Reasoning in the Bernays-Schönfinkel-Ramsey Fragment of Separation Logic

Andrew Reynolds\textsuperscript{1}, Radu Iosif\textsuperscript{2}, Cristina Serban\textsuperscript{2}

\textsuperscript{1}The University of Iowa
\textsuperscript{2}Université de Grenoble Alpes, CNRS, VERIMAG

January 15, 2017
Outline

1 Motivation
   ● Why separation logic?
   ● Inductive proofs
   ● Syntax and semantics

2 The Bernays-Schönfinkel-Ramsey fragment of SL

3 An SMT procedure for $\exists^*\forall^*\text{SL}(T)$

4 Evaluation

5 Conclusions and future work

6 References
Why separation logic?

- Program analyzers ignore or build coarse abstractions of the heap.
- Reasoning about the heap is crucial for program analysis tasks, but hard: compositional methods required to ensure scalability.
Why separation logic?

- Program analyzers ignore or build coarse abstractions of the heap.
- Reasoning about the heap is crucial for program analysis tasks, but hard: compositional methods required to ensure scalability.

Separation logic (SL) framework

- Reasoning about heaps and dynamic linked data structures.
- **Local reasoning**: inferring contracts for different parts of the program, regardless of how they are used, and combine them.
Building a list segment $ls(x, y)$

$emp \land x = y$
Building a list segment $ls(x, y)$

\[\text{empty} \quad \text{emp} \land x = y \quad x \mapsto u\]
Building a list segment $ls(x, y)$

\[
\text{empty} \\
emp \land x = y
\]

\[
x \mapsto u \quad ls(u, y)
\]
Building a list segment $\text{ls}(x, y)$

$\text{empty}$

$\text{emp} \land x = y$

$x \mapsto u \ast \text{ls}(u, y)$
Building a list segment $ls(x, y)$

$$\text{empty}$$

$$\text{emp} \land x = y$$

$$x \mapsto u \ast ls(u, y)$$

$$ls(x, y) \equiv \text{emp} \land x = y \lor \exists u . x \mapsto u \ast ls(u, y)$$

list segment
Building a list segment $ls(x, y)$

\[
\text{empty} \equiv \text{emp} \land x = y
\]

\[
x \mapsto u \ast \text{ls}(u, y)
\]

\[
\text{ls}(x, y) \equiv \text{emp} \land x = y \lor \exists u. \ x \mapsto u \ast \text{ls}(u, y)
\]

list segment
Building a list segment \( ls(x, y) \)

\[
\text{empty} \quad \text{emp} \land x = y
\]

\[
x \mapsto u \ast ls(u, y)
\]

\[
ls(x, y) \equiv \text{emp} \land x = y \lor \exists u . \ x \mapsto u \ast ls(u, y)
\]

\[
\hat{ls}(x, y) \equiv \text{emp} \land x = y \lor x \neq y \land \exists u . \ x \mapsto u \ast \hat{ls}(u, y)
\]

list segment

acyclic list segment
Inductive proofs

\[ \hat{ls}(x, y) \vdash ls(x, y) \]

\[ ls(x, y) \equiv emp \land x = y \lor \exists u . x \mapsto u \ast ls(u, y) \quad \text{list segment} \]

\[ \hat{ls}(x, y) \equiv emp \land x = y \lor x \neq y \land \exists u . x \mapsto u \ast \hat{ls}(u, y) \quad \text{acyclic list segment} \]
**Inductive proofs**

\[ x \neq y \land x \mapsto z \ast \hat{ls}(z, y) \vdash \exists u . x \mapsto u \ast ls(u, y) \]

\[ \hat{ls}(x, y) \vdash ls(x, y) \]

\[ \text{ls}(x, y) \equiv \text{emp} \land x = y \lor \exists u . x \mapsto u \ast \text{ls}(u, y) \]

