Implementation of Lambda-Free Higher-Order Superposition

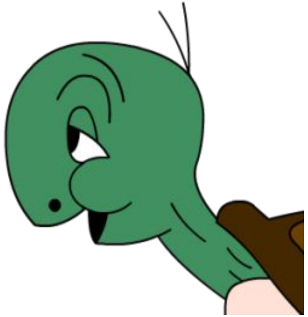
Petar Vukmirović

VRIJE UNIVERSITEIT AMSTERDAM

Automatic theorem proving – state of the art

FOL

HOL

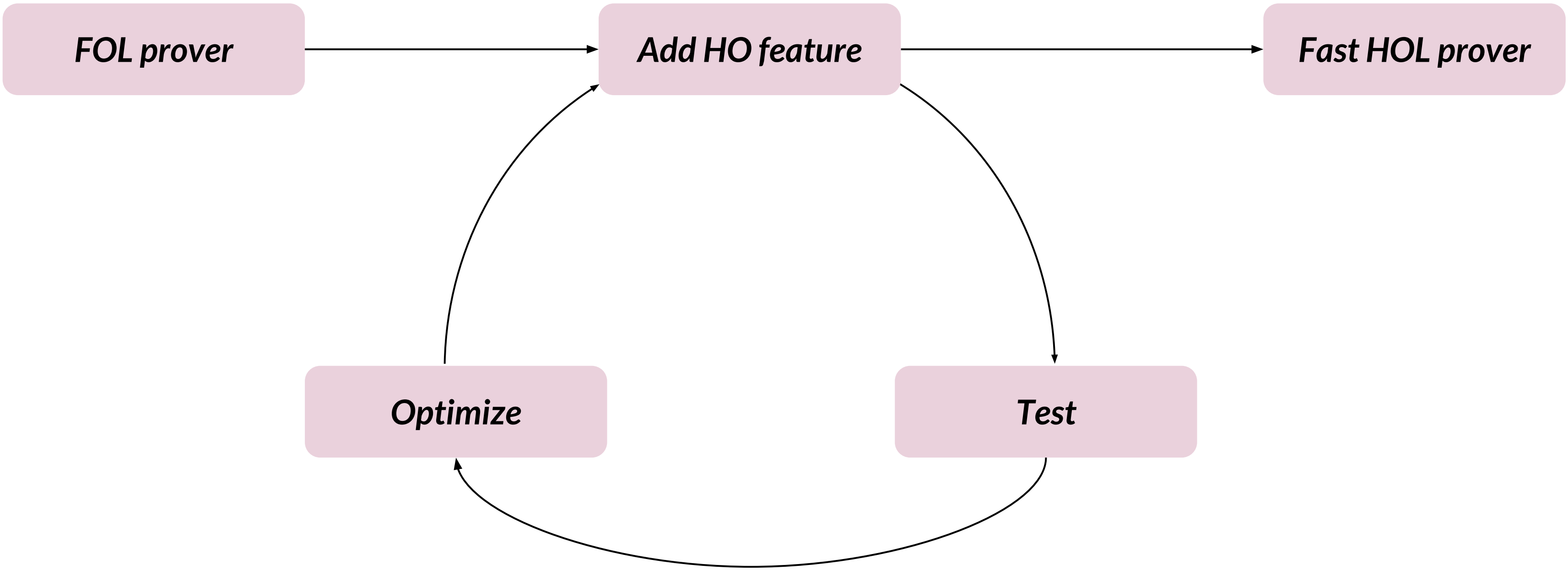


Automatic theorem proving – challenge

HOL

High-performance higher-order theorem prover
*that extends first-order theorem proving **gracefully**.*

My approach



Syntax

Types:

$$\tau ::= a$$
$$\mid \tau \rightarrow \tau$$

Terms:

$t ::= X$	variable
$\mid f$	symbol
$\mid t\ t$	application

Supported HO features

Example:

