On the Almighty Wand

Stéphane Demri

LSV, ENS Cachan, CNRS, INRIA

Joint work with Rémi Brochenin and Etienne Lozes

Pointer programs

- Pointer: reference to a memory cell
  (non fixed memory address).

- Dynamic memory allocation/deallocation.
  (mutable data structures)

- Examples of instructions:
  - $\mathtt{y} \rightarrow l := \mathtt{x}$: write $\mathtt{x}$ to the l-field of the cell pointed to by $\mathtt{y}$,
  - `free` $\mathtt{x}$: deallocate the cell pointer to by x,
  - $\mathtt{x} := \mathtt{malloc}(i)$: allocate i memory cells and assign its address to $\mathtt{x}$.

- Specific properties for pointer programs:
  - No `null` dereference.
  - Memory leak: a memory cell can no longer be reached.
  - Shape analysis: checking the structure of the heap.

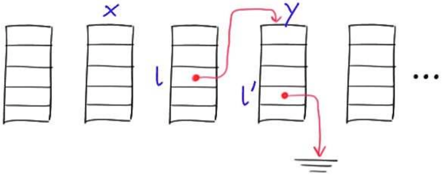
Reasoning about pointer programs

- Examples of logical specification languages
  - Separation logic [Reynolds, LICS 02]
  - Pointer assertion logic (PAL) [Jensen et al. 97]
  - TVLA [Lev-Ami & Sagiv, SAS 00]: abstract interpretation technique with Kleene's logic (op. semantics in FOL + TC)
  - Evolution Logic [Yahav et al., ESOP 03]: to specify temporal properties of programs with dynamically evolving heaps.

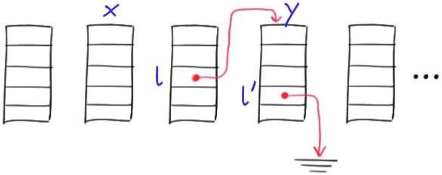
Reasoning about pointer programs

- Examples of logical specification languages
  - Separation logic [Reynolds, LICS 02]
  - Pointer assertion logic (PAL) [Jensen et al. 97]
  - TVLA [Lev-Ami & Sagiv, SAS 00]: abstract interpretation technique with Kleene's logic (op. semantics in FOL + TC)
  - Evolution Logic [Yahav et al., ESOP 03]: to specify temporal properties of programs with dynamically evolving heaps.
- Model-checking
  - Navigation Temporal Logic [Distefano & Katoen & Rensink, FSTTCS 04]
  - Regular model-checking [Bouajjani et al., TACAS 05]
  - Translation into counter automata [Bouajjani et al, CAV 06; Sangnier, PhD 08]

Memory states (I)



Memory states (I)



- Set of variables $\mathtt{Var}$.
- Set of selectors/labels $\mathtt{Lab}$.
- Set of values $\mathtt{Val} = \mathbb{N} \uplus \{nil\}$.

- Set of stores: $\mathcal{S} \stackrel{\text{def}}{=} \mathtt{Var} \to \mathtt{Val}$.
- Set of heaps:
  $\mathcal{H} \stackrel{\text{def}}{=} \mathbb{N} \rightharpoonup_{fin} (\mathtt{Lab} \rightharpoonup_{fin+} \mathtt{Val})$.

Memory state (s, h)

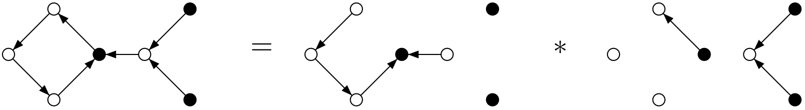
In the sequel, we restrict ourselves to two selectors only or to one selector only.

Disjoint heaps

- h_1 and h_2 are disjoint whenever $\mathtt{dom}(h_1) \cap \mathtt{dom}(h_2) = \emptyset$. Notation: $h_1 \perp h_2$.
- Disjointness does not concern records.
- Disjoint union $h_1 * h_2$ whenever $h_1 \perp h_2$.

Disjoint heaps

- h_1 and h_2 are disjoint whenever $\mathtt{dom}(h_1) \cap \mathtt{dom}(h_2) = \emptyset$.
  Notation: $h_1 \perp h_2$.
- Disjointness does not concern records.
- Disjoint union $h_1 * h_2$ whenever $h_1 \perp h_2$.
- Disjoint heap graphs (with a unique selector and $\mathtt{Val} = \mathbb{N}$):



Separation logic

- Introduced by Reynolds, Pym and O'Hearn.
- Reasoning about the heap with a strong form of locality built-in.
- $\mathcal{A} * \mathcal{B}$ is true whenever the heap can be divided into two disjoint parts, one satisfies $\mathcal{A}$, the other one $\mathcal{B}$.
  (second-order existential modality)
- $\mathcal{A} \mathrel{-\!\!*} \mathcal{B}$ is true whenever $\mathcal{A}$ is true for a (fresh) disjoint heap, $\mathcal{B}$ is true for the combined heap.
  (second-order universal modality)

Hoare triples

- Hoare triple: $\{\mathcal{A}\}\ \mathtt{PROG}\ \{\mathcal{B}\}$ (total correctness).

