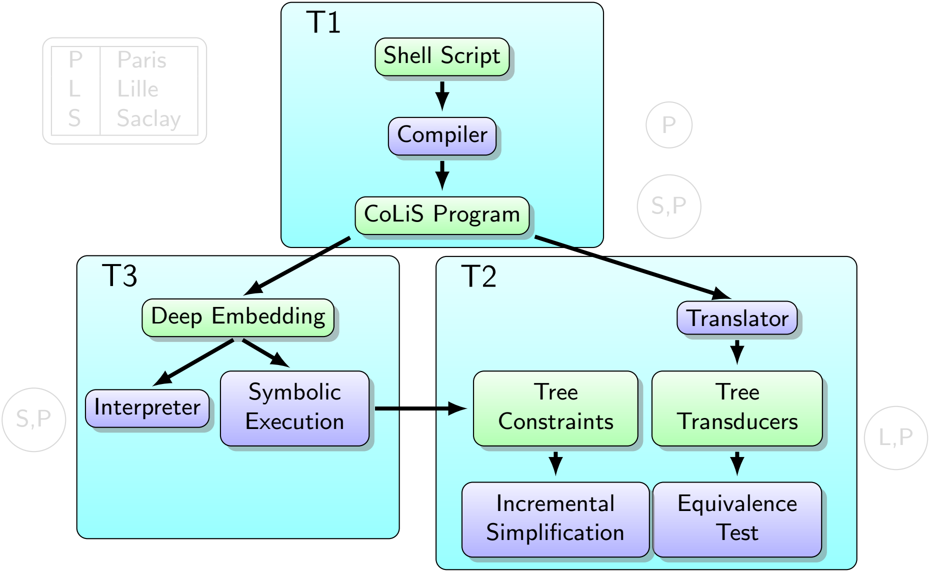
Unix filesystem and graph constraints

Nicolas Jeannerod

INSTITUT
DE RECHERCHE
EN INFORMATIQUE
FONDAMENTALE

Journées PPS, October 12, 2017

The CoLiS project



The CoLiS project

P	Paris
L	Lille
S	Saclay

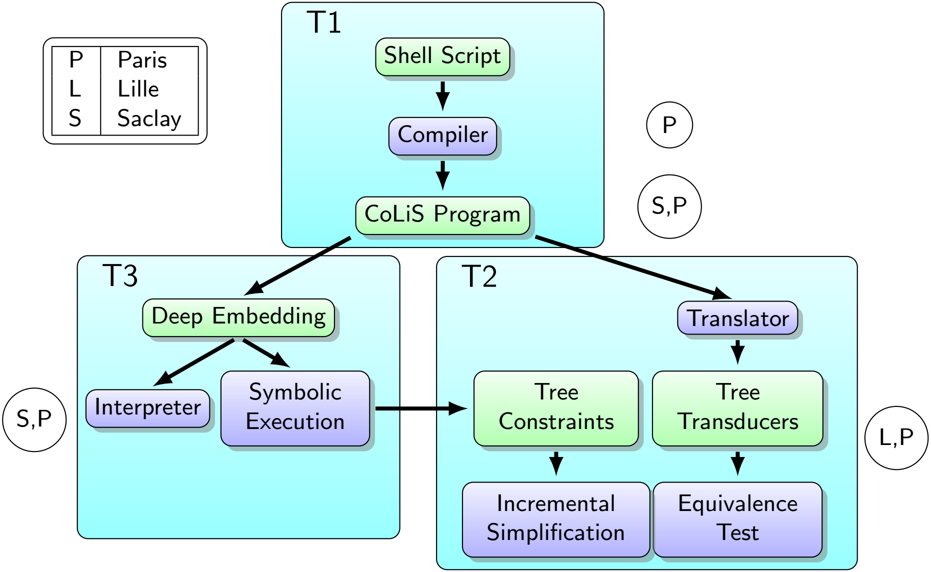
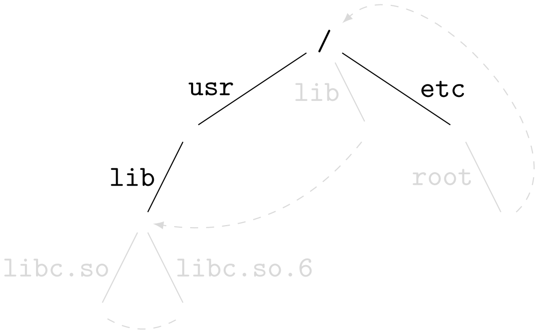