$$X\,(f\,a)\,f$$

Applied variable

Partial application

Applied variables

+

Partial application

=

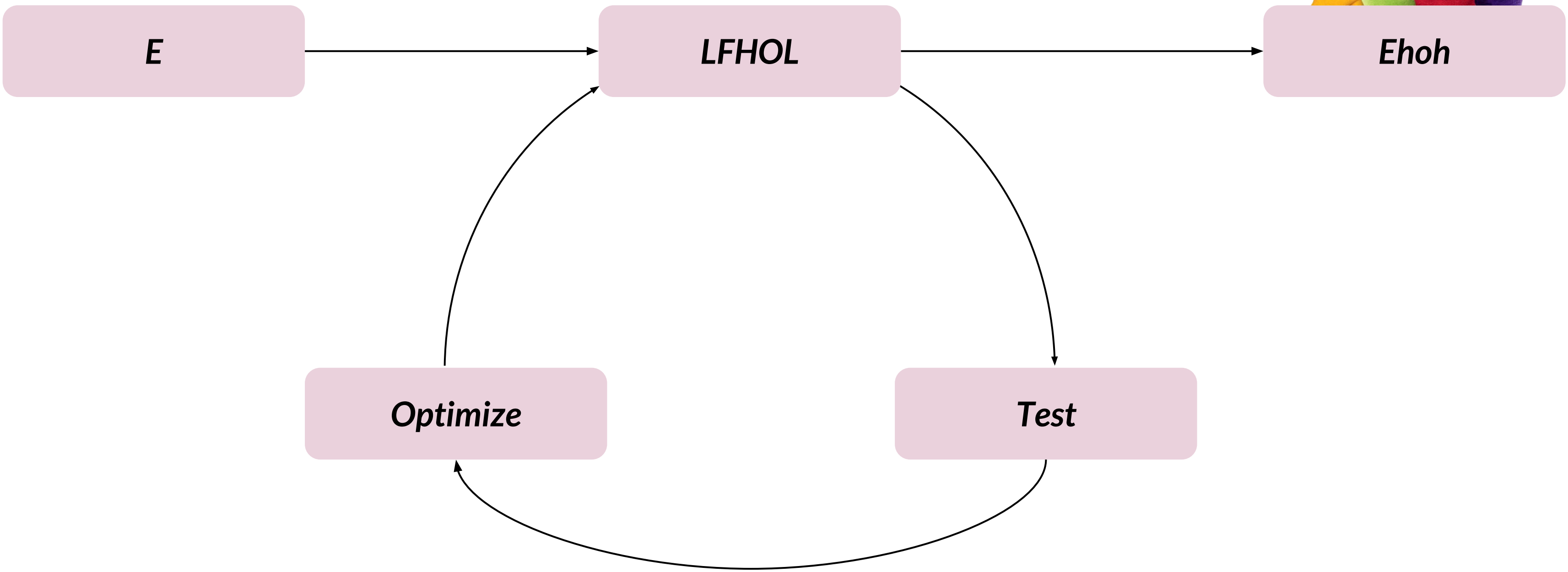
Lambda-Free

Higher-Order Logic

$$map\ F\ nil = nil$$

$$map\ F\ (cons\ x\ xs) = cons\ (F\ x)\ (map\ F\ xs)$$

LFHOL iteration



Generalization of term representation

Approach 1:
Native representation

$$X\ (f\ a)\ f$$

Approach 2:
Applicative encoding

$$@(@(X, @(f, a)), f)$$

Differences between the approaches

Approach 1:
Native representation

- Compact
- Fast
- Keeps metainformation

Approach 2:
Applicative encoding

- Easy to implement

Unification problem

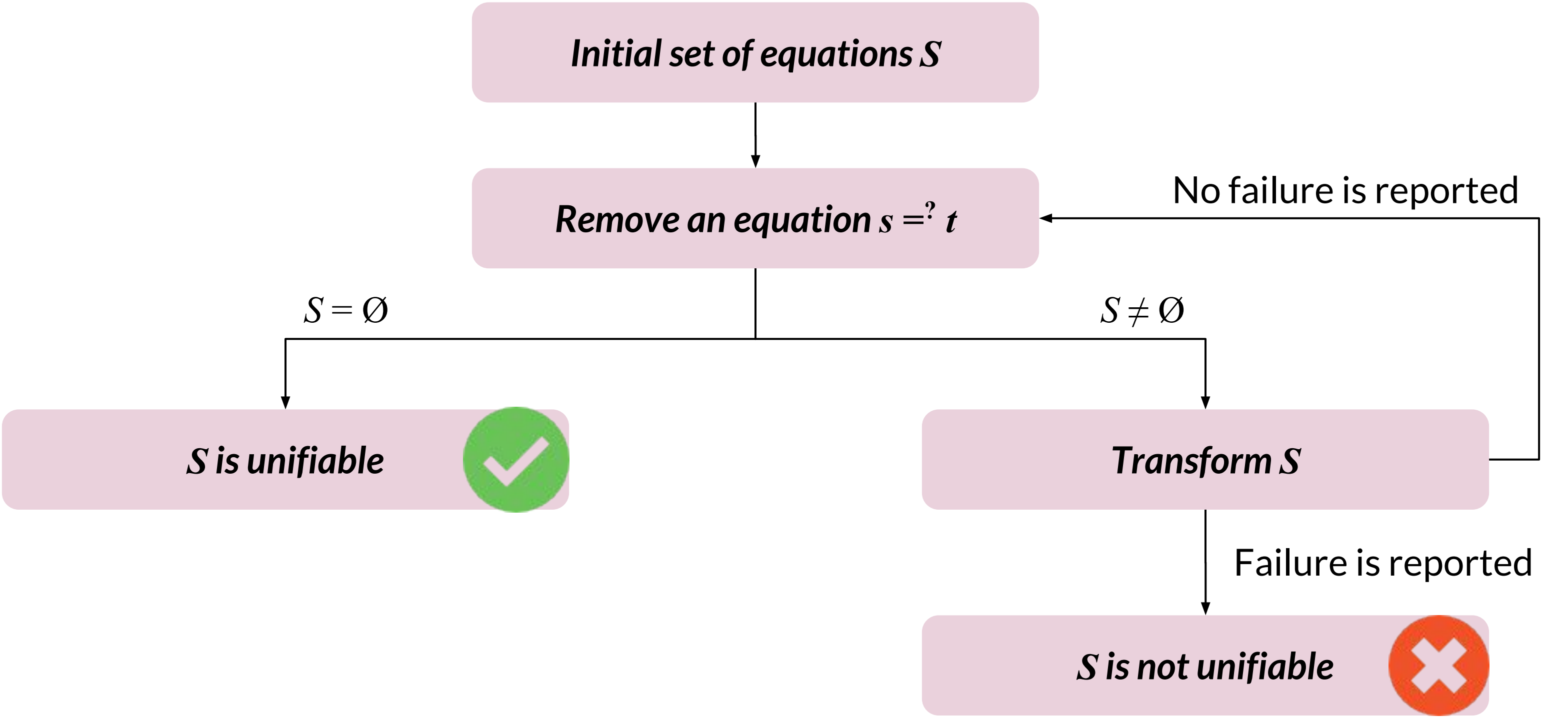
Given the set of equations

$$\{ \, s_1 =^? t_1, \, \ldots, \, s_n =^? t_n \, \}$$

find the substitution σ such that

$$\{ \, \sigma(s_1) = \sigma(t_1), \, \ldots, \, \sigma(s_n) = \sigma(t_n) \, \}$$

FOL unification algorithm



Transformation of the equation set

Case $s =^? t$		
$f(s_1, \ldots, s_m) =^? f(t_1, \ldots, t_m)$	*Add* $\{ s_1 =^? t_1, \ldots, s_n =^? t_n \}$	decomposition
$f(s_1, \ldots, s_m) =^? g(t_1, \ldots, t_n)$	*Report failure*	collision
$f(s_1, \ldots, s_m) =^? X$	*Add* $\{ t =^? s \}$	reorientation
$X =^? t$; X not in t	*Apply* $[X \leftarrow t]$	application
$X =^? f(s_1, \ldots, s_m)$; X in t	*Report failure*	occurs-check
$X =^? X$	*No changes*	identity

FOL algorithm fails on LFHOL terms

$X\, b =^? f\, a\, b$

$$X\, b =^? f\, a\, b \xrightarrow{X \neq f} \textit{Report failure}$$

Report failure collision

Yet, $\{\, X \leftarrow f\, a \,\}$ is a unifier.

Example

$$X\,(Z\,b\,c)\,d \;=^?\; f\,a\,(Y\,c)\,d \xrightarrow[\text{prefix match}]{X \leftarrow f\,a} Z\,b\,c =^? Y\,c,\; d =^? d$$

orientation

$$Y\,c =^? Z\,b\,c,\; d =^? d \xrightarrow[\text{prefix match}]{Y \leftarrow Z\,b} c =^? c,\; d =^? d$$

decomposition

$$d =^? d \xrightarrow[\text{decomposition}]{} \text{Unifier } \{\, X \leftarrow f\,a,\; Y \leftarrow Z\,b \,\}$$

