Logical Investigations on Separation Logics

Day 5: Decision Procedures

Stéphane Demri (CNRS) & Morgan Deters (NYU)

ESSLLI 2015, Barcelona, August 2015
Plan of the talk

▶ Yesterday’s lecture:
  ▶ Undecidability proofs by reduction from data logics.
  ▶ Non-elementarity proof by reduction from propositional interval temporal logic.
  ▶ A modal logic of heap.

▶ SMT framework and translations
▶ Expressiveness of $k$SL0 and $k$PSPACE upper bound.
▶ Translations into first-order logic and QBF.
Plan of the talk

- Yesterday's lecture:
  - Undecidability proofs by reduction from data logics.
  - Non-elementarity proof by reduction from propositional interval temporal logic.
  - A modal logic of heap.

- SMT framework and translations

- Expressiveness of $k$SL0 and PSPACE upper bound.

- Translations into first-order logic and QBF.
Direct vs. translation approach

- Direct approach: specialised proof systems leading to fine-tuned tools.
  - Example: tableaux systems for modal logics. See e.g. [Gasquet et al., 2014]

- Translation approach: reduction to logics with well-established provers.
  - Example: translation into first-order logic. See e.g. [Ohlbach et al., 2001]
Similar dichotomy with separation logics

Direct approach:

★ Tableaux calculus for 2SL0.
  [Galmiche & Mery, JLC 10]

★ A proof system for the fragment SF.
  [Berdine & Calcagno & O’Hearn, FSTTCS’04]

★ Model-checking by abstraction.
  [Calcagno & Yang & O’Hearn, FSTTCS’01]

★ Graph-based (semantical) methods.
  [Cook et al., CONCUR’11]
  [Enea & Saveluc & Sighireanu, ESOP’13]

Translation approach:

★ Translation into FO or propositional calculus.
  [Lozes, SPACE’04; Calcagno & Gardner & Hague, FOSSACS’05]

★ Translation into theories handled by SMT solvers.
  See e.g. [Navarro P´erez & Rybalchenko, APLAS’13]
Similar dichotomy with separation logics

Direct approach:

- Tableaux calculus for 2SL0. [Galmiche & Mery, JLC 10]

- A proof system for the fragment SF. [Berdine & Calcagno & O’Hearn, FSTTCS’04]

- Model-checking by abstraction. [Calcagno & Yang & O’Hearn, FSTTCS’01]

- Graph-based (semantical) methods. [Cook et al., CONCUR’11]
  [Enea & Saveluc & Sighireanu, ESOP’13]

Translation approach:

- Translation into FO or propositional calculus. [Lozes, SPACE’04; Calcagno & Gardner & Hague, FOSSACS’05]

- Translation into theories handled by SMT solvers. See e.g. [Navarro Pérez & Rybalchenko, APLAS’13]
Translation into a Reachability Logic
A robust framework: using SMT solvers

- Convergence of SAT and theory reasoning: Satisfiability Modulo Theories (SMT).
- Mature and robust tools: CVC4, MathSAT, Yices, Z3, etc.
- SMT solvers at the heart of verification tools.
- Pointer logic is one of the new theories being developed.
- Recent works using SMT solvers for separation logic.
  - E.g., translating SLLB into the logic of graph reachability and stratified sets. [Piskac & Wies & Zufferey, CAV’13]
  - Tool Asterix [Navarro Pérez & A. Rybalchenko, APLAS’13]
- Ability to combine separation logics with other theories.
Satisfiability modulo theories

Problem: given a (quantifier-free) formula $\varphi$ and a combination of theories $\mathcal{T}_1 \cup \ldots \cup \mathcal{T}_n$, is there a variable assignment that satisfies $\varphi$?

Example:

$$x = y + 1 \land z < x \land f(x) = f(y + z)$$

A selection of theories supported by SMT solvers:

- Linear integer arithmetic LIA.
- Theory of equality over uninterpreted functions (EUF).
- Bit vectors (BV).
- Arrays (A).
Techniques for deciding SMT

- Use efficient SAT solvers for the Boolean skeleton of formulae and theory solvers for deciding conjunctions of literals.

- Decision procedure for combination of theories $\mathcal{T}_1 \cup \ldots \cup \mathcal{T}_n$ from the respective procedure for each $\mathcal{T}_i$. See e.g. [Nelson & Oppen, ACM 79]

- Handling quantifiers via axiom instantiation.
GRASS as a combination of two theories

- The logic GRASS combines
  1. a theory $T_G$ of reachability in function graphs,
  2. a theory $T_S$ of stratified sets.

- ... but extension with set comprehensions that define sets of locations in terms of properties stated in the logic.

- The exact definition fits well the notion of memory states and allows to get NP-completeness by using the state of the art of results in SMT technology.

- Reduction from separation logic SLL$^B$ into quantifier-free GRASS.
Syntactic objects

\[ T ::= x \mid f(T) \]

\[ A ::= T = T \mid T \xrightarrow{T} T \]

\[ R ::= A \mid \neg R \mid R \land R \]

\[ S ::= x \mid \emptyset \mid S \setminus S \mid S \cup S \mid S \cap S \mid \{x.R\} \]

proviso: \( x \) does not occur below \( f \) in \( R \)

\[ B ::= S = S \mid T \in S \]

\[ \varphi ::= A \mid B \mid \neg \varphi \mid \varphi \land \varphi \]

\[ f(f(x_1)) \in \{x. (x = x_1)\} \cap \{x'. (x' \xrightarrow{x_2} x_3)\} \]
GRASS-models

\[ A = (\mathcal{L}, \mathcal{h}) \]

1. \( \mathcal{L} \) is a non-empty set.