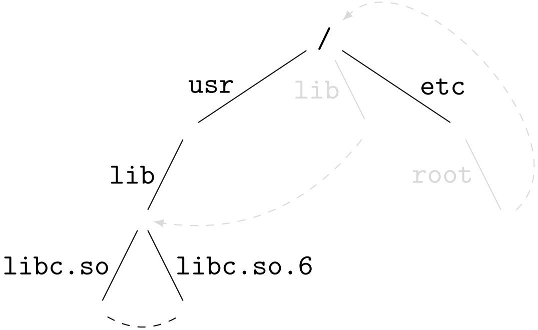
Table of Contents

Unix file system



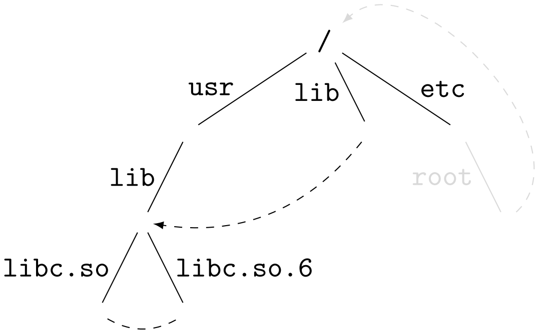
- Basically a tree with labelled nodes and edges;
- There can be sharing at the leafs (hard link between files);
- There can be pointers to other parts of the tree (symbolic links) which may form cycles.

Unix file system



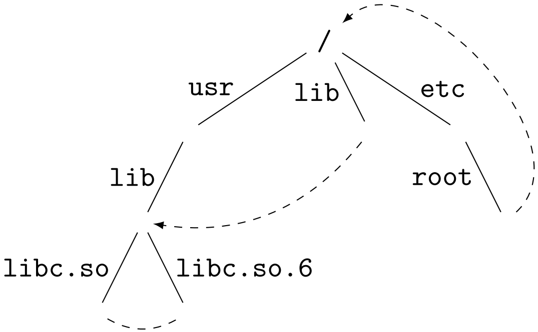
- Basically a tree with labelled nodes and edges;
- There can be sharing at the leafs (hard link between files);
- There can be pointers to other parts of the tree (symbolic links) which may form cycles.

Unix file system



- 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.

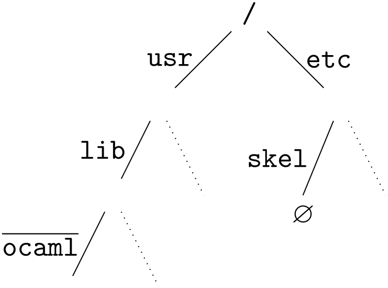
Unix file system



- 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.

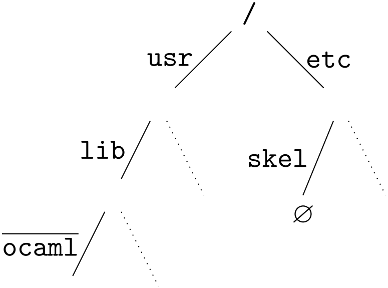
Table of Contents

Static description



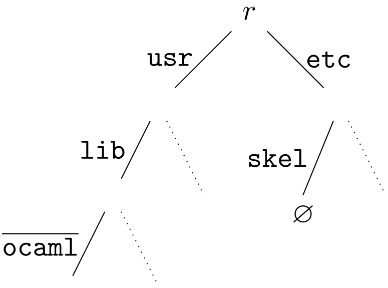
$$r \;\middle|\; \exists u, v, x, w. \quad \begin{array}{l} r[\mathtt{usr}]v \wedge v[\mathtt{lib}]x \wedge x[\mathtt{ocaml}]\uparrow \\ \wedge\, r[\mathtt{etc}]w \wedge w[\mathtt{skel}]u \wedge u\varnothing \end{array}$$

Static description



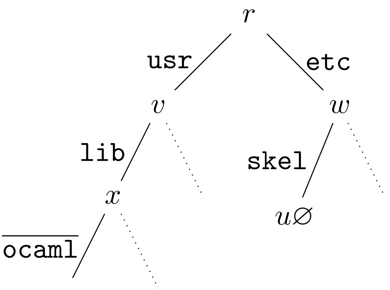
$$r \;\Big|\; \exists u, v, x, w.\; \begin{array}{l} r[\texttt{usr}]v \wedge v[\texttt{lib}]x \wedge x[\texttt{ocaml}]\uparrow \\ \wedge\, r[\texttt{etc}]w \wedge w[\texttt{skel}]u \wedge u\varnothing \end{array}$$

