Two Variables And the Magic Wand

Stéphane Demri
Joint work with Morgan Deters

CNRS – Marie Curie Fellow

Nancy, September 2014
• Heap $\mathcal{h} : \mathbb{N} \rightarrow \mathbb{N}$ with finite domain.
Disjoint heaps

- Disjoint heaps: $\text{dom}(h_1) \cap \text{dom}(h_2) = \emptyset$ (noted $h_1 \perp h_2$).

- When $h_1 \perp h_2$, disjoint heap $h_1 \uplus h_2$. 
Logic 1SL

- Quantified variables $FVAR = \{u_1, u_2, u_3, \ldots \}$.

- Atomic formulae: $\pi ::= u_i = u_j \mid u_i \rightarrow u_j \mid \text{emp} \mid \bot$

- Formulae

$$\phi ::= \pi \mid \phi \land \psi \mid \neg \phi \mid \phi \ast \psi \mid \phi \ast \psi \mid \exists u \phi$$
• Quantified variables \( FVAR = \{ u_1, u_2, u_3, \ldots \} \).

• Atomic formulae: \( \pi ::= u_i = u_j \mid u_i \rightarrow u_j \mid \text{emp} \mid \bot \)

• Formulae

\[
\phi ::= \pi \mid \phi \land \psi \mid \neg \phi \mid \phi \ast \psi \mid \phi \ast \psi \mid \exists u \phi
\]

• \( h \models_f \text{emp} \overset{\text{def}}{\iff} \text{dom}(h) = \emptyset \).

• \( h \models_f u_i = u_j \overset{\text{def}}{\iff} f(u_i) = f(u_j) \).

• \( h \models_f u_i \rightarrow u_j \overset{\text{def}}{\iff} f(u_i) \in \text{dom}(h) \text{ and } h(f(u_i)) = f(u_j) \).
Separating conjunction

\[ h \models_f \phi_1 * \phi_2 \]

\[ \iff \]

for some \( h_1, h_2 \) such that \( h = h_1 \uplus h_2 \),

\[ h_1 \models_f \phi_1 \text{ and } h_2 \models_f \phi_2 \]
Satisfiability problem

- $h \models_f \phi_1 \ast \phi_2 \overset{\text{def}}{\iff} \text{for all } h', \text{ if } h \perp h' \text{ and } h' \models_f \phi_1, \text{ then } h \uplus h' \models_f \phi_2$.

- $h \models_f \exists u \phi \overset{\text{def}}{\iff} \text{there is } l \in \mathbb{N} \text{ such that } h \models_{f[u \mapsto l]} \phi \text{ where } f[u \mapsto l] \text{ is the assignment equal to } f \text{ except that } u \text{ takes the value } l$. 

Each sentence (closed formula) defines a class of heaps.
Satisfiability problem

• \( h \models_f \phi_1 \equiv \phi_2 \iff \text{for all } h', \text{ if } h \perp h' \text{ and } h' \models_f \phi_1, \text{ then } h \cup h' \models_f \phi_2. \)

• \( h \models_f \exists u \phi \iff \text{there is } l \in \mathbb{N} \text{ such that } h \models_f [u \mapsto l] \phi \text{ where } f[u \mapsto l] \text{ is the assignment equal to } f \text{ except that } u \text{ takes the value } l. \)

• Satisfiability problem:
  
  - **input:** formula \( \phi \) in 1SL
  - **question:** are there \( h \) and \( f \) such that \( h \models_f \phi \)?

• Each sentence (closed formula) defines a class of heaps.
Helpful macro: septraction

- Septraction \( \overrightarrow{\star} \): existential version of \( \star \).

\[
\phi_1 \overrightarrow{\star} \phi_2 \overset{\text{def}}{=} \neg (\phi_1 \star \neg \phi_2)
\]

iff there is \( h' \bot h \) such that \( h' \vdash_f \phi_1 \) and \( h' \cup h \vdash_f \phi_2 \).
Simple properties stated in 1SL

- The value of $\overline{u}$ is in the domain of the heap:

$$\text{alloc}(\overline{u}) \overset{\text{def}}{=} \exists u \overline{u} \leftrightarrow u \quad \text{(variant of } (\overline{u} \leftrightarrow \overline{u}) \ast \bot)$$