2. \( \mathcal{h} \) is a map \( \mathcal{L} \to \mathcal{L} \) satisfying the conditions below.
   - For all \( l \in \mathcal{L} \), the set \( \{ l' \mid (l, l') \in \mathcal{h}^* \} \) is finite.
   - For all \( l \in \mathcal{L} \), the set \( \{ l' \mid (l', l) \in \mathcal{h}^* \} \) is finite.

\[ \left[ x \right]_{A, f} \overset{\text{def}}{=} f(x) \quad \left[ f(T) \right]_{A, f} \overset{\text{def}}{=} h\left( \left[ T \right]_{A, f} \right) \]

(the variable assignment \( f \) plays the role of a store)

\[ \left[ X \right]_{A, f} \overset{\text{def}}{=} f(X) \]
\[ \left[ \emptyset \right]_{A, f} \overset{\text{def}}{=} \emptyset \]
\[ \left[ S_1 \circ S_2 \right]_{A, f} \overset{\text{def}}{=} \left[ S_1 \right]_{A, f} \circ \left[ S_2 \right]_{A, f} \quad (\circ \in \{ \setminus, \cup, \cap \}) \]
\[ \left[ \{ x . R \} \right]_{A, f} \overset{\text{def}}{=} \{ l \mid A \models_{f[x \mapsto l]} R \} \]

where \( \models \) is defined next.
Satisfaction relation

\[ \mathcal{A} \models_f T_1 = T_2 \] if \[ (\llbracket T_1 \rrbracket_{\mathcal{A},f}, \llbracket T_2 \rrbracket_{\mathcal{A},f}) \in \mathcal{R}^* \] where
\[ \mathcal{R} = \{ (l, h(l)) \mid l \in \mathcal{L}, l \neq \llbracket T_2 \rrbracket_{\mathcal{A},f} \} \]

\[ \mathcal{A} \models_f T_1 \rightarrow^T T_3 \] if \[ \mathcal{A} \models_f T_1 \rightarrow^T T_3 \]

\[ \mathcal{A} \models_f S_1 = S_2 \] if \[ \llbracket S_1 \rrbracket_{\mathcal{A}} = \llbracket S_2 \rrbracket_{\mathcal{A}} \]

\[ \mathcal{A} \models_f T \in S \] if \[ \llbracket T \rrbracket_{\mathcal{A},f} \in \llbracket S \rrbracket_{\mathcal{A},f} \]

\[ \mathcal{A} \models_f \neg \varphi \] if \[ \text{not } \mathcal{A} \models_f \varphi \]

\[ \mathcal{A} \models_f \varphi_1 \land \varphi_2 \] if \[ \mathcal{A} \models_f \varphi_1 \text{ and } \mathcal{A} \models_f \varphi_2. \]
NP-completeness of the satisfiability for GRASS

- Translation from GRASS into the logical theory $\mathcal{T}_{GS}$, the disjoint combination of
  1. a theory $\mathcal{T}_G$ of reachability in function graphs,
  2. a theory $\mathcal{T}_S$ of stratified sets.

- NP-completeness of the theory $\mathcal{T}_G$ [Totla & Wies, POPL’13].

- NP-completeness of $\mathcal{T}_{GS}$ by using
  1. the standard Nelson-Oppen combination of decision procedures for $\mathcal{T}_G$ and $\mathcal{T}_S$,
  2. the method for stratified sets from [Zarba, 2004].

- Translation by elimination of set comprehensions (introduction of set variables) and by finite instantiation of universal quantifiers.
  
  [Piskac & Wies & Zufferey, CAV’13; Bansal et al., CAV’15]
Separation logic SLL\(\mathbb{B}\)

\[ \varphi_s ::= (x_i = x_j) \mid \neg(x_i = x_j) \mid x_i \mapsto x_j \mid \text{sreach}(x_i, x_j) \mid \varphi_s \ast \varphi_s \]

SLL\(\mathbb{B}\) formulae: Boolean combinations of \(\varphi_s\)'s.
Separation logic SLL\textsuperscript{B}

\( \varphi_s ::= (x_i = x_j) \mid \neg (x_i = x_j) \mid x_i \mapsto x_j \mid \text{sreach}(x_i, x_j) \mid \varphi_s \ast \varphi_s \)

SLL\textsuperscript{B} formulae: Boolean combinations of \( \varphi_s \)'s.

\[ \begin{align*}
\mathcal{A} \models^X_f x_i = x_j & \quad \text{iff} \quad f(x_i) = f(x_j) \quad \text{and} \quad f(X) = \emptyset \\
\mathcal{A} \models^X_f \neg (x_i = x_j) & \quad \text{iff} \quad f(x_i) \neq f(x_j) \quad \text{and} \quad f(X) = \emptyset
\end{align*} \]
Separation logic SLL\(\mathcal{B}\)

