Deciding the First-Order Theory of an Algebra of Feature Trees with Updates

Nicolas Jeannerod, Ralf Treinen

IRIF, Université Paris-Diderot

June 25, 2018

The CoLiS Project

- ANR project with IRIF, Inria Saclay, Inria Lille.

The CoLiS Project

- ANR project with IRIF, Inria Saclay, Inria Lille.
- Verifying Debian packages

The CoLiS Project

- ANR project with IRIF, Inria Saclay, Inria Lille.
- Verifying Debian packages:
  - A tar archive containing files;
  - A few shell scripts.

The CoLiS Project

- ANR project with IRIF, Inria Saclay, Inria Lille.
- Verifying Debian packages:
  - A tar archive containing files;
  - A few shell scripts.
- Giving them specifications

The CoLiS Project

- ANR project with IRIF, Inria Saclay, Inria Lille.
- Verifying Debian packages:
  - A tar archive containing files;
  - A few shell scripts.
- Giving them specifications:
  - Input:
    - Environment,
    - Execution mode (install, update, removal, purge, ..),
    - Input filesystem;

The CoLiS Project

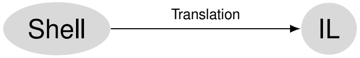
- ANR project with IRIF, Inria Saclay, Inria Lille.
- Verifying Debian packages:
  - A tar archive containing files;
  - A few shell scripts.
- Giving them specifications:
  - Input:
    - Environment,
    - Execution mode (install, update, removal, purge, ..),
    - Input filesystem;
  - Output:
    - Success / Error,
    - Output filesystem.

The CoLiS Project

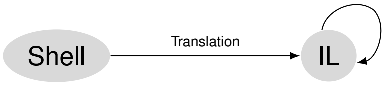
- ANR project with IRIF, Inria Saclay, Inria Lille.
- Verifying Debian packages:
  - A tar archive containing files;
  - A few shell scripts.
- Giving them specifications:
  - Input:
    - Environment,
    - Execution mode (install, update, removal, purge, ..),
    - Input filesystem;
  - Output:
    - Success / Error,
    - Output filesystem.

Big Picture

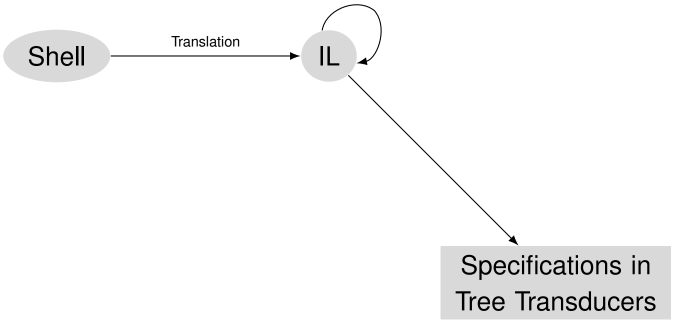
Big Picture



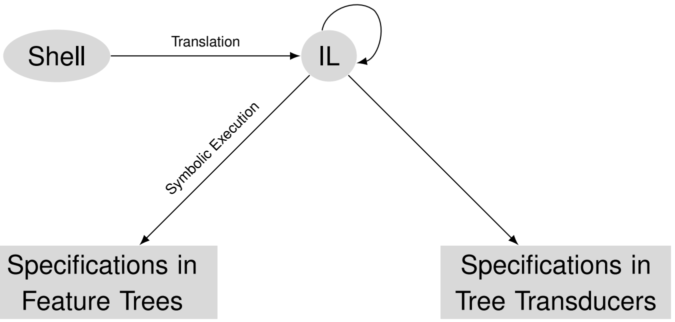
Big Picture



Big Picture

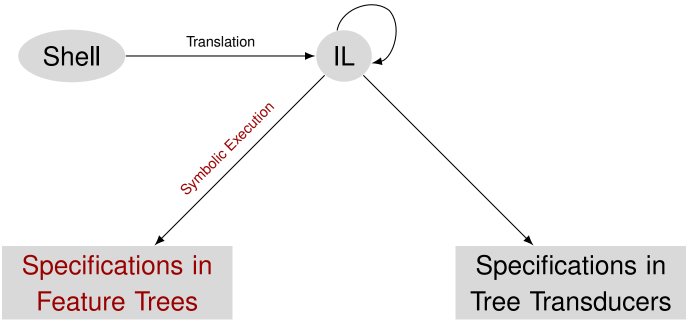


Big Picture



Shell
Translation
IL
Symbolic Execution
Specifications in Feature Trees
Specifications in Tree Transducers

Big Picture


Shell
Translation
IL
Symbolic Execution
Specifications in Feature Trees
Specifications in Tree Transducers

What For?

- Find executions that lead to errors.
  - Provide an understandable explanation of why.

What For?

- Find executions that lead to errors.
  - Provide an understandable explanation of why.
- Check properties

What For?

- Find executions that lead to errors.
  - Provide an understandable explanation of why.
- Check properties:
  - Script doing nothing:
    $\forall in, out \cdot (\texttt{spec}_s(in, out) \leftrightarrow out \doteq in)$

What For?

- Find executions that lead to errors.
  - Provide an understandable explanation of why.
- Check properties:
  - Script doing nothing:
    $\forall in, out \cdot (\texttt{spec}_s(in, out) \leftrightarrow out \doteq in)$
  - Equivalence of two scripts:
    $\forall in, out \cdot (\texttt{spec}_s(in, out) \leftrightarrow \texttt{spec}_t(in, out))$

What For?

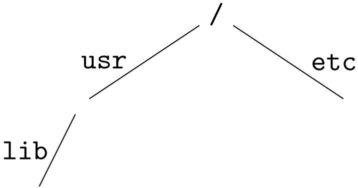
- Find executions that lead to errors.
  - Provide an understandable explanation of why.
- Check properties:
  - Script doing nothing:
    $\forall in, out \cdot (\texttt{spec}_s(in, out) \;\leftrightarrow\; out \doteq in)$
  - Equivalence of two scripts:
    $\forall in, out \cdot (\texttt{spec}_s(in, out) \;\leftrightarrow\; \texttt{spec}_t(in, out))$
  - Script that don't modify `/home`:
    $\forall in, out \cdot (\texttt{spec}_s(in, out) \;\rightarrow\; out[\texttt{home}] = in[\texttt{home}])$

What For?