- The heap has a unique cell $u_1 \mapsto u_2$:

$$u_1 \mapsto u_2 \overset{\text{def}}{=} u_1 \mapsto u_2 \land \neg \exists u' (u' \neq u_1 \land \text{alloc}(u'))$$

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

- $\overline{u}$ has at least $k$ predecessors:

$$\exists u_1, \ldots, u_k \bigwedge_{i \neq j} u_i \neq u_j \land \bigwedge_{i=1}^{k} u_i \mapsto \overline{u}$$

$k$ times

$$\underbrace{(\exists u (u \mapsto \overline{u})) \ast \cdots \ast (\exists u (u \mapsto \overline{u}))}_{k \text{ times}}$$

- Formulae $\#u \sim k$ with $k \in \mathbb{N}$.  

Reachability predicate in 1SL2(\(\ast\))

- Non-empty path from \(u\) to \(\overline{u}\) and nothing else except loops:

\[
reach'(u, \overline{u}) \overset{\text{def}}{=} \#u = 0 \land \text{alloc}(u) \land \neg \text{alloc}(\overline{u}) \land \\
\forall \overline{u} ((\text{alloc}(\overline{u}) \land \#\overline{u} = 0) \Rightarrow \overline{u} = u) \land \\
\forall u ((\#u \neq 0 \land u \neq \overline{u}) \Rightarrow (\#u = 1 \land \text{alloc}(u)))
\]

- There is a path from \(u\) to \(\overline{u}\):

\[
reach(u, \overline{u}) \overset{\text{def}}{=} u = \overline{u} \lor (\top \ast reach'(u, \overline{u}))
\]
Finite binary trees

- The heap is a forest of (possibly incomplete) binary trees:

\[ \forall u (\#u \leq 2 \land \exists \overline{u} (\text{reach}(u, \overline{u}) \land \neg \text{alloc}(\overline{u})) ) \]

- The heap has a single tree:

\[ \exists u \neg \text{alloc}(u) \land (\forall \overline{u} (\text{alloc}(\overline{u}) \Rightarrow \text{reach}(\overline{u}, u))) \]
What is the expressive power of 1SL?

• Is there a sentence stating that there is \( l \) such that \( \tilde{\#}_l > 2 \) and \( \tilde{\#}_l \) is prime?

• Is there a sentence stating that there are \( l_1 \) and \( l_2 \) such that

\[
\tilde{\#}_1 = \tilde{\#}_2 + 6
\]

• Is there a sentence stating that there are \( l_1, l_2, l_3 \) such that

\[
\tilde{\#}_1 = \tilde{\#}_2 \times \tilde{\#}_3 \quad \text{and} \quad \tilde{\#}_1, \tilde{\#}_2, \tilde{\#}_3 \geq 1
\]
Expressive power / Decidability / Complexity

$1SL \equiv DSOL \equiv WSOL \equiv 1SL(\neg \ast)$, undec.

$1SL2$, undec.  $1SL(\ast)$, dec., non-elem.

$1SL2(\neg \ast) \equiv DSOL$, undec.  $1SL2(\ast)$, non-elem.

$1SL1 + PV$, PSPACE-C  $1SL2(\neg \ast)$

$1SL0 + PV$, PSPACE-C

- [Calcagno & Yang & O’Hearn, APLAS’01] $1SL0$
- [Brochenin & Demri & Lozes, IC 12] $1SL(\neg \ast)$
- [Demri & Galmiche & Larchey-Wendling & Mery, CSR’14] $1SL1$
- [Demri & Deters, LICS’14] $1SL2(\neg \ast)$
Weak second-order logic WSOL

- **Formulae:**

\[
\phi ::= u_i = u_j \mid u_i \rightarrow u_j \mid \phi \land \phi \mid \neg \phi \mid \\
\exists u_i \phi \mid \exists P \phi \mid P(u_1, \ldots, u_n)
\]

- \( h \models \exists P \phi \) iff there is a finite \( R \subseteq \mathbb{N}^n \) such that \( h \models [P \mapsto R] \phi \).

- \( h \models P(u_1, \ldots, u_n) \) iff \( (\mathcal{V}(u_1), \ldots, \mathcal{V}(u_n)) \in \mathcal{V}(P) \).