\(\varphi_s ::= (x_i = x_j) | \neg(x_i = x_j) | x_i \mapsto x_j | \text{sreach}(x_i, x_j) | \varphi_s \ast \varphi_s\)

SLL\(\mathcal{B}\) formulae: Boolean combinations of \(\varphi_s\)’s.

\[
\begin{align*}
\mathcal{A} \models_f^X x_i = x_j & \quad \text{iff} \quad f(x_i) = f(x_j) \text{ and } f(X) = \emptyset \\
\mathcal{A} \models_f^X \neg(x_i = x_j) & \quad \text{iff} \quad f(x_i) \neq f(x_j) \text{ and } f(X) = \emptyset \\
\mathcal{A} \models_f^X x_i \mapsto x_j & \quad \text{iff} \quad h(f(x_i)) = f(x_j) \text{ and } f(X) = \{f(x_i)\} \\
\mathcal{A} \models_f^X \varphi_s^1 \ast \varphi_s^2 & \quad \text{iff} \quad \text{there are } X_1, X_2 \text{ s.t. } f(X) = X_1 \cup X_2, \\
\mathcal{A} \models_f^X \text{sreach}(x_i, x_j) & \quad \text{iff} \quad \text{either } f(x_i) = f(x_j) \text{ and } f(X) = \emptyset, \text{ or there is } n \geq 1 \text{ such that } h^n(f(x_i)) = f(x_j) \text{ and } f(X) = \{f(x_i), h(f(x_i)), \ldots, h^{n-1}(f(x_i))\}.
\end{align*}
\]
Properties

- Footprint of $\varphi$: $f(X)$ when $A \models^X \varphi$.

- Let $f, f'$ be assignments that differ at most for $X$. If $A \models^X \varphi_S$ and $A \models^X \varphi_s$ then $f = f'$ and $f(X)$ is finite.

- $(s, h)$ is encoded $(\mathbb{N}, h', f, x)$ where
  1. $(\mathbb{N}, h')$ is a GRASS-model,
  2. $s$ and $f$ agree on the program variables and $f(x) = \text{dom}(h)$,
  3. $h'$ restricted to $f(x)$ is equal to $h$. 
GRASS-FO

- GRASS-FO = GRASS + existential quantification.

- $A \models f \exists Y \varphi$ iff there is $X \subseteq L$ such that $A \models f[Y \mapsto X] \varphi$.

- Let $\varphi_1$ and $\varphi_2$ be GRASS-FO formulae so that $Y_1$ is not free in $\varphi_2$ and $Y_2$ is not free in $\varphi_1$. Then, $(\exists Y_1 \varphi_1) \circ (\exists Y_2 \varphi_2)$ is logically equivalent to $\exists Y_1, Y_2 (\varphi_1 \circ \varphi_2)$ with $\circ \in \{\land, \lor\}$. Classically, the quantifiers can be pushed outside.

- Given a GRASS-FO formula of the form $\exists Y \varphi$, we have $\varphi$ is satisfiable iff $\exists Y \varphi$ is satisfiable.
Translation from SLL\mathbb{B} to GRASS-FO

<table>
<thead>
<tr>
<th>Atomic formula $\varphi$</th>
<th>$\psi_1$</th>
<th>$\psi_2(Y)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_i = x_j$</td>
<td>$x_i = x_j$</td>
<td>$Y = \emptyset$</td>
</tr>
<tr>
<td>$x_i \neq x_j$</td>
<td>$x_i \neq x_j$</td>
<td>$Y = \emptyset$</td>
</tr>
<tr>
<td>$x_i \mapsto x_j$</td>
<td>$f(x_i) = x_j$</td>
<td>$Y = {y. (x_i = y)}$</td>
</tr>
<tr>
<td>sreach($x_i, x_j$)</td>
<td>$x_i \xrightarrow{x_j} x_j$</td>
<td>$Y = {x. (x_i \xrightarrow{x_j} x) \land x \neq x_j}$</td>
</tr>
</tbody>
</table>

- $tr$ is homomorphic for $\land$ and $\lor$.

- For $\text{sign} \in \{\neg, \varepsilon\}$:

$$tr(\text{sign}(\varphi_s^1 \cdots \varphi_s^n)) \overset{\text{def}}{=} \exists Y_1 \cdots Y_n \text{sign} ((\psi_1^1 \land \cdots \land \psi_1^n \land \bigwedge_{i \neq i'} Y_i \cap Y_{i'} = \emptyset) \land ((\psi_2^1(Y_1) \land \cdots \land \psi_2^n(Y_n) \land X = Y_1 \cup \cdots \cup Y_n),$$
Time to wrap-up

- $tr(\neg(\varphi^1_s \cdots \varphi^n_s))$ is logically equivalent to the negation of $tr(\varphi^1_s \cdots \varphi^n_s)$ even though both translations involve existential quantifications.

- $T(\varphi) \overset{\text{def}}{=} tr(\varphi) + \text{removal of all } \exists Y_1 \cdots Y_n$.

$$
T((x_1 \mapsto x_2) \land x_1 = x_3) =
(f(x_1) = x_2 \land Y_1 = \{y. (x_1 = y)\} \land X = Y_1) \land
(x_1 = x_3 \land Y_2 = \emptyset \land X = Y_2)
$$
Time to wrap-up

- $tr(\neg (\varphi_1 * \cdots * \varphi_n))$ is logically equivalent to the negation of $tr(\varphi_1 * \cdots * \varphi_n)$ even though both translations involve existential quantifications.

