

# Introduction to Separation Logic

Heavily borrows from slides by Cristiano Calcagno, Imperial College  
London

October 15, 2019

# Table of Contents

- 1 Introducing Separation Logic
- 2 Relation with Pointer Logic
- 3 Inductive predicates

# Table of Contents

1 Introducing Separation Logic

2 Relation with Pointer Logic

3 Inductive predicates

# 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

$$\begin{aligned} P, Q ::= & \text{false} \mid P \wedge Q \mid P \vee Q \mid P \rightarrow Q \\ & \mid P * Q \mid P \multimap Q \\ & \mid E = E' \mid E \hookrightarrow E' \mid \text{empty} \end{aligned}$$

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

[Srivivas: 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 ( $h$ )

$$I : \text{Var} \rightarrow \text{Loc}$$

$$h : \text{Loc} \rightarrow \text{Data}$$

$$I, h \models \text{false} \qquad \text{never satisfied}$$

$$I, h \models P \wedge Q \qquad I, h \models P \text{ and } I, h \models Q$$

$$I, h \models P \vee Q \qquad I, h \models P \text{ or } I, h \models Q$$

$$I, h \models P \rightarrow Q \qquad I, h \models P \text{ implies } I, h \models Q$$

$$I, h \models E = E' \qquad \llbracket E \rrbracket_I = \llbracket E' \rrbracket_I$$

We use  $\llbracket E \rrbracket_I$ , to denote the value of  $E$  under the interpretation  $I$ .

[Srivvas: Must note domain of  $h$  has to be a finite subset of  $\text{Loc}$  and  $h$  can be partial]

# Semantics of Separation Logic

Empty heap

$$\begin{aligned} I, h &\models \text{empty} \\ \text{iff } h &= \phi \end{aligned}$$

Separating conjunction

$$\begin{aligned} I, h &\models P * Q \\ \text{iff } \exists h_1, h_2. &(h_1 \# h_2) \wedge (h = h_1 \circ h_2) \wedge I, h_1 \models P \wedge I, h_2 \models Q \end{aligned}$$

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]

## Separating Implication

$$\begin{aligned} I, h &\models P \multimap Q \\ \text{iff } \forall h'. (h \# h') \wedge (I, h' &\models P) \rightarrow I, h \circ h' \models Q \end{aligned}$$

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

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

# Examples

Points to,

$$F : x \hookrightarrow 10$$

$$I : \{(x, 0)\}$$

$$h : \{(0, 10)\}$$

$$I, h \models F$$

Separating conjunction,

$$F : x \hookrightarrow 10 * y \hookrightarrow 20$$

$$I : \{(x, 0), (y, 1)\}$$

$$h : \{(0, 10), (1, 20)\}$$

$$I, h \models F$$

[Srivastava: 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]



## Separating Implication

$$\begin{aligned} I, h &\models P \multimap Q \\ \text{iff } \forall h'. (h \# h') \wedge (I, h' &\models P) \rightarrow I, h \circ h' \models Q \end{aligned}$$

Example,

$$F : (x \hookrightarrow 10) \multimap (x \hookrightarrow 10 * y \hookrightarrow 20)$$

$$I : \{(x, 0), (y, 1)\}$$

$$h' : \{(0, 10)\}$$

$$h : \{(1, 20)\}$$

$$h \circ h' : \{(0, 10), (1, 20)\}$$

$$I, h \models F$$

# Table of Contents

1 Introducing Separation Logic

2 Relation with Pointer Logic

3 Inductive predicates

# Translating Separation Logic into Pointer Logic

Points to,

$$\begin{aligned} I, h &\models x \hookrightarrow v \\ &\iff \\ L, M &\models *x = v \end{aligned}$$

Separating conjunction,

$$\begin{aligned} I, h &\models x \hookrightarrow v_1 * y \hookrightarrow v_2 \\ &\iff \\ L, M &\models *x = v_1 \wedge *y = v_2 \wedge x \neq y \end{aligned}$$

# Table of Contents

1 Introducing Separation Logic

2 Relation with Pointer Logic

3 Inductive predicates

# Need for inductive predicates

- Most interesting data structures in programs are defined as inductive systems
- For example : linked lists, trees, graphs
- Being able to reason about these in SL is useful
- But inductive predicates introduce quantifiers

## Example - List

$$\text{list } 0 \ x \equiv \text{empty} \wedge x = \text{nil}$$
$$\text{list } n \ x \equiv \exists y. (x \hookrightarrow n, y) * (\text{list } (n - 1) \ y)$$

- This defines a linked-list rooted at  $x$
- Base case : empty list where heap is empty and root pointer is null
- Inductive case : the root points to a struct which has a value and the pointer for the remaining list
- Separately, the next pointer points to a list. Prevents pointer aliasing, each pointer is different

# Comparison with pointers

Begin by defining a list in pointer logic,

$$\text{list } 0 \ x \equiv x = \text{nil}$$

$$\text{list } n \ x \equiv \exists y. (x \hookrightarrow v, y) \wedge (\text{list } (n - 1) \ y)$$

This is a satisfying assignment for *list* 3 *x*,

$$l = \{ (x, 0), (y, 0), (z, 1), (w, \text{nil}) \}$$

$$h = \{ (0, (v, 1)), (1, (v, w)) \}$$

The pointers *x* and *y* got aliased to point to *z*.

$$\text{list } 3 \ x \equiv$$

$$(x = 0) \hookrightarrow v, 1 \wedge$$

$$(y = 0) \hookrightarrow v, 1 \wedge$$

$$(z = 1) \hookrightarrow v, \text{nil} \wedge$$