Quantifier Elimination in Separation Logic: One Quantified Variable Fragment

S. Demri

Joint work with D. Galmiche, D. Larchey-Wendling, D. Mery

NYU & CNRS – Marie Curie Fellow

February 2014

Overview

1. Separation Logic in a Nutshell
2. Separation Logic 1SL1
3. Expressive Completeness
4. Computational Complexity

Separation logic

- Introduced by Ishtiaq, Reynolds, O'Hearn, Pym.
- Extension of Hoare Logic by J.C. Reynolds with separating connectives.
- Reasoning about the heap with a strong form of locality built-in.
- $\phi * \psi$ is true whenever the heap can be divided into two disjoint parts, one satisfies ϕ, the other one ψ.
- $\phi \mathrel{-\!\!*} \psi$ is true whenever ϕ is true for a (fresh) disjoint heap, ψ is true for the combined heap.

Hoare triples

- Hoare triple: $\{\phi\}\ \texttt{PROG}\ \{\psi\}$ (total correctness).

- Rule of constancy:

$$\frac{\{\phi\}\ \texttt{PROG}\ \{\psi\}}{\{\phi \wedge \psi'\}\ \texttt{PROG}\ \{\psi \wedge \psi'\}}$$

  where no variable free in ψ' is modified by $\texttt{PROG}$.

- Unsoundness of the rule of constancy with pointers:

$$\frac{\{(\exists \texttt{z}.\ \texttt{x} \mapsto \texttt{z})\}\ [\texttt{x}] := 4\ \{\texttt{x} \mapsto 4\}}{\{(\exists \texttt{z}.\ \texttt{x} \mapsto \texttt{z}) \wedge \texttt{y} \mapsto 3\}\ [\texttt{x}] := 4\ \{\texttt{x} \mapsto 4 \wedge \texttt{y} \mapsto 3\}}$$

  (when $\texttt{x} = \texttt{y}$)
  $\texttt{x} \mapsto \texttt{z}$: "memory has a unique memory cell $\texttt{x} \mapsto \texttt{z}$"

When separation logic enters into the play

- Reparation with frame rule:

$$\frac{\{\phi\}\ \texttt{PROG}\ \{\psi\}}{\{\phi * \psi'\}\ \texttt{PROG}\ \{\psi * \psi'\}}$$

  where no variable free in ψ' is modified by $\texttt{PROG}$.

- Strengthening precedent (SP)

$$\frac{\phi \Rightarrow \psi' \quad \{\psi'\}\ \texttt{PROG}\ \{\psi\}}{\{\phi\}\ \texttt{PROG}\ \{\psi\}}$$

- Checking entailment/validity/satisfiability in separation logic is a building block of the verification process.

Standard inference rules for mutation

- Local form (MUL)

$$\frac{}{\{(\exists \mathtt{z}.\ \mathtt{x} \mapsto \mathtt{z})\}\ [\mathtt{x}] := \mathtt{y}\ \{\mathtt{x} \mapsto \mathtt{y}\}}$$

- Global form (MUG)

$$\frac{}{\{(\exists \mathtt{z}.\ \mathtt{x} \mapsto \mathtt{z}) * \phi\}\ [\mathtt{x}] := \mathtt{y}\ \{\mathtt{x} \mapsto \mathtt{y} * \phi\}}$$

- Backward-reasoning form (MUBR)

$$\frac{}{\{(\exists \mathtt{z}.\ \mathtt{x} \mapsto \mathtt{z}) * ((\mathtt{x} \mapsto \mathtt{y}) \mathrel{-\!\!*} \phi)\}\ [\mathtt{x}] := \mathtt{y}\ \{\phi\}}$$

Memory states for 2SL (two record fields)

- Program variables $\mathtt{PVAR} = \{\mathtt{x}_1, \mathtt{x}_2, \mathtt{x}_3, \ldots\}$.
- Memory state:
  - Store $\mathfrak{s} : \mathtt{PVAR} \to \mathtt{Val}$.
  - Heap $\mathfrak{h} : \mathtt{Loc} \rightharpoonup \mathtt{Val} \times \mathtt{Val}$ with finite domain.

  ($\mathtt{Loc} = \{\mathfrak{l}, \mathfrak{l}', \ldots\}$, $\mathtt{Val} = \mathbb{N} \uplus \mathtt{Loc} \uplus \{\mathit{nil}\}$)
- Simplification: $\mathtt{Loc} = \mathtt{Val} = \mathbb{N}$.
- Disjoint heaps: $\mathrm{dom}(\mathfrak{h}_1) \cap \mathrm{dom}(\mathfrak{h}_2) = \varnothing$ (noted $\mathfrak{h}_1 \perp \mathfrak{h}_2$).
- When $\mathfrak{h}_1 \perp \mathfrak{h}_2$, $\mathfrak{h}_1 \uplus \mathfrak{h}_2 \stackrel{\text{def}}{=} \mathfrak{h}_1 \uplus \mathfrak{h}_2$.

Syntax and semantics for 2SL

- Quantified variables $\mathtt{FVAR} = \{\mathtt{u}_1, \mathtt{u}_2, \mathtt{u}_3, \ldots\}$.
- Expressions: $e ::= \mathtt{x}_i \mid \mathtt{u}_j$
- Atomic formulae: $\pi ::= e = e' \mid e \hookrightarrow e', e'' \mid \mathtt{emp}$
- Formulae in 2SL

$$\phi ::= \bot \mid \pi \mid \phi \wedge \psi \mid \neg\phi \mid \phi * \psi \mid \phi \mathrel{-\!\!*} \psi \mid \exists\, \mathtt{u}_j\, \phi$$

- $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{f}} \mathtt{emp} \overset{\text{def}}{\Leftrightarrow} \mathrm{dom}(\mathfrak{h}) = \varnothing$.
- $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{f}} e = e' \overset{\text{def}}{\Leftrightarrow} [e] = [e']$, with $[\mathtt{x}_i] \overset{\text{def}}{=} \mathfrak{s}(\mathtt{x}_i)$ and $[\mathtt{u}_j] \overset{\text{def}}{=} \mathfrak{f}(\mathtt{u}_j)$.
- $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{f}} e \hookrightarrow e', e'' \overset{\text{def}}{\Leftrightarrow} \mathfrak{h}([e]) = ([e'], [e''])$.
- $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{f}} \phi_1 * \phi_2 \overset{\text{def}}{\Leftrightarrow} \mathfrak{h} = \mathfrak{h}_1 \uplus \mathfrak{h}_2$, $(\mathfrak{s}, \mathfrak{h}_1) \models_{\mathfrak{f}} \phi_1$, $(\mathfrak{s}, \mathfrak{h}_2) \models_{\mathfrak{f}} \phi_2$ for some $\mathfrak{h}_1, \mathfrak{h}_2$.