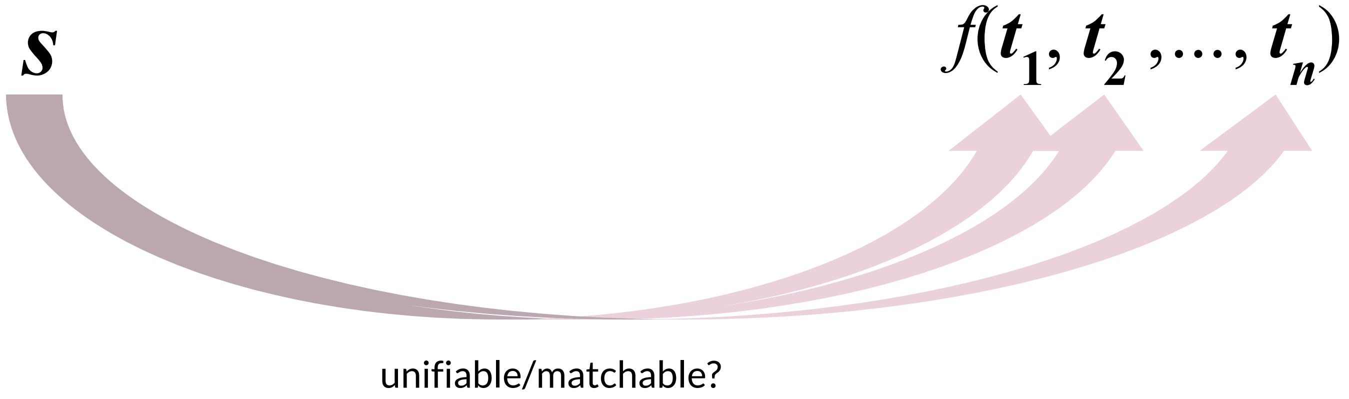
LFHOL equation set transformation

Case $s =^? t$			
$\alpha\, s_1 \ldots s_n =^? \alpha\, t_1 \ldots t_n$		*Add* $\{ s_1 =^? t_1, \ldots, s_n =^? t_n \}$	decomposition
$\alpha\, s_1 \ldots s_m =^? \beta\, t_1 \ldots t_n$	β is var, either α is not or $m > n$	*Add* $\{ t =^? s \}$	reorientation
	Neither α nor β vars	*Report failure*	collision
	α is var, matches prefix of t	*Add* $\{\alpha =^? t[:p], s_1 =^? t_{p+1}, \ldots, s_m =^? t_n\}$	prefix match
$X =^? t$; X not in t		*Apply* $[X \leftarrow t]$	application
$X =^? \alpha\, s_1 \ldots s_n$; X in t		*Report failure*	occurs-check
$X =^? X$		*No changes*	identity

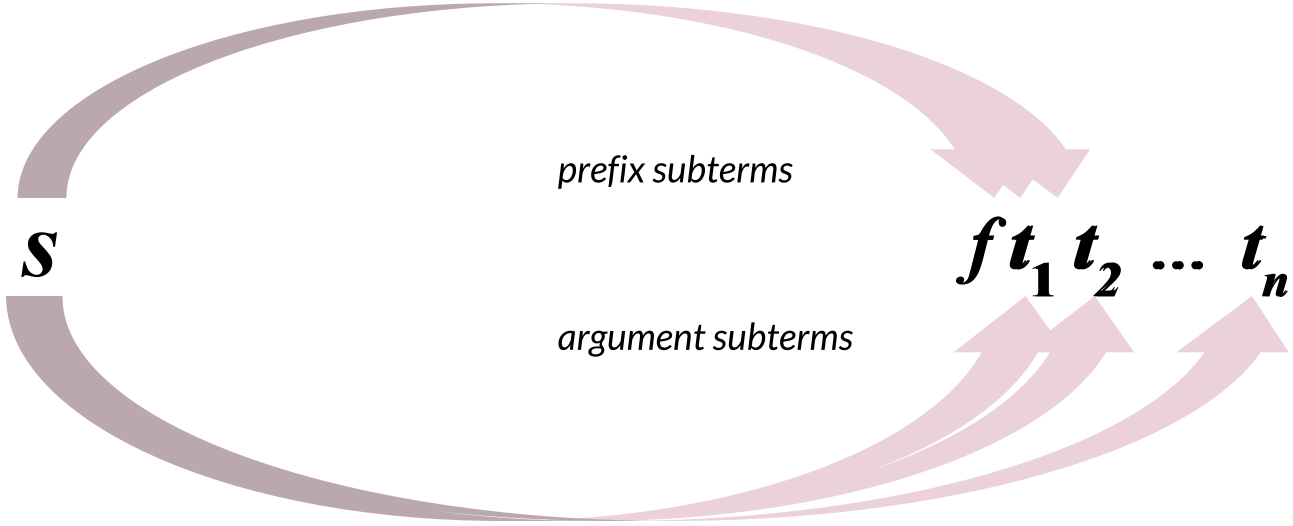
Standard FOL operations

$$\begin{tikzcd} s \arrow[r, bend right, "\text{unifiable/matchable?}"'] & t \end{tikzcd}$$

… are performed on subterms recursively,



… and there are twice as many subterms in HOL



unifiable/matchable?

Prefix optimization

- Traverse only argument subterms
- Use types & arity to determine the only unifiable/matchable prefix