\[ \hat{ls}(x, y) \equiv \text{emp} \land x = y \lor x \neq y \land \exists u . x \mapsto u \ast \hat{ls}(u, y) \]

list segment

acyclic list segment
Inductive proofs

\[
\begin{align*}
\neg x = y \land x \mapsto z \vdash \exists u . x \mapsto u \ast \text{ls}(u, y) \\
\hat{\text{ls}}(x, y) \vdash \text{ls}(x, y)
\end{align*}
\]

by instantiation \( u \leftarrow z \)

\[
\begin{align*}
\text{ls}(x, y) & \equiv \text{emp} \land x = y \lor \exists u . x \mapsto u \ast \text{ls}(u, y) \\
\hat{\text{ls}}(x, y) & \equiv \text{emp} \land x = y \lor x \neq y \land \exists u . x \mapsto u \ast \hat{\text{ls}}(u, y)
\end{align*}
\]

list segment

acyclic list segment
Inductive proofs

\[
\hat{ls}(z, y) \vdash ls(z, y)
\]

\[
x \neq y \land x \mapsto z \implies \hat{ls}(z, y) \vdash \exists u . x \mapsto u \ast ls(u, y)
\]

\[
\hat{ls}(x, y) \vdash ls(x, y)
\]

\[
ls(x, y) \equiv \text{emp} \land x = y \lor \exists u . x \mapsto u \ast ls(u, y)
\]

\[
\hat{ls}(x, y) \equiv \text{emp} \land x = y \lor x \neq y \land \exists u . x \mapsto u \ast \hat{ls}(u, y)
\]
Inductive proofs

\[ \widehat{ls}(z, y) \vdash ls(z, y) \]

\[ x \neq y \land x \mapsto z \ast \widehat{ls}(z, y) \vdash \exists u . \ x \mapsto u \ast ls(u, y) \]

\[ \widehat{ls}(x, y) \vdash ls(x, y) \]

\[ x \neq y \land x \mapsto z \models \exists u . \ x \mapsto u \]

by instantiation \( u \leftarrow z \)

\[ x \neq y \land x \mapsto z \land \forall u . \ \lnot x \mapsto u \] is unsatisfiable
Inductive proofs

\[ \hat{ls}(z, y) \vdash ls(z, y) \]
\[ x \neq y \land x \mapsto z \ast \hat{ls}(z, y) \vdash \exists u \cdot x \mapsto u \ast ls(u, y) \]
\[ \hat{ls}(x, y) \vdash ls(x, y) \]

\[ x \neq y \land x \mapsto z \models \exists u \cdot x \mapsto u \]
by instantiation \( u \leftarrow z \)

\[ x \neq y \land x \mapsto z \iff \exists u \cdot x \mapsto u \]
\[ \iff x \neq y \land x \mapsto z \land \forall u \cdot \neg x \mapsto u \text{ is unsatisfiable} \]

We ask for a decision procedure for \( \exists^* \forall^* \text{SL} \) that is also capable of producing unsatisfiability witnesses (instantiations)
Syntax of SL

Let $\Sigma = (\Sigma^s, \Sigma^f)$ be a signature where:

- $\Sigma^s$ consists of a set of sort symbols $\sigma$
- $\Sigma^f$ consists of a set of function symbols $f^{\sigma_1 \times \ldots \times \sigma_n \times \sigma}$
- $\text{Loc}, \text{Data} \in \Sigma^s$ are the location and data sorts
Syntax of SL

Let $\Sigma = (\Sigma^s, \Sigma^f)$ be a signature where:

- $\Sigma^s$ consists of a set of sort symbols $\sigma$
- $\Sigma^f$ consists of a set of function symbols $f^{\sigma_1 \times \ldots \times \sigma_n \times \sigma}$
- Loc, Data $\in \Sigma^s$ are the location and data sorts

