Parallele disjunkte Programme

Programm S_1, S_2 **disjunkt**:
$change(S_1) \cap var(S_2) = \emptyset$
$\wedge change(S_2) \cap var(S_1) = \emptyset$

S und ein Ausdruck p
eine Zusicherung t **disjunkt:**
$change(S) \cap var(t) = \emptyset$

Beispiel

$$\begin{array}{ll} disjunkt: & S_1 \equiv x := z, S_2 \equiv y := z \\ nicht\ disjunkt: & S_1 \equiv x := y, S_2 \equiv y := z \end{array}$$

Syntax und Semantik deterministischer Programme plus:

disjunkte parallele Komposition
$S ::= [S_1 \| \dots \| S_n]$
$S_1, \dots, S_n : \ disjunkt, \ deterministisch$

Transitionsregel:

$$\frac{< S_i, \sigma > \to < T_i, \tau >}{< [\dots \| S_i \| \dots], \sigma > \to < [\dots \| T_i \| \dots], \tau >}$$

"interleaving concurrency modeling"

Beispiel:

$$\begin{aligned}
&< [x := 1 \| y := 2 \| z := 3], \sigma > \\
&\to < [E \| y := 2 \| z := 3], \sigma[x := 1] > \\
&\to < [E \| E \| z := 3], \sigma[x := 1][y := 2] > \\
&\to < [E \| E \| E], \sigma[x := 1][y := 2][z := 3] >
\end{aligned}$$

Termination:

$[E\| \ldots \|E] \equiv E$

$S\ terminiert \leftrightarrow S_1 \ldots S_n\ terminiert$

Kein Blockieren:

$$\begin{array}{l} \forall < S, \sigma >, \\ S \neq E,\ \sigma\ ein\ g\ddot{u}ltiger\ Zustand : \\ \exists < T, \tau >:< S, \sigma > \rightarrow < T, \tau > \end{array}$$

Menge der Endzustände:

$$\begin{array}{lcl} \mathcal{M}[\![S]\!](\sigma) & = & \{\tau| < S, \sigma > \rightarrow^* < E, \tau >\} \\ \mathcal{M}_{tot}[\![S]\!](\sigma) & = & \mathcal{M}[\![S]\!](\sigma) \cup \\ & & \{\perp | S\ divergiert\ in\ \sigma\} \end{array}$$

Korrektheit:

$$\models_{tot} \{p\}S\{q\}\ wenn\ \mathcal{M}_{tot}[\![S]\!]([\![p]\!]) \subseteq [\![q]\!]$$

Disjunktivität verbietet:

- Beeinflussung des Kontrollfluß

- Veränderung der Daten

zwischen den parallelen Programmen. Folge:

Determinismus:
Alle Durchgänge eines disjunkten parallelen Programmes produzieren vom selben Initialzustand aus denselben Zustand:

$$|\mathcal{M}_{tot}[\![S]\!](\sigma)| = 1$$

Daraus folgt die Möglichkeit der **Sequenzialisierung**:

$$\mathcal{M}[\![[S_1 \| \dots \| S_n]]\!] = \mathcal{M}[\![S_1; \dots; S_n]\!]$$
$$\mathcal{M}_{tot}[\![[S_1 \| \dots \| S_n]]\!] = \mathcal{M}_{tot}[\![S_1; \dots; S_n]\!]$$

AXIOM 1: SKIP

$$\{p\}skip\{p\}$$

AXIOM 2: ZUWEISUNG

$$\{p[u := t]\}u := t\{p\}$$

REGEL 3: KOMPOSITION

$$\frac{\{p\}S_1\{r\}, \{r\}S_2\{q\}}{\{p\}S_1; S_2\{q\}}$$

REGEL 4: BEDINGUNG

$$\frac{\{p \wedge B\}S_1\{q\}, \{p \wedge \neg B\}S_2\{q\}}{\{p\} \textbf{ if } B \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi } \{q\}}$$

REGEL 5: SCHLEIFE^{part}

$$\frac{\{p \wedge B\}S\{p\}}{\{p\} \textbf{ while } B \textbf{ do } S \textbf{ od } \{p \wedge \neg B\}}$$

REGEL 6: KONSEQUENZ

$$\frac{p \to p_1, \{p_1\}S\{q_1\}, q_1 \to q}{\{p\}S\{q\}}$$

REGEL 7: SCHLEIFE II^{tot}

$$\frac{\begin{array}{l}\{p \wedge B\}\ S\ \{p\},\\ \{p \wedge B \wedge t = z\}\ S\ \{t < z\},\\ p \to t \geq 0\end{array}}{\{p\}\textbf{while } B \textbf{ do } S \textbf{ od}\{p \wedge \neg B\}}$$

Hilfsregeln:

`AXIOM A1: INVARIANZ`^{part}

$$\{p\}S\{p\},$$

wobei $free(p) \cap change(S) = \emptyset$.

