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

Nicolas Jeannerod Ralf Treinen

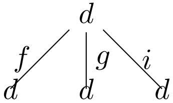
INSTITUT
DE RECHERCHE
EN INFORMATIQUE
FONDAMENTALE

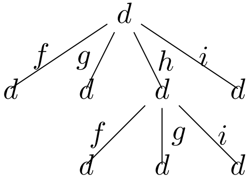
PROJET FINANCÉ PAR L'ANR - PROJECT FUNDED BY THE ANR
ANR

IJCAR'18 – July 16, 2018

Features Trees

- Unranked unordered trees.





- Least fixpoint of:

$$\mathcal{FT} = \mathcal{D} \times \left(\mathcal{F} \rightsquigarrow \mathcal{FT} \right)$$

Decoration s
(left abstract)

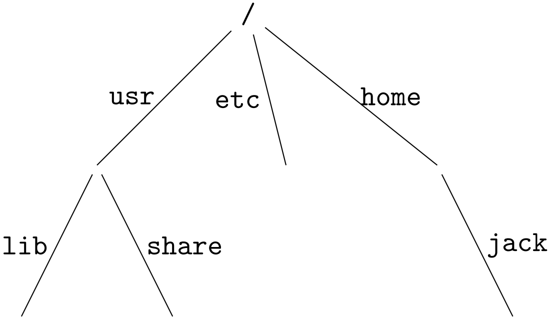
Infinite set
of features

Partial function
with finite domain

Origin of Feature Trees

- Computational linguistics [eg. Smolka, ’92]
- Artificial intelligence [Aït-Kaci]
- (Constraint) (logic) programming [Aït-Kaci, Backofen, Podelski, Smolka, Treinen, ’94]

Our Use Case – The Unix Filesystem



/

usr

etc

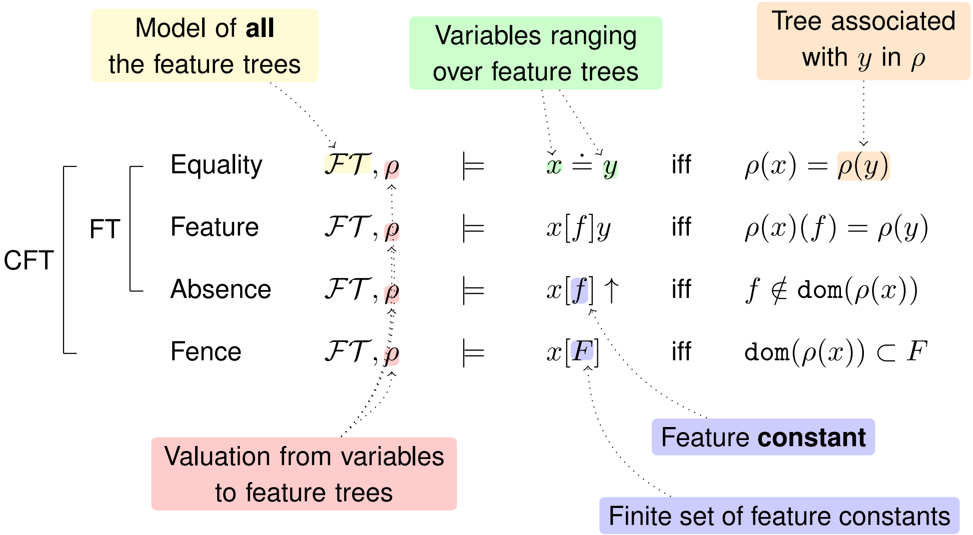
home

lib

share

jack

First Order Logics of Feature Trees



Model of **all** the feature trees

Variables ranging over feature trees

Tree associated with y in ρ

CFT	FT	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	$\mathtt{dom}(\rho(x)) \subset F$