- Find executions that lead to errors.
  - Provide an understandable explanation of why.
- Check properties:
  - Script doing nothing:
    $\forall in, out \cdot (\texttt{spec}_s(in, out) \leftrightarrow out \doteq in)$
  - Equivalence of two scripts:
    $\forall in, out \cdot (\texttt{spec}_s(in, out) \leftrightarrow \texttt{spec}_t(in, out))$
  - Script that don't modify `/home`:
    $\forall in, out \cdot (\texttt{spec}_s(in, out) \rightarrow out[\texttt{home}] = in[\texttt{home}])$
  - Sequence of scripts that do nothing:
    $\forall in, out \cdot (\exists r \cdot (\texttt{spec}_s(in, r) \wedge \texttt{spec}_t(r, out)) \leftrightarrow out \doteq in)$

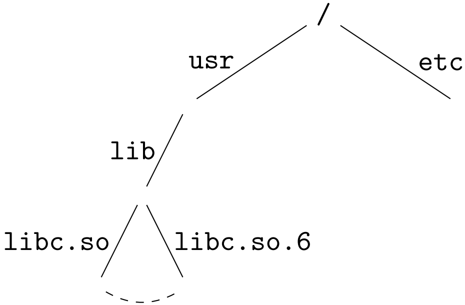
Feature Trees and Update

Unix Filesystem



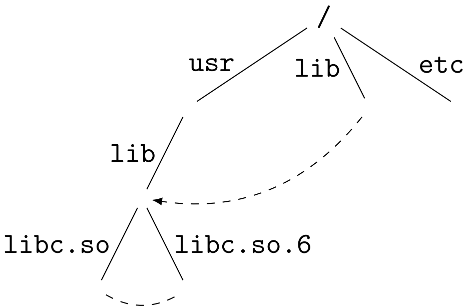
- Basically a tree with labelled nodes and edges;

Unix Filesystem



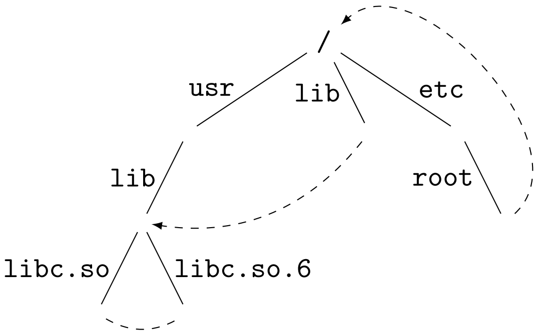
- Basically a tree with labelled nodes and edges;
- There can be sharing at the leafs (hard link between files);

Unix Filesystem



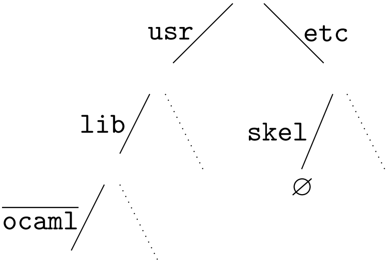
- Basically a tree with labelled nodes and edges;
- There can be sharing at the leafs (hard link between files);
- There can be pointers to other parts of the tree (symbolic links)

Unix Filesystem

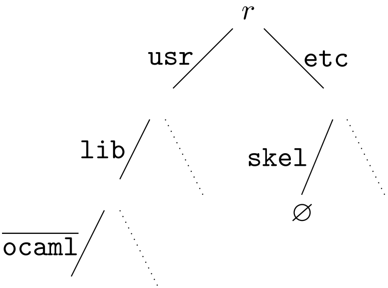


- Basically a tree with labelled nodes and edges;
- There can be sharing at the leafs (hard link between files);
- There can be pointers to other parts of the tree (symbolic links) which may form cycles.

Here Come the Feature Trees

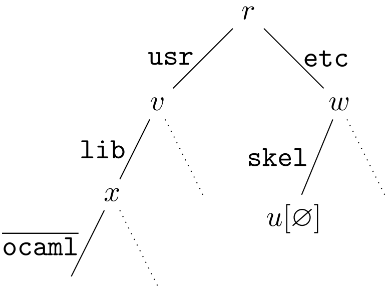


Here Come the Feature Trees



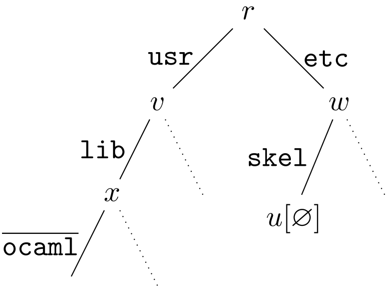
$c(r) =$

Here Come the Feature Trees



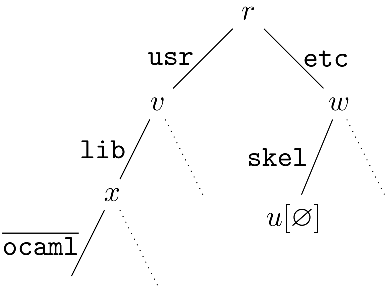
$$c(r) \ = \ \exists u, v, x, w \cdot \Bigg\{$$

Here Come the Feature Trees



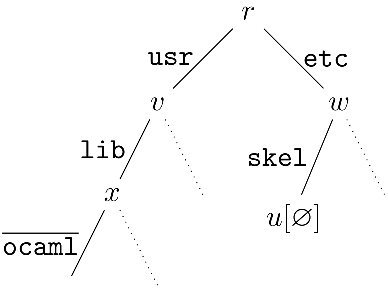
$$c(r) \ = \ \exists u, v, x, w \cdot \begin{cases} r[\texttt{usr}]v \wedge v[\texttt{lib}]x \\ \wedge\, r[\texttt{etc}]w \wedge w[\texttt{skel}]u \end{cases}$$

Here Come the Feature Trees



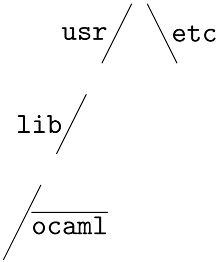
$$c(r) \;=\; \exists u, v, x, w \cdot \left\{ \begin{array}{l} r[\texttt{usr}]v \wedge v[\texttt{lib}]x \wedge x[\texttt{ocaml}]\uparrow \\ \wedge\, r[\texttt{etc}]w \wedge w[\texttt{skel}]u \end{array} \right.$$

Here Come the Feature Trees

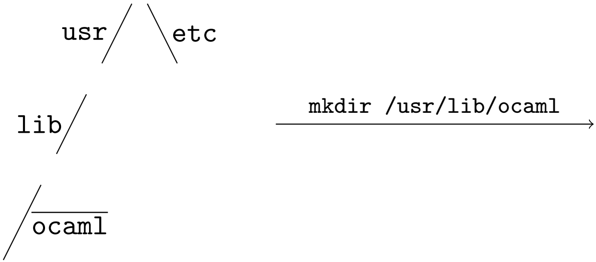


$$c(r) \;=\; \exists u, v, x, w \cdot \left\{ \begin{array}{l} r[\texttt{usr}]v \wedge v[\texttt{lib}]x \wedge x[\texttt{ocaml}]\uparrow \\ \wedge\, r[\texttt{etc}]w \wedge w[\texttt{skel}]u \wedge u[\varnothing] \end{array} \right.$$