Static description



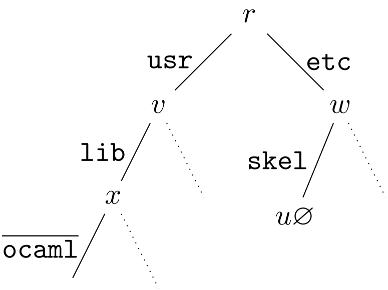
$$r \;\Big|\; \exists u, v, x, w.\; r[\texttt{usr}]v \wedge v[\texttt{lib}]x \wedge x[\texttt{ocaml}]\uparrow \wedge r[\texttt{etc}]w \wedge w[\texttt{skel}]u \wedge u\varnothing$$

Static description



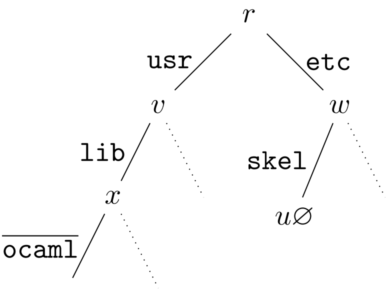
$$r \;\Big|\; \exists u, v, x, w. \quad r[\texttt{usr}]v \wedge v[\texttt{lib}]x \wedge x[\overline{\texttt{ocaml}}]\uparrow \wedge r[\texttt{etc}]w \wedge w[\texttt{skel}]u \wedge u\varnothing$$

Static description



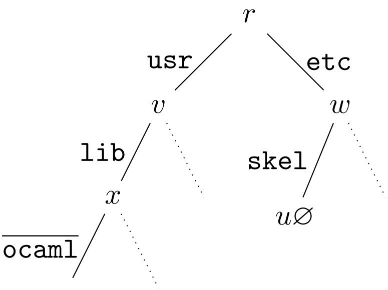
$$r \;\Big|\; \exists u, v, x, w.\quad r[\texttt{usr}]v \land v[\texttt{lib}]x \land x[\overline{\texttt{ocaml}}]\uparrow \land r[\texttt{etc}]w \land w[\texttt{skel}]u \land u\varnothing$$

Static description



$$r \;\middle|\; \exists u, v, x, w.\ \ r[\texttt{usr}]v \wedge v[\texttt{lib}]x \wedge x[\texttt{ocaml}]\uparrow \wedge r[\texttt{etc}]w \wedge w[\texttt{skel}]u \wedge u\varnothing$$

Static description



$$r \;\Big|\; \exists u, v, x, w. \quad \begin{array}{l} r[\texttt{usr}]v \wedge v[\texttt{lib}]x \wedge x[\overline{\texttt{ocaml}}]\uparrow \\ \wedge\, r[\texttt{etc}]w \wedge w[\texttt{skel}]u \wedge u\varnothing \end{array}$$

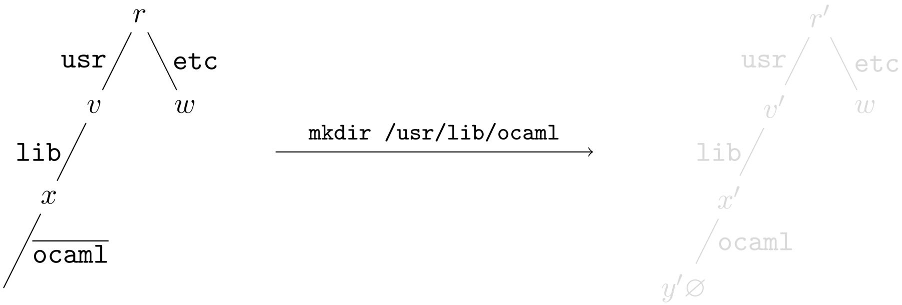
Table of Contents

Directory update

We want something like:

$$r' = r[\texttt{usr} \to v'] \wedge v' = v[\texttt{lib} \to x'] \wedge x' = x[\texttt{ocaml} \to y'] \wedge y'\varnothing$$

