Two Variables And the Magic Wand

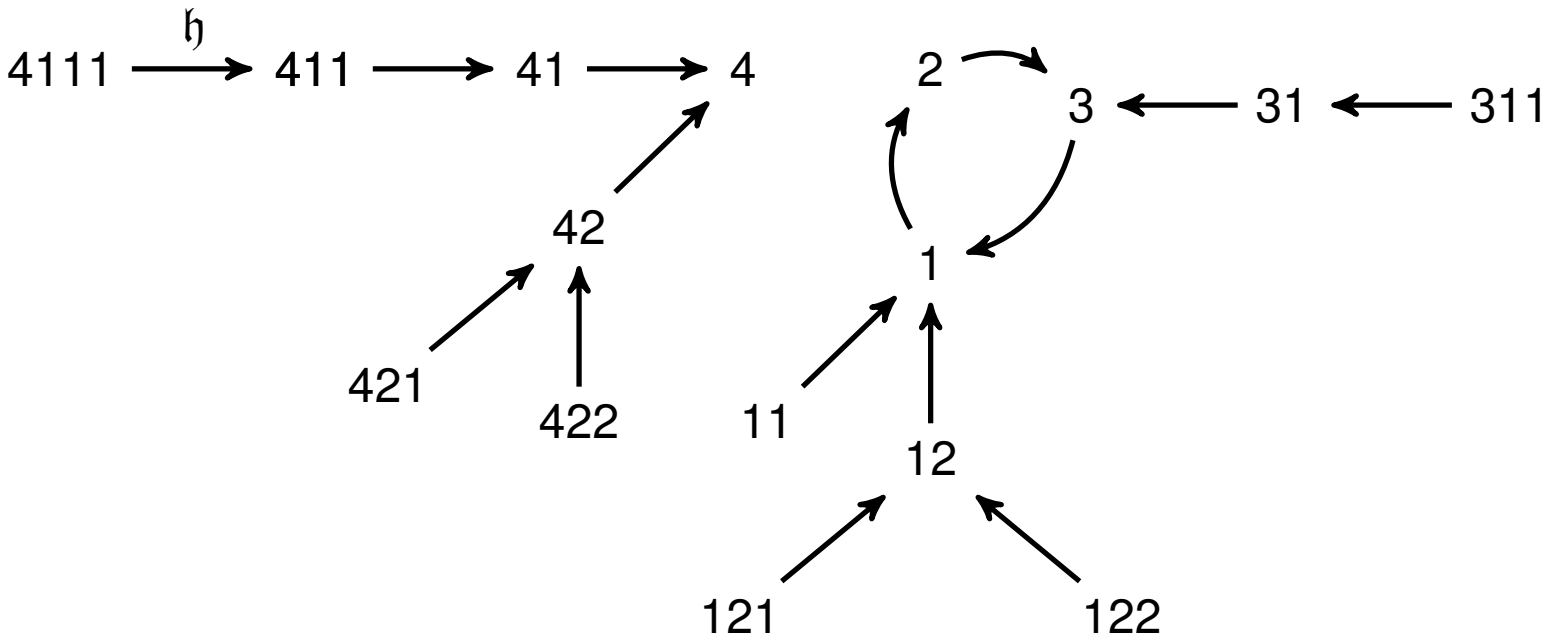
Stéphane Demri
Joint work with Morgan Deters

CNRS – Marie Curie Fellow

Nancy, September 2014

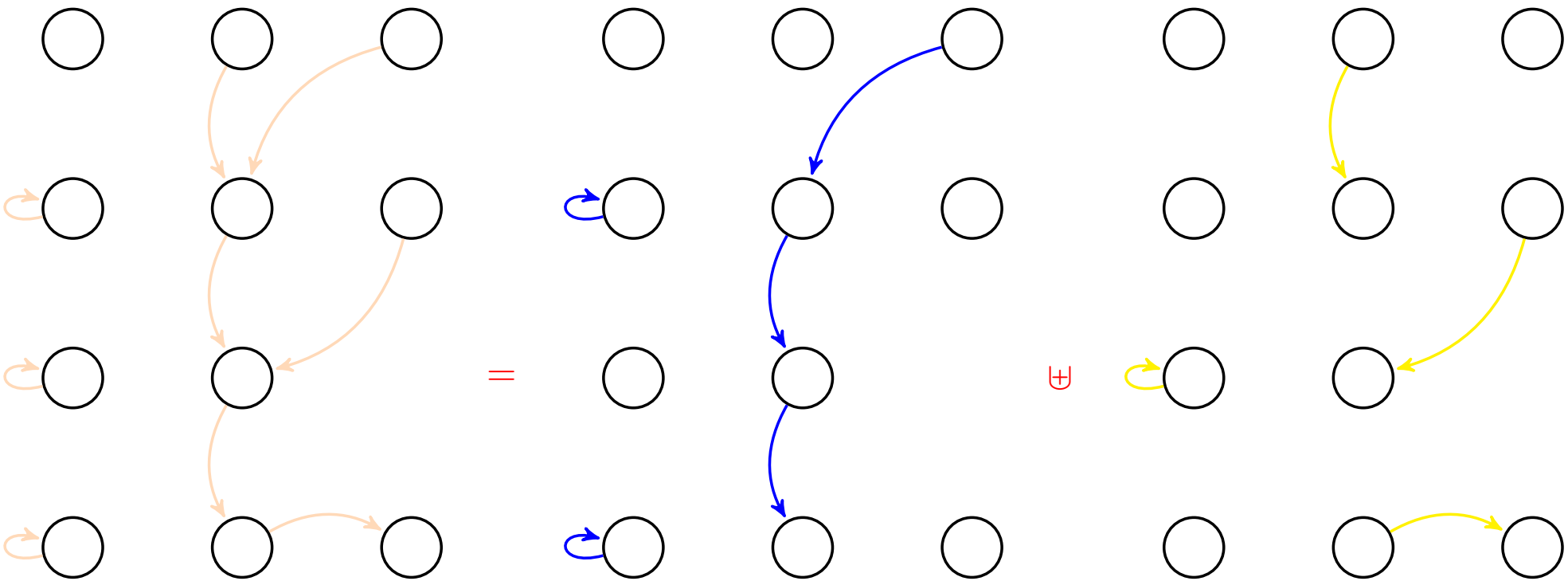
Heaps

- Heap $\mathfrak{h} : \mathbb{N} \rightharpoonup \mathbb{N}$ with finite domain.



Disjoint heaps

- Disjoint heaps: $\mathrm{dom}(\mathfrak{h}_1) \cap \mathrm{dom}(\mathfrak{h}_2) = \emptyset$ (noted $\mathfrak{h}_1 \perp \mathfrak{h}_2$).
- When $\mathfrak{h}_1 \perp \mathfrak{h}_2$, disjoint heap $\mathfrak{h}_1 \uplus \mathfrak{h}_2$.



Logic 1SL

- Quantified variables $\mathtt{FVAR} = \{\mathtt{u}_1, \mathtt{u}_2, \mathtt{u}_3, \ldots\}$.
- Atomic formulae: $\pi ::= \mathtt{u}_i = \mathtt{u}_j \mid \mathtt{u}_i \hookrightarrow \mathtt{u}_j \mid \mathtt{emp} \mid \bot$
- Formulae

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

Logic 1SL

- Quantified variables $\mathtt{FVAR} = \{\mathtt{u}_1, \mathtt{u}_2, \mathtt{u}_3, \ldots\}$.
- Atomic formulae: $\pi ::= \mathtt{u}_i = \mathtt{u}_j \mid \mathtt{u}_i \hookrightarrow \mathtt{u}_j \mid \mathtt{emp} \mid \bot$
- Formulae

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

- $\mathfrak{h} \models_{\mathfrak{f}} \mathtt{emp} \overset{\text{def}}{\Leftrightarrow} \mathsf{dom}(\mathfrak{h}) = \emptyset$.
- $\mathfrak{h} \models_{\mathfrak{f}} \mathtt{u}_i = \mathtt{u}_j \overset{\text{def}}{\Leftrightarrow} \mathfrak{f}(\mathtt{u}_i) = \mathfrak{f}(\mathtt{u}_j)$.
- $\mathfrak{h} \models_{\mathfrak{f}} \mathtt{u}_i \hookrightarrow \mathtt{u}_j \overset{\text{def}}{\Leftrightarrow} \mathfrak{f}(\mathtt{u}_i) \in \mathsf{dom}(\mathfrak{h})$ and $\mathfrak{h}(\mathfrak{f}(\mathtt{u}_i)) = \mathfrak{f}(\mathtt{u}_j)$.

