Feature constraints to modelise Unix filesystems

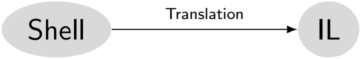
Nicolas Jeannerod

IRIF

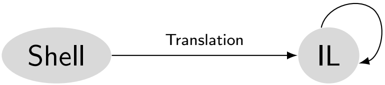
February 7, 2018

The CoLiS Project

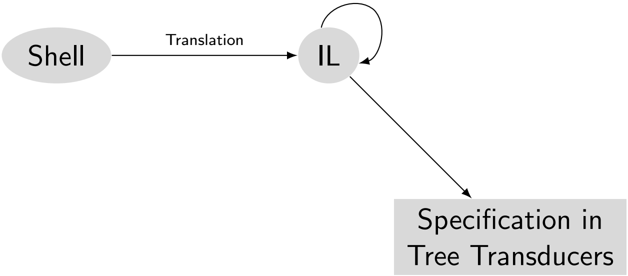
The CoLiS Project



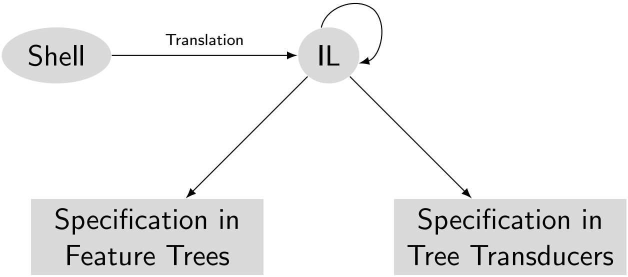
The CoLiS Project



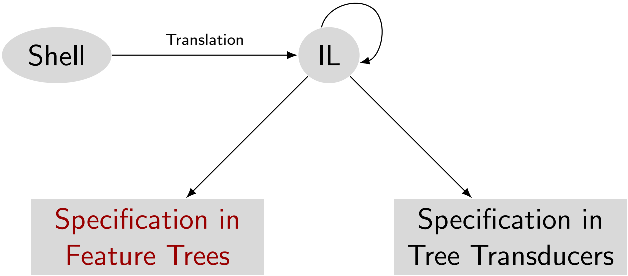
The CoLiS Project



The CoLiS Project



The CoLiS Project



Specifications.. then what?

Find accessible states that lead to errors.

Specifications.. then what?

Find accessible states that lead to errors.

- "Accessible"? Where the specification is satisfiable.

Specifications.. then what?

Find accessible states that lead to errors.

- "Accessible"? Where the specification is satisfiable.
- "Lead to errors"? Where the script exists abnormally.

Specifications.. then what?

Find accessible states that lead to errors.

- "Accessible"? Where the specification is satisfiable.
- "Lead to errors"? Where the script exists abnormally.

Fill automated report to script's maintainer.

Specifications.. then what?

Find accessible states that lead to errors.

- “Accessible”? Where the specification is satisfiable.
- “Lead to errors”? Where the script exists abnormally.

Fill automated report to script’s maintainer.

Check properties

Specifications.. then what?

Find accessible states that lead to errors.

- "Accessible"? Where the specification is satisfiable.
- "Lead to errors"? Where the script exists abnormally.

Fill automated report to script's maintainer.

Check properties:

- $\forall r_{in}, r_{out} \cdot \big(\mathtt{spec}_{s_1}(r_{in}, r_{out}) \;\leftrightarrow\; \mathtt{spec}_{s_2}(r_{out}, r_{in})\big)$

Specifications.. then what?

Find accessible states that lead to errors.

- "Accessible"? Where the specification is satisfiable.
- "Lead to errors"? Where the script exists abnormally.

Fill automated report to script's maintainer.

Check properties:

- $\forall r_{in}, r_{out} \cdot \big(\texttt{spec}_{s_1}(r_{in}, r_{out}) \ \leftrightarrow \ \texttt{spec}_{s_2}(r_{out}, r_{in})\big)$
- $\forall r_{in}, r_{out} \cdot (\texttt{spec}_s(r_{in}, r_{out}) \ \rightarrow \ r_{out}[\texttt{home}] = r_{in}[\texttt{home}])$

Specifications.. then what?

Find accessible states that lead to errors.

- "Accessible"? Where the specification is satisfiable.
- "Lead to errors"? Where the script exists abnormally.

Fill automated report to script's maintainer.

Check properties:

- $\forall r_{in}, r_{out} \cdot \big(\mathtt{spec}_{s_1}(r_{in}, r_{out}) \leftrightarrow \mathtt{spec}_{s_2}(r_{out}, r_{in})\big)$
- $\forall r_{in}, r_{out} \cdot (\mathtt{spec}_s(r_{in}, r_{out}) \rightarrow r_{out}[\mathtt{home}] = r_{in}[\mathtt{home}])$
- $\forall r_{in}, r_{out} \cdot (\mathtt{spec}_s(r_{in}, r_{out}) \leftrightarrow r_{out} \doteq r_{in})$

Specifications.. then what?

Find accessible states that lead to errors.

- "Accessible"? Where the specification is satisfiable.
- "Lead to errors"? Where the script exists abnormally.

Fill automated report to script's maintainer.

Check properties:

- $\forall r_{in}, r_{out} \cdot \big(\texttt{spec}_{s_1}(r_{in}, r_{out}) \;\leftrightarrow\; \texttt{spec}_{s_2}(r_{out}, r_{in})\big)$
- $\forall r_{in}, r_{out} \cdot (\texttt{spec}_{s}(r_{in}, r_{out}) \;\rightarrow\; r_{out}[\texttt{home}] = r_{in}[\texttt{home}])$
- $\forall r_{in}, r_{out} \cdot (\texttt{spec}_{s}(r_{in}, r_{out}) \;\leftrightarrow\; r_{out} \doteq r_{in})$
- $\forall r_{in}, r_{out} \cdot \big(\exists r' \cdot (\texttt{spec}_{s_1}(r_{in}, r') \wedge \texttt{spec}_{s_2}(r', r_{out})) \;\leftrightarrow\; r_{out} \doteq r_{in}\big)$

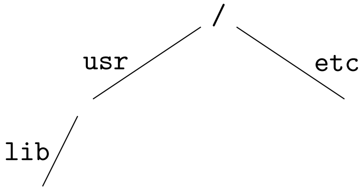
Table of Contents

1. Description of filesystems
   Unix filesystems
   Static description
   Directory update

2. Constraints
   Definitions
   Basic constraints
   Negation

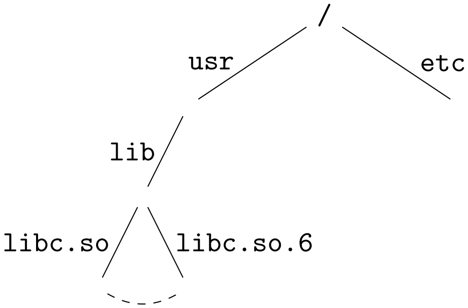
3. Usages
   Decidability of the First-Order Theory
   Automated Specification for Scripts: Proof of Concept

Unix filesystem



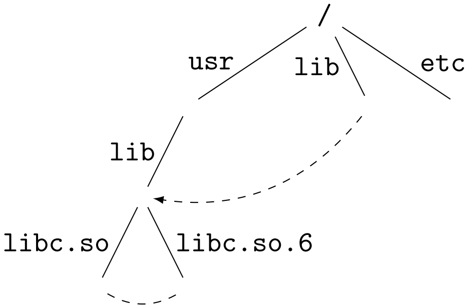
- Basically a tree with labelled nodes and edges;

Unix filesystem



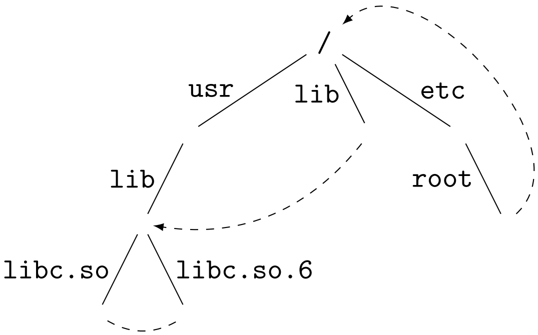
- Basically a tree with labelled nodes and edges;
- There can be sharing at the leafs (hard link between files);

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.