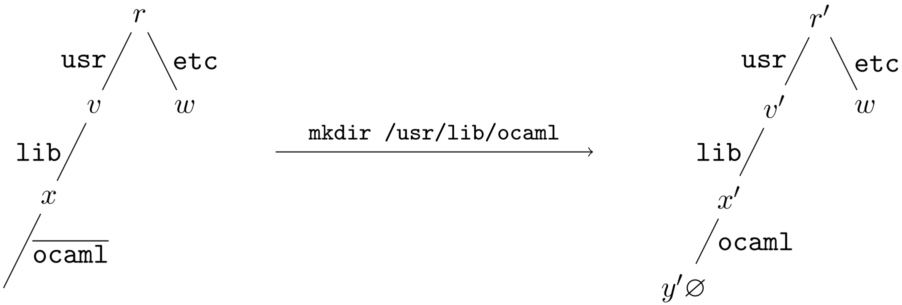
Directory update



We want something like:

$$r' = r[\texttt{usr} \rightarrow v'] \wedge v' = v[\texttt{lib} \rightarrow x'] \wedge x' = x[\texttt{ocaml} \rightarrow y'] \wedge y'\varnothing$$

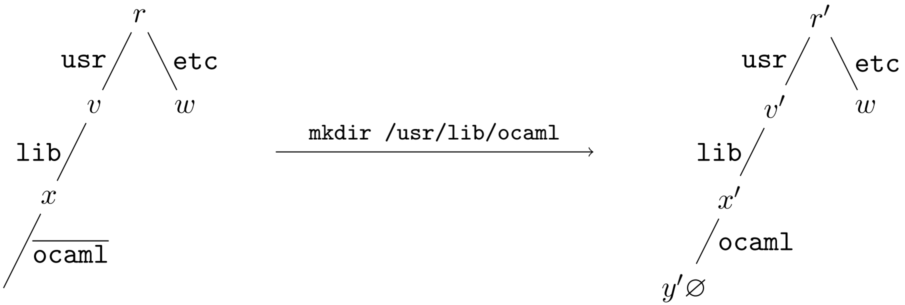
Directory update



We want something like:

$$r' = r[\texttt{usr} \to v'] \land v' = v[\texttt{lib} \to x'] \land x' = x[\texttt{ocaml} \to y'] \land y'\varnothing$$

Directory update



We want something like:

$$r' = r[\mathtt{usr} \to v'] \land v' = v[\mathtt{lib} \to x'] \land x' = x[\mathtt{ocaml} \to y'] \land y'\varnothing$$

Er.. is that really what we want?

- Asymmetric:

$$y = x[f \to v]$$

- Makes it hard to eliminate variables:

$$y = x[f \to v] \land z = x[g \to w]$$

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

$$y \sim_f x$$

  - "y points to v through f"

$$y[f]v$$

Er.. is that really what we want?

- Asymmetric:

$$y = x[f \to v]$$

- Makes it hard to eliminate variables:

$$y = x[f \to v] \land z = x[g \to w]$$

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

Er.. is that really what we want?

- Asymmetric:

$$y = x[f \to v]$$

- Makes it hard to eliminate variables:

$$y = x[f \to v] \land z = x[g \to w]$$

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

$$y \sim_f x$$

  - "y points to v through f":

$$y[f]v$$

Er.. is that really what we want?

- Asymmetric:

$$y = x[f \to v]$$

- Makes it hard to eliminate variables:

$$y = x[f \to v] \wedge z = x[g \to w]$$

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

$$y \sim_f x$$

  - "y points to v through f":

$$y[f]v$$

Er.. is that really what we want?

- Asymmetric:

$$y = x[f \to v]$$

- Makes it hard to eliminate variables:

$$y = x[f \to v] \wedge z = x[g \to w]$$

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

$$y \sim_f x$$

  - "y points to v through f":

$$y[f]v$$

Er.. is that really what we want?

- Asymmetric:

$$y = x[f \to v]$$

- Makes it hard to eliminate variables:

$$y = x[f \to v] \land z = x[g \to w]$$

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

$$y \sim_f x$$

  - "y points to v through f":

$$y[f]v$$

Er.. is that really what we want?

