Biabduction for Separation Logic
Deliverable D1-4
ANR project VECOLIB
September 2017

Abstract
This deliverable reports on the bi-abduction procedures for Separation Logic developed during the VECOLIB project. This includes work on Separation Logic with and without inductive predicates, carried out at IRIF and VERIMAG, respectively.

1 Definition of Separation Logic

1.1 Syntax
We consider a signature \( \Sigma \), such that \( \Sigma^s = \{ \text{Loc}, \text{Bool} \} \) and \( \Sigma^f = \emptyset \), i.e. the only sorts are the boolean and location sort, with no function symbols defined on it, other than equality. Observe that, in this case \( T_{\Sigma}(x) = x \), for any \( x \subseteq \text{Var} \), i.e. the only terms occurring in a formula are variables of sort Loc. In the rest of this section we consider systems whose constraints are Separation Logic formulae, generated by the following syntax:

\[ \varphi ::= \bot | x \approx y | \text{emp} | x \mapsto (y_1, \ldots, y_k) | \varphi_1 \ast \varphi_2 | \neg \varphi_1 | \varphi_1 \land \varphi_2 | \exists x. \varphi_1 \]

where \( k > 0 \) is a fixed constant denoting the number of outgoing selector fields from a memory cell.

We consider the following shorthand notations:
- \( \top \equiv \neg \bot \)
- \( x \mapsto (y_1, \ldots, y_k) \equiv x \mapsto (y_1, \ldots, y_k) \ast \top \)
- \( \text{alloc}(x) \equiv x \mapsto (x, \ldots, x) \ast \bot \)
- for any \( n \in \mathbb{Z} \): \( |h| \geq n \equiv \begin{cases} \top & \text{if } n \leq 0 \\ (|h| \geq n - 1) \ast \neg \text{emp} & \text{otherwise} \end{cases} \)

1.2 Semantics
We interpret \( \text{Loc} \) as a countably infinite set \( L \). A heap is a finite partial mapping \( h : L \rightarrow \text{fin} L^k \) associating locations with \( k \)-tuples of locations. We denote by \( \text{dom}(h) \) the set of locations on which \( h \) is defined, by \( \text{img}(h) \) the set of locations occurring in the range of \( h \), and by \( \text{Heaps} \) the set of heaps. Two heaps \( h_1 \) and \( h_2 \) are disjoint if \( \text{dom}(h_1) \cap \text{dom}(h_2) = \emptyset \). In this case \( h_1 \cup h_2 \) denotes their union, which is undefined if \( h_1 \) and \( h_2 \) are not disjoint. Given a valuation \( \nu : \text{Var} \rightarrow L \) and a heap \( h \), the semantics of
SL formulae is defined as follows:
\[
\begin{align*}
\nu, h &\models x \approx y \iff \nu(x) = \nu(y) \\
\nu, h &\models \text{emp} \iff h = \emptyset \\
\nu, h &\models x \longmapsto (y_1, \ldots, y_k) \iff h = \{(\nu(x), (\nu(y_1), \ldots, \nu(y_k)))\} \\
\nu, h &\models \phi_1 \ast \phi_2 \iff \text{there exist } h_1, h_2 \in \text{Heaps} : h = h_1 \uplus h_2 \text{ and } \nu, h_i \models \phi_i, i = 1, 2 \\
\nu, h &\models \exists x. \phi(x) \iff \nu[x \leftarrow \ell], h \models \phi(x), \text{ for some } \ell \in L
\end{align*}
\]

The semantics of boolean connectives is the usual one, omitted for space reasons. A formula \( \phi \) is satisfiable if there exists a valuation \( \nu \) and a heap \( h \) such that \( \nu, h \models \phi \). Given formulae \( \varphi \) and \( \psi \), we say that \( \phi \) entails \( \psi \), denoted \( \phi \models \psi \) if \( \nu, h \models \varphi \) implies \( \nu, h \models \psi \), for each valuation \( \nu \) and each heap \( h \).

### 1.2.1 Bi-Abduction Problems

Consider two formulae \( \varphi \) and \( \psi \) of Separation Logic (SL). The problems of abduction, frame inference and bi-abduction are defined below:

- the abduction problem asks for a formula \( X \) such that \( \varphi \ast X \models \psi \),
- the frame inference problem asks for a formula \( Y \) such that \( \varphi \models \psi \ast Y \),
- the bi-abduction problem asks for formulae \( X \) and \( Y \) such that \( \varphi \ast X \models \psi \ast Y \).

### 2 Separation Logic without Inductive Definitions

We address first the three problems above in case \( \varphi \) and \( \psi \) do not contain predicate atoms. It is not difficult to prove that:

1. the weakest solution of the abduction problem is \( X \equiv \varphi \rightarrow \psi \), and
2. the strongest solution of the frame inference problem is \( Y \equiv \neg (\varphi \rightarrow \neg \psi) \), also denoted as \( \varphi \leftarrow \neg \psi \).

Based on these observations, a possible solution to the bi-abduction problem can be defined as follows:

\[
\begin{align*}
X &\equiv \varphi \rightarrow (\psi \ast \top) \\
Y &\equiv (\varphi \ast X) \rightarrow \neg \psi \equiv (\varphi \ast (\varphi \leftarrow (\psi \ast \top))) \rightarrow \neg \psi
\end{align*}
\]

The main difficulty with the above solutions is the use of the magic wand \( \rightarrow \) connective, which poses important problems for automated reasoning. The solution we consider is to translate any SL formula as a boolean combination of SL-minterms, defined in the following.

**Definition 1** An SL-minterm is either \( \bot \) or a formula of the form:

\[
\phi^{eq} \land \phi^{pl} \land \phi^a \land |h| \geq \min_\phi \land |h| < \max_\phi
\]

where:

- \( \phi^{eq} \) is a conjunction of equalities and disequalities,
- \( \phi^{pl} \) is a conjunction of literals of the form \( x \leftarrow (y_1, \ldots, y_k) \) or \( x \not\leftarrow (y_1, \ldots, y_k) \),
- \( \phi^a \) is a conjunction of literals of the form \( \text{alloc}(x) \) or \( \neg \text{alloc}(x) \),
- \( \min_\phi \in \mathbb{N} \) and \( \max_\phi \in \mathbb{N} \cup \{\infty\} \).

The main result here is that any quantifier-free SL formula is equivalent to a boolean combination of SL-minterms. This result is obtained by giving translations for the formulae \( \phi_1 \ast \phi_2 \) and \( \phi_1 \leftarrow \phi_2 \), where \( \phi_1 \) and \( \phi_2 \) are SL-minterms satisfying a few additional restrictions, such as completeness w.r.t.
equalities and allocations. This latter conditions can always be enforced by considering a finite disjunction of cases, stating which variables are equal, not equal, allocated and not allocated, respectively. The translation can be implemented in polynomial space, which means that the bi-abduction problems above can be solved in polynomial space and exponential time.