Semantics for 2SL

- $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{f}} \phi_1 \mathrel{-\!\!*} \phi_2 \overset{\text{def}}{\Leftrightarrow}$ for all $\mathfrak{h}'$, if $\mathfrak{h} \perp \mathfrak{h}'$ and $(\mathfrak{s}, \mathfrak{h}') \models_{\mathfrak{f}} \phi_1$ then $(\mathfrak{s}, \mathfrak{h} \uplus \mathfrak{h}') \models_{\mathfrak{f}} \phi_2$.

- $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{f}} \exists\, \mathtt{u}_j\, \phi \overset{\text{def}}{\Leftrightarrow}$ there is $\mathfrak{l} \in \mathbb{N}$ such that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{f}[\mathtt{u}_j \mapsto \mathfrak{l}]} \phi$ where $\mathfrak{f}[\mathtt{u}_j \mapsto \mathfrak{l}]$ is the assignment equal to $\mathfrak{f}$ except that $\mathtt{u}_j$ takes the value $\mathfrak{l}$.

- Satisfiability problem:

  input: formula ϕ in 2SL

  question: are there $(\mathfrak{s}, \mathfrak{h})$ and $\mathfrak{f}$ such that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{f}} \phi$?

2SL Undecidability

[Calcagno & Yang & O'Hearn, APLAS 01]

- Reduction from finitary satisfiability for classical predicate logic restricted to a single binary predicate symbol.

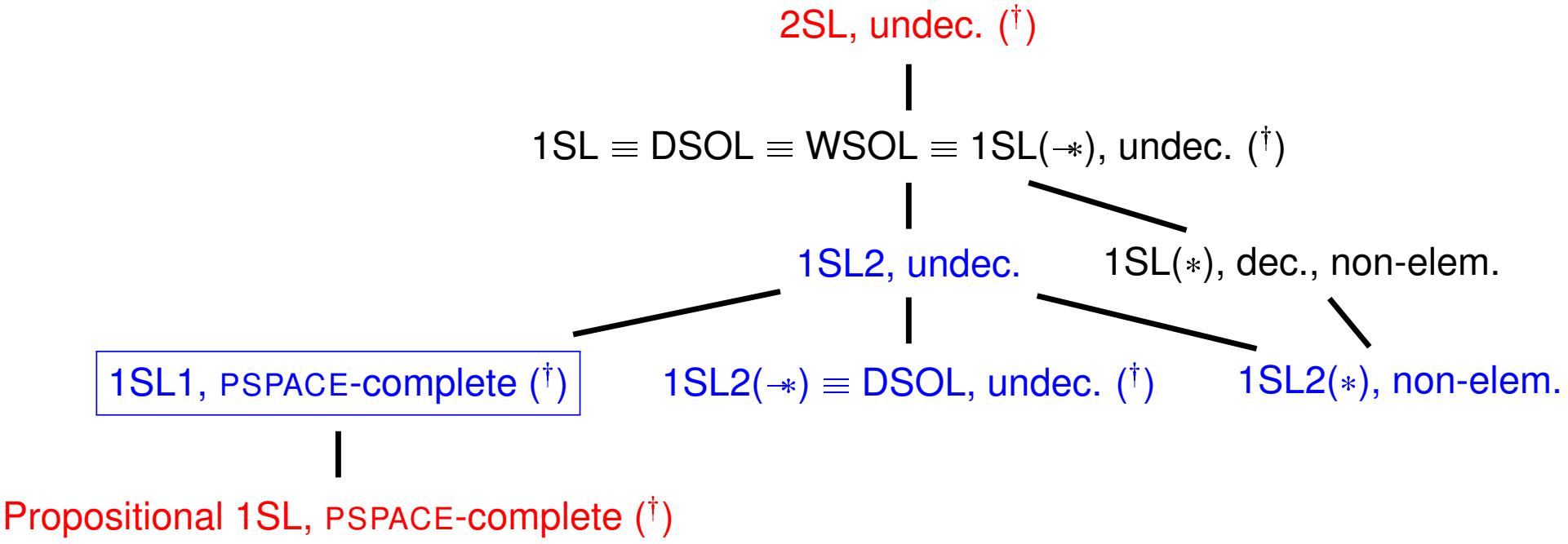
  [Trakhtenbrot, 50]

- $\overset{\iota}{\boxed{\mathtt{n}}\boxed{\mathtt{m}}} \approx R(n, m)$.
- $\mathtt{D}(\mathtt{u}) \stackrel{\text{def}}{=} \mathtt{u} \hookrightarrow \mathit{nil}, \mathit{nil}$.
- Translation

$$\exists \mathtt{u}, \mathit{nil}\ \mathtt{D}(\mathtt{u}) \wedge (\neg \exists \mathtt{u}', \mathtt{u}''\ \mathit{nil} \hookrightarrow \mathtt{u}', \mathtt{u}'') \wedge t(\phi)$$

  - t is homomorphic for Boolean connectives.
  - $t(\mathtt{R}(\mathtt{u}_i, \mathtt{u}_j)) = \mathtt{D}(\mathtt{u}_i) \wedge \mathtt{D}(\mathtt{u}_j) \wedge \exists \mathtt{u}\ (\mathtt{u} \hookrightarrow \mathtt{u}_i, \mathtt{u}_j)$.
  - $t(\exists \mathtt{u}\ \psi) \stackrel{\text{def}}{\equiv} \exists \mathtt{u}\ \mathtt{D}(\mathtt{u}) \wedge t(\psi)$.

