Einladung zu Antrittsvorlesungen

Die Fakultät für Informatik der TU Wien lädt ein zu den Antrittsvorlesungen von...

- Herrn **Univ.-Prof. Dr. Schahram Dustdar** über *Service-Oriented Computing – Fundament moderner Internet-Technologien*
- Herrn **Univ.-Prof. Dr. Reinhard Pichler** über *Daten im Web – Neue Herausforderungen für die Datenbankforschung*

heute

Montag, den 17.10.2005, um 17:00 Uhr s.t.,
in den Hlawka-Hörsaal EI 9, Stiege 1, EG, Gußhausstr. 25-29

Alle Interessenten sind herzlich willkommen!

Hinweise in eigener Sache...

- *Anmeldesystem ist freigeschaltet!*
  `http://www.complang.tuwien.ac.at/anmeldung/`
  ...Anmeldungen sind bis zum 24.10.2005 möglich

  (Abmeldungen sind bis zum 31.10.2005 möglich, ebenfalls über das elektronische Anmeldesystem.)

Weitere Literaturhinweise

1. Krzysztof R. Apt, Ernst-Rüdiger Olderog. *Programmverifikation – Sequentielle, parallele und verteilte Programme*. Springer, 1994.
2. Ernst-Rüdiger Olderog, Bernhard Steffen. *Formale Semantik und Programmverifikation*. In "Handbuch der Informatik", Rechenberg und Pomberger (Hrsg.), Hanser Verlag, 143-163.
3. Hanne R. Nielson, Flemming Nielson. *Semantics with Applications: A Formal Introduction*, Wiley Professional Computing, Wiley, 1992.
   (Siehe `http://www.daimi.au.dk/~bra8130/Wiley_book/wiley.html` für eine frei verfügbare (überarbeitete) Version.)

Die Programmiersprache WHILE

...die Sprache WHILE, der sog. "while"-Kern imperativer Programmiersprachen, besitzt

- Zuweisungen (einschließlich der leeren Anweisung und der Fehleranweisung)
- Fallunterscheidungen
- while-Schleifen
- Sequentielle Komposition

Beachte: Die Sprache WHILE ist "schlank", nichtsdestotrotz aber *Turing-mächtig*!

Überblick über Syntax und Semantik

- **Syntax**

$$\begin{array}{rcl} \pi & ::= & x := t \mid skip \mid abort \mid \\ & & \text{if } b \text{ then } \pi_1 \text{ else } \pi_2 \text{ fi} \mid \\ & & \text{while } b \text{ do } \pi_1 \text{ od} \mid \\ & & \pi_1;\, \pi_2 \end{array}$$

  ...in der Folge bezeichnen wir mit **Prg** die Menge aller WHILE-Programme π.

- **Semantik**
  ...gegeben durch *Zustandstransformationen*:

$$[\![\ ]\!] : \mathbf{Prg} \to (\Sigma \to \Sigma)$$

  mit
  – $\Sigma =_{df} \{\sigma \mid \sigma : \mathbf{Var} \to D\}$ Menge aller *Zustände* über der *Variablenmenge* **Var** und geeignetem *Datenbereich* D.
    (In der Folge werden wir für D oft die Menge der ganzen Zahlen $\mathbb{Z}$ betrachten.)

Zur Semantik von WHILE im Detail

Zunächst: ...in Form einer *strukturell operationellen Semantik* (*SOS*) i.S.v. Gordon Plotkin.

Zentral:

- (*Speicher-*) *Zustände:* $\Sigma =_{df} \{\, \sigma \mid \sigma : \mathbf{Var} \to \mathbb{Z}\,\}$
- *Semantik arithmetischer Ausdrücke:*
$$[\![\, . \,]\!]_A : \mathbf{Aexpr} \to (\Sigma \to \mathbb{Z})$$
- *Semantik Boolescher Ausdrücke:*
$$[\![\, . \,]\!]_B : \mathbf{Bexpr} \to (\Sigma \to \mathbb{B})$$
- *SO-Semantik von WHILE:*
$$[\![\, . \,]\!] : \mathbf{Prg} \to (\Sigma \to \Sigma)$$

Semantik arithmetischer Ausdrücke (1)

$[\![\, . \,]\!]_A : \mathbf{Aexpr} \to (\Sigma \to \mathbb{Z})$ induktiv definiert durch