SL is the set of formulae:

$$\phi ::= t^\sigma \approx u^\sigma \mid \bot \mid t^{\text{Loc}} \rightarrow u^{\text{Data}} \mid \text{emp} \mid \phi_1 \land \phi_2 \mid \phi_1 \rightarrow \phi_2 \mid \phi_1 \ast \phi_2 \mid \phi_1 \ast \ast \phi_2 \mid \exists x^\sigma . \phi_1(x)$$

We consider a special function symbol nil^{Loc}
Let $T = (\Sigma, I)$ be a first-order theory, where:

- $\Sigma = (\Sigma^s, \Sigma^f)$ is a signature
- $I$ is a set of $\Sigma$-interpretations $\mathcal{I}$
Semantics of SL (\( T \))

Let \( T = (\Sigma, I) \) be a **first-order theory**, where:

1. \( \Sigma = (\Sigma^s, \Sigma^f) \) is a signature
2. \( I \) is a set of \( \Sigma \)-interpretations \( I \)
   - \( \sigma^I \) is the interpretation of sort \( \sigma \in \Sigma^s \)
   - \( f^I : \sigma_1^I \times \ldots \times \sigma_n^I \rightarrow \sigma^I \) is the interpretation of \( f_{\sigma_1 \times \ldots \times \sigma_n \times \sigma} \in \Sigma^f \)
   - \( t^I \) is the interpretation of a term over \( \Sigma \), defined inductively
Semantics of SL ($T$)

Let $T = (\Sigma, I)$ be a **first-order theory**, where:

- $\Sigma = (\Sigma^s, \Sigma^f)$ is a signature
- $I$ is a set of $\Sigma$-interpretations $I$
  - $\sigma^I$ is the interpretation of sort $\sigma \in \Sigma^s$
  - $f^I : \sigma^I_1 \times \ldots \times \sigma^I_n \rightarrow \sigma^I$ is the interpretation of $f_{\sigma_1 \times \ldots \times \sigma_n \times \sigma} \in \Sigma^f$
  - $t^I$ is the interpretation of a term over $\Sigma$, defined inductively

For $I \in I$, a **heap** is a finite partial function $h : \text{Loc}^I \rightarrow_{\text{fin}} \text{Data}^I$
Semantics of SL (T)

Let \( T = (\Sigma, I) \) be a **first-order theory**, where:

- \( \Sigma = (\Sigma^s, \Sigma^f) \) is a signature
- \( I \) is a set of \( \Sigma \)-interpretations \( \mathcal{I} \)
  - \( \sigma^I \) is the interpretation of sort \( \sigma \in \Sigma^s \)
  - \( f^I : \sigma^I_1 \times \ldots \times \sigma^I_n \rightarrow \sigma^I \) is the interpretation of \( f^{\sigma_1 \times \ldots \times \sigma_n \times \sigma} \in \Sigma^f \)
  - \( t^I \) is the interpretation of a term over \( \Sigma \), defined inductively

For \( I \in I \), a **heap** is a finite partial function \( h : \text{Loc}^I \rightarrow_{\text{fin}} \text{Data}^I \)

\( \text{SL (T)}_{\text{Loc,Data}} \) is SL parametrized by \( T \) and the choice of Loc and Data
Semantics of SL (T)

The satisfaction relation $\mathcal{I}, h \models_{\text{SL}} \phi$ is defined inductively:

- $\mathcal{I}, h \models_{\text{SL}} \text{emp} \iff h = \emptyset$
- $\mathcal{I}, h \models_{\text{SL}} t \mapsto u \iff h = \{(t^\mathcal{I}, u^\mathcal{I})\}$ and $t^\mathcal{I} \not\approx \text{nil}^\mathcal{I}$
- $\mathcal{I}, h \models_{\text{SL}} \phi_1 \ast \phi_2 \iff \exists h_1, h_2. h = h_1 \cup h_2$ and $\mathcal{I}, h_i \models_{\text{SL}} \phi_i$, for all $i = 1, 2$
- $\mathcal{I}, h \models_{\text{SL}} \phi_1 \rightarrow\phi_2 \iff \forall h' \text{ if } h' \not\approx h \text{ and } \mathcal{I}, h' \models_{\text{SL}} \phi_1 \text{ then } \mathcal{I}, h' \cup h \models_{\text{SL}} \phi_2$
- $\mathcal{I}, h \models_{\text{SL}} \exists x^\sigma. \phi \iff \mathcal{I}[x \leftarrow v], h \models_{\text{SL}} \phi \text{ for some } v \in \sigma^\mathcal{I}$