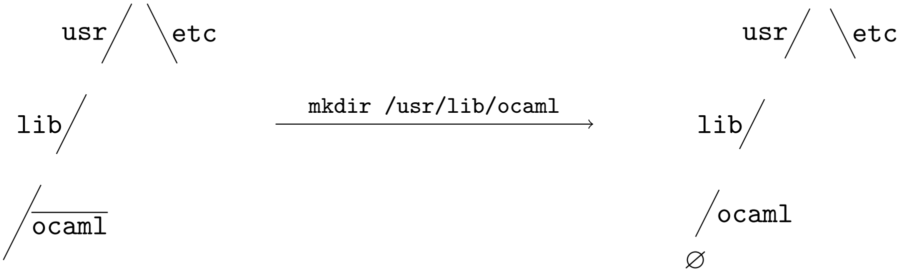
...and Here Come the Update



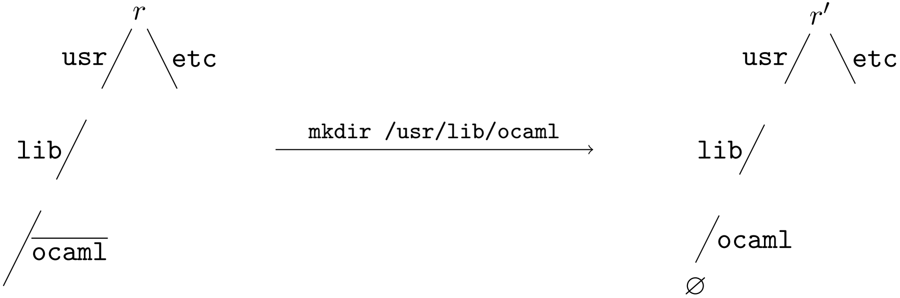
...and Here Come the Update



...and Here Come the Update

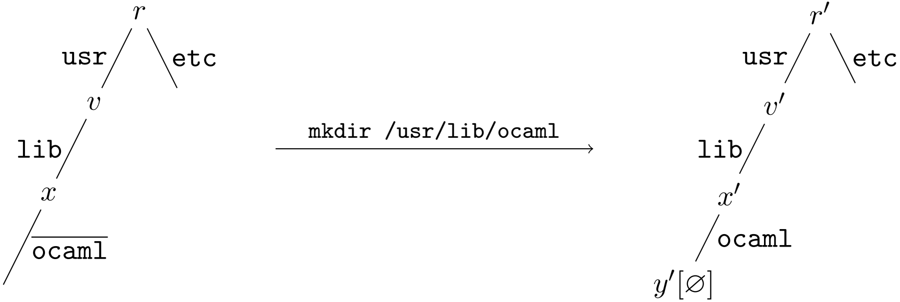


...and Here Come the Update



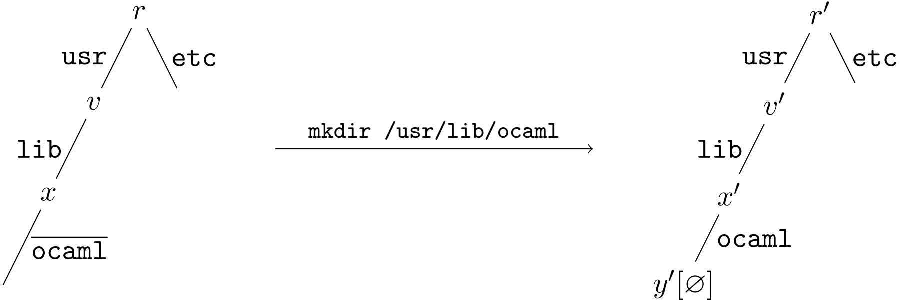
$$c'(r, r') \ =$$

...and Here Come the Update



$$c'(r,r') \ = \ \exists v, v', x, x', y' \cdot \left\{ \right.$$

...and Here Come the Update



$$c'(r, r') \;=\; \exists v, v', x, x', y' \cdot \begin{cases} r' \text{ is } r \text{ with } \texttt{usr} \to v' \\ \wedge\ v' \text{ is } v \text{ with } \texttt{lib} \to x' \\ \wedge\ x' \text{ is } x \text{ with } \texttt{ocaml} \to y' \\ \wedge\ y'[\varnothing] \end{cases}$$

Er.. Is That Really What We Want?

- Asymmetric:

$$y \text{ is } x \text{ with } f \to v$$

Er.. Is That Really What We Want?

- Asymmetric:

$$y \text{ is } x \text{ with } f \to v$$

- Makes it hard to eliminate variables:

$$\exists x \cdot \left(\begin{array}{c} y \text{ is } x \text{ with } f \to v \\ \wedge\ z \text{ is } x \text{ with } g \to w \end{array} \right)$$

Er.. Is That Really What We Want?

- Asymmetric:

$$y \text{ is } x \text{ with } f \to v$$

- Makes it hard to eliminate variables:

$$\exists x \cdot \left(\begin{array}{c} y \text{ is } x \text{ with } f \to v \\ \wedge\ z \text{ is } x \text{ with } g \to w \end{array} \right)$$

- Contains in fact two pieces of information:

Er.. Is That Really What We Want?

- Asymmetric:

$$y \text{ is } x \text{ with } f \rightarrow v$$

- Makes it hard to eliminate variables:

$$\exists x \cdot \left(\begin{array}{c} y \text{ is } x \text{ with } f \rightarrow v \\ \wedge\ z \text{ is } x \text{ with } g \rightarrow w \end{array} \right)$$

- Contains in fact two pieces of information:
  - "y and x may be different in f but are identical everywhere else"

Er.. Is That Really What We Want?

- Asymmetric:

$$y \text{ is } x \text{ with } f \to v$$

- Makes it hard to eliminate variables:

$$\exists x \cdot \left(\begin{array}{c} y \text{ is } x \text{ with } f \to v \\ \land\ z \text{ is } x \text{ with } g \to w \end{array} \right)$$

- Contains in fact two pieces of information:
  - "y and x may be different in f but are identical everywhere else"
  - "y points to v through f"

Er.. Is That Really What We Want?

- Asymmetric:

$$y \text{ is } x \text{ with } f \to v$$

- Makes it hard to eliminate variables:

$$\exists x \cdot \left(\begin{array}{c} y \text{ is } x \text{ with } f \to v \\ \wedge\ z \text{ is } x \text{ with } g \to w \end{array} \right)$$

- Contains in fact two pieces of information:
  - "y and x may be different in f but are identical everywhere else"
  - "y points to v through f":

$$y[f]v$$

Er.. Is That Really What We Want?