Separating conjunction

$$\mathfrak{h} \models_{\mathfrak{f}} \phi_1 * \phi_2$$

$$\overset{\text{def}}{\Leftrightarrow}$$

for some $\mathfrak{h}_1, \mathfrak{h}_2$ such that $\mathfrak{h} = \mathfrak{h}_1 \uplus \mathfrak{h}_2$,

$$\mathfrak{h}_1 \models_{\mathfrak{f}} \phi_1 \text{ and } \mathfrak{h}_2 \models_{\mathfrak{f}} \phi_2$$

Satisfiability problem

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

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

Satisfiability problem

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

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

- Satisfiability problem:

  input: formula ϕ in 1SL

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

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

Helpful macro: septraction

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

$$\phi_1 \overset{\neg}{\twoheadrightarrow} \phi_2 \overset{\text{def}}{=} \neg(\phi_1 \twoheadrightarrow \neg\phi_2)$$

$$\mathfrak{h} \models_{\mathfrak{f}} \phi_1 \overset{\neg}{\twoheadrightarrow} \phi_2$$

iff

there is $\mathfrak{h}' \perp \mathfrak{h}$ such that $\mathfrak{h}' \models_{\mathfrak{f}} \phi_1$ and $\mathfrak{h}' \uplus \mathfrak{h} \models_{\mathfrak{f}} \phi_2$.

Simple properties stated in 1SL

- The value of $\overline{\mathtt{u}}$ is in the domain of the heap:
  $\mathtt{alloc}(\overline{\mathtt{u}}) \stackrel{\text{def}}{=} \exists \mathtt{u}\ \overline{\mathtt{u}} \hookrightarrow \mathtt{u}$ (variant of $(\overline{\mathtt{u}} \hookrightarrow \overline{\mathtt{u}}) \mathrel{-\!\!*} \bot$)
- The heap has a unique cell $\mathtt{u}_1 \mapsto \mathtt{u}_2$:

$$\mathtt{u}_1 \mapsto \mathtt{u}_2 \stackrel{\text{def}}{=} \mathtt{u}_1 \hookrightarrow \mathtt{u}_2 \wedge \neg \exists \mathtt{u}' \, (\mathtt{u}' \neq \mathtt{u}_1 \wedge \mathtt{alloc}(\mathtt{u}'))$$

- The domain of the heap is empty: $\mathtt{emp} \stackrel{\text{def}}{=} \neg \exists\, \mathtt{u}\ \mathtt{alloc}(\mathtt{u})$
- $\overline{\mathtt{u}}$ has at least k predecessors:

$$\exists \mathtt{u}_1, \ldots, \mathtt{u}_k \bigwedge_{i \neq j} \mathtt{u}_i \neq \mathtt{u}_j \wedge \bigwedge_{i=1}^{k} \mathtt{u}_i \hookrightarrow \overline{\mathtt{u}}$$

$$\overbrace{(\exists\, \mathtt{u}\, (\mathtt{u} \hookrightarrow \overline{\mathtt{u}})) * \cdots * (\exists\, \mathtt{u}\, (\mathtt{u} \hookrightarrow \overline{\mathtt{u}}))}^{k \text{ times}}$$

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

Reachability predicate in 1SL2(∗)

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

$$\begin{aligned}\mathtt{reach}'(\mathtt{u},\overline{\mathtt{u}}) \stackrel{\text{def}}{=}\ & \sharp\mathtt{u} = 0 \wedge \mathtt{alloc}(\mathtt{u}) \wedge \neg\mathtt{alloc}(\overline{\mathtt{u}}) \wedge \\ & \forall\, \overline{\mathtt{u}}\ ((\mathtt{alloc}(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} = 0) \Rightarrow \overline{\mathtt{u}} = \mathtt{u}) \wedge \\ & \forall\, \mathtt{u}\ ((\sharp\mathtt{u} \neq 0 \wedge \mathtt{u} \neq \overline{\mathtt{u}}) \Rightarrow (\sharp\mathtt{u} = 1 \wedge \mathtt{alloc}(\mathtt{u})))\end{aligned}$$

- There is a path from $\mathtt{u}$ to $\overline{\mathtt{u}}$:

$$\mathtt{reach}(\mathtt{u},\overline{\mathtt{u}}) \stackrel{\text{def}}{=} \mathtt{u} = \overline{\mathtt{u}} \vee (\top * \mathtt{reach}'(\mathtt{u},\overline{\mathtt{u}}))$$

Finite binary trees

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

$$\forall\, \mathtt{u}\ (\sharp \mathtt{u} \leq 2 \wedge \exists\, \overline{\mathtt{u}}\ (\mathtt{reach}(\mathtt{u}, \overline{\mathtt{u}}) \wedge \neg \mathtt{alloc}(\overline{\mathtt{u}})))$$

- The heap has a single tree:

$$\exists \mathtt{u}\ \neg \mathtt{alloc}(\mathtt{u}) \wedge (\forall\, \overline{\mathtt{u}}\ (\mathtt{alloc}(\overline{\mathtt{u}}) \Rightarrow \mathtt{reach}(\overline{\mathtt{u}}, \mathtt{u})))$$

What is the expressive power of 1SL ?

- Is there a sentence stating that there is $\mathfrak{l}$ such that $\widetilde{\sharp}\mathfrak{l} > 2$ and $\widetilde{\sharp}\mathfrak{l}$ is prime?

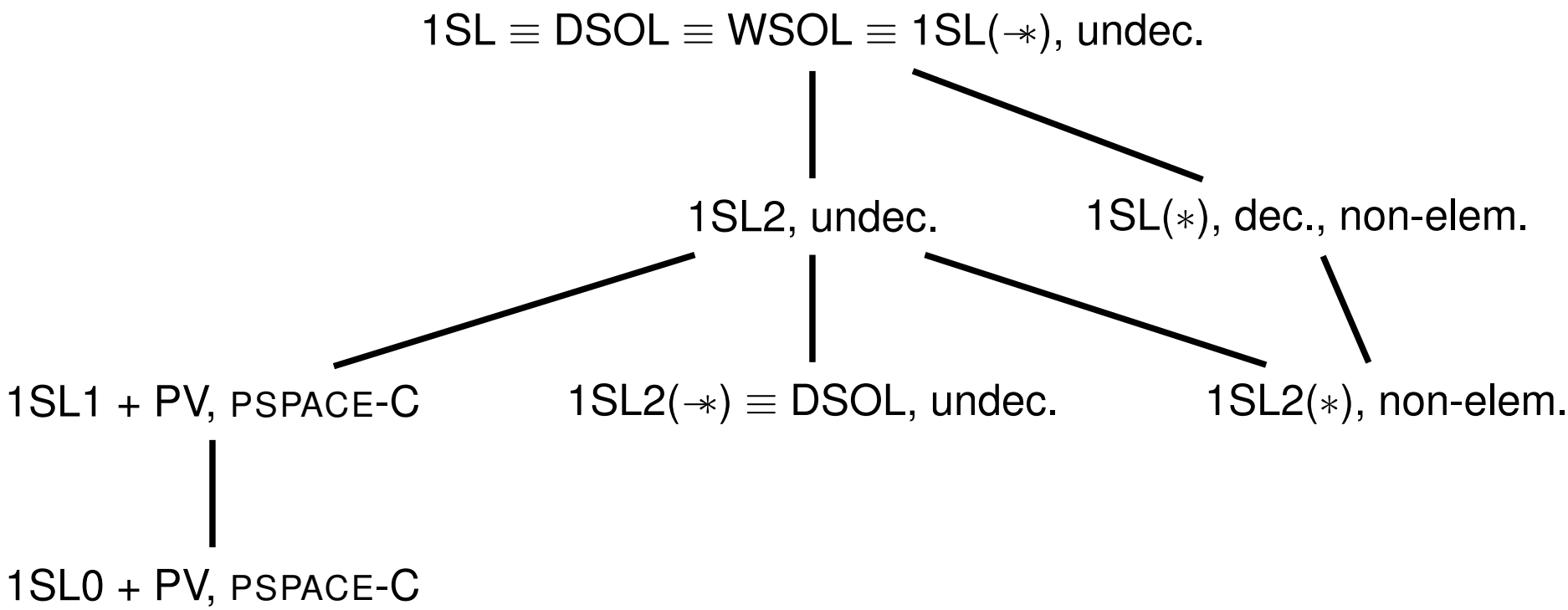
- Is there a sentence stating that there are $\mathfrak{l}_1$ and $\mathfrak{l}_2$ such that

$$\widetilde{\sharp\mathfrak{l}_1} = \widetilde{\sharp\mathfrak{l}_2} + 6 \ ?$$

- Is there a sentence stating that there are $\mathfrak{l}_1$, $\mathfrak{l}_2$, $\mathfrak{l}_3$ such that

$$\widetilde{\sharp\mathfrak{l}_1} = \widetilde{\sharp\mathfrak{l}_2} \times \widetilde{\sharp\mathfrak{l}_3} \ \text{ and } \ \widetilde{\sharp\mathfrak{l}_1}, \widetilde{\sharp\mathfrak{l}_2}, \widetilde{\sharp\mathfrak{l}_3} \geq 1 \ ?$$