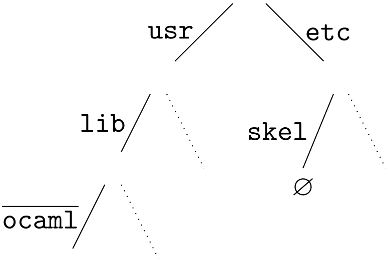
Table of Contents

1. Description of filesystems
   - Unix filesystems
   - Static description
   - Directory update

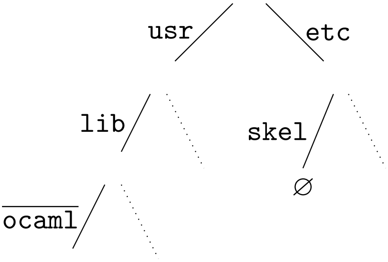
2. Constraints
   - Definitions
   - Basic constraints
   - Negation

3. Usages
   - Decidability of the First-Order Theory
   - Automated Specification for Scripts: Proof of Concept

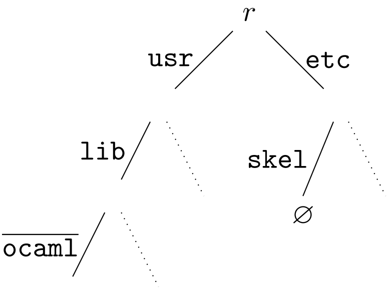
Static description



Static description

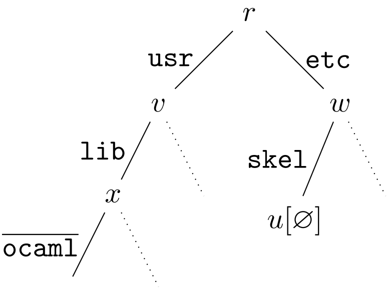


Static description



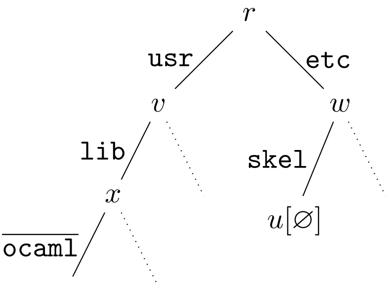
$c \;=$

Static description



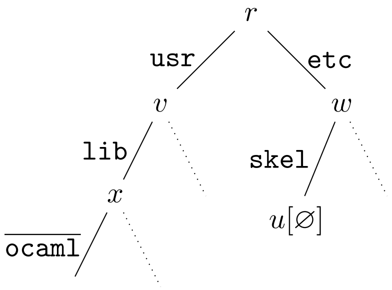
$$c \;=\; \exists u, v, x, w \cdot \left\{ \right.$$

Static description



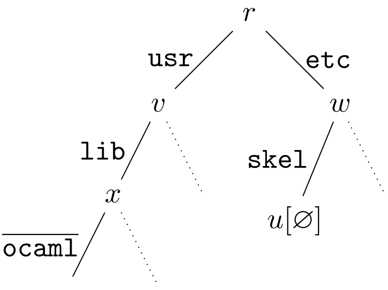
$$c \;=\; \exists u, v, x, w \cdot \begin{cases} r[\texttt{usr}]v \wedge v[\texttt{lib}]x \\ \wedge\, r[\texttt{etc}]w \wedge w[\texttt{skel}]u \end{cases}$$

Static description



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

Static description

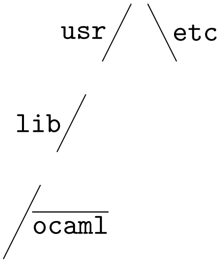


$$c \;=\; \exists u, v, x, w \cdot \left\{ \begin{array}{l} r[\texttt{usr}]v \wedge v[\texttt{lib}]x \wedge x[\texttt{ocaml}]\uparrow \\ \wedge\, r[\texttt{etc}]w \wedge w[\texttt{skel}]u \wedge u[\varnothing] \end{array} \right.$$

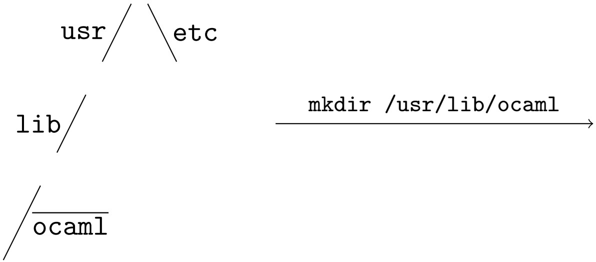
Table of Contents

1. Description of filesystems
   - Unix filesystems
   - Static description
   - Directory update
2. Constraints
   - Definitions
   - Basic constraints
   - Negation
3. Usages
   - Decidability of the First-Order Theory
   - Automated Specification for Scripts: Proof of Concept

Directory update

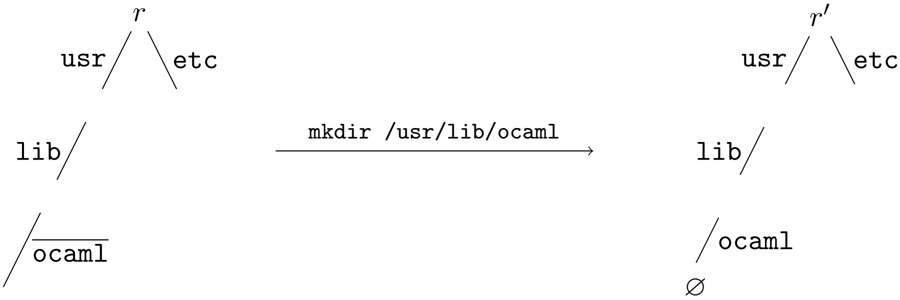


Directory update



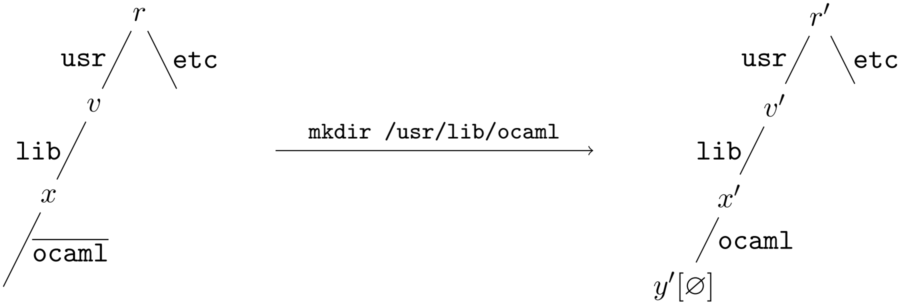
Directory update

Directory update



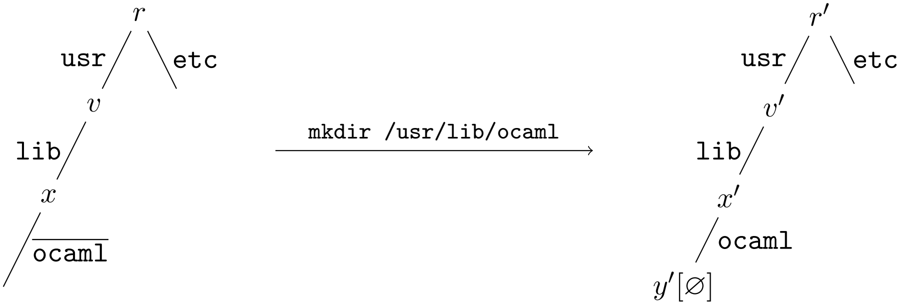
$c' \; =$

Directory update



$$c' \;=\; \exists v, v', x, x', y' \cdot \left\{ \right.$$

Directory update



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

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:

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

- 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"

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"
  - "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":

$$y \mathrel{\dot{\sim}_f} x$$

  - "y points to v through f":

$$y[f]v$$

$\sim$: Much better

- Allows to express the update:

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

$\sim$: Much better

- Allows to express the update:

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

- Symmetric and transitive:

$$y \mathrel{\dot{\sim}_f} x \quad \Longleftrightarrow \quad x \mathrel{\dot{\sim}_f} y$$
$$y \mathrel{\dot{\sim}_f} x \wedge z \mathrel{\dot{\sim}_f} x \quad \Longrightarrow \quad y \mathrel{\dot{\sim}_f} z$$

$\sim$: Much better

- Allows to express the update:

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

- Symmetric and transitive:

$$y \,\dot{\sim}_f\, x \quad \Longleftrightarrow \quad x \,\dot{\sim}_f\, y$$
$$y \,\dot{\sim}_f\, x \wedge z \,\dot{\sim}_f\, x \quad \Longrightarrow \quad y \,\dot{\sim}_f\, z$$