- DSOL: Dyadic fragment of WSOL.
How to express \( \tilde{f}(u_1) = \tilde{f}(u_2) \times \tilde{f}(u_3) \)

\[
\exists P_1, P_2, P_3 \quad (\exists Q (\forall u, \bar{u} (Q(u, \bar{u}) \iff (P_2(u) \land P_3(\bar{u})))) \land \\
(\exists Q' ((\forall \bar{u}_1 P_1(\bar{u}_1) \implies (\exists \bar{u}_2, \bar{u}_3 Q'(\bar{u}_1, \bar{u}_2, \bar{u}_3) \land Q(\bar{u}_2, \bar{u}_3))) \land \\
(\forall \bar{u}_1, \bar{u}_2, \bar{u}_3 Q'(\bar{u}_1, \bar{u}_2, \bar{u}_3) \implies (P_1(\bar{u}_1) \land Q(\bar{u}_2, \bar{u}_3)))) \land \\
(\forall \bar{u}_1, \ldots, \bar{u}_5 \\
Q'(\bar{u}_1, \bar{u}_2, \bar{u}_3) \land Q'(\bar{u}_1, \bar{u}_4, \bar{u}_5) \implies ((\bar{u}_4 = \bar{u}_2) \land (\bar{u}_5 = \bar{u}_3))) \land \\
(Q'(\bar{u}_1, \bar{u}_2, \bar{u}_3) \land Q'(\bar{u}_4, \bar{u}_2, \bar{u}_3) \implies \bar{u}_4 = \bar{u}_1) \land \\
(Q(\bar{u}_2, \bar{u}_3) \implies \exists \bar{u}_6 Q'((\bar{u}_6, \bar{u}_2, \bar{u}_3))))))})
From 1SL to DSOL  
(internalization of 1SL semantics)

\[ hp(P) \overset{\text{def}}{=} \forall u, u', u'' (P(u, u') \land P(u, u'')) \Rightarrow u' = u'' \]

\[ P = Q \ast R \overset{\text{def}}{=} \forall u, u' (P(u, u') \iff (Q(u, u') \lor R(u, u')) \land \neg(Q(u, u') \land R(u, u'))) \]

- Translation \( \exists P (\forall u, u' P(u, u') \iff u \hookrightarrow u') \land t_P(\phi): \)

\[ t_P(u \hookrightarrow u') \overset{\text{def}}{=} P(u, u') \]

\[ t_P(\psi \ast \varphi) \overset{\text{def}}{=} \exists Q, Q' P = Q \ast Q' \land t_Q(\psi) \land t_{Q'}(\varphi) \]

\[ t_P(\psi \rightarrow \varphi) \overset{\text{def}}{=} \forall Q ((\exists Q' hp(Q') \land Q' = Q \ast P) \land hp(Q) \land t_Q(\psi)) \Rightarrow (\exists Q' hp(Q') \land Q' = Q \ast P \land t_{Q'}(\varphi)) \]
From WSOL to DSOL

• For every sentence $\phi$ in WSOL, there is a sentence $\phi'$ in DSOL (computable in logspace) such that for all heaps $h$, $h \models \phi$ iff $h \models \phi'$.

• $P(u) \mapsto P^{new}(u, u)$.

• $P(u_1, \ldots, u_n) \mapsto \exists u \ \wedge_{i=1}^{n} P^{new}_i(u, u_i)$.

• So, it remains to show how to encode DSOL into $1SL2(\neg\star)$. 
Structure of the proof DSOL into 1SL(¬∗)

Principles from [Brochenin & Demri & Lozes, IC 12]

(1) To express \( \#u_j + k \sim \#u_j + k' \) in 1SL(¬∗).

(2) To encode the second-order valuation as a disjoint subheap by using arithmetical constraints to identify patterns.
Structure of the proof DSOL into 1SL(−*)

Principles from [Brochenin & Demri & Lozes, IC 12]

(1) To express $\#u_i + k \sim \#u_j + k'$ in 1SL(−*).

(2) To encode the second-order valuation as a disjoint subheap by using arithmetical constraints to identify patterns.