Expressive power / Decidability / Complexity



- [Calcagno & Yang & O'Hearn, APLAS'01] 1SL0
- [Brochenin & Demri & Lozes, IC 12] 1SL(−∗)
- [Demri & Galmiche & Larchey-Wendling & Mery, CSR'14] 1SL1
- [Demri & Deters, LICS'14] 1SL2(−∗)

Weak second-order logic WSOL

- Formulae:

$$\phi ::= \mathtt{u}_i = \mathtt{u}_j \mid \mathtt{u}_i \hookrightarrow \mathtt{u}_j \mid \phi \wedge \phi \mid \neg\phi \mid$$

$$\exists\, \mathtt{u}_i\, \phi \mid \exists\, \mathtt{P}\, \phi \mid \mathtt{P}(\mathtt{u}_1, \ldots, \mathtt{u}_n)$$

- $\mathfrak{h} \models_{\mathfrak{V}} \exists\, \mathtt{P}\, \phi$ iff there is a *finite* $R \subseteq \mathbb{N}^n$ such that $\mathfrak{h} \models_{\mathfrak{V}[\mathtt{P} \mapsto R]} \phi$.
- $\mathfrak{h} \models_{\mathfrak{V}} \mathtt{P}(\mathtt{u}_1, \ldots, \mathtt{u}_n)$ iff $(\mathfrak{V}(\mathtt{u}_1), \ldots, \mathfrak{V}(\mathtt{u}_n)) \in \mathfrak{V}(\mathtt{P})$.
- DSOL: Dyadic fragment of WSOL.

How to express $\widetilde{\sharp f(u_1)} = \widetilde{\sharp f(u_2)} \times \widetilde{\sharp f(u_3)}$

$$\exists\, P_1,\ P_2,\ P_3\ ($$

$$(\bigwedge_{i=1}^{3}(\forall\, u\ (u \hookrightarrow u_i) \Leftrightarrow P_i(u))) \wedge$$

$$(\exists\, Q\ (\forall\, u,\ \overline{u}\ (Q(u,\overline{u}) \Leftrightarrow (P_2(u) \wedge P_3(\overline{u}))) \wedge$$

$$(\exists\, Q'$$

$$((\forall\, \overline{u}_1\ P_1(\overline{u}_1) \Rightarrow (\exists\, \overline{u}_2, \overline{u}_3\ Q'(\overline{u}_1, \overline{u}_2, \overline{u}_3) \wedge Q(\overline{u}_2, \overline{u}_3))) \wedge$$

$$(\forall\, \overline{u}_1, \overline{u}_2, \overline{u}_3\ Q'(\overline{u}_1, \overline{u}_2, \overline{u}_3) \Rightarrow (P_1(\overline{u}_1)\ \wedge\ Q(\overline{u}_2, \overline{u}_3))) \wedge$$

$$(\forall\, \overline{u}_1, \ldots, \overline{u}_5$$

$$(Q'(\overline{u}_1, \overline{u}_2, \overline{u}_3)\ \wedge\ Q'(\overline{u}_1, \overline{u}_4, \overline{u}_5) \Rightarrow ((\overline{u}_4 = \overline{u}_2) \wedge (\overline{u}_5 = \overline{u}_3))) \wedge$$

$$(Q'(\overline{u}_1, \overline{u}_2, \overline{u}_3)\ \wedge\ Q'(\overline{u}_4, \overline{u}_2, \overline{u}_3) \Rightarrow \overline{u}_4 = \overline{u}_1) \wedge$$

$$(Q(\overline{u}_2, \overline{u}_3) \Rightarrow \exists\, \overline{u}_6\ Q'(\overline{u}_6, \overline{u}_2, \overline{u}_3)))))))$$

From 1SL to DSOL (internalization of 1SL semantics)

$$\mathtt{hp}(\mathtt{P}) \stackrel{\text{def}}{=} \forall\, \mathtt{u}, \mathtt{u}', \mathtt{u}''\, (\mathtt{P}(\mathtt{u}, \mathtt{u}') \wedge \mathtt{P}(\mathtt{u}, \mathtt{u}'')) \Rightarrow \mathtt{u}' = \mathtt{u}''$$

$$\mathtt{P} = \mathtt{Q} * \mathtt{R} \stackrel{\text{def}}{=} \forall\, \mathtt{u}, \mathtt{u}'\, (\mathtt{P}(\mathtt{u}, \mathtt{u}') \Leftrightarrow (\mathtt{Q}(\mathtt{u}, \mathtt{u}') \vee \mathtt{R}(\mathtt{u}, \mathtt{u}')) \wedge \neg(\mathtt{Q}(\mathtt{u}, \mathtt{u}') \wedge \mathtt{R}(\mathtt{u}, \mathtt{u}'))$$

- Translation $\exists\, \mathtt{P}\ (\forall\, \mathtt{u}, \mathtt{u}'\ \mathtt{P}(\mathtt{u}, \mathtt{u}') \Leftrightarrow \mathtt{u} \hookrightarrow \mathtt{u}') \wedge \mathtt{t}_{\mathtt{P}}(\phi)$:

$$\begin{aligned}
\mathtt{t}_{\mathtt{P}}(\mathtt{u} \hookrightarrow \mathtt{u}') &\stackrel{\text{def}}{=} \mathtt{P}(\mathtt{u}, \mathtt{u}') \\
\mathtt{t}_{\mathtt{P}}(\psi * \varphi) &\stackrel{\text{def}}{=} \exists\, \mathtt{Q}, \mathtt{Q}'\ \mathtt{P} = \mathtt{Q} * \mathtt{Q}' \wedge \mathtt{t}_{\mathtt{Q}}(\psi) \wedge \mathtt{t}_{\mathtt{Q}'}(\varphi) \\
\mathtt{t}_{\mathtt{P}}(\psi \mathrel{-\!\!*} \varphi) &\stackrel{\text{def}}{=} \forall\, \mathtt{Q}\ ((\exists\, \mathtt{Q}'\ \mathtt{hp}(\mathtt{Q}') \wedge \mathtt{Q}' = \mathtt{Q} * \mathtt{P}) \wedge \mathtt{hp}(\mathtt{Q}) \wedge \mathtt{t}_{\mathtt{Q}}(\psi)) \\
&\qquad \Rightarrow (\exists\, \mathtt{Q}'\ \mathtt{hp}(\mathtt{Q}') \wedge \mathtt{Q}' = \mathtt{Q} * \mathtt{P} \wedge \mathtt{t}_{\mathtt{Q}'}(\varphi))
\end{aligned}$$

From WSOL to DSOL

- For every sentence ϕ in WSOL, there is a sentence ϕ' in DSOL (computable in logspace) such that for all heaps $\mathfrak{h}$, $\mathfrak{h} \models \phi$ iff $\mathfrak{h} \models \phi'$.
- $\mathtt{P}(\mathtt{u}) \mapsto \mathtt{P}^{new}(\mathtt{u}, \mathtt{u})$.
- $\mathtt{P}(\mathtt{u}_1, \ldots, \mathtt{u}_n) \mapsto \exists\, \mathtt{u}\ \bigwedge_{i=1}^{n} \mathtt{P}_i^{new}(\mathtt{u}, \mathtt{u}_i)$.
- So, it remains to show how to encode DSOL into 1SL2($\mathrel{-\!\!*}$).

Structure of the proof DSOL into 1SL($\rightarrow\!\!*$)

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

(1) To express $\sharp \mathrm{u}_i + k \sim \sharp \mathrm{u}_j + k'$ in 1SL($\rightarrow\!\!*$).

(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($\rightarrow\!\!*$)

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

(1) To express $\sharp \mathrm{u}_i + k \sim \sharp \mathrm{u}_j + k'$ in 1SL($\rightarrow\!\!*$).