- Other properties:

$$y \,\dot{\sim}_f\, x \wedge z \,\dot{\sim}_g\, x \quad \Longrightarrow \quad y \,\dot{\sim}_{\{f,g\}}\, z$$
$$y \,\dot{\sim}_f\, x \wedge y \,\dot{\sim}_g\, x \quad \Longleftrightarrow \quad y \,\dot{\sim}_\varnothing\, x$$

$\sim$: Much better

- Allows to express the update:

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

- Symmetric and transitive:

$$y \mathrel{\dot{\sim}_f} x \iff x \mathrel{\dot{\sim}_f} y$$
$$y \mathrel{\dot{\sim}_f} x \wedge z \mathrel{\dot{\sim}_f} x \implies y \mathrel{\dot{\sim}_f} z$$

- Other properties:

$$y \mathrel{\dot{\sim}_f} x \wedge z \mathrel{\dot{\sim}_g} x \implies y \mathrel{\dot{\sim}_{\{f,g\}}} z$$
$$y \mathrel{\dot{\sim}_f} x \wedge y \mathrel{\dot{\sim}_g} x \iff y \mathrel{\dot{\sim}_\varnothing} x$$

- Allows to remove variables:

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

$\sim$: Much better

- Allows to express the update:

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

- Symmetric and transitive:

$$y \mathrel{\dot\sim_f} x \iff x \mathrel{\dot\sim_f} y$$
$$y \mathrel{\dot\sim_f} x \wedge z \mathrel{\dot\sim_f} x \implies y \mathrel{\dot\sim_f} z$$

- Other properties:

$$y \mathrel{\dot\sim_f} x \wedge z \mathrel{\dot\sim_g} x \implies y \mathrel{\dot\sim_{\{f,g\}}} z$$
$$y \mathrel{\dot\sim_f} x \wedge y \mathrel{\dot\sim_g} x \iff y \mathrel{\dot\sim_\varnothing} x$$

- Allows to remove variables:

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

$\sim$: Much better

- Allows to express the update:

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

- Symmetric and transitive:

$$y \mathrel{\dot{\sim}_f} x \quad \Longleftrightarrow \quad x \mathrel{\dot{\sim}_f} y$$
$$y \mathrel{\dot{\sim}_f} x \wedge z \mathrel{\dot{\sim}_f} x \quad \Longrightarrow \quad y \mathrel{\dot{\sim}_f} z$$

- Other properties:

$$y \mathrel{\dot{\sim}_f} x \wedge z \mathrel{\dot{\sim}_g} x \quad \Longrightarrow \quad y \mathrel{\dot{\sim}_{\{f,g\}}} z$$
$$y \mathrel{\dot{\sim}_f} x \wedge y \mathrel{\dot{\sim}_g} x \quad \Longleftrightarrow \quad y \mathrel{\dot{\sim}_{\varnothing}} x$$

- Allows to remove variables:

$$\exists x \cdot \begin{pmatrix} y \mathrel{\dot{\sim}_f} x \wedge y[f]v \\ \wedge\ z \mathrel{\dot{\sim}_g} x \wedge z[g]w \end{pmatrix} \leftrightarrow\ y[f]v \wedge z[g]w$$

$\sim$: Much better

- Allows to express the update:

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

- Symmetric and transitive:

$$\begin{aligned} y \mathrel{\dot{\sim}_f} x \quad &\Longleftrightarrow \quad x \mathrel{\dot{\sim}_f} y \\ y \mathrel{\dot{\sim}_f} x \land z \mathrel{\dot{\sim}_f} x \quad &\Longrightarrow \quad y \mathrel{\dot{\sim}_f} z \end{aligned}$$

- Other properties:

$$\begin{aligned} y \mathrel{\dot{\sim}_f} x \land z \mathrel{\dot{\sim}_g} x \quad &\Longrightarrow \quad y \mathrel{\dot{\sim}_{\{f,g\}}} z \\ y \mathrel{\dot{\sim}_f} x \land y \mathrel{\dot{\sim}_g} x \quad &\Longleftrightarrow \quad y \mathrel{\dot{\sim}_{\varnothing}} x \end{aligned}$$

- Allows to remove variables:

$$\exists x \cdot \begin{pmatrix} y \mathrel{\dot{\sim}_f} x \land y[f]v \\ \land \; z \mathrel{\dot{\sim}_g} x \land z[g]w \end{pmatrix} \leftrightarrow \; y[f]v \land z[g]w \land y \mathrel{\dot{\sim}_{\{f,g\}}} z$$

Table of Contents

1. Description of filesystems
   - Unix filesystems
   - Static description
   - Directory update
2. Constraints
   - Definitions
   - Basic constraints
   - Negation
3. Usages
   - Decidability of the First-Order Theory
   - Automated Specification for Scripts: Proof of Concept

Model and Constraints

$$\texttt{ftree} \quad ::= \quad \mathcal{F} \rightsquigarrow \texttt{ftree}$$

Model and Constraints

$$\texttt{ftree} \quad ::= \quad \mathcal{F} \rightsquigarrow \texttt{ftree}$$

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

Model and Constraints

$$\texttt{ftree} \quad ::= \quad \mathcal{F} \rightsquigarrow \texttt{ftree}$$

- $\mathcal{F}$ infinite set of features (names for the edges);
- $\mathcal{F} \rightsquigarrow \texttt{ftree}$: partial function with finite domain;
- Infinite set of variables x, y, etc.;
- $f \in \mathcal{F}$, $F \subset \mathcal{F}$ finite.

Equality	$x \doteq y$		
Feature	$x[f]y$	$x[f]\uparrow$	Absence
Fence	$x[F]$	$x \mathrel{\dot\sim}_F y$	Similarity

Model and Constraints

$$\texttt{ftree} \quad ::= \quad \mathcal{F} \rightsquigarrow \texttt{ftree}$$

- $\mathcal{F}$ infinite set of features (names for the edges);
- $\mathcal{F} \rightsquigarrow \texttt{ftree}$: partial function with finite domain;
- Infinite set of variables x, y, etc.;
- $f \in \mathcal{F}$, $F \subset \mathcal{F}$ finite.

Equality	$x \doteq y$		
Feature	$x[f]y$	$x[f]\uparrow$	Absence
Fence	$x[F]$	$x \mathrel{\dot{\sim}_F} y$	Similarity

- Composed with $\neg$, $\wedge$, $\vee$, $\exists x$, $\forall x$ (no quantification on features);

Model and Constraints

$$\texttt{ftree} \quad ::= \quad \mathcal{F} \rightsquigarrow \texttt{ftree}$$

- $\mathcal{F}$ infinite set of features (names for the edges);
- $\mathcal{F} \rightsquigarrow \texttt{ftree}$: partial function with finite domain;
- Infinite set of variables x, y, etc.;
- $f \in \mathcal{F}$, $F \subset \mathcal{F}$ finite.

Equality	$x \doteq y$		
Feature	$x[f]y$	$x[f]\uparrow$	Absence
Fence	$x[F]$	$x \mathrel{\dot\sim}_F y$	Similarity

- Composed with $\neg$, $\wedge$, $\vee$, $\exists x$, $\forall x$ (no quantification on features);
- Wanted: (un)satisfiability of these constraints;
- Bonus point for incremental procedures.

Semantics

$$\mathcal{T}, \rho \models c$$

- $\mathcal{T}$ the model of all feature trees;
- $\rho : \mathcal{V}(c) \to \mathcal{T}$;

Semantics

$$\mathcal{T}, \rho \models c$$

- $\mathcal{T}$ the model of all feature trees;
- $\rho : \mathcal{V}(c) \to \mathcal{T}$;

Equality: $\quad \mathcal{T}, \rho \quad \models \quad x \doteq y \qquad$ if $\quad \rho(x) = \rho(y)$

Semantics

$$\mathcal{T}, \rho \models c$$

- $\mathcal{T}$ the model of all feature trees;
- $\rho : \mathcal{V}(c) \to \mathcal{T}$;

Equality:	$\mathcal{T}, \rho$	$\models$	$x \doteq y$	if	$\rho(x) = \rho(y)$
Feature:	$\mathcal{T}, \rho$	$\models$	$x[f]y$	if	$\rho(x)(f) = \rho(y)$
Absence:	$\mathcal{T}, \rho$	$\models$	$x[f]\uparrow$	if	$f \notin \texttt{dom}(\rho(x))$

Semantics

$$\mathcal{T}, \rho \models c$$

- $\mathcal{T}$ the model of all feature trees;
- $\rho : \mathcal{V}(c) \to \mathcal{T}$;

