]\sigma = 1$ und $\sigma \subseteq \mathfrak{A}(\mathit{store}) \circ \mathfrak{A}(\mathit{env})$, dann ist $\mathcal{B}[\![b]\!](\mathfrak{A}(\mathit{store}) \circ \mathfrak{A}(\mathit{env})) = 1$.

Also gilt im Nachfolgezustand $\mathfrak{A}'$ von $\mathfrak{A}$, dass $\mathfrak{A}'(\mathit{pos}) = {}^{\blacktriangledown}s$.

Mit der Induktionsvoraussetzung für $\sigma \text{ -}[s]\rightarrow \sigma'$ erreicht man von $\mathfrak{A}'$ aus einen Zustand $\mathfrak{B}$ mit $\mathfrak{B}(\mathit{pos}) = {}^{\blacktriangle}s$ und $\sigma' \subseteq \mathfrak{B}(\mathit{store}) \circ \mathfrak{B}(\mathit{env})$.

Im Nachfolgezustand $\mathfrak{B}'$ von $\mathfrak{B}$ gilt $\mathfrak{B}'(\mathit{pos}) = {}^{\blacktriangledown}\texttt{while } b \texttt{ do } s \texttt{ end}$.

Mit der Induktionsvoraussetzung für $\sigma' \text{ -}[\texttt{while } b \texttt{ do } s \texttt{ end}]\rightarrow \sigma''$ erreicht man von $\mathfrak{B}'$ aus schliesslich einen Zustand $\mathfrak{C}$ mit $\mathfrak{C}(\mathit{pos}) = {}^{\blacktriangle}\texttt{while } b \texttt{ do } s \texttt{ end}$ und $\sigma'' \subseteq \mathfrak{C}(\mathit{store}) \circ \mathfrak{C}(\mathit{env})$.

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Beweis (Fortsetzung) *

Fall 7: $\dfrac{}{\sigma -[\texttt{while } b \texttt{ do } s \texttt{ end}]\rightarrow \sigma}$ falls $\mathcal{B}[\![b]\!]\sigma = 0$

Sei $\mathfrak{A}(\mathit{pos}) = {}^{\blacktriangledown}\texttt{while } b \texttt{ do } s \texttt{ end}$.

Falls $\mathcal{B}[\![b]\!]\sigma = 1$ und $\sigma \subseteq \mathfrak{A}(\mathit{store}) \circ \mathfrak{A}(\mathit{env})$, dann ist $\mathcal{B}[\![b]\!](\mathfrak{A}(\mathit{store}) \circ \mathfrak{A}(\mathit{env})) = 1$.

Also gilt im Nachfolgezustand $\mathfrak{A}'$ von $\mathfrak{A}$, dass $\mathfrak{A}'(\mathit{pos}) = {}^{\blacktriangle}\texttt{while } b \texttt{ do } s \texttt{ end}$.

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Beweis (lokale Variablendeklaration) $*$

Fall 8:

$$\frac{\sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma] \multimap[s]\rightarrowtail \sigma'}{\sigma \multimap[\texttt{var } x \texttt{ := } e \texttt{ in } s \texttt{ end}]\rightarrowtail \sigma' \setminus \{x \mapsto \sigma'(x)\}} \quad \text{falls } x \notin \mathsf{dom}(\sigma)$$

Sei $\mathfrak{A}(\mathit{pos}) = {}^{\blacktriangledown}\texttt{var } x \texttt{ := } e \texttt{ in } s \texttt{ end}$.

Dann gilt im Nachfolgezustand $\mathfrak{A}'$ von $\mathfrak{A}$, dass

- $\mathfrak{A}'(\mathit{pos}) = {}^{\blacktriangledown}s$
- $\mathfrak{A}'(\mathit{env})(x) = \alpha$
- $\mathfrak{A}'(\mathit{store})(\alpha) = \mathcal{A}[\![e]\!](\mathfrak{A}(\mathit{store}) \circ \mathfrak{A}(\mathit{env})) = v$

wobei α eine neue Adresse ist mit $\alpha \notin \mathsf{ran}(\mathfrak{A}(\mathit{env})) \cup \mathsf{dom}(\mathfrak{A}(\mathit{store}))$.

Da $\sigma \subseteq \mathfrak{A}(\mathit{store}) \circ \mathfrak{A}(\mathit{env})$, ist $\mathcal{A}[\![e]\!]\sigma = v$.

