How to build a (resolution) prover?

Andreas Meier

Saarland University, Saarbrücken, Germany

Proof Search by Saturation

Given: Set of clauses

+ Inferences (resolution, factorization, paramodulation)

⇒ Saturate clause set with all possible inferences

1. if empty clause in clause set, then terminate
2. select clauses
3. apply inferences to selected clauses
4. add result to clause set; goto 1

Problem?

Dealing with millions of clauses . . .

Efficient automated theorem prover by

good theory
\+ efficient implementation
\+ clever heuristics

References: [Voronkov, IJCAR, 2001]

Theory

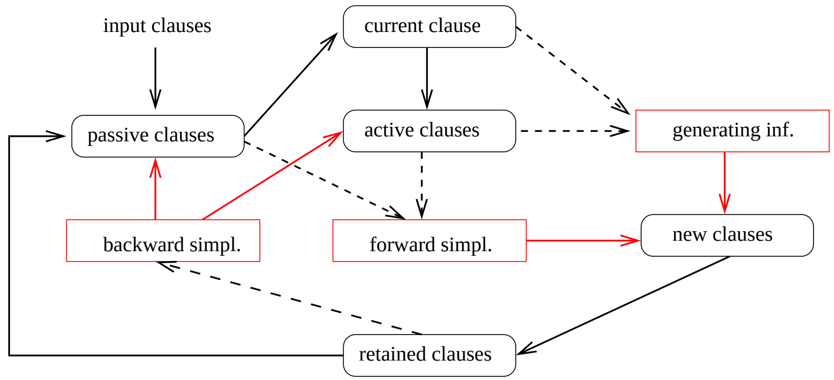
Progress in the theory of resolution-based systems:

- reduction of possible inferences without impair of completeness (e.g., superposition calculus)
- decision procedures for some fragments (e.g., realized in BLIKSEM)

References: [Bachmair+Ganzinger, Handbook of Aut. Reas. I]
[Nieuwenhuis+Rubio, Handbook of Aut. Reas. I]
[Bachmair+Ganzinger, Diverse CADE,LPAR]

Implementation

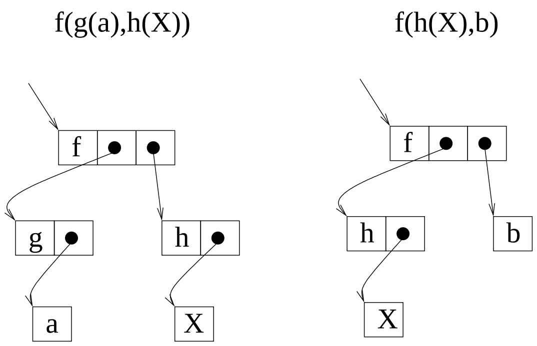
Organization



References: [McCune, OTTER 3.0 Manual, 1994]