Equality:	$\mathcal{T}, \rho$	$\models$	$x \doteq y$	if	$\rho(x) = \rho(y)$
Feature:	$\mathcal{T}, \rho$	$\models$	$x[f]y$	if	$\rho(x)(f) = \rho(y)$
Absence:	$\mathcal{T}, \rho$	$\models$	$x[f]\uparrow$	if	$f \notin \mathtt{dom}(\rho(x))$
Fence:	$\mathcal{T}, \rho$	$\models$	$x[F]$	if	$\mathtt{dom}(\rho(x)) \subseteq F$

Semantics

$$\mathcal{T}, \rho \models c$$

- $\mathcal{T}$ the model of all feature trees;
- $\rho : \mathcal{V}(c) \to \mathcal{T}$;

Equality:	$\mathcal{T}, \rho$	$\models$	$x \doteq y$	if	$\rho(x) = \rho(y)$
Feature:	$\mathcal{T}, \rho$	$\models$	$x[f]y$	if	$\rho(x)(f) = \rho(y)$
Absence:	$\mathcal{T}, \rho$	$\models$	$x[f]\uparrow$	if	$f \notin \mathtt{dom}(\rho(x))$
Fence:	$\mathcal{T}, \rho$	$\models$	$x[F]$	if	$\mathtt{dom}(\rho(x)) \subseteq F$
Similarity:	$\mathcal{T}, \rho$	$\models$	$x \mathrel{\dot{\sim}_F} y$	if	$\rho(x) \restriction \overline{F} = \rho(y) \restriction \overline{F}$

Table of Contents

1. Description of filesystems
   - Unix filesystems
   - Static description
   - Directory update

2. Constraints
   - Definitions
   - Basic constraints
   - Negation

3. Usages
   - Decidability of the First-Order Theory
   - Automated Specification for Scripts: Proof of Concept

Game plan

- Rewriting system;

Game plan

- Rewriting system;
- Puts constraints in normal form (not necessarily unique);

Game plan

- Rewriting system;
- Puts constraints in normal form (not necessarily unique);
- Respects equivalences;

Game plan

- Rewriting system;
- Puts constraints in normal form (not necessarily unique);
- Respects equivalences;
- Normal forms: either $\bot$ or with nice properties.

Basic rewriting system

$$
\begin{array}{ll}
x_1[f_1]x_2 \wedge \ldots \wedge x_n[f_n]x_1 & (n \geq 1) \\
x[f]y \wedge x[f]\uparrow & \\
x[f]y \wedge x[F] & (f \notin F)
\end{array}
$$

Clash Patterns

Basic rewriting system

$$
\begin{array}{ll}
x_1[f_1]x_2 \wedge \ldots \wedge x_n[f_n]x_1 & (n \geq 1) \\
x[f]y \wedge x[f]\uparrow & \\
x[f]y \wedge x[F] & (f \notin F)
\end{array}
$$

Clash Patterns

$$
\begin{array}{rcll}
\exists X, x \cdot (x \doteq y \wedge c) & \Rightarrow & \exists X \cdot c\{x \mapsto y\} & (x \neq y) \\
\exists X, z \cdot (x[f]y \wedge x[f]z \wedge c) & \Rightarrow & \exists X \cdot (x[f]y \wedge c\{z \mapsto y\}) & (y \neq z) \\
x \mathrel{\dot\sim}_F y \wedge x \mathrel{\dot\sim}_G y \wedge c & \Rightarrow & x \mathrel{\dot\sim}_{F \cap G} y \wedge c &
\end{array}
$$

Simplification Rules

Basic rewriting system

$$x_1[f_1]x_2 \wedge \ldots \wedge x_n[f_n]x_1 \qquad (n \geq 1)$$
$$x[f]y \wedge x[f]\uparrow$$
$$x[f]y \wedge x[F] \qquad (f \notin F)$$

Clash Patterns

$$\exists X, x \cdot (x \doteq y \wedge c) \quad \Rightarrow \quad \exists X \cdot c\{x \mapsto y\} \qquad (x \neq y)$$
$$\exists X, z \cdot (x[f]y \wedge x[f]z \wedge c) \quad \Rightarrow \quad \exists X \cdot (x[f]y \wedge c\{z \mapsto y\}) \qquad (y \neq z)$$
$$x \mathrel{\dot{\sim}_F} y \wedge x \mathrel{\dot{\sim}_G} y \wedge c \quad \Rightarrow \quad x \mathrel{\dot{\sim}_{F \cap G}} y \wedge c$$

Simplification Rules

$$x \mathrel{\dot{\sim}_F} y \wedge x[f]z \wedge c \quad \Rightarrow \quad x \mathrel{\dot{\sim}_F} y \wedge x[f]z \wedge y[f]z \wedge c \qquad (f \notin F)$$
$$x \mathrel{\dot{\sim}_F} y \wedge x[f]\uparrow \wedge c \quad \Rightarrow \quad x \mathrel{\dot{\sim}_F} y \wedge x[f]\uparrow \wedge y[f]\uparrow \wedge c \qquad (f \notin F)$$
$$x \mathrel{\dot{\sim}_F} y \wedge x[G] \wedge c \quad \Rightarrow \quad x \mathrel{\dot{\sim}_F} y \wedge x[G] \wedge y[F \cup G] \wedge c$$
$$x \mathrel{\dot{\sim}_F} y \wedge x \mathrel{\dot{\sim}_G} z \wedge c \quad \Rightarrow \quad x \mathrel{\dot{\sim}_F} y \wedge x \mathrel{\dot{\sim}_G} z \wedge y \mathrel{\dot{\sim}_{F \cup G}} z \wedge c$$
$$(\text{if } \textstyle\bigcap_{y \dot{\sim}_H z} H \not\subseteq F \cup G)$$

Propagation Rules

Properties

Lemma

The basic constraint system terminates and yields a clause that is equivalent to the first one.

Properties

Lemma

The basic constraint system terminates and yields a clause that is equivalent to the first one.

Lemma

Let c be a clause $c = g_c \wedge \exists X \cdot l_c$ such that

Properties

Lemma

The basic constraint system terminates and yields a clause that is equivalent to the first one.

Lemma

Let c be a clause $c = g_c \wedge \exists X \cdot l_c$ such that:

- *c is in normal form;*

Properties

Lemma

The basic constraint system terminates and yields a clause that is equivalent to the first one.

Lemma

Let c be a clause $c = g_c \wedge \exists X \cdot l_c$ such that:

- *c is in normal form;*
- *$\mathcal{V}(g_c) \cap X = \varnothing$;*
- *every literal in l_c is about X;*

Properties

Lemma

The basic constraint system terminates and yields a clause that is equivalent to the first one.

Lemma

Let c *be a clause* $c = g_c \wedge \exists X \cdot l_c$ *such that:*

- c *is in normal form;*
- $\mathcal{V}(g_c) \cap X = \varnothing$*;*
- *every literal in* l_c *is about* X*;*
- *there is no* $y[f]x$ *with* $x \in X$ *and* $y \notin X$*.*

Properties

Lemma

The basic constraint system terminates and yields a clause that is equivalent to the first one.

Lemma

Let c be a clause $c = g_c \wedge \exists X \cdot l_c$ such that:

- *c is in normal form;*
- *$\mathcal{V}(g_c) \cap X = \varnothing$;*
- *every literal in l_c is about X;*
- *there is no $y[f]x$ with $x \in X$ and $y \notin X$.*

Then c is equivalent to g_c.