- Asymmetric:

$$y \text{ is } x \text{ with } f \to v$$

- Makes it hard to eliminate variables:

$$\exists x \cdot \left(\begin{array}{c} y \text{ is } x \text{ with } f \to v \\ \wedge\ z \text{ is } x \text{ with } g \to w \end{array} \right)$$

- Contains in fact two pieces of information:
  - "y and x may be different in f but are identical everywhere else":

$$y \sim_f x$$

  - "y points to v through f":

$$y[f]v$$

Nah. This Tildy-Thingy Looks Much Better

- Allows to express the update:

$$\text{“}y \text{ is } x \text{ with } f \to v\text{”} \quad := \quad y \sim_f x \land y[f]v$$

Nah. This Tildy-Thingy Looks Much Better

- Allows to express the update:

$$\text{“}y\text{ is }x\text{ with }f \to v\text{”} \quad := \quad y \sim_f x \wedge y[f]v$$

- Equivalence relation:

$$y \sim_f x \quad \Longleftrightarrow \quad x \sim_f y$$
$$y \sim_f x \wedge x \sim_f z \quad \Longrightarrow \quad y \sim_f z$$

Nah. This Tildy-Thingy Looks Much Better

- Allows to express the update:

$$\text{“}y \text{ is } x \text{ with } f \rightarrow v\text{”} \quad := \quad y \sim_f x \wedge y[f]v$$

- Equivalence relation:

$$y \sim_f x \quad \Longleftrightarrow \quad x \sim_f y$$
$$y \sim_f x \wedge x \sim_f z \quad \Longrightarrow \quad y \sim_f z$$

- Other properties:

$$y \sim_f x \wedge x \sim_g z \quad \Longrightarrow \quad y \sim_{\{f,g\}} z$$
$$y \sim_f x \wedge y \sim_g x \quad \Longleftrightarrow \quad y \sim_{\varnothing} x$$

Nah. This Tildy-Thingy Looks Much Better

- Allows to express the update:

$$\text{“}y \text{ is } x \text{ with } f \to v\text{”} \quad := \quad y \sim_f x \wedge y[f]v$$

- Equivalence relation:

$$\begin{aligned} y \sim_f x \quad &\iff \quad x \sim_f y \\ y \sim_f x \wedge x \sim_f z \quad &\implies \quad y \sim_f z \end{aligned}$$

- Other properties:

$$\begin{aligned} y \sim_f x \wedge x \sim_g z \quad &\implies \quad y \sim_{\{f,g\}} z \\ y \sim_f x \wedge y \sim_g x \quad &\iff \quad y \sim_\varnothing x \end{aligned}$$

- Allows to remove variables:

$$\exists x \cdot \left(\begin{array}{c} y \text{ is } x \text{ with } f \to v \\ \wedge\ z \text{ is } x \text{ with } g \to w \end{array} \right)$$

Nah. This Tildy-Thingy Looks Much Better

- Allows to express the update:

$$\text{“}y \text{ is } x \text{ with } f \to v\text{”} \quad := \quad y \sim_f x \wedge y[f]v$$

- Equivalence relation:

$$\begin{aligned} y \sim_f x \quad &\Longleftrightarrow \quad x \sim_f y \\ y \sim_f x \wedge x \sim_f z \quad &\Longrightarrow \quad y \sim_f z \end{aligned}$$

- Other properties:

$$\begin{aligned} y \sim_f x \wedge x \sim_g z \quad &\Longrightarrow \quad y \sim_{\{f,g\}} z \\ y \sim_f x \wedge y \sim_g x \quad &\Longleftrightarrow \quad y \sim_\varnothing x \end{aligned}$$

- Allows to remove variables:

$$\exists x \cdot \begin{pmatrix} y \sim_f x \wedge y[f]v \\ \wedge\, z \sim_g x \wedge z[g]w \end{pmatrix}$$

Nah. This Tildy-Thingy Looks Much Better

- Allows to express the update:

$$\text{“}y \text{ is } x \text{ with } f \to v\text{”} \quad := \quad y \sim_f x \land y[f]v$$

- Equivalence relation:

$$\begin{aligned} y \sim_f x \quad &\Longleftrightarrow \quad x \sim_f y \\ y \sim_f x \land x \sim_f z \quad &\Longrightarrow \quad y \sim_f z \end{aligned}$$

- Other properties:

$$\begin{aligned} y \sim_f x \land x \sim_g z \quad &\Longrightarrow \quad y \sim_{\{f,g\}} z \\ y \sim_f x \land y \sim_g x \quad &\Longleftrightarrow \quad y \sim_\varnothing x \end{aligned}$$

- Allows to remove variables:

$$\exists x \cdot \begin{pmatrix} y \sim_f x \land y[f]v \\ \land\, z \sim_g x \land z[g]w \end{pmatrix} \leftrightarrow y[f]v \land z[g]w$$

Nah. This Tildy-Thingy Looks Much Better

- Allows to express the update:

$$\text{“}y \text{ is } x \text{ with } f \rightarrow v\text{”} \quad := \quad y \sim_f x \wedge y[f]v$$

- Equivalence relation:

$$\begin{array}{rcl} y \sim_f x & \iff & x \sim_f y \\ y \sim_f x \wedge x \sim_f z & \implies & y \sim_f z \end{array}$$

- Other properties:

$$\begin{array}{rcl} y \sim_f x \wedge x \sim_g z & \implies & y \sim_{\{f,g\}} z \\ y \sim_f x \wedge y \sim_g x & \iff & y \sim_\varnothing x \end{array}$$

- Allows to remove variables:

$$\exists x \cdot \left(\begin{array}{c} y \sim_f x \wedge y[f]v \\ \wedge\, z \sim_g x \wedge z[g]w \end{array} \right) \leftrightarrow y[f]v \wedge z[g]w \wedge y \sim_{\{f,g\}} z$$

Model and Examples

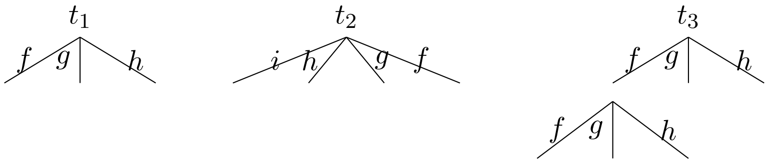
$$\mathcal{FT} = \mathcal{F} \rightsquigarrow \mathcal{FT}$$

- $\mathcal{F}$ infinite set of features (names for the edges);
- $\mathcal{F} \rightsquigarrow \mathcal{FT}$: partial function with finite domain;

Model and Examples

$$\mathcal{FT} = \mathcal{F} \rightsquigarrow \mathcal{FT}$$

- $\mathcal{F}$ infinite set of features (names for the edges);
- $\mathcal{F} \rightsquigarrow \mathcal{FT}$: partial function with finite domain;



Constraints and their Interpretation

Equality	$x \doteq y$
Feature	$x[f]y$
Absence	$x[f]\uparrow$
Fence	$x[F]$
Similarity	$x \sim_F y$

- x, y variables.
- $f \in \mathcal{F}$, $F \subset \mathcal{F}$ finite.