Implementation

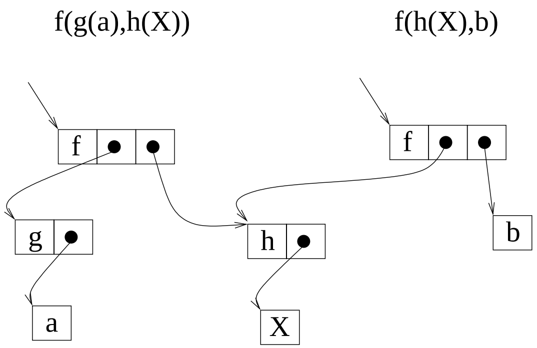
Efficient Data Structures



tree representation

Implementation

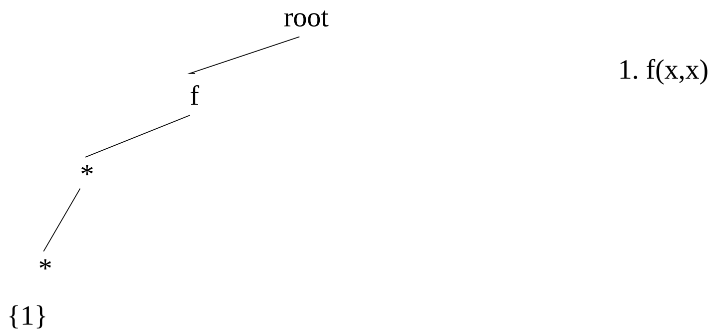
Efficient Data Structures



shared tree-like representation

Implementation

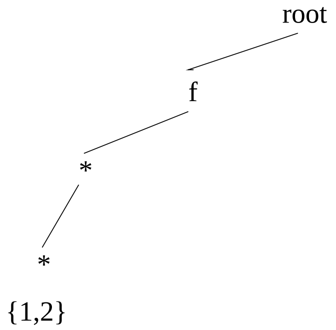
Efficient Data Structures



discrimination tree index

Implementation

Efficient Data Structures

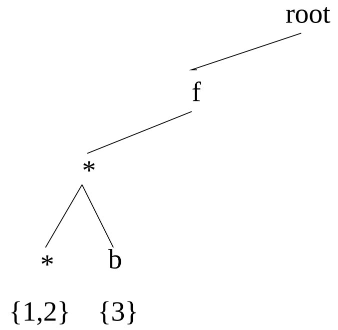


1. f(x,x)
2. f(x,y)

discrimination tree index

Implementation

Efficient Data Structures

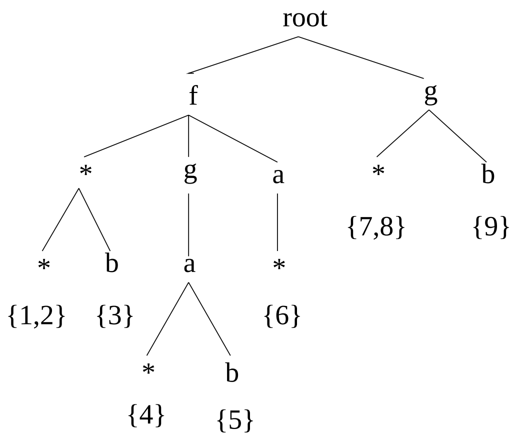


1. f(x,x)
2. f(x,y)
3. f(x,b)

discrimination tree index

Implementation

Efficient Data Structures

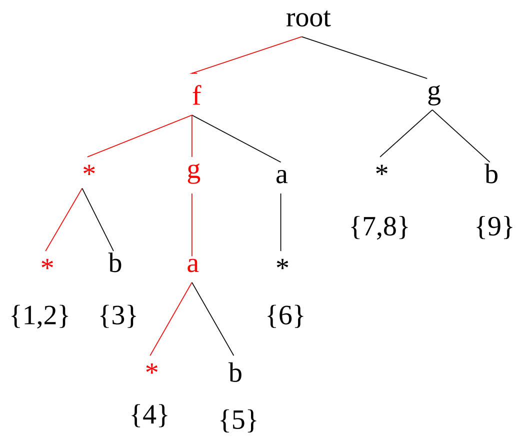


1. f(x,x)
2. f(x,y)
3. f(x,b)
4. f(g(a),x)
5. f(g(a),b)
6. f(a,y)
7. g(x)
8. g(z)
9. g(b)

discrimination tree index

Implementation

Efficient Data Structures



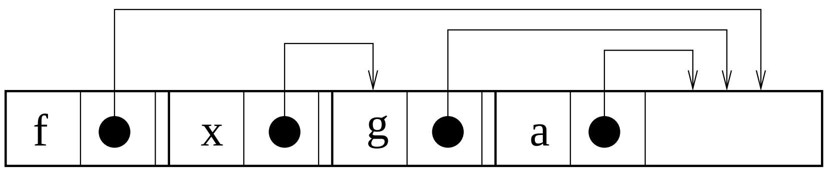
1. f(x,x)
2. f(x,y)
3. f(x,b)
4. f(g(a),x)
5. f(g(a),b)
6. f(a,y)
7. g(x)
8. g(z)
9. g(b)

retrieving variants of: $f(g(x), a)$

Implementation

Efficient Data Structures

f(x,g(a))



array-based flat-term representation

Implementation

Efficient Data Structures

References: [Graf, LNCS 1053, 1996]

[McCune, JAR, 1992]

[Ramakrishnan+Sekar+Voronkov,
Handbook of Aut. Reas. II]

Control

parameterized algorithms

(e.g., 32 parameters to determine OTTER's main algorithm)

⇒ selection of the "right" parameterization is crucial

either by the user

or automatically by the system

For instance:

- OTTER: auto mode
- VAMPIRE: preprocessor
- WALDMEISTER: self-adaption component

Control

parameterized algorithms

(e.g., 32 parameters to determine OTTER's main algorithm)

selection of the "right" parameterization is crucial

OR: try different instances competitively

- RCTHEO randomized decision points in SETHEO
  [Ertel, LPAR, 1992]
- SICOTHEO pre-defined strategies in SETHEO
  [Schumann, 1995]