- Rule of constancy:

$$\frac{\{\mathcal{A}\}\ \mathtt{PROG}\ \{\mathcal{B}\}}{\{\mathcal{A} \wedge \mathcal{B}'\}\ \mathtt{PROG}\ \{\mathcal{B} \wedge \mathcal{B}'\}}$$

  where no variable free in $\mathcal{B}'$ is modified by $\mathtt{PROG}$.

- Unsoundness of the rule of constancy in separation logic:

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

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

When separation logic enters into the play

- Reparation with frame rule:

$$\frac{\{\mathcal{A}\}\ \texttt{PROG}\ \{\mathcal{B}\}}{\{\mathcal{A} * \mathcal{B}'\}\ \texttt{PROG}\ \{\mathcal{B} * \mathcal{B}'\}}$$

  where no variable free in $\mathcal{B}'$ is modified by PROG.

- Strengthening precedent (SP)

$$\frac{\mathcal{A} \Rightarrow \mathcal{B}' \quad \{\mathcal{B}'\}\ \texttt{PROG}\ \{\mathcal{B}\}}{\{\mathcal{A}\}\ \texttt{PROG}\ \{\mathcal{B}\}}$$

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

Standard inference rules for mutation

- Local form (MUL)

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

- Global form (MUG)

$$\overline{\{(\exists \mathtt{z}.\ \mathtt{x} \mapsto \mathtt{z}) * \mathcal{A}\}\ [\mathtt{x}] := \mathtt{y}\ \{\mathtt{x} \mapsto \mathtt{y} * \mathcal{A}\}}$$

- Backward-reasoning form (MUBR)

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

Memory states (II)

- Set of variables $\mathtt{Var} = \{\mathtt{x}, \mathtt{y}, \mathtt{z}, \ldots\}$.

- Set of locations $\mathtt{Loc} = \{l, l', \ldots\}$.

- Set of values $\mathtt{Val} = \mathbb{N} \uplus \mathtt{Loc} \uplus \{nil\}$.

Memory states (II)

- Set of variables $\mathtt{Var} = \{\mathtt{x}, \mathtt{y}, \mathtt{z}, \ldots\}$.

- Set of locations $\mathtt{Loc} = \{l, l', \ldots\}$.

- Set of values $\mathtt{Val} = \mathbb{N} \uplus \mathtt{Loc} \uplus \{nil\}$.

- Memory state:
  - Store $s : \mathtt{Var} \rightarrow \mathtt{Val}$.
  - Heap $h : \mathtt{Loc} \rightharpoonup \mathtt{Val} \times \mathtt{Val}$ with finite domain.

- Simplification: $\mathtt{Loc} = \mathtt{Val} = \mathbb{N}$.

Memory states (II)

- Set of variables $\mathtt{Var} = \{\mathtt{x}, \mathtt{y}, \mathtt{z}, \ldots\}$.
- Set of locations $\mathtt{Loc} = \{l, l', \ldots\}$.
- Set of values $\mathtt{Val} = \mathbb{N} \uplus \mathtt{Loc} \uplus \{nil\}$.
- Memory state:
  - Store $s : \mathtt{Var} \rightarrow \mathtt{Val}$.
  - Heap $h : \mathtt{Loc} \rightharpoonup \mathtt{Val} \times \mathtt{Val}$ with finite domain.
- Simplification: $\mathtt{Loc} = \mathtt{Val} = \mathbb{N}$.
- Disjoint heaps: $\mathtt{dom}(h_1) \cap \mathtt{dom}(h_2) = \emptyset$ (noted $h_1 \perp h_2$).
- When $h_1 \perp h_2$, $h_1 * h_2 \stackrel{\text{def}}{=} h_1 \uplus h_2$.

Separation logic with two record fields

- Formulae:

$$\mathcal{A} := \neg\mathcal{A} \mid \mathcal{A} \wedge \mathcal{A} \mid \exists \mathtt{x}\ \mathcal{A} \mid \overbrace{\mathtt{x} \hookrightarrow \mathtt{y}, \mathtt{z} \mid \mathtt{x} = \mathtt{y}}^{\text{atomic formulae}} \mid \mathcal{A} * \mathcal{A} \mid \mathcal{A} \mathrel{-\!\!*} \mathcal{A}$$

- Satisfaction relation:

$$\begin{array}{rcl}
(s,h) \models \neg\mathcal{A} & \text{iff} & \text{not } (s,h) \models \mathcal{A} \\
(s,h) \models \mathcal{A} \wedge \mathcal{B} & \text{iff} & (s,h) \models \mathcal{A} \text{ and } (s,h) \models \mathcal{B} \\
(s,h) \models \exists \mathtt{x}\ \mathcal{A} & \text{iff} & \text{there is } l \in \mathtt{Loc} \text{ s. t. } (s[\mathtt{x} \mapsto l], h) \models \mathcal{A} \\
(s,h) \models \mathtt{x} \hookrightarrow \mathtt{y}, \mathtt{z} & \text{iff} & h(s(\mathtt{x})) = (s(\mathtt{y}), s(\mathtt{z})) \\
(s,h) \models \mathtt{x} = \mathtt{y} & \text{iff} & s(\mathtt{x}) = s(\mathtt{y}) \\
(s,h) \models \mathcal{A}_1 * \mathcal{A}_2 & \text{iff} & \text{there are two heaps } h_1, h_2 \text{ such that} \\
& & h = h_1 * h_2, (s,h_1) \models \mathcal{A}_1 \ \&\ (s,h_2) \models \mathcal{A}_2, \\
(s,h) \models \mathcal{A}_1 \mathrel{-\!\!*} \mathcal{A}_2 & \text{iff} & \text{for all heaps } h' \perp h, \\
& & \text{if } (s,h') \models \mathcal{A}_1 \text{ then } (s, h' * h) \models \mathcal{A}_2.
\end{array}$$

