A Theory of Substitutions for Separation Logic

Frank de Boer

Trigger

Assignment axiom in Hoare logic:

$$\{p[x := e]\}\ x := e\ \{p\}$$

Reynolds:

> *Expressions do not contain notations, ..., that refer to the heap. It follows that none of the new heap manipulating instructions are instances of the simple assignment instruction. In fact, they will not obey Hoare's inference rule for assignment.*

Contribution

A theory of substitutions for separation logic which

- describes the effect of the basic heap manipulating instructions compositionally in terms of the logical structure of the given pre/postcondition,
- provides a direct account of aliasing in terms of basic equational predicate logic, and
- does not generate additional complexity measured by the maximal depth of nested separating conjunction/implication connectives, e.g., as in the *backwards mutation axiom*

$$\{(\exists y(x \mapsto y)) * ((x \mapsto e) \mathrel{-\!\!*} p)\}\ [x] := e\ \{p\}$$

The Basic Instructions

Basic assignment $\langle x := e, h, s\rangle \Rightarrow (h, s[x := s(e)])$,
Look-up $\langle x := [e], h, s\rangle \Rightarrow (h, s[x := h(s(e))])$ if $s(e) \in dom(h)$,
$\langle x := [e], h, s\rangle \Rightarrow$ **fail** if $s(e) \notin dom(h)$,
Mutation $\langle [x] := e, h, s\rangle \Rightarrow (h[s(x) := s(e)], s)$ if $s(x) \in dom(h)$,
$\langle [x] := e, h, s\rangle \Rightarrow$ **fail** if $s(x) \notin dom(h)$,
Allocation $\langle x := \mathbf{new}(e), h, s\rangle \Rightarrow (h[n := s(e)], s[x := n])$ if
$n \notin dom(h)$.
Dispose $\langle [x] := \bot, h, s\rangle \Rightarrow (h[s(x) := \bot], s)$ if $s(x) \in dom(h)$,
$\langle [x] := \bot, h, s\rangle \Rightarrow$ **fail** if $s(x) \notin dom(h)$

Separation Logic

$h, s \models b$ iff $s(b) = \mathbf{true}$,
$h, s \models \mathbf{emp}$ iff $dom(h) = \emptyset$,
$h, s \models (e \mapsto e')$ iff $dom(h) = \{s(e)\}$ and $h(s(e)) = s(e')$,
$h, s \models (e \hookrightarrow e')$ iff $s(e) \in dom(h)$ and $h(s(e)) = s(e')$,
$h, s \models (p \wedge q)$ iff $h, s \models p$ and $h, s \models q$,
$h, s \models (p \vee q)$ iff $h, s \models p$ or $h, s \models q$,
$h, s \models (p \rightarrow q)$ iff $h, s \models p$ implies $h, s \models q$,
$h, s \models \exists x p$ iff $h, s[x := n] \models p$ for some n,
$h, s \models \forall x p$ iff $h, s[x := n] \models p$ for all n,
$h, s \models (p * q)$ iff $h_1, s \models p$ and $h_2, s \models q$ for some *partition*
$h = h_1 \uplus h_2$,
$h, s \models (p \mathrel{-\!\!*} q)$ iff for all h' *disjoint* from h: $h', s \models p$ implies
$h \uplus h', s \models q$.

Weakest Precondition Calculus

Basic assignment $\{p[x := e]\}\ x := e\ \{p\}$
Look-up $\{\exists y((e \hookrightarrow y) \wedge p[x := y])\}\ x := [e]\ \{p\}$
where y is fresh
Mutation $\{(x \hookrightarrow -) \wedge p[\langle x \rangle := e]\}\ [x] := e\ \{p\}$
Allocation $\{\forall x((x \not\hookrightarrow -) \rightarrow p[\langle x \rangle := e])\}\ x := \mathbf{new}(e)\ \{p\}$
Dispose $\{(x \hookrightarrow -) \wedge p[\langle x \rangle := \bot]\}\ [x] := \bot\ \{p\}$

VS ("OLD SCHOOL"):

Mutation $\{(x \mapsto -) * ((x \mapsto e) \mathrel{-\!\!*} p)\}\ [x] := e\ \{p\}$
Allocation $\{\forall x((x \mapsto e) \mathrel{-\!\!*} p)\}\ x := \mathbf{new}(e)\ \{p\}$
Dispose $\{(x \hookrightarrow -) * p\}\ [x] := \bot\ \{p\}$