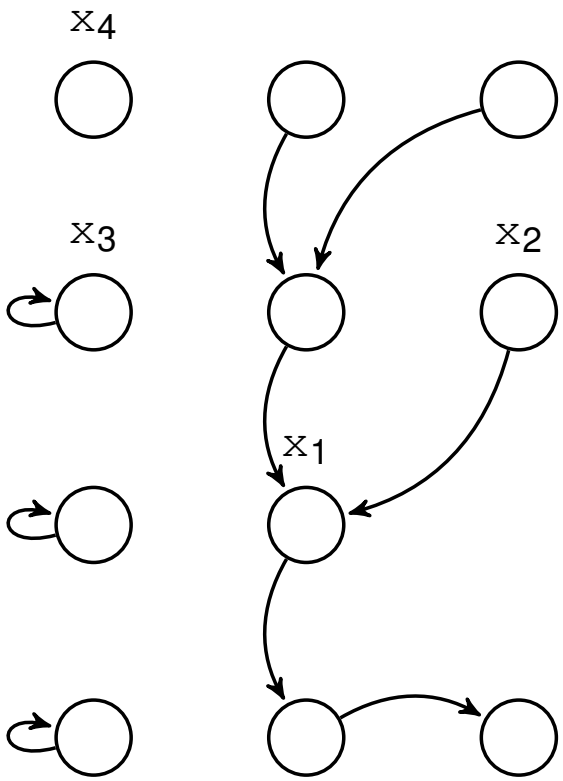
Fragments



(†) + Extension to $k > 1$ record fields.

Memory states (one field)

- Memory state $(\mathfrak{s}, \mathfrak{h})$:
  - Store $\mathfrak{s} : \mathtt{PVAR} \to \mathbb{N}$.
  - Heap $\mathfrak{h} : \mathbb{N} \rightharpoonup \mathbb{N}$ with finite domain.
    Graph of a unary function with finite domain.



Specialization for 1SL1

- Expressions: $e ::= \mathtt{x}_i \mid \mathtt{u}$
- Atomic formulae: $\pi ::= e = e' \mid e \hookrightarrow e' \mid \mathtt{emp}$
- Formulae in 1SL1

$$\phi ::= \bot \mid \pi \mid \phi \wedge \psi \mid \neg\phi \mid \phi * \psi \mid \phi \mathrel{-\!\!*} \psi \mid \exists\, \mathtt{u}\, \phi$$

- $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{l}} \mathtt{emp} \overset{\text{def}}{\Leftrightarrow} \mathrm{dom}(\mathfrak{h}) = \varnothing$.
- $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{l}} e = e' \overset{\text{def}}{\Leftrightarrow} [e] = [e']$, with $[\mathtt{x}_i] \overset{\text{def}}{=} \mathfrak{s}(\mathtt{x}_i)$ and $[\mathtt{u}] \overset{\text{def}}{=} \mathfrak{l}$.
- $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{l}} e \hookrightarrow e' \overset{\text{def}}{\Leftrightarrow} [e] \in \mathrm{dom}(\mathfrak{h})$ and $\mathfrak{h}([e]) = [e']$.
- $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{l}} \phi_1 * \phi_2 \overset{\text{def}}{\Leftrightarrow} \mathfrak{h} = \mathfrak{h}_1 \uplus \mathfrak{h}_2$, $(\mathfrak{s}, \mathfrak{h}_1) \models_{\mathfrak{l}} \phi_1$, $(\mathfrak{s}, \mathfrak{h}_2) \models_{\mathfrak{l}} \phi_2$ for some $\mathfrak{h}_1, \mathfrak{h}_2$.

Semantics for 1SL1

- $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{l}} \phi_1 \mathrel{-\!\!*} \phi_2 \overset{\text{def}}{\Leftrightarrow}$ for all $\mathfrak{h}'$, if $\mathfrak{h} \perp \mathfrak{h}'$ and $(\mathfrak{s}, \mathfrak{h}') \models_{\mathfrak{l}} \phi_1$ then $(\mathfrak{s}, \mathfrak{h} \uplus \mathfrak{h}') \models_{\mathfrak{l}} \phi_2$.

- $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{l}} \exists\, \mathtt{u}\, \phi \overset{\text{def}}{\Leftrightarrow}$ there is $\mathfrak{l}' \in \mathbb{N}$ such that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{l}'} \phi$.

- Satisfiability problem:

  input: formula ϕ in 1SL1

  question: are there $(\mathfrak{s}, \mathfrak{h})$ and $\mathfrak{l}$ such that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{l}} \phi$?

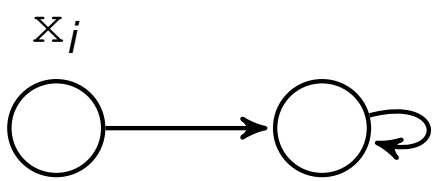
- Satisfiability problem for 1SL0 is PSPACE-complete and satisfiability problem for 1SL2 is undecidable.

Simple properties stated in 1SL1

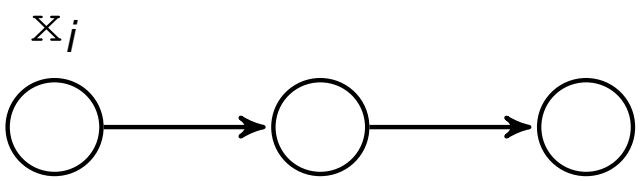
- The domain of the heap has at least k elements:
  $\neg\mathtt{emp} * \cdots * \neg\mathtt{emp}$ (k times).

- The variable $\mathtt{x}_i$ is allocated in the heap:
  $\mathtt{alloc}(\mathtt{x_i}) \stackrel{\text{def}}{=} (\mathtt{x_i} \hookrightarrow \mathtt{x_i}) \mathrel{-\!\!*} \bot$.

- The variable $\mathtt{x}_i$ points to a location that is a loop:
  $\mathtt{toloop}(\mathtt{x_i}) \stackrel{\text{def}}{=} \exists\, \mathtt{u}\, (\mathtt{x}_i \hookrightarrow \mathtt{u} \wedge \mathtt{u} \hookrightarrow \mathtt{u})$.

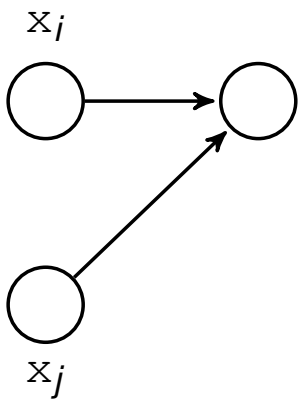


- The variable $\mathtt{x}_i$ points to a location that is allocated:
  $\mathtt{toalloc}(\mathtt{x_i}) \stackrel{\text{def}}{=} \exists\, \mathtt{u}\, (\mathtt{x_i} \hookrightarrow \mathtt{u} \wedge \mathtt{alloc}(\mathtt{u}))$.

