9.2 RGSep-Aussagen

Ziel: • RGSep = Rely-Guarantee + Separation-Logik.

• Hat neue Aussagen.

Idee: RGSep unterteilt den Heap in

h_S = shared Heap

h_L = lokaler Heap.

mit der Forderung, dass

$h_L \oplus h_S$

definiert ist.

Definition:

Die RGSep-Aussagen sind wie folgt definiert:

$$P ::= \underbrace{A}_{\text{lokale Aussagen}} \mid \underbrace{\boxed{A}}_{\substack{\text{shared Aussagen, auch} \\ \text{boxed Aussagen}}} \mid P_1 \overset{*}{\underset{\vee}{\wedge}} P_2 \mid \exists x.\, P \mid \forall x.\, P$$

Dabei ist A eine SL-Aussage wie zuvor.

Beachte, dass Boxen nicht geschachtelt werden können.

Definition:

Die Semantik von RGSep-Aussagen ist die Funktion:

$$[\![-]\!] : \text{RGSepAssn} \longrightarrow \text{Stack} \times \text{Heap} \times \text{Heap} \longrightarrow \mathbb{B}$$

mit

- $[\![A]\!](s, h_L, h_S) \quad :\Leftrightarrow \quad [\![A]\!]_{SL}(s, h_L) \wedge h_L \cap h_S = \emptyset$
- $[\![\boxed{A}]\!](s, h_L, h_S) \quad :\Leftrightarrow \quad [\![A]\!]_{SL}(s, h_S) \wedge h_L = \emptyset$

$$[\![P_1 * P_2]\!](s, h_L, h_S) \iff \exists h_1, h_2.\ h_L = h_1 \cdot h_2 \wedge [\![P_1]\!](s, h_1, h_S) \wedge [\![P_2]\!](s, h_2, h_S).$$

$$[\![P_1 \hat{\vee} P_2]\!](s, h_L, h_S) \iff [\![P_1]\!](s, h_L, h_S) \hat{\vee} [\![P_2]\!](s, h_L, h_S)$$

$$[\![\forall x.\ P]\!](s, h_L, h_S) \iff \forall v.\ [\![P]\!](s[x \mapsto v], h_L, h_S)$$

$$[\![\exists x.\ P]\!](s, h_L, h_S) \iff \exists v.\ [\![P]\!](s[x \mapsto v], h_L, h_S).$$

Bemerkungen:

- $*$ spaltet den lokalen also nicht den shared State.
  Terminologie: $*$ ist multiplikativ über den lokalen State
  additiv über den shared State.
- Insbesondere gilt: $\boxed{\Pi_1} * \boxed{\Pi_2} \iff \boxed{\Pi_1 \wedge \Pi_2}$.
- Man kann auch eine Definition ohne $h_L = \emptyset$ bei $\boxed{\Pi}$ wählen.
  Unser Ansatz führt in der Praxis zu kürzeren Annotationen.

Bemerkung:
RGSep überlädt die Operatoren $*, \wedge, \vee, \exists$ und $\forall$.
Also

$$x \mapsto 1 * y \mapsto 2$$

kann gemeint werden als RGSep-Operator

$$P = \Pi_1 * \Pi_2$$

lokale Assertion in RGSep

oder

$$P = \Pi$$

Folgendes Lemma erlaubt diese Überladung / schlägt sie vor.

Lemma:

Schreibe $Local(P)$ für ein lokales RGSep-Prädikat (Π, wie folgt).

Es gilt:

$$\frac{RGSep}{Local(P)} \iff \underset{SL}{P}$$

$$Local(P) \underset{(RG)}{*} Local(Q) \iff Local(\underset{(SL)}{P * Q})$$

$$Local(P) \underset{(RG)}{\hat{\vee}} Local(Q) \iff Local(\underset{(SL)}{P \hat{\vee} Q})$$

$$\underbrace{\exists x.\ Local(P)}_{RGSep} \iff Local(\underbrace{\exists x.\ P}_{SL})$$

9.3 RGSep-Rechts

Ziel:
- Rely-Guarantee nutzt Prädikate über Paaren von Zuständen, um die Interferenz zwischen den Threads zu beschreiben (R, G).
- RGSep nutzt dafür Prädikate

$$\overline{x}.\ A \rightsquigarrow B.$$

Definition:

$$[\![\overline{x}.\ A \rightsquigarrow B]\!] := \{(s, h_1 \circledast h_0, h_2 \circledast h_0) \mid \exists \overline{v}.\ [\![A[\overline{x}/\overline{v}]]\!](s, h_1) \wedge [\![B[\overline{x}/\overline{v}]]\!](s, h_2)\}$$

Bemerkung:
- Relation zwischen shared Heap h_1, der die Precondition A erfüllt, und resultierendem shared Heap h_2, der die Postcondition B erfüllt.
- Es muss nicht $h_0 = h_0$ gelten, Ownership-Transfer in lokalen Heap
- Es kann disjunkt shared Heaps geben, die von der Relation unverändert bleib

→ Framing, kompakte Relationen.

· Existenzquantor $\exists \bar{v}$ erlaubt A und B Zugriff auf gemeinsame logische Variablen.