Relationship between $*$ and $-\!\!*$

- $-\!\!*$ is the *adjunct* of $*$:

  $(\mathcal{A} * \mathcal{B}) \Rightarrow \mathcal{C}$ is valid iff $\mathcal{A} \Rightarrow (\mathcal{B} -\!\!* \mathcal{C})$ is valid.

Relationship between $*$ and $-\!\!*$

- $-\!\!*$ is the *adjunct* of $*$:

  $(\mathcal{A} * \mathcal{B}) \Rightarrow \mathcal{C}$ is valid iff $\mathcal{A} \Rightarrow (\mathcal{B} -\!\!* \mathcal{C})$ is valid.

- ...but the formula below is not valid

$$((\mathcal{A} * \mathcal{B}) \Rightarrow \mathcal{C}) \;\Leftrightarrow\; (\mathcal{A} \Rightarrow (\mathcal{B} -\!\!* \mathcal{C}))$$

Relationship between $*$ and $-\!\!*$

- $-\!\!*$ is the *adjunct* of $*$:

  $(\mathcal{A} * \mathcal{B}) \Rightarrow \mathcal{C}$ is valid iff $\mathcal{A} \Rightarrow (\mathcal{B} -\!\!* \mathcal{C})$ is valid.

- ... but the formula below is not valid

$$((\mathcal{A} * \mathcal{B}) \Rightarrow \mathcal{C}) \;\Leftrightarrow\; (\mathcal{A} \Rightarrow (\mathcal{B} -\!\!* \mathcal{C}))$$

- *Septraction* $\overset{\neg}{-\!\!*}$: existential version of $-\!\!*$.

$$\mathcal{A} \overset{\neg}{-\!\!*} \mathcal{B} \stackrel{\text{def}}{=} \neg(\mathcal{A} -\!\!* \neg\mathcal{B})$$

  $(s, h) \models \mathcal{A} \overset{\neg}{-\!\!*} \mathcal{B}$ iff there is $h' \perp h$ such that $(s, h') \models \mathcal{A}$ and $(s, h' * h) \models \mathcal{B}$.

Undecidability

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

- Reduction from finitary satisfiability for classical predicate logic restricted to a single binary predicate symbol, see e.g. [Trakhtenbrot, 50].

- $D(\mathtt{x}) \stackrel{\text{def}}{=} \mathtt{x} \hookrightarrow \textbf{nil}, \textbf{nil}.$

- Translation

$$\exists \mathtt{x}, \textbf{nil}\ D(\mathtt{x}) \wedge (\neg \exists \mathtt{y}, \mathtt{z}\ \textbf{nil} \hookrightarrow \mathtt{y}, \mathtt{z}) \wedge t(\mathcal{A})$$

  - t is homomorphic for Boolean connectives.

  - $t(\mathtt{R}(\mathtt{x}, \mathtt{y})) = D(\mathtt{x}) \wedge D(\mathtt{y}) \wedge \exists \mathtt{z}\ \mathtt{z} \hookrightarrow \mathtt{x}, \mathtt{y}.$

  - $t(\exists \mathtt{x}\ \mathcal{B}) \stackrel{\text{def}}{=} \exists \mathtt{x}\ D(\mathtt{x}) \wedge t(\mathcal{B}).$

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

- Reduction from finitary satisfiability for classical predicate logic restricted to a single binary predicate symbol, see e.g. [Trakhtenbrot, 50].
- $D(\mathtt{x}) \stackrel{\text{def}}{=} \mathtt{x} \hookrightarrow \textbf{nil}, \textbf{nil}$.
- Translation

$$\exists \mathtt{x}, \textbf{nil}\ D(\mathtt{x}) \wedge (\neg \exists \mathtt{y}, \mathtt{z}\ \textbf{nil} \hookrightarrow \mathtt{y}, \mathtt{z}) \wedge t(\mathcal{A})$$

  - t is homomorphic for Boolean connectives.
  - $t(\mathtt{R}(\mathtt{x}, \mathtt{y})) = D(\mathtt{x}) \wedge D(\mathtt{y}) \wedge \exists \mathtt{z}\ \mathtt{z} \hookrightarrow \mathtt{x}, \mathtt{y}$.
  - $t(\exists \mathtt{x}\ \mathcal{B}) \stackrel{\text{def}}{=} \exists \mathtt{x}\ D(\mathtt{x}) \wedge t(\mathcal{B})$.

What is the decidability status with a unique selector?