- Asymmetric:

$$y = x[f \to v]$$

- Makes it hard to eliminate variables:

$$y = x[f \to v] \wedge z = x[g \to w]$$

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

$$y \sim_f x$$

  - "y points to v through f":

$$y[f]v$$

Much better

- Allows to express the update:

$$y = x[f \to v] := y \sim_f x \wedge y[f]v$$

- Symmetric:

$$y \sim_f x \iff x \sim_f y$$

- Transitive:

$$y \sim_f x \wedge z \sim_f x \implies y \sim_f z$$

Much better

- Allows to express the update:

$$y = x[f \to v] := y \sim_f x \wedge y[f]v$$

- Symmetric:

$$y \sim_f x \iff x \sim_f y$$

- Transitive:

$$y \sim_f x \wedge z \sim_f x \implies y \sim_f z$$

Much better

- Allows to express the update:

$$y = x[f \to v] := y \sim_f x \land y[f]v$$

- Symmetric:

$$y \sim_f x \iff x \sim_f y$$

- Transitive:

$$y \sim_f x \land z \sim_f x \implies y \sim_f z$$

Table of Contents

Constraints

- $\mathcal{K}$ finite set; $Dir \in \mathcal{K}$; $\mathcal{F}$ infinite set
- Finite trees labelled with $\mathcal{K}$ on nodes and $\mathcal{F}$ on edges
- x, y variables; $K \in \mathcal{K}$, $f \in \mathcal{F}$, $F \subseteq \mathcal{F}$

Equality	$x \doteq y$	$K(x)$	Kind
Feature	$x f y$	$x f \uparrow$	Absence
Fence	$x F$	$x \sim_F y$	Similarity

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

Constraints

- $\mathcal{K}$ finite set; $Dir \in \mathcal{K}$; $\mathcal{F}$ infinite set
- Finite trees labelled with $\mathcal{K}$ on nodes and $\mathcal{F}$ on edges
- x, y variables; $K \in \mathcal{K}$, $f \in \mathcal{F}$, $F \subseteq \mathcal{F}$

Equality	$x \doteq y$	$K(x)$	Kind
Feature	$x f y$	$x f \uparrow$	Absence
Fence	$x F$	$x \sim_F y$	Similarity

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

Constraints

- $\mathcal{K}$ finite set; $Dir \in \mathcal{K}$; $\mathcal{F}$ infinite set
- Finite trees labelled with $\mathcal{K}$ on nodes and $\mathcal{F}$ on edges
- x, y variables; $K \in \mathcal{K}$, $f \in \mathcal{F}$, $F \subseteq \mathcal{F}$

Equality	$x \doteq y$	$K(x)$	Kind
Feature	$x f y$	$x f \uparrow$	Absence
Fence	$x F$	$x \sim_F y$	Similarity

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

Constraints

- $\mathcal{K}$ finite set; $Dir \in \mathcal{K}$; $\mathcal{F}$ infinite set
- Finite trees labelled with $\mathcal{K}$ on nodes and $\mathcal{F}$ on edges
- x, y variables; $K \in \mathcal{K}$, $f \in \mathcal{F}$, $F \subseteq \mathcal{F}$

Equality	$x \doteq y$	$K(x)$	Kind
Feature	$x f y$	$x f \uparrow$	Absence
Fence	$x F$	$x \sim_F y$	Similarity

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

Constraints

- $\mathcal{K}$ finite set; $Dir \in \mathcal{K}$; $\mathcal{F}$ infinite set
- Finite trees labelled with $\mathcal{K}$ on nodes and $\mathcal{F}$ on edges
- x, y variables; $K \in \mathcal{K}$, $f \in \mathcal{F}$, $F \subseteq \mathcal{F}$

Equality	$x \doteq y$	$K(x)$	Kind
Feature	$x f y$	$x f \uparrow$	Absence
Fence	$x F$	$x \sim_F y$	Similarity

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

Game plan

1. Write a system of rewriting rules;
2. Prove that the system terminates (help it if needed);
3. Prove that the rules respect equivalences:

Lemma

If ϕ reduces to ψ, then $\models \phi \leftrightarrow \psi$.

4. Prove nice properties on the normal forms:

Lemma

If ϕ is in normal form, then it is either satisfiable or $\bot$.

Game plan

1. Write a system of rewriting rules;
2. Prove that the system terminates (help it if needed);

3. Prove that the rules respect equivalences:

Lemma

If ϕ reduces to ψ, then $\models \phi \leftrightarrow \psi$.

4. Prove nice properties on the normal forms:

Lemma

If ϕ is in normal form, then it is either satisfiable or $\bot$.