- $[\![\, n \,]\!]_A(\sigma) =_{df} [\![\, n \,]\!]_N$
- $[\![\, x \,]\!]_A(\sigma) =_{df} \sigma(x)$
- $[\![\, a_1 + a_2 \,]\!]_A(\sigma) =_{df} plus([\![\, a_1 \,]\!]_A(\sigma), [\![\, a_2 \,]\!]_A(\sigma))$
- $[\![\, a_1 * a_2 \,]\!]_A(\sigma) =_{df} mal([\![\, a_1 \,]\!]_A(\sigma), [\![\, a_2 \,]\!]_A(\sigma))$
- $[\![\, a_1 - a_2 \,]\!]_A(\sigma) =_{df} minus([\![\, a_1 \,]\!]_A(\sigma), [\![\, a_2 \,]\!]_A(\sigma))$
- $[\![\, a_1 / a_2 \,]\!]_A(\sigma) =_{df} durch([\![\, a_1 \,]\!]_A(\sigma), [\![\, a_2 \,]\!]_A(\sigma))$
- ... (andere Operatoren analog)

wobei

- $plus,\ mal,\ minus,\ durch\ : \mathbb{Z} \to \mathbb{Z}$ die übliche Addition, Multiplikation, Subtraktion und (ganzzahlige) Division auf den ganzen Zahlen $\mathbb{Z}$ bezeichnen.

Semantik arithmetischer Ausdrücke (2)

$[\![\, . \,]\!]_N : \mathbf{Num} \to \mathbb{Z}$ induktiv definiert durch

- $[\![\, 0 \,]\!]_N =_{df} \mathbf{0}$, ..., $[\![\, 9 \,]\!]_N =_{df} \mathbf{9}$
- $[\![\, n\,i \,]\!]_N =_{df} plus(mal(\mathbf{10}, [\![\, n \,]\!]_A), [\![\, i \,]\!]_N),\ i \in \{0, \ldots, 9\}$
- $[\![\, -n \,]\!]_N =_{df} minus([\![\, n \,]\!]_N)$

wobei

- die Menge der Zahldarstellungen **Num** durch die von n gemäß folgender Grammatik erzeugte Sprache gegeben ist:
$$\begin{array}{rcl} z & ::= & 0 \mid 1 \mid 2 \mid \ldots \mid 9 \\ n & ::= & z \mid n\,z \end{array}$$

Beachte: $0, 1, 2, \ldots$ bezeichnen *syntaktische* Entitäten, $\mathbf{0}, \mathbf{1}, \mathbf{2}, \ldots$ bezeichnen *semantische* Entitäten, in diesem Falle ganze Zahlen.

Semantik Boolescher Ausdrücke (1)

$[\![\,.\,]\!]_B$: **Bexpr** $\rightarrow (\Sigma \rightarrow \mathbb{B})$ induktiv definiert durch

- $[\![\mathit{true}]\!]_B(\sigma) =_{df} \mathtt{tt}$
- $[\![\mathit{false}]\!]_B(\sigma) =_{df} \mathtt{ff}$
- $[\![a_1 = a_2]\!]_B(\sigma) =_{df} \begin{cases} \mathtt{tt} & \text{falls } \mathit{equal}([\![a_1]\!]_A(\sigma), [\![a_2]\!]_A(\sigma)) \\ \mathtt{ff} & \text{sonst} \end{cases}$
- ... (andere Relatoren (z.B. $<$, $\leq$, ...) analog)
- $[\![\neg b]\!]_B(\sigma) =_{df} \mathit{neg}([\![b]\!]_B(\sigma))$
- $[\![b_1 \wedge b_2]\!]_B(\sigma) =_{df} \mathit{conj}([\![b_1]\!]_B(\sigma), [\![b_2]\!]_B(\sigma))$
- $[\![b_1 \vee b_2]\!]_B(\sigma) =_{df} \mathit{disj}([\![b_1]\!]_B(\sigma), [\![b_2]\!]_B(\sigma))$

Semantik Boolescher Ausdrücke (2)

...wobei

- tt und ff die Wahrheitswertkonstanten "wahr" und "falsch" sowie
- conj, disj : $\mathbb{B} \rightarrow \mathbb{B} \rightarrow \mathbb{B}$ und neg : $\mathbb{B} \rightarrow \mathbb{B}$ die übliche zweistellige logische Konjunktion und Disjunktion und einstellige Negation auf der Menge der Wahrheitswerte und
- equal : $\mathbb{Z} \rightarrow \mathbb{Z} \rightarrow \mathbb{B}$ die übliche Gleichheitsrelation auf der Menge der ganzen Zahlen

bezeichnen.

Beachte auch hier den Unterschied zwischen den *syntaktischen* Entitäten true und false und ihren *semantischen* Gegenstücken tt und ff.

Strukturell operationelle Semantik

... beschreibt den Berechnungsvorgang von Programmen π aus WHILE als Folge elementarer Speicherzustandsübergänge.

Zentral dafür: ...der Begriff der *Konfigurationen*!