- $T(\varphi) \overset{\text{def}}{=} tr(\varphi) +$ removal of all $\exists Y_1 \cdots Y_n$.

  \[
  T((x_1 \mapsto x_2) \land x_1 = x_3) = \\
  (f(x_1) = x_2 \land Y_1 = \{y.(x_1 = y)\} \land X = Y_1) \land \\
  (x_1 = x_3 \land Y_2 = \emptyset \land X = Y_2)
  \]

- $\varphi$ be an $\text{SLLB}$ formula in NNF. Then $A \models_x \varphi$ iff $A \models f tr(\varphi)$.

- The satisfiability, validity and entailment problems for $\text{SLLB}$ are in NP. \cite{PiskacWiesZuffereyCAV13}

- Implementation and benchmarks following that reduction. \cite{PiskacWiesZuffereyTACAS14}
Expressiveness and $\text{PSPACE}$ upper bound
Roadmap

- To characterise the expressive power of 1SL0 by introducing the essential properties that can be stated.

- To design a symbolic model-checking algorithm by abstracting memory states.

- Polynomial-space satisfiability and model-checking problems for 1SL0 and polynomial-time fragments with $q$ fixed program variables.

- Translation into QBF by simulating nondeterministic step in the algorithm by propositional quantifications.
Sets of test formulae

- The set $\text{Test}(q, \alpha)$ contains the following formulae:
  1. $x_i = x_j$, $x_i \leftrightarrow x_j$ with $i, j \in [1, q]$.
  2. $\text{alloc}(x_i)$ with $i \in [1, q]$; \text{size} $\geq \beta$ with $\beta \in [0, \alpha]$.

- $\text{alloc}(x_i)$ and \text{size} $\geq \beta$ understood as atomic formulae.
Sets of test formulae

- The set $\text{Test}(q, \alpha)$ contains the following formulae:
  
  1. $x_i = x_j \land x_i \leftrightarrow x_j$ with $i, j \in [1, q]$.
  
  2. $\text{alloc}(x_i)$ with $i \in [1, q]$; $\text{size} \geq \beta$ with $\beta \in [0, \alpha]$.

- $\text{alloc}(x_i)$ and $\text{size} \geq \beta$ understood as atomic formulae.

- $\text{size}_{\overline{q}} \geq \beta$:
  
  $$\bigvee_{X \subseteq \{x_1, \ldots, x_q\}} (\bigwedge_{x \in X} \text{alloc}(x)) \land \text{Eq}(X) = \beta' \ldots$$

  $$\ldots (\bigwedge_{x \in \{x_1, \ldots, x_q\} \setminus X} \neg \text{alloc}(x)) \land \text{size} \geq (\beta' + \beta).$$

- $\text{Test}'(q, \alpha) \overset{\text{def}}{=} \text{Test}(q, \alpha)[\text{size}_{\overline{q}} \geq \beta / \text{size} \geq \beta]$. 
Playing with test formulae

- $(s, h)$ and $h \subseteq h'$.

- For all $\psi \in Test'(q, \alpha)$, $(s, h) \models \psi$ implies $(s, h') \models \psi$.
Playing with test formulae

- \((s, h)\) and \(h \subseteq h'\).

- For all \(\psi \in \text{Test}'(q, \alpha)\), \((s, h) \models \psi\) implies \((s, h') \models \psi\).

- \((s, h) \approx^q_{\alpha} (s', h')\) iff the memory states agree on the formulae in \(\text{Test}'(q, \alpha)\).

- We use \(\text{size}_q \geq \beta\) instead of \(\text{size} \geq \beta\).
Playing with test formulae

- $(s, h)$ and $h \subseteq h'$. 

- For all $\psi \in \text{Test}'(q, \alpha)$, $(s, h) \models \psi$ implies $(s, h') \models \psi$. 

- $(s, h) \approx^q_{\alpha} (s', h')$ iff the memory states agree on the formulae in $\text{Test}'(q, \alpha)$.

- We use $\text{size}_q \geq \beta$ instead of $\text{size} \geq \beta$. 

\[
\approx^q_{\alpha+1} \subseteq \approx^q_{\alpha} \quad \approx^{q+1}_{\alpha} \subseteq \approx^q_{\alpha}
\]
Distributivity Lemma

\[ \alpha = \alpha_1 + \alpha_2 \text{ and } (s, h) \approx^q (s', h'). \]

If \( h = h_1 \uplus h_2 \), then there are \( h'_1, h'_2 \) such that

\[ h' = h'_1 \uplus h'_2, \]

\[ (s, h_1) \approx^q_{\alpha_1} (s', h'_1), \]

\[ (s, h_2) \approx^q_{\alpha_2} (s', h'_2). \]
Proof (1/3)

- Since \((s, h) \approx^q_{\alpha} (s', h')\), for all \(i \in [1, q]\), \(s(x_i) \in \text{dom}(h)\) iff \(s'(x_i) \in \text{dom}(h')\).

- Let \(C_q(s, h)\) be the cardinal of the set
  \[
  \text{dom}(h) \setminus \{s(x_1), \ldots, s(x_q)\}
  \]
  We have \(C_q(s, h) = C_q(s, h_1) + C_q(s, h_2)\).