Game plan

1. Write a system of rewriting rules;
2. Prove that the system terminates (help it if needed);
3. Prove that the rules respect equivalences:

Lemma

If ϕ reduces to ψ, then $\models \phi \leftrightarrow \psi$.

4. Prove nice properties on the normal forms:

Lemma

If ϕ is in normal form, then it is either satisfiable or $\bot$.

Game plan

1. Write a system of rewriting rules;
2. Prove that the system terminates (help it if needed);
3. Prove that the rules respect equivalences:

Lemma

If ϕ reduces to ψ, then $\models \phi \leftrightarrow \psi$.

4. Prove nice properties on the normal forms:

Lemma

If ϕ is in normal form, then it is either satisfiable or $\bot$.

Table of Contents

Basic rules

Basic constraints: conjunction of positive atoms.

Simpl-Feats
$$\frac{x f y \wedge x f z}{x f y \wedge y \doteq z}$$

C-Feat-Abs
$$\frac{x f y \wedge x f \uparrow}{\bot}$$

Intro-Feat-Sim
$$\frac{x \sim_F y \wedge x f z}{x \sim_F y \wedge x f z \wedge y f z}\ f \notin F$$

Intro-Sim-Sims
$$\frac{x \sim_F y \wedge y \sim_G z}{x \sim_F y \wedge y \sim_G z \wedge x \sim_{(F \cup G)} z}$$

Basic rules

Basic constraints: conjunction of positive atoms.

$$\textsc{Simpl-Feats}\ \frac{x f y \wedge x f z}{x f y \wedge y \doteq z} \qquad \textsc{C-Feat-Abs}\ \frac{x f y \wedge x f \uparrow}{\bot}$$

$$\textsc{Intro-Feat-Sim}\ \frac{x \sim_F y \wedge x f z}{x \sim_F y \wedge x f z \wedge y f z}\ f \notin F \qquad \textsc{Intro-Sim-Sims}\ \frac{x \sim_F y \wedge y \sim_G z}{x \sim_F y \wedge y \sim_G z \wedge x \sim_{(F \cup G)} z}$$

Basic rules

Basic constraints: conjunction of positive atoms.

SIMPL-FEATS
$$\frac{x f y \wedge x f z}{x f y \wedge y \doteq z}$$

C-FEAT-ABS
$$\frac{x f y \wedge x f \uparrow}{\bot}$$

INTRO-FEAT-SIM
$$\frac{x \sim_F y \wedge x f z}{x \sim_F y \wedge x f z \wedge y f z} \; f \notin F$$

INTRO-SIM-SIMS
$$\frac{x \sim_F y \wedge y \sim_G z}{x \sim_F y \wedge y \sim_G z \wedge x \sim_{(F \cup G)} z}$$

Basic rules

Basic constraints: conjunction of positive atoms.

$$\textsc{Simpl-Feats}\quad \frac{x\,f\,y \wedge x\,f\,z}{x\,f\,y \wedge y \doteq z}$$

$$\textsc{C-Feat-Abs}\quad \frac{x\,f\,y \wedge x\,f\!\uparrow}{\bot}$$

$$\textsc{Intro-Feat-Sim}\quad \frac{x \sim_F y \wedge x\,f\,z}{x \sim_F y \wedge x\,f\,z \wedge y\,f\,z}\ f \notin F$$

$$\textsc{Intro-Sim-Sims}\quad \frac{x \sim_F y \wedge y \sim_G z}{x \sim_F y \wedge y \sim_G z \wedge x \sim_{(F \cup G)} z}$$

Basic constraints

Basic constraints: conjunction of positive atoms

- Equality: rewritten
- Kind: Static "positive" information
- Feature: Static "positive" information
- Absence: Static "negative" information
- Fence: Static "negative" information
- Similarity: **Dynamic** information

$r[\texttt{usr}]v \wedge v[\texttt{lib}]x \wedge x[\texttt{ocaml}]\uparrow$
$\wedge r[\texttt{etc}]w \wedge w[\texttt{skel}]u \wedge u\varnothing$
$\wedge \dots$

Basic constraints

Basic constraints: conjunction of positive atoms

- Equality: rewritten
- Kind: Static "positive" information
- Feature: Static "positive" information
- Absence: Static "negative" information
- Fence: Static "negative" information
- Similarity: **Dynamic** information