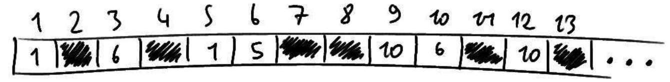
Complexity of propositional fragments [Calcagno & Yang & O'Hearn, APLAS 01]

- Model-checking and satisfiability for propositional separation logic is PSPACE-complete.
- See complexity of other fragments in [Reynolds, LICS 02].

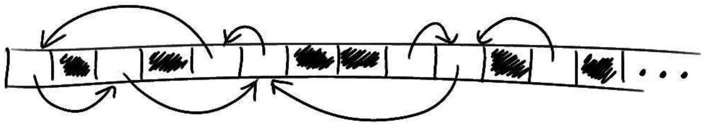
$\boxed{\textbf{Separation logic with one field}}$

Memory states (one field)

- Memory state:
  - Store $s : \mathtt{Var} \rightarrow \mathbb{N}$.
  - Heap $h : \mathbb{N} \rightharpoonup \mathbb{N}$ with finite domain.
    Graph of a unary function with finite domain.



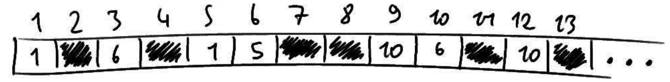
At most one value in a location.



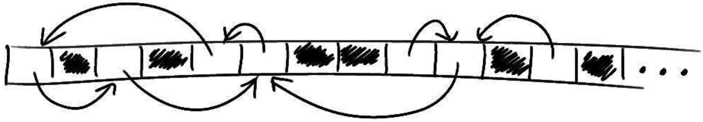
Values are only locations.

Memory states (one field)

- Memory state:
  - Store $s : \mathtt{Var} \rightarrow \mathbb{N}$.
  - Heap $h : \mathbb{N} \rightharpoonup \mathbb{N}$ with finite domain.
    Graph of a unary function with finite domain.



At most one value in a location.



Values are only locations.

- Number of predecessors $\widetilde{\sharp l}$: cardinal of $\{l' : h(l') = l\}$.

$$\widetilde{\sharp 10} \geq 2.$$

Syntax and semantics (bis)

$$\mathcal{A} := \neg\mathcal{A} \mid \mathcal{A} \wedge \mathcal{A} \mid \exists \mathtt{x}.\mathcal{A} \mid \overbrace{\mathtt{x} \hookrightarrow \mathtt{y} \mid \mathtt{x} = \mathtt{y}}^{\text{atomic formulae}} \mid \mathcal{A} * \mathcal{A} \mid \mathcal{A} \mathrel{-\!\!*} \mathcal{A}$$

- Satisfaction relation:

$$
\begin{array}{rcl}
(s,h) \models \neg\mathcal{A} & \text{iff} & \text{not } (s,h) \models \mathcal{A} \\
(s,h) \models \mathcal{A} \wedge \mathcal{B} & \text{iff} & (s,h) \models \mathcal{A} \text{ and } (s,h) \models \mathcal{B} \\
(s,h) \models \exists \mathtt{x}.\ \mathcal{A} & \text{iff} & \text{there is } l \in \mathtt{Loc} \text{ s.t. } (s[\mathtt{x} \mapsto l], h) \models \mathcal{A} \\
(s,h) \models \mathtt{x} \hookrightarrow \mathtt{y} & \text{iff} & h(s(\mathtt{x})) = s(\mathtt{y}) \\
(s,h) \models \mathtt{x} = \mathtt{y} & \text{iff} & s(\mathtt{x}) = s(\mathtt{y}) \\
(s,h) \models \mathcal{A}_1 * \mathcal{A}_2 & \text{iff} & \text{there are two heaps } h_1, h_2 \text{ such that} \\
 & & h = h_1 * h_2, (s,h_1) \models \mathcal{A}_1 \text{ and } (s,h_2) \models \mathcal{A}_2 \\
(s,h) \models \mathcal{A}_1 \mathrel{-\!\!*} \mathcal{A}_2 & \text{iff} & \text{for all heaps } h' \perp h, \\
 & & \text{if } (s,h') \models \mathcal{A}_1 \text{ then } (s,h' * h) \models \mathcal{A}_2.
\end{array}
$$

A selection of properties in $\mathtt{SL}$

- The value of $\mathtt{x}$ is in the domain of the heap: $\mathtt{alloc}\,(\mathtt{x}) \stackrel{\text{def}}{=} \exists \mathtt{y}\; \mathtt{x} \hookrightarrow \mathtt{y}$.

- The heap has a unique cell $\mathtt{x} \mapsto \mathtt{y}$:

$$\mathtt{x} \mapsto \mathtt{y} \stackrel{\text{def}}{=} \mathtt{x} \hookrightarrow \mathtt{y} \wedge \neg \exists \mathtt{z}\; \mathtt{z} \neq \mathtt{x} \wedge \mathtt{alloc}\,(\mathtt{z})$$

- The domain of the heap is empty: emp $\stackrel{\text{def}}{=} \neg \exists \mathtt{x}.\mathtt{alloc}\,(\mathtt{x})$