Constraints and their Interpretation

Equality	$\mathcal{FT}, \rho$	$\models$	$x \doteq y$	iff
Feature	$\mathcal{FT}, \rho$	$\models$	$x[f]y$	iff
Absence	$\mathcal{FT}, \rho$	$\models$	$x[f]\uparrow$	iff
Fence	$\mathcal{FT}, \rho$	$\models$	$x[F]$	iff
Similarity	$\mathcal{FT}, \rho$	$\models$	$x \sim_F y$	iff

- x, y variables.
- $f \in \mathcal{F}$, $F \subset \mathcal{F}$ finite.
- ρ a valuation from variables to $\mathcal{FT}$.

Constraints and their Interpretation

Equality	$\mathcal{FT}, \rho$	$\models$	$x \doteq y$	iff	$\rho(x) = \rho(y)$
Feature	$\mathcal{FT}, \rho$	$\models$	$x[f]y$	iff	
Absence	$\mathcal{FT}, \rho$	$\models$	$x[f]\uparrow$	iff	
Fence	$\mathcal{FT}, \rho$	$\models$	$x[F]$	iff	
Similarity	$\mathcal{FT}, \rho$	$\models$	$x \sim_F y$	iff	

- x, y variables.
- $f \in \mathcal{F}$, $F \subset \mathcal{F}$ finite.
- ρ a valuation from variables to $\mathcal{FT}$.

Constraints and their Interpretation

Equality	$\mathcal{FT}, \rho$	$\models$	$x \doteq y$	iff	$\rho(x) = \rho(y)$
Feature	$\mathcal{FT}, \rho$	$\models$	$x[f]y$	iff	$\rho(x)(f) = \rho(y)$
Absence	$\mathcal{FT}, \rho$	$\models$	$x[f]\uparrow$	iff	
Fence	$\mathcal{FT}, \rho$	$\models$	$x[F]$	iff	
Similarity	$\mathcal{FT}, \rho$	$\models$	$x \sim_F y$	iff	

- x, y variables.
- $f \in \mathcal{F}$, $F \subset \mathcal{F}$ finite.
- ρ a valuation from variables to $\mathcal{FT}$.

Constraints and their Interpretation

Equality	$\mathcal{FT}, \rho$	$\models$	$x \doteq y$	iff	$\rho(x) = \rho(y)$
Feature	$\mathcal{FT}, \rho$	$\models$	$x[f]y$	iff	$\rho(x)(f) = \rho(y)$
Absence	$\mathcal{FT}, \rho$	$\models$	$x[f]\uparrow$	iff	$f \notin \mathtt{dom}(\rho(x))$
Fence	$\mathcal{FT}, \rho$	$\models$	$x[F]$	iff	
Similarity	$\mathcal{FT}, \rho$	$\models$	$x \sim_F y$	iff	

- x, y variables.
- $f \in \mathcal{F}$, $F \subset \mathcal{F}$ finite.
- ρ a valuation from variables to $\mathcal{FT}$.

Constraints and their Interpretation

Equality	$\mathcal{FT}, \rho$	$\models$	$x \doteq y$	iff	$\rho(x) = \rho(y)$
Feature	$\mathcal{FT}, \rho$	$\models$	$x[f]y$	iff	$\rho(x)(f) = \rho(y)$
Absence	$\mathcal{FT}, \rho$	$\models$	$x[f]\uparrow$	iff	$f \notin \mathtt{dom}(\rho(x))$
Fence	$\mathcal{FT}, \rho$	$\models$	$x[F]$	iff	$\mathtt{dom}(\rho(x)) \subseteq F$
Similarity	$\mathcal{FT}, \rho$	$\models$	$x \sim_F y$	iff	

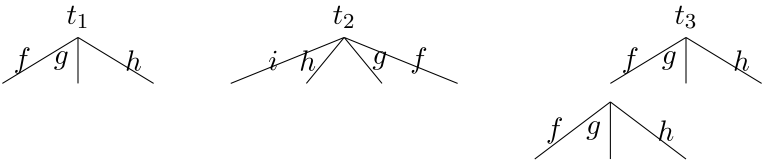
- x, y variables.
- $f \in \mathcal{F}$, $F \subset \mathcal{F}$ finite.
- ρ a valuation from variables to $\mathcal{FT}$.

Constraints and their Interpretation

Equality	$\mathcal{FT}, \rho$	$\models$	$x \doteq y$	iff	$\rho(x) = \rho(y)$
Feature	$\mathcal{FT}, \rho$	$\models$	$x[f]y$	iff	$\rho(x)(f) = \rho(y)$
Absence	$\mathcal{FT}, \rho$	$\models$	$x[f]\uparrow$	iff	$f \notin \mathtt{dom}(\rho(x))$
Fence	$\mathcal{FT}, \rho$	$\models$	$x[F]$	iff	$\mathtt{dom}(\rho(x)) \subseteq F$
Similarity	$\mathcal{FT}, \rho$	$\models$	$x \sim_F y$	iff	$\rho(x) \upharpoonright \overline{F} \;=\; \rho(y) \upharpoonright \overline{F}$

- x, y variables.
- $f \in \mathcal{F}$, $F \subset \mathcal{F}$ finite.
- ρ a valuation from variables to $\mathcal{FT}$.

Examples (Again)



The following constraints are satisfied in $\mathcal{FT}, [x \to t_1, y \to t_2, z \to t_3]$:

$$z[f]x, \quad x[i]\uparrow, \quad x[\{f,g,h,i\}], \quad x \sim_{\{i\}} y, \quad x \sim_{\{h,i\}} y$$

Existential Fragment

Existential Fragment

- Constraint system for symbolic execution.

Existential Fragment

- Constraint system for symbolic execution.
- Existential quantification on the outside.

Existential Fragment

- Constraint system for symbolic execution.
- Existential quantification on the outside.
- "Saturation" system:

Existential Fragment

- Constraint system for symbolic execution.
- Existential quantification on the outside.
- "Saturation" system:
  - that terminates,

Existential Fragment

- Constraint system for symbolic execution.
- Existential quantification on the outside.
- "Saturation" system:
  - that terminates,
  - that keeps equivalences,

Existential Fragment

- Constraint system for symbolic execution.
- Existential quantification on the outside.
- “Saturation” system:
  - that terminates,
  - that keeps equivalences,
  - with nice properties on the normal form.

Existential Fragment

- Constraint system for symbolic execution.
- Existential quantification on the outside.
- “Saturation” system:
  - that terminates,
  - that keeps equivalences,
  - with nice properties on the normal form.
- Normal form: incremental.

Existential Fragment

- Constraint system for symbolic execution.
- Existential quantification on the outside.
- “Saturation” system:
  - that terminates,
  - that keeps equivalences,
  - with nice properties on the normal form.
- Normal form: incremental.
- The rules come from properties of the constructions.