Table of Contents

1. Description of filesystems
   - Unix filesystems
   - Static description
   - Directory update
2. Constraints
   - Definitions
   - Basic constraints
   - Negation
3. Usages
   - Decidability of the First-Order Theory
   - Automated Specification for Scripts: Proof of Concept

Negation: new players, new rules

aka *La Slide de la Mort*

Negation: new players, new rules

$$\begin{array}{rcl} \neg x[f]y \wedge c & \Rightarrow & (x[f]\uparrow \vee \exists z \cdot (x[f]z \wedge y \not\sim_{\varnothing} z)) \wedge c \\ \neg x[f]\uparrow \wedge c & \Rightarrow & \exists z \cdot x[f]z \wedge c \end{array}$$

Simple Replacement Rules

Negation: new players, new rules

$$\begin{aligned}
\neg x[f]y \wedge c &\Rightarrow (x[f]\uparrow \vee \exists z \cdot (x[f]z \wedge y \not\sim_{\varnothing} z)) \wedge c \\
\neg x[f]\uparrow \wedge c &\Rightarrow \exists z \cdot x[f]z \wedge c
\end{aligned}$$

Simple Replacement Rules

$$\begin{aligned}
x[F] \wedge \neg x[G] \wedge c &\Rightarrow x[F] \wedge x\langle F \setminus G\rangle \wedge c \\
x[F] \wedge x \not\sim_G y \wedge c &\Rightarrow x[F] \wedge \left(\neg y[F \cup G] \vee x \dot{\neq}_{F\setminus G} y\right) \wedge c \\
x \dot{\sim}_F y \wedge x \not\sim_G y \wedge c &\Rightarrow x \dot{\sim}_F y \wedge x \dot{\neq}_{F\setminus G} y \wedge c
\end{aligned}$$

More Replacement Rules

Negation: new players, new rules

$$\begin{array}{rcl} x[F] \wedge \neg x[G] \wedge c & \Rightarrow & x[F] \wedge x\langle F \setminus G\rangle \wedge c \\ x[F] \wedge x \not\sim_G y \wedge c & \Rightarrow & x[F] \wedge \big(\neg y[F \cup G] \vee x \dot{\neq}_{F\setminus G}\, y\big) \wedge c \\ x \dot{\sim}_F y \wedge x \not\sim_G y \wedge c & \Rightarrow & x \dot{\sim}_F y \wedge x \dot{\neq}_{F\setminus G}\, y \wedge c \end{array}$$

More Replacement Rules

Negation: new players, new rules

$$
\begin{array}{rcl}
x[F] \wedge \neg x[G] \wedge c & \Rightarrow & x[F] \wedge x\langle F \setminus G\rangle \wedge c \\
x[F] \wedge x \not\sim_G y \wedge c & \Rightarrow & x[F] \wedge \left(\neg y[F \cup G] \vee x \dot{\neq}_{F\setminus G}\, y\right) \wedge c \\
x \dot{\sim}_F\, y \wedge x \not\sim_G y \wedge c & \Rightarrow & x \dot{\sim}_F\, y \wedge x \dot{\neq}_{F\setminus G}\, y \wedge c
\end{array}
$$

More Replacement Rules

Negation: new players, new rules

$$x\langle F\rangle := \bigvee_{f\in F} \exists z \cdot x[f]z$$

$$\begin{array}{rcl} x[F] \wedge \neg x[G] \wedge c & \Rightarrow & x[F] \wedge x\langle F \setminus G\rangle \wedge c \\ x[F] \wedge x \not\sim_G y \wedge c & \Rightarrow & x[F] \wedge \left(\neg y[F \cup G] \vee x \dot{\neq}_{F\setminus G}\ y\right) \wedge c \\ x \dot{\sim}_F y \wedge x \not\sim_G y \wedge c & \Rightarrow & x \dot{\sim}_F y \wedge x \dot{\neq}_{F\setminus G}\ y \wedge c \end{array}$$

More Replacement Rules

Negation: new players, new rules

$$x \doteq\not_F y := \bigvee_{f \in F} \left(\begin{array}{c} \exists z' \cdot (x[f] \uparrow \wedge y[f]z') \vee \exists z \cdot (x[f]z \wedge y[f] \uparrow) \\ \vee \exists z, z' \cdot (x[f]z \wedge y[f]z' \wedge z \not\sim_\varnothing z') \end{array} \right)$$

$$\begin{array}{rcl} x[F] \wedge \neg x[G] \wedge c & \Rightarrow & x[F] \wedge x\langle F \setminus G \rangle \wedge c \\ x[F] \wedge x \not\sim_G y \wedge c & \Rightarrow & x[F] \wedge \left(\neg y[F \cup G] \vee x \doteq\not_{F \setminus G} y\right) \wedge c \\ x \dot\sim_F y \wedge x \not\sim_G y \wedge c & \Rightarrow & x \dot\sim_F y \wedge x \doteq\not_{F \setminus G} y \wedge c \end{array}$$

More Replacement Rules

Negation: new players, new rules

$$x \dot{\neq}_F\, y := \bigvee_{f \in F} \left(\begin{array}{c} \exists z' \cdot (x[f]\uparrow \wedge y[f]z') \vee \exists z \cdot (x[f]z \wedge y[f]\uparrow) \\ \vee \exists z, z' \cdot (x[f]z \wedge y[f]z' \wedge z \not\sim_{\varnothing} z') \end{array} \right)$$

$$\begin{array}{rcl} x[F] \wedge \neg x[G] \wedge c & \Rightarrow & x[F] \wedge x\langle F \setminus G \rangle \wedge c \\ x[F] \wedge x \not\sim_G y \wedge c & \Rightarrow & x[F] \wedge \big(\neg y[F \cup G] \vee x \dot{\neq}_{F \setminus G}\, y\big) \wedge c \\ x \dot{\sim}_F\, y \wedge x \not\sim_G y \wedge c & \Rightarrow & x \dot{\sim}_F\, y \wedge x \dot{\neq}_{F \setminus G}\, y \wedge c \end{array}$$

More Replacement Rules

$$\begin{array}{rcll} x \dot{\sim}_F\, y \wedge \neg x[G] \wedge c & \Rightarrow & x \dot{\sim}_F\, y \wedge (\neg x[F \cup G] \vee x\langle F \setminus G \rangle) \wedge c & (F \not\subseteq G \\ x \dot{\sim}_F\, y \wedge \neg x[G] \wedge c & \Rightarrow & x \dot{\sim}_F\, y \wedge \neg x[G] \wedge \neg y[G] \wedge c & (F \subseteq G \\ x \dot{\sim}_F\, y \wedge x \not\sim_G z \wedge c & \Rightarrow & x \dot{\sim}_F\, y \wedge \big(x \not\sim_{F \cup G} z \vee x \dot{\neq}_{F \setminus G}\, z\big) \wedge c & (F \not\subseteq G \\ x \dot{\sim}_F\, y \wedge x \not\sim_G z \wedge c & \Rightarrow & x \dot{\sim}_F\, y \wedge x \not\sim_G z \wedge y \not\sim_G z \wedge c & (F \subseteq G \end{array}$$

Enlargement and Propagation Rules

Negation: new players, new rules

$$x \dot{\neq}_F y := \bigvee_{f \in F} \left(\begin{array}{c} \exists z' \cdot (x[f] \uparrow \wedge y[f] z') \vee \exists z \cdot (x[f] z \wedge y[f] \uparrow) \\ \vee \exists z, z' \cdot (x[f] z \wedge y[f] z' \wedge z \not\sim_{\varnothing} z') \end{array} \right)$$

$$\begin{array}{rcl} x[F] \wedge \neg x[G] \wedge c & \Rightarrow & x[F] \wedge x\langle F \setminus G \rangle \wedge c \\ x[F] \wedge x \not\sim_G y \wedge c & \Rightarrow & x[F] \wedge \big(\neg y[F \cup G] \vee x \dot{\neq}_{F \setminus G} y \big) \wedge c \\ x \dot{\sim}_F y \wedge x \not\sim_G y \wedge c & \Rightarrow & x \dot{\sim}_F y \wedge x \dot{\neq}_{F \setminus G} y \wedge c \end{array}$$

More Replacement Rules

Negation: new players, new rules

$$x \dot{\neq}_F \; y := \bigvee_{f \in F} \left(\begin{array}{c} \exists z' \cdot (x[f] \uparrow \wedge y[f]z') \vee \exists z \cdot (x[f]z \wedge y[f] \uparrow) \\ \vee \exists z, z' \cdot (x[f]z \wedge y[f]z' \wedge z \not\sim_{\varnothing} z') \end{array} \right)$$

$$\begin{array}{rcl} x[F] \wedge \neg x[G] \wedge c & \Rightarrow & x[F] \wedge x\langle F \setminus G \rangle \wedge c \\ x[F] \wedge x \not\sim_G y \wedge c & \Rightarrow & x[F] \wedge \left(\neg y[F \cup G] \vee x \dot{\neq}_{F \setminus G} \; y\right) \wedge c \\ x \dot{\sim}_F \; y \wedge x \not\sim_G y \wedge c & \Rightarrow & x \dot{\sim}_F \; y \wedge x \dot{\neq}_{F \setminus G} \; y \wedge c \end{array}$$

