Theorie reaktiver Systeme

Reaktive Systeme

Transitionssysteme

Operationelle Semantik

Korrigierte Fassung 25.10.2007

Reaktive Systeme

Bekannt: transformationelle Systeme
Klassischer Algorithmenbegriff
Eingabe – Verarbeitung – Ausgabe – Terminierung

Reaktive Systeme:

- ständige Interaktion mit der Umwelt
- i.d.R. Terminierung unerwünscht
- Reaktion auch abhängig von vorherigem Verhalten

Transitionssysteme

Ein Zustandsübergangssystem (Transitionssystem) ist ein Tripel (S, s_0, T) mit

- S, eine Menge von Zuständen
- $s_0 \in S$, Anfangszustand
- $T \subseteq S \times S$, Transitionsrelation

Interpretation: $(s_1, s_2) \in T$: Zustandsübergang von s_1 nach s_2 möglich.
Deterministisches TS: T ist (partielle) Funktion $T \in S \not\longrightarrow S$
Eine Berechnung (Ausführung) ist eine Folge $s_0, s_1, s_2, \ldots$ mit $(s_i, s_{i+1}) \in T$ für $i \geq 0$.

Ein Labelled Transition System ist ein Tupel (S, s_0, T, Σ) mit

- Σ, Labelmenge, Alphabet
- $T \subseteq S \times \Sigma \times S$

Notation: $s_1 \xrightarrow{e} s_2 \in T$ mit $e \in \Sigma$.

Eine Berechnung hat die Form $s_0 \xrightarrow{e_1} s_1 \xrightarrow{e_2} s_2 \cdots$ mit $s_i \xrightarrow{e_{i+1}} s_{i+1} \in T$

Ein Trace (Spur) ist eine Folge von Markierungen einer Berechnung, also $\langle e_1, e_2, e_3 \ldots \rangle$.

Semantik einer **while**-Sprache als STS

$S ::= x := expr(y_1, \ldots, y_k) \mid S_1;S_2 \mid$ **if** $bexpr$ **then** S_1
while $bexpr$ **do** S
über einer endlichen Variablenmenge Var.

Zustandsraum : $S = String \times VState$
Variablenbelegung $\sigma \in VState$
$VState = V \mapsto D = \bigcup_{v \in Var} D_v$

Für Programm P mit initialer Variablenbelegung σ_0 ist der Anfangszustand $s_0 = (P, \sigma_0)$.

Zuweisung: $(x := expr(y_1, \ldots, y_k); P, \sigma) \rightarrow$

$$(P, (\sigma : x \mapsto \underline{expr}(\sigma(y_1), \ldots, \sigma(y_k))))$$

if-1: $(\textbf{if } b(y_1, \ldots, y_k) \textbf{ then } S_1; S_2, \sigma) \rightarrow (S_1; S_2, \sigma)$, falls $\underline{b}(\sigma(y_1), \ldots, \sigma(y_k))$

if-2: $(\textbf{if } b(y_1, \ldots, y_k) \textbf{ then } S_1; S_2, \sigma) \rightarrow (S_2, \sigma)$, falls $\neg\underline{b}(\sigma(y_1), \ldots, \sigma(y_k))$

while -1: $(\textbf{while } b(y_1, \ldots, y_k) \textbf{ do } S_1; S_2, \sigma) \rightarrow$

$(S_1; \textbf{ while } b(y_1, \ldots, y_k) \textbf{ do } S_1; S_2, \sigma)$, falls $\underline{b}(\sigma(y_1), \ldots, \sigma(y_k))$

while -2: $(\textbf{while } b(y_1, \ldots, y_k) \textbf{ do } S_1; S_2, \sigma) \rightarrow (S_2, \sigma)$,

falls $\neg\underline{b}(\sigma(y_1), \ldots, \sigma(y_k))$

Sequenz **;**: $(S_1; S_2, \sigma) \rightarrow (S_1'; S_2, \sigma')$, falls $(S_1, \sigma) \rightarrow (S_1', \sigma')$

CSP - Communicating Sequential Processes

[C.A.R. Hoare, 1985]

STOP	Deadlock
SKIP	Terminierung
$a \rightarrow Q$	Event prefix
$Q \,\Box\, R$	External choice
$Q \sqcap R$	Internal choice
$Q \,;\, R$	Sequentielle Komposition
$Q \underset{A}{\parallel} R$	Parallele Komposition, synchronisiert über A
$Q \underset{A}{\lvert\lvert\lvert} R$	Interleaving
$Q \setminus A$	Hiding

und weitere Operatoren....