(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 $\sharp \mathrm{u} \geq k$ become problematic with only two variables and no separating conjunction.

$$\exists \mathrm{u}_1, \ldots, \mathrm{u}_k \bigwedge_{i \neq j} \mathrm{u}_i \neq \mathrm{u}_j \wedge \bigwedge_{i=1}^{k} \mathrm{u}_i \hookrightarrow \mathrm{u} \quad \text{or} \quad \overbrace{(\exists\, \overline{\mathrm{u}}\, (\overline{\mathrm{u}} \hookrightarrow \mathrm{u})) * \cdots * (\exists\, \overline{\mathrm{u}}\, (\overline{\mathrm{u}} \hookrightarrow \mathrm{u}))}^{k \text{ times}}$$

Structure of the proof DSOL into 1SL($\rightarrow\!\!*$)

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

(1) To express $\sharp \mathrm{u}_i + k \sim \sharp \mathrm{u}_j + k'$ in 1SL($\rightarrow\!\!*$).

(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 $\sharp \mathrm{u} \geq k$ become problematic with only two variables and no separating conjunction.

$$\exists \mathrm{u}_1, \ldots, \mathrm{u}_k \bigwedge_{i \neq j} \mathrm{u}_i \neq \mathrm{u}_j \wedge \bigwedge_{i=1}^{k} \mathrm{u}_i \hookrightarrow \mathrm{u} \quad \text{or} \quad \overbrace{(\exists\, \overline{\mathrm{u}}\, (\overline{\mathrm{u}} \hookrightarrow \mathrm{u})) * \cdots * (\exists\, \overline{\mathrm{u}}\, (\overline{\mathrm{u}} \hookrightarrow \mathrm{u}))}^{k \text{ times}}$$

- Even more embarassing: how to express $\sharp \mathrm{u} = 1$ in 1SL2($\rightarrow\!\!*$)?

Structure of the proof DSOL into 1SL2($\twoheadrightarrow$)

- Step I: To express $\sharp \mathtt{u} \geq k$ in 1SL2($\twoheadrightarrow$).
- Step II: To express $\sharp \mathtt{u} + k \sim \sharp \overline{\mathtt{u}} + k'$ in 1SL2($\twoheadrightarrow$).
- 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.

A principle behing Step I

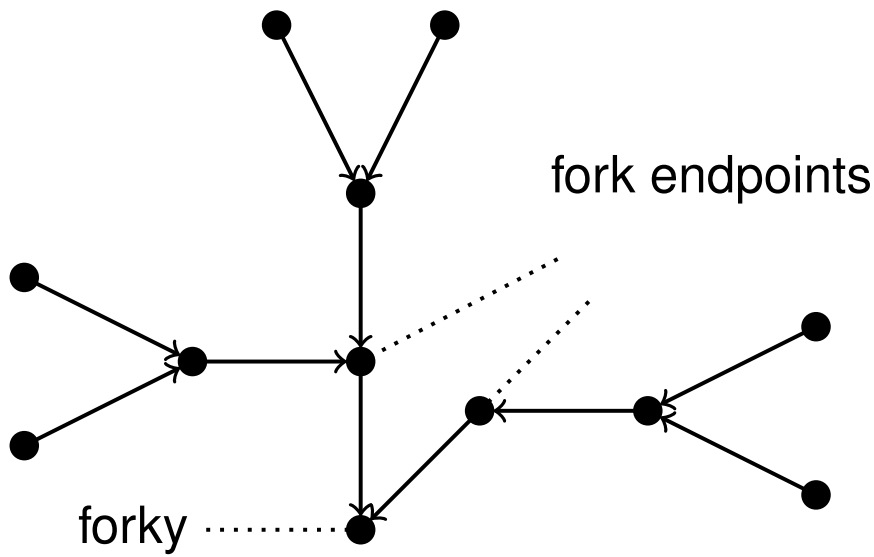
Instead of chopping the heap in k disjoint subheaps, we add $\mathcal{O}(k)$ new patterns so that the combined heap satisfies properties witnessing k patterns in the original heap.

Formula for $\sharp \mathrm{u} = 1$

- $\sharp \mathrm{u} \geq 1 \stackrel{\text{def}}{=} \exists\, \overline{\mathrm{u}}\, (\overline{\mathrm{u}} \hookrightarrow \mathrm{u})$.
- $\sharp \mathrm{u} \geq 2 \stackrel{\text{def}}{=} \exists\, \mathrm{u}_1, \mathrm{u}_2\, (\mathrm{u}_1 \neq \mathrm{u}_2) \wedge (\mathrm{u}_1 \hookrightarrow \mathrm{u}) \wedge (\mathrm{u}_2 \hookrightarrow \mathrm{u})$.
  (easy with three variables)
- $\sharp \mathrm{u} \geq 2 \stackrel{\text{def}}{=} (\exists\, \overline{\mathrm{u}}\, (\overline{\mathrm{u}} \hookrightarrow \mathrm{u})) * (\exists\, \overline{\mathrm{u}}\, (\overline{\mathrm{u}} \hookrightarrow \mathrm{u}))$
  (easy with separating conjunction)

Formula for $\sharp u = 1$

- $\sharp u \geq 1 \stackrel{\text{def}}{=} \exists\, \overline{u}\, (\overline{u} \hookrightarrow u)$.
- $\sharp u \geq 2 \stackrel{\text{def}}{=} \exists\, u_1, u_2\, (u_1 \neq u_2) \wedge (u_1 \hookrightarrow u) \wedge (u_2 \hookrightarrow u)$.
  (easy with three variables)
- $\sharp u \geq 2 \stackrel{\text{def}}{=} (\exists\, \overline{u}\, (\overline{u} \hookrightarrow u)) * (\exists\, \overline{u}\, (\overline{u} \hookrightarrow u))$
  (easy with separating conjunction)
- When the forks enter into the play.



Forky bussiness

- There is $\mathtt{1fork}$ in 1SL2($-\!\!*$) such that for all $\mathfrak{h}$, we have $\mathfrak{h} \models \mathtt{1fork}$ iff $\mathfrak{h}$ is only made of a single, isolated fork.

- There is $\mathtt{forky}(\mathtt{u})$ in 1SL2($-\!\!*$) stating that all predecessors of $\mathtt{u}$ (possibly except $\mathtt{u}$) are endpoints of some fork.

- There is $\mathtt{antiforky}(\mathtt{u})$ in 1SL2($-\!\!*$) stating that none of the predecessors of $\mathtt{u}$ are endpoints of some fork.

- Formula $\sharp\mathtt{u} = 1$:

$$((\mathtt{u} \hookrightarrow \mathtt{u}) \wedge (\forall\, \overline{\mathtt{u}}\ (\mathtt{u} \neq \overline{\mathtt{u}}) \Rightarrow \neg(\overline{\mathtt{u}} \hookrightarrow \mathtt{u})))\vee$$

$$((\neg(\mathtt{u} \hookrightarrow \mathtt{u}) \wedge \sharp\mathtt{u} \geq 1)\wedge$$

$$(\sharp\mathtt{u} = 0) \mathrel{\overrightarrow{-\!\!*}} ((\mathtt{antiforky}(\mathtt{u}) \wedge (\mathtt{1fork} \mathrel{\overrightarrow{-\!\!*}} \mathtt{forky}(\mathtt{u})))))$$

$\sharp \mathrm{u} \leq k \quad (k > 0)$

$$\sharp \mathrm{u} \leq k \stackrel{\text{def}}{=}$$

$$(\mathrm{u} \hookrightarrow \mathrm{u} \wedge \overset{\star}{\sharp} \mathrm{u} \leq k - 1) \vee (\neg(\mathrm{u} \hookrightarrow \mathrm{u}) \wedge \overset{\star}{\sharp} \mathrm{u} \leq k)$$

where

- $\overset{\star}{\sharp} \mathrm{u} \leq 0 \stackrel{\text{def}}{=} \neg \exists\, \overline{\mathrm{u}}\, (\overline{\mathrm{u}} \hookrightarrow \mathrm{u} \wedge \overline{\mathrm{u}} \neq \mathrm{u})$,
- $(k' > 0)\ \overset{\star}{\sharp} \mathrm{u} \leq k' \stackrel{\text{def}}{=}$

$$(\sharp \mathrm{u} = 0) \mathrel{\overrightarrow{-\!\!*}} (\texttt{antiforky}(\mathrm{u}) \wedge (\underbrace{\texttt{1fork} \mathrel{\overrightarrow{-\!\!*}} \cdots \mathrel{\overrightarrow{-\!\!*}} \texttt{1fork}}_{k' \text{ times}} \mathrel{\overrightarrow{-\!\!*}} \texttt{forky}(\mathrm{u}))$$

