Syntax von deterministischen Programmen

http://www.informatik.uni-bremen.de/theorie/teach/veri

Renate Klempien-Hinrichs

- Erzeugende Grammatik
- Typen und Datenbereiche
- Konstanten und Werte
- Variablen und Zustände
- Ausdrücke und Auswertung

Syntax

Ein deterministisches Programm ist ein Wort über einem Alphabet, das u.a. die folgenden Symbole enthält:

$$skip, :=, ;, \mathbf{if}, \mathbf{then}, \mathbf{else}, \mathbf{fi}, \mathbf{while}, \mathbf{do}, \mathbf{od}$$

Es wird erzeugt von einer Grammatik mit den Regeln:

Wertzuweisung — bedingte Anweisung

$$S ::= skip \mid x := t \mid S; S \mid \mathbf{if}\ B\ \mathbf{then}\ S\ \mathbf{else}\ S\ \mathbf{fi} \mid \mathbf{while}\ B\ \mathbf{do}\ S\ \mathbf{od}$$

leere Anweisung — sequentielle Komposition — Schleife

Dabei ist x eine einfache oder indizierte Variable, t ist ein Ausdruck und B ist ein Boolescher Ausdruck. In $x := t$ müssen x und t vom selben Typ sein.

Abkürzung: $\mathbf{if}\ B\ \mathbf{then}\ S\ \mathbf{fi} \equiv \mathbf{if}\ B\ \mathbf{then}\ S\ \mathbf{else}\ skip\ \mathbf{fi}$

syntaktische Gleichheit

Typen und Datenbereiche

Typ T (Syntax) $\longmapsto$ **Datenbereich** $\mathcal{D}_T$ (Semantik)

Basistypen

integer

$\mathcal{D}_{\mathbf{integer}} = \{\ldots, -1, 0, 1 \ldots\} = \mathbb{Z}$

Menge der ganzen Zahlen

Boolean

$\mathcal{D}_{\mathbf{Boolean}} = \{\mathbf{true}, \mathbf{false}\} = \mathbb{B}$

Menge der Wahrheitswerte

---

höhere Typen

$T_1 \times \cdots \times T_n \to T$

mit $T_1, \ldots, T_n, T$ Basistypen.

$T_1, \ldots, T_n$ heißen Argumenttypen.

T heißt Wertetyp.

$\mathcal{D}_{T_1 \times \cdots \times T_n \to T} =$

$[\mathcal{D}_{T_1} \times \cdots \times \mathcal{D}_{T_n} \to \mathcal{D}_T]$

Menge der Funktionen von $\mathcal{D}_{T_1} \times \cdots \times \mathcal{D}_{T_n}$ nach $\mathcal{D}_T$

Konstanten und ihre Interpretation

Konstante c des Typs T (Syntax) $\longmapsto$ **Wert** $\mathcal{I}(c)$ in $\mathcal{D}_T$ (Semantik)

Konstante c von einem **Basistyp**

Wert $\mathcal{I}(c) = c$

integer-Konstanten
$\ldots, -2, -1, 0, 1, 2, \ldots$

Boolesche Konstanten
true, **false**

Konstante c von einem **höheren Typ**
$T_1 \times \cdots \times T_n \to T$

Funktionssymbol c (der Stelligkeit n)

Funktion
$\mathcal{I}(c) \colon \mathcal{D}_{T_1} \times \cdots \times \mathcal{D}_{T_n} \to \mathcal{D}_T$

Für $T \equiv$ **Boolean**:
Relationssymbol, Prädikatssymbol

Einige Funktions- und Relationssymbole

- $|\ |$: **integer** $\rightarrow$ **integer**
- $+, -, min, max, *, div, mod$: **integer** $\times$ **integer** $\rightarrow$ **integer**
- $=_{int}, <_{int}, divides$: **integer** $\times$ **integer** $\rightarrow$ **Boolean**
- int: **Boolean** $\rightarrow$ **integer**
- $\neg$: **Boolean** $\rightarrow$ **Boolean**
- $=_{Bool}, \vee, \wedge, \rightarrow, \leftrightarrow$: **Boolean** $\times$ **Boolean** $\rightarrow$ **Boolean**

Variablen (Syntax)

Eine Variable x von einem **Basistyp** heißt einfache Variable.

Eine Variable a von einem **höheren Typ** heißt Feldvariable (kurz: Feld oder Array).

Eine Feldvariable

$$a\colon T_1 \times \cdots \times T_n \to T$$

steht für ein n-dimensionales Feld, dessen Zellen mit Tupeln $[t_1, \ldots, t_n]$ indiziert werden (wobei für alle $i \in \{1, \ldots, n\}$ t_i ein Ausdruck vom Typ T_i ist) und dessen Einträge den Typ T haben.