`REGEL A2: DISJUNKTION`

$$\frac{\{p\}S\{q\}, \{r\}S\{q\}}{\{p \vee r\}S\{q\}}$$

`REGEL A3: KONJUNKTION`

$$\frac{\{p_1\}S\{q_1\}, \{p_2\}S\{q_2\}}{\{p_1 \wedge p_2\}S\{q_1 \wedge q_2\}}$$

`REGEL A4:` $\exists$`-INTRODUCTION`

$$\frac{\{p\}S\{q\}}{\{\exists x : p\}S\{q\}},$$

wobei x nicht in S oder $free(q)$ vorkommt.

`REGEL A5: INVARIANZ`

$$\frac{\{r\}S\{q\}}{\{p \wedge r\}S\{p \wedge q\}},$$

wobei $free(p) \cap change(S) = \emptyset$.

REGEL 15: SEQUENZIALISIERUNG

$$\frac{\{p\}S_1;\ldots;S_n\{q\}}{\{p\}[S_1\|\ldots\|S_n]\{q\}}$$

REGEL 16: DISJUNKTE PARALLELITÄT

$$\frac{\{p_i\}S_i\{q_i\}, i \in \{1,\ldots,n\}}{\left\{\bigwedge_{i=1}^{n} p_i\right\}[S_1\|\ldots\|S_n]\left\{\bigwedge_{i=1}^{n} q_i\right\}}$$

REGEL 17: HILFSVARIABLEN

Kommt eine Variable x in einem Programm nur in der Form $x := t$ vor, ist x eine Hilfsvariable.

$$\frac{\{p\}S\{q\}}{\{p\}S_0\{q\}},$$

wobei S_0 aus S entsteht, wenn alle Zuweisungen an Hilfsvariablen aus S entfernt werden.

Parallele Programme mit gemeinsamen Variablen

- parallele Komposition ohne Disjunktivität
- atomare Regionen
- “interleaving concurrency modeling”

Syntax und Semantik deterministischer Programme plus:

atomare Regionen

$S ::= \langle S_0 \rangle$

$S_0 :\ deterministisch,$
$schleifenfrei,$
$enthält\ keine\ atomare\ Region$

parallele Komposition

$S ::= [S_1 \| \ldots \| S_n]$

$S_1, \ldots, S_n :\ deterministisch$

Abwechselnde Ausführung der atomaren Schritte:

atomare Region
Boolescher Ausdruck
Zuweisung
skip

Ein **normales Subprogramm** von S darf nicht Anteile eines atomaren Schrittes von S enthalten.

$\{p\}\ S^*\ \{q\}$ ist **Standard Beweis Skizze**, wenn jedem normalen Subprogramm T genau eine Zusicherung $pre(T)$ vorangeht.

Transitionsregel:

$$\frac{< \ S, \sigma \ > \rightarrow^{*} < \ E, \tau \ >}{< \ \langle S \rangle, \sigma \ > \rightarrow < \ E, \tau \ >}$$

$$\frac{< S_i, \sigma > \rightarrow < T_i, \tau >}{< [\ldots \| S_i \| \ldots], \sigma > \rightarrow < [\ldots \| T_i \| \ldots], \tau >}$$

Kein Blockieren

Begrenzter Nichtdeterminismus:
$\mathcal{M}_{tot}[\![S]\!](\sigma)$ ist endlich oder enthält $\perp$.

Keine Kompositionalität:
Es gibt keine Operation op, die aus den Endzuständen der Komponenten die Enzustände der Komposition bildet:

$$\mathcal{M}_{tot}[\![[S_1\|\ldots\|S_n]]\!] = op(\mathcal{M}[\![S_1]\!], \ldots, [\![S_n]\!])$$

`Beweis System PSV für partielle Korrektheit:`
Regeln 1-6,17 und A1-A5 und folgende.

Beweis System ohne Regel für Hilfsvariablen nicht vollständig!

`REGEL 18: ATOMARE REGION`

$$\frac{\{p\}\ S\ \{q\}}{\{p\}\ \langle S\rangle\ \{q\}}$$

Interferenz Freiheit

Ein Programm R interferiert nicht mit der Standard Beweis Skizze $\{p\}\ S^*\ \{q\}$, wenn für jede Zusicherung r gilt: $\{r \wedge pre(R)\}\ R\ \{r\}$.