More Replacement Rules

Negation: new players, new rules

$$x \dot{\neq}_F y := \bigvee_{f \in F} \left(\begin{array}{c} \exists z' \cdot (x[f] \uparrow \wedge y[f]z') \vee \exists z \cdot (x[f]z \wedge y[f] \uparrow) \\ \vee \exists z, z' \cdot (x[f]z \wedge y[f]z' \wedge z \not\sim_{\varnothing} z') \end{array} \right)$$

$$\begin{array}{rcl} x[F] \wedge \neg x[G] \wedge c & \Rightarrow & x[F] \wedge x\langle F \setminus G \rangle \wedge c \\ x[F] \wedge x \not\sim_G y \wedge c & \Rightarrow & x[F] \wedge \big(\neg y[F \cup G] \vee x \dot{\neq}_{F \setminus G} y\big) \wedge c \\ x \dot{\sim}_F y \wedge x \not\sim_G y \wedge c & \Rightarrow & x \dot{\sim}_F y \wedge x \dot{\neq}_{F \setminus G} y \wedge c \end{array}$$

More Replacement Rules

$x[F]$ = "x has no feature outside F"
$x \not\sim_G y$ = "there is a feature outside G that differentiates x and y"

Negation: new players, new rules

$$x \dot{\neq}_F y := \bigvee_{f \in F} \left(\begin{array}{c} \exists z' \cdot (x[f]\uparrow \wedge y[f]z') \vee \exists z \cdot (x[f]z \wedge y[f]\uparrow) \\ \vee \exists z, z' \cdot (x[f]z \wedge y[f]z' \wedge z \not\sim_{\varnothing} z') \end{array} \right)$$

$$\begin{array}{rcl} x[F] \wedge \neg x[G] \wedge c & \Rightarrow & x[F] \wedge x\langle F \setminus G\rangle \wedge c \\ x[F] \wedge x \not\sim_G y \wedge c & \Rightarrow & x[F] \wedge \big(\neg y[F \cup G] \vee x \dot{\neq}_{F \setminus G} y\big) \wedge c \\ x \dot{\sim}_F y \wedge x \not\sim_G y \wedge c & \Rightarrow & x \dot{\sim}_F y \wedge x \dot{\neq}_{F \setminus G} y \wedge c \end{array}$$

More Replacement Rules

$x[F]$ = "x has no feature outside F"

$x \not\sim_G y$ = "there is a feature outside G that differentiates x and y"

- either it is in F,
- or it is not,

Negation: new players, new rules

$$x \dot{\neq}_F y := \bigvee_{f \in F} \left(\begin{array}{c} \exists z' \cdot (x[f] \uparrow \wedge y[f] z') \vee \exists z \cdot (x[f] z \wedge y[f] \uparrow) \\ \vee \exists z, z' \cdot (x[f] z \wedge y[f] z' \wedge z \not\sim_{\varnothing} z') \end{array} \right)$$

$$\begin{array}{rcl} x[F] \wedge \neg x[G] \wedge c & \Rightarrow & x[F] \wedge x\langle F \setminus G \rangle \wedge c \\ x[F] \wedge x \not\sim_G y \wedge c & \Rightarrow & x[F] \wedge \left(\neg y[F \cup G] \vee x \dot{\neq}_{F \setminus G} y\right) \wedge c \\ x \dot{\sim}_F y \wedge x \not\sim_G y \wedge c & \Rightarrow & x \dot{\sim}_F y \wedge x \dot{\neq}_{F \setminus G} y \wedge c \end{array}$$

More Replacement Rules

$x[F] =$ "x has no feature outside F"

$x \not\sim_G y =$ "there is a feature outside G that differentiates x and y"

- either it is in F, and we can list all the cases;
- or it is not,

Negation: new players, new rules

$$x \dot{\neq}_F\ y := \bigvee_{f \in F} \left(\begin{array}{c} \exists z' \cdot (x[f] \uparrow \wedge y[f]z') \vee \exists z \cdot (x[f]z \wedge y[f] \uparrow) \\ \vee \exists z, z' \cdot (x[f]z \wedge y[f]z' \wedge z \not\sim_{\varnothing} z') \end{array} \right)$$

$$\begin{array}{rcl} x[F] \wedge \neg x[G] \wedge c & \Rightarrow & x[F] \wedge x\langle F \setminus G \rangle \wedge c \\ x[F] \wedge x \not\sim_G y \wedge c & \Rightarrow & x[F] \wedge \big(\neg y[F \cup G] \vee x \dot{\neq}_{F \setminus G}\ y\big) \wedge c \\ x \dot{\sim}_F\ y \wedge x \not\sim_G y \wedge c & \Rightarrow & x \dot{\sim}_F\ y \wedge x \dot{\neq}_{F \setminus G}\ y \wedge c \end{array}$$

More Replacement Rules

$x[F]$ = "x has no feature outside F"

$x \not\sim_G y$ = "there is a feature outside G that differentiates x and y"

- either it is in F, and we can list all the cases;
- or it is not, and since $x[F]$ then $\neg y[F \cup G]$.

Properties

Lemma

The constraint system terminates and yields a clause that is equivalent to the first one.

Properties

Lemma

The constraint system terminates and yields a clause that is equivalent to the first one.

Lemma

Let c be a clause $c = g_c \wedge \exists X \cdot l_c$ such that:

- *c is in normal form;*
- $\mathcal{V}(g_c) \cap X = \varnothing$;
- *every literal in l_c is about X;*
- *there is no $y[f]x$ with $x \in X$ and $y \notin X$.*

Then c is equivalent to g_c.

Does that even terminate?

R-NSim-Fence:

$$x[F] \land x \not\sim_G y \land c$$
$$\Rightarrow \quad x[F] \land \big(\neg y[F \cup G] \lor x \dot{\neq}_{F \setminus G}\, y\big) \land c$$

Does that even terminate?

R-NSim-Fence (for $F = \{f\}$ and $G = \varnothing$):

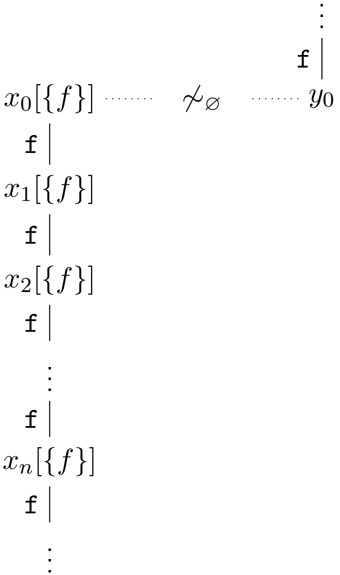
$$x[\{f\}] \wedge x \not\sim_{\varnothing} y \wedge c$$
$$\Rightarrow \quad x[\{f\}] \wedge (\neg y[\{f\}] \vee x \mathrel{\dot{\neq}_f} y) \wedge c$$

Does that even terminate?

R-NSim-Fence (for $F = \{f\}$ and $G = \varnothing$):

$$x[\{f\}] \wedge x \not\sim_\varnothing y \wedge c$$
$$\Rightarrow \quad \exists z, z' \cdot x[f]z \wedge y[f]z' \wedge z \not\sim_\varnothing z' \wedge x[\{f$$

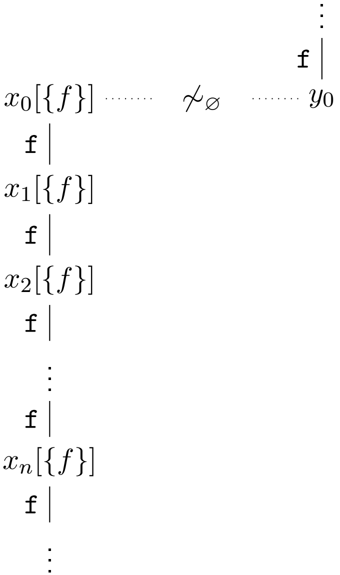
Does that even terminate?



R-NSIM-FENCE (for $F = \{f\}$ and $G = \varnothing$):

$$x[\{f\}] \wedge x \not\sim_\varnothing y \wedge c$$
$$\Rightarrow \quad \exists z, z' \cdot x[f]z \wedge y[f]z' \wedge z \not\sim_\varnothing z' \wedge x[\{f$$

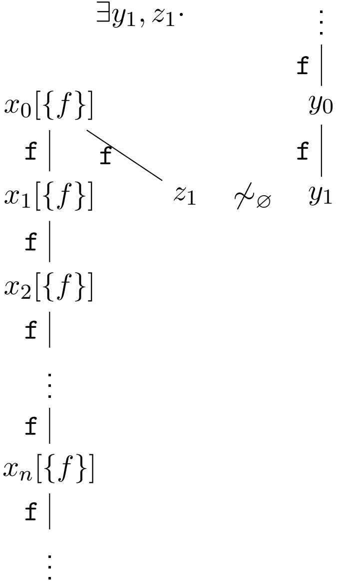
Does that even terminate?