Report 1 argument trailing

$$f\ X\ Y \qquad\qquad\qquad \underline{f\ a\ b}\ c$$

Advantages of prefix optimization

2x fewer subterms

No unnecessary prefixes created

No changes to E term traversal

Indexing data structures

$$f(x,g(h(y),a))$$

Query term

Generalizations
$s =^? \sigma(t)$

Instances
$\sigma(s) =^? t$

Unifiable terms
$\sigma(s) =^? \sigma(t)$

$f(a,g(b,a))$

c

$f(x,y)$

$h(g(x,f(x,x)))$

a

x

$f(f(x,x), f(y,y))$

Set of terms

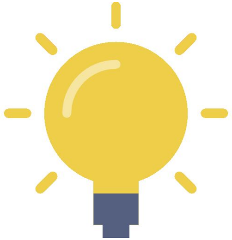
E’s indexing data structures

Discrimination trees

Fingerprint indexing

Feature vector indexing

Discrimination trees



Factor out operations common for many terms

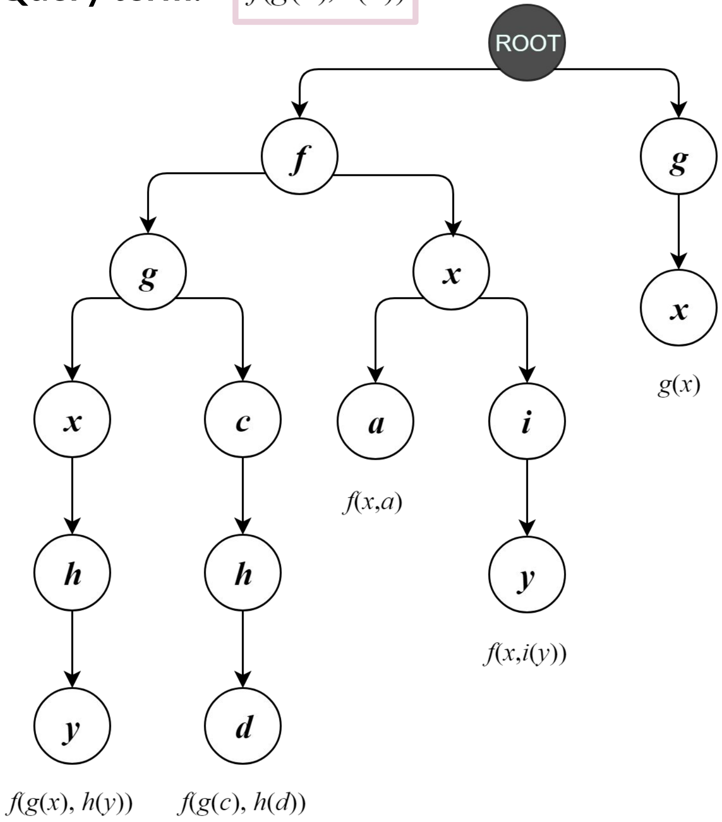
Flatten the term and use it as a key

Query term: $f(x, f(h(x), y))$

Flattening: f x f h x y

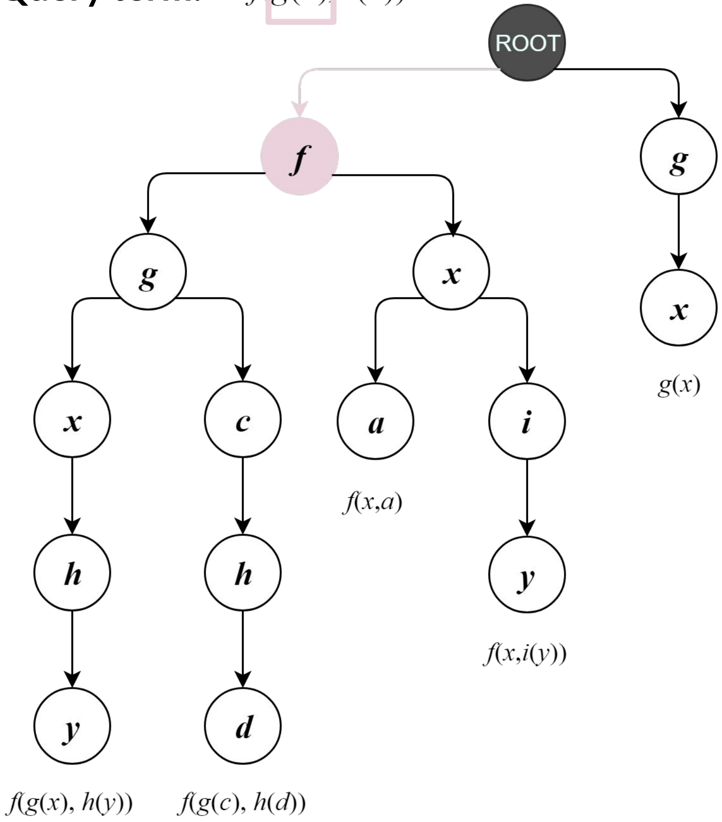
Example

Query term: $f(g(a),\ i(a))$



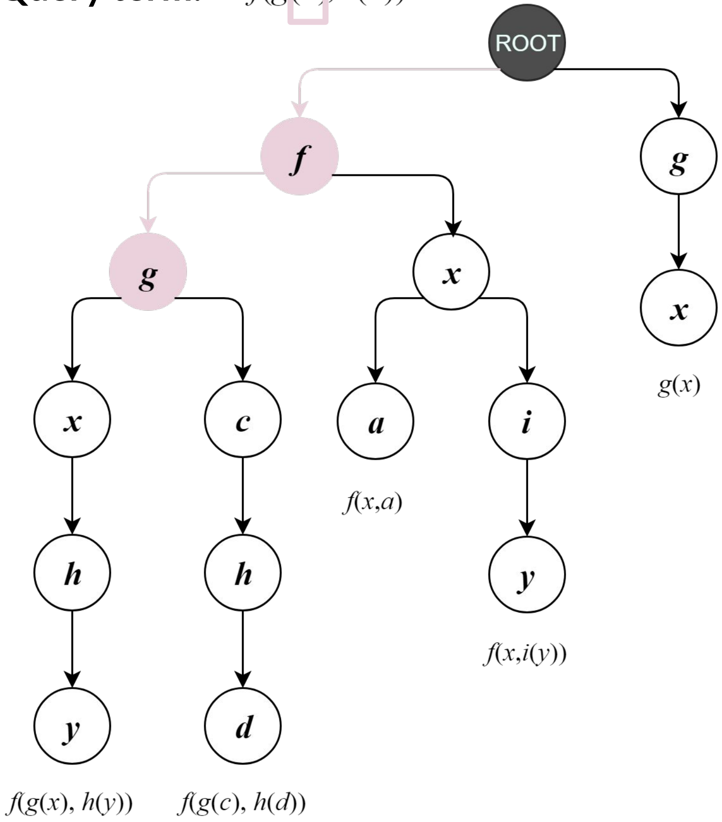
Example

Query term: $f(\boxed{g(a)}, i(a))$



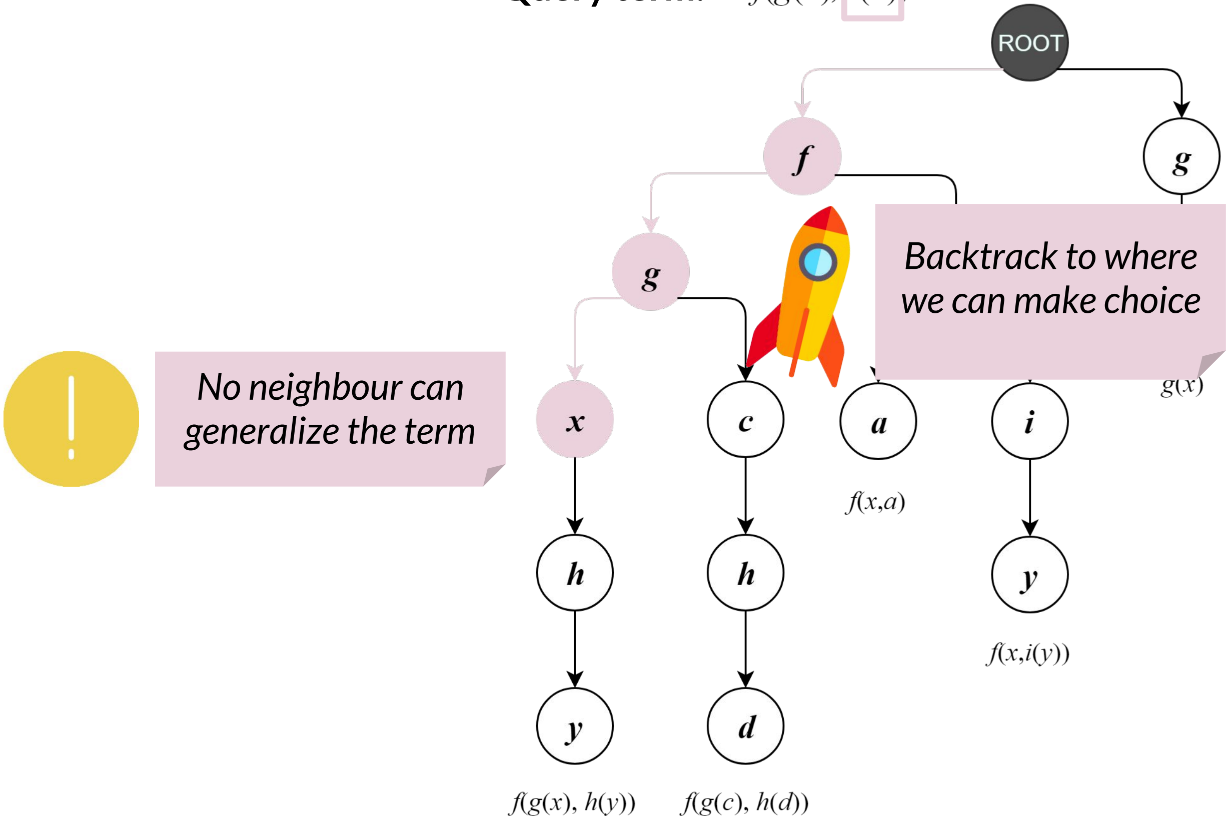
Example

Query term: $f(g(\boxed{a}), i(a))$



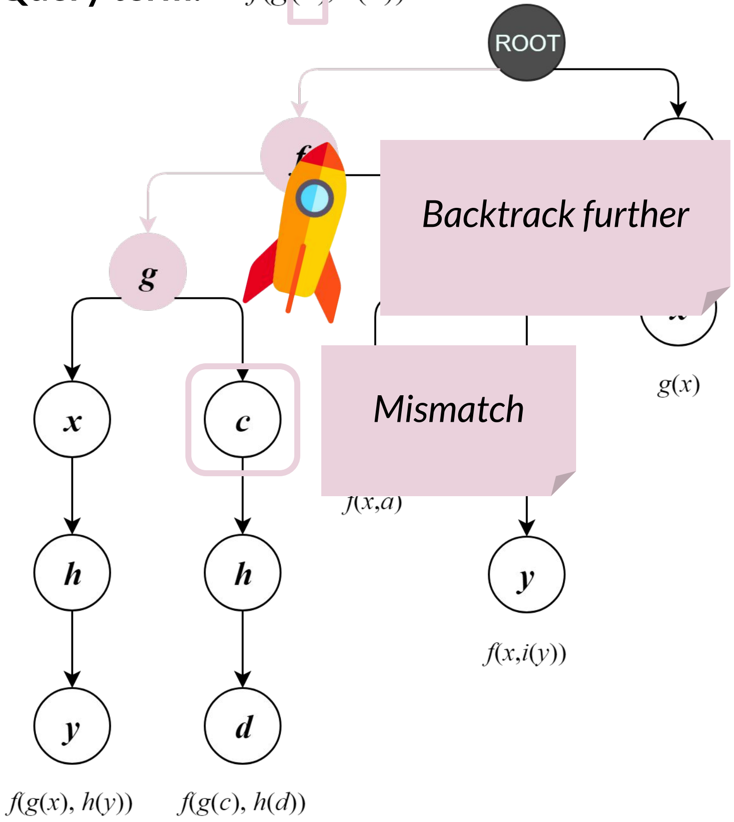
Example

Query term: $f(g(a), \boxed{i(a)})$



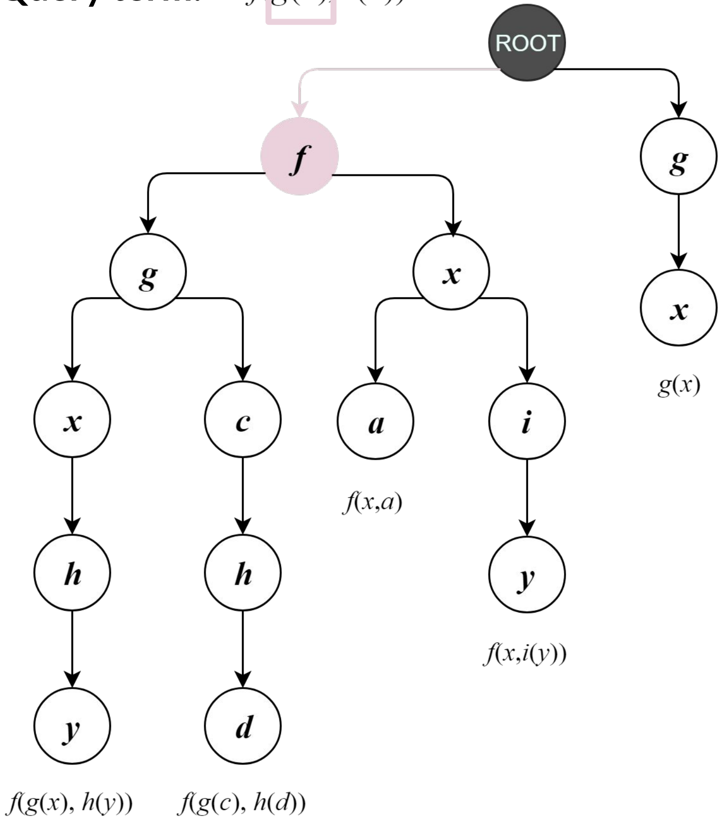
