Separation Logic: A Hoare Logic for Pointer Programs

Tjark Weber

webertj@in.tum.de

Technische Universität München

Overview

- The Language
- Natural Semantics
- Separation Logic
- Hoare Logic
- Example: In-Place List Reversal

Stores, Heaps, States

typedecl loc

types

$$
\begin{aligned}
\text{val} &= \text{nat} \\
\text{addr} &= \text{nat} \\
\text{store} &= \text{loc} \Rightarrow \text{val} \\
\text{heap} &= \text{addr} \rightsquigarrow \text{val} \\
\text{state} &= (\text{store} \times \text{heap})\ \text{option} \\
\text{aexp} &= \text{store} \Rightarrow \text{val} \\
\text{bexp} &= \text{store} \Rightarrow \text{bool}
\end{aligned}
$$

The Language: IMP ...

SKIP	skip
Assign loc aexp	x :== a
Semi com com	c1; c2
Cond bexp com com	if b then c1 else c2
While bexp com	while b do c

… with Pointers

- ListAlloc loc aexplist — x :== list as
- ArrAlloc loc aexp — x :== alloc n
- Lookup loc aexp — x :== @a
- Mutate aexp aexp — @a :== v
- Dealloc aexp — dispose a

Natural Semantics: ListAlloc

- ListAlloc:

$$[\![\mathit{heap\text{-}isfree}\ h\ a\ (\mathit{length}\ \mathit{as});\ \mathit{vs} = \mathit{map}\ (\lambda e.\ e\ s)\ \mathit{as}]\!]$$
$$\Longrightarrow \langle x :== \mathrm{list}\ \mathit{as}, \mathit{Some}\ (s, h)\rangle$$
$$\longrightarrow_c \mathit{Some}\ (s[x \mapsto a], \mathit{heap\text{-}update}\ h\ a\ \mathit{vs})$$

- ListAllocFail:

$$\forall\, a.\ \neg\ \mathit{heap\text{-}isfree}\ h\ a\ (\mathit{length}\ \mathit{as}) \Longrightarrow$$
$$\langle x :== \mathrm{list}\ \mathit{as}, \mathit{Some}\ (s, h)\rangle \longrightarrow_c \mathit{None}$$

Natural Semantics: ArrAlloc

- ArrAlloc:

$$[\![\mathit{heap\text{-}isfree}\ h\ a\ (n\ s);\ \mathit{length}\ \mathit{vs} = n\ s]\!]$$
$$\Longrightarrow \langle x :== \text{alloc}\ n, \mathit{Some}\ (s, h)\rangle$$
$$\longrightarrow_c \mathit{Some}\ (s[x \mapsto a], \mathit{heap\text{-}update}\ h\ a\ \mathit{vs})$$

- ArrAllocFail:

$$\forall\, a.\ \neg\ \mathit{heap\text{-}isfree}\ h\ a\ (n\ s) \Longrightarrow$$
$$\langle x :== \text{alloc}\ n, \mathit{Some}\ (s, h)\rangle \longrightarrow_c \mathit{None}$$

Natural Semantics: Lookup

- Lookup:

$$a\ s \in dom\ h \Longrightarrow$$
$$\langle x :== @a, Some\ (s, h)\rangle$$
$$\longrightarrow_c Some\ (s[x \mapsto heap\text{-}lookup\ h\ (a\ s)], h)$$

- LookupFail:

$$a\ s \notin dom\ h \Longrightarrow \langle x :== @a, Some\ (s, h)\rangle \longrightarrow_c None$$

Natural Semantics: Mutate

- Mutate:

$$a\ s \in dom\ h \Longrightarrow$$
$$\langle @a :== v, Some\ (s, h)\rangle$$
$$\longrightarrow_c Some\ (s, heap\text{-}update\ h\ (a\ s)\ [v\ s])$$

- MutateFail:

$$a\ s \notin dom\ h \Longrightarrow \langle @a :== v, Some\ (s, h)\rangle \longrightarrow_c None$$