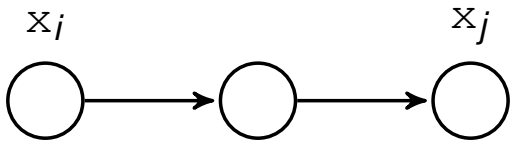


More properties

- Variables $\mathtt{x}_i$ and $\mathtt{x}_j$ point to a shared location:
  $\mathtt{conv}(\mathtt{x}_i, \mathtt{x}_j) \stackrel{\text{def}}{=} \exists\, \mathtt{u}\ (\mathtt{x}_\mathtt{i} \hookrightarrow \mathtt{u} \wedge \mathtt{x}_\mathtt{j} \hookrightarrow \mathtt{u})$.



- there is a location between $\mathtt{x}_i$ and $\mathtt{x}_j$:
  $\mathtt{inbetween}(\mathtt{x}_i, \mathtt{x}_j) \stackrel{\text{def}}{=} \exists \mathtt{u}\ (\mathtt{x}_i \hookrightarrow \mathtt{u} \wedge \mathtt{u} \hookrightarrow \mathtt{x}_j)$.



What Else?

Partition one: loops, predecessors, etc.

- $\mathrm{pred}(\mathfrak{s}, \mathfrak{h}) \stackrel{\text{def}}{=} \bigcup_i \mathrm{pred}(\mathfrak{s}, \mathfrak{h}, i)$ with
  $\mathrm{pred}(\mathfrak{s}, \mathfrak{h}, i) \stackrel{\text{def}}{=} \{\mathfrak{l}' : \mathfrak{h}(\mathfrak{l}') = \mathfrak{s}(\mathtt{x}_i)\}$ for every $i \in [1, q]$.

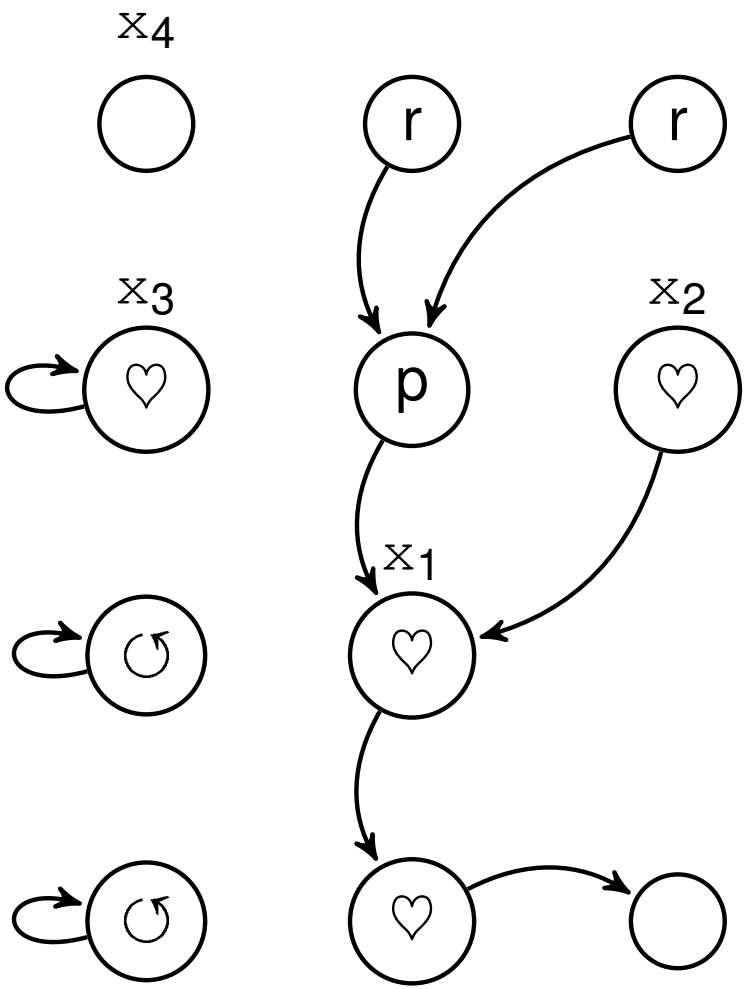
- $\mathrm{loop}(\mathfrak{s}, \mathfrak{h}) \stackrel{\text{def}}{=} \{\mathfrak{l} \in \mathrm{dom}(\mathfrak{h}) : \mathfrak{h}(\mathfrak{l}) = \mathfrak{l}\}$.

- $\mathrm{rem}(\mathfrak{s}, \mathfrak{h}) \stackrel{\text{def}}{=} \mathrm{dom}(\mathfrak{h}) \backslash (\mathrm{pred}(\mathfrak{s}, \mathfrak{h}) \cup \mathrm{loop}(\mathfrak{s}, \mathfrak{h}))$.

- $\boxed{\mathrm{dom}(\mathfrak{h}) = \mathrm{rem}(\mathfrak{s}, \mathfrak{h}) \uplus (\mathrm{pred}(\mathfrak{s}, \mathfrak{h}) \cup \mathrm{loop}(\mathfrak{s}, \mathfrak{h}))}$.

Partition two: introducing the core

- $\mathrm{ref}(\mathfrak{s},\mathfrak{h}) \stackrel{\text{def}}{=} \mathrm{dom}(\mathfrak{h}) \cap \mathfrak{s}(\mathcal{V})$; $\mathrm{acc}(\mathfrak{s},\mathfrak{h}) \stackrel{\text{def}}{=} \mathrm{dom}(\mathfrak{h}) \cap \mathfrak{h}(\mathfrak{s}(\mathcal{V}))$.
- $\heartsuit(\mathfrak{s},\mathfrak{h}) \stackrel{\text{def}}{=} \mathrm{ref}(\mathfrak{s},\mathfrak{h}) \cup \mathrm{acc}(\mathfrak{s},\mathfrak{h})$; $\overline{\heartsuit}(\mathfrak{s},\mathfrak{h}) \stackrel{\text{def}}{=} \mathrm{dom}(\mathfrak{h}) \backslash \heartsuit(\mathfrak{s},\mathfrak{h})$.