Valuation from variables to feature trees

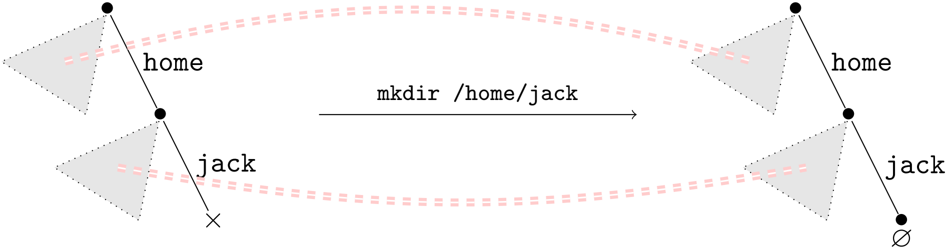
Feature **constant**

Finite set of feature constants

Known Decidability of First Order Logics

- FT: $x \doteq y$ $x[f]y$ $x[f]\uparrow$ [Backofen, Smolka, ’92]
- CFT: $x \doteq y$ $x[f]y$ $x[f]\uparrow$ $x[F]$ [Backofen, ’94]
  [Backofen, Treinen, ’94]
- FT with first-class features proven **undecidable** [Treinen, ’93]

Why We Need More

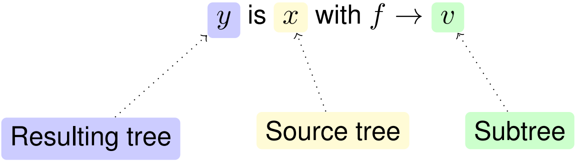


```
mkdir /home/jack
```

$$C(r,r') = \exists x, x', y' \left\{ \begin{array}{l} r[\texttt{home}]x \land x[\texttt{jack}] \uparrow \\ \land\, r'[\texttt{home}]x' \land x'[\texttt{jack}]y' \land y'[\varnothing] \\ \land\, r' \text{ is } r \text{ with } \texttt{home} \to x' \land x' \text{ is } x \text{ with } \texttt{jack} \to y' \end{array} \right.$$

How To Reason About Update Constraints?

- **Problem:** It is completely asymmetric.



y is x with $f \rightarrow v$

Resulting tree

Source tree

Subtree

- Hard to simplify when we have several of them:

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

Equivalent Presentation – The Similarity

$$\mathcal{FT}, \rho \quad \models \quad x \sim_F y \quad \text{iff} \quad \rho(x)|_{c_F} = \rho(y)|_{c_F}$$

Finite set of feature constants

- ▷ Same expressivity:

$$y \text{ is } x \text{ with } f \to z \quad \leftrightarrow \quad y \sim_{\{f\}} x \wedge y[f]z$$

$$x \sim_{\{f\}} y \quad \leftrightarrow \quad \exists z, v \cdot \left(\begin{array}{r} z \text{ is } x \text{ with } f \to v \\ \wedge z \text{ is } y \text{ with } f \to v \end{array} \right)$$

- ▷ Convenient to manipulate:
  - ▷ Equivalence relation for every F.
  - ▷ But also:

$$\begin{array}{ccc} x \sim_F y \wedge y \sim_G z & \to & x \sim_{F \cup G} z \\ x \sim_F y \wedge x \sim_G y & \leftrightarrow & x \sim_{F \cap G} y \end{array}$$

- ▷ Similar technique found in arrays. [Stump, Barrett, Dill, Levitt, 2001]

Our Contribution

Theorem

The first order theory of feature trees with update is decidable.

First Step: Existential Fragment

$$\exists x, z \cdot \left(y[f]x \wedge \neg(x \sim_{\{h,i\}} y) \wedge \cdots \right)$$

Existential quantification on the outside

Positive and negative literals

Conjunctive clause

Principle of the Algorithm

- We have a set of transformation rules $l \Rightarrow r$.

-
  ```
  function normalize(c: clause):
    while some rule r applies to c:
      c = apply r to c
    return c
  ```

- The rules are equivalences in our model.
- The system terminates.
- Irreducible forms have nice properties.
  - eg. they are either $\perp$ or satisfiable.

Examples of Rules

Associative commutative conjunction

Equivalences in our model

Replacement of z by y in c

Simplification: features

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

Quantifications (omitted when irrelevant)

(Not shown) side-conditions for termination

Clash: feature with absence

$$x[f]y \wedge x[f]\uparrow \wedge c \quad \Rightarrow \quad \bot$$

Propagation: feature $(f {\notin} F)$

$$x \sim_F y \wedge x[f]z \wedge c \quad \Rightarrow \quad x \sim_F y \wedge x[f]z \wedge y[f]z \wedge c$$

Satisfiability of Irreducible Clauses

Theorem

Every irreducible clause that is not $\bot$ is satisfiable.