- $\mathtt{x}$ has at least n predecessors (two options):

$$\exists \mathtt{x}_1, \ldots, \mathtt{x}_n.\ \bigwedge_{i \neq j} \mathtt{x}_i \neq \mathtt{x}_j \wedge \bigwedge_{i=1}^{n} \mathtt{x}_i \hookrightarrow \mathtt{x}$$

$$\overbrace{(\exists \mathtt{y}.\ \mathtt{y} \hookrightarrow \mathtt{x}) * \cdots * (\exists \mathtt{y}.\ \mathtt{y} \hookrightarrow \mathtt{x})}^{n \text{ times}} * \top$$

Properties about lists in $\mathtt{SL}(*)$

- The properties below can be expressed in $\mathtt{SL}(*)$:
  - (s, h) contains *only* a list between $\mathtt{x}$ and $\mathtt{y}$: $\mathsf{ls}(\mathtt{x}, \mathtt{y})$.
  - There is a list between $\mathtt{x}$ and $\mathtt{y}$: $\mathtt{x} \rightarrow^* \mathtt{y}$.
- List properties and other recursive properties can be easily expressed in second-order logics.

Weak second-order logic $\mathtt{SO}$ (or how to speak differently about memory states)

- Family $(\mathtt{VAR}^i)_{i \geq 1}$ of second-order variables interpreted as finite relations.

- Environment $\mathcal{E}$: valuation for variables in $(\mathtt{VAR}^i)_{i \geq 1}$.

- Satisfaction relation:

$$
\begin{array}{lll}
(s,h),\mathcal{E} \models \exists \mathtt{P}\ \mathcal{A} & \text{iff} & \text{there is a finite subset } \mathcal{R} \text{ of } \mathtt{Loc}^n, \\
 & & \text{ such that } (s,h),\mathcal{E}[\mathtt{P} \mapsto \mathcal{R}] \models \mathcal{A} \\
(s,h),\mathcal{E} \models \mathtt{P}(\mathtt{x}_1,\cdots,\mathtt{x}_n) & & \\
 & \text{iff} & (s(\mathtt{x}_1),\ldots,s(\mathtt{x}_n)) \in \mathcal{E}(\mathtt{P})
\end{array}
$$

- Fragments: $\mathtt{MSO}$ (only $\mathtt{VAR}^1$) & $\mathtt{DSO}$ (only $\mathtt{VAR}^2$)

- $\mathrm{L} \sqsubseteq \mathrm{L}'$ whenever for every $\mathcal{A} \in \mathrm{L}$, there is $\mathcal{A}' \in \mathrm{L}'$ that holds true in the same memory states.

$\mathtt{SL} \sqsubseteq \mathtt{DSO}$ (internalization of $\mathtt{SL}$ semantics)

- Abbreviations:
  - $\mathit{heap}(\mathtt{P}) \stackrel{\text{def}}{=} \forall \mathtt{x}, \mathtt{y}, \mathtt{z}.\ \mathtt{xPy} \wedge \mathtt{xPz} \Rightarrow \mathtt{y} = \mathtt{z}$,
  - $\mathtt{P} = \mathtt{Q} * \mathtt{R} \stackrel{\text{def}}{=} \forall \mathtt{x}, \mathtt{y}.\ (\mathtt{xPy} \Leftrightarrow (\mathtt{xQy} \vee \mathtt{xRy})) \wedge \neg(\mathtt{xQy} \wedge \mathtt{xRy})$.

- Translation $\exists \mathtt{P}.\ (\forall \mathtt{x}, \mathtt{y}.\ \mathtt{xPy} \Leftrightarrow \mathtt{x} \hookrightarrow \mathtt{y}) \wedge t_{\mathtt{P}}(\mathcal{A})$:

$$
\begin{aligned}
t_{\mathtt{P}}(\mathtt{x} \hookrightarrow \mathtt{y}) &\stackrel{\text{def}}{=} \mathtt{xPy} \\
t_{\mathtt{P}}(\mathcal{B} * \mathcal{C}) &\stackrel{\text{def}}{=} \exists \mathtt{Q}, \mathtt{Q}'.\ \mathtt{P} = \mathtt{Q} * \mathtt{Q}' \wedge t_{\mathtt{Q}}(\mathcal{B}) \wedge t_{\mathtt{Q}'}(\mathcal{C}) \\
t_{\mathtt{P}}(\mathcal{B} \mathrel{-\!\!*} \mathcal{C}) &\stackrel{\text{def}}{=} \forall \mathtt{Q}.((\exists \mathtt{Q}'.\ \mathit{heap}(\mathtt{Q}') \wedge \mathtt{Q}' = \mathtt{Q} * \mathtt{P}) \wedge \mathit{heap}(\mathtt{Q}) \wedge t_{\mathtt{Q}}(\mathcal{B}) \\
&\quad \Rightarrow (\exists \mathtt{Q}'.\ \mathit{heap}(\mathtt{Q}') \wedge \mathtt{Q}' = \mathtt{Q} * \mathtt{P} \wedge t_{\mathtt{Q}'}(\mathcal{C}))
\end{aligned}
$$

$$\boxed{\textbf{Complexity of } \mathtt{SL}(*)}$$

$\mathtt{SL}(*)$ is decidable

- Weak monadic 2nd order theory of $(D, f, =)$ where
  - D is a countable set,
  - f is a unary function,
  - $=$ is equality,

  is decidable. [Rabin, Trans. of AMS 69]

- $\mathtt{MSO}$ can be translated into this theory.