Locations outside of the core

- Locations in the core are easy to identify thanks to program variables.
- $\mathrm{pred}_{\overline{\heartsuit}}(\mathfrak{s}, \mathfrak{h}, i) \stackrel{\text{def}}{=} \mathrm{pred}(\mathfrak{s}, \mathfrak{h}, i) \backslash \heartsuit(\mathfrak{s}, \mathfrak{h})$.
- $\mathrm{loop}_{\overline{\heartsuit}}(\mathfrak{s}, \mathfrak{h}) \stackrel{\text{def}}{=} \mathrm{loop}(\mathfrak{s}, \mathfrak{h}) \backslash \heartsuit(\mathfrak{s}, \mathfrak{h})$.
- $\mathrm{rem}_{\overline{\heartsuit}}(\mathfrak{s}, \mathfrak{h}) \stackrel{\text{def}}{=} \mathrm{rem}(\mathfrak{s}, \mathfrak{h}) \backslash \heartsuit(\mathfrak{s}, \mathfrak{h})$.
- $\boxed{\mathrm{dom}(\mathfrak{h}) = \heartsuit(\mathfrak{s}, \mathfrak{h}) \uplus \mathrm{pred}_{\overline{\heartsuit}}(\mathfrak{s}, \mathfrak{h}) \uplus \mathrm{loop}_{\overline{\heartsuit}}(\mathfrak{s}, \mathfrak{h}) \uplus \mathrm{rem}_{\overline{\heartsuit}}(\mathfrak{s}, \mathfrak{h})}$.

Test formulae

- $\mathtt{Equality} \stackrel{\text{def}}{=} \{\mathtt{x}_i = \mathtt{x}_j \mid i, j \in [1, q]\}$.

- $\mathtt{Pattern} \stackrel{\text{def}}{=}$
  $\{\mathtt{x}_i \hookrightarrow \mathtt{x}_j, \mathtt{conv}(\mathtt{x_i}, \mathtt{x_j}), \mathtt{inbetween}(\mathtt{x}_i, \mathtt{x}_j) \mid i, j \in [1, q]\}$
  $\cup \{\mathtt{toalloc}(\mathtt{x}_i), \mathtt{toloop}(\mathtt{x}_i), \mathtt{alloc}(\mathtt{x_i}) \mid i \in [1, q]\}$.

- $\mathtt{Extra}^{\mathtt{u}} \stackrel{\text{def}}{=}$
  $\{\mathtt{u} \hookrightarrow \mathtt{u}, \mathtt{alloc}(\mathtt{u})\} \cup \{\mathtt{x}_i = \mathtt{u}, \mathtt{x}_i \hookrightarrow \mathtt{u}, \mathtt{u} \hookrightarrow \mathtt{x}_i \mid i \in [1, q]\}$.

- $\mathtt{Size}_\alpha \stackrel{\text{def}}{=}$
  $\{\#\mathtt{pred}^{\mathtt{i}}_{\overline{\heartsuit}} \geqslant \mathtt{k} \mid i \in [1, q], k \in [1, \alpha]\}$
  $\cup \{\#\mathtt{loop}_{\overline{\heartsuit}} \geqslant \mathtt{k}, \#\mathtt{rem}_{\overline{\heartsuit}} \geqslant \mathtt{k} \mid k \in [1, \alpha]\}$.

- $\mathtt{Test}^{\mathtt{u}}_\alpha \stackrel{\text{def}}{=} \mathtt{Equality} \cup \mathtt{Pattern} \cup \mathtt{Size}_\alpha \cup \mathtt{Extra}^{\mathtt{u}} \cup \{\bot\}$.

Counting loops outside of the core

- Need for expressing test formulae in 1SL1 !

- $\mathtt{T} \stackrel{\text{def}}{=} \{\mathtt{alloc}(\mathtt{x}_1), \ldots, \mathtt{alloc}(\mathtt{x}_q)\} \cup \{\mathtt{toalloc}(\mathtt{x}_1), \ldots, \mathtt{toalloc}(\mathtt{x}_q)\}$.

- $\mathfrak{f} : \mathtt{T} \to \{0, 1\}$.

$$\phi_{\mathfrak{f}} \stackrel{\text{def}}{=} \bigwedge\{\psi \mid \psi \in \mathtt{T} \text{ and } \mathfrak{f}(\psi) = 1\} \wedge \bigwedge\{\neg\psi \mid \psi \in \mathtt{T} \text{ and } \mathfrak{f}(\psi) = 0\}$$

- $\#\,\mathtt{loop}_{\overline{\heartsuit}} \geqslant \mathtt{k} \stackrel{\text{def}}{=} \bigvee_{\mathfrak{f}} \phi_{\mathfrak{f}} \wedge \left(\phi_{\mathfrak{f}}^{\text{pos}} * (\#\,\mathtt{loop} \geqslant \mathtt{k})\right)$ with
  - $\phi_{\mathfrak{f}}^{\text{pos}}$ = the positive part of $\phi_{\mathfrak{f}}$.
  - $\#\,\mathtt{loop} \geqslant \mathtt{k} \stackrel{\text{def}}{=} (\exists \mathtt{u}\ \mathtt{u} \hookrightarrow \mathtt{u}) * \cdots * (\exists \mathtt{u}\ \mathtt{u} \hookrightarrow \mathtt{u})$ (*k* times).

Deciding satisfiability for test formulae

- Satisfiability of conjunctions of test formulae or their negation can be checked in polynomial time.
  (with binary encoding of bounds)
- Satisfiability problem for Boolean combinations of test formulae in the set $\bigcup_{\alpha \geqslant 1} \mathtt{Test}^{\mathtt{u}}_{\alpha}$ is NP-complete.
- Polynomial-time decision procedure based on a saturation algorithm. (see examples of rules)

$$\frac{\phi \vdash \mathtt{x}_i \hookrightarrow \mathtt{x} \quad \phi \vdash \mathtt{x} \hookrightarrow \mathtt{y} \quad \phi \vdash \mathtt{x} = \mathtt{y}}{\phi \vdash \mathtt{toloop}(\mathtt{x_i})}$$

$$\frac{\phi \vdash \mathtt{conv}(\mathtt{x}_i, \mathtt{x}_j) \quad \phi \vdash \mathtt{toloop}(\mathtt{x_i})}{\phi \vdash \mathtt{toloop}(\mathtt{x_j})}$$

$$\frac{\phi \vdash \neg\mathtt{alloc}(\mathtt{x}_i)}{\phi \vdash \neg\mathtt{toloop}(\mathtt{x_i})}$$

