Nachtrag:

$Configs := \mathbb{U} \times \underbrace{Stacks \times Heap}_{Zustand}$

5.2 Assertions

Ziel: Wie in Hoare-Logik sollen Assertions Zustandsmengen beschreiben.
Nur bestehen die Zustände jetzt aus Stack und Heap

Ansatz: Neue Operatoren

emp:	Der Heap ist leer	(Empty-Heap)
$a_1 \mapsto a_2$:	Der Heap besteht aus einer Zelle mit Adresse a_1 und Inhalt a_2.	(Singleton-Heap / points-to)
$\Pi_1 * \Pi_2$:	Der Heap kann in zwei disjunkte Teile zerlegt werden, die Π_1 bzw. Π_2 erfüllen	(Sep.-Conj.)
$\Pi_1 -\!\!* \Pi_2$:	Wenn der aktuelle Heap um einen disjunkten Teil erweitert wird, auf dem Π_1 gilt, gilt für den resultierenden gesamten Heap Π_2	(Sep.-Impl.)

Definition:

Die Syntax von Separation-Logik-Assertions ist

$\Pi ::= 0 \mid 1 \mid a_1 = a_2 \mid a_1 > a_2 \mid \neg\Pi \mid \Pi_1 \wedge \Pi_2 \mid \exists x.\, \Pi$ // wie bisher

$\mid emp \mid a_1 \mapsto a_2 \mid \Pi_1 * \Pi_2 \mid \Pi_1 -\!\!* \Pi_2 \mid \circledast_{i \in I} \Pi_i$

Iterated Sep.-Conj.

Abkürzungen:

- Logische Verknüpfungen: $\vee, \rightarrow, \leftrightarrow, \forall$
- $a \mapsto \_ \;:=\; \exists x.\, a \mapsto x$, mit $x \notin fv(a)$

// Adresse a ist aktiv und der Heap besteht nur aus a.

- $a_1 \hookrightarrow a_2 \;:=\; a_1 \mapsto a_2 * \text{true}$

  // a_1 points-to a_2 irgendwo im Heap

- $a \mapsto a_1, \ldots, a_n \;:=\; a \mapsto a_1 * \ldots * a+n-1 \mapsto a_n$

  // a zeigt auf ein Struct mit n Feldern

$a \hookrightarrow a_1, \ldots, a_n \;:=\; a \mapsto a_1, \ldots, a_n * \text{true}$.

Beobachtung:

$a_1 \mapsto a_2$, $a_1 \mapsto \_$ und $a \mapsto a_1, \ldots, a_n$
beschreiben die Domäne des Heaps genau.
Bei $a_1 \hookrightarrow a_2$ und $a \hookrightarrow a_1, \ldots, a_n$ gilt das nicht.
Wir kommen darauf zurück (Precise-Assertions).

Beispiele:

1) $x \mapsto ?, y$ beschreibt $x \to$ 

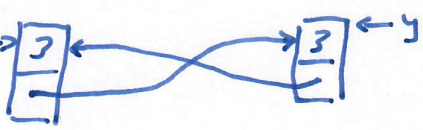
, wobei adr die Adr. in y ist.

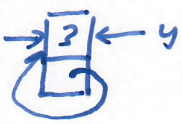
Einfach: x zeigt auf adjazente Heapknoten mit den Werten ? und y.

Genauer: Der Stack mappt x auf eine Adresse α im Heap. In α und $\alpha+1$ stehen die Werte ? und die aktuelle (Stack-)Adr von y.

2) $y \mapsto ?, x$ beschreibt $y \to$ 

. wobei adr die Adr in x ist.

3) $x \mapsto ?, y * y \mapsto ?, x$ beschreibt 

4) $x \mapsto ?, y \wedge y \mapsto ?, x$ beschreibt 

Beachte, dass die Formel nur gelten kann, wenn x und y dieselbe Adr haben.

5) $x \hookrightarrow 3,y \wedge y \hookrightarrow 3,x$ sagt

3) oder 4) gilt und es gibt
potentiell weitere Zellen im Heap.

Beispiel (Separating Implication):

- Die Assertion $x \mapsto 3,4$ beschreibt,
was (im Heap) vorhanden ist:

  Stack: $x : \alpha$

  Heap: $\alpha : 3,\ \alpha+1 : 4$

- Die Assertion

  $x \mapsto 3,4 \quad -\!\!* \ \Pi$

  beschreibt, was fehlt:
  Füge dem aktuellen Heap einen Teil wie oben hinzu
  und erhalte Π.

- Dann sieht

  $x \mapsto \_,\_ \ * \ (x \mapsto 3,4 \ -\!\!* \ \Pi)$

  wie eine schwächste Vorbedingung aus:

  $\{ x \mapsto \_,\_ * (x \mapsto 3,4 \ -\!\!* \ \Pi) \}\ [x] := 3; [x+1] := 4 \ \{\Pi\}$

  → Wenn man auf x resp. $x+1$ zugreifen darf
  → und uns nur $x \mapsto 3,4$ fehlt, um Π zu erhalten
  dann können wir die entsprechenden Zuweisungen ausführen
  und bekommen tatsächlich Π.

Das war bisher Intuition.
Zur Definition der Semantik: $h_1 \perp h_2 :=$ dom$(h_1) \cap$ dom$(h_2) = \emptyset$.

schreibe $h_1 \oplus h_2$ für die disjunkte Vereinigung.

Definition:

Die Semantik von Separation-Logik-Aussagen ist eine Funktion

$$[\![A]\!] : \text{Stacks} \times \text{Heaps} \to \mathbb{B},$$

die induktiv über die Struktur von Aussagen definiert ist:

$$[\![\text{emp}]\!]\, s\, h := \text{dom}(h) = \emptyset$$

$$[\![a_1 \mapsto a_2]\!]\, s\, h := \text{dom}(h) = \{[\![a_1]\!] s\} \quad // \text{Ein Element,}$$
$$\wedge\ h([\![a_1]\!] s) = [\![a_2]\!] s \quad \text{Auswertung auf Stack}$$

$$[\![A_1 * A_2]\!]\, s\, h := \exists h_1, h_2.\ h = h_1 \uplus h_2$$
$$\wedge\ [\![A_1]\!]\, s\, h_1 \wedge [\![A_2]\!]\, s\, h_2$$

$$[\![A_1 \mathrel{-\!\!*} A_2]\!]\, s\, h := \forall h_1.\ h_1 \perp h \wedge [\![A_1]\!]\, s\, h_1$$
$$\Rightarrow [\![A_2]\!]\, s\, h \uplus h_1.$$

$$[\![\circledast_{i \in I} A_i]\!]\, s\, h := \exists H : I \to \text{Heaps}.\ h = \biguplus_{i \in I} H(i)$$
$$\wedge\ \forall i.\ [\![A_i]\!]\, s\, H(i)$$

wie bisher:

$$[\![0/1]\!]\, s\, h := 0/1$$

$$[\![a_1 \leqslant a_2]\!]\, s\, h := [\![a_1]\!] s \leqslant [\![a_2]\!] s \quad // \text{Unabhängig von } h$$

$$[\![\neg A]\!]\, s\, h := 1 - [\![A]\!]\, s\, h$$

$$[\![A_1 \wedge A_2]\!]\, s\, h := [\![A_1]\!]\, s\, h \wedge [\![A_2]\!]\, s\, h$$

$$[\![\exists x.\ A]\!]\, s\, h := \exists v \in \mathbb{Z}.\ [\![A]\!]\, s[x \mapsto v]\, h.$$