Local Reasoning for Termination

Ina Schaefer
ina.schaefer@mpi-sb.mpg.de

joint work with Andreas Podelski

Max-Planck-Institut für Informatik, Saarbrücken
Motivation

For a pointer program $P$, separation logic proves

$$\{pre\} \ P \ \{post\}$$

where $pre$ and $post$ are separation logic assertions over program states $(s,h)$, e.g.

$$\phi \equiv (y = x \land y.n = z) \ast \text{list}(z)$$
Motivation (2)

For a program $P$ with transition relation $R$, construct relation $T$ such that $T$ is a transition invariant, i.e.

$$R^+ \subseteq T$$

Then we conclude:

- Partial Correctness: $T \wedge pre \models post'$
- Termination of $P$: $T$ finite union of well-founded relations
Motivation (2)

For a program $P$ with transition relation $R$, construct relation $T$ such that $T$ is a transition invariant, i.e.

$$R^+ \subseteq T$$

Then we conclude:

- Partial Correctness: $T \land pre \models post'$
- Termination of $P$: $T$ finite union of well-founded relations

So, what if we lift separation logic assertions to relations ...
Outline

- Separated Transition Logic
  - Separated Transition Constraints
  - Reasoning with Separated Transition Constraints
- Proof Rule for Total Correctness
  - Based on Transition Invariants
  - Completeness Result wrt Separation Logic
- Discussion
Separated Transition Constraints

- Capture local effect of program statements
- Define relation over program states, i.e.

\[ ((s, h), (s', h')) \models \varphi \]
Separated Transition Constraints

- Capture local effect of program statements
- Define relation over program states, i.e.

\[(s,h), (s',h')) \models \varphi\]

Example:

\[\varphi \equiv (x.n = x' \land x = y' \land y'.n' = y)\]

\[x = y'\]

\[n\]

\[n'\]

\[y\]

\[x'\]

unchanged
Let $v$ be a program variable. Then we define the syntax of separated transition constraints as

$$w ::= v \mid v'$$

$$\varphi ::= \text{emp} \mid \text{emp}'$$

$$\mid w_1 = w_2$$

$$\mid w_1.n = w_2$$

$$\mid w_1.n' = w_2$$

$$\mid \varphi_1 \ast \varphi_2$$

$$\mid \varphi_1 \land \varphi_2$$

$$\mid \exists w \varphi$$
Semantics

- $\textit{Var}$: set of program variables
- $\textit{Val}$: set of values with $\textit{Loc} \subseteq \textit{Val}$
- Program State $(s, h)$: pair of stack $s$ and heap $h$
- Stack: $s : \textit{Var} \rightarrow_{\text{fin}} \textit{Val}$
- Heap: $h : \textit{Loc} \rightarrow_{\text{fin}} \textit{Val}$
Semantics

- \textit{Var}: set of program variables
- \textit{Val}: set of values with \( \text{Loc} \subseteq \text{Val} \)
- Program State \((s, h)\): pair of stack \(s\) and heap \(h\)
- Stack: \(s : \text{Var} \rightarrow_{\text{fin}} \text{Val}\)
- Heap: \(h : \text{Loc} \rightarrow_{\text{fin}} \text{Val}\)

Empty Heap

\[
((s, h), (s', h')) \models \text{emp}^\gamma \text{ if } \text{dom}(h'^\gamma) = \emptyset
\]
Variable Assignments

For $v_1, v_2 \in \text{Var}$:

$(((s, h), (s', h'))) \models (v_1 = v_2)$ if $s(v_1) = s(v_2)$

$(((s, h), (s', h'))) \models (v'_1 = v_2)$ if $s'(v_1) = s(v_2)$

$\vdots$

$(((s, h), (s', h'))) \models (v_1.n = v_2)$ if $h(s(v_1)) = s(v_2)$

$\vdots$

$(((s, h), (s', h'))) \models (v'_1.n' = v'_2)$ if $h'(s'(v_1)) = s'(v_2)$

and $\{s(v_1)\} = \text{dom}(h)$

$\vdots$

and $\{s'(v_1)\} = \text{dom}(h')$
Logical Conjunction

\[((s, h), (s', h')) \models \varphi_1 \land \varphi_2 \text{ if } ((s, h), (s', h')) \models \varphi_1 \land ((s, h), (s', h')) \models \varphi_2\]
Spatial Conjunction

\[ ((s, h), (s', h')) \models \varphi_1 \ast \varphi_2 \text{ if } h = h_1 \uplus h_2 \text{ and } h' = h'_1 \uplus h'_2 \]

