

On the Almighty Wand

Stéphane Demri

LSV, ENS Cachan, CNRS, INRIA Saclay IdF

Joint work with
Rémi Brochenin and Etienne Lozes

November 16th, 2009

Pointer programs

- Pointer: reference to a memory cell (non fixed memory address).
- Dynamic memory allocation/deallocation. (mutable data structures)
- Examples of instructions:
 - $y \rightarrow l := x$: write x to the l -field of the cell pointed to by y ,
 - `free x`: deallocate the cell pointer to by x ,
 - `x := malloc(i)`: allocate i memory cells and assign its address to 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.

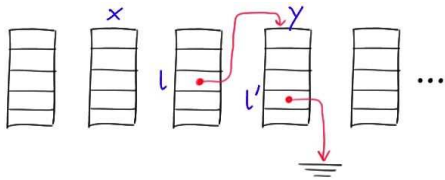
Reasoning about pointer programs

- 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.

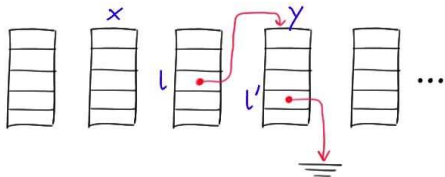
Reasoning about pointer programs

- 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 Var .
- Set of selectors/labels Lab .
- Set of values $\text{Val} = \mathbb{N} \uplus \{\text{nil}\}$.
- Set of stores: $\mathcal{S} \stackrel{\text{def}}{=} \text{Var} \rightarrow \text{Val}$.
- Set of heaps:
 $\mathcal{H} \stackrel{\text{def}}{=} \mathbb{N} \rightarrow_{\text{fin}} (\text{Lab} \rightarrow_{\text{fin}+} \text{Val})$.

Memory state (s, h)

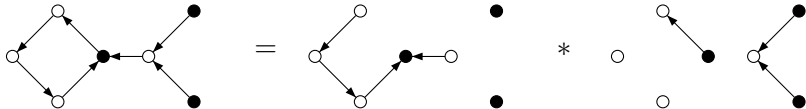
In the sequel, we restrict ourselves to two selectors only or to one selector only.

Disjoint heaps

- h_1 and h_2 are disjoint whenever $\text{dom}(h_1) \cap \text{dom}(h_2) = \emptyset$.
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 $\text{dom}(h_1) \cap \text{dom}(h_2) = \emptyset$.
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 $\text{val} = \mathbb{N}$):



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)
- $\mathcal{A} \text{-} * \mathcal{B}$ 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: $\{\mathcal{A}\} \text{ PROG } \{\mathcal{B}\}$ (total correctness).
- Rule of constancy:

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

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

- Unsoundness of the rule of constancy in separation logic:

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

(when $x = y$)

$x \mapsto z$: “memory has a unique memory cell $x \mapsto z$ ”

When separation logic enters into the play

- Reparation with frame rule:

$$\frac{\{A\} \text{ PROG } \{B\}}{\{A * B'\} \text{ PROG } \{B * B'\}}$$

where no variable free in B' is modified by PROG.

- Strengthening precedent (SP)

$$\frac{A \Rightarrow B' \quad \{B'\} \text{ PROG } \{B\}}{\{A\} \text{ PROG } \{B\}}$$

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

Standard inference rules for mutation

- Local form (MUL)

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

- Global form (MUG)

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

- Backward-reasoning form (MUBR)

$$\frac{}{\{(\exists z. x \mapsto z) * ((x \mapsto y) \text{-} * \mathcal{A})\} [x] := y \{\mathcal{A}\}}$$

Memory states (II)

- Set of variables $\text{Var} = \{x, y, z, \dots\}$.
- Set of locations $\text{Loc} = \{l, l', \dots\}$.
- Set of values $\text{Val} = \mathbb{N} \uplus \text{Loc} \uplus \{\text{nil}\}$.