Most complex rules

$$\frac{\phi \vdash \mathtt{u} \hookrightarrow \mathtt{u} \quad (\phi \vdash \neg(\mathtt{u} = \mathtt{x}_j))_{j \in [1,q]} \quad (\phi \vdash \neg(\mathtt{x}_j \hookrightarrow \mathtt{u}))_{j \in [1,q]}}{\phi \vdash \#\,\mathtt{loop}_{\overline{\heartsuit}} \geqslant 1}$$

$$\frac{\phi \vdash \mathtt{u} \hookrightarrow \mathtt{x}_i \quad (\phi \vdash \neg(\mathtt{u} = \mathtt{x}_j))_{j \in [1,q]} \quad (\phi \vdash \neg(\mathtt{x}_j \hookrightarrow \mathtt{u}))_{j \in [1,q]}}{\phi \vdash \#\,\mathtt{pred}^{\mathtt{i}}_{\overline{\heartsuit}} \geqslant 1}$$

$$\frac{\phi \vdash \mathtt{alloc}(\mathtt{u})\ \phi \vdash \neg(\mathtt{u} \hookrightarrow \mathtt{u})\ (\phi \vdash \neg(\mathtt{u} \hookrightarrow \mathtt{x}_j))_j\ (\phi \vdash \neg(\mathtt{u} = \mathtt{x}_j))_j\ (\phi \vdash \neg(\mathtt{x}_j \hookrightarrow \mathtt{u}))_j}{\phi \vdash \#\,\mathtt{rem}_{\overline{\heartsuit}} \geqslant 1}$$

Expressive Completeness

Memory threshold

- $\mathtt{th}(q, \phi) \stackrel{\text{def}}{=} 1$ for every atomic formula ϕ.
- $\mathtt{th}(q, \phi_1 \wedge \phi_2) \stackrel{\text{def}}{=} \mathtt{max}(\mathtt{th}(q, \phi_1), \mathtt{th}(q, \phi_2))$.
- $\mathtt{th}(q, \neg\phi_1) \stackrel{\text{def}}{=} \mathtt{th}(q, \phi_1)$.
- $\mathtt{th}(q, \exists\, \mathtt{u}\ \phi_1) \stackrel{\text{def}}{=} \mathtt{th}(q, \phi_1)$.
- $\mathtt{th}(q, \phi_1 * \phi_2) \stackrel{\text{def}}{=} \mathtt{th}(q, \phi_1) + \mathtt{th}(q, \phi_2)$.
- $\mathtt{th}(q, \phi_1 \mathrel{-\!\!*} \phi_2) \stackrel{\text{def}}{=} q + \mathtt{max}(\mathtt{th}(q, \phi_1), \mathtt{th}(q, \phi_2))$.
- $\boxed{\text{For all } \phi \text{ built over } \{\mathtt{x}_1, \ldots, \mathtt{x}_q\},\ 1 \leqslant \mathtt{th}(q, \phi) \leqslant q \times |\phi|.}$

α-equivalence

- α-equivalence: indistinguishability with respect to test formulae from $\mathtt{Test}^{\mathtt{u}}_{\alpha}$.

$$(\mathfrak{s}, \mathfrak{h}, \mathfrak{l}) \simeq_{\alpha} (\mathfrak{s}', \mathfrak{h}', \mathfrak{l}')$$
$$\overset{\text{def}}{\Leftrightarrow}$$

for all $\psi \in \mathtt{Test}^{\mathtt{u}}_{\alpha}$, we have $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{l}} \psi$ iff $(\mathfrak{s}', \mathfrak{h}') \models_{\mathfrak{l}'} \psi$.

- Cardinality constraints are precise up to α.

$$\boxed{\mathtt{th}(q, \phi) \leqslant \alpha \ \& \ (\mathfrak{s}, \mathfrak{h}, \mathfrak{l}) \simeq_{\alpha} (\mathfrak{s}', \mathfrak{h}', \mathfrak{l}') \rightarrow (\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{l}} \phi \text{ iff } (\mathfrak{s}', \mathfrak{h}') \models_{\mathfrak{l}'} \phi.}$$

Quantifier elimination

- Any ϕ in 1SL1 (with q program variables) is equivalent to a Boolean combination ϕ' of test formulae in $\mathtt{Test}^{\mathtt{u}}_{\mathtt{th}(q,\phi)}$.
- $\alpha = \mathtt{th}(q, \phi)$.
- $at(\mathfrak{s}, \mathfrak{h}, \mathfrak{l}) \stackrel{\text{def}}{=} \{\psi \mid \psi \in \mathtt{Test}^{\mathtt{u}}_{\alpha}$ and $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{l}} \psi\} \cup \{\neg\psi \mid \psi \in \mathtt{Test}^{\mathtt{u}}_{\alpha}$ and $(\mathfrak{s}, \mathfrak{h}) \not\models_{\mathfrak{l}} \psi\}$.
- Finiteness of $\mathtt{Test}^{\mathtt{u}}_{\alpha}$ entails $at(\mathfrak{s}, \mathfrak{h}, \mathfrak{l})$ is finite and $\bigwedge at(\mathfrak{s}, \mathfrak{h}, \mathfrak{l})$ is a well-defined atom. (characteristic formula)
- $(\mathfrak{s}', \mathfrak{h}') \models_{\mathfrak{l}'} \bigwedge at(\mathfrak{s}, \mathfrak{h}, \mathfrak{l})$ iff $(\mathfrak{s}, \mathfrak{h}, \mathfrak{l}) \simeq_{\alpha} (\mathfrak{s}', \mathfrak{h}', \mathfrak{l}')$.
- $\phi' \stackrel{\text{def}}{=} \bigvee\{\bigwedge at(\mathfrak{s}, \mathfrak{h}, \mathfrak{l}) \mid (\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{l}} \phi\}$.

non-constructive proof !

