8.4 Beweisskizzen

Ziel: Führe eine Notation für Beweise mittels Programmlogiken ein.

Ansatz: Annotierte Programme mit Zwischenzusicherungen, die an den entsprechenden Stellen gelten.

→ **Beweisskizze / Proof-Outline**

Sequentielle Programme:

Beispiel:

$\{emp\}$
$\quad x := cons(0);$
$\{x \mapsto 0\}$
$\quad [x] := 1;$
$\{x \mapsto 1\}$

Diese Beweisskizze repräsentiert Ableitungsschritte in der Programmlogik:

- M.1 (ALLOCV) erhalten wir
  $\{emp\}\ x := cons(0)\ \{x \mapsto 0\}$
- M.1 (MUTV) gilt:
  $\{x \mapsto 0\}\ [x] := 1\ \{x \mapsto 1\}.$
- M.1 (SEQ) folgt aus beiden Fakten:
  $\{emp\}\ x := cons(0);\ [x] := 1\ \{x \mapsto 1\}.$

Konvention:

- Wir benutzen die Struktural-Rules + (FRAME) implizit und schreiben
  $\{x \mapsto 0\}\ c := cons(0)\ \{x \mapsto 0 * c \mapsto 0\}.$

Hier verwenden wir (ALLOCV) + (FRAME) + (CONSEQUENCE).

• Wenn wir die Verwendung von (CONSEQUENCE) explizit machen wollen, schreiben wir aufeinander folgende Zusicherungen:

$\{x \mapsto 0\} \; \{x \mapsto 0 * \text{emp}\} \; c := \text{cons}(0) \; \{x \mapsto 0 * c \mapsto 0\}$.

Parallele Programme:

Hier nutzen wir Zwischenzusicherungen auf ganz ähnliche Weise, allerdings gehen wir auch die Verwendung der Ressourcen-Invarianten an.

(PAR)

$$\frac{\{P_1 * P_2\} \quad \{P_1\}\, c_1\, \{Q_1\} \;\|\; \{P_2\}\, c_2\, \{Q_2\}}{\{Q_1 * Q_2\}}$$

(ATOM)

$$\frac{J \vdash \left(\begin{array}{l} \{P\} \\ \quad \text{atomic}\, \{ \\ \quad\quad \{P * J\} \\ \quad\quad\quad c \\ \quad\quad \{Q * J\} \\ \quad \} \\ \{Q\} \end{array}\right)}{\{P\}\; c \;\{Q\}}$$

(FRAME)

$$\frac{J \vdash \left(\begin{array}{l} \{P\} \\ \quad c \\ \{Q\} \end{array}\right)}{\{P * J\}\; c \;\{Q * J\}}$$

8.5 Ownership-Transfer

Ziel: Zeige, dass sich eine atomare Variante des Message-Passing-Idioms in CSL verifizieren lässt.

Message-Passing:

T_1: { : } schreibe Speicher
$[flag] := 1;$

$\;\|\;$

T_2: do { $r := [flag];$ } Busy-Waiting
$\text{while}\ (r == 0);$
{ : } Lies den Speicher aus

Genauer wollen wir zeigen, dass der geschriebene Speicher mithilfe der Ressourcen-Invariante vom lokalen Speicher des einen in den lokalen Speicher des anderen Threads wandert

Define $J := c \mapsto 0 \vee (c \mapsto 1 * x \mapsto 1)$

$\{emp\}$
$x := cons(0);$
$\{x \mapsto 0\}$
$c := cons(0);$
$\{x \mapsto 0 * c \mapsto 0\}$
$\{x \mapsto 0 * emp * J\}$

$J \vdash$

$\{x \mapsto 0 * emp\}$

$\{x \mapsto 0\}$	$\{emp\}$
$[x] := 1;$	atomic $\{$
$\{x \mapsto 1\}$	$\{emp * J\}$
atomic $\{$	$\{c \mapsto 0 \vee (c \mapsto 1 * x \mapsto 1)\}$
$\{x \mapsto 1 * J\}$	$t := [c];$
$\{x \mapsto 1 * c \mapsto 0\}$	$\{(c \mapsto 0 \wedge t = 0) \vee (c \mapsto 1 * x \mapsto 1 \wedge t = 1)\}$
$[c] := 1;$	assume $(t = 1);$
$\{x \mapsto 1 * c \mapsto 1\}$	$\{[(c \mapsto 0 \wedge t = 0) \vee (c \mapsto 1 * x \mapsto 1 \wedge t = 1)] \wedge t = 1\}$
$\{emp * J\}$	$\{x \mapsto 1 * c \mapsto 1\}$
$\}$	$[c] := 0;$
$\{emp\}$	$\{x \mapsto 1 * c \mapsto 0\}$
	$\{x \mapsto 1 * J\}$
	$\}$
	$\{x \mapsto 1\}$
	$t := [x];$
	$\{x \mapsto 1 \wedge t = 1\}$

$\{emp * (x \mapsto 1 \wedge t = 1)\}$

$\{x \mapsto 1 \wedge t = 1\}$

$\{(x \mapsto 1 \wedge t = 1) * J\}$
$\{x \mapsto 1 \wedge t = 1 * c \mapsto 0\}$