Example

Query term: $f(g(\boxed{a}), i(a))$



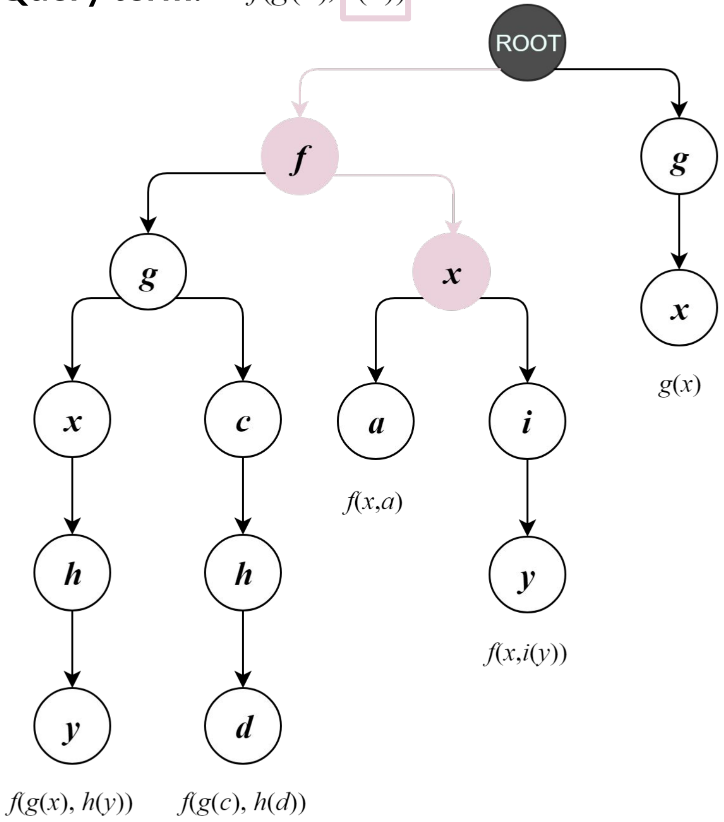
Example

Query term: $f(\boxed{g(a)}, i(a))$



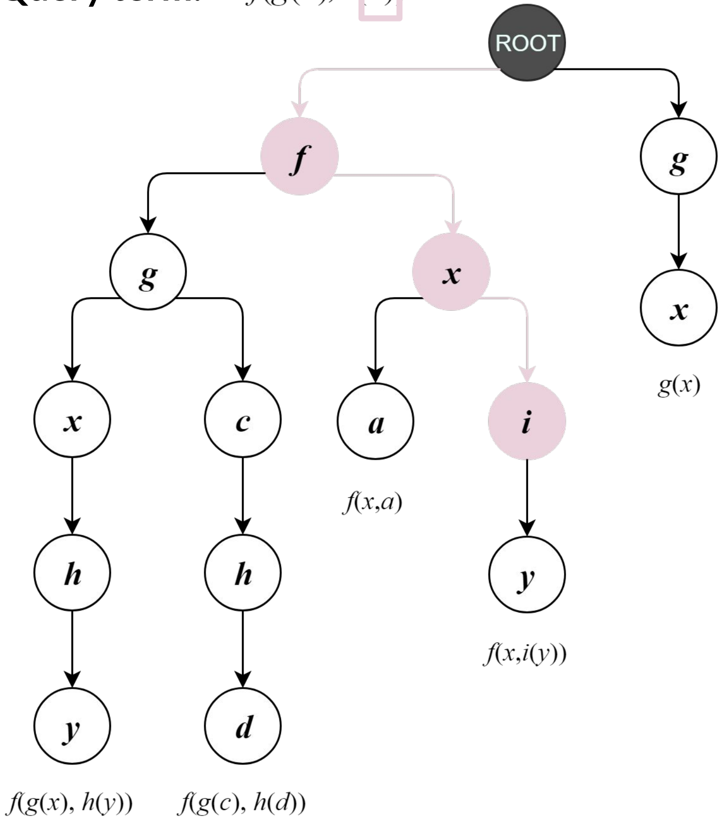
Example

Query term: $f(g(a), \boxed{i(a)})$



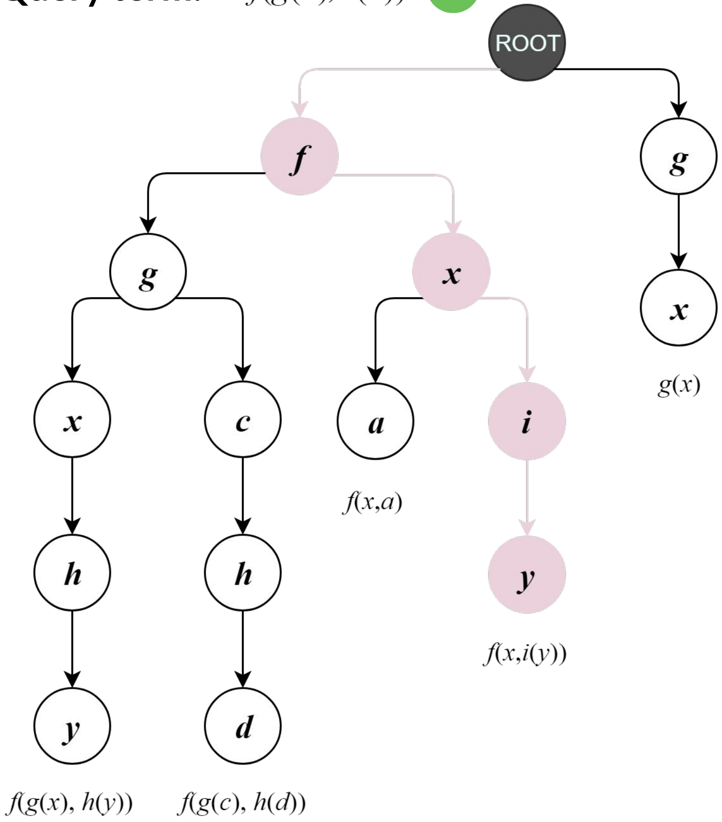
Example

Query term: $f(g(a), i(\boxed{a}))$

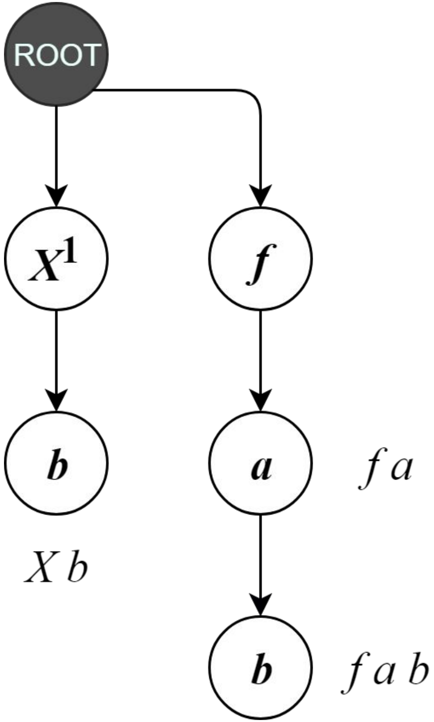


Example

Query term: $f(g(a),\ i(a))$

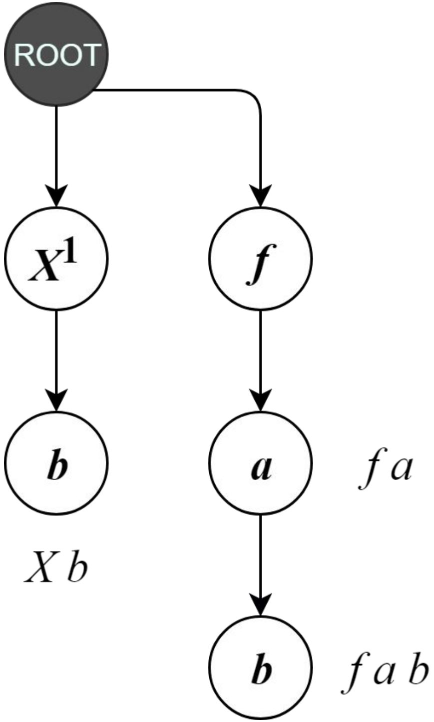


LFHOL challenges



1. Applied variables
2. Terms prefixes of one another
3. Prefix optimization

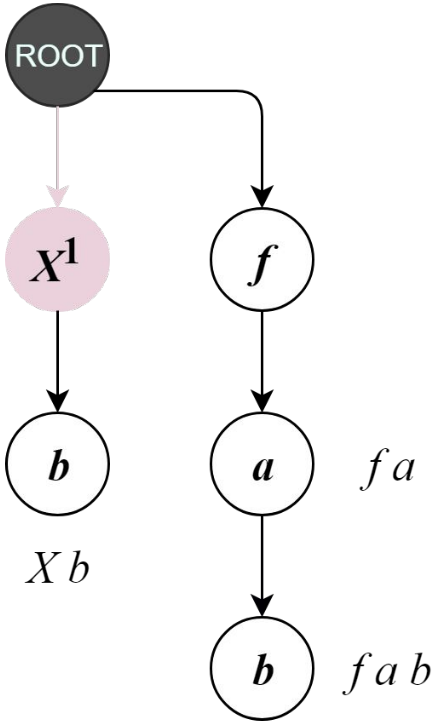
LFHOL challenges