\[\phi_1 \ast \phi_2\]

\[\phi_1 \rightarrow\phi_2\]
Outline

1 Motivation
   - Why separation logic?
   - Inductive proofs
   - Syntax and semantics

2 The Bernays-Schönfinkel-Ramsey fragment of SL

3 An SMT procedure for $\exists^*\forall^*\text{SL}(T)$

4 Evaluation

5 Conclusions and future work

6 References
The Bernays-Schönfinkel-Ramsey fragment of SL

Given a theory $T$, with designated sorts Loc and Data, we denote by $\exists^*\forall^*\text{SL}(T)_{\text{Loc},\text{Data}}$ the set of formulae:

$$\exists x_1 \ldots \exists x_m \forall y_1 \ldots \forall y_n \cdot \varphi(x_1, \ldots, x_m, y_1, \ldots, y_n)$$

where $\varphi$ is a quantifier-free formula of $\text{SL}(T)_{\text{Loc},\text{Data}}$
Given a theory $T$, with designated sorts Loc and Data, we denote by $\exists^* \forall^* \text{SL}(T)_{\text{Loc}, \text{Data}}$ the set of formulae:

$$\exists x_1 \ldots \exists x_m \forall y_1 \ldots \forall y_n \cdot \varphi(x_1, \ldots, x_m, y_1, \ldots, y_n)$$

where $\varphi$ is a quantifier-free formula of $\text{SL}(T)_{\text{Loc}, \text{Data}}$

We consider the following cases (not exhaustive):

1. $\text{Loc} = U$, $\text{Data} = U^k$, $k \geq 1$, $U$ sort of the theory of equality $T = E$
The Bernays-Schönfinkel-Ramsey fragment of SL

Given a theory $T$, with designated sorts Loc and Data, we denote by $\exists^*\forall^*SL(T)_{\text{Loc,Data}}$ the set of formulae:

$$\exists x_1 \ldots \exists x_m \forall y_1 \ldots \forall y_n \cdot \varphi(x_1, \ldots, x_m, y_1, \ldots, y_n)$$

where $\varphi$ is a quantifier-free formula of $SL(T)_{\text{Loc,Data}}$

We consider the following cases (not exhaustive):

1. Loc = $U$, Data = $U^k$, $k \geq 1$, $U$ sort of the theory of equality $T = E$
2. Loc = $U$, Data = $U^k$, $k \geq 1$, $U$ is as above and $\|U\| = \aleph_0$
The Bernays-Schönfinkel-Ramsey fragment of SL

Given a theory $T$, with designated sorts Loc and Data, we denote by $\exists^*\forall^*\text{SL}(T)_{\text{Loc},\text{Data}}$ the set of formulae:

$$\exists x_1 \ldots \exists x_m \forall y_1 \ldots \forall y_n \cdot \varphi(x_1, \ldots, x_m, y_1, \ldots, y_n)$$

where $\varphi$ is a quantifier-free formula of $\text{SL}(T)_{\text{Loc},\text{Data}}$

We consider the following cases (not exhaustive):

1. Loc = $U$, Data = $U^k$, $k \geq 1$, $U$ sort of the theory of equality $T = E$
2. Loc = $U$, Data = $U^k$, $k \geq 1$, $U$ is as above and $\|U\| = \aleph_0$
3. Loc = Data = Int and $T = \text{LIA}$
The Bernays-Schönfinkel-Ramsey fragment of SL