Die Standard Beweis Skizzen $\{p_i\}\ S_i^*\ \{q_i\},\ i \in \{1, \ldots, n\}$ interferieren nicht, wenn kein normales Subprogramm von S_j mit der Standard Beweis Skizze $\{p_i\}\ S_i^*\ \{q_i\}$ interferiert, $i \neq j$.

REGEL 19: PARALLELISMUS MIT GEMEINSAMEN VARIABLEN

$$\frac{\{p_i\}\ S_i^*\ \{q_i\},\ i \in \{1, \ldots, n\}}{\{\bigwedge_{i=1}^n p_i\}\ [S_1 \| \ldots \| S_n]\ \{\bigwedge_{i=1}^n q_i\}},$$

wobei $\{p_i\}\ S_i^*\ \{q_i\},\ i \in \{1, \ldots, n\}$ nicht interferieren.

Synchronisierte Programme

Syntax
Deterministische Programmsyntax plus:

- Parallele Komposition

  $$S ::= [S_1 \| \ldots \| S_n]$$

  S_1 bis S_n sequentielle Komponenten.

- Synchronisation

  $$S ::= \textbf{await } B \textbf{ then } S_0 \textbf{ end}$$

  S_0 schleifenfrei und ohne **await** Anweisungen.

Semantik
Transition

- Parallele Komposition

$$\frac{< S_i, \sigma > \rightarrow < T_i, \tau >}{< [\ldots \| S_i \| \ldots], \sigma > \rightarrow < [\ldots \| T_i \| \ldots], \tau >}$$

- Synchronisation

$$\frac{< \; S, \sigma \; > \rightarrow^* < \; E, \tau \; >}{< \textbf{await } B \textbf{ then } S \textbf{ end}, \sigma > \rightarrow < E, \tau >}$$

wenn $\sigma \models B$.

deadlock
Ein *deadlock* ist vorhanden, wenn alle Komponenten blockiert oder terminiert sind.

1. Eine Konfiguration $< S, \sigma >$ nennt sich Deadlock, wenn $S \not\equiv E$ und keine nachfolgende Konfiguration von $< S, \sigma >$ in der Transitionsrelation existiert.

2. Das Programm kann einen Deadlock erzeugen, wenn eine Berechnung von S aus einem Zustand σ in einem Deadlock endet.

3. Das Programm S ist Deadlockfrei, wenn es keinen Status σ gibt, durch den S zum Deadlock kommt.

Endzustand

- Semantik für partielle Korrektheit

$$\begin{aligned}\mathcal{M}[\![S]\!](\sigma) &= \{\tau \mid \\ &\quad < S, \sigma > \rightarrow^* < E, \tau >\}\end{aligned}$$

- Semantik für schwache totale Korrektheit

$$\begin{aligned}\mathcal{M}_{wtot}[\![S]\!](\sigma) &= \mathcal{M}[\![S]\!](\sigma) \cup \{\perp \mid \\ &\quad S \text{ kann divergieren ab } \sigma\}\end{aligned}$$

- Semantik für totale Korrektheit

$$\begin{aligned}\mathcal{M}_{tot}[\![S]\!](\sigma) &= \mathcal{M}_{wtot}[\![S]\!](\sigma) \cup \{\Delta \mid \\ &\quad S \text{ kann blockieren ab } \sigma\}\end{aligned}$$

Beweis Theorie

`REGEL 20: SYNCHRONISATION`

$$\frac{\{p \land B\}\ S\ \{q\}}{\{p\}\ \textbf{await}\ B\ \textbf{then}\ S\ \textbf{end}\ \{q\}}$$

Beweis Skizze

$$\frac{\{p \land B\}\ S^*\ \{q\}}{\{p\}\ \textbf{await}\ B\ \textbf{then}\ \{p \land B\}\ S^*\ \{q\}\ \textbf{end}\ \{q\}}$$

Schwach totale Korrektheit

Die Definition von $path$ wird um

- $$\begin{aligned} & path(\textbf{await } B \textbf{ then } S \textbf{ end}) \\ = & \{\textbf{await } B \textbf{ then } S \textbf{ end}\} \end{aligned}$$

erweitert.

Totale Korrektheit

Nachweis der *deadlock* Freiheit – potentielle *deadlock* Situationen:

- $(R_1, \ldots, R_n)$ potenteller *deadlock*, wenn jedes R_i **await** Anweisung in S_i oder E ist und zumindest ein R_i eine **await** Anweisung in S_i ist.