- Both steps require an unbounded amount of variables.
- Even easy steps such as expressing $\#u \geq k$ become problematic with only two variables and no separating conjunction.

\[
\exists u_1, \ldots, u_k \bigwedge_{i \neq j} u_i \neq u_j \bigwedge_{i=1}^{k} u_i \leftrightarrow u \text{ or } \underbrace{(\exists \bar{u} (\bar{u} \leftrightarrow u)) \ast \cdots \ast (\exists \bar{u} (\bar{u} \leftrightarrow u))}_{k \text{ times}}
\]
Structure of the proof DSOL into 1SL(\(-\star\))

Principles from [Brochenin & Demri & Lozes, IC 12]

(1) To express \(\#u_i + k \sim \#u_j + k'\) in 1SL(\(-\star\)).

(2) To encode the second-order valuation as a disjoint subheap by using arithmetical constraints to identify patterns.

- Both steps require an unbounded amount of variables.

- Even easy steps such as expressing \(\#u \geq k\) become problematic with only two variables and no separating conjunction.

\[\exists u_1, \ldots, u_k \bigwedge_{i \neq j} u_i \neq u_j \bigwedge_{i=1}^{k} u_i \rightarrow u \text{ or } (\exists \overline{u} (\overline{u} \rightarrow u)) \ast \cdots \ast (\exists \overline{u} (\overline{u} \rightarrow u)) \text{ \(k\) times}\]

- Even more embarrassing: how to express \(\#u = 1\) in 1SL2(\(-\star\))? 
Structure of the proof DSOL into $1SL2(\neg\star)$

- Step I: To express $\#u \geq k$ in $1SL2(\neg\star)$.

- Step II: To express $\#u + k \sim \#u + k'$ in $1SL2(\neg\star)$.

- Step III: To encode the second-order valuation as a disjoint subheap by using arithmetical constraints to identify new patterns and to use only two variables.
Instead of chopping the heap in $k$ disjoint subheaps, we add $\mathcal{O}(k)$ new patterns so that the combined heap satisfies properties witnessing $k$ patterns in the original heap.
Formula for $\#u = 1$

- $\#u \geq 1 \overset{\text{def}}{=} \exists \, u \, (\overline{u} \leftrightarrow u)$.

- $\#u \geq 2 \overset{\text{def}}{=} \exists \, u_1, u_2 \, (u_1 \neq u_2) \land (u_1 \leftrightarrow u) \land (u_2 \leftrightarrow u)$.
  (easy with three variables)

- $\#u \geq 2 \overset{\text{def}}{=} (\exists \, \overline{u} \, (\overline{u} \leftrightarrow u)) \ast (\exists \, \overline{u} \, (\overline{u} \leftrightarrow u))$
  (easy with separating conjunction)
Formula for $\#u = 1$

- $\#u \geq 1 \overset{\text{def}}{=} \exists \overline{u} \ (\overline{u} \leftrightarrow u)$.

- $\#u \geq 2 \overset{\text{def}}{=} \exists u_1, u_2 \ (u_1 \neq u_2) \land (u_1 \leftrightarrow u) \land (u_2 \leftrightarrow u)$.
  (easy with three variables)

- $\#u \geq 2 \overset{\text{def}}{=} (\exists \overline{u} \ (\overline{u} \leftrightarrow u)) \ast (\exists \overline{u} \ (\overline{u} \leftrightarrow u))$
  (easy with separating conjunction)

- When the forks enter into the play.
Forky bussiness

- There is 1fork in 1SL2(−∗) such that for all ℓ, we have ℓ |= 1fork iff ℓ is only made of a single, isolated fork.

- There is forky(u) in 1SL2(−∗) stating that all predecessors of u (possibly except u) are endpoints of some fork.

- There is antiforky(u) in 1SL2(−∗) stating that none of the predecessors of u are endpoints of some fork.

- Formula ♯u = 1:

  \(((u \hookrightarrow u) \land (\forall \vec{u} (u \neq \vec{u}) \Rightarrow \neg(\vec{u} \hookrightarrow u))) \lor \\
  (((\neg(u \hookrightarrow u) \land \#u \geq 1)) \land \\
  (\#u = 0) \not\Rightarrow ((antiforky(u) \land (1fork \not\Rightarrow forky(u))))\)))\)
\[ \#u \leq k \quad (k > 0) \]

\[ \#u \leq k \overset{\text{def}}{=} (u \leftrightarrow u \land \#u \leq k - 1) \lor (\neg(u \leftrightarrow u) \land \#u \leq k) \]

where

- \[ \#u \leq 0 \overset{\text{def}}{=} \neg \exists \bar{u} (\bar{u} \leftrightarrow u \land \bar{u} \neq u) \]