Weiter ist $\sigma[x \mapsto v] \subseteq \mathfrak{A}'(\mathit{store}) \circ \mathfrak{A}'(\mathit{env})$.

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Beweis (Fortsetzung) *

Mit der Induktionsvoraussetzung für $\sigma[x \mapsto v] \multimap[s]\rightarrow \sigma'$ errreicht man von $\mathfrak{A}'$ aus einen Zustand $\mathfrak{B}$ mit $\mathfrak{B}(\mathit{pos}) = {}^{\blacktriangle}s$ und $\sigma' \subseteq \mathfrak{B}(\mathit{store}) \circ \mathfrak{B}(\mathit{env})$.

Im Nachfolgezustand $\mathfrak{B}'$ von $\mathfrak{B}$ gilt $\mathfrak{B}'(\mathit{pos}) = {}^{\blacktriangle}\texttt{var}\ x\ \texttt{:=}\ e\ \texttt{in}\ s\ \texttt{end}$.

Bemerkung: Den Fall, dass $x \in \mathrm{dom}(\sigma)$, müssen wir nicht betrachten, da wir bei der ASM-Semantik von Programmen annehmen, dass alle lokale Variablen verschiedene Namen tragen (und nicht überschrieben werden können).

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Beweis (Prozeduraufruf) *

Fall 9:
$$\frac{\{\vec{x} \mapsto \mathcal{A}[\![\vec{e}]\!]\sigma,\ \vec{y} \mapsto \sigma(\vec{z})\} -[s]\rightarrow \sigma'}{\sigma -[p(\vec{e};\vec{z})]\rightarrow \sigma[\vec{z} \mapsto \sigma'(\vec{y})]}$$

mit der Prozedurdeklaration `procedure` $p(\vec{x};\vec{y})$ `begin` s `end`

Sei $\mathfrak{A}(\mathit{pos}) = {}^{\blacktriangledown}p(\vec{e};\vec{z})$.

Dann gilt im Nachfolgezustand $\mathfrak{A}'$ von $\mathfrak{A}$, dass

- $\mathfrak{A}'(\mathit{pos}) = {}^{\blacktriangledown}s$
- $\mathfrak{A}'(\mathit{env}) = \{\vec{x} \mapsto \vec{\alpha}, \vec{y} \mapsto \mathfrak{A}(\mathit{env})(\vec{z})\}$
- $\mathfrak{A}'(\mathit{store})(\vec{\alpha}) = \vec{v}$
- $\mathfrak{A}'(\mathit{frames}) = \mathit{push}(\mathfrak{A}(\mathit{frames}), \langle \mathfrak{A}(\mathit{pos}), \mathfrak{A}(\mathit{env}) \rangle)$

wobei $\vec{v} = \mathcal{A}[\![\vec{e}]\!](\mathfrak{A}(\mathit{store}) \circ \mathfrak{A}(\mathit{env}))$ und $\vec{\alpha}$ neue Adressen sind mit $\vec{a} \notin \mathsf{ran}(\mathfrak{A}(\mathit{env})) \cup \mathsf{dom}(\mathfrak{A}(\mathit{store}))$.

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Beweis (Fortsetzung) *

Da $\sigma \subseteq \mathfrak{A}(\mathit{store}) \circ \mathfrak{A}(\mathit{env})$, ist $\mathcal{A}[\![\vec{e}]\!]\sigma = \vec{v}$.

Da $\mathfrak{A}'(\mathit{env})(\vec{y}) = \mathfrak{A}(\mathit{env})(\vec{z})$, ist

$$(\mathfrak{A}'(\mathit{store}) \circ \mathfrak{A}'(\mathit{env}))(\vec{y}) = (\mathfrak{A}(\mathit{store}) \circ \mathfrak{A}(\mathit{env}))(\vec{z}) = \sigma(\vec{z}).$$

Also ist $\{\vec{x} \mapsto \vec{v},\ \vec{y} \mapsto \sigma(\vec{z})\} \subseteq \mathfrak{A}'(\mathit{store}) \circ \mathfrak{A}'(\mathit{env})$.