Beispiele:

(1) $M, N.\ x \mapsto M \rightsquigarrow x \mapsto N \wedge N \neq M$ (Increment)

Ändere den Wert in Heapzelle x von M auf N.
Der neue Wert ist dabei nicht kleiner als der alte Wert.
Beachte, dass der Scope der existentiell quantifizierten Variablen M und N über Pre- und Postcondition reicht.

(2) $x \mapsto 0 * \mathit{lock}(y) \rightsquigarrow x \mapsto 1$ (Acquire)

Setze das Lock von 0 auf 1.
Entferne $\mathit{lock}(y)$ aus dem shared State.
Intuitiv wird $\mathit{lock}(y)$ in den lokalen State übergehen.

(3) $x \mapsto 1 \rightsquigarrow x \mapsto 0 * \mathit{lock}(y)$ (Release)

Ändere das Lock zurück von 1 auf 0.
Bringe die gehaltene Lock zurück in den shared State.

In RGSep sind Rely und Guarantee Mengen von Aktionen.
Dem Semantik ist:

- die reflexive, transitive Abschluss
- die Vereinigung der Semantiken.

Es können also die Aktionen wiederholt und gemischt ausgeführt werden.

Definition:

$$[\![\exists \bar{x}_1.\ A_1 \rightsquigarrow B_1, \ldots, \exists \bar{x}_n.\ A_n \rightsquigarrow B_n]\!] := \left(\bigcup_{i=1}^{n} [\![\exists \bar{x}_i.\ A_i \rightsquigarrow B_i]\!] \right)^*$$

Eine Aktion $\bar{x}.\, A \rightsquigarrow B$ ist *erlaubt* von einer Garantie G, wenn ihr Effekt von G abgedeckt ist.

Definition:

$(\bar{x}.\, A \rightsquigarrow B) \subseteq G$, falls $[\![\bar{x}.\, A \rightsquigarrow B]\!] \subseteq [\![G]\!]$.

9.4 Stabilität

Erinnerung:
- Eine Assertion A heißt *stabil* unter einer Relation R, falls die Relation die Assertion nicht zerstören kann.
- Rely-Guarantee arbeitet mit Assertions, die stabil unter dem Rely sind.

Ziele
- Definiere Stabilität.
- Entwickle ein Verfahren zur Prüfung von Stabilität.

Definition:

$\mathrm{sem\_stable}(A, R)$, falls

$$\forall s, h, h'.\ [\![A]\!]_{SL}(s,h) \wedge (s,h,h') \in R \Rightarrow [\![A]\!](s,h').$$

Indem wir Interferenzen R als Aktion darstellt haben, können wir Stabilität *syntaktisch* prüfen.

Dazu brauchen wir einen neuen Operator.

Definition:

Septraktion: $A \mathrel{-\!\!\circledast} B := \neg(A \mathrel{-\!\!*} \neg B)$.

B ohne A, man kann etwas aus A herausnehmen, um B zu erhalten.

formal, $[\![A \mathrel{-\!\!\circledast} B]\!]\, s\, h$ gdw. $\exists h_1.\ [\![A]\!]\, s\, h_1 \wedge [\![B]\!]\, s\, (h \uplus h_1)$.

Septachin wirkt zusammen mit

Definition:

$A \downarrow x := A \wedge \nexists v.\ x \hookrightarrow v$. "$A \wedge x$ ist voll alloziiert"

Das folgende Lemma liefert Rechenregeln zur Elimination von $-\!\circledast$.

Lemma:

- $x \mapsto y \mathrel{-\!\circledast} z \mapsto w \iff emp \wedge x = z \wedge y = w.$
- $x \mapsto y \mathrel{-\!\circledast} (A * B) \iff (x \mapsto y \mathrel{-\!\circledast} A) * B\downarrow x \vee (x \mapsto y \mathrel{-\!\circledast} B) * A\downarrow x.$
- $(A * B) \mathrel{-\!\circledast} C \iff A \mathrel{-\!\circledast} (B \mathrel{-\!\circledast} C).$
- $A \mathrel{-\!\circledast} (C \vee B) \iff (A \mathrel{-\!\circledast} C) \vee (A \mathrel{-\!\circledast} B)$
- $A \mathrel{-\!\circledast} \exists x. B \iff \exists x. (A \mathrel{-\!\circledast} B).$

Im Allgemeinen gilt nicht:

$$A \mathrel{-\!\circledast} (A * B) \iff B.$$

Spezialfälle gelten:

$$x \mapsto y \mathrel{-\!\circledast} (x \mapsto y * B) \iff B\downarrow x.$$

Regeln zu Elimination von $\downarrow x$.

Lemma:

- $x \mapsto y \downarrow z \iff x \mapsto y \wedge x \neq z.$
- $(A * B)\downarrow z \iff A\downarrow z * B\downarrow z$
- $(A \hat{\vee} B)\downarrow z \iff A\downarrow z \hat{\vee} B\downarrow z$
- $A\downarrow x \downarrow y \iff A\downarrow y \downarrow x$
- $emp \downarrow x \iff emp.$