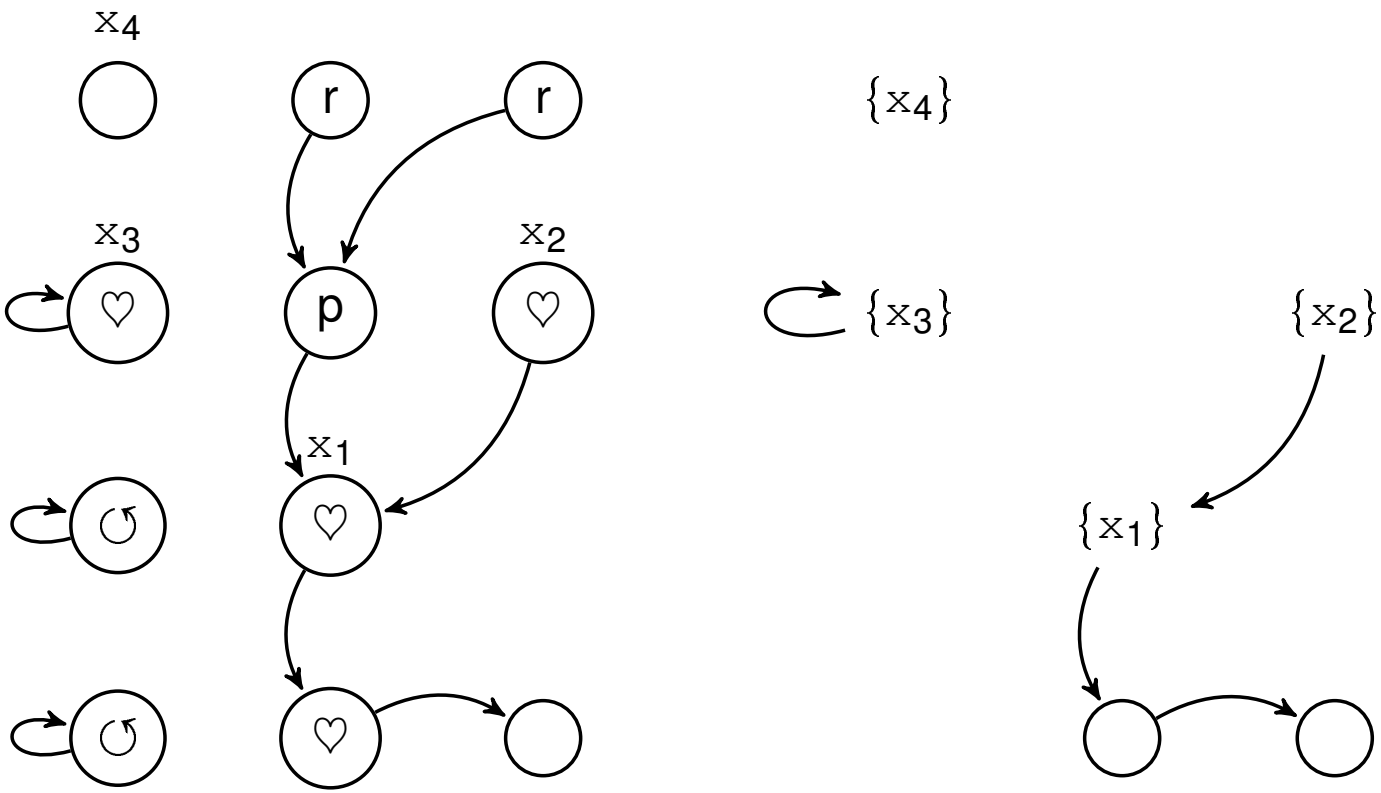
Corollaries

- Any satisfiable ϕ in 1SL1 has a polynomial-size model.
- 1SL2 is strictly more expressive than 1SL1.
- 1SL1 cannot distinguish the two models below, 1SL2 can:

$$\mathrm{x}_1 \rightarrow \bullet \rightarrow \bullet \rightarrow \mathrm{x}_2 \quad \Big| \quad \mathrm{x}_1 \rightarrow \bullet \rightarrow \bullet \qquad \circ \rightarrow \mathrm{x}_2$$

Computational Complexity

Abstract memory states ≈ atoms



$$\mathfrak{l} = 2,\ \mathfrak{r} = 2,\ \mathfrak{p}_1 = 1,\ \mathfrak{p}_2 = \mathfrak{p}_3 = \mathfrak{p}_4 = 0.$$

Abstract memory state: $\mathfrak{a} = ((V, E), \mathfrak{l}, \mathfrak{r}, \mathfrak{p}_1, \ldots, \mathfrak{p}_q)$.
$V_{\mathtt{par}} \subseteq V$ partition of $\{\mathtt{x}_1, \ldots, \mathtt{x}_q\}$.

Abstraction

- $\mathrm{abs}(\mathfrak{s}, \mathfrak{h})$: abstract memory states abstracting the memory state $(\mathfrak{s}, \mathfrak{h})$.
- Isomorphic $\mathfrak{a}$ and $\mathfrak{a}'$: identical partition, isomorphic graphs, identical numerical values.
- Given $q, \alpha \geqslant 1$, the number of abstract memory states is exponential in $q + \alpha$ and each abstract memory state can be encoded in space $q + log(\alpha)$.

Pointed abstract memory states

- Need for abstracting the interpretation of variable $\mathtt{u}$.
- Pointed abstract memory states $(\mathfrak{a}, \mathfrak{u})$ with $\mathfrak{u}$ in the set

$$V^{d \leqslant 1} \cup \{\overline{D}, P(1), \ldots, P(q), L, R\}$$

  (conditions apply)

- $\mathrm{abs}(\mathfrak{s}, \mathfrak{h}, \mathfrak{l})$ defined accordingly.
- $(\mathfrak{a}, \mathfrak{u})$ and $(\mathfrak{a}', \mathfrak{u}')$ are isomorphic $\overset{\text{def}}{\Leftrightarrow}$ $\mathfrak{a}$ and $\mathfrak{a}'$ are isomorphic and, $\mathfrak{u} = \mathfrak{u}'$ or $\mathfrak{u}$ and $\mathfrak{u}'$ are related by the isomorphism.

Correctness of the abstraction

- Given $(\mathfrak{a}, \mathfrak{u})$ over (q, α), there exist $(\mathfrak{s}, \mathfrak{h})$ and $\mathfrak{l} \in \mathbb{N}$ such that $\mathrm{abs}(\mathfrak{s}, \mathfrak{h}, \mathfrak{l})$ and $(\mathfrak{a}, \mathfrak{u})$ are isomorphic.

- Let $(\mathfrak{s}, \mathfrak{h})$, $(\mathfrak{s}', \mathfrak{h}')$ be memory states and $\mathfrak{l}, \mathfrak{l}' \in \mathbb{N}$.