- $\mathtt{SL}(*) \sqsubseteq \mathtt{MSO}$.

$\mathtt{SL}(*)$ is not elementary recursive (lists as finite words)

- FO3 over finite words is not elementary recursive.
  [Stockmeyer, PhD 74]

- Encoding a word by a list: position i has letter a_j iff the $(i+1)$th location has j predecessors.

- Word formula $\mathcal{B}_{word}$:

  $(\mathtt{x}_{beg} \rightarrow^+ \mathtt{x}_{end}) \wedge (\forall \mathtt{x}\ (\mathtt{x}_{beg} \rightarrow^+ \mathtt{x}) \wedge (\mathtt{x} \rightarrow^+ \mathtt{x}_{end}) \Rightarrow \sharp\mathtt{x} \leq |\Sigma|)$

- Translation of $\mathcal{A}$: $\mathcal{B}_{word} \wedge t(\mathcal{A})$
  - $t(\mathtt{x} < \mathtt{y}) \stackrel{\text{def}}{=} (\mathtt{x} \rightarrow^+ \mathtt{y})$,
  - $t(\forall \mathtt{x}\ \mathcal{B}) \stackrel{\text{def}}{=} \forall \mathtt{x}.\ (\mathtt{x}_{beg} \rightarrow^+ \mathtt{x}) \wedge (\mathtt{x} \rightarrow^+ \mathtt{x}_{end}) \Rightarrow t(\mathcal{B})$,
  - $t(\mathtt{P}_{a_i}(\mathtt{x})) \stackrel{\text{def}}{=} \sharp\mathtt{x} = i$
    (shortcut for a formula in $\mathtt{SL}(*)$ of size $\mathcal{O}(i)$)

$\mathtt{SL}(*)$ is not the ultimate decidable fragment!

- $\mathtt{MSO}$ is strictly more expressive than $\mathtt{SL}(*)$ (and decidable).
  [Antonopoulos & Dawar, FOSSACS'09]

$\mathtt{SL}(*)$ is not the ultimate decidable fragment!

- $\mathtt{MSO}$ is strictly more expressive than $\mathtt{SL}(*)$ (and decidable).
  [Antonopoulos & Dawar, FOSSACS'09]

- Satisfiability for $\mathtt{SL}(* + \mathrel{-\!\!*}^n)$ is also decidable.
  $(s, h) \models \mathcal{A}_1 \mathrel{-\!\!*}^n \mathcal{A}_2$ iff there is $h' \perp h$ such that $|\mathtt{dom}(h')| \leq n$, $(s, h') \models \mathcal{A}_1$ and $(s, h * h') \models \mathcal{A}_2$.

$\mathtt{SL}(*)$ is not the ultimate decidable fragment!

- $\mathtt{MSO}$ is strictly more expressive than $\mathtt{SL}(*)$ (and decidable).
  [Antonopoulos & Dawar, FOSSACS'09]

- Satisfiability for $\mathtt{SL}(* \ + \ \overset{\neg}{*}^{n})$ is also decidable.
  $(s, h) \models \mathcal{A}_1 \overset{\neg}{*}^{n} \mathcal{A}_2$ iff there is $h' \perp h$ such that
  $|\mathtt{dom}(h')| \leq n$, $(s, h') \models \mathcal{A}_1$ and $(s, h * h') \models \mathcal{A}_2$.

- Fragment L:

  $\mathcal{A} ::= \perp \mid \mathtt{x} \mapsto \mathtt{y} \mid \mathtt{size} \leq k \mid \mathtt{size} = k \mid \mathcal{A} * \mathcal{A} \mid \mathcal{A} \vee \mathcal{A} \mid \mathcal{A} \wedge \mathcal{A}$

- Pushing the decidability border further!
  Satisfiability for $\mathtt{SL}$ restricted to formulae such that the left argument of any $-\!*$-formula belongs to L is decidable.

$\mathtt{SL}(-\!\!*)$ is equivalent to $\mathtt{SO}$

[Brochenin & Demri & Lozes, CSL'08]

Proof schema for the equivalence

- $\mathtt{SL}(\twoheadrightarrow\!\!*) \sqsubseteq \mathtt{SL} \sqsubseteq \mathtt{DSO}$ & $\mathtt{SO} \sqsubseteq \mathtt{DSO}$.
- $\mathtt{DSO} \sqsubseteq \mathtt{SL}(\twoheadrightarrow\!\!*)$.
  Encoding finite set of pairs by specialized patterns in memory.
- All translations are in logarithmic space.

Key ingredient: comparing numbers of predecessors

- $\widetilde{\sharp \mathtt{x}} + c \bowtie \widetilde{\sharp \mathtt{y}} + c'$ can be expressed in $\mathtt{SL}(\multimap)$:
  - $\bowtie \in \{<, >, \leq, \geq, =\}$ and $c, c' \in \mathbb{N}$,
  - by a formula of quadratic size in $(c + c')$.