$r[\texttt{usr}]v \wedge v[\texttt{lib}]x \wedge x[\texttt{ocaml}]\uparrow$
$\wedge r[\texttt{etc}]w \wedge w[\texttt{skel}]u \wedge u\varnothing$
$\wedge \dots$

Basic constraints

Basic constraints: conjunction of positive atoms

- Equality: rewritten
- Kind: Static "positive" information
- Feature: Static "positive" information
- Absence: Static "negative" information
- Fence: Static "negative" information
- Similarity: **Dynamic** information

$r[\mathtt{usr}]v \wedge v[\mathtt{lib}]x \wedge x[\mathtt{ocaml}]\uparrow$
$\wedge r[\mathtt{etc}]w \wedge w[\mathtt{skel}]u \wedge u\varnothing$
$\wedge \ldots$

Basic constraints

Basic constraints: conjunction of positive atoms

- Equality: rewritten
- Kind: Static "positive" information
- Feature: Static "positive" information
- Absence: Static "negative" information
- Fence: Static "negative" information
- Similarity: **Dynamic** information

$r[\mathtt{usr}]v \wedge v[\mathtt{lib}]x \wedge x[\mathtt{ocaml}]\uparrow$
$\wedge r[\mathtt{etc}]w \wedge w[\mathtt{skel}]u \wedge u\varnothing$
$\wedge \ldots$

Basic constraints

Basic constraints: conjunction of positive atoms

- Equality: rewritten
- Kind: Static "positive" information
- Feature: Static "positive" information
- Absence: Static "negative" information
- Fence: Static "negative" information
- Similarity: **Dynamic** information

$r[\texttt{usr}]v \land v[\texttt{lib}]x \land x[\texttt{ocaml}]\uparrow$
$\land r[\texttt{etc}]w \land w[\texttt{skel}]u \land u\emptyset$
$\land \dots$

Basic constraints

Basic constraints: conjunction of positive atoms

- Equality: rewritten
- Kind: Static "positive" information
- Feature: Static "positive" information
- Absence: Static "negative" information
- Fence: Static "negative" information
- Similarity: **Dynamic** information

$r[\texttt{usr}]v \wedge v[\texttt{lib}]x \wedge x[\texttt{ocaml}] \uparrow$
$\wedge r[\texttt{etc}]w \wedge w[\texttt{skel}]u \wedge u\varnothing$
$\wedge \ldots$

Basic constraints

Basic constraints: conjunction of positive atoms

- Equality: rewritten
- Kind: Static “positive” information
- Feature: Static “positive” information
- Absence: Static “negative” information
- Fence: Static “negative” information
- Similarity: **Dynamic** information

$r[\texttt{usr}]v \wedge v[\texttt{lib}]x \wedge x[\texttt{ocaml}]\uparrow$
$\wedge r[\texttt{etc}]w \wedge w[\texttt{skel}]u \wedge u\varnothing$
$\wedge \dots$

Basic constraints

Basic constraints: conjunction of positive atoms

- Equality: rewritten
- Kind: Static "positive" information
- Feature: Static "positive" information
- Absence: Static "negative" information
- Fence: Static "negative" information
- Similarity: **Dynamic** information

$r[\texttt{usr}]v \wedge v[\texttt{lib}]x \wedge x[\texttt{ocaml}]\uparrow$
$\wedge r[\texttt{etc}]w \wedge w[\texttt{skel}]u \wedge u\varnothing$
$\wedge \ldots$

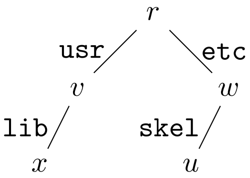


Table of Contents

Negation: new players, new rules

- $\neg xF$: there exist a $g \notin F$ such that $xg{\downarrow}$;
- $x \not\sim_F y$: there exist a $g \notin F$ such that $x \doteq\!\!\!\!/_g\ y$;

Repl-NKind

$$\frac{\neg K(x)}{\bigvee_{L \in \mathcal{K}} L(x)}$$

Repl-NAbs

$$\frac{\neg xf{\uparrow}}{\exists z.xfz}$$

Repl-NSim-NFence

$$\frac{xF \wedge x \not\sim_G y}{xF \wedge \neg xG} \quad F \subseteq G$$

Negation: new players, new rules

- $\neg xF$: there exist a $g \notin F$ such that $xg\downarrow$;
- $x \not\sim_F y$: there exist a $g \notin F$ such that $x \dot{\neq}_g y$;

$$\textsc{Repl-NKind}\quad \frac{\neg K(x)}{\bigvee_{L \in \mathcal{K}} L(x)}$$

$$\textsc{Repl-NAbs}\quad \frac{\neg xf\uparrow}{\exists z . xfz}$$

$$\textsc{Repl-NSim-NFence}\quad \frac{xF \wedge x \not\sim_G y}{xF \wedge \neg xG} \quad F \subseteq G$$