  (I) $(\mathfrak{s}, \mathfrak{h}, \mathfrak{l}) \simeq_\alpha (\mathfrak{s}', \mathfrak{h}', \mathfrak{l}')$.

  $$\Leftrightarrow$$

  (II) $\mathrm{abs}(\mathfrak{s}, \mathfrak{h}, \mathfrak{l})$ and $\mathrm{abs}(\mathfrak{s}', \mathfrak{h}', \mathfrak{l}')$ are isomorphic.

  $$\Leftrightarrow$$

  (III) There is a unique atom ψ from $\mathtt{Test}_\alpha^{\mathtt{u}}$ such that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{l}} \psi$ and $(\mathfrak{s}', \mathfrak{h}') \models_{\mathfrak{l}'} \psi$.

Abstract separation

- $*_a((\mathfrak{a}, \mathfrak{u}), (\mathfrak{a}_1, \mathfrak{u}_1), (\mathfrak{a}_2, \mathfrak{u}_2)) \overset{\text{def}}{\Leftrightarrow}$

  there exist $\mathfrak{l} \in \mathbb{N}$, a store $\mathfrak{s}$ and disjoint heaps $\mathfrak{h}_1$ and $\mathfrak{h}_2$ such that $\mathrm{abs}(\mathfrak{s}, \mathfrak{h}_1 \uplus \mathfrak{h}_2, \mathfrak{l}) = (\mathfrak{a}, \mathfrak{u})$, $\mathrm{abs}(\mathfrak{s}, \mathfrak{h}_1, \mathfrak{l}) = (\mathfrak{a}_1, \mathfrak{u}_1)$ and $\mathrm{abs}(\mathfrak{s}, \mathfrak{h}_2, \mathfrak{l}) = (\mathfrak{a}_2, \mathfrak{u}_2)$.

- Given $q, \alpha \geqslant 1$, the ternary relation $*_a$ can be decided in polynomial time in $q + \mathit{log}(\alpha)$ for all the pointed abstract memory states over (q, α).

Model-checking algorithm

1: **if** ψ is atomic **then** return AMC$((\mathfrak{a}, \mathfrak{u}), \psi)$;
2: **if** $\psi = \neg\psi_1$ **then** return not MC$((\mathfrak{a}, \mathfrak{u}), \psi_1)$;
3: **if** $\psi = \psi_1 \wedge \psi_2$ **then** return (MC$((\mathfrak{a}, \mathfrak{u}), \psi_1)$ and MC$((\mathfrak{a}, \mathfrak{u}), \psi_2)$);

4: **if** $\psi = \exists\ \mathfrak{u}\ \psi_1$ **then** return $\top$ iff there is $\mathfrak{u}'$ such that MC$((\mathfrak{a}, \mathfrak{u}'), \psi_1) = \top$;

5: **if** $\psi = \psi_1 * \psi_2$ **then** return $\top$ iff there are $(\mathfrak{a}_1, \mathfrak{u}_1)$ and $(\mathfrak{a}_2, \mathfrak{u}_2)$ such that $*_a((\mathfrak{a}, \mathfrak{u}), (\mathfrak{a}_1, \mathfrak{u}_1), (\mathfrak{a}_2, \mathfrak{u}_2))$ and MC$((\mathfrak{a}_1, \mathfrak{u}_1), \psi_1) =$ MC$((\mathfrak{a}_2, \mathfrak{u}_2), \psi_2) = \top$;

6: **if** $\psi = \psi_1 \mathrel{-\!\!*} \psi_2$ **then** return $\bot$ iff for some $(\mathfrak{a}', \mathfrak{u}')$ and $(\mathfrak{a}'', \mathfrak{u}'')$ such that $*_a((\mathfrak{a}'', \mathfrak{u}''), (\mathfrak{a}', \mathfrak{u}'), (\mathfrak{a}, \mathfrak{u}))$, MC$((\mathfrak{a}', \mathfrak{u}'), \psi_1) = \top$ and MC$((\mathfrak{a}'', \mathfrak{u}''), \psi_2) = \bot$;

PSPACE-completeness

- $q, \alpha \geqslant 1$, $(\mathfrak{a}, \mathfrak{u})$ pointed abstract memory state over (q, α), ϕ built over $\mathtt{x}_1, \ldots, \mathtt{x}_q$ with $\mathtt{th}(q, \phi) \leqslant \alpha$.

- (I), (II) and (III) are equivalent:

  (I) MC$((\mathfrak{a}, \mathfrak{u}), \phi)$ returns $\top$.

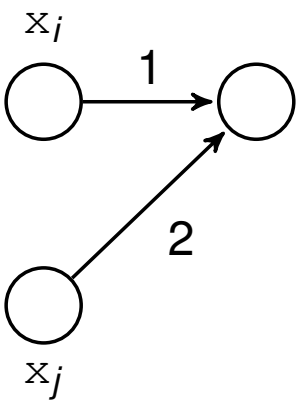
  (II) There exist $(\mathfrak{s}, \mathfrak{h})$ and $\mathfrak{l} \in \mathbb{N}$ such that $\mathrm{abs}(\mathfrak{s}, \mathfrak{h}, \mathfrak{l}) = (\mathfrak{a}, \mathfrak{u})$ and $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{l}} \phi$.

  (III) For all $(\mathfrak{s}, \mathfrak{h})$ and $\mathfrak{l} \in \mathbb{N}$ such that $\mathrm{abs}(\mathfrak{s}, \mathfrak{h}, \mathfrak{l}) = (\mathfrak{a}, \mathfrak{u})$, we have $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{l}} \phi$.

- Model-checking and satisfiability problems for 1SL1 are PSPACE-complete.
  (PSPACE-hardness is inherited from 1SL0)

By-products

- Let $q \geqslant 1$. The satisfiability problem for 1SL1 restricted to formulae with at most q program variables can be solved in polynomial time.
- Given ϕ in 1SL1, computing a Boolean combination of test formulae equivalent to ϕ can be done in polynomial space.
- PSPACE upper bound still holds when formulae are encoded as DAGs. (algorithm runs in space $log(\alpha)$).

Concluding remarks

- Quantifier elimination property for 1SL1 formulae.
- Conjunction of test formulae decidable in polynomial time.
- Satisfiability and model-checking problems for 1SL1 are PSPACE-complete.
- Restriction to q program variables in polynomial time too.
- Possible extension with $k > 1$ record fields.