Key ingredient: comparing numbers of predecessors

- $\widetilde{\sharp\mathtt{x}} + c \bowtie \widetilde{\sharp\mathtt{y}} + c'$ can be expressed in $\mathtt{SL}(-\!\!*)$:
  - $\bowtie \in \{<, >, \leq, \geq, =\}$ and $c, c' \in \mathbb{N}$,
  - by a formula of quadratic size in $(c + c')$.
- For instance, $\widetilde{\sharp\mathtt{x}} + c \leq \widetilde{\sharp\mathtt{y}} + c'$ is equivalent to:

$$\forall n \;\; \widetilde{\sharp\mathtt{y}} - c \leq n \text{ implies } \widetilde{\sharp\mathtt{x}} - c' \leq n.$$

1. $\widetilde{\sharp\mathtt{y}} - c \leq n$ is encoded by adding extra arrows in a controlled way.
2. The cardinal of the domain of the extra heap is precisely n.

Key ingredient: comparing numbers of predecessors

- $\widetilde{\sharp \mathtt{x}} + c \bowtie \widetilde{\sharp \mathtt{y}} + c'$ can be expressed in $\mathtt{SL}(\mathrel{-\!\!*})$:
  - $\bowtie \in \{<, >, \leq, \geq, =\}$ and $c, c' \in \mathbb{N}$,
  - by a formula of quadratic size in $(c + c')$.
- For instance, $\widetilde{\sharp \mathtt{x}} + c \leq \widetilde{\sharp \mathtt{y}} + c'$ is equivalent to:

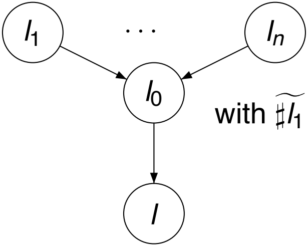
$$\forall n \ \ \widetilde{\sharp \mathtt{y}} - c \leq n \text{ implies } \widetilde{\sharp \mathtt{x}} - c' \leq n.$$

  1. $\widetilde{\sharp \mathtt{y}} - c \leq n$ is encoded by adding extra arrows in a controlled way.
  2. The cardinal of the domain of the extra heap is precisely n.
- Finite runs of Minsky machines can be encoded as memory states.
  ...but establishing $\mathtt{DSO} \sqsubseteq \mathtt{SL}(\mathrel{-\!\!*})$ is stronger than showing undecidability.

Elementary bits: the markers

- A *marker* is a specific pattern in the memory heap.

- A marker of *degree n* and endpoint l.



with $\widetilde{\sharp l_1} = \ldots = \widetilde{\sharp l_n} = 0$

- The location l_0 is an *extremity* in the marker (`extr(z)`).

A discipline on quantifications

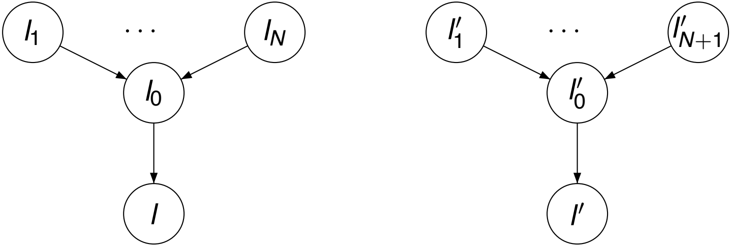
- Quantification over $\mathtt{P}_i$ can only occur in the scope of quantifications over $\mathtt{P}_1, \ldots, \mathtt{P}_{i-1}$.

- Quantifier depth of $\mathcal{B}$ in $\mathcal{A}$: maximal i such that this occurrence of $\mathcal{B}$ is in the scope of $\exists\ \mathtt{P}_i$.

- Translation map of the form $t_i(\mathcal{B})$ depending of the quantifier depth i.

Principle to encode an environment

- A pair $(l, l') \in \mathcal{E}(\mathtt{P}_i)$ is encoded by markers of consecutive degree N and $N+1$.



- The markers are introduced with septraction operator $-\!\!\circledast$.

How to identify the original heap h

- No location has more than k predecessors in h where $s(\mathtt{z}_0^m)$ is the endpoint of some new k-marker.

- Spectrum: sequence of degrees of new markers

$$\bullet \circ \bullet \bullet \circ \bullet \bullet \cdots \circ \bullet \bullet \circ \bullet \bullet \circ \bullet \bullet \cdots \circ \bullet \bullet \circ \bullet$$

$\overset{n}{\bullet}$: There is a unique extremity l with $\widetilde{\sharp l} = n$ (in the environment part)

- A discipline for adding new markers

$$\overset{\mathtt{z}_0^m}{\bullet} \circ \bullet \bullet \ldots \circ \overset{\mathtt{z}_0^M}{\bullet} :: \underbrace{\overset{\mathtt{z}_1^m}{\bullet} \circ \bullet \bullet \ldots \circ \overset{\mathtt{z}_1^M}{\bullet}}_{\text{encodes } \mathcal{E}(\mathtt{P}_1)} = \overset{\mathtt{z}_0^m}{\bullet} \circ \bullet \bullet \ldots \circ \overset{\mathtt{z}_1^M}{\bullet}$$

$$\overset{\mathtt{z}_0^m}{\bullet} \circ \bullet \bullet \ldots \circ \overset{\mathtt{z}_i^M}{\bullet} :: \underbrace{\overset{\mathtt{z}_{i+1}^m}{\bullet} \circ \bullet \bullet \ldots \circ \overset{\mathtt{z}_{i+1}^M}{\bullet}}_{\text{encodes } \mathcal{E}(\mathtt{P}_{i+1})} = \overset{\mathtt{z}_0^m}{\bullet} \circ \bullet \bullet \ldots \circ \overset{\mathtt{z}_{i+1}^M}{\bullet}$$

Translating $\mathtt{P}_j(\mathtt{x}, \mathtt{y})$ – Summary

- $(l, l') \in \mathcal{E}(\mathtt{P}_i)$ iff there are markers with respective endpoint l and l' whose degrees are *consecutive* values strictly between $\widetilde{\sharp \mathtt{z}_i^m}$ and $\widetilde{\sharp \mathtt{z}_i^M}$.

