A relational Model for Confined Separation Logic

Palwasha Afsar
MAP-i Doctoral Student

Outline

- The Problem
- Aim of the Paper
- PF transform
- Separation Logic
- Confined Separation logic
- Reasoning
- Conclusion

The Problem

- Formal reasoning try to prove the correctness of the program by converting the program to its semantics.
- Should try to reduce the size.
- Semantics concepts of Object Oriented Programming introduce the concepts of heap, Stacks, Array etc.
- These concepts are introduced to solve the main problem of references: Aliasing

Aliasing

- Aliasing is the phenomena that two different names refer to the same storage location. By changing the value of one name, the other changes as well. So, aliasing complicates the reasoning about the correctness of a program.
- It’s a well-known problem in Object Oriented Programming.

Aims of the paper

- *Develops a relational model using confined separation logic to handle the problem of dangling references in mutable structures.*
- Separation logic is a modern system for reasoning about the correctness of a program.
- Its an extension to Hoare Logic where formulae are interpreted over suitable model of stores and heaps.

PF-Transform

- The key technique of this work is the *point-free* (PF) *transform*.
- This means to convert predicate logic formulae into binary relation by removing bound variables and quantifiers.
- This technique was introduced in the 19th century and today its known as "algebra of programming".

Summary of PF-transforms

ψ	Φ_ψ
$\langle \forall\, a,b \;::\; b\,R\,a \Rightarrow b\,S\,a \rangle$	$R \subseteq S$
$\langle \forall\, a \;::\; f\,a = g\,a \rangle$	$f \subseteq g$
$\langle \forall\, a \;::\; a\,R\,a \rangle$	$id \subseteq R$
$\langle \exists\, a \;::\; b\,R\,a \;\wedge\; a\,S\,c \rangle$	$b(R \cdot S)c$
$b\,R\,a \;\wedge\; b\,S\,a$	$b\,(R \cap S)\,a$
$b\,R\,a \vee b\,S\,a$	$b\,(R \cup S)\,a$
$(f\,b)\,R\,(g\,a)$	$b(f^\circ \cdot R \cdot g)a$
TRUE	$b \top a$
FALSE	$b \perp a$

Binary relation

- Let $B \xleftarrow{R} A$ denotes a binary relation from A(target) to B(Source).
- R ⊆ S — the obvious "R is at most S" inclusion ordering.
- R ∪ S denotes the union of two relations and ⊤ is the largest relation of its type. Its dual is ⊥, the smallest such relation (the empty one).
- b R a — "R relates b to a", that is, (b,a) ∈ R.
- id such that R · id = id · R = R
- Converse of R — R◦ such that a(R◦)b iff b R a.

Separation logic

- Syntax:

  $*$: separating conjunction

  $-*$: separating implication

- emp : The heap is empty.
- $P * Q$: The heap contains disjoint parts such that P holds in one and Q holds in the other.

- $P -* Q$: If the heap were extended with a disjoint part such that P holds, then Q holds for the new larger heap.
- $e \rightarrow e$: Singleton Heap

Confined Separation Logic

- A type is said to be confined in a domain if and only if all references to instances of that type originate from objects of the domain.
- For confinement, the paper describes the following variants of conjunction to handle the problem of dangling references.
- The *In* variant, *notIn* variant, and *inBoth* Variant.

Confined Separation LOgic

- *notIn* variant denoted by $p \neg\triangleright q$ — hold for disjoint parts of heap such that no references of the first point to the other
- *In* variant denoted by $p \triangleright q,$ — is a heap disjoint such that all references in the first do point into the other
- *inBoth* variant denoted by $p \triangleleft\triangleright q,$ — is a heap disjoint such that all references in the first are confined to both.

Generic Separation Logic

- Separation logic is typically interpreted on a storage model coupling a store σ, for variables, and a heap H for addresses.

$$\begin{tikzcd}
Variables \arrow[r, "\sigma"] \arrow[d, "Aliases = \in \cdot \sigma"'] & Atom + Address \arrow[dl, "\in"] \\
Address \arrow[r, "H"'] & Atom + Address
\end{tikzcd}$$

Separability

- First separability relation is defined on heaps:

Eq(1)

$$H \ * \ (H_1, H_2) \quad \overset{\text{def}}{=} \quad (H_1 \parallel H_2) \ \wedge \ (H = H_1 \cup H_2)$$

If they are disjoint then:

$$\neg \langle \exists\, b, a, k \ :: \ b\, H_1\, k \ \wedge \ a\, H_2\, k \rangle$$

Now lets try to solve these equations using PF-transform

Contd…

- $\neg\langle\exists b,a,k :: bH_1 k \wedge aH_2 k\rangle$
  ≡ { ∃ -nesting (Eindhoven quantifier calculus) }
- $\neg\langle\exists b,a :: \langle\exists k :: bH_1 k \wedge aH_2 k\rangle\rangle$
  ≡ { relational converse: b R∘a the same as a R b }
- $\neg\langle\ \exists\ b\ ,\ a\ ::\langle\ \exists\ k :: b\ H_1\ k\ \wedge\ k\ H_2{}^{\circ}\ a\ \rangle\ \rangle$
  ≡ { introduce relational composition }
- $\neg\langle\exists b,a :: b(H_1\cdot H_2{}^{\circ})a\rangle$
  ≡ { de Morgan ; negation }
- $\langle\forall b,a :: b(H_1\cdot H_2{}^{\circ})a \Rightarrow False\rangle$

Contd..