Natural Semantics: Dealloc

- Dealloc:

$$a\ s \in dom\ h \Longrightarrow$$
$$\langle \text{dispose } a, Some\ (s, h)\rangle \longrightarrow_c Some\ (s, heap\text{-}remove\ h\ (a\ s))$$

- DeallocFail:

$$a\ s \notin dom\ h \Longrightarrow \langle \text{dispose } a, Some\ (s, h)\rangle \longrightarrow_c None$$

Separation Logic

- $\wedge, \vee, \neg, \longrightarrow, \ldots$
- Separating conjunction:

$$(P \wedge* Q)\ h \equiv \exists\, h'\, h''.\ h' \bowtie h'' \wedge h' +\!\!+ h'' = h \wedge P\ h' \wedge Q\ h''$$

- Separating implication:

$$(P \longrightarrow* Q)\ h \equiv \forall\, h'.\ h' \bowtie h \wedge P\ h' \longrightarrow Q\ (h +\!\!+ h')$$

Assertions

- $\mathit{emp}\ h \equiv \mathit{dom}\ h = \{\}$
- $(a \mapsto v)\ h \equiv \mathit{dom}\ h = \{a\} \wedge \mathit{heap\text{-}lookup}\ h\ a = v$
- $(a \mapsto -)\ h \equiv \exists\, v.\ (a \mapsto v)\ h$
- $a \hookrightarrow v \equiv a \mapsto v \wedge\!* \ \mathit{true}$

Properties of $\wedge*$

- $P \wedge* (Q \wedge* R) = P \wedge* Q \wedge* R$
- $P \wedge* Q = Q \wedge* P$
- $emp \wedge* P = P$
- $P \wedge* emp = P$

Properties of $\wedge*$, cntd.

- $((\lambda h.\, P\, h \vee Q\, h) \wedge*\, R)\, h = (P \wedge*\, R)\, h \vee (Q \wedge*\, R)\, h$
- $((\lambda h.\, P\, h \wedge Q\, h) \wedge*\, R)\, h \longrightarrow (P \wedge*\, R)\, h \wedge (Q \wedge*\, R)\, h$
- $((\lambda h.\, \exists x.\, P\, x\, h) \wedge*\, Q)\, h = (\exists x.\, (P\, x \wedge*\, Q)\, h)$
- $((\lambda h.\, \forall x.\, P\, x\, h) \wedge*\, Q)\, h \longrightarrow (\forall x.\, (P\, x \wedge*\, Q)\, h)$

Pure Assertions

- $\mathit{pure}\ P \equiv \forall\, h\ h'.\ P\ h = P\ h'$
- $[\![\mathit{pure}\ P;\ \mathit{pure}\ Q]\!] \Longrightarrow \big(P\ h \wedge Q\ h\big) = \big(P \wedge\!* \ Q\big)\ h$
- $[\![\mathit{pure}\ P;\ \mathit{pure}\ Q]\!] \Longrightarrow \big(P \longrightarrow\!* \ Q\big)\ h = \big(P\ h \longrightarrow Q\ h\big)$

Intuitionistic Assertions

- $\mathit{intuitionistic}\ P \equiv \forall\, h\ h'.\ P\ h \longrightarrow h \subseteq_m h' \longrightarrow P\ h'$
- $\mathit{intuitionistic}\ P \Longrightarrow \big(P \wedge\!* \ \mathit{true}\big)\ h \longrightarrow P\ h$
- $\mathit{intuitionistic}\ P \Longrightarrow P\ h \longrightarrow \big(\mathit{true} \longrightarrow\!*\ P\big)\ h$

Domain-Exact Assertions

- $\textit{domain-exact}\ P \equiv \forall\, h\, h'.\ P\, h \longrightarrow P\, h' \longrightarrow \textit{dom}\ h = \textit{dom}\ h'$

- $\textit{domain-exact}\ R \Longrightarrow$
  $\big(P \wedge\!* R\big)\, h \wedge \big(Q \wedge\!* R\big)\, h \longrightarrow \big(\big(\lambda h.\ P\, h \wedge Q\, h\big) \wedge\!* R\big)\, h$

