An abstract domain for separation logic formulae

Élodie-Jane Sims
Elodie-Jane.Sims@polytechnique.fr
Plan

1. Introduction
2. Separation logic
3. Introduction to the domain: translations logic $\rightarrow$ the domain
4. About the domain
5. Translation of $\star$
6. Comparisons
Goal: **pointer analysis**: check dereferencing errors, aliases, ...

**$BI^{\mu\nu}$** a separation logic which permit easy descriptions of the memory, e.g.

- $x$ points to a list of [1;2;3]
  \[
  \exists x_2, x_3. \ (x \leftarrow 1, x_2) \ast (x_2 \leftarrow 2, x_3) \ast (x_3 \leftarrow 3, \text{nil})
  \]
- $x$ and $y$ are aliased pointers
  \[
  x = y \land \exists x_1, x_2. \ (x \leftarrow x_1, x_2)
  \]
- Partitionning: $x$ and $y$ belong to two disjoints parts of the heap which have no pointers from one to the other...
Example of a pointer program with a bug

\[
\{(\exists z_1, z_2. \text{nil} \leftrightarrow z_1, z_2) \equiv FALSE\}
\]

\[
\begin{align*}
  x & : = \text{nil}; \\
  \{\exists z_1, z_2. x \leftrightarrow z_1, z_2\} \\
  z & : = x; \\
  \{\exists z_1, z_2. z \leftrightarrow z_1, z_2\} \\
  y & : = z \cdot 1; \\
  \{TRUE\}
\end{align*}
\]
Point-to analyses: Shape/alias analyses

- **Shape analyses**: the analysis builds a graph where
  - the nodes represent locations in the heap
  - the edges represent fields between locations

  The analysis usually do approximation (represent more or less nodes/fields than what is in the heap) and computes some more informations about the nodes or edges of the graph.

  Recent examples: TVLA (Sagiv, Reps, Wilhelm,...), Smallfoot (O’Hearn, Yang, Berdine, Calcagno, Distefano,...)

- **Alias analyses**: a point-to analysis which computes sets of variables
Goal: pointer analysis: check dereferencing errors, aliases, ...

- **$BI^{\mu\nu}$** a separation logic which permit easy descriptions of the memory, e.g.
  
  - $x$ points to a list of [1;2;3]
    \[ \exists x_2, x_3. (x \leftarrow 1, x_2) \ast (x_2 \leftarrow 2, x_3) \ast (x_3 \leftarrow 3, \text{nil}) \]
  
  - $x$ and $y$ are aliased pointers
    \[ x = y \land \exists x_1, x_2. (x \leftarrow x_1, x_2) \]
  
  - Partitionning: $x$ and $y$ belong to two disjoins parts of the heap which have no pointers from one to the other...
Example for a piece of code inserting a cell in a linked list
\[
\{(x \mapsto 1, y) \ast (y \mapsto 3, \text{nil})\}
\]

\[
t : = \text{cons}(2, y);
\]

\[
\{(x \mapsto 1, y) \ast (y \mapsto 3, \text{nil}) \ast (t \mapsto 2, y)\}
\]

\[
x \cdot 2 : = t;
\]

\[
\{(x \mapsto 1, t) \ast (t \mapsto 2, y) \ast (y \mapsto 3, \text{nil})\}
\]
Local reasoning
\[
\begin{align*}
\{ & (x \mapsto 1, y) \ast (y \mapsto 3, \text{nil}) \} \\
& \ast(z \mapsto 4, y)
\end{align*}
\]

\[
t : = \text{cons}(2, y);
\]

\[
\begin{align*}
\{ & (x \mapsto 1, y) \ast (y \mapsto 3, \text{nil}) \ast(t \mapsto 2, y) \} \\
& \ast(z \mapsto 4, y)
\end{align*}
\]

\[
x \cdot 2 : = t;
\]

\[
\begin{align*}
\{ & (x \mapsto 1, t) \ast(t \mapsto 2, y) \ast(y \mapsto 3, \text{nil}) \} \\
& \ast(z \mapsto 4, y)
\end{align*}
\]
We want to use this logic as an interface language for modular analysis.