R-NSIM-FENCE (for $F = \{f\}$ and $G = \varnothing$):

$$x[\{f\}] \wedge x \not\sim_\varnothing y \wedge c$$
$$\Rightarrow \quad \exists z, z' \cdot x[f]z \wedge y[f]z' \wedge z \not\sim_\varnothing z' \wedge x[\{f$$

- R-NSIM-FENCE with x_0 and y_0;

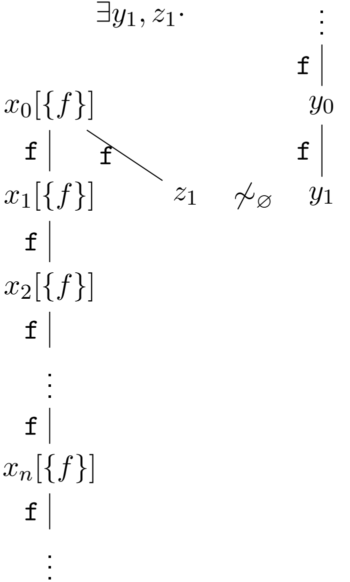


R-NSim-Fence (for $F = \{f\}$ and $G = \varnothing$):

$$x[\{f\}] \wedge x \not\sim_\varnothing y \wedge c$$
$$\Rightarrow \quad \exists z, z' \cdot x[f]z \wedge y[f]z' \wedge z \not\sim_\varnothing z' \wedge x[\{f$$

- R-NSim-Fence with x_0 and y_0;

Does that even terminate?

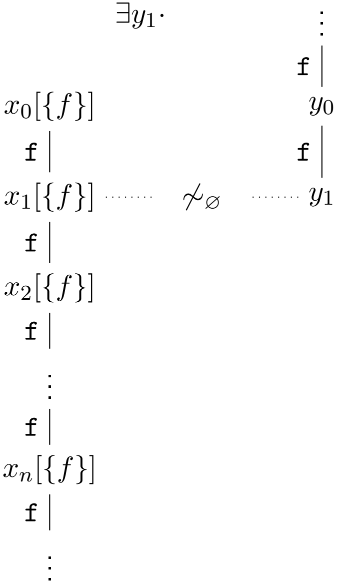


R-NSIM-FENCE (for $F = \{f\}$ and $G = \varnothing$):

$$x[\{f\}] \wedge x \not\sim_\varnothing y \wedge c$$
$$\Rightarrow \quad \exists z, z' \cdot x[f]z \wedge y[f]z' \wedge z \not\sim_\varnothing z' \wedge x[\{f$$

- R-NSIM-FENCE with x_0 and y_0;
- S-FEATS with x_1 and z_1

Does that even terminate?

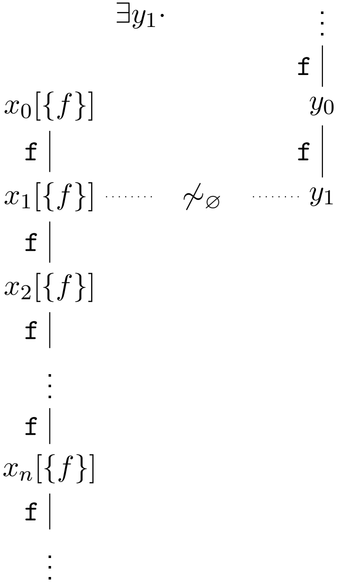


R-NSIM-FENCE (for $F = \{f\}$ and $G = \varnothing$):

$$x[\{f\}] \wedge x \not\sim_\varnothing y \wedge c$$
$$\Rightarrow \quad \exists z, z' \cdot x[f]z \wedge y[f]z' \wedge z \not\sim_\varnothing z' \wedge x[\{f$$

- R-NSIM-FENCE with x_0 and y_0;
- S-FEATS with x_1 and z_1

Does that even terminate?

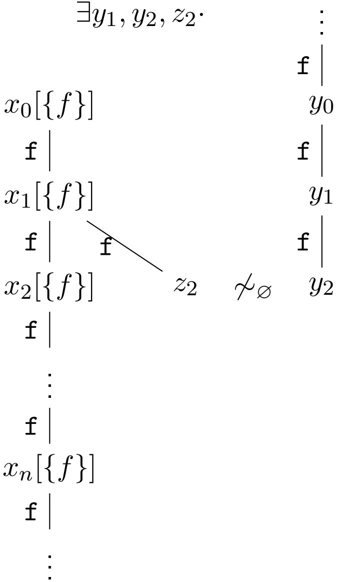


R-NSim-Fence (for $F = \{f\}$ and $G = \varnothing$):

$$x[\{f\}] \wedge x \not\sim_\varnothing y \wedge c$$
$$\Rightarrow \quad \exists z, z' \cdot x[f]z \wedge y[f]z' \wedge z \not\sim_\varnothing z' \wedge x[\{f$$

- R-NSim-Fence with x_0 and y_0;
- S-Feats with x_1 and z_1
- R-NSim-Fence with x_1 and y_1;

Does that even terminate?

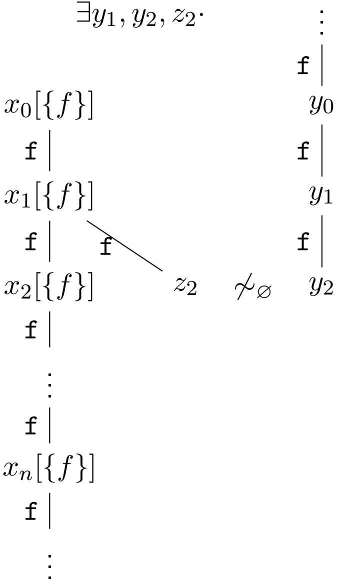


R-NSIM-FENCE (for $F = \{f\}$ and $G = \varnothing$):

$$x[\{f\}] \wedge x \not\sim_\varnothing y \wedge c$$
$$\Rightarrow \quad \exists z, z' \cdot x[f]z \wedge y[f]z' \wedge z \not\sim_\varnothing z' \wedge x[\{f$$

- R-NSIM-FENCE with x_0 and y_0;
- S-FEATS with x_1 and z_1
- R-NSIM-FENCE with x_1 and y_1;

Does that even terminate?

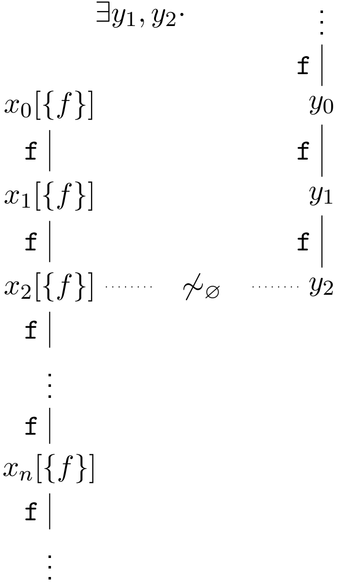


R-NSim-Fence (for $F = \{f\}$ and $G = \varnothing$):

$$x[\{f\}] \wedge x \not\sim_\varnothing y \wedge c$$
$$\Rightarrow \quad \exists z, z' \cdot x[f]z \wedge y[f]z' \wedge z \not\sim_\varnothing z' \wedge x[\{f$$

- R-NSim-Fence with x_0 and y_0;
- S-Feats with x_1 and z_1
- R-NSim-Fence with x_1 and y_1;
- S-Feats with x_2 and z_2

Does that even terminate?

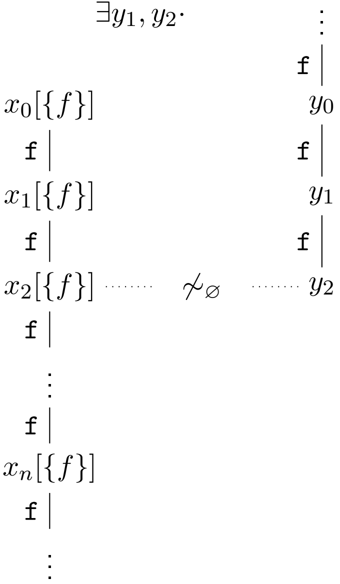


R-NSIM-FENCE (for $F = \{f\}$ and $G = \varnothing$):

$$x[\{f\}] \wedge x \not\sim_\varnothing y \wedge c$$
$$\Rightarrow \quad \exists z, z' \cdot x[f]z \wedge y[f]z' \wedge z \not\sim_\varnothing z' \wedge x[\{f$$

- R-NSIM-FENCE with x_0 and y_0;
- S-FEATS with x_1 and z_1
- R-NSIM-FENCE with x_1 and y_1;
- S-FEATS with x_2 and z_2

Does that even terminate?