Given a theory $T$, with designated sorts Loc and Data, we denote by $\exists^*\forall^*\text{SL}(T)_{\text{Loc,Data}}$ the set of formulae:

$$\exists x_1 \ldots \exists x_m \forall y_1 \ldots \forall y_n \cdot \varphi(x_1, \ldots, x_m, y_1, \ldots, y_n)$$

where $\varphi$ is a quantifier-free formula of $\text{SL}(T)_{\text{Loc,Data}}$

We consider the following cases (not exhaustive):

1. Loc = $U$, Data = $U^k$, $k \geq 1$, $U$ sort of the theory of equality $T = E$
2. Loc = $U$, Data = $U^k$, $k \geq 1$, $U$ is as above and $\|U\| = \aleph_0$
3. Loc = Data = Int and $T = \text{LIA}$
4. Loc = $U$, Data = $U \times \text{Int}$ and $T = \text{ELIA}$ is the union of $E$ and LIA
Lemma (Small Model Property)

\[ \forall \equiv \forall x_1^U \ldots \forall x_n^U \cdot \varphi(x_1^U, \ldots, x_n^U), \varphi \text{ quantifier-free SL(E) } U, U_k \text{-formula} \]

Then \( \forall \) has a model iff there exists \( \mathcal{I}, h \models_{SL} \forall \) such that \( \|U^\mathcal{I}\|, \|\text{dom}(h)\|, \|\text{img}(h)\| \) are linear with respect to the size of \( \varphi \).
### Lemma (Small Model Property)

\[ \varphi^\forall \equiv \forall x_1^U \ldots \forall x_n^U \cdot \varphi(x_1^U, \ldots, x_n^U), \ \varphi \text{ quantifier-free } \text{SL}(E)_{U, U^k}\text{-formula} \]

*Then* \( \varphi^\forall \) *has a model* iff there exists \( \mathcal{I}, h \models_{\text{SL}} \varphi^\forall *such that* \( \| U^\mathcal{I} \|, \| \text{dom}(h) \|, \| \text{img}(h) \| *are linear with respect to the size of* \( \varphi. \)

### The satisfiability problem for \( \exists^* \forall^* \text{SL}(E)_{U, U^k} \) is PSPACE-complete.

- **PSPACE-hardness** follows from the PSPACE-hardness of the quantifier-free fragment
- **PSPACE-membership** is because we can guess a model and enumerate all instantiations of the universal quantifiers using polynomially many bits
Lemma (Small Model Property)

\[ \phi^\forall \equiv \forall x_1^U \ldots \forall x_n^U \cdot \varphi(x_1^U, \ldots, x_n^U), \varphi \text{ quantifier-free } SL(E)_{U,U_k}\text{-formula} \]

Then \( \phi^\forall \) has a model iff there exists \( \mathcal{I}, h \models_{SL} \phi^\forall \) such that \( \|U^\mathcal{I}\|, \|\text{dom}(h)\|, \|\text{img}(h)\| \) are linear with respect to the size of \( \varphi \).

The satisfiability problem for \( \exists^* \forall^* \exists^* SL(E)_{U,U_k} \) is PSPACE-complete.

- PSPACE-hardness follows from the PSPACE-hardness of the quantifier-free fragment
- PSPACE-membership is because we can guess a model and enumerate all instantiations of the universal quantifiers using polynomially many bits

The satisfiability problem for \( \exists^* \forall^* \exists^* SL(E)_{U,U_k} \) is undecidable, when \( k \geq 2 \).
Countably infinite uninterpreted locations

Consider the definition [Brochenin, Demri, Lozes ’12]

\[ \text{alloc}(x) \equiv x \mapsto (x, \ldots, x) \rightarrow^* \bot \]
Countably infinite uninterpreted locations