Analysis 1 $\rightarrow BI^{\mu\nu}$ $\rightarrow$ Analysis 2
Program $\rightarrow BI^{\mu\nu}$ $\rightarrow$ Analysis 3
We have build an intermediate domain such that:

- it is similar to the existing shape/alias analysis domains to allow translations from/to those domains
- it comes with a concrete semantics in term of sets of states which is the same domain as for the formulae’s semantics
- we can translate the formulas into our domain
- it is a cartesian product of different subdomains so that we can cheaply tune the precision depending on the needs (for example, the domain is parametrised by a numerical domain which can be forgotten if we do not care about numericals)
Plan

1. Introduction

2. Separation logic

3. Introduction to the domain: translations logic $\rightarrow$ the domain

4. About the domain

5. Translation of $\ast$

6. Comparisons
Domaine of interpretation: *State*

We have a set of variables $\text{Var}$.

\[
\begin{align*}
\text{Val} &= \text{Int} \cup \text{Bool} \cup \text{Atoms} \cup \text{Loc} & \text{Values} \\
S &= \text{Var} \rightarrow \text{Val} & \text{Stacks} \\
H &= \text{Loc} \rightarrow \text{Val} \times \text{Val} & \text{Heaps} \\
\text{State} &= S \times H
\end{align*}
\]

Rq: stacks can be partial functions
The logic: $BI^{\mu\nu}$

<table>
<thead>
<tr>
<th>Classical connectives</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E = E'$</td>
</tr>
<tr>
<td>$P \Rightarrow Q$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Spatial connectives</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{emp}$</td>
</tr>
<tr>
<td>$P \ast Q$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fixpoints connectives</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_v$</td>
</tr>
<tr>
<td>$\nu X_v. P$</td>
</tr>
</tbody>
</table>

$Var_v = \{X_v, Y_v, \ldots\}$ infinite set of variables of formulae
Semantic of $\ast$

$$[P \ast Q]_\rho = \left\{ s, h_0 \cdot h_1 \mid \begin{array}{l}
\text{dom}(h_0) \cap \text{dom}(h_1) = \emptyset \\
 s, h_0 \in [P]_\rho \\
 s, h_1 \in [Q]_\rho
\end{array} \right\}$$
Examples of formulae

Ex. 1

\[ s = [x \mapsto l_1, y \mapsto l_2] \]
\[ h_1 = [l_1 \mapsto \langle 3, l_2 \rangle] \]

Ex. 2

\[ s = [x \mapsto l_1, y \mapsto l_2] \]
\[ h_2 = [l_2 \mapsto \langle 4, l_1 \rangle] \]

Ex. 3

\[ s = [x \mapsto l_1, y \mapsto l_2] \]
\[ h_1 \cdot h_2 = \left[ \begin{array}{c} l_1 \mapsto \langle 3, l_2 \rangle, \\ l_2 \mapsto \langle 4, l_1 \rangle \end{array} \right] \]

\[ \models (x \mapsto 3, y) \]
\[ \models (y \mapsto 4, x) \]
\[ \models (x \mapsto 3, y) \neq (y \mapsto 4, x) \]
Plan

1. Introduction

2. Separation logic

3. Introduction to the domain: translations logic → the domain

4. About the domain

5. Translation of *

6. Comparisons
Ex1

<table>
<thead>
<tr>
<th>Formulae</th>
<th>$x = \text{nil}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semantics</td>
<td>${s, h \mid s(x) = \text{nil}}, \ldots$</td>
</tr>
<tr>
<td>Translation</td>
<td>$\xrightarrow{x} \text{Nilt}, \ldots$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Formulae</th>
<th>$(x = \text{nil} \lor x = \text{true})$</th>
</tr>
</thead>
</table>
| Translation    | $\xrightarrow{x} \text{Nilt}, \ldots$  
                        $\xrightarrow{} \text{Truet}$ |
Ex2

<table>
<thead>
<tr>
<th>Formula</th>
<th>$A \land B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constraints</td>
<td>cheap translation of $\land$</td>
</tr>
<tr>
<td>Translation</td>
<td>$T(A \land B) \triangleq T'(T'(\top, A), B)$</td>
</tr>
</tbody>
</table>