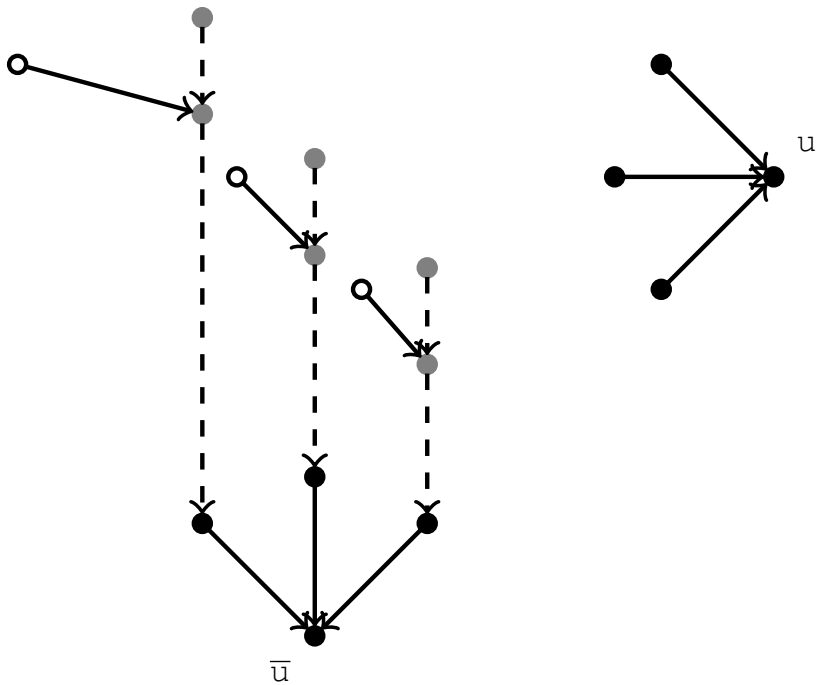
Principles behing Step II – $\sharp \mathtt{u} + k \leq \sharp \overline{\mathtt{u}} + k'$

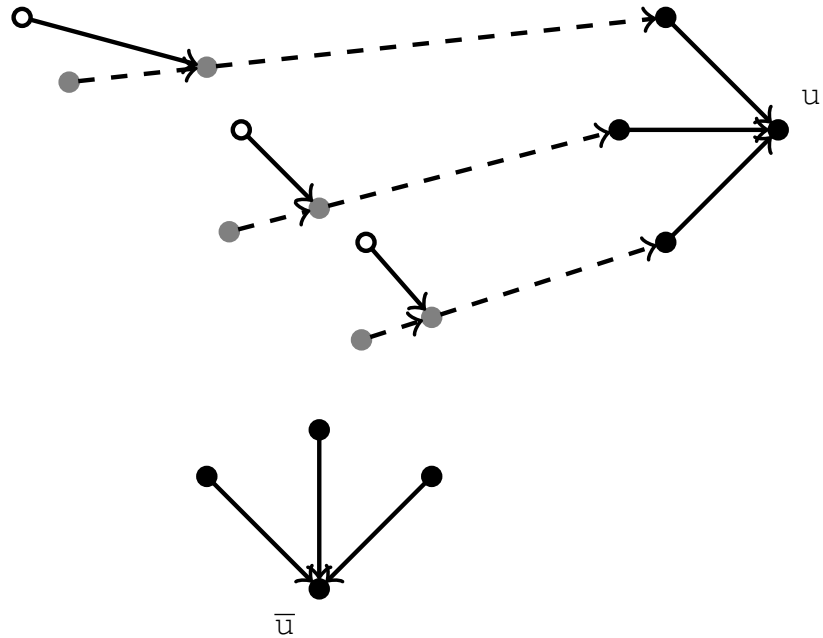
- **Preparing the heap**:
  - To destroy any forks and knives whose endpoints are predecessors $\mathfrak{f}(\mathtt{u})$ and $\mathfrak{f}(\overline{\mathtt{u}})$.
  - To destroy isolated memory cells while maintaining the number of predecessors at $\mathfrak{f}(\mathtt{u})$ and $\mathfrak{f}(\overline{\mathtt{u}})$.

Principles behing Step II – $\sharp \mathtt{u} + k \leq \sharp \overline{\mathtt{u}} + k'$

- **Preparing the heap**:
  - To destroy any forks and knives whose endpoints are predecessors $\mathfrak{f}(\mathtt{u})$ and $\mathfrak{f}(\overline{\mathtt{u}})$.
  - To destroy isolated memory cells while maintaining the number of predecessors at $\mathfrak{f}(\mathtt{u})$ and $\mathfrak{f}(\overline{\mathtt{u}})$.
- **Inequality encoded by universal quantification**
  - Equivalences between:
    (assumption : $\widetilde{\sharp}\mathfrak{f}(\overline{\mathtt{u}}) - k \geq 0$ and $\widetilde{\sharp}\mathfrak{f}(\mathtt{u}) - k' \geq 0$)
    1. $\widetilde{\sharp}\mathfrak{f}(\mathtt{u}) + k \leq \widetilde{\sharp}\mathfrak{f}(\overline{\mathtt{u}}) + k'$.
    2. $\widetilde{\sharp}\mathfrak{f}(\mathtt{u}) - k' \leq \widetilde{\sharp}\mathfrak{f}(\overline{\mathtt{u}}) - k$.
    3. for all $n \in \mathbb{N}$, $n \geq \widetilde{\sharp}\mathfrak{f}(\overline{\mathtt{u}}) - k$ implies $n \geq \widetilde{\sharp}\mathfrak{f}(\mathtt{u}) - k'$.
  - Universal quantification simulated by $-\!\!*$.

$\sharp u \leq \sharp \overline{u}$





Properties

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

Properties

- There is a formula $\texttt{ksfs}_{=k}$ ($k \geq 0$) in 1SL2($\mathrel{-\!\!*}$) such that for every heap $\mathfrak{h}$, we have $\mathfrak{h} \models \texttt{ksfs}_{=k}$ iff $\mathfrak{h}$ is a collection of knives and forks with exactly k forks.

- Let $k \geq 0$, $\mathfrak{h}$ be a heap and $\mathfrak{f}$ be a valuation such that $\mathfrak{h} \models_{\mathfrak{f}} \texttt{antiforky}(\texttt{u}) \wedge \texttt{antiknify}(\texttt{u})$, $\mathfrak{h}$ has n isolated memory cells and $m = \widetilde{\sharp \mathfrak{f}(\texttt{u})}^{\star}$.

$$\mathfrak{h} \models_{\mathfrak{f}} (\texttt{ksfs}_{=k} \mathrel{\neg\!\!*} \texttt{forky}(\texttt{u})) \text{ iff } n \geq m - k$$

Final touch - Step II

$$\texttt{anti}(\texttt{u}, \overline{\texttt{u}}) \stackrel{\text{def}}{=} \texttt{antiforky}(\texttt{u}) \wedge \texttt{antiknify}(\texttt{u}) \wedge$$
$$\texttt{antiforky}(\overline{\texttt{u}}) \wedge \texttt{antiknify}(\overline{\texttt{u}})$$

Final touch - Step II

$$\mathtt{anti}(\mathtt{u},\overline{\mathtt{u}}) \stackrel{\text{def}}{=} \mathtt{antiforky}(\mathtt{u}) \wedge \mathtt{antiknify}(\mathtt{u}) \wedge \mathtt{antiforky}(\overline{\mathtt{u}}) \wedge \mathtt{antiknify}(\overline{\mathtt{u}})$$

$$\mathtt{comp}(\mathtt{u},\overline{\mathtt{u}},k,k') \stackrel{\text{def}}{=} \Big[(\mathtt{seg} \wedge \sharp\mathtt{u} = 0 \wedge \sharp\overline{\mathtt{u}} = 0) \mathrel{-\!\!*} \big(\mathtt{anti}(\mathtt{u},\overline{\mathtt{u}}) \Rightarrow \big(\underbrace{\big[\mathtt{ksfs}_{=k} \mathrel{\neg\!\!*} \mathtt{forky}(\overline{\mathtt{u}})\big]}_{n \geq \widetilde{\sharp\mathtt{f}}^{\star}(\overline{\mathtt{u}})-k} \Rightarrow \underbrace{\big[\mathtt{ksfs}_{=k'} \mathrel{\neg\!\!*} \mathtt{forky}(\mathtt{u})\big]}_{n \geq \widetilde{\sharp\mathtt{f}}^{\star}(\mathtt{u})-k'}\big)\big)\Big]$$