Rules with the Feature Constraint

Clash Rules

C-FEAT-ABS $x[f]y \wedge x[f]\uparrow$

C-FEAT-FEN $x[f]y \wedge x[F]$ $(f \notin F)$

Rules with the Feature Constraint

Clash Rules

C-FEAT-ABS $\quad x[f]y \wedge x[f]\uparrow$

C-FEAT-FEN $\quad x[f]y \wedge x[F] \qquad (f \notin F)$

Simplification Rules

S-FEATS
$$\exists X, z \cdot (x[f]y \wedge x[f]z \wedge c)$$
$$\Rightarrow \exists X \cdot (x[f]y \wedge c\{z \mapsto y\})$$

Rules with the Similarity Constraint

Propagation Rules

P-FEAT
$$x \sim_F y \wedge x[f]z \wedge c$$
$$\Rightarrow x \sim_F y \wedge x[f]z \wedge y[f]z \wedge c \qquad (f \notin F)$$

Rules with the Similarity Constraint

Propagation Rules

P-FEAT
$$x \sim_F y \wedge x[f]z \wedge c$$
$$\Rightarrow x \sim_F y \wedge x[f]z \wedge y[f]z \wedge c \qquad (f \notin F)$$

P-FEN
$$x \sim_F y \wedge x[G] \wedge c$$
$$\Rightarrow x \sim_F y \wedge x[G] \wedge y[F \cup G] \wedge c$$

Rules with the Similarity Constraint

Propagation Rules

P-FEAT
$$x \sim_F y \wedge x[f]z \wedge c \\ \Rightarrow x \sim_F y \wedge x[f]z \wedge y[f]z \wedge c \qquad (f \notin F)$$

P-FEN
$$x \sim_F y \wedge x[G] \wedge c \\ \Rightarrow x \sim_F y \wedge x[G] \wedge y[F \cup G] \wedge c$$

P-SIM
$$x \sim_F y \wedge x \sim_G z \wedge c \\ \Rightarrow x \sim_F y \wedge x \sim_G z \wedge y \sim_{F \cup G} z \wedge c$$

Properties of the Normal Forms

Lemma

Take a clause c ($\neq \bot$) [...]

$$c = g \land \exists X \cdot l$$

- *in normal form;*

Properties of the Normal Forms

Lemma

Take a clause c *(*$\neq \bot$*) [...]*

$$c = g \wedge \exists X \cdot l$$

- *in normal form;*
- *such that there is no* $y[f]x$ *with* $x \in X$ *and* $y \notin X$*.*

Properties of the Normal Forms

Lemma

Take a clause c ($\neq \bot$) [...]

$$c = g \wedge \exists X \cdot l$$

- *in normal form;*
- *such that there is no $y[f]x$ with $x \in X$ and $y \notin X$.*

Then

$$\mathcal{FT} \models \tilde{\forall} \cdot c \leftrightarrow g$$

Properties of the Normal Forms

Lemma

Take a clause c *(*$\neq \bot$*) [...]*

$$c = g \wedge \exists X \cdot l$$

- *in normal form;*
- *such that there is no* $y[f]x$ *with* $x \in X$ *and* $y \notin X$*.*

Then

$$\mathcal{FT} \models \tilde{\forall} \cdot c \leftrightarrow g$$

- Corollary: all normal forms ($\neq \bot$) are satisfiable:
  - If c is a clause in normal form: $\mathcal{FT} \models \tilde{\exists} \cdot c$

Properties of the Normal Forms

Lemma

Take a clause c ($\neq \bot$) [...]

$$c = g \wedge \exists X \cdot l$$

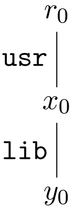
- *in normal form;*
- *such that there is no $y[f]x$ with $x \in X$ and $y \notin X$.*

Then

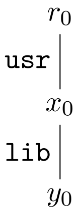
$$\mathcal{FT} \models \tilde{\forall} \cdot c \leftrightarrow g$$

- Corollary: all normal forms ($\neq \bot$) are satisfiable:
  - If c is a clause in normal form: $\mathcal{FT} \models \tilde{\exists} \cdot c$
- We can “garbage collect” the normal forms to make them smaller.