$\top$ is the empty graph, representing no information
Ex3, Ex4
<table>
<thead>
<tr>
<th>Formula</th>
<th>$x = y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constraints</td>
<td>refine the information for one variables while also refining the information of the second one in a cheap way</td>
</tr>
<tr>
<td>Adds</td>
<td>infinite set of auxiliary variables $TVar$</td>
</tr>
<tr>
<td>$VAR \triangleq Var \uplus TVar$</td>
<td></td>
</tr>
<tr>
<td>Translation</td>
<td><img src="" alt="Diagram" /></td>
</tr>
<tr>
<td>Formula</td>
<td>$x = y \land x = \text{nil}$</td>
</tr>
<tr>
<td>Translation</td>
<td><img src="" alt="Diagram" /></td>
</tr>
</tbody>
</table>
### Ex5

<table>
<thead>
<tr>
<th>Formula</th>
<th>$x = y \land x = \text{nil}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translation</td>
<td><img src="" alt="Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Formula</th>
<th>$(\exists x. x = y \land x = \text{nil}) \equiv (y = \text{nil})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translation</td>
<td><img src="" alt="Diagram" /></td>
</tr>
</tbody>
</table>
## Ex6

<table>
<thead>
<tr>
<th>Formula</th>
<th>$(x &lt; y + 3)$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Translation</strong></td>
<td><img src="Image" alt="Translation Diagram" /></td>
</tr>
<tr>
<td></td>
<td>$d \in D$ encodes that $\alpha &lt; \beta + 3$</td>
</tr>
</tbody>
</table>
**Ex7**

<table>
<thead>
<tr>
<th>Formula</th>
<th>“x is a location not allocated”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semantic</td>
<td>( { s, h \mid s(x) \in \text{Loc} \land s(x) \notin \text{dom}(h) } )</td>
</tr>
<tr>
<td>Translation</td>
<td>( x \xrightarrow{Dangling_{\text{Loc}}} )</td>
</tr>
</tbody>
</table>

An abstract domain for separation logic formulae - p. 24/44
### Ex8

<table>
<thead>
<tr>
<th>Formula</th>
<th><code>emp</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Semantic</td>
<td>{s, h \mid \text{dom}(h) = \emptyset}</td>
</tr>
</tbody>
</table>
| Adds | \(HU \triangleq \mathcal{P}(TVar)\)  \
| | \(HO \triangleq \mathcal{P}(TVar) \cup \text{full}\) |
| Translation | \((\top, \bot, \emptyset, \bot, \bot, \bot)\) |
### Ex9

<table>
<thead>
<tr>
<th>Formula</th>
<th>((x \mapsto \text{true, nil}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semantic</td>
<td>({s, h\mid s(x) \to \langle \text{True, nil}\rangle = h})</td>
</tr>
<tr>
<td>Translation</td>
<td><img src="" alt="Translation Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Formula</th>
<th>((x \mapsto \text{true, nil}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semantic</td>
<td>({s, h\mid s(x) \to \langle \text{True, nil}\rangle \subseteq h})</td>
</tr>
<tr>
<td>Translation</td>
<td><img src="" alt="Translation Diagram" /></td>
</tr>
</tbody>
</table>
Variables represent at most one value. To allow approximation we introduce summary nodes which can represent several values.

<table>
<thead>
<tr>
<th>Formula</th>
<th>approx. of ((x = \text{true} \land y = \text{false})) by ((x = \text{true}) \land (\neg x = \text{false}) \land (y = \text{false}) \land (\neg y = \text{true}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translation</td>
<td>(x \rightarrow \alpha \rightarrow \text{True} \land \neg\neg {\alpha} \land \neg\neg\neg) (y \rightarrow \text{False} \land \neg\neg\neg)</td>
</tr>
</tbody>
</table>
Ex11: finite acyclic list of True starting from x

Formula

\[
\mu X_v. \left( (x = \text{nil}) \lor \exists x_2. \left( x \mapsto (\text{true}, x_2) \ast X_v[x_2/x] \right) \right)
\]

Translation

∅ is the set of infinite summary nodes, for infinite list μ would be replaced by ν and ∅ by \{α\}.

\begin{align*}
\end{align*}
**Ex12: increase precision of union**

<table>
<thead>
<tr>
<th>Formula</th>
<th>((x = \text{nil} \land y = \text{true}) \lor (x = \text{true} \land y = \text{nil}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translation</td>
<td><img src="" alt="Translation Diagram" /></td>
</tr>
</tbody>
</table>