Query term: $\boxed{g\ a\ b}$

1. ***Applied variables***
   Variable can match a prefix
2. Terms prefixes of one another
3. Prefix optimization

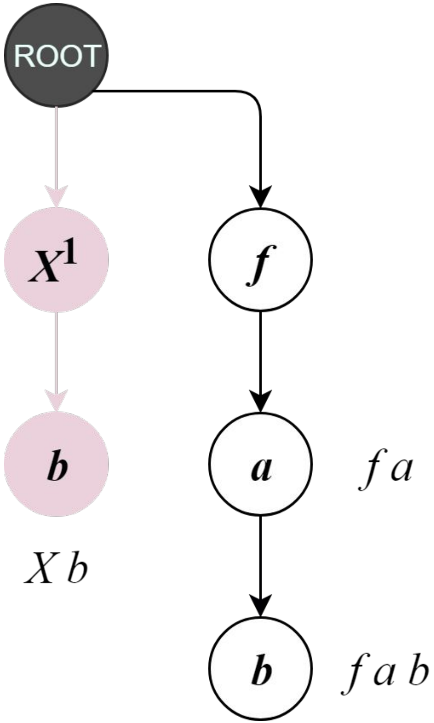
LFHOL challenges



Query term: $g\ a\ \boxed{b}$

1. ***Applied variables***
   Variable can match a prefix
2. Terms prefixes of one another
3. Prefix optimization

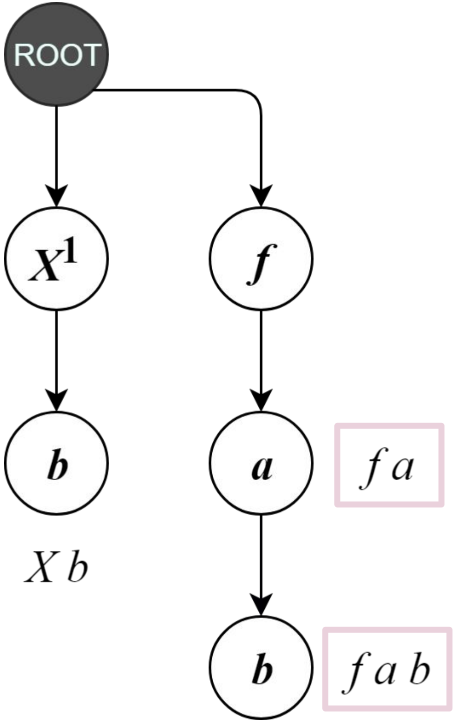
LFHOL challenges



Query term: $g\ a\ b$

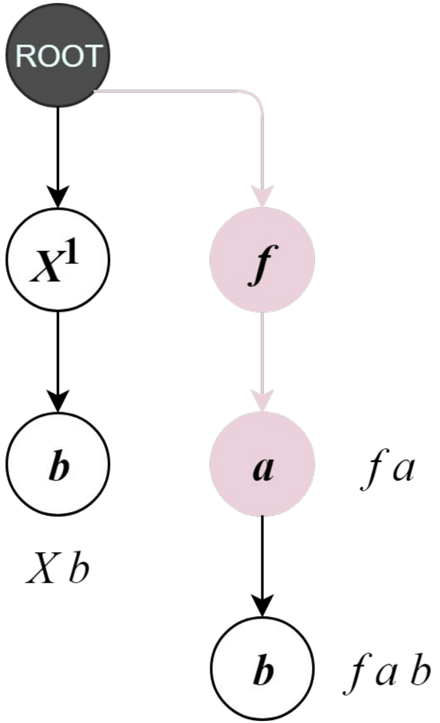
1. ***Applied variables***
   Variable can match a prefix
2. Terms prefixes of one another
3. Prefix optimization