Final touch - Step II

$$\mathtt{anti}(\mathtt{u},\overline{\mathtt{u}}) \stackrel{\text{def}}{=} \mathtt{antiforky}(\mathtt{u}) \wedge \mathtt{antiknify}(\mathtt{u}) \wedge \mathtt{antiforky}(\overline{\mathtt{u}}) \wedge \mathtt{antiknify}(\overline{\mathtt{u}})$$

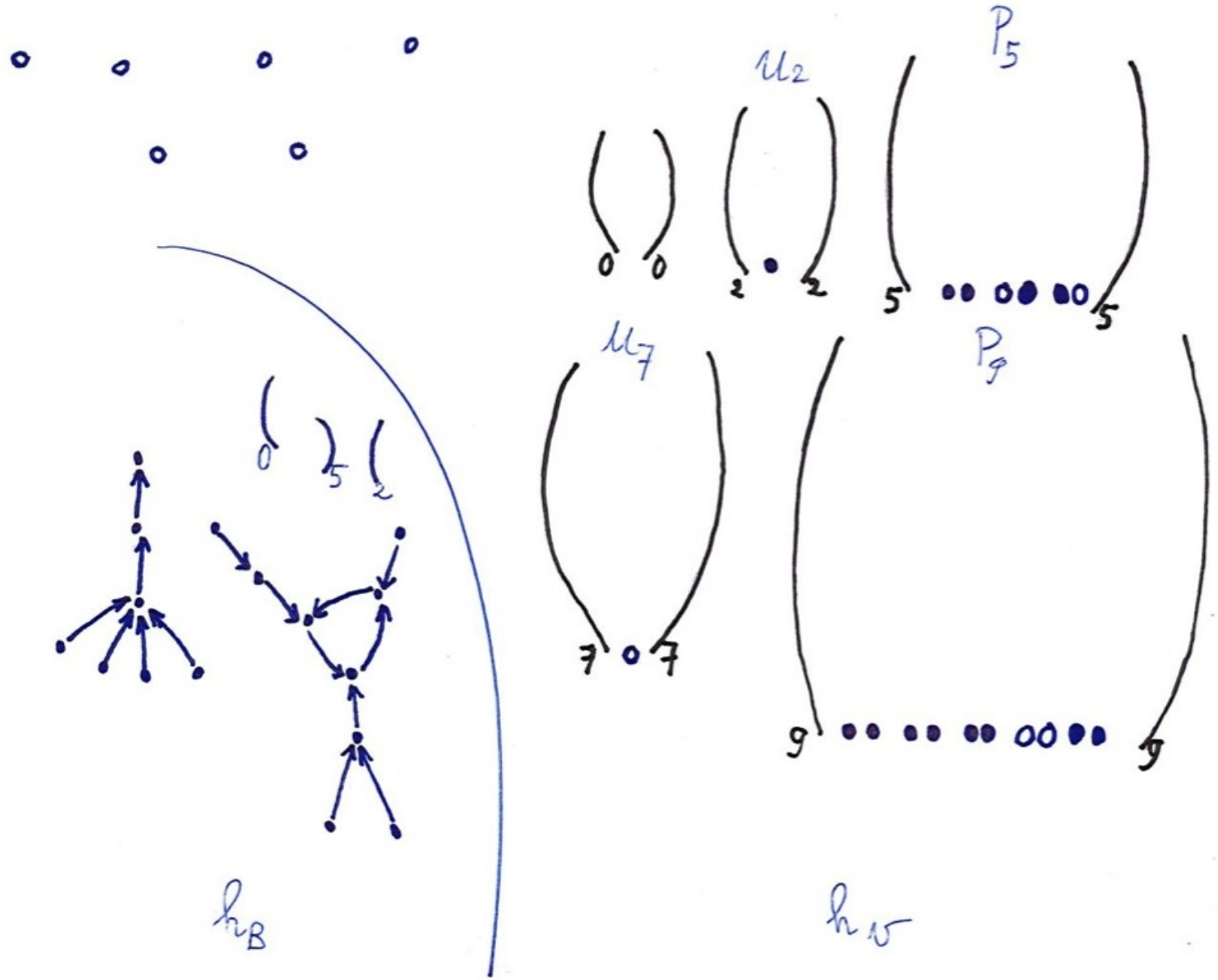
$$\mathtt{comp}(\mathtt{u},\overline{\mathtt{u}},k,k') \stackrel{\text{def}}{=} \Big[(\mathtt{seg} \wedge \sharp\mathtt{u} = 0 \wedge \sharp\overline{\mathtt{u}} = 0) \mathrel{-\!\!*} \big(\mathtt{anti}(\mathtt{u},\overline{\mathtt{u}}) \Rightarrow \big(\underbrace{\big[\mathtt{ksfs}_{=k} \mathrel{\neg\!\!*} \mathtt{forky}(\overline{\mathtt{u}})\big]}_{n \geq \widetilde{\sharp\mathtt{f}}^{\star}(\overline{\mathtt{u}})-k} \Rightarrow \underbrace{\big[\mathtt{ksfs}_{=k'} \mathrel{\neg\!\!*} \mathtt{forky}(\mathtt{u})\big]}_{n \geq \widetilde{\sharp\mathtt{f}}^{\star}(\mathtt{u})-k'}\big)\big)\Big]$$

- Suppose $\mathfrak{h} \models_{\mathfrak{f}} \mathtt{anti}(\mathtt{u},\overline{\mathtt{u}}) \wedge \neg\exists\, \mathtt{u}\ \mathtt{isocell}(\mathtt{u})$, $\widetilde{\sharp\mathtt{f}(\mathtt{u})}^{\star} - k' \geq 0$ and $\widetilde{\sharp\mathtt{f}(\overline{\mathtt{u}})}^{\star} - k \geq 0$. We have $\mathfrak{h} \models_{\mathfrak{f}} \mathtt{comp}(\mathtt{u},\overline{\mathtt{u}},k,k')$ iff $\widetilde{\sharp\mathtt{f}(\mathtt{u})}^{\star} + k \leq \widetilde{\sharp\mathtt{f}(\overline{\mathtt{u}})}^{\star} + k'$.
- Formula for stating $\widetilde{\sharp\mathtt{f}(\mathtt{u})} + k \leq \widetilde{\sharp\mathtt{f}(\overline{\mathtt{u}})} + k'$ can be then defined (a bit of work is still needed).

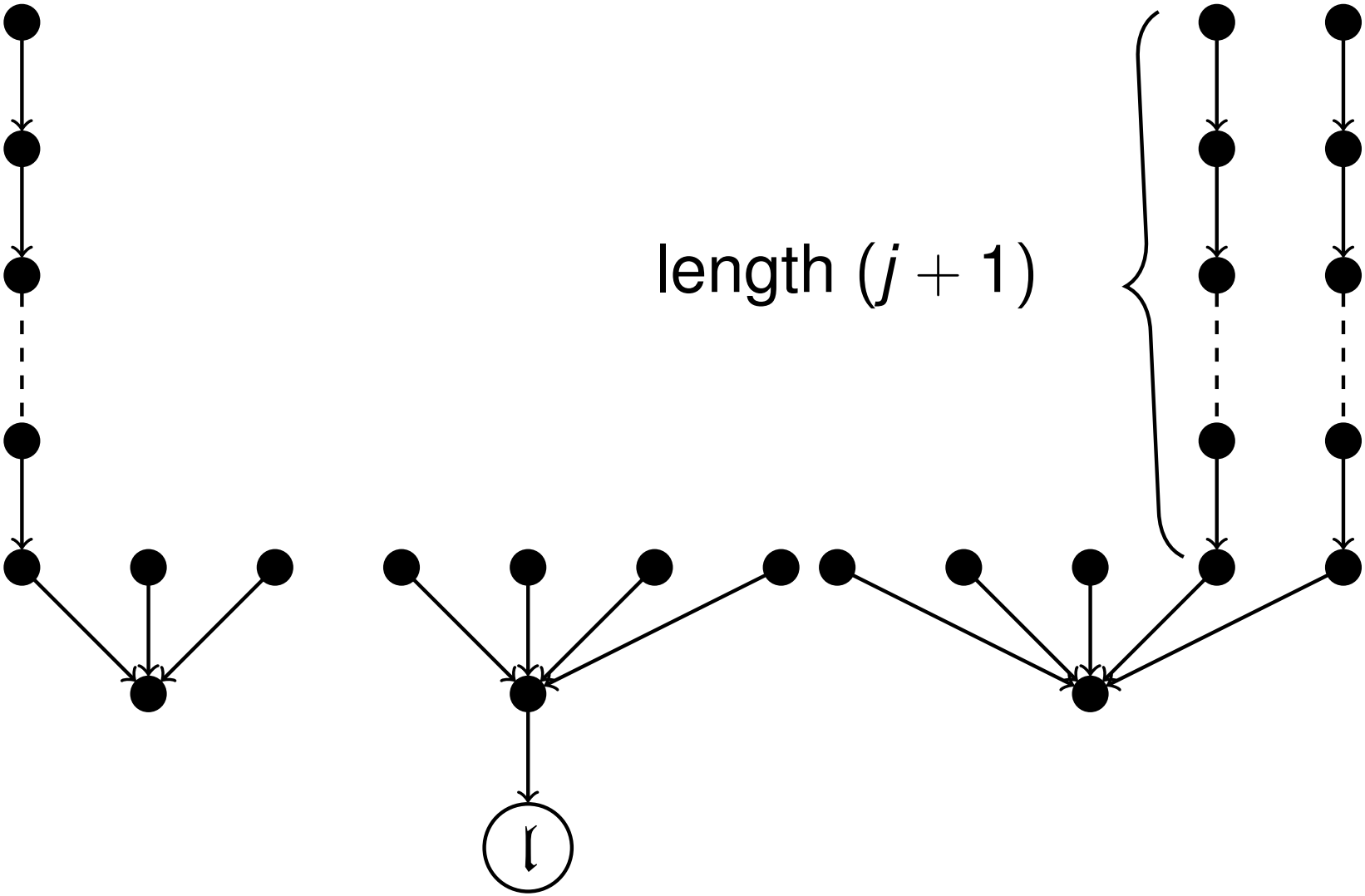
Step III: from DSOL to 1SL2($\twoheadrightarrow$)