Consider the definition [Brochenin, Demri, Lozes ’12]

$$\text{alloc}(x) \equiv x \mapsto (x, \ldots, x) \mapsto \bot$$

$$x \mapsto (x, \ldots, x) \quad \exists y_1 \ldots \exists y_k \cdot x \mapsto (y_1, \ldots, y_k) \mapsto \top$$
Countably infinite uninterpreted locations

Consider the definition [Brochenin, Demri, Lozes ’12]

\[ \text{alloc}(x) \equiv x \mapsto (x, \ldots, x) \rightarrow^* \bot \]

\[ x \mapsto (x, \ldots, x) \quad \exists y_1 \ldots \exists y_k \cdot x \mapsto (y_1, \ldots, y_k) \rightarrow^* \top \]

\[ \forall x \cdot x \not\approx \text{nil} \rightarrow \text{alloc}(x) \text{ is unsatisfiable if } \| U \| = \aleph_0 \]
Countably infinite uninterpreted locations

Consider the definition [Brochenin, Demri, Lozes ’12]

\[ alloc(x) \equiv x \mapsto (x, \ldots, x) \mapsto \perp \]

\[ x \mapsto (x, \ldots, x) \quad \exists y_1 \ldots \exists y_k \cdot x \mapsto (y_1, \ldots, y_k) \mapsto \top \]

\[ \forall x . \ x \not\approx \text{nil} \rightarrow \text{alloc}(x) \text{ is unsatisfiable if } \|U\| = \aleph_0 \]

Satisfiability of \( \exists^* \forall^* \text{SL}(E)_{U, U^k} \) is PSPACE-complete, when \( \|U\| = \aleph_0 \).

Reduction to \( \|U\| < \infty \) - If the interpretation of a free variable is
  \bullet not in the domain of the heap
  \bullet nor equal to the interpretation of some constant
then the chosen location does not matter.
Integer locations with linear arithmetic

Consider the definition [Brochenin, Demri, Lozes ’12]

$$\exists x \exists y. x \xrightarrow{\bigcirc}^+ y$$

a heap consisting of a list from x to y and several cycles
Integer locations with linear arithmetic

Consider the definition [Brochenin, Demri, Lozes ’12]

$$\exists x \exists y . x \xrightarrow{+} y \land \text{nocyc}$$

a heap consisting of a list of strictly increasing addresses from \(x\) to \(y\)
Integer locations with linear arithmetic

Consider the definition [Brochenin, Demri, Lozes ’12]

\[ \exists x \exists y . x \xrightarrow{+} y \land \text{nocyc} \land \text{consqr}(x) \]

heap consisting of a list of consecutive perfect squares 0, 1, 4, 9, …
Integer locations with linear arithmetic

Consider the definition [Brochenin, Demri, Lozes ’12]

\[ \exists x \exists y . x \xrightarrow{+} y \land \text{nocyc} \land \text{consqr}(x) \]

heap consisting of a list of consecutive perfect squares 0, 1, 4, 9, …

In this context:

\[ x = y^2 \equiv \text{alloc}(x) \land \text{alloc}(x + 2y + 1) \land \]
\[ \forall z . x < z < y + 2y + 1 \rightarrow \neg \text{alloc}(z) \]

\[ z = xy \equiv 4z = (x + y)^2 - (x - y)^2 \]
The satisfiability problem for $\exists^* \forall^* \text{SL}(\text{LIA})_{\text{Int}, \text{Int}}$ is undecidable.

By reduction from Hilbert’s 10th problem:

*Given a multi-variate Diophantine polynomial $R(x_1, \ldots, x_n)$, the problem “Does $R(x_1, \ldots, x_n) = 0$ have a solution in $\mathbb{N}$?” is undecidable.*
The satisfiability problem for $\exists^*\forall^*\text{SL}(\text{LIA})_{\text{Int,Int}}$ is undecidable.