**Translation Diagram**

- \(\alpha_1\) to \(\text{Nilt}\) via \(x\): \(\{\text{eq}\}\)
- \(\alpha_2\) to \(\text{Tru et}\)
- \(\alpha_3\) to \(\text{Nilt}\) via \(y\): \(\{\text{eq}\}\)
- \(\alpha_4\) to \(\text{Tru et}\)

\(\{\text{eq}\}\) indicates the set of equations.
Plan

1. Introduction
2. Separation logic
3. Introduction to the domain: translations logic → the domain
4. About the domain
5. Translation of *
6. Comparisons
We want

\[
\begin{align*}
[x] & \xrightarrow{\alpha} N\text{umt} \\
[y] \\
\end{align*}
\]

\[
= [x] \xrightarrow{\alpha} N\text{umt} \cap [y] \xrightarrow{\alpha} N\text{umt}
\]

\[
\text{4}\quad \text{4}\quad \text{4}
\]
Semantic

\[
\begin{align*}
\text{Val} & \triangleq \mathbb{Z} \cup \text{Bool} \cup \text{nil} \cup \text{Loc} & \text{Val'} & \triangleq \text{Val} \cup \{\text{ood}\} \\
S & \triangleq \text{Var} \rightarrow \text{Val} & S' & \triangleq \text{Var}^{\text{total}} \rightarrow \text{Val'} \\
H & \triangleq \text{Loc} \rightarrow (\text{Val} \times \text{Val}) & F & \triangleq \text{TVar}^{\text{total}} \rightarrow \mathcal{P}(\text{Val'}) \\
\text{State} & \triangleq S \times H & R & \triangleq \text{Loc} \rightarrow \mathcal{P}(\text{Loc}) \\
\text{MFR} & \triangleq \mathcal{P}(S' \times H \times F \times R)
\end{align*}
\]
\[\boxed{\cdot} \in \mathbb{AR} \rightarrow \mathcal{P}(\text{State})\]
\[\boxed{\cdot} \triangleq \{s, h \mid s, h, f, r \in \boxed{\cdot}\}\]

\[\boxed{(ad, hu, ho, sn, sn^{\infty}, t, d)} \triangleq \boxed{ad}^4 \cap \boxed{hu}^1 \cap \boxed{ho}^{1'} \cap \boxed{sn}^2 \cap \boxed{sn^{\infty}}^{2'} \cap \boxed{t}^3 \cap \boxed{d}^{7} \cap \text{sem}^*\]

\[\boxed{\cdot} \triangleq \boxed{ad}^4 \cap \boxed{v, ad(v)}^5, \quad \forall v \in \text{VAR}\]
Operations

- union, intersection

- extension (replace \([v \rightarrow S]\) by \([v \rightarrow \{v'\}|v' \rightarrow S]\) with a fresh \(v'\) )
  used to tune the precision of the union

- merging (replace \([v_1 \rightarrow S_1 | v_2 \rightarrow S_2]\) by \([v_2 \rightarrow (S_1 \cup S_2)]\))
  used with the widening

- translations from formulae to the domain
Plan

1. Introduction
2. Separation logic
3. Introduction to the domain: translations logic $\rightarrow$ the domain
4. About the domain
5. Translation of $*$
6. Comparisons
**for simple types (no variables)**

When $A, B \in \{\text{Nilt, Truet, Falset, Oodt, Numt}\}, A \neq B$

<table>
<thead>
<tr>
<th>$*$</th>
<th>${A}$</th>
<th>${B}$</th>
<th>${Dgt}$</th>
<th>${Loc}$</th>
<th>${Loc(...})$</th>
<th>$\top$</th>
</tr>
</thead>
<tbody>
<tr>
<td>${A}$</td>
<td>${A}$</td>
<td>$\Omega$</td>
<td>$\Omega$</td>
<td>$\Omega$</td>
<td>$\Omega$</td>
<td>${A}$</td>
</tr>
<tr>
<td>${Dgt}$</td>
<td></td>
<td>${Dgt}$</td>
<td>${Loc}$</td>
<td>${Loc(...})$</td>
<td>${Dgt, Loct}$</td>
<td></td>
</tr>
<tr>
<td>${Loc}$</td>
<td></td>
<td></td>
<td></td>
<td>$\Omega$</td>
<td>$\Omega$</td>
<td>${Loc}$</td>
</tr>
<tr>
<td>${Loc(...})$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\Omega$</td>
<td>${Loc(...})$</td>
</tr>
<tr>
<td>$\top$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\top$</td>
</tr>
</tbody>
</table>