Garbage Collection

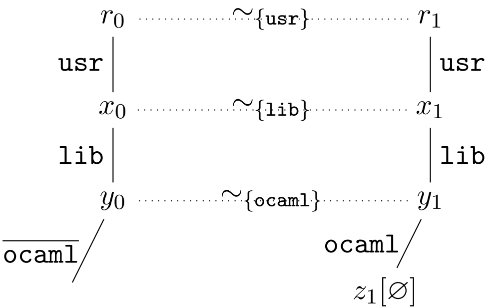


Garbage Collection



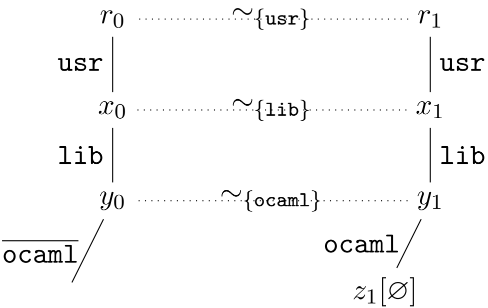
▷ `mkdir /usr/lib/ocaml;`

Garbage Collection



- ▷ `mkdir /usr/lib/ocaml;`
- ▷ Normal form: satisfiable

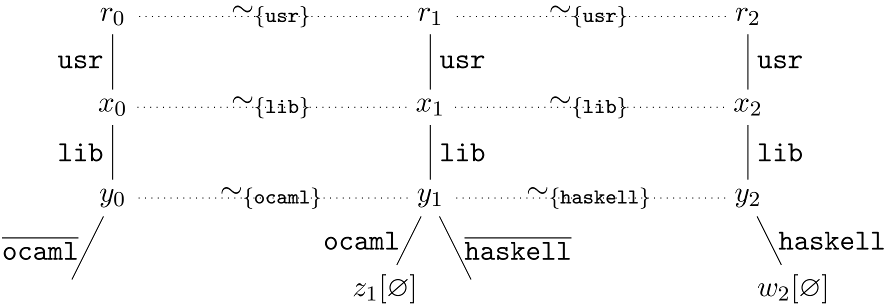
Garbage Collection



- `mkdir /usr/lib/ocaml;`
- `mkdir /usr/lib/haskell;`

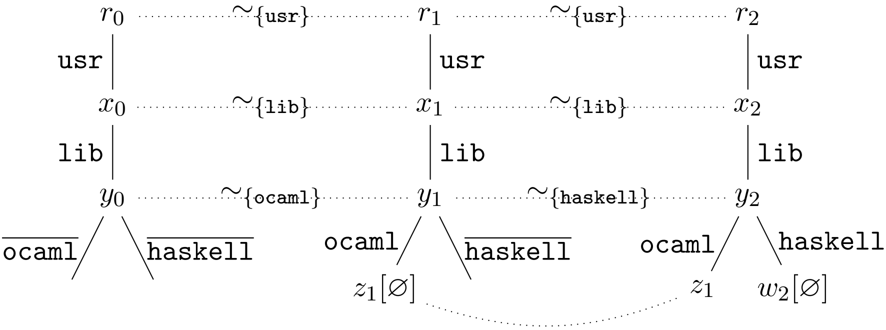
- Normal form: satisfiable

Garbage Collection



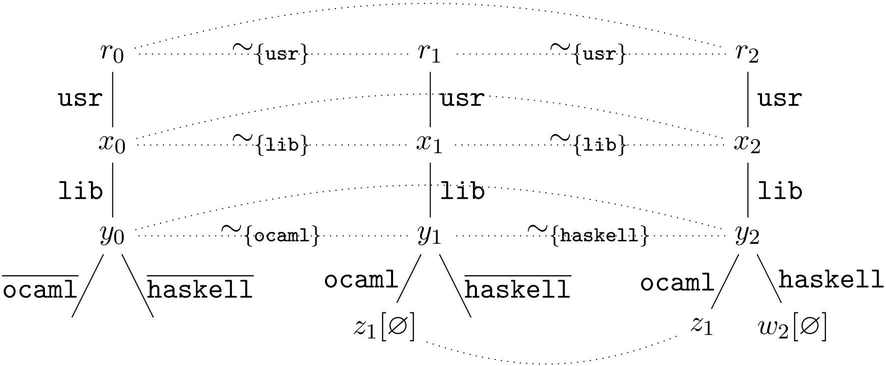
- ▷ `mkdir /usr/lib/ocaml;`
- ▷ `mkdir /usr/lib/haskell;`

Garbage Collection



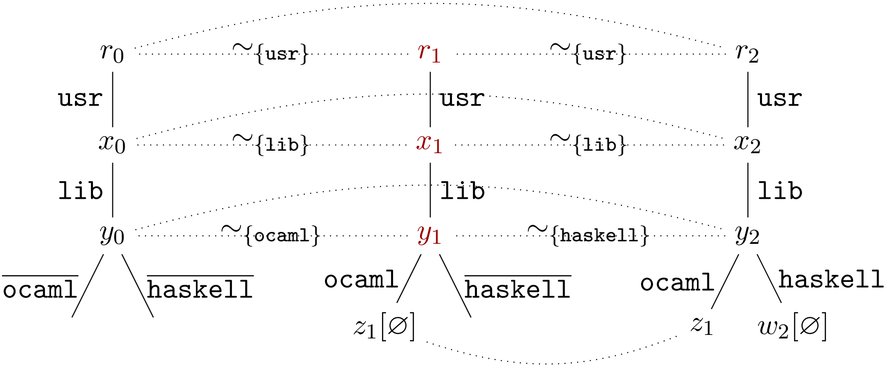
- ▷ `mkdir /usr/lib/ocaml;`
- ▷ `mkdir /usr/lib/haskell;`

Garbage Collection



- `mkdir /usr/lib/ocaml;`
- `mkdir /usr/lib/haskell;`
- Normal form: satisfiable

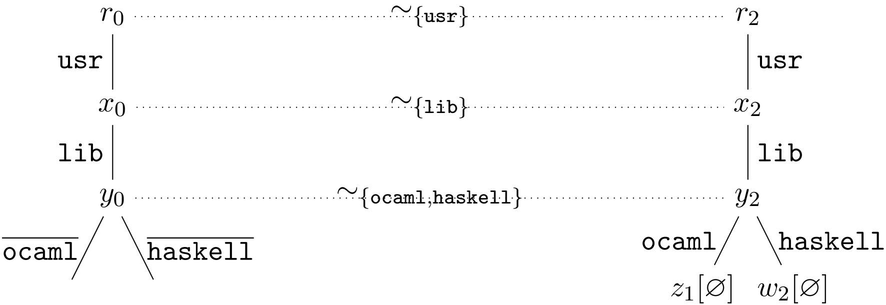
Garbage Collection



- `mkdir /usr/lib/ocaml;`
- `mkdir /usr/lib/haskell;`

- Normal form: satisfiable

Garbage Collection



- `mkdir /usr/lib/ocaml;`
- `mkdir /usr/lib/haskell;`

- Normal form: satisfiable

First Order

Quantifier Switching

- What can we express with local variables?

$$\exists x \cdot (y[f]x \wedge x[g] \uparrow)$$

Quantifier Switching

- What can we express with local variables?

$$\exists x \cdot (y[f]x \wedge x[g] \uparrow)$$

- Usually: add predicates to the language that cover these cases
  - Here: predicates about paths (hard to work with).

Quantifier Switching

- What can we express with local variables?

$$\exists x \cdot (y[f]x \wedge x[g] \uparrow)$$

- Usually: add predicates to the language that cover these cases
  - Here: predicates about paths (hard to work with).

- The feature constraint is a function:

$$\textsc{Feat-Fun} \qquad \begin{array}{ll} \exists X, x \cdot (y[f]x \wedge c) & y \notin X \\ \Rightarrow \neg y[f] \uparrow \wedge \forall x \cdot (y[f]x \rightarrow \exists X \cdot (y[f]x \wedge c)) & y \neq x \end{array}$$

Quantifier Switching

- What can we express with local variables?

$$\exists x \cdot (y[f]x \wedge x[g] \uparrow)$$

- Usually: add predicates to the language that cover these cases
  - Here: predicates about paths (hard to work with).

- The feature constraint is a function:

FEAT-FUN

$$\begin{array}{ll} \exists X, x \cdot (y[f]x \wedge c) & y \notin X \\ \Rightarrow \neg y[f] \uparrow \wedge \forall x \cdot (y[f]x \rightarrow \exists X \cdot (y[f]x \wedge c)) & y \neq x \end{array}$$

- In the example:

$$\neg y[f] \uparrow \wedge \forall x \cdot (y[f]x \rightarrow x[g] \uparrow)$$

How Does That Help?

FEAT-FUN

$$\exists X, x \cdot (y[f]x \wedge c) \qquad y \notin X$$
$$\Rightarrow \neg y[f] \uparrow \wedge \forall x \cdot (y[f]x \rightarrow \exists X \cdot (y[f]x \wedge c)) \qquad y \neq x$$

How Does That Help?

FEAT-FUN

$$\exists X, x \cdot (y[f]x \wedge c) \qquad y \notin X$$
$$\Rightarrow \neg y[f] \uparrow \wedge \forall x \cdot (y[f]x \rightarrow \exists X \cdot (y[f]x \wedge c)) \qquad y \neq x$$

Lemma (reminder)

Take a clause c ($\neq \bot$) [...]

$$c = g \wedge \exists X \cdot l$$

- *in normal form;*
- *such that there is no $y[f]x$ with $x \in X$ and $y \notin X$.*

Then

$$\mathcal{FT} \models \tilde{\forall} \cdot c \leftrightarrow g$$

How Does That Help?