Memory states (II)

- Set of variables $\text{Var} = \{x, y, z, \dots\}$.
- Set of locations $\text{Loc} = \{l, l', \dots\}$.
- Set of values $\text{Val} = \mathbb{N} \uplus \text{Loc} \uplus \{\text{nil}\}$.
- Memory state:
 - Store $s : \text{Var} \rightarrow \text{Val}$.
 - Heap $h : \text{Loc} \rightarrow \text{Val} \times \text{Val}$ with finite domain.
- Simplification: $\text{Loc} = \text{Val} = \mathbb{N}$.

Memory states (II)

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

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

- Satisfaction relation:

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

Relationship between $*$ and \rightarrow

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

$(A * B) \Rightarrow C$ is valid iff $A \Rightarrow (B \rightarrow C)$ is valid.

Relationship between $*$ and \rightarrow

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

$(A * B) \Rightarrow C$ is valid iff $A \Rightarrow (B \rightarrow C)$ is valid.

- ... but the formula below is not valid

$$((A * B) \Rightarrow C) \Leftrightarrow (A \Rightarrow (B \rightarrow C))$$

Relationship between $*$ and \rightarrow

- \rightarrow 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

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

- Sepraction* \rightarrow : existential version of $*$.

$$\mathcal{A} \rightarrow \mathcal{B} \stackrel{\text{def}}{=} \neg(\mathcal{A} * \neg \mathcal{B})$$

$(s, h) \models \mathcal{A} \rightarrow \mathcal{B}$ 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].
- $D(x) \stackrel{\text{def}}{=} x \hookrightarrow \mathbf{nil}, \mathbf{nil}$.
- Translation

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

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

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].
- $D(\mathbf{x}) \stackrel{\text{def}}{=} \mathbf{x} \hookrightarrow \mathbf{nil}, \mathbf{nil}$.
- Translation

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

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

What is the decidability status with a unique selector?

Complexity of propositional fragments

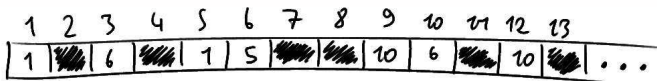
[Calcagno & Yang & O'Hearn, APLAS 01]

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

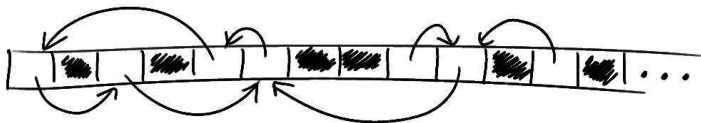
Separation logic with one field

Memory states (one field)

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



At most one value in a location.

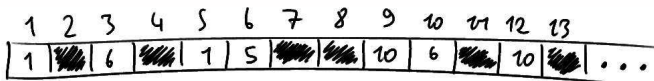


Values are only locations.

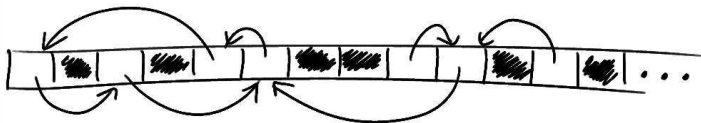
Memory states (one field)

- Memory state:

- Store $s : \text{Var} \rightarrow \mathbb{N}$.
- Heap $h : \mathbb{N} \rightarrow \mathbb{N}$ 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 $\#l$: cardinal of $\{l' : h(l') = l\}$.
 $\#10 \geq 2$.

Syntax and semantics (bis)

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

- Satisfaction relation:

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

A selection of properties in SL

- The value of x is in the domain of the heap:

$$\text{alloc}(x) \stackrel{\text{def}}{=} \exists y. x \hookrightarrow y.$$

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

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

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

- x has at least n predecessors (two options):

$$\exists x_1, \dots, x_n. \bigwedge_{i \neq j} x_i \neq x_j \wedge \bigwedge_{i=1}^n x_i \hookrightarrow x$$

$$\underbrace{(\exists y. y \hookrightarrow x) * \dots * (\exists y. y \hookrightarrow x)}_{n \text{ times}} * \top$$

Properties about lists in $SL(*)$

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

Weak second-order logic SO

(or how to speak differently about memory states)