- $(r_1, \ldots, r_n)$ ein Tupel von Zusicherungen:

$$\begin{aligned} r_i &\equiv pre(R_i) \wedge \neg B \text{ wenn} \\ R_i &\equiv \textbf{await } B \textbf{ then } S \textbf{ end} \end{aligned}$$

$$\begin{aligned} r_i &\equiv q_i \text{ wenn} \\ R_i &\equiv E \end{aligned}$$

RULE 21: DEADLOCK FREIER PARALLELISMUS

$$\frac{\begin{array}{ll}(1) & \{p_i\}S^*\{q_i\} \text{ interferenzfreie Beweisskizze} \\ & \text{für schwach totale Korrektheit} \\ (2) & \forall(R_1,\ldots,R_n) : \neg \bigwedge_{i=1}^{n}\ r_i\end{array}}{\{\bigwedge_{i=1}^{n}\ p_i\}[S_1\|\ldots\|S_n]\{\bigwedge_{i=1}^{n}\ q_i\}}$$

Beweis System für totale Korrektheit:
Regel 1-5,7,17,20,21 und A2-A5.

Beweis Skizzen werden entsprechend angefertigt.

Beispiel:

$$\{x = 0\}$$
$$\textbf{await } x = 1 \textbf{ then } \textit{skip} \textbf{ end} \| x := 1]$$
$$\{x = 1\}$$

$$\{x = 0 \vee x = 1\}$$
$$\textbf{await } x = 1 \textbf{ then } \textit{skip} \textbf{ end}$$
$$\{x = 1\}$$

$$\{x = 0\}\ x := 1\ \{x = 1\}$$

Potentielle *deadlocks*:

$$(\textbf{await } x = 1 \textbf{ then } \textit{skip} \textbf{ end}, E)$$

mit den Zusicherungen:

$$((x = 0 \vee x = 1) \wedge x \neq 1, x = 1)$$

Kein *deadlock* möglich !

Fallstudie: Gegenseitiger Ausschlußes

$$
\begin{array}{rl}
S_i \equiv & \textbf{while true do} \\
& \quad NC_i; \\
& \quad ACQ_i; \\
& \quad CS_i; \\
& \quad REL_i \\
& \textbf{od}
\end{array}
$$

Gesamtprogramm:

$$S \equiv INIT;\ [S_1 \| \ldots \| S_n]$$

1. Gegenseitiger Ausschluß

   $$\forall i, j \in \{1, \ldots, n\} : \neg(pre(CS_i) \wedge pre(CS_j))$$

2. *deadlock* Freiheit

Lösung mit binärer Semaphore

$\{$**inv : true**$\}$
while true do
$\{$**true**$\}$
NC_i;
$\{$**true**$\}$
await out **then** $out :=$ **false**; $who := i$ **end**;
$\{\neg out \wedge who = i\}$
CS_i; $\{\neg out \wedge who = i\}$
$out :=$**true**
od
$\{$**false**$\}$

Beweis der Interferenzfreiheit

1. $\{\neg out \wedge who = i \wedge out\}$
   $\{$**false**$\}$
   $out :=$ **false**$; who := j$
   $\{$**false**$\}$
   $\{\neg out \wedge who = i\}$

   Nach Regel 20:

   $\{\neg out \wedge who = i \wedge \{$**true**$\}$
   await out **then** $out :=$ **false**$; who :=$
   j **end**
   $\{\neg out \wedge who = i\}$

2. $\{\neg out \wedge who = i \wedge \neg out \wedge who = j\}$
   $\{$**false**$\}$
   $out :=$ **true**
   $\{$**false**$\}$
   $\{\neg out \wedge who = i\}$

Beweis des gegenseitigen Ausschluß

$$
\begin{array}{rl}
 & pre(CS_i) \wedge pre(CS_j) \\
\rightarrow & who = i \wedge who = j \\
\rightarrow & \neg(pre(CS_I) \wedge pre(CS_j))
\end{array}
$$

deadlock **Freiheit**

$\{$**inv** : $who \neq i \wedge I\}$
while true do
  $\{who \neq i \wedge I\}$
  NC_i;
  $\{who \neq i \wedge I\}$
  await out **then** $out :=$ **false**; $who := i$ **end**;
  $\{\neg out \wedge who = i\}$
  CS_i; $\{\neg out \wedge who = i\}$
  $\langle out :=$ **true**; $who := 0\rangle$
od
$\{$**false**$\}$

$$I \equiv (\vee_{j=0}^{n}\ who = j) \wedge (who = 0 \leftrightarrow out)$$

$$\begin{array}{ll} & r_i \equiv who \neq i \wedge I \wedge \neg out \\ \rightarrow & (\wedge_{i=1}^{n}\ r_i) \rightarrow who = 0 \\ & \wedge (who = 0 \leftrightarrow out) \wedge \neg out \\ \rightarrow & \neg \wedge_{i=1}^{n}\ \ r_i) \end{array}$$