≡ { empty relation: b $\perp$ a is always false }

$\langle \forall b,a :: b(H_1 \cdot H_2^\circ)a \Rightarrow b \perp a \rangle$

≡ { drop points a,b }

$H_1 \cdot H_2^\circ \subseteq \perp$

- So we can redefine eq(2) $H_1 \parallel H_2 \overset{\text{def}}{=} H_1 \cdot H_2^\circ \subseteq \perp$
- In a similar way, we can write for dangling references i-e

Contd..

- For dangling references:

$$H_1 \neg\triangleright H_2 \quad \overset{\text{def}}{=} \quad H_1 \parallel H_2 \ \wedge\ H_2 \cdot \in_{\mathsf{F}} \cdot H_1 \subseteq \bot$$

- Asserts that no out going reference in H_1 goes into separated H_2. Back to pointwise notation:

$$\neg\langle \exists\, k, k' \ :\ k \in \delta\, H_1 \ \wedge\ k' \in \delta\, H_2 :\ k' \in_F (H_1\, k)\rangle$$

- Similarly for other variants:

$$H_1 \triangleright H_2 \ \overset{\text{def}}{=}\ H_1 \parallel H_2 \ \wedge\ \in_{\mathsf{F}} \cdot H_1 \subseteq H_2^{\circ} \cdot \top$$

$$H_1 \triangleleft\triangleright H_2 \ \overset{\text{def}}{=}\ H_1 \parallel H_2 \ \wedge\ \in_{\mathsf{F}} \cdot H_1 \subseteq (H_1 \cup H_2)^{\circ} \cdot \top$$

Semantics for Separational Logic

- The preorder on assertion is defined by:
- $p \rightarrow q \overset{\text{def}}{=} [\![p]\!] \subseteq [\![q]\!]$ eq(3)

so that it can be distinguished from standard logic implication ⇒.

- By recalling *composition* and *splitting*, the separating conjunction equation

$$H[\![p * q]\!]\sigma \overset{\text{def}}{=} \langle \exists\, H_0, H_1 \ :: \ H * (H_0, H_1) \ \wedge \ H_0[\![p]\!]\sigma \ \wedge \ H_1[\![q]\!]\sigma \rangle$$

Becomes: $[\![p * q]\!] \overset{\text{def}}{=} (*) \cdot \langle [\![p]\!], [\![q]\!] \rangle$ eq(4)

Contd..

- The other variants becomes:
- eq(5) $[\![p \neg\triangleright q]\!] \overset{\text{def}}{=} (*) \cdot \Phi_{\neg\triangleright} \cdot \langle [\![p]\!], [\![q]\!] \rangle$
- eq(6) $[\![p \triangleright q]\!] \overset{\text{def}}{=} (*) \cdot \Phi_{\triangleright} \cdot \langle [\![p]\!], [\![q]\!] \rangle$
- eq(7) $[\![p \triangleleft\triangleright q]\!] \overset{\text{def}}{=} (*) \cdot \Phi_{\triangleleft\triangleright} \cdot \langle [\![p]\!], [\![q]\!] \rangle$

Conjunction and Implication

- Beginning by the fact that p∗ and p−∗ constitute Galois connection. So,

$$(p * x) \rightarrow y \equiv x \rightarrow (p \mathrel{-\!\!*} y)$$

Where we know everything, apart from (p−∗) which we want to calculate:

$$\begin{array}{ll}
& [\![p * x]\!] \subseteq [\![y]\!] \\
\equiv & \qquad \{\ \text{(20) following by GC of left division}\ \} \\
& \langle [\![p]\!], [\![x]\!] \rangle \subseteq (*) \setminus [\![y]\!] \\
\equiv & \qquad \{\ (25)\ \}\ \text{two GC in a row} \\
& [\![x]\!] \subseteq [\![p]\!] \ \blacktriangleright ((*) \setminus [\![y]\!]) \\
\equiv & \qquad \{\ \text{introduce } p \mathrel{-\!\!*} y \text{ st } [\![p \mathrel{-\!\!*} y]\!] = [\![p]\!] \ \blacktriangleright ((*) \setminus [\![y]\!])\ \} \\
& [\![x]\!] \subseteq [\![p \mathrel{-\!\!*} y]\!] \\
\equiv & \qquad \{\ (19)\ \} \\
& x \rightarrow (p \mathrel{-\!\!*} y)
\end{array}$$