- Family $(\text{VAR}^i)_{i \geq 1}$ of second-order variables interpreted as **finite** relations.
- Environment \mathcal{E} : valuation for variables in $(\text{VAR}^i)_{i \geq 1}$.
- Satisfaction relation:
 - $(s, h), \mathcal{E} \models \exists P \mathcal{A}$ iff there is a finite subset \mathcal{R} of Loc^n ,
such that $(s, h), \mathcal{E}[P \mapsto \mathcal{R}] \models \mathcal{A}$
 - $(s, h), \mathcal{E} \models P(x_1, \dots, x_n)$
iff $(s(x_1), \dots, s(x_n)) \in \mathcal{E}(P)$
- Fragments: MSO (only VAR^1) & DSO (only VAR^2)
- $L \sqsubseteq L'$ whenever for every $\mathcal{A} \in L$, there is $\mathcal{A}' \in L'$ that holds true in the same memory states.

SL \sqsubseteq DSO (internalization of SL semantics)

- Abbreviations:

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

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

$$t_P(x \hookrightarrow y) \stackrel{\text{def}}{=} xPy$$

$$t_P(\mathcal{B} * \mathcal{C}) \stackrel{\text{def}}{=} \exists Q, Q'. P = Q * Q' \wedge t_Q(\mathcal{B}) \wedge t_{Q'}(\mathcal{C})$$

$$t_P(\mathcal{B} \rightarrow * \mathcal{C}) \stackrel{\text{def}}{=} \forall Q. ((\exists Q'. heap(Q') \wedge Q' = Q * P) \wedge heap(Q) \wedge t_Q(\mathcal{B}) \Rightarrow (\exists Q'. heap(Q') \wedge Q' = Q * P \wedge t_{Q'}(\mathcal{C})))$$

Complexity of $SL(*)$

$SL(*)$ 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]

- MSO can be translated into this theory.
- $SL(*) \sqsubseteq MSO$.

$SL(*)$ 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} :

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

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

$SL(*)$ is not the ultimate decidable fragment!

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

SL(*) is not the ultimate decidable fragment!

- MSO is strictly more expressive than SL(*) (and decidable).
[Antonopoulos & Dawar, FOSSACS'09]
- Satisfiability for $SL(* + \overline{*}^n)$ is also decidable.
 $(s, h) \models \mathcal{A}_1 \overline{*}^n \mathcal{A}_2$ iff there is $h' \perp h$ such that
 $|\text{dom}(h')| \leq n$, $(s, h') \models \mathcal{A}_1$ and $(s, h * h') \models \mathcal{A}_2$.

$SL(*)$ is not the ultimate decidable fragment!

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

- Satisfiability for $SL(* + \overrightarrow{*}^n)$ is also decidable.

$(s, h) \models \mathcal{A}_1 \overrightarrow{*}^n \mathcal{A}_2$ iff there is $h' \perp h$ such that $|\text{dom}(h')| \leq n$, $(s, h') \models \mathcal{A}_1$ and $(s, h * h') \models \mathcal{A}_2$.

- Fragment L:

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

- Pushing the decidability border further!

Satisfiability for SL restricted to formulae such that the left argument of any $\overrightarrow{*}$ -formula belongs to L is decidable.

$SL(\rightarrow^*)$ **is equivalent to** SO
[Brochenin & Demri & Lozes, CSL'08]

Proof schema for the equivalence

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

Key ingredient: comparing numbers of predecessors

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

Key ingredient: comparing numbers of predecessors

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

$$\forall n \#y - c \leq n \text{ implies } \#x - c' \leq n.$$

- 1 $\#y - c \leq n$ 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

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

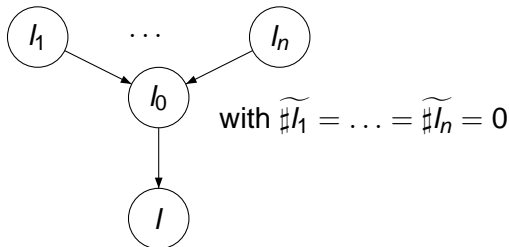
- For instance, $\#x + c \leq \#y + c'$ is equivalent to:

$$\forall n \quad \#y - c \leq n \text{ implies } \#x - c' \leq n.$$

- 1 $\#y - c \leq n$ 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 $DSO \sqsubseteq SL(-*)$ 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 .