- \[ (k' > 0) \#u \leq k' \overset{\text{def}}{=} \]
  
  \[ (#u = 0) \overset{*}{(\text{antiforky}(u))} \land (1\text{fork} \overset{*}{\cdots} \overset{*}{1\text{fork}} \overset{*}{\text{forky}(u)})^{k' \text{ times}} \]
Principles behing Step II – \( \#u + k \leq \#u + k' \)

- **Preparing the heap:**
  - To destroy any forks and knives whose endpoints are predecessors \( f(u) \) and \( f(\overline{u}) \).
  - To destroy isolated memory cells while maintaining the number of predecessors at \( f(u) \) and \( f(\overline{u}) \).
Principles behind Step II – $\#u + k \leq \#\overline{u} + k'$

- **Preparing the heap:**
  - To destroy any forks and knives whose endpoints are predecessors $f(u)$ and $f(\overline{u})$.
  - To destroy isolated memory cells while maintaining the number of predecessors at $f(u)$ and $f(\overline{u})$.

- **Inequality encoded by universal quantification**
  - Equivalences between:
    (assumption : $\tilde{\#}f(\overline{u}) - k \geq 0$ and $\tilde{\#}f(u) - k' \geq 0$)
    1. $\tilde{\#}f(u) + k \leq \tilde{\#}f(\overline{u}) + k'$.
    2. $\tilde{\#}f(u) - k' \leq \tilde{\#}f(\overline{u}) - k$.
    3. for all $n \in \mathbb{N}$, $n \geq \tilde{\#}f(\overline{u}) - k$ implies $n \geq \tilde{\#}f(u) - k'$.

- Universal quantification simulated by $\neg$.
\[ \#u \leq \#\overline{u} \]
Properties

- There is a formula $\mathsf{ksfs}_{\leq k}$ ($k \geq 0$) in $\mathsf{1SL2}(\star)$ such that for every heap $h$, we have $h \models \mathsf{ksfs}_{\leq k}$ iff $h$ is a collection of knives and forks with exactly $k$ forks.
Properties

• There is a formula $\text{ksfs}_{=k}$ ($k \geq 0$) in $1\text{SL2}(\neg \star)$ such that for every heap $h$, we have $h \models \text{ksfs}_{=k}$ iff $h$ is a collection of knives and forks with exactly $k$ forks.

• Let $k \geq 0$, $h$ be a heap and $f$ be a valuation such that $h \models_f \text{antiforky}(u) \land \text{antiknify}(u)$, $h$ has $n$ isolated memory cells and $m = \#f(u)^\star$.

\[ h \models_f (\text{ksfs}_{=k} \overset{\star}{\rightarrow} \text{forky}(u)) \text{ iff } n \geq m - k \]
Final touch - Step II

$$\text{anti}(u, \bar{u}) \overset{\text{def}}{=} \text{antiforky}(u) \land \text{antiknify}(u) \land \text{antiforky}(\bar{u}) \land \text{antiknify}(\bar{u})$$
\textbf{Final touch - Step II}

\begin{align*}
\text{anti}(u, \overline{u}) & \overset{\text{def}}{=} \text{antiforky}(u) \land \text{antiknify}(u) \land \\
\text{antiforky}(\overline{u}) & \land \text{antiknify}(\overline{u})
\end{align*}

\text{comp}(u, \overline{u}, k, k') \overset{\text{def}}{=} \left[(\text{seg} \land \#u = 0 \land \#\overline{u} = 0) \Rightarrow \\
\left(\text{anti}(u, \overline{u}) \Rightarrow \left(\left[\text{ksfs}_{\neq k} \dashv* \text{forky}(\overline{u})\right] \Rightarrow \left[\text{ksfs}_{\neq k'} \dashv* \text{forky}(u)\right]\right)\right)\right]\\
\begin{align*}
& \quad n \geq \overline{\#}^{*}(\overline{u}) - k \\
& \quad n \geq \overline{\#}^{*}(u) - k'
\end{align*}
Final touch - Step II

$$\text{anti}(u, \bar{u}) \overset{\text{def}}{=} \text{antiforky}(u) \land \text{antiknify}(u) \land \text{antiforky}(\bar{u}) \land \text{antiknify}(\bar{u})$$