- Since \((s, h) \approx^q_{\alpha} (s', h')\), \(\min(\alpha, C_q(s, h)) = \min(\alpha, C_q(s', h'))\).
Proof (2/3)

- If \( s(x_i) \in \text{dom}(h_1) \), then \( s'(x_i) \in \text{dom}(h'_1) \) by definition.

- If \( s(x_i) \in \text{dom}(h_2) \), then \( s'(x_i) \in \text{dom}(h'_2) \) by definition.

- **Case** \( C_q(s, h) \leq \alpha \).
  - Since \( (s, h) \approx_q^{\alpha} (s', h') \), we have \( C_q(s', h') = C_q(s, h) \leq \alpha \) too.

- \( \beta_1 \overset{\text{def}}{=} C_q(s, h_1) \).

  - Since \( C_q(s', h') = C_q(s, h) \), and \( \beta_1 \leq C_q(s', h') \), there are \( \beta_1 \) locations \( l_1, \ldots, l_{\beta_1} \) in \( \text{dom}(h') \setminus \{s'(x_1), \ldots, s'(x_q)\} \).

- Then \( \{l_1, \ldots, l_{\beta_1}\} \subseteq \text{dom}(h'_1) \) by definition.
Case $C_q(s, h) > \alpha$, $C_q(s, h_1) < \alpha_1$ and $C_q(s, h_2) > \alpha_2$.

- $\beta_1 \overset{\text{def}}{=} C_q(s, h_1)$.

- Since $\min(\alpha, C_q(s, h)) = \min(\alpha, C_q(s', h')) = \alpha$, $\beta_1 \leq \alpha$ and $\beta_1 \leq \min(\alpha, C_q(s', h'))$, let $l_1 < \cdots < l_{\beta_1}$ be the $\beta_1$ smallest locations in $\text{dom}(h') \setminus \{s'(x_1), \ldots, s'(x_q)\}$.

- Then $\{l_1, \ldots, l_{\beta_1}\} \subseteq \text{dom}(h'_1)$ by definition.

The two missing cases are similar and the constructed subheaps satisfy the stated properties.
Compositionality Lemma

- $\alpha \in \mathbb{N}$ and $(s, h) \approx^q_{\alpha} (s', h')$.

- For any heap $h_1 \perp h$, there is a $h'_1 \perp h'$ such that
  
  1. $(s, h_1) \approx^q_{\alpha} (s', h'_1)$.
  
  2. $(s, h \sqcup h_1) \approx^q_{\alpha} (s', h' \sqcup h'_1)$.

  3. $\maxval(s', h'_1) \leq \maxval(s', h') + \alpha$.

  ($\maxval(s, h) \overset{\text{def}}{=} \max(\text{ran}(s) \cup \text{dom}(h) \cup \text{ran}(h))$)

- If we give up the condition 3, the condition 1 can be strengthened by $\text{card}(\text{dom}(h_1)) = \text{card}(\text{dom}(h'_1))$. 
Memory size

\[ \text{msize}(x_i \leftrightarrow x_{i'}) \quad \overset{\text{def}}{=} \quad 1 \]

\[ \text{msize}(x_i = x_{i'}) \quad \overset{\text{def}}{=} \quad 0 \]

\[ \text{msize}(\text{emp}) \quad \overset{\text{def}}{=} \quad 1 \]

\[ \text{msize}(\neg \psi) \quad \overset{\text{def}}{=} \quad \text{msize}(\psi) \]

\[ \text{msize}(\psi_1 \land \psi_2) \quad \overset{\text{def}}{=} \quad \max(\text{msize}(\psi_1), \text{msize}(\psi_2)) \]

\[ \text{msize}(\psi_1 \ast \psi_2) \quad \overset{\text{def}}{=} \quad \text{msize}(\psi_1) + \text{msize}(\psi_2) \]

\[ \text{msize}(\psi_1 \ast \neg \psi_2) \quad \overset{\text{def}}{=} \quad \max(\text{msize}(\psi_1), \text{msize}(\psi_2)) \]
Memory size as a refinement of the formula size

\[ \text{msize}(\text{size} = 3) = 4 \text{ with } \text{size} = 3 \text{ equal to} \]

\[ (((\neg\text{emp})\ast(\neg\text{emp})\ast(\neg\text{emp}))\land\neg((\neg\text{emp})\ast(\neg\text{emp})\ast(\neg\text{emp})\ast(\neg\text{emp})) \]

\[ \text{For every } \varphi \text{ in } 1\text{SL0, msize}(\varphi) \text{ is smaller than the size of } \varphi, \text{ assuming that formulae are encoded as trees.} \]
\( \varphi \) is blind over \( \text{Test'}(q, \alpha) \)

- \( \varphi \) built over \( x_1, \ldots, x_q \).

- \( \text{msize}(\varphi) \leq \alpha \) and \( (s, h) \approx^q (s', h') \).

- Then, we have \( (s, h) \models \varphi \) iff \( (s', h') \models \varphi \).

- Proof by structural induction, sufficient to establish one direction.

- The base case with atomic formulae and the cases with Boolean connectives in the induction step are by an easy verification.
Proof for the case $\psi = \psi_1 \ast \psi_2$ (1/2)

- Suppose that $\text{msize}(\psi_1 \ast \psi_2) = \max(\text{msize}(\psi_1), \text{msize}(\psi_2)) \leq \alpha$ and $(s, h) \models \psi_1 \ast \psi_2$.