The function sometimes return $\Omega$ as the error value.
The initial graph of auxiliary variables

\[
\begin{align*}
G_0 &= \text{only aux. var.} \\
G_1 &= \text{only aux. var.} \\
G_0 \cup G_1 &= \text{only aux. var.}
\end{align*}
\]

Since \( \text{dom}(G_0) \cap \text{dom}(G_1) = \emptyset \).
From graph of sets to sets of graph

For $v$ variables (auxiliary or not) not summary nodes.
“Jumping” variables (simplified)

- \( \alpha \) not summary node

\[
\begin{align*}
(x & \rightarrow \alpha & \rightarrow A) \ast (x & \rightarrow B) \\
= & (x & \rightarrow \alpha & \rightarrow (\alpha & \rightarrow A \ast x & \rightarrow B)) \\
= & (x & \rightarrow \alpha & \rightarrow A \ast B)
\end{align*}
\]

Because \( x \rightarrow \alpha \) means \( s(x) = f(\alpha) \) so the \( \ast \)-information about \( x \) can be treated for \( \alpha \) instead

- \( \alpha \) cycles

\[
\begin{align*}
(x & \rightarrow \alpha_0 & \rightarrow \cdots & \rightarrow \alpha_n \rightarrow (x & \rightarrow B) \\
= & (x & \rightarrow \alpha_0 & \rightarrow \cdots & \rightarrow \alpha_n & \rightarrow (\top \ast (x & \rightarrow B)) \\
= & (x & \rightarrow \alpha_0 & \rightarrow \cdots & \rightarrow \alpha_n & \rightarrow (\top \ast B))
\end{align*}
\]

\( s(x) = f(\alpha_0) = \ldots = f(\alpha_n) = f(\alpha_0) \) is \( s(x) = f(\alpha_0) = \ldots = f(\alpha_n) \)
\( \alpha \) summary node

\[ \left( x \xrightarrow{\alpha} A \right) \neq \left( x \xrightarrow{\alpha} A \right) \cup \left( x \xrightarrow{\alpha} B \right) \]

but

\[ \left( x \xrightarrow{\alpha} A \right) \subseteq \left( \begin{array}{c} \alpha \\ A \end{array} \xrightarrow{} \begin{array}{c} \alpha \\ B \end{array} \right) \cup \left( \begin{array}{c} \alpha \\ A \end{array} \xrightarrow{} \begin{array}{c} \alpha \\ B \end{array} \right) \]

Because \( s(x) \in f(\alpha) \subseteq (A \cup B) \Rightarrow (s(x) \in A \land f(\alpha) \subseteq (A \cup B)) \lor (s(x) \in B \land f(\alpha) \subseteq (A \cup B)) \)

For example, we get
An abstract domain for separation logic formulae - p. 41/44
Plan

1. Introduction
2. Separation logic
3. Introduction to the domain: translations logic → the domain
4. About the domain
5. Translation of *
6. Comparisons

Élodie-Jane Sims
**Comparisons**

- the • represent nodes in the usual shape graphs

- summary nodes as for other shape graphs, seems to give more possibilities than predicate abstraction (with each time a specific predicate for list, etc...) but the technics of predicate and their algorithm/heuristics (like folding/unfolding) could probably also be use on our graphs

- a lonely outgoing edge can be seen as a “must” arrow (or valued 1), several outgoing edges from a variable can be seen as a “may” arrow (or valued 1/2, but it is a bit more precise because we know that one of them should exist), and an edge to $\emptyset$ can be seen as a “must not” arrow (or valued 0)
we deal with numerical (for what we know, only Magill & al. also do)

we have a formal semantic of our domain, the semantics of auxiliary variables are formally defined and formally used in the proofs

we don’t have to check for equalities of variables

the domain is a cartesian product, we can add or remove some parts depending on the precision we want

we directly have in the domain the “Dangling” information which is suitable for cleaning checking
End