- We need something stronger:

Lemma (Garbage collection)

Literals that do not talk about X

$$\exists X \cdot (g \wedge l)$$

Literals that mention at least one variable of X

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

Then

$$\mathcal{FT} \models (\exists X \cdot (g \wedge l)) \leftrightarrow g$$

First Order

$\forall \quad \exists \quad \wedge \quad \vee \quad \neg$

Quantifier Elimination

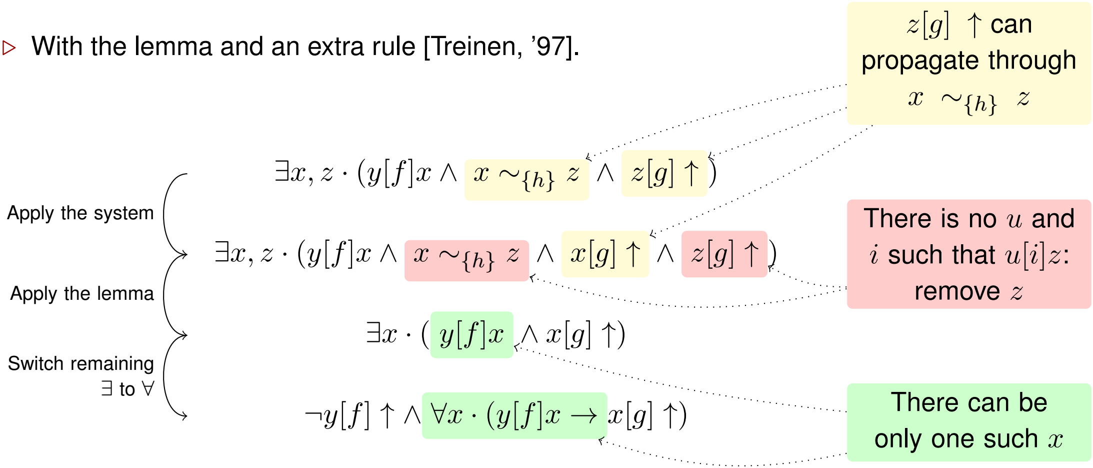
- **Problem:** our theory does not have the quantifier elimination property
- What is the meaning for y of:

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

- Two possible solutions:
  - Make the language richer [Presburger, '29]
    - with path constraints: $y[f][g] \uparrow$
    - potentially leads to complex simplification rules.
  - Weak Quantifier Elimination [Malc'ev, '71]
    - with a procedure: $\exists Y \cdot c \;\Rightarrow\; \forall Z \cdot d$
    - we can eliminate all the quantifier blocks except one.

Switching Quantifiers

- With the lemma and an extra rule [Treinen, ’97].



$z[g] \uparrow$ can propagate through $x \sim_{\{h\}} z$

$$\exists x, z \cdot (y[f]x \wedge x \sim_{\{h\}} z \wedge z[g] \uparrow)$$

Apply the system

$$\exists x, z \cdot (y[f]x \wedge x \sim_{\{h\}} z \wedge x[g] \uparrow \wedge z[g] \uparrow)$$

There is no u and i such that $u[i]z$: remove z

Apply the lemma

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

Switch remaining $\exists$ to $\forall$

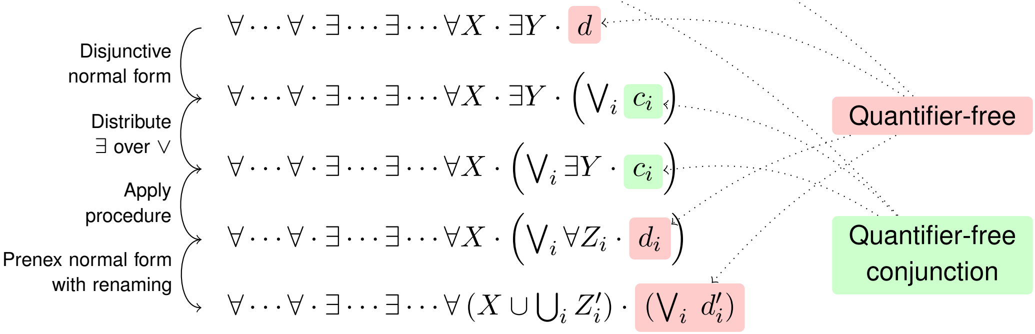
$$\neg y[f] \uparrow \wedge \forall x \cdot (y[f]x \rightarrow x[g] \uparrow)$$

There can be only one such x

- We can turn all $\exists$ into $\forall$ which allows us to go for Weak Quantifier Elimination.