- Let us prove that $(s', h') \models \psi_1 \ast \psi_2$.

- Let $h'_1 \perp h'$ such that $(s', h'_1) \models \psi_1$.

- By the Compositionality Lemma, there is $h_1 \perp h$ such that $(s, h_1) \approx^q \alpha (s', h'_1)$ and $(s, h \cup h_1) \approx^q \alpha (s', h' \cup h'_1)$.
Proof for the case $\psi = \psi_1 \ast \psi_2$ (2/2)

- By (IH), $(s, h_1) \models \psi_1$.
- Since $(s, h) \models \psi_1 \ast \psi_2$, this implies that $(s, h \sqcup h_1) \models \psi_2$.
- By (IH), we conclude that $(s', h' \sqcup h'_1) \models \psi_2$.
- Since $h'_1$ is an arbitrary disjoint heap from $h'$, we obtain $(s', h') \models \psi_1 \ast \psi_2$. 
Expressive power

- $\varphi$ built over the variables in $x_1, \ldots, x_q$.

- $\varphi$ is equivalent to a Boolean combination of test formulae from $\text{Test}(q, q + m\text{size}(\varphi))$.

$$\text{LIT}(s, h) \overset{\text{def}}{=} \{ \chi \in \text{Test}^\prime(q, \alpha) \mid (s, h) \models \chi \} \cup \{ \neg \chi \mid (s, h) \not\models \chi \text{ with } \chi \in \text{Test}^\prime(q, \alpha) \}$$

- $\psi^\prime \overset{\text{def}}{=} \bigvee \left( \bigwedge \psi \right)$

  $(s, h) \models \varphi \ \psi \in \text{LIT}(s, h)$

- $\psi^\prime$ admits an equivalent (finite) formula $\varphi^\prime$ since $\text{LIT}(s, h)$ can take only a finite amount of values.

- $\text{size}_q \geq \beta$ with $\beta \leq \alpha$ is equivalent to a Boolean combination of test formulae from $\text{Test}(q, q + \alpha)$. 
Small heap property

Let $\varphi$ be a satisfiable 1SL0 formula built over $x_1, \ldots, x_q$.

There is a memory state $(s, h)$ such that

$$ (s, h) \models \varphi \text{ and } \text{maxval}(s, h) \leq q + \text{msize}(\varphi). $$
Symbolic memory state $\text{sms} = (P, A, H, n)$ over $(q, \alpha)$

- $P$ is a partition of $\{x_1, \ldots, x_q\}$.
- $A \subseteq P$.
- $H$ is a functional relation on $P$ such that $\text{dom}(H) = A$.
- $n \in [0, \alpha]$.
Abstraction

$\text{Symb}[s, h]$ over $(q, \alpha)$ is equal to $(P, A, H, n)$

- $n = \min(\alpha, \text{card}(\text{dom}(h) \setminus \{s(x_i) \mid i \in [1, q]\}))$.

- $P$ is a partition of $\{x_1, \ldots, x_q\}$ so that for all $x, x'$, we have $s(x) = s(x')$ iff $x$ and $x'$ belong to the same set in $P$.

- $A = \{X \in P \mid \text{there is } x \in X, s(x) \in \text{dom}(h)\}$.

- $X H X'$ iff there are $x \in X$ and $x' \in X'$ such that $h(s(x)) = s(x')$.

- Main property: $(s, h) \approx_\alpha (s', h')$ iff $\text{Symb}[s, h] = \text{Symb}[s', h']$. 


Symbolic separating conjunction

\[ *_s(sms, sms_1, sms_2) \] whenever there exist a store \( s \) and disjoint heaps \( h_1 \) and \( h_2 \) such that

\[ \text{Symb}[s, h_1 \cup h_2] = sms. \]
\[ \text{Symb}[s, h_1] = sms_1. \]
\[ \text{Symb}[s, h_2] = sms_2. \]

\[ *_s(sms, sms_1, sms_2) \] is easy to verify.

\[ P = P_1 = P_2. \]
\[ A = A_1 \cup A_2, A_1 \cap A_2 = \emptyset \text{ and } H = H_1 \cup H_2. \]
\[ n = \min(\alpha, n_1 + n_2). \]
Model-checking algorithm

- Memory states are replaced by symbolic memory states.

- Algorithm is similar to nondeterministic procedures for modal logics, see e.g. [Ladner, SIAM 77].