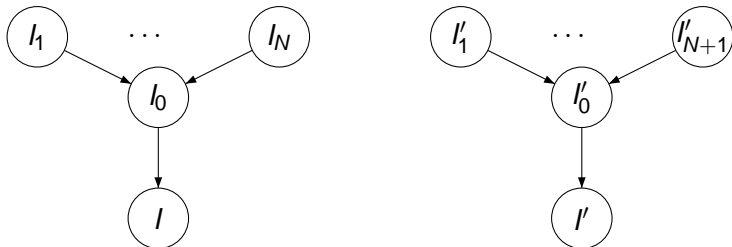
- The location l_0 is an *extremity* in the marker ($\text{extr}(z)$).

A discipline on quantifications

- Quantification over P_i can only occur in the scope of quantifications over P_1, \dots, 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 $\exists P_i$.
- Translation map of the form $t_i(\mathcal{B})$ depending of the quantifier depth i .

Principle to encode an environment

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



- The markers are introduced with septraction operator $\overrightarrow{\ast}$.

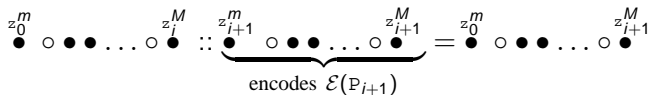
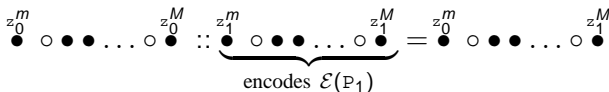
How to identify the original heap h

- No location has more than k predecessors in h where $s(z_0^m)$ is the endpoint of some new k -marker.
- Spectrum: sequence of degrees of new markers



\bullet^n : There is a unique extremity l with $\#l = n$
(in the environment part)

- A discipline for adding new markers



Translating $P_j(x, y)$ – Summary

- $(l, l') \in \mathcal{E}(P_j)$ iff there are markers with respective endpoint l and l' whose degrees are *consecutive* values strictly between $\widetilde{\#z_j^m}$ and $\widetilde{\#z_j^M}$.
- z_j^m and z_j^M are interpreted as locations outside the original memory heap.
- $\widetilde{\#z_j^m}$ is strictly greater than the degree of any location in the original memory heap.
- Translation $t_j(P_j(x, y))$:

$$\exists z, z' (z \hookrightarrow x) \wedge (z' \hookrightarrow y) \wedge (\#z > \#z_j^m) \wedge (\#z' < \#z_j^M) \wedge (\#z' = 1 + \#z) \wedge \text{extr}(z) \wedge \text{extr}(z')$$

Translation

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

$$\exists z_i^m, z_i^M \text{isol}(z_i^m) \wedge \text{isol}(z_i^M) \wedge \\ \left(\overset{z_i^m}{\bullet} \circ \bullet \bullet \dots \circ \overset{z_i^M}{\bullet} \rightarrow^* \left(\overset{z_0^m}{\bullet} \circ \bullet \bullet \dots \circ \overset{z_i^M}{\bullet} \wedge t_i(\mathcal{B}) \right) \right)$$

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

- t_i is the identity for $x = y$ and $x \leftrightarrow y$.
- $t_i(\exists x \mathcal{B})$ is defined as $\exists x \text{notonenv}(x) \wedge t_i(\mathcal{B})$ where $\text{notonenv}(x)$ guarantees that x is not interpreted as a location used to encode environments.

Conclusion

Summary

This is mainly about SL with one selector !

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

Summary

This is mainly about SL with one selector !

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

$SL(-*) \equiv SL \equiv SO$ also holds with more than one selector.
(auxiliary memory cells are even easier to identify)