$$\text{comp}(u, \bar{u}, k, k') \overset{\text{def}}{=} \left( (\text{seg} \land \#u = 0 \land \#\bar{u} = 0) \implies \begin{aligned} (\text{anti}(u, \bar{u}) \Rightarrow \left( \left[ \text{ksfs}_{=k}(\bar{u}) \land \neg(*) \text{forky}(u) \right] \Rightarrow \left[ \text{ksfs}_{=k'}(\bar{u}) \land \neg(*) \text{forky}(u) \right] \right) \right) \end{aligned} \right)$$

\[ n \geq \overline{\text{sf}}(u) - k \text{ and } n \geq \overline{\text{sf}}(\bar{u}) - k' \]

- Suppose $\mathfrak{h} \models \neg(\text{anti}(u, \bar{u})) \land \neg \exists u \text{ isocell}(u)$, \( \overline{\text{sf}}(u)^* - k' \geq 0 \) and \( \overline{\text{sf}}(\bar{u})^* - k \geq 0 \). We have $\mathfrak{h} \models \text{comp}(u, \bar{u}, k, k')$ iff \( \overline{\text{sf}}(u)^* + k \leq \overline{\text{sf}}(\bar{u})^* + k' \).

- Formula for stating \( \overline{\text{sf}}(u) + k \leq \overline{\text{sf}}(\bar{u}) + k' \) can be then defined (a bit of work is still needed).
Step III: from DSOL to 1SL2(*)

- Valuation heap encodes first-order and second-order valuations.

- Pair \((l, l')\) belongs to \(P_i\) whenever \(l\) and \(l'\) can be identified thanks to some special patterns with arithmetical constraints on the number of predecessors.

- To be able to distinguish the original heap from the valuation heap.

- To be able to have distinct patterns for different variables.
\{2, 5, 7, 9\}-well-formed heap
$j$-parentheses of degree 3 and 5

length $(j + 1)$
Encoding valuations

Satisfaction of $P_i(u_j, u_k) \ (j < i < k)$
Translation into $1SL2(\star)$

\[
t\left(P_i(u_j, u_k)\right) \overset{\text{def}}{=} \exists u \left(\text{on}_j(u) \land \exists \overline{u} \left(\overline{u} \leftrightarrow u\right) \land \text{vind}_j(\overline{u}) \land \exists u \left(\#u = \#\overline{u} + 1 \land \text{vind}_i(u) \land \exists \overline{u} \left(u \leftrightarrow \overline{u} \land \text{on}_k(\overline{u})\right)\right)\right)
\]
More about the translation

\[ T(\phi) \overset{\text{def}}{=} \exists u \text{ isoloc}(u) \land (\text{localval}_0(u) \rightarrow \]
\[ (\text{wfh}\{0\} \land \text{imin}(u) \land (\forall \overline{u}((u \neq \overline{u}) \land \neg \text{lrp}_0(\overline{u})) \Rightarrow (\#\overline{u} < \#u) \land t(\{0\}, \phi))) \]
More about the translation

\[ T(\phi) \overset{\text{def}}{=} \exists u \ \text{isoloc}(u) \land (\text{localval}_0(u) \not\sim^* \]
\[ (\text{wfh}_{\{0\}} \land \text{imin}(u) \land (\forall \bar{u} ((u \neq \bar{u}) \land \neg \text{lrp}_0(\bar{u})) \Rightarrow (\#\bar{u} < \#u) \land t(\{0\}, \phi))) \]

- \[ t(X, u_i = u_j) \overset{\text{def}}{=} \exists u \ \text{on}_i(u) \land \text{on}_j(u). \]
More about the translation

\[ T(\phi) \overset{\text{def}}{=} \exists u \text{ isoloc}(u) \land (\text{localval}_0(u) \rightarrow)
\]
\[ (\text{wfh}_0 \land \text{imin}(u) \land (\forall \overline{u}(u \neq \overline{u}) \land \neg \text{lrp}_0(\overline{u})) \Rightarrow (\#\overline{u} < \#u) \land t(\{0\}, \phi)) \]

• \[ t(X, u_i = u_j) \overset{\text{def}}{=} \exists u \text{ on}_i(u) \land \text{on}_j(u). \]