such that

\[ ((s, h_1), (s', h'_1)) \models \varphi_1 \]
\[ ((s, h_2), (s', h'_2)) \models \varphi_2 \]
Existential Quantification

\[ ((s, h), (s', h')) \models \exists z \, \varphi \]

if \[ \exists v \in Val((s[z \mapsto v], h)(s', h')) \models \varphi \]

\[ ((s, h), (s', h')) \models \exists z' \, \varphi \]

if \[ \exists v \in Val((s, h)(s'[z \mapsto v], h')) \models \varphi \]
Additionally, we define the **non-tight interpretation**

\[ ((s, h), (s', h')) \models \varphi \text{ if } h = h_1 \uplus h_2 \text{ and } h' = h'_1 \uplus h'_2 \]

and \( h_2 = h'_2 \) such that

\[ ((s, h_1), (s', h'_1)) \models \varphi \]

<table>
<thead>
<tr>
<th>( \varphi_1 )</th>
<th>unchanged</th>
</tr>
</thead>
<tbody>
<tr>
<td>( h_1 )</td>
<td>( h_2 )</td>
</tr>
</tbody>
</table>
Semantics (7)

- **Allocation**

\[ ((s, h), (s', h')) \models alloc(x) \text{ if } ((s, h)(s', h')) \models emp \land \exists z' . n'.n' = z' \]

- **Deallocation**

\[ (s, h), (s', h') \models free(x) \text{ if } ((s, h)(s', h')) \models \exists z . n = z \land emp' \]
Reasoning with SepTC

- **Separation Logic:**

  \{ pre \} statements \{ post \}

  by Axioms and Inference Rules, e.g. Frame Rule

- **Relational Reasoning:**

  \( pre \land T_{statements} \models post \)

  **Main Concept:**

  Relational Composition to construct \( T_{statements} \)
Composition

\[(\varphi_1 \circ \varphi_2)[\bar{x}, \bar{x}'] \equiv \exists \bar{x}' \ (\varphi_1[\bar{x}, \bar{x}'] \ast (v.n = v.n')) \land (\varphi_2[\bar{x}', \bar{x}''] \ast (w.n' = w.n''))\]
Distributive Law

If \( \varphi_1 \) and \( \psi_1 \) have the same domain,

\[
(\varphi_1 \ast \varphi_2) \land (\psi_1 \ast \psi_2) \equiv (\varphi_1 \land \psi_1) \ast (\varphi_2 \land \psi_2)
\]
Outline

- Separated Transition Logic
  - Separated Transition Constraints
  - Reasoning with Separated Transition Constraints

- Proof Rule for Total Correctness
  - Based on Transition Invariants
  - Completeness Result wrt Separation Logic

- Discussion
Separated Transition Invariants

Given a program $P$ with transition relation $R$.

A **separated transition invariant** $T$ is a disjunction

$$T = \{T_1, \ldots, T_n\}$$

of separated transition constraints $T_1, \ldots, T_n$ such that

$$R^+ \subseteq T$$
Proof Rule

\[ P = (\Sigma, I, R) \] program

\[ T \subseteq \Sigma \times \Sigma \] \text{separated transition invariant}

\text{pre and post} \text{ pre and post condition for } \varphi

1. \( T \land \text{pre} \models \text{post} \)

2. \( T \) is a finite union of well founded relations.

\hline
\end{tabular}

\[ P \] is totally correct
Completeness Results

Theorem 1
Separated transition logic is **complete relative to separation logic**: If for a program $P$ without allocation and deallocation of heap cells, separation logic proves the Hoare triple

$$\{ \varphi_1 \} P \{ \varphi_2 \}$$

correct, then there exists a separated transition invariant denoting a relation $T$ such that

$$\varphi_1 \land T \models \varphi_2$$

**Proof:** By Induction on Axioms and Inference Rules of Separation Logic
Theorem 2 Separated transition logic is relatively complete for termination proofs:

Whenever a program $P$ terminates,

there exists a separated transition invariant denoting a relation $T$ such that $T$ is a finite union of well-founded relations.

Proof: Immediately from results of transition invariants
Conclusion

Contributions:

- Verification of Safety and Liveness Properties of Pointer Programs
- Separated Transition Constraints: local effect of programs statements by relations
- Separated Transition Logic: local reasoning about relations over program states
- Proof Rule based on Separated Transition Invariants and complete wrt. Separation Logic.

Future Work

- Recursive Pointer Programs [ESOP’05]
- Automatization of Proof Rule by Abstraction