Weak Quantifier Elimination [Malc'ev, '71]

- With a procedure: $\exists Y \cdot \boxed{c} \;\Rightarrow\; \forall Z \cdot \boxed{d}$



$$\forall \cdots \forall \cdot \exists \cdots \exists \cdots \forall X \cdot \exists Y \cdot \boxed{d}$$

Disjunctive normal form

$$\forall \cdots \forall \cdot \exists \cdots \exists \cdots \forall X \cdot \exists Y \cdot \Big(\bigvee_i \boxed{c_i}\Big)$$

Distribute $\exists$ over $\vee$

$$\forall \cdots \forall \cdot \exists \cdots \exists \cdots \forall X \cdot \Big(\bigvee_i \exists Y \cdot \boxed{c_i}\Big)$$

Apply procedure

$$\forall \cdots \forall \cdot \exists \cdots \exists \cdots \forall X \cdot \Big(\bigvee_i \forall Z_i \cdot \boxed{d_i}\Big)$$

Prenex normal form with renaming

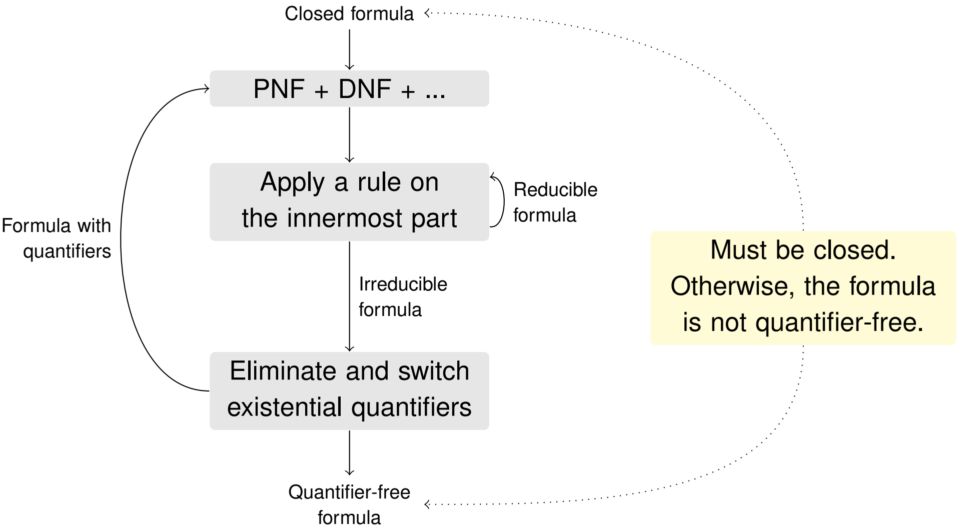
$$\forall \cdots \forall \cdot \exists \cdots \exists \cdots \forall \big(X \cup \textstyle\bigcup_i Z_i'\big) \cdot \boxed{\big(\bigvee_i \; d_i'\big)}$$

Quantifier-free

Quantifier-free conjunction

- Eliminate one quantifier alternation at a time.

Full Procedure



Closed formula

PNF + DNF + ...

Apply a rule on the innermost part

Reducible formula

Irreducible formula

Formula with quantifiers

Eliminate and switch existential quantifiers

Quantifier-free formula

Must be closed. Otherwise, the formula is not quantifier-free.

Conclusion

- Contribution:
  - Feature tree with update.
  - Decidability of first order theory.

Theorem

The first order theory of feature trees with update is decidable.

  - Procedure parametrized by a theory of node decorations.
  - Complexity: non-elementary lower bound. [Vorobyov, ’96]
- Perspectives:
  - Implementation.
  - Efficient implementation of a smaller fragment.
  - Symbolic execution of Shell scripts.
  - “Correctness of Linux Scripts” (`http://colis.irif.fr`).