- $\mathtt{z}_i^m$ and $\mathtt{z}_i^M$ are interpreted as locations outside the original memory heap.

- $\widetilde{\sharp \mathtt{z}_i^m}$ is strictly greater than the degree of any location in the original memory heap.

- Translation $t_i(\mathtt{P}_j(\mathtt{x}, \mathtt{y}))$:

$$\exists \mathtt{z}, \mathtt{z}' \, (\mathtt{z} \hookrightarrow \mathtt{x}) \wedge (\mathtt{z}' \hookrightarrow \mathtt{y}) \wedge (\sharp \mathtt{z} > \sharp \mathtt{z}_j^m) \wedge (\sharp \mathtt{z}' < \sharp \mathtt{z}_j^M) \wedge (\sharp \mathtt{z}' = 1 + \sharp \mathtt{z}) \wedge$$

$$\mathtt{extr}(\mathtt{z}) \wedge \mathtt{extr}(\mathtt{z}')$$

Translation

- Translation of $\exists \mathtt{P}_i\ \mathcal{B}$ at the $(i-1)$ quantification depth:

$$\exists \mathtt{z}_i^m, \mathtt{z}_i^M\ \mathtt{isol}(\mathtt{z}_i^m) \wedge \mathtt{isol}(\mathtt{z}_i^M) \wedge$$

$$(\overset{\mathtt{z}_i^m}{\bullet} \circ \bullet \bullet \ldots \circ \overset{\mathtt{z}_i^M}{\bullet} \overset{\neg}{-\!\!*} (\overset{\mathtt{z}_0^m}{\bullet} \circ \bullet \bullet \ldots \circ \overset{\mathtt{z}_i^M}{\bullet} \wedge t_i(\mathcal{B})))$$

  $\mathtt{isol}(\mathtt{x})$ is an abbreviation for $\neg\exists \mathtt{y}.\ (\mathtt{x} \hookrightarrow \mathtt{y}) \vee (\mathtt{y} \hookrightarrow \mathtt{x})$.

- t_i is the identity for $\mathtt{x} = \mathtt{y}$ and $\mathtt{x} \hookrightarrow \mathtt{y}$.

- $t_i(\exists \mathtt{x}\ \mathcal{B})$ is defined as $\exists \mathtt{x}\ \mathtt{notonenv}(\mathtt{x}) \wedge t_i(\mathcal{B})$ where $\mathtt{notonenv}(\mathtt{x})$ guarantees that $\mathtt{x}$ is not interpreted as a location used to encode environments.

Conclusion

Summary

This is mainly about $\mathtt{SL}$ with one selector !

- $\mathtt{SL}$ is as expressive as $\mathtt{SO}$.
- Satisfiability/validity problem for $\mathtt{SL}$ is undecidable.
- $\mathtt{SL}(\mathrel{-\!\!*}) \equiv \mathtt{SL}$: $*$ is redundant in $\mathtt{SL}$.
- $\mathtt{SL}(*)$ is decidable with non-elementary complexity.

Summary

This is mainly about $\mathtt{SL}$ with one selector !

- $\mathtt{SL}$ is as expressive as $\mathtt{SO}$.
- Satisfiability/validity problem for $\mathtt{SL}$ is undecidable.
- $\mathtt{SL}(\twoheadrightarrow) \equiv \mathtt{SL}$: $*$ is redundant in $\mathtt{SL}$.
- $\mathtt{SL}(*)$ is decidable with non-elementary complexity.

$\mathtt{SL}(\twoheadrightarrow) \equiv \mathtt{SL} \equiv \mathtt{SO}$ also holds with more than one selector.
(auxiliary memory cells are even easier to identify)

A selection of open problems for DynRes

- Is $\mathtt{SL}$ restricted to one variable decidable?
  (see Task 2.3 “Decidable fragments”)

- Can we extend further $\mathtt{SL}(*)$ with a weak $-\!\!*$?
  (see Task 2.3 “Decidable fragments”)

- Is $\mathtt{SL2}$ as expressive as $\mathtt{SO}$?
  (see Task 2 “Separation and update: from Expressivity to Decidability”)

- What is the decidability status of $\mathtt{SL}(-\!\!*) \cap \mathtt{SL2}$?
  (see Task 2.3 “Decidable fragments”)

A selection of open problems for DynRes (II)

- Tableaux calculus for `SL` restricted to one variable, if decidable?
  (see Task 3 “Proof Systems for Separation and Update Logics”)

- Automata-based decision procedures for known decidable fragments of SL?
  (see Task 3.1 “Structures, calculi and automata”)