Mit der Induktionsvoraussetzung für $\{\vec{x} \mapsto \vec{v},\ \vec{y} \mapsto \sigma(\vec{z})\} \multimap[s]\rightarrow \sigma'$ erreicht man von $\mathfrak{A}'$ aus einen Zustand $\mathfrak{B}$ mit

- $\mathfrak{B}(\mathit{pos}) = {}^{\blacktriangle}s$
- $\sigma' \subseteq \mathfrak{B}(\mathit{store}) \circ \mathfrak{B}(\mathit{env})$
- $\mathfrak{A}'(\mathit{env}) \subseteq \mathfrak{B}(\mathit{env})$
- $\mathfrak{A}'(\mathit{frames}) = \mathfrak{B}(\mathit{frames})$

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Beweis (Fortsetzung) *

Im Nachfolgezustand $\mathfrak{B}'$ von $\mathfrak{B}$ (nach der Rückkehr vom Prozeduraufruf) gilt

- $\mathfrak{B}'(\mathit{pos}) = {}^{\blacktriangle}p(\vec{e}\,;\vec{z})$
- $\mathfrak{B}'(\mathit{env}) = \mathfrak{A}(\mathit{env})$
- $\mathfrak{B}'(\mathit{frames}) = \mathfrak{A}(\mathit{frames})$

Für die Variablen $\vec{z}$ gilt

$$\begin{aligned}(\mathfrak{B}'(\mathit{store}) \circ \mathfrak{B}'(\mathit{env}))(\vec{z}) &= (\mathfrak{B}(\mathit{store}) \circ \mathfrak{A}(\mathit{env}))(\vec{z}) \\ &= (\mathfrak{B}(\mathit{store}) \circ \mathfrak{A}'(\mathit{env}))(\vec{y}) \\ &= (\mathfrak{B}(\mathit{store}) \circ \mathfrak{B}(\mathit{env}))(\vec{y}) = \sigma'(\vec{y})\end{aligned}$$

Sei $u \in \mathsf{dom}(\sigma) \setminus \{\vec{z}\}$ und $\beta = \mathfrak{A}(\mathit{env})(u)$.

Dann ist $\beta \notin \mathsf{ran}(\mathfrak{A}'(\mathit{env}))$ und somit $\mathfrak{A}(\mathit{store})(\beta) = \mathfrak{B}'(\mathit{store})(\beta)$.

Insgesamt ist $\sigma[\vec{z} \mapsto \sigma'(\vec{y})] \subseteq \mathfrak{B}'(\mathit{store}) \circ \mathfrak{B}'(\mathit{env})$. □

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

ASM-Semantik $\Longrightarrow$ natürliche Semantik

Wir betrachten einen (möglicherweise unendlichen) Lauf

$$\mathfrak{A}_0, \mathfrak{A}_1, \mathfrak{A}_2, \ldots, \mathfrak{A}_n, \ldots$$

eines Programms in der ASM-Semantik.

Mit pos_n, mode_n, env_n, frames_n, store_n bezeichnen wir die Werte der Zustandsvariablen im n-ten Zustand $\mathfrak{A}_n$ des Laufs.

Definition. Der Cursor *pos* ist *innerhalb* der Anweisung s auf dem Intervall $[m, n]$, falls für alle $i \in [m \,..\, n]$ gilt:

- Der Stack frames_m ist ein Anfangsstück von frames_i.
- Falls $\mathit{frames}_m = \mathit{frames}_i$, dann ist pos_i in s drin.

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

ASM-Semantik $\Longrightarrow$ natürliche Semantik (Fortsetzung)

Definition. Die Anweisung s wird auf dem Intervall $[m, n]$ ausgeführt, falls gilt:

- $\mathit{pos}_m = {}^{\blacktriangledown}s$
- *pos* ist innerhalb s auf dem Intervall $[m, n]$
- $\mathit{frames}_n = \mathit{frames}_m$
- $\mathit{pos}_n = {}^{\blacktriangle}s$

Theorem. Falls die Anweisung s ausgeführt wird auf dem Intervall $[m, n]$ und $\sigma \subseteq \mathit{store}_m \circ \mathit{env}_m$ mit $\mathsf{FV}(s) \subseteq \mathsf{dom}(\sigma)$, dann ist in der natürlichen Semantik

$$\sigma \,{-}[s]{\rightarrow}\, \sigma'$$

herleitbar für ein σ' mit $\sigma' \subseteq \mathit{store}_n \circ \mathit{env}_n$.