- Wir unterscheiden:
  - *Nichtterminale* bzw. (*Zwischen-*) *Konfigurationen* von der Form $\langle \pi, \sigma \rangle$:
    ...(Rest-) Programm π ist auf den (Zwischen-) Zustand σ anzuwenden.
  - *Terminale* bzw. *finale Konfigurationen* von den Formen σ oder undef
    ...beschreiben das Resultat nach Ende der Berechnung, wobei Ende nach...
    * *regulärer* Terminierung: angezeigt durch gewöhnliche Zustände σ
    * *irregulärer* Terminierung: angezeigt durch undef

SOS-Regeln von WHILE 1(2)

[skip$_{sos}$] $\dfrac{—}{\langle \mathit{skip}, \sigma \rangle \Rightarrow \sigma}$

[abort$_{sos}$] $\dfrac{—}{\langle \mathit{abort}, \sigma \rangle \Rightarrow \mathit{undef}}$

[ass$_{sos}$] $\dfrac{—}{\langle x{:=}t, \sigma \rangle \Rightarrow \sigma[[\![t]\!]_A(\sigma)/x]}$

[comp$^1_{sos}$] $\dfrac{\langle \pi_1, \sigma \rangle \Rightarrow \langle \pi_1', \sigma' \rangle}{\langle \pi_1;\pi_2, \sigma \rangle \Rightarrow \langle \pi_1';\pi_2, \sigma' \rangle}$

[comp$^2_{sos}$] $\dfrac{\langle \pi_1, \sigma \rangle \Rightarrow \sigma'}{\langle \pi_1;\pi_2, \sigma \rangle \Rightarrow \langle \pi_2, \sigma' \rangle}$

SOS-Regeln von WHILE 2(2)

[if$^{tt}_{sos}$] $\dfrac{—}{\langle \text{if } b \text{ then } \pi_1 \text{ else } \pi_2 \text{ fi}, \sigma \rangle \Rightarrow \langle \pi_1, \sigma \rangle}$ $\quad [\![b]\!]_B(\sigma) = \mathtt{tt}$

[if$^{ff}_{sos}$] $\dfrac{—}{\langle \text{if } b \text{ then } \pi_1 \text{ else } \pi_2 \text{ fi}, \sigma \rangle \Rightarrow \langle \pi_2, \sigma \rangle}$ $\quad [\![b]\!]_B(\sigma) = \mathtt{ff}$

[while$_{sos}$] $\dfrac{—}{\langle \text{while } b \text{ do } \pi \text{ od}, \sigma \rangle \Rightarrow \langle \text{if } b \text{ then } \pi; \text{ while } b \text{ do } \pi \text{ od else } \mathit{skip} \text{ fi}, \sigma \rangle}$

Sprechweisen (1)

Wir unterscheiden

- Prämissenlose *Axiome* der Form

$$\frac{—}{\text{Konklusion}}$$

- Prämissenbehaftete *Regeln* der Form

$$\frac{\text{Prämisse}}{\text{Konklusion}}$$

ggf. mit *Randbedingungen* (*Seitenbedingungen*) wie z.B. in Form von $[\![b]\!]_B(\sigma) = \mathtt{ff}$ in der Regel [if$^{ff}_{sos}$].

Sprechweisen (2)

Im Fall der SO-Semantik von WHILE haben wir demnach

- 6 Axiome
  ...für die leere Anweisung, Fehleranweisung, Zuweisung, Fallunterscheidung und while-Schleife.
- 2 Regeln
  ...für die sequentielle Komposition.

Berechnungsschritte, Berechnungsfolgen

- Ein *Berechnungsschritt* ... ist von der Form

$$\langle \pi, \sigma \rangle \Rightarrow \gamma \quad \text{mit} \quad \gamma \in (\mathbf{Prg} \times \Sigma) \cup \Sigma \cup \{\mathit{undef}\}$$

- Eine *Berechnungsfolge* zu einem Programm π angesetzt auf einen (Start-) Zustand $\sigma \in \Sigma$ ist
  - eine endliche Folge $\gamma_0, \ldots, \gamma_k$ von Konfigurationen mit $\gamma_0 = \langle \pi, \sigma \rangle$ und $\gamma_i \Rightarrow \gamma_{i+1}$ für alle $i \in \{0, \ldots, k-1\}$,
  - eine unendliche Folge von Konfigurationen mit $\gamma_0 = \langle \pi, \sigma \rangle$ und $\gamma_i \Rightarrow \gamma_{i+1}$ für alle $i \in \mathbb{N}$.

Terminierende und divergierende Berechnungsfolgen

- Eine maximale (d.h. nicht mehr verlängerbare) Berechnungsfolge heißt
  - *regulär terminierend*, wenn sie endlich ist und die letzte Konfiguration aus Σ ist,
  - *irregulär terminierend*, wenn sie endlich ist und die letzte Konfiguration $undef$ ist,
  - *divergierend*, falls sie unendlich ist.

Determinismus von WHILE

Lemma Für die SO-Semantik von WHILE gilt:

$$\forall \pi, \sigma, \gamma, \gamma'.\ \langle \pi, \sigma \rangle \Rightarrow \gamma \ \wedge \langle \pi, \sigma \rangle \Rightarrow \gamma' \quad \Rightarrow \quad \gamma = \gamma'$$

Korollar Die SO-Semantik der Sprache WHILE ist *deterministisch*.

...oder etwas plakativer, wenn auch weniger präzise:

WHILE-Programme sind deterministisch!