Introduction to Separation Logic

Heavily borrows from slides by Cristiano Calcagno, Imperial College London

October 15, 2019

Table of Contents

Table of Contents

Syntax of Separation Logic

- Given a decidable base-theory T, the syntax of separation logic $SL(T)_{Loc,Data}$ is presented
- Loc and $Data$ represent the type of the address and the values
- E.g Setting Loc and $Data$ to be Int, then our addresses and values are integers

$$\begin{aligned} P, Q ::= \; & \mathit{false} \mid P \wedge Q \mid P \vee Q \mid P \rightarrow Q \\ & \mid P * Q \mid P \mathrel{-\!\!*} Q \\ & \mid E = E' \mid E \hookrightarrow E' \mid \mathit{empty} \end{aligned}$$

We use E and E' to denote expressions in the base theory, where pointer indirection is not used.

[Srivas: What do you mean by "pointer indirection not used"? Do you mean no dereferencing? Why not? It is strange to have Loc to be integers, but, I guess, it's OK; the paper has the restriction that Loc domain has to be countably infinite for obvious reasons]

Semantics of Separation Logic

The model consists of an interpretation (I) and a heap (h)

$$I : \mathrm{Var} \to \mathrm{Loc}$$
$$h : \mathrm{Loc} \to \mathrm{Data}$$

$$\begin{aligned}
I, h &\models \mathit{false} && \text{never satisfied} \\
I, h &\models P \wedge Q && I, h \models P \text{ and } I, h \models Q \\
I, h &\models P \vee Q && I, h \models P \text{ or } I, h \models Q \\
I, h &\models P \to Q && I, h \models P \text{ implies } I, h \models Q \\
I, h &\models E = E' && [\![E]\!]_I = [\![E']\!]_I
\end{aligned}$$

We use $[\![E]\!]_I$, to denote the value of E under the interpretation I.

[Srivas: Must note domain of h has to be a finite subset of Loc and h can be partial]

Semantics of Separation Logic

Empty heap

$$I, h \models empty$$
$$\text{iff } h = \phi$$

Separating conjunction

$$I, h \models P * Q$$
$$\text{iff } \exists h_1, h_2.(h_1 \# h_2) \wedge (h = h1 \circ h2) \wedge I, h_1 \models P \wedge I, h_2 \models Q$$

Where $h_1 \# h_2$ denotes that the heap domains are disjoint and $h_1 \circ h_2$ means their union.

[Srivas: I have changed $\perp$ to hash symbol as used in the paper]

Semantics of Separation Logic

Separating Implication

$$
\begin{gathered}
I, h \models P \mathrel{-\!\!*} Q \\
\text{iff } \forall h'.(h \# h') \wedge (I, h' \models P) \rightarrow I, h \circ h' \models Q
\end{gathered}
$$

Interpretation : If we extend the current heap with a disjoint heap satisfying P, then the new heap satisfies Q. In some ways, we can imagine that our current heap is only missing the records of P, to make it satisfy Q.

Points to

$$
\begin{gathered}
I, h \models E \hookrightarrow E' \\
\text{iff } h(\llbracket E \rrbracket_I) = \llbracket E' \rrbracket_I
\end{gathered}
$$

Examples

Points to,

$$F : x \hookrightarrow 10$$
$$I : \{(x, 0)\}$$
$$h : \{(0, 10)\}$$
$$I, h \models F$$

Separating conjunction,

$$F : x \hookrightarrow 10 * y \hookrightarrow 20$$
$$I : \{(x, 0), (y, 1)\}$$
$$h : \{(0, 10), (1, 20)\}$$
$$I, h \models F$$

[Srivas: Also add a more interesting version of this example: x points-to y and y points x, with x and y in two disjoint partitions of heap]

Examples

Separating Implication

$$I, h \models P \mathrel{-\!\!*} Q$$
$$\text{iff } \forall h'.(h \# h') \wedge (I, h' \models P) \rightarrow I, h \circ h' \models Q$$

Example,

$$F : (x \hookrightarrow 10) \mathrel{-\!\!*} (x \hookrightarrow 10 * y \hookrightarrow 20)$$
$$I : \{(x, 0), (y, 1)\}$$
$$h' : \{(0, 10)\}$$
$$h : \{(1, 20)\}$$
$$h \circ h' : \{(0, 10), (1, 20)\}$$
$$I, h \models F$$

Table of Contents

Translating Separation Logic into Pointer Logic

Points to,

$$I, h \models x \hookrightarrow v$$
$$\iff$$
$$L, M \models \ ^*x = v$$

Separating conjunction,

$$I, h \models x \hookrightarrow v_1 * y \hookrightarrow v_2$$
$$\iff$$
$$L, M \models \ ^*x = v_1 \ \wedge \ ^*y = v_2 \wedge x \neq y$$

Table of Contents

[Srivas: You should show more examples (list would be good) to make the following points:

- How the expressiveness of * allows to you to ensure no alias without separation guarantee to state inequalities : show some example of a list where inadvertent cyclicity can be introduced without explicit inequalities to ensure no aliasing
- explicit unwound list examples can then be used to motivate need for inductive predicates
- you can then introduce inductive predicates

]

[Srivas: Will you be able to formally cover the following in this lecture itself:

- Formally introduce the satisfaction problem for quantifier=free fragment of SL w/o inductive definitions
- The labeled inference rules shown in the paper that allows you to reduce the satisfaction problem of SL into the theory of T + Universa qunt instantiation
- we can do the lazy version and extension of satisfaction procedure for inductive pred later

]