By reduction from Hilbert's 10th problem:

*Given a multi-variate Diophantine polynomial $R(x_1, \ldots, x_n)$, the problem "Does $R(x_1, \ldots, x_n) = 0$ have a solution in $\mathbb{N}$?" is undecidable.*

The satisfiability problem for $\exists^*\forall^*\text{SL}(\text{ELIA})_{U,U \times \text{Int}}$ is undecidable.
An SMT procedure for $\exists^*\forall^*\text{SL}(T)$

solve($\exists x\forall y . \Psi(x, y)$)

Establishes satisfiability via cooperation between

- a ground solver for SL$(T)$ inputs
- an instantiation module for SL
An SMT procedure for $\exists^*\forall^*\text{SL}(T)$
\[\text{solve}(\exists x \forall y . \Psi(x, y))\]

\[\Gamma = \emptyset\]

- Equisatisfiable set $\Gamma$ (initially $\emptyset$) of quantifier-free SL$(T)$ formulae

Ground SL$(T)$ solver

SL inst. module
An SMT procedure for $\exists^* \forall^* \text{SL}(T)$
\[
\text{solve} (\exists x \forall y . \Psi(x, y))
\]

- Equisatisfiable set $\Gamma$ (initially $\emptyset$) of quantifier-free $\text{SL}(T)$ formulae
- Instantiation module computes a finite set of terms $L$ such that
\[
\exists x \forall y . \Psi(x, y) \iff \exists x . \bigwedge \{ \Psi(x, t) \mid t \in L \}
\]

$L$ can be built for fragments which satisfy the small model property
An SMT procedure for $\exists^* \forall^* \text{SL}(T)$
solve($\exists x \forall y . \Psi(x, y)$)

\[ \Gamma = \emptyset \]

Ground SL($T$) solver

SL inst. module

$\{t_1, \ldots, t_n\}$
An SMT procedure for $\exists^*\forall^*\text{SL}(T)$
solve($\exists x \forall y . \Psi(x, y)$)

$\Gamma = \emptyset$

Ground SL($T$) solver

unsat

$\Gamma \land \neg \Psi(x, y) \land y \in L$

SL inst. module

$L = \{t_1, \ldots, t_n\}$
An SMT procedure for $\exists^*\forall^*\text{SL}(T)$
solve($\exists x \forall y . \Psi(x, y)$)

$\Gamma = \emptyset$

Ground $\text{SL}(T)$ solver

$\Gamma \land \neg\Psi(x, y) \land y \in L$

Ground $\text{SL}(T)$ solver

SL inst. module

$L = \{t_1, \ldots, t_n\}$
An SMT procedure for $\exists^*\forall^*\text{SL}(T)$
solve($\exists x \forall y . \Psi(x, y)$)

$\Gamma = \emptyset$

If $\Gamma \land \neg \Psi(x, y) \land y \in L$ is unsat, then $\Gamma \land y \in L \rightarrow \Psi(x, y)$ is valid. Since $L$ contains all possible instantiations for $y$, the input is sat.
An SMT procedure for $\exists^* \forall^* \text{SL}(T)$

\[
\text{solve}(\exists x \forall y . \Psi(x, y))
\]

\[
\Gamma = \emptyset \cup \{\Psi(x, t_i)\}
\]

- **Ground SL($T$) solver**
  - $\Gamma \land \neg \Psi(x, y) \land y \in L$
  - Otherwise, add $\Psi(x, t_i)$ to $\Gamma$, where $y^M = t_i^M$, $t_i \in L$
An SMT procedure for $\exists^*\forall^*\text{SL}(T)$
solve($\exists x \forall y . \Psi(x, y)$)

\[ \Gamma = \emptyset \cup \{ \Psi(x, t_i) \} \]

\[ \Gamma \land \neg \Psi(x, y) \land y \in L \]

\[ L = \{ t_1, \ldots, t_n \} \]