In diesem Fall wird $a[t_1, \ldots, t_n]$ indizierte Variable genannt.

Variablen in einem Programm

Sei S ein Programm.

- $var(S)$ ist die Menge aller einfachen Variablen und Feldvariablen, die in S vorkommen.
- $change(S)$ ist die Menge aller einfachen Variablen und Feldvariablen, die auf der linken Seite einer Zuweisung in S stehen.

Zustände

- Var_T ist die Menge aller (einfachen oder Feld-)Variablen vom Typ T. $Var = \bigcup_{T \text{ ist Typ}} Var_T$ ist die Menge aller Variablen.
- $\mathcal{D} = \bigcup_{T \text{ ist Typ}} \mathcal{D}_T$ ist die Vereinigung aller Datenbereiche.
- Ein Zustand (auch: Belegung) ist eine Abbildung $\sigma : Var \to \mathcal{D}$, die jeder Variablen $x \in Var_T$ einen Wert $\sigma(x) \in \mathcal{D}_T$ zuordnet (Semantik von x).
- Σ ist die Menge aller Zustände.
- Fehlerzustände:

$\bot$	Divergenz, Nichttermination
Δ	Deadlock, Verklemmung
fail	Laufzeitfehler

  (Diese Zustände sind spezielle Symbole, keine Abbildungen.)

Ausdrücke

Ausdrücke vom (Basis-)Typ T (Syntax) sind induktiv wie folgt definiert:

(i) Jede Konstante c vom Basistyp T ist ein Ausdruck vom Typ T.

(ii) Jede einfache Variable x vom Typ T ist ein Ausdruck vom Typ T.

(iii) $op(t_1, \ldots, t_n)$ ist ein Ausdruck vom Typ T, falls
- op eine Konstante vom Typ $T_1 \times \cdots \times T_n \to T$ ist und
- für alle $i \in \{1, \ldots, n\}$ t_i ein Ausdruck vom Typ T_i ist.

(iv) $a[t_1, \ldots, t_n]$ ist ein Ausdruck vom Typ T, falls
- a eine Feldvariable vom Typ $T_1 \times \cdots \times T_n \to T$ ist und
- für alle $i \in \{1, \ldots, n\}$ t_i ein Ausdruck vom Typ T_i ist.

(v) **if** B **then** t_1 **else** t_2 **fi** ist ein Ausdruck vom Typ T, falls
- B ein Boolescher Ausdruck ist und
- t_1, t_2 Ausdrücke vom Typ T sind.

Syntaxerweiterungen

- Infixnotation für zweistellige Konstanten: $t_1 \, op \, t_2$ statt $op(t_1, t_2)$
- keine Klammern bei $\neg$: $\neg B$ statt $\neg(B)$
- Klammereinsparung durch unterschiedliche Bindungsstärke:

$\cdot$, mod, div
$+$, $-$
$=$, $<$, $divides$
$\vee$, $\wedge$
$\rightarrow$, $\leftrightarrow$

zunehmende Bindungsstärke

Auswertung von Ausdrücken

Der Wert eines Ausdrucks t vom Typ T im Zustand σ (Semantik von t) ist der Wert $\hat{\sigma}(t) \in \mathcal{D}_T$, der induktiv wie folgt definiert ist:

(i) $\hat{\sigma}(c) = \mathcal{I}(c) = c$ für jede Konstante c von einem Basistyp.

(ii) $\hat{\sigma}(x) = \sigma(x)$ für jede einfache Variable x.

(iii) $\hat{\sigma}(op(t_1, \ldots, t_n)) = \mathcal{I}(op)(\hat{\sigma}(t_1), \ldots, \hat{\sigma}(t_n))$ für jede Konstante op von einem höheren Typ und passende Ausdrücke $t_1, \ldots, t_n$.

(iv) $\hat{\sigma}(a[t_1, \ldots, t_n]) = \sigma(a)(\hat{\sigma}(t_1), \ldots, \hat{\sigma}(t_n))$ für jede indizierte Variable $a[t_1, \ldots, t_n]$.

(v) $\hat{\sigma}(\textbf{if } B \textbf{ then } t_1 \textbf{ else } t_2 \textbf{ fi}) = \begin{cases} \hat{\sigma}(t_1) & \text{falls } \hat{\sigma}(B) = \textbf{true} \\ \hat{\sigma}(t_2) & \text{falls } \hat{\sigma}(B) = \textbf{false} \end{cases}$

für jeden bedingten Ausdruck $\textbf{if } B \textbf{ then } t_1 \textbf{ else } t_2 \textbf{ fi}$.