Negation: new players, new rules

- $\neg xF$: there exist a $g \notin F$ such that $xg{\downarrow}$;
- $x \not\sim_F y$: there exist a $g \notin F$ such that $x \mathrel{\dot{\neq}}_g y$;

$$\textsc{Repl-NKind}\quad \frac{\neg K(x)}{\bigvee_{L \in \mathcal{K}} L(x)}$$

$$\textsc{Repl-NAbs}\quad \frac{\neg xf{\uparrow}}{\exists z. xfz}$$

$$\textsc{Repl-NSim-NFence}\quad \frac{xF \wedge x \not\sim_G y}{xF \wedge \neg xG} \quad F \subseteq G$$

Negation: new players, new rules

- $\neg xF$: there exist a $g \notin F$ such that $xg\downarrow$;
- $x \not\sim_F y$: there exist a $g \notin F$ such that $x \mathrel{\dot{\neq}}_g y$;

$$\textsc{Repl-NKind}\quad \frac{\neg K(x)}{\bigvee_{L \in \mathcal{K}} L(x)}$$

$$\textsc{Repl-NAbs}\quad \frac{\neg xf\uparrow}{\exists z . xfz}$$

$$\textsc{Repl-NSim-NFence}\quad \frac{xF \land x \not\sim_G y}{xF \land \neg xG} \quad F \subseteq G$$

Quantifier elimination

- Goal: be able to change an existentially quantified block into a universally quantified one.

$$\left(\exists\vec{X}.\bigwedge\dots\right)\leftrightarrow\left(\forall\vec{Y}.\bigvee\bigwedge\dots\right)$$

- Special rules:

$$\textsc{Enlarg-Feat-Local}\quad\frac{\exists x.\exists\vec{X}.(yfx\wedge\phi(x,\vec{X}))}{yf{\downarrow}\wedge\forall x.\exists\vec{X}.(yfx\rightarrow\phi(x,\vec{X}))}$$

Quantifier elimination

- Goal: be able to change an existentially quantified block into a universally quantified one.

$$\left(\exists \vec{X}. \bigwedge \dots\right) \leftrightarrow \left(\forall \vec{Y}. \bigvee \bigwedge \dots\right)$$

- Special rules:

$$\textsc{Enlarg-Feat-Local}\quad \frac{\exists x. \exists \vec{X}. (y f x \wedge \phi(x, \vec{X}))}{y f \downarrow \wedge \forall x. \exists \vec{X}. (y f x \rightarrow \phi(x, \vec{X}))}$$

Lemma of 31 August

Lemma (31 August)

Let ϕ be a conjunction of the form:

$$\phi(\vec{X}, \vec{Y}) = \left(\bigwedge \textit{stuff about } \vec{X}\right) \wedge \psi(\vec{Y})$$

in normal form for our system.
Then we have:

$$\models \forall \vec{Y}.\left(\exists \vec{X}.\ \phi(\vec{X}, \vec{Y})\right) \leftrightarrow \psi(\vec{Y})$$

- The system propagates all the useful information
- We can just remove what we don't need!

Lemma of 31 August

Lemma (31 August)

Let ϕ be a conjunction of the form:

$$\phi(\vec{X},\vec{Y}) = \left(\bigwedge \textit{stuff about } \vec{X}\right) \wedge \psi(\vec{Y})$$

in normal form for our system.
Then we have:

$$\models \forall\vec{Y}.\left(\exists\vec{X}.\ \phi(\vec{X},\vec{Y})\right) \leftrightarrow \psi(\vec{Y})$$

- The system propagates all the useful information
- We can just remove what we don't need!

Thank you for your attention!

Recap':

- Need constraints on graphs to represent relations on file systems;
- Extend "feature trees" with $x \sim_F y$ ("x and y are the same, except maybe for the features in F");
- Use a system of rewrite rules whose normal forms have nice properties.

Future work:

- Cleanup, formalise in a technical report;
- Add inodes, permissions, timestamps, etc.
- Implement an efficient version for the existential subset.