- In the decidable case the procedure is sound, complete and terminates
  (only a finite number of possible formulae can be added to $\Gamma$)
An SMT procedure for $\exists^* \forall^* \text{SL}(T)$

$\text{solve}(\exists x \forall y . \psi(x, y))$

- $\Gamma = \emptyset \cup \{ \psi(x, t_i) \}$
- $\Gamma \land \neg \psi(x, y) \land y \in L$
- $L = \{ t_1, \ldots, t_n \}$

If we cannot build $L$ such that it contains all possible instantiations for $y$, the procedure cannot determine if the input is sat (unknown)

The procedure is still sound for unsat.
Outline

1 Motivation
   - Why separation logic?
   - Inductive proofs
   - Syntax and semantics

2 The Bernays-Schönfinkel-Ramsey fragment of SL

3 An SMT procedure for $\exists^*\forall^*\text{SL}(T)$

4 Evaluation

5 Conclusions and future work

6 References
Evaluation - Unfoldings of inductive predicates

- Pairs of inductive definitions for list segments, trees, tree segments, etc.

\[ \text{ts}(x, y) \equiv \text{emp} \land x = y \]
\[ \lor \ \exists l \exists r . \ x \mapsto (l, r) \ast \text{ts}(l, y) \ast \text{tree}(r) \]
\[ \lor \ \exists l \exists r . \ x \mapsto (l, r) \ast \text{tree}(l) \ast \text{ts}(r, y) \]

\[ \hat{\text{ts}}(x, y) \equiv \text{emp} \land x = y \]
\[ \lor \ \exists l \exists r . \ x \neq y \land x \mapsto (l, r) \ast \hat{\text{ts}}(l, y) \ast \text{tree}(r) \]
\[ \lor \ \exists l \exists r . \ x \neq y \land x \mapsto (l, r) \ast \text{tree}(l) \ast \hat{\text{ts}}(r, y) \]

- \( \text{ts}^n(x, y) \vdash \hat{\text{ts}}^n(x, y) \) where predicates are unfolded \( n = 1, 2, 3, 4, 8 \) times
Evaluation - Verification conditions

1: while $w \neq \text{nil}$ do
2: $v := w$;
3: $w := w\text{.next}$;
4: $\text{dispose}(v)$;
5: do

\[
\text{disp}
\]

\[
\text{list}^0(x) \equiv \text{emp} \land x = \text{nil} \\
\text{list}^n(x) \equiv \exists y . x \mapsto y \ast \text{list}^{n-1}(y)
\]

\[
\text{list}^n(w) \vdash \text{wp}(\text{disp}, \text{list}^{n-1}(w)) \\
\vdash \text{wp}^n(\text{disp}, \text{list}^0(w))
\]

where $n = 1, 2, 3, 4, 8$
Evaluation - Results

- 74 benchmarks
- 5 (6.75%) timeouts (>300s)
Evaluation - Results

- 74 benchmarks
- 5 (6.75%) timeouts (>300s)
- 64 (86.5%) under 5s
Evaluation - Results

- 74 benchmarks
- 5 (6.75%) timeouts (>300s)
- 64 (86.5%) under 5s
- 57 (77%) under 1.5s
Outline

1 Motivation
   - Why separation logic?
   - Inductive proofs
   - Syntax and semantics

2 The Bernays-Schönfinkel-Ramsey fragment of SL

3 An SMT procedure for $\exists^*\forall^*\text{SL}(T)$

4 Evaluation

5 Conclusions and future work

6 References
Conclusions

- Theoretical and practical results on the decidability of separation logic restricted to the $\exists^*\forall^*$ quantifier prefix
- Decision procedure using an effective counterexample-driven instantiation of the universal quantifiers
- Integrated in the main branch of CVC4 (version 1.5 prerelease)
- Open decidability problem: $\exists^*\forall^*\text{SL(ELIA)}_{U,\text{Int}}$
References