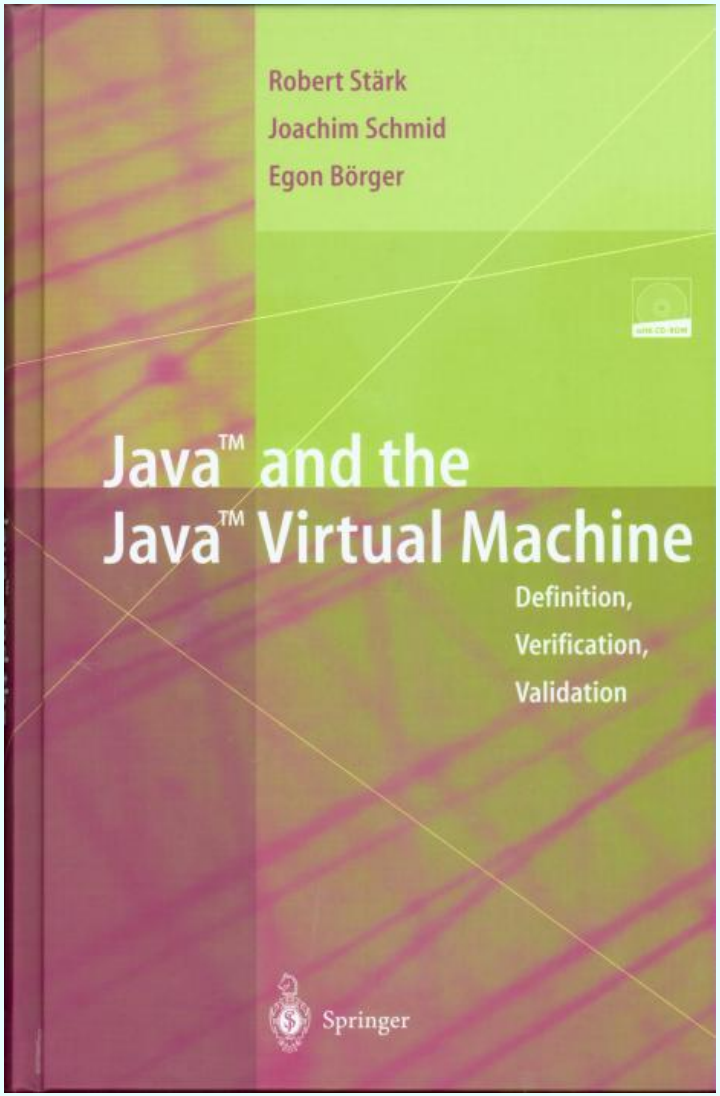
Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Beweis *

Beweis. Mit Induktion nach der Länge des Intervalls und einer Fallunterscheidung nach der Gestalt von s. □

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Ausblick: ASM-Semantik für Java *



Robert Stärk
Joachim Schmid
Egon Börger
Java and the Java Virtual Machine
Definition, Verification, Validation
Springer-Verlag, 2001
(381 pages)

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Der dynamische Zustand von Java *

pos :	Pos	[Position im Syntaxbaum]
$restbody$:	$Phrase$	[Syntaxbaum für die Auswertung]
$locals$:	$Loc \longrightarrow Value$	[Lokale Variablen]
$meth$:	$Class \times MethSig$	[Aktuelle Methode]
$frames$:	$List(Pos \times Phrase \times Locals \times Meth)$	[Stack]
$classState$:	$Class \longrightarrow \{Linked, InProgress, Initialized, Unusable\}$	
$globals$:	$Class \times Field \longrightarrow Value$	[Globale Variablen]
$heap$:	$Ref \longrightarrow Class \times (Class \times Field \hookrightarrow Value)$	[Objekte]

[$\rightsquigarrow$ AsmGoferJava]

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Der dynamische Zustand der JVM *

pc :	$\mathbb{N}$	[Programmzähler]
opd :	$List(Value)$	[Operandenstack]
reg :	$RegNo \longrightarrow Value$	[Lokale Register]
$meth$:	$Class \times MethSig$	[Aktuelle Methode]
$stack$:	$List(Pc \times Opd \times Reg \times Meth)$	
$classState$:	$Class \longrightarrow \{Linked, InProgress, Initialized, Unusable\}$	
$globals$:	$Class \times Field \longrightarrow Value$	
$heap$:	$Ref \longrightarrow Class \times (Class \times Field \hookrightarrow Value)$	

[$\rightsquigarrow$ AsmGoferJVM]

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Liste der Folien

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.

Copyright © 2004 Robert F. Stärk, Computer Science Department, ETH Zürich, Switzerland.