Substitution Heap Update

- $b[\langle x \rangle := e] = b$,
- $(e' \hookrightarrow e'')[\langle x \rangle := e] = (x = e' \wedge e'' = e) \vee (x \neq e' \wedge e' \hookrightarrow e'')$,
- $(p \wedge q)[\langle x \rangle := e] = p[\langle x \rangle := e] \wedge q[\langle x \rangle := e]$, and similar for $\vee$ and $\rightarrow$,
- $(\exists y p)[\langle x \rangle := e] = \exists y (p[\langle x \rangle := e])$ and similar for $\forall$,
- $(p * q)[\langle x \rangle := e] = (p[\langle x \rangle := e] * q') \vee (p' * q[\langle x \rangle := e])$ where $p' = p \wedge (x \not\hookrightarrow -)$ and similarly $q' = q \wedge (x \not\hookrightarrow -)$,
- $(p \mathrel{-\!\!*} q)[\langle x \rangle := e] = p' \mathrel{-\!\!*} q[\langle x \rangle := e]$ where as above $p' = p \wedge (x \not\hookrightarrow -)$.

Lemma (heap update substitution lemma)

$h, s \models p[\langle x \rangle := e]$ *iff* $h[s(x) := s(e)], s \models p$.

Example

$$\{(x \hookrightarrow -) \wedge ((y = x \wedge 1 = 0) \vee (y \neq x \wedge y \hookrightarrow 1))\}\ [x] := 0\ \{y \hookrightarrow 1\}$$

Compare this with the standard backwards rule:

$$\{x \mapsto - * (x \mapsto 0 \mathrel{-\!\!*} y \hookrightarrow 1)\}\ [x] := 0\ \{y \hookrightarrow 1\}$$

Substitution Heap Clear

- $b[\langle x \rangle := \bot] = b$
- $\mathbf{emp}[\langle x \rangle := \bot] = \mathbf{emp} \vee x \mapsto -$
- $(e \mapsto e')[\langle x \rangle := \bot] =$
  $e \hookrightarrow e' \wedge x \neq e \wedge \forall y((y \hookrightarrow -) \rightarrow (y = e \vee y = x))$
- $(e \hookrightarrow e')[\langle x \rangle := \bot] = x \neq e \wedge e \hookrightarrow e'$
- $(p \wedge q)[\langle x \rangle := \bot] = p[\langle x \rangle := \bot] \wedge q[\langle x \rangle := \bot]$, and similar for $\vee$ and $\rightarrow$,
- $(\exists y p)[\langle x \rangle := \bot] = \exists y(p[\langle x \rangle := \bot])$
- $(p * q)[\langle x \rangle := \bot] = (p[\langle x \rangle := \bot]) * (q[\langle x \rangle := \bot])$
- $(p \mathrel{-\!\!*} q)[\langle x \rangle := \bot] =$
  $((p \wedge x \not\hookrightarrow -) \mathrel{-\!\!*} q[\langle x \rangle := \bot]) \wedge \forall y(p[\langle x \rangle := y] \mathrel{-\!\!*} q[\langle x \rangle := y])$
  where y is a fresh variable.

Lemma (heap clear substitution lemma)

$h, s \models p[\langle x \rangle := \bot]$ *iff* $h[s(x) := \bot], s \models p$.

Soundness and Completeness WP calculus

Theorem

For any basic instruction S, we have
$\models \{p\}\ S\ \{q\}$ *if and only if* $\{p\}\ S\ \{q\}$ *is derivable from the above axioms and (a single application of) the rule of consequence.*

Strongest Postcondition Calculus

Basic assignment $\{p\}\ x := e\ \{\exists y(p[x := y] \wedge (e[x := y] = x))\}$
Look-up
$\{p \wedge e \hookrightarrow -\}\ x := [e]\ \{\exists y(p[x := y] \wedge (x \hookrightarrow e[x := y]))\}$
Mutation $\{p \wedge x \hookrightarrow -\}\ [x] := e\ \{\exists y(p[\langle x \rangle := y]) \wedge x \hookrightarrow e\}$
Allocation
$\{p\}\ x := \mathbf{new}(e)\ \{(\exists y(p[x := y]))[\langle x \rangle := \bot] \wedge x \hookrightarrow e\}$

Dispose $\{p \wedge x \hookrightarrow -\}\ [x] := \bot\ \{\exists y(p[\langle x \rangle := y]) \wedge x \not\hookrightarrow -\}$

Conclusion

Substitutions describe the effect of the instructions in terms of the logical structure of the given pre/postcondition,

whereas

in the standard approach heap operations are used to describe the effect of the instructions, abstracting from the logical structure of the given pre/postcondition.

And What About ...

The Frame Rule

$$\frac{\{p\}\ S\ \{q\}}{\{p * r\}\ S\ \{q * r\}}$$

- Not needed for completeness while programs.
- Completeness recursive procedures.
  See *Completeness for recursive procedures in separation logic* by Mahmudul Faisal Al Ameen and Makoto Tatsuta, TCS, 2016: Also does not use the frame rule!
- Modular completeness.
  Frame rule applied in thesis *Local Reasoning for Stateful Programs* by Hongseok Yang, 2001.