FEAT-FUN	$\exists X, x \cdot (y[f]x \wedge c)$	$y \notin X$
	$\Rightarrow \neg y[f] \uparrow \wedge \forall x \cdot (y[f]x \rightarrow \exists X \cdot (y[f]x \wedge c))$	$y \neq x$

Lemma (reminder)

Take a clause c ($\neq \bot$) [...]

$$c = g \wedge \exists X \cdot l$$

- *in normal form;*
- *such that there is no $y[f]x$ with $x \in X$ and $y \notin X$.*

Then

$$\mathcal{FT} \models \tilde{\forall} \cdot c \leftrightarrow g$$

- FEAT-FUN puts us in the hypothesis of the lemma.

How Does That Help?

FEAT-FUN
$$\begin{array}{ll} \exists X, x \cdot (y[f]x \wedge c) & y \notin X \\ \Rightarrow \neg y[f] \uparrow \wedge \forall x \cdot (y[f]x \rightarrow \exists X \cdot (y[f]x \wedge c)) & y \neq x \end{array}$$

Lemma (reminder)

Take a clause c *(*$\neq \bot$*) [...]*

$$c = g \wedge \exists X \cdot l$$

- *in normal form;*
- *such that there is no* $y[f]x$ *with* $x \in X$ *and* $y \notin X$*.*

Then

$$\mathcal{FT} \models \tilde{\forall} \cdot c \leftrightarrow g$$

- FEAT-FUN puts us in the hypothesis of the lemma.
- Switch an existential quantification into an universal one.

How Does That Help?

FEAT-FUN

$$\exists X, x \cdot (y[f]x \wedge c) \qquad y \notin X$$
$$\Rightarrow \neg y[f]\uparrow \wedge \forall x \cdot (y[f]x \rightarrow \exists X \cdot (y[f]x \wedge c)) \qquad y \neq x$$

Lemma (reminder)

Take a clause c *(*$\neq \bot$*) [...]*

$$c = g \wedge \exists X \cdot l$$

- *in normal form;*
- *such that there is no* $y[f]x$ *with* $x \in X$ *and* $y \notin X$*.*

Then

$$\mathcal{FT} \models \tilde{\forall} \cdot c \leftrightarrow g$$

- FEAT-FUN puts us in the hypothesis of the lemma.
- Switch an existential quantification into an universal one.
- We can go for a weak quantifier elimination.

Weak Quantifier Elimination

- If we have a procedure:

$$\exists X \cdot c \Rightarrow \forall Y \cdot c'$$

Weak Quantifier Elimination

- If we have a procedure:

$$\exists X \cdot c \Rightarrow \forall Y \cdot c'$$

- Then:

$$\forall X_1 \cdot \exists X_2 \cdots \forall X_{n-1} \cdot \exists X_n \cdot c$$

Weak Quantifier Elimination

- If we have a procedure:

$$\exists X \cdot c \Rightarrow \forall Y \cdot c'$$

- Then:

$$\begin{array}{ll} & \forall X_1 \cdot \exists X_2 \cdots \forall X_{n-1} \cdot \exists X_n \cdot c \\ \Rightarrow & \forall X_1 \cdot \exists X_2 \cdots \forall X_{n-1} \cdot \forall Y_n \cdot c' \end{array}$$

Weak Quantifier Elimination

- If we have a procedure:

$$\exists X \cdot c \Rightarrow \forall Y \cdot c'$$

- Then:

$$\begin{array}{ll} & \forall X_1 \cdot \exists X_2 \cdots \forall X_{n-1} \cdot \exists X_n \cdot c \\ \Rightarrow & \forall X_1 \cdot \exists X_2 \cdots \forall X_{n-1} \cdot \forall Y_n \cdot c' \\ = & \forall X_1 \cdot \exists X_2 \cdots \forall X_{n-1} Y_n \cdot c' \end{array}$$

Weak Quantifier Elimination

- If we have a procedure:

$$\exists X \cdot c \Rightarrow \forall Y \cdot c'$$

- Then:

$$\begin{array}{lll} & & \forall X_1 \cdot \exists X_2 \cdots \forall X_{n-1} \cdot \exists X_n \cdot c \\ \Rightarrow & & \forall X_1 \cdot \exists X_2 \cdots \forall X_{n-1} \cdot \forall Y_n \cdot c' \\ = & & \forall X_1 \cdot \exists X_2 \cdots \forall X_{n-1} Y_n \cdot c' \\ \Rightarrow & \neg & \exists X_1 \cdot \forall X_2 \cdots \exists X_{n-1} Y_n \cdot \neg c' \end{array}$$

Weak Quantifier Elimination

- If we have a procedure:

$$\exists X \cdot c \Rightarrow \forall Y \cdot c'$$

- Then:

$$\begin{array}{rcl} & & \forall X_1 \cdot \exists X_2 \cdots \forall X_{n-1} \cdot \exists X_n \cdot c \\ \Rightarrow & & \forall X_1 \cdot \exists X_2 \cdots \forall X_{n-1} \cdot \forall Y_n \cdot c' \\ = & & \forall X_1 \cdot \exists X_2 \cdots \forall X_{n-1} Y_n \cdot c' \\ \Rightarrow & \neg & \exists X_1 \cdot \forall X_2 \cdots \exists X_{n-1} Y_n \cdot \neg c' \\ & & \vdots \\ \Rightarrow & ? & \exists Y_1 \cdot c'' \end{array}$$

Weak Quantifier Elimination

- If we have a procedure:

$$\exists X \cdot c \Rightarrow \forall Y \cdot c'$$

- Then:

$$\begin{array}{rcl} & & \forall X_1 \cdot \exists X_2 \cdots \forall X_{n-1} \cdot \exists X_n \cdot c \\ \Rightarrow & & \forall X_1 \cdot \exists X_2 \cdots \forall X_{n-1} \cdot \forall Y_n \cdot c' \\ = & & \forall X_1 \cdot \exists X_2 \cdots \forall X_{n-1} Y_n \cdot c' \\ \Rightarrow & \neg & \exists X_1 \cdot \forall X_2 \cdots \exists X_{n-1} Y_n \cdot \neg c' \\ & & \vdots \\ \Rightarrow & ? & \exists Y_1 \cdot c'' \end{array}$$

- We can remove all quantifier blocks but one.
- If we know how to handle the last block, it's won.
  - in our case, we do for closed formula.

Conclusion

- CoLiS project: verifying Debian packages and their shell scripts.
- Feature trees with update to model modifications of filesystems.
- Incremental procedure to decide satisfiability of an existential fragment.
- Extends to first order via weak quantifier elimination.

- Article:

Nicolas Jeannerod, Ralf Treinen. *Deciding the First-Order Theory of an Algebra of Feature Trees with Updates*. IJCAR 2018

- Thank you for your attention! Any questions?