- Valuation heap encodes first-order and second-order valuations.
- Pair $(\mathfrak{l}, \mathfrak{l}')$ belongs to $\mathtt{P}_i$ whenever $\mathfrak{l}$ and $\mathfrak{l}'$ 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.

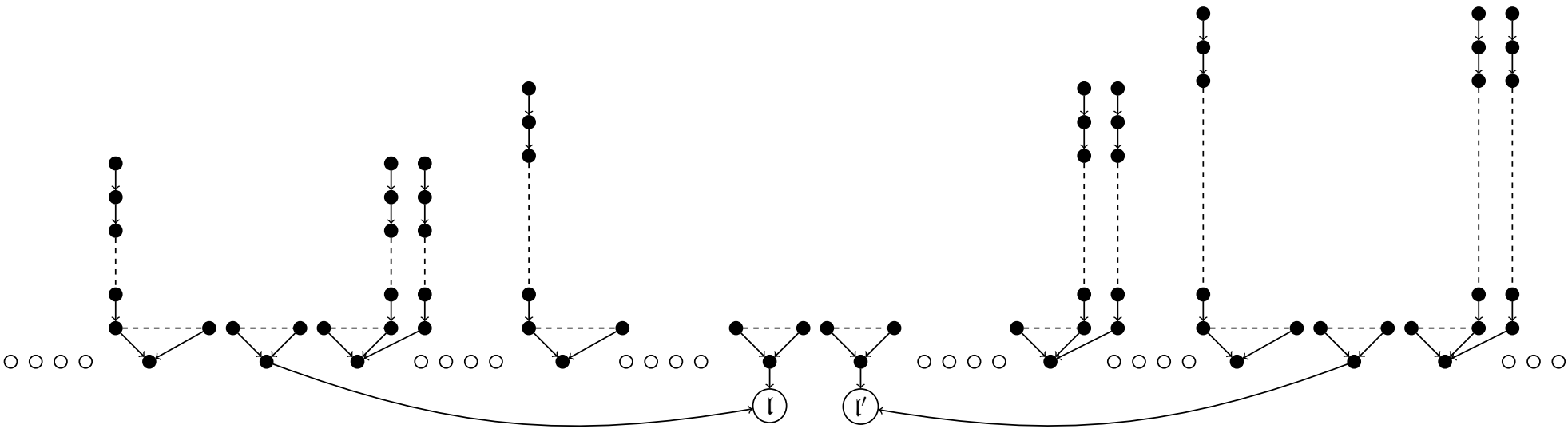
$\{2, 5, 7, 9\}$-well-formed heap



j-parentheses of degree 3 and 5



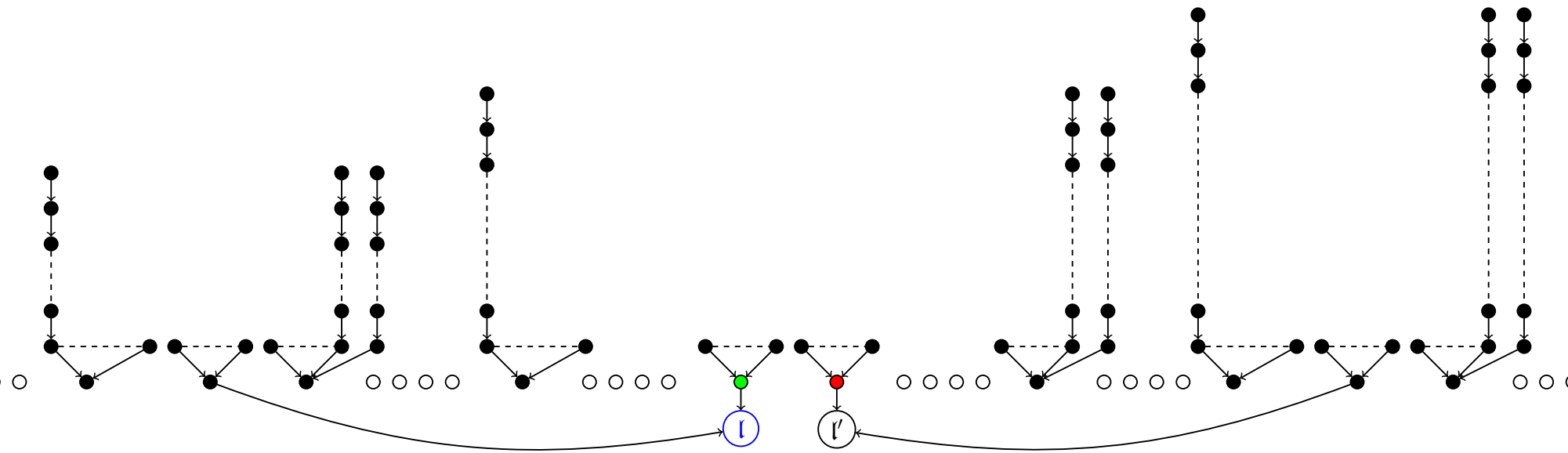
Encoding valuations
Satisfaction of $\mathtt{P}_i(\mathtt{u}_j, \mathtt{u}_k)$ ($j < i < k$)



Translation into 1SL2($\rightarrow\!\!*$)

$$\mathtt{t}\big(\mathtt{P}_i(\mathtt{u}_j,\mathtt{u}_k)\big) \stackrel{\text{def}}{=} \exists\, \mathtt{u}\ \big(\mathtt{on}_j(\mathtt{u}) \wedge \exists\, \overline{\mathtt{u}}\ \big(\overline{\mathtt{u}} \hookrightarrow \mathtt{u} \wedge$$

$$\mathtt{vind}_i(\overline{\mathtt{u}}) \wedge \exists\, \mathtt{u}\ \big(\sharp\mathtt{u} = \sharp\overline{\mathtt{u}} + 1 \wedge \mathtt{vind}_i(\mathtt{u}) \wedge \exists\, \overline{\mathtt{u}}\ \big(\mathtt{u} \hookrightarrow \overline{\mathtt{u}} \wedge \mathtt{on}_k(\overline{\mathtt{u}})\big)\big)\big)\big)$$



More about the translation

$$\mathtt{T}(\phi) \stackrel{\text{def}}{=} \exists\, \mathtt{u}\ \mathtt{isoloc}(\mathtt{u}) \wedge (\mathtt{localval}_{\mathsf{0}}(\mathtt{u}) \mathrel{\neg\!\!*}$$

$$(\mathtt{wfh}_{\{\mathsf{0}\}} \wedge \mathtt{imin}(\mathtt{u}) \wedge (\forall\, \overline{\mathtt{u}}\, ((\mathtt{u} \neq \overline{\mathtt{u}}) \wedge \neg \mathtt{lrp}_{\mathsf{0}}(\overline{\mathtt{u}})) \Rightarrow (\sharp\overline{\mathtt{u}} < \sharp\mathtt{u}) \wedge \mathtt{t}(\{\mathsf{0}\}, \phi)))$$

More about the translation

$$\mathtt{T}(\phi) \stackrel{\text{def}}{=} \exists\, \mathtt{u}\ \mathtt{isoloc}(\mathtt{u}) \wedge (\mathtt{localval}_0(\mathtt{u}) \mathrel{\neg\!\!*}$$
$$(\mathtt{wfh}_{\{0\}} \wedge \mathtt{imin}(\mathtt{u}) \wedge (\forall\, \overline{\mathtt{u}}\, ((\mathtt{u} \neq \overline{\mathtt{u}}) \wedge \neg \mathtt{lrp}_0(\overline{\mathtt{u}})) \Rightarrow (\sharp\overline{\mathtt{u}} < \sharp\mathtt{u}) \wedge \mathtt{t}(\{0\}, \phi)))$$

- $\mathtt{t}(X, \mathtt{u}_i = \mathtt{u}_j) \stackrel{\text{def}}{=} \exists\, \mathtt{u}\ \mathtt{on}_i(\mathtt{u}) \wedge \mathtt{on}_j(\mathtt{u})$.