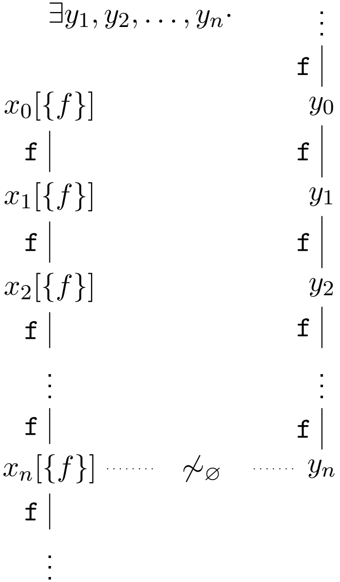


R-NSIM-FENCE (for $F = \{f\}$ and $G = \varnothing$):

$$x[\{f\}] \land x \not\sim_{\varnothing} y \land c$$
$$\Rightarrow \quad \exists z, z' \cdot x[f]z \land y[f]z' \land z \not\sim_{\varnothing} z' \land x[\{f$$

- R-NSIM-FENCE with x_0 and y_0;
- S-FEATS with x_1 and z_1
- R-NSIM-FENCE with x_1 and y_1;
- S-FEATS with x_2 and z_2
- ...

Does that even terminate?



R-NSIM-FENCE (for $F = \{f\}$ and $G = \varnothing$):

$$x[\{f\}] \wedge x \not\sim_{\varnothing} y \wedge c$$
$$\Rightarrow \quad \exists z, z' \cdot x[f]z \wedge y[f]z' \wedge z \not\sim_{\varnothing} z' \wedge x[\{f$$

- R-NSIM-FENCE with x_0 and y_0;
- S-FEATS with x_1 and z_1
- R-NSIM-FENCE with x_1 and y_1;
- S-FEATS with x_2 and z_2
- ...

Table of Contents

1. Description of filesystems
   - Unix filesystems
   - Static description
   - Directory update

2. Constraints
   - Definitions
   - Basic constraints
   - Negation

3. Usages
   - Decidability of the First-Order Theory
   - Automated Specification for Scripts: Proof of Concept

Weak Quantifier Elimination

Assume given a technique to transform $\exists X \cdot c$ into an equivalent $\forall X' \cdot c'$.

Weak Quantifier Elimination

Assume given a technique to transform $\exists X \cdot c$ into an equivalent $\forall X' \cdot c'$.

Take any closed formula

Weak Quantifier Elimination

Assume given a technique to transform $\exists X \cdot c$ into an equivalent $\forall X' \cdot c'$.

Take any closed formula, look at the last quantifier bloc

Weak Quantifier Elimination

Assume given a technique to transform $\exists X \cdot c$ into an equivalent $\forall X' \cdot c'$.

Take any closed formula, look at the last quantifier bloc:

- Universal

$$\forall \; \exists \cdots \forall X \cdot c$$

Weak Quantifier Elimination

Assume given a technique to transform $\exists X \cdot c$ into an equivalent $\forall X' \cdot c'$.

Take any closed formula, look at the last quantifier bloc:

- Universal, switch it to existential:

$$\forall\ \exists \cdots \forall X \cdot c \quad \Longrightarrow \quad \neg\ \exists\ \forall \cdots \exists X \cdot \neg c$$

Weak Quantifier Elimination

Assume given a technique to transform $\exists X \cdot c$ into an equivalent $\forall X' \cdot c'$.

Take any closed formula, look at the last quantifier bloc:

- Universal, switch it to existential:

$$\forall \; \exists \cdots \forall X \cdot c \quad \Longrightarrow \quad \neg \; \exists \; \forall \cdots \exists X \cdot \neg c$$

- Existential:

Weak Quantifier Elimination

Assume given a technique to transform $\exists X \cdot c$ into an equivalent $\forall X' \cdot c'$.

Take any closed formula, look at the last quantifier bloc:

- Universal, switch it to existential:

$$\forall \; \exists \cdots \forall X \cdot c \quad \Longrightarrow \quad \neg \; \exists \; \forall \cdots \exists X \cdot \neg c$$

- Existential:
  - If there is an other bloc before

$$\forall \; \exists \cdots \forall Y \cdot \exists X \cdot c$$

Weak Quantifier Elimination

Assume given a technique to transform $\exists X \cdot c$ into an equivalent $\forall X' \cdot c'$.

Take any closed formula, look at the last quantifier bloc:

- Universal, switch it to existential:

$$\forall \; \exists \cdots \forall X \cdot c \quad \Longrightarrow \quad \neg \; \exists \; \forall \cdots \exists X \cdot \neg c$$

- Existential:
  - If there is an other bloc before, use the given technique:

$$\forall \; \exists \cdots \forall Y \cdot \exists X \cdot c \quad \Longrightarrow \quad \forall \; \exists \cdots \forall Y, X' \cdot c'$$

Weak Quantifier Elimination

Assume given a technique to transform $\exists X \cdot c$ into an equivalent $\forall X' \cdot c'$.

Take any closed formula, look at the last quantifier bloc:

- Universal, switch it to existential:

$$\forall\ \exists \cdots \forall X \cdot c \quad \Longrightarrow \quad \neg\ \exists\ \forall \cdots \exists X \cdot \neg c$$

- Existential:
  - If there is an other bloc before, use the given technique:

$$\forall\ \exists \cdots \forall Y \cdot \exists X \cdot c \quad \Longrightarrow \quad \forall\ \exists \cdots \forall Y, X' \cdot c'$$

  - If not

Weak Quantifier Elimination

Assume given a technique to transform $\exists X \cdot c$ into an equivalent $\forall X' \cdot c'$.

Take any closed formula, look at the last quantifier bloc:

- Universal, switch it to existential:

$$\forall\, \exists \cdots \forall X \cdot c \quad \Longrightarrow \quad \neg\, \exists\, \forall \cdots \exists X \cdot \neg c$$

- Existential:
  - If there is an other bloc before, use the given technique:

$$\forall\, \exists \cdots \forall Y \cdot \exists X \cdot c \quad \Longrightarrow \quad \forall\, \exists \cdots \forall Y, X' \cdot c'$$

  - If not, then it is only a satisfiability question.

Weak Quantifier Elimination

Previous slide said: "Assume given a technique to transform $\exists X \cdot c$ into an equivalent $\forall X' \cdot c'$."

Weak Quantifier Elimination

Previous slide said: "Assume given a technique to transform $\exists X \cdot c$ into an equivalent $\forall X' \cdot c'$."

Here is what we have:

Lemma

Let c be a clause $c = g_c \wedge \exists X \cdot l_c$ such that:

- *...*
- *there is no $y[f]x$ with $x \in X$ and $y \notin X$.*

Then c is equivalent to g_c.

Weak Quantifier Elimination

Previous slide said: "Assume given a technique to transform $\exists X \cdot c$ into an equivalent $\forall X' \cdot c'$."

Here is what we have:

Lemma

Let c be a clause $c = g_c \wedge \exists X \cdot l_c$ such that:

- *...*
- *there is no $y[f]x$ with $x \in X$ and $y \notin X$.*

Then c is equivalent to g_c.

Weak Quantifier Elimination

Previous slide said: "Assume given a technique to transform $\exists X \cdot c$ into an equivalent $\forall X' \cdot c'$."

Here is what we have:

Lemma

Let c be a clause $c = g_c \wedge \exists X \cdot l_c$ such that:

- *...*
- *there is no $y[f]x$ with $x \in X$ and $y \notin X$.*

Then c is equivalent to g_c.

Lukily:

$$\exists X, x \cdot (y[f]x \wedge c)$$

Weak Quantifier Elimination

Previous slide said: "Assume given a technique to transform $\exists X \cdot c$ into an equivalent $\forall X' \cdot c'$."

Here is what we have:

Lemma

Let c be a clause $c = g_c \wedge \exists X \cdot l_c$ such that:

- *...*
- *there is no $y[f]x$ with $x \in X$ and $y \notin X$.*

Then c is equivalent to g_c.

Lukily:

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

Table of Contents

1. Description of filesystems
   - Unix filesystems
   - Static description
   - Directory update

2. Constraints
   - Definitions
   - Basic constraints
   - Negation

3. Usages
   - Decidability of the First-Order Theory
   - Automated Specification for Scripts: Proof of Concept

Demo!