• \[ t(X, u_i \leftrightarrow u_j) \overset{\text{def}}{=} \exists u \exists \overline{u} (\text{on}_i(u) \land \text{on}_j(\overline{u}) \land u \leftrightarrow \overline{u}). \]
More about the translation

\[ T(\phi) \overset{\text{def}}{=} \exists u \text{ isoloc}(u) \land (\text{localval}_0(u) \overset{\rightarrow}{\notin} \) \\
(\text{wfh}_{\{0\}} \land \text{imin}(u) \land (\forall \overline{u} ((u \neq \overline{u}) \land \neg \text{lrp}_0(\overline{u})) \Rightarrow (\#\overline{u} < \#u) \land t(\{0\}, \phi))) \]

- \[ t(X, u_i = u_j) \overset{\text{def}}{=} \exists u \text{ on}_i(u) \land \text{on}_j(u). \]
- \[ t(X, u_i \leftrightarrow u_j) \overset{\text{def}}{=} \exists u \exists \overline{u} \ (\text{on}_i(u) \land \text{on}_j(\overline{u}) \land u \leftrightarrow \overline{u}). \]
- \[ t(X, \exists u_i \psi) \overset{\text{def}}{=} \exists u \exists \overline{u} ((\text{imin}(u) \land \text{isoloc}(\overline{u})) \land (\text{localval}_i(\overline{u}) \overset{\rightarrow}{\notin} \) \\
(\text{wfh}_{X \cup \{i\}} \land \text{imin}(u) \land \text{llp}_i(\overline{u}) \land t(X \cup \{i\}, \psi))))}
More about the translation

\[ T(\phi) \overset{\text{def}}{=} \exists u \ \text{isoloc}(u) \land (\text{localval}_0(u) \overset{\sim}{\rightarrow} \) \\
(\text{wfh}_{\{0\}} \land \text{imin}(u) \land (\forall \overline{u} ((u \neq \overline{u}) \land \neg \text{lrp}_0(\overline{u})) \Rightarrow (\#\overline{u} < \#u) \land t(\{0\}, \phi))) \]

- \[ t(X, u_i = u_j) \overset{\text{def}}{=} \exists u \ \text{on}_i(u) \land \text{on}_j(u). \]
- \[ t(X, u_i \leftrightarrow u_j) \overset{\text{def}}{=} \exists u \exists \overline{u} (\text{on}_i(u) \land \text{on}_j(\overline{u}) \land u \leftrightarrow \overline{u}). \]
- \[ t(X, \exists u_i \psi) \overset{\text{def}}{=} \]
  \[ \exists u \exists \overline{u} ((\text{imin}(u) \land \text{isoloc}(\overline{u})) \land (\text{localval}_i(\overline{u}) \overset{\sim}{\rightarrow} \) \\
(\text{wfh}_{X \cup \{i\}} \land \text{imin}(u) \land \text{llp}_i(\overline{u}) \land t(X \cup \{i\}, \psi))) \]

- \[ t(X, \exists P_i \psi) \overset{\text{def}}{=} \]
  \[ \exists u \exists \overline{u} ((\text{imin}(u) \land \text{isoloc}(\overline{u})) \land (\text{localval}_i(\overline{u}) \overset{\sim}{\rightarrow} \) \\
(\text{wfh}_{X \cup \{i\}} \land \text{imin}(u) \land \text{llp}_i(\overline{u}) \land t(X \cup \{i\}, \psi))) \]
Properties of the translation

- \( \psi \) subformula of \( \phi \) with \( (\forall x(\psi) \cup \{0\}) \subseteq X \subseteq [0, K] \).
  \( X \)-well-formed \( h = h_B \uplus h_V \) and extracted valuation \( \mathcal{V}_h \).
  Then, \( h_B \models_{\mathcal{V}_h} \psi \iff h \models t(X, \psi) \).

- For every sentence \( \phi \) in DSOL, for every heap \( h \), we have \( h \models \phi \iff h \models T(\phi) \).
Conclusion

- WSOL and $1SL2(\rightarrow\star)$ have the same expressive power.
- Satisfiability problem for $1SL2(\rightarrow\star)$ is undecidable.
- The set of valid formulae in $1SL2(\rightarrow\star)$ is not recursively enumerable.
- Robustness of principles in [Brochenin & Demri & Lozes, IC 12].

What’s next?