- $\textit{domain-exact}\ Q \Longrightarrow$
  $\big(\forall\, x.\ \big(P\, x \wedge\!* Q\big)\, h\big) \longrightarrow \big(\big(\lambda h.\ \forall\, x.\ P\, x\, h\big) \wedge\!* Q\big)\, h$

Hoare Logic

- $\models \{P\}\ c\ \{Q\} \equiv$
  $\forall s\, h.\ (P\, s\, h \longrightarrow (Some\ (s, h), None) \notin C\, c) \wedge$
  $(\forall s'\, h'.\ (Some\ (s, h), Some\ (s', h')) \in C\, c \longrightarrow P\, s\, h$
  $\longrightarrow Q\, s'\, h')$
  - No error state reachable
  - Partial correctness

Hoare Rules

- ListAlloc:

$$\vdash \{\lambda s\, h.\ (\exists\, a.\ \textit{heap-isfree}\ h\ a\ (\textit{length as})) \wedge (\forall\, a.\ (a[\mapsto]\textit{map}\ (\lambda e.\ e\ s)\ \textit{as} \longrightarrow\!* P\ (s[x \mapsto a]))\ h)\}$$
$$x :== \text{list}\ \textit{as}\ \{P\}$$

- ArrAlloc:

$$\vdash \{\lambda s\, h.\ (\exists\, a.\ \textit{heap-isfree}\ h\ a\ (n\ s)) \wedge (\forall\, a\ \textit{vs}.\ \textit{length vs} = n\ s \longrightarrow (a[\mapsto]\textit{vs} \longrightarrow\!* P\ (s[x \mapsto a]))\ h)\}$$
$$x :== \text{alloc}\ n\ \{P\}$$

Hoare Rules, cntd.

- Lookup:

$$\vdash \{\lambda s\, h.\ \exists\, v.\ (a\ s \hookrightarrow v)\ h \wedge P\ (s[x \mapsto v])\ h\}\ x :== @a\ \{P\}$$

- Mutate:

$$\vdash \{\lambda s.\ a\ s \mapsto -\ \wedge *\ (a\ s \mapsto v\ s \longrightarrow *\ P\ s)\}\ @a :== v\ \{P\}$$

- Dealloc:

$$\vdash \{\lambda s.\ a\ s \mapsto -\ \wedge *\ P\ s\}\ \text{dispose}\ a\ \{P\}$$

Soundness and Completeness

- Soundness:

$$\vdash \{P\}\ c\ \{Q\} \Longrightarrow \models \{P\}\ c\ \{Q\}$$

- Relative completeness:

$$\models \{P\}\ c\ \{Q\} \Longrightarrow \vdash \{P\}\ c\ \{Q\}$$

  - Weakest precondition:

$$\vdash \{wp\ c\ Q\}\ c\ \{Q\}$$

Example: In-place List Reversal

$$
\begin{array}{l}
\text{reverse} :: \text{loc} \Rightarrow \text{loc} \Rightarrow \text{loc} \Rightarrow \text{com} \\
\text{reverse i j k} == (\text{j} :== (\lambda \text{s. } 0)); \\
\qquad\qquad \text{while } (\lambda \text{s. s i} \neq 0) \text{ do} \\
\qquad\qquad (\\
\qquad\qquad\quad (((\text{k} :== @(\lambda \text{s. Suc (s i)})); \\
\qquad\qquad\quad (@(\lambda \text{s. Suc (s i)}) :== (\lambda \text{s. s j}))); \\
\qquad\qquad\quad (\text{j} :== (\lambda \text{s. s i}))); \\
\qquad\qquad\quad (\text{i} :== (\lambda \text{s. s k})) \\
\qquad\qquad)
\end{array}
$$

In-place List Reversal, cntd.

theorem reverse-correct:

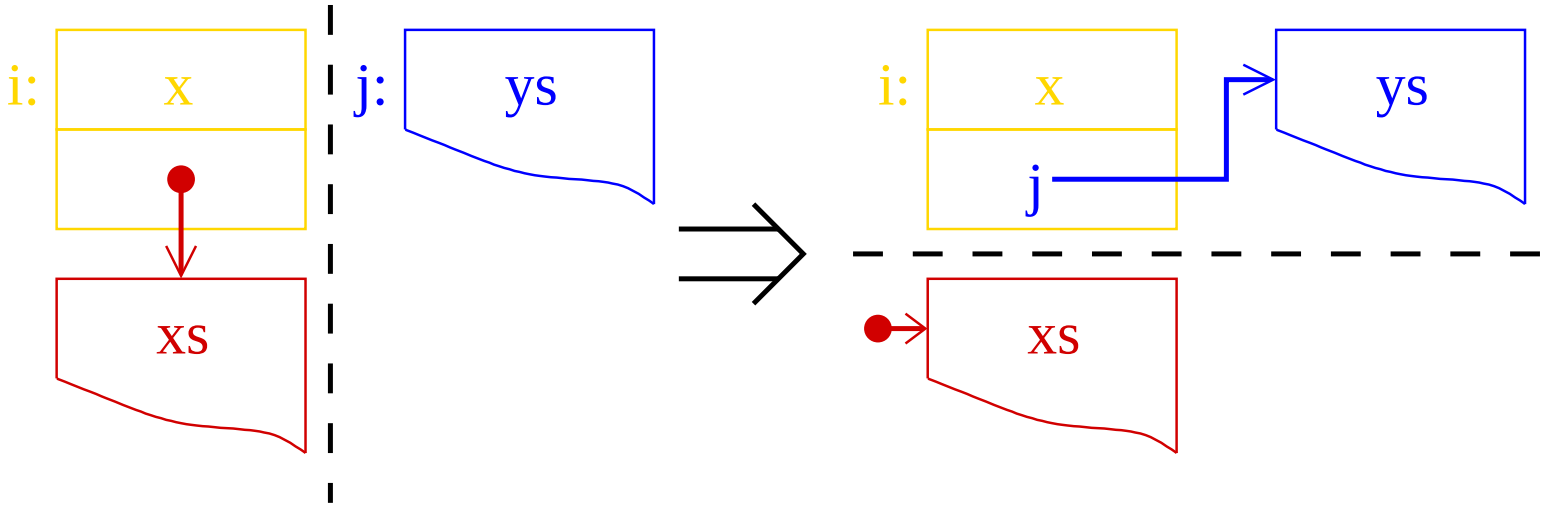
$\models \{\ \lambda$s h. heap-list vs (s i) h $\wedge$ i$\neq$j $\wedge$ i$\neq$k $\wedge$ j$\neq$k $\}$
reverse i j k
$\{\ \lambda$s h. heap-list (rev vs) (s j) h $\}$

- Loop invariant:

$$\lambda s\ h.\ (\exists\, xs\ ys.$$
$$(\textit{heap-list}\ xs\ (s\ i) \wedge\!* \textit{heap-list}\ ys\ (s\ j))\ h\ \wedge$$
$$\textit{rev}\ vs = \textit{rev}\ xs\ @\ ys)\ \wedge$$
$$i \neq j \wedge i \neq k \wedge j \neq k$$

List Reversal: The Proof

- $(\textit{heap-list}\ ys\ j \wedge\!*\ \textit{heap-list}\ (x \# xs)\ i)\ h \Longrightarrow$
  $(\textit{heap-list}\ xs\ (\textit{heap-lookup}\ h\ (\textit{Suc}\ i)) \wedge\!*\ \textit{heap-list}\ (x \# ys)\ i)$
  $(\textit{heap-update}\ h\ (\textit{Suc}\ i)\ [j])$



Conclusions and Future Work

- No complete axiom scheme for $\wedge *$ and $\longrightarrow *$
- Frame rule
- Syntactic sugar
- Annotations, Verification Condition Generator
- Total correctness, procedures, concurrency, . . .

Discussion

?*