More about the translation

$$\mathtt{T}(\phi) \stackrel{\text{def}}{=} \exists\, \mathtt{u}\ \mathtt{isoloc}(\mathtt{u}) \wedge (\mathtt{localval}_0(\mathtt{u}) \mathrel{\neg\!\!*}$$
$$(\mathtt{wfh}_{\{0\}} \wedge \mathtt{imin}(\mathtt{u}) \wedge (\forall\, \overline{\mathtt{u}}\, ((\mathtt{u} \neq \overline{\mathtt{u}}) \wedge \neg \mathtt{lrp}_0(\overline{\mathtt{u}})) \Rightarrow (\sharp\overline{\mathtt{u}} < \sharp\mathtt{u}) \wedge \mathtt{t}(\{0\}, \phi)))$$

- $\mathtt{t}(X, \mathtt{u}_i = \mathtt{u}_j) \stackrel{\text{def}}{=} \exists\, \mathtt{u}\ \mathtt{on}_i(\mathtt{u}) \wedge \mathtt{on}_j(\mathtt{u})$.
- $\mathtt{t}(X, \mathtt{u}_i \hookrightarrow \mathtt{u}_j) \stackrel{\text{def}}{=} \exists\, \mathtt{u}\ \exists\, \overline{\mathtt{u}}\ (\mathtt{on}_i(\mathtt{u}) \wedge \mathtt{on}_j(\overline{\mathtt{u}}) \wedge \mathtt{u} \hookrightarrow \overline{\mathtt{u}})$.

More about the translation

$$\mathtt{T}(\phi) \stackrel{\text{def}}{=} \exists\, \mathtt{u}\ \mathtt{isoloc}(\mathtt{u}) \wedge (\mathtt{localval}_0(\mathtt{u}) \mathrel{\neg\!\!*}$$
$$(\mathtt{wfh}_{\{0\}} \wedge \mathtt{imin}(\mathtt{u}) \wedge (\forall\, \overline{\mathtt{u}}\, ((\mathtt{u} \neq \overline{\mathtt{u}}) \wedge \neg \mathtt{lrp}_0(\overline{\mathtt{u}})) \Rightarrow (\sharp\overline{\mathtt{u}} < \sharp\mathtt{u}) \wedge \mathtt{t}(\{0\}, \phi)))$$

- $\mathtt{t}(X, \mathtt{u}_i = \mathtt{u}_j) \stackrel{\text{def}}{=} \exists\, \mathtt{u}\ \mathtt{on}_i(\mathtt{u}) \wedge \mathtt{on}_j(\mathtt{u})$.
- $\mathtt{t}(X, \mathtt{u}_i \hookrightarrow \mathtt{u}_j) \stackrel{\text{def}}{=} \exists\, \mathtt{u}\ \exists\, \overline{\mathtt{u}}\ (\mathtt{on}_i(\mathtt{u}) \wedge \mathtt{on}_j(\overline{\mathtt{u}}) \wedge \mathtt{u} \hookrightarrow \overline{\mathtt{u}})$.
- $\mathtt{t}(X, \exists\, \mathtt{u}_i\ \psi) \stackrel{\text{def}}{=}$

  $\exists\, \mathtt{u}\ \exists\, \overline{\mathtt{u}}\ ((\mathtt{imin}(\mathtt{u}) \wedge \mathtt{isoloc}(\overline{\mathtt{u}})) \wedge (\mathtt{localval}_i(\overline{\mathtt{u}}) \mathrel{\neg\!\!*}$
  $(\mathtt{wfh}_{X \cup \{i\}} \wedge \mathtt{imin}(\mathtt{u}) \wedge \mathtt{llp}_i(\overline{\mathtt{u}}) \wedge \mathtt{t}(X \cup \{i\}, \psi))))$

More about the translation

$$\mathtt{T}(\phi) \stackrel{\text{def}}{=} \exists\, \mathtt{u}\ \mathtt{isoloc}(\mathtt{u}) \wedge (\mathtt{localval}_0(\mathtt{u}) \mathrel{\neg\!\!*}$$
$$(\mathtt{wfh}_{\{0\}} \wedge \mathtt{imin}(\mathtt{u}) \wedge (\forall\, \overline{\mathtt{u}}\, ((\mathtt{u} \neq \overline{\mathtt{u}}) \wedge \neg \mathtt{lrp}_0(\overline{\mathtt{u}})) \Rightarrow (\sharp\overline{\mathtt{u}} < \sharp\mathtt{u}) \wedge \mathtt{t}(\{0\}, \phi)))$$

- $\mathtt{t}(X, \mathtt{u}_i = \mathtt{u}_j) \stackrel{\text{def}}{=} \exists\, \mathtt{u}\ \mathtt{on}_i(\mathtt{u}) \wedge \mathtt{on}_j(\mathtt{u})$.
- $\mathtt{t}(X, \mathtt{u}_i \hookrightarrow \mathtt{u}_j) \stackrel{\text{def}}{=} \exists\, \mathtt{u}\ \exists\, \overline{\mathtt{u}}\ (\mathtt{on}_i(\mathtt{u}) \wedge \mathtt{on}_j(\overline{\mathtt{u}}) \wedge \mathtt{u} \hookrightarrow \overline{\mathtt{u}})$.
- $\mathtt{t}(X, \exists\, \mathtt{u}_i\ \psi) \stackrel{\text{def}}{=}$

  $\exists\, \mathtt{u}\ \exists\, \overline{\mathtt{u}}\ ((\mathtt{imin}(\mathtt{u}) \wedge \mathtt{isoloc}(\overline{\mathtt{u}})) \wedge (\mathtt{localval}_i(\overline{\mathtt{u}}) \mathrel{\neg\!\!*}$
  $(\mathtt{wfh}_{X \cup \{i\}} \wedge \mathtt{imin}(\mathtt{u}) \wedge \mathtt{llp}_i(\overline{\mathtt{u}}) \wedge \mathtt{t}(X \cup \{i\}, \psi))))$

- $\mathtt{t}(X, \exists\, \mathtt{P}_i\ \psi) \stackrel{\text{def}}{=}$

  $\exists\, \mathtt{u}\ \exists\, \overline{\mathtt{u}}\ ((\mathtt{imin}(\mathtt{u}) \wedge \mathtt{isoloc}(\overline{\mathtt{u}})) \wedge (\mathtt{localval}_i(\overline{\mathtt{u}}) \mathrel{\neg\!\!*}$

  $(\mathtt{wfh}_{X \cup \{i\}} \wedge \mathtt{imin}(\mathtt{u}) \wedge \mathtt{llp}_i(\overline{\mathtt{u}}) \wedge \mathtt{t}(X \cup \{i\}, \psi))))$

Properties of the translation

- ψ subformula of ϕ with $(\mathtt{fr}(\psi) \cup \{0\}) \subseteq X \subseteq [0, K]$. X-well-formed $\mathfrak{h} = \mathfrak{h}_B \uplus \mathfrak{h}_V$ and extracted valuation $\mathfrak{V}_\mathfrak{h}$. Then, $\mathfrak{h}_B \models_{\mathfrak{V}_\mathfrak{h}} \psi$ iff $\mathfrak{h} \models \mathtt{t}(X, \psi)$.

- For every sentence ϕ in DSOL, for every heap $\mathfrak{h}$, we have $\mathfrak{h} \models \phi$ iff $\mathfrak{h} \models \mathtt{T}(\phi)$.

Conclusion

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

  What's next?