- \( \text{MC}(\text{sms}, \varphi) = \top \) iff there is \((s, h)\) such that \(\text{Symb}[s, h] = \text{sms} \text{ and } (s, h) \models \varphi\).
1. if $\psi$ is atomic then return $\text{AMC}(\text{sms}, \psi)$;

2. if $\psi = \neg \psi_1$ then return not $\text{MC}(\text{sms}, \psi_1)$;

3. if $\psi = \psi_1 \land \psi_2$ then return $(\text{MC}(\text{sms}, \psi_1)$ and $\text{MC}(\text{sms}, \psi_2))$;

4. if $\psi = \psi_1 \ast \psi_2$ then return $\top$ iff there are $\text{sms}_1$ and $\text{sms}_2$ such that $\text{MC}(\text{sms}, \text{sms}_1, \text{sms}_2)$ and $\text{MC}(\text{sms}_1, \psi_1) = \text{MC}(\text{sms}_2, \psi_2) = \top$;

5. if $\psi = \psi_1 \ast \psi_2$ then return $\bot$ iff for some $\text{sms}'$ and $\text{sms}''$ such that $\text{MC}(\text{sms}''', \text{sms}', \text{sms})$, $\text{MC}(\text{sms}', \psi_1) = \top$ and $\text{MC}(\text{sms}''', \psi_2) = \bot$;
AMC(sms, $\psi$)

1. **if** $\psi$ is $\text{emp}$ **then** return $\top$ iff $A = \emptyset$ and $n = 0$;

2. **if** $\psi$ is $x_i = x_j$ **then** return $\top$ iff $x_i, x_j \in X$, for some $X \in P$;

3. **if** $\psi$ is $x_i \hookrightarrow x_j$ **then** return $\top$ iff $(X, X') \in H$ where $x_i \in X \in P$ and $x_j \in X' \in P$;
Model-checking and satisfiability problems for 1SL0 are in \textsc{PSPACE}. \cite{Calcagno2001}

For satisfiability, guess $\mathit{sms}$ over $(q, \mathit{msize}(\varphi))$ and check whether $\mathit{MC}(\mathit{sms}, \varphi) = \top$.

$\mathit{MC}(\mathit{sms}, \varphi)$ runs in space $\mathcal{O}(d(q + \log(\alpha)))$ where $d$ is the depth of syntactic tree for $\varphi$.

For model-checking, $(s, h) \models \varphi$ iff $\mathit{MC}(\mathit{Symb}[s, h], \varphi) = \top$ with $\alpha = \mathit{msize}(\varphi)$.

\textsc{PSPACE} upper bound even when the formulae are encoded as DAGs.
By-products

- Computing a Boolean combination of atomic formulae from $\text{Test}(q, q + \text{msize}(\varphi))$ equivalent to $\varphi$ can be done in polynomial space.

\[
\bigvee\{\bigwedge_{\psi \in \text{LIT}(s, h)} \psi \mid \text{MC}(\text{Symb}[s, h], \varphi) = \top \text{ and maxval}(s, h) \leq q + \alpha\}
\]
By-products

- Computing a Boolean combination of atomic formulae from Test\((q, q + m\text{size}(\varphi))\) equivalent to \(\varphi\) can be done in polynomial space.

\[
\bigvee\left\{\left(\bigwedge_{\psi \in \text{LIT}(s, h)} \psi\right) \mid \text{MC}(\text{Symb}[s, h], \varphi) = \top \text{ and } \maxval(s, h) \leq q + \alpha\right\}
\]

- The satisfiability problem for 1SL0 restricted to formulae with at most \(q\) program variables is in PTIME.

- The number of symbolic memory states over \((q, m\text{size}(\varphi))\) is polynomial in the size of \(\varphi\).
By-products

▶ Computing a Boolean combination of atomic formulae from $\text{Test}(q, q + \text{msize}(<\varphi>)$ equivalent to $\varphi$ can be done in polynomial space.

\[
\bigvee\{\bigwedge_{\psi \in \text{LIT}(s, h)} \psi \mid \text{MC}(\text{Symb}[s, h], \varphi) = \top \text{ and maxval}(s, h) \leq q + \alpha\}
\]

▶ The satisfiability problem for 1SL0 restricted to formulae with at most $q$ program variables is in $\text{PTIME}$.

▶ The number of symbolic memory states over $(q, \text{msize}(<\varphi>))$ is polynomial in the size of $\varphi$.

▶ Dynamic programming with $A[sms, \psi] \in \{\text{unknown}, \top, \bot\}$ to compute $\text{MC}(sms, \varphi)$. 
Translation into QBF
Now, let us encode symbolic memory states !!

- Atomic propositions encoding ($P, A, H, n$):
  - $EQ(i, j)$ \quad \quad \quad \quad i, j \in [1, q]
  - $A(i)$ \quad \quad \quad \quad i \in [1, q]
  - $H(i, j)$ \quad \quad \quad i, j \in [1, q]
  - $N(\beta)$ \quad \quad \quad \beta \in [0, \alpha]

- $Symb[v]$: unique symbolic memory state (if it is defined) encoded by the propositional valuation $v$.

- Typically, $Symb[v]$ is undefined when $v(EQ(1, 1)) = \perp$. 
Encoding symbolic separation

- There is $\text{SMS}(x)$ built over $x$ such that for all $v$, we have $v \models \text{SMS}(x)$ iff there is $\text{sms}$ such that $\text{Symb}[v] = \text{sms}$.

- There is $\ast_p(x, x', x'')$ such that for all $v$, we have $v \models \ast_p(x, x', x'')$ iff there are $\text{sms}$, $\text{sms}'$ and $\text{sms}''$ such that $\ast_s(\text{sms}, \text{sms}', \text{sms}'')$ and $\text{Symb}[v] = (\text{sms}, \text{sms}', \text{sms}'')$.