Contd…

- For pointwise meaning, we resort to the Eindhoven quantifier:

$$
\begin{array}{ll}
 & H[\![p \mathrel{-\!\!*} y]\!]\sigma \\
\equiv & \quad \{ \text{ above } \} \\
 & H([\![p]\!] \blacktriangleright ((*) \setminus [\![y]\!]))\sigma \\
\equiv & \quad \{ \text{ (26) } \} \\
 & \langle \forall\, H_0 \ : \ H_0[\![p]\!]\sigma : \ (H_0, H)((*) \setminus [\![y]\!])\sigma \rangle \\
\equiv & \quad \{ \text{ left division (pointwise) } \} \\
 & \langle \forall\, H_0 \ : \ H_0[\![p]\!]\sigma : \ \langle \forall\, H_1 \ : \ H_1 * (H_0, H) : \ H_1[\![y]\!])\sigma \rangle\rangle \\
\equiv & \quad \{ \text{ nesting: (4.21) of [2] } \} \\
 & \langle \forall\, H_0, H_1 \ : \ H_0[\![p]\!]\sigma \ \wedge \ H_1 * (H_0, H) : \ H_1[\![y]\!])\sigma \rangle \\
\equiv & \quad \{ \star \text{ definition (11) and one-point rule (4.24) of [2] } \} \\
 & \langle \forall\, H_0 \ : \ H_0[\![p]\!]\sigma \ \wedge \ H_0 \parallel H : \ (H_0 \cup H)[\![y]\!])\sigma \rangle \\
\equiv & \quad \{ \text{ trading: (4.28) of [2] } \} \\
 & \langle \forall\, H_0 \ : \ H_0 \parallel H : \ H_0[\![p]\!]\sigma \Rightarrow (H_0 \cup H)[\![y]\!])\sigma \rangle
\end{array}
$$

-
-

Advantages of (*),(−∗)

- The following are immediate consequences of the conection, where ↔ denotes the antisymmetric closure of →:
- *p∗(x1 ⋁x2) ↔ (p∗x1) ⋁(p∗x2)*
- (x1 ⋁x2)∗p↔ (x1 ∗p)⋁(x2 ∗p)
- p−∗(x1 ⋀x2) ↔ (p−∗x1)⋀(p−∗x2)
- and monotonicity, cancellations,
  x → (p −∗ (p ∗ x)) p ∗ (p −∗ y) → y
- etc. and some others, usually not mentioned in the literature
- emp → p−∗p
- p∗x ↔ p∗(p−∗(p∗x))
- p−∗x ↔ p−∗(p∗(p−∗x))

GC for Confined Separational Logic

- If we compare eq(5,6,7) with the standard case eq(4), we see the difference resides in extra coreflexive($\Phi_{\neg\triangleright}, \Phi_{\triangleright}$ and $\Phi_{\triangleleft\triangleright}$)mediating separate union (*)and the split of relations which captures the semantics of p,q.
- Just stick the relevant coreflexive (eg. ΦIn) to separate union ($*$) and carry on:

Contd..

- So, for confined separating conjunction, this leads to the upper adjoint of:
- $$H[\![p \mathrel{-\!\!\neg\!\triangleright} y]\!]\sigma \quad \overset{\text{def}}{=}$$
  $$\langle \forall\, H_0 \;:\; H_0 \mathrel{\neg\!\triangleright} H \;:\; H_0[\![p]\!]\sigma \Rightarrow (H_0 \cup H)[\![y]\!]\sigma \rangle$$

- And lower adjoint of:

$$H[\![p \mathrel{-\!\!\triangleleft\!\triangleright} y]\!]\sigma \quad \overset{\text{def}}{=}$$
$$\langle \forall H_0 : H_0 \mathrel{\triangleleft\!\triangleright} H \;:\; H_0[\![p]\!]\sigma \Rightarrow (H_0 \cup H)[\![y]\!]\sigma \rangle$$

$$H[\![p \mathrel{-\!\!\triangleright} y]\!]\sigma \quad \overset{\text{def}}{=}$$
$$\langle \forall H_0 : H_0 \triangleright H \;:\; H_0[\![p]\!]\sigma \Rightarrow (H_0 \cup H)[\![y]\!]\sigma \rangle$$

Contd…

- In comparison with the standard separated implication, all of the variants place an extra restriction on augmented heap.
- Because of Galois connection, all of the properties derived from (p*), hold for free for all of its variants.

Reasoning

- Semantics of confinement can be checked against eg. what happens to standard property
- $emp*p \leftrightarrow p \leftrightarrow p*emp$
- arising from two facts
- $H[emp]S \equiv H=\perp$
- $H*(H',\perp) \equiv H=H'$
- In the confined variants, semantics rules eventually lead us eg.
- $H[p]S \wedge \Phi\alpha(H, \perp) \equiv H[p]S$ or
- $H[p]S \wedge \Phi\alpha(\perp, H) \equiv H[p]S$
- where α ranges over the three given variants.

Conclusion

- This paper achieves two goals. It provides a semantic characterization of a new extension to separation logic design to reason about confinement of references. And also shows how the calculation of binary relations can help in calculating proofs. The discovery of new operators with the help of Galois connection is particularly useful.