- Examples of conjuncts in $\ast_p(x, x', x'')$:
  1. $x$, $x'$ and $x''$ encode a symbolic memory state:
     \[ \text{SMS}(x) \land \text{SMS}(x') \land \text{SMS}(x'') \]
  2. Encoding of ‘$P = P_1 = P_2$’:
     \[ \bigwedge_{i,j \in [1, q]} (\text{EQ}(i, j) \Leftrightarrow \text{EQ}'(i, j)) \land (\text{EQ}'(i, j) \Leftrightarrow \text{EQ}''(i, j)). \]
  3. Encoding of ‘$n = \max(\alpha, n_1 + n_2)$’:
     \[ \bigwedge_{\beta, \beta' \in [0, \alpha]} (N'(\beta) \land N''(\beta')) \Rightarrow N(\min(\alpha, \beta + \beta')). \]
Principles for simulating the model-checking algorithm

- Quantifications in the model-checking algorithm are substituted by corresponding quantifications in QBF.
- Alternatively, the quantifications in the semantics are internalised in QBF.
- ...but the small heap property is used.
- Translation is performed in logarithmic space.
Translation

\( tr(\text{emp}, X) \quad \overset{\text{def}}{=} \quad N(0) \land \neg A(1) \land \cdots \land \neg A(q) \)

\( tr(x_i \mapsto x_j, X) \quad \overset{\text{def}}{=} \quad H(i, j) \)

\( tr(x_i = x_j, X) \quad \overset{\text{def}}{=} \quad EQ(i, j) \)

\( tr(\psi_1 \ast \psi_2, X) \quad \overset{\text{def}}{=} \quad \exists X', X'' \ast_p (X, X', X'') \land tr(\psi_1, X') \land tr(\psi_2, X'') \)

\( tr(\psi_1 \ast\ast \psi_2, X) \quad \overset{\text{def}}{=} \quad \forall X'', (\exists X' \ast_p (X'', X, X')) \implies \\
(\exists X' \ast_p (X'', X, X') \land (tr(\psi_1, X') \implies tr(\psi_2, X''))) \)

(\( \exists \{p_1, \ldots, p_m\} \psi \) is a shortcut for \( \exists p_1, \ldots, \exists p_m \psi \))
Translation into FO regained!

- $\exists X \; SMS(X) \land tr(\varphi, X)$ is QBF satisfiable iff there is $sms$ over $(q, \alpha)$ such that $MC(sms, \varphi) = T$.

- Separation logic 1SL0 can be decided by QBF solvers. See e.g. [Lonsing & Biere, JS 10]
Translation into FO regained!

- \( \exists x \ SMS(x) \land tr(\varphi, x) \) is QBF satisfiable iff there is \( sms \) over \((q, \alpha)\) such that \( MC(sms, \varphi) = T \).

- Separation logic 1SL0 can be decided by QBF solvers. See e.g. [Lonsing & Biere, JS 10]

- Translation into FO restricted to equality predicate: [Calcagno & Gardner & Hague, FOSSACS’05]

\[
\exists x_0, x_1 \ (x_0 \neq x_1) \land tr'(\chi)
\]

\[
tr'(\exists p \ \psi) \quad \text{def} \quad \exists x_p \ (x_p = x_0 \lor x_p = x_1) \land tr'(\psi)
\]

\( x_p \) is a fresh variable

\[
tr'(\forall p \ \psi) \quad \text{def} \quad \forall x_p \ (x_p = x_0 \lor x_p = x_1) \Rightarrow tr'(\psi)
\]

\( x_p \) is a fresh variable

\[
tr'(p) \quad \text{def} \quad (x_p = x_1)
\]
Concluding remarks for Day 5

Today’s lecture:

- Translation of SLL\(\mathcal{B}\) into the logic GRASS.
- Expressiveness of 1SL0.
- PSPACE algorithm.
- Translation into QBF and FO.

We have illustrated two proof techniques:

1. How to possibly tame the magic wand operator.
2. Translation into first-order logics.
2SL, undec.

1SL \equiv 1DSOL \equiv 1WSOL \equiv 1SL(\neg \ast), \text{ undec.}

1SL2, undec. 1SL(\ast), \text{ dec., non-elem.}

1SL1, PSPACE-C 1SL2(\neg \ast) \equiv 1DSOL, \text{ undec.} 1SL2(\ast), \text{ non-elem.}

1SL0, PSPACE-C

1SLk = 1SL \text{ restricted to } k \text{ quantified variables}
Further topics

- Program verification using separation logics
  See e.g. [Calcagno et al., NFM’15]
Further topics

- Program verification using separation logics
  See e.g. [Calcagno et al., NFM’15]

- Reasoning tasks with inductive predicates
  See e.g. [Brotherston et al. CSL-LICS’14]
Further topics

- Program verification using separation logics
  See e.g. [Calcagno et al., NFM’15]

- Reasoning tasks with inductive predicates
  See e.g. [Brotherston et al. CSL-LICS’14]

- Computational complexity of more fragments
  See e.g. [Haase et al., CONCUR’11]
A few current trends

▶ Even more translations into SMT solvers.
   See e.g. [Navarro Pérez & Rybalchenko, APLAS’13]
A few current trends

- Even more translations into SMT solvers. See e.g. [Navarro Pérez & Rybalchenko, APLAS’13]

- Reasoning about data. See e.g. [Enea et al., ESOP’13]
Slides and lecture notes

http://www.lsv.fr/~demri/esslli15-course.html