LFHOL challenges



1. Applied variables
2. ***Terms prefixes of one another***
   Terms can be stored in inner nodes
3. Prefix optimization

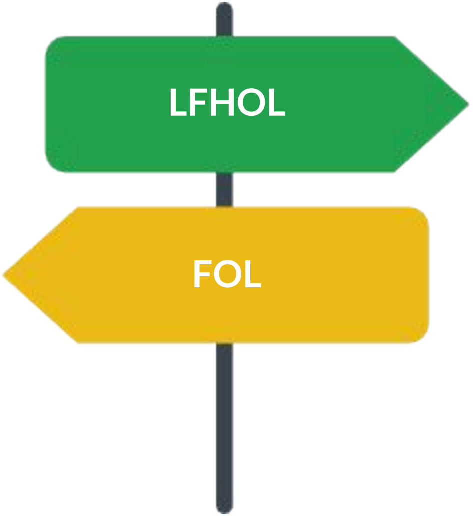
LFHOL challenges



Query term: $f\ a\ \boxed{b}$

1. Applied variables
2. Terms prefixes of one another
3. ***Prefix optimization***
   Prefix matches are allowed

Experimentation results

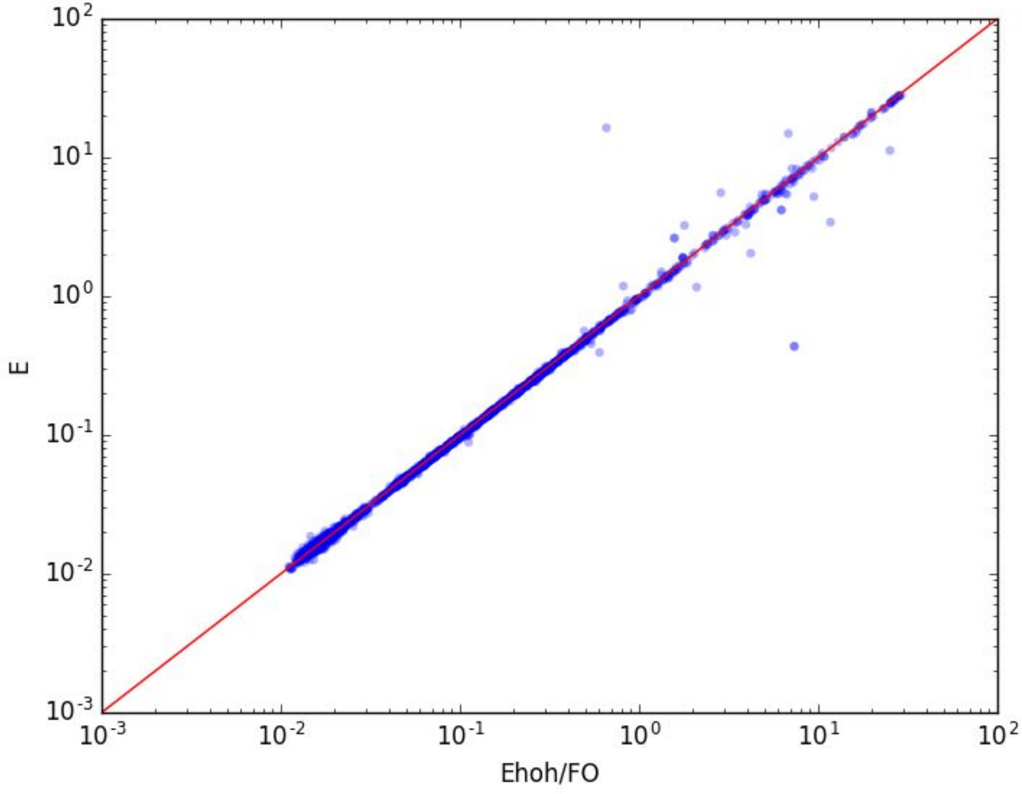


Two compilation modes:

Ehoh- support for LFHOL

Ehoh/FO - support only for FOL

Overhead: Ehoh/FO over E

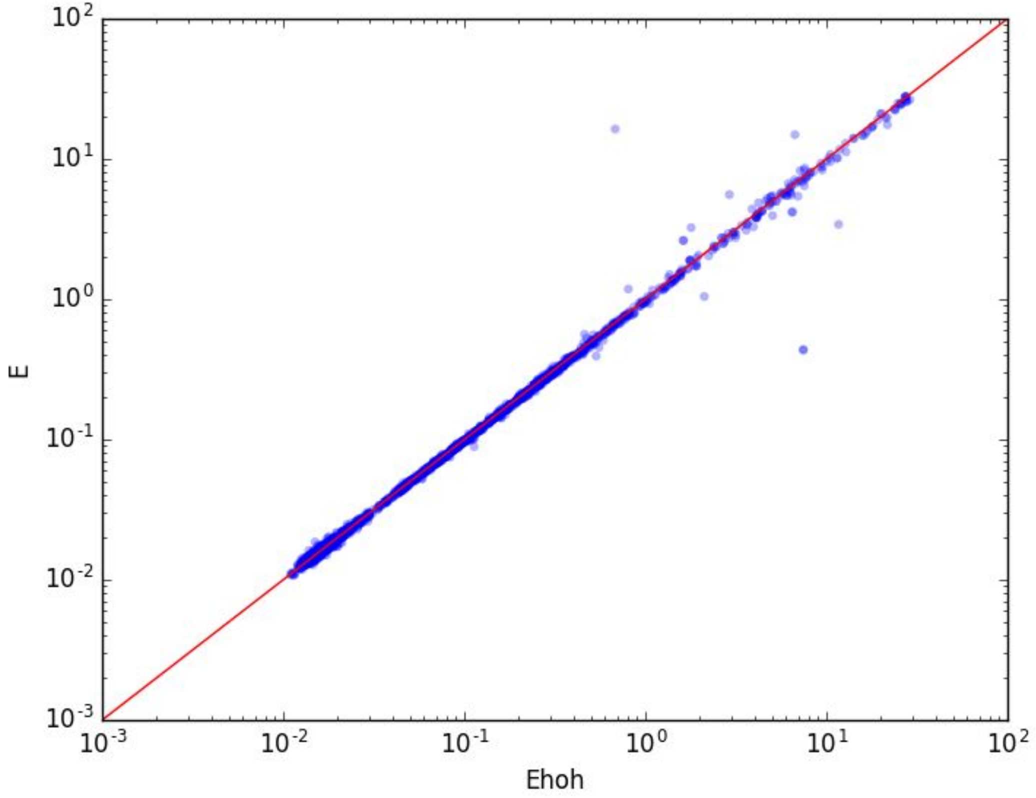


Sledgehammer
first-order benchmark
5012 TFF problems

Ehoh/FO solved 3 problems less

2.6% time overhead on average

Overhead: Ehoh over E

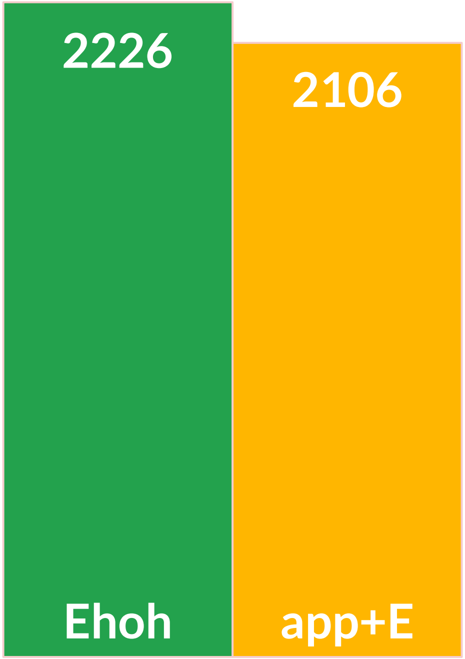


Sledgehammer
first-order benchmark
5012 TFF problems

Ehoh solved 4
problems less

3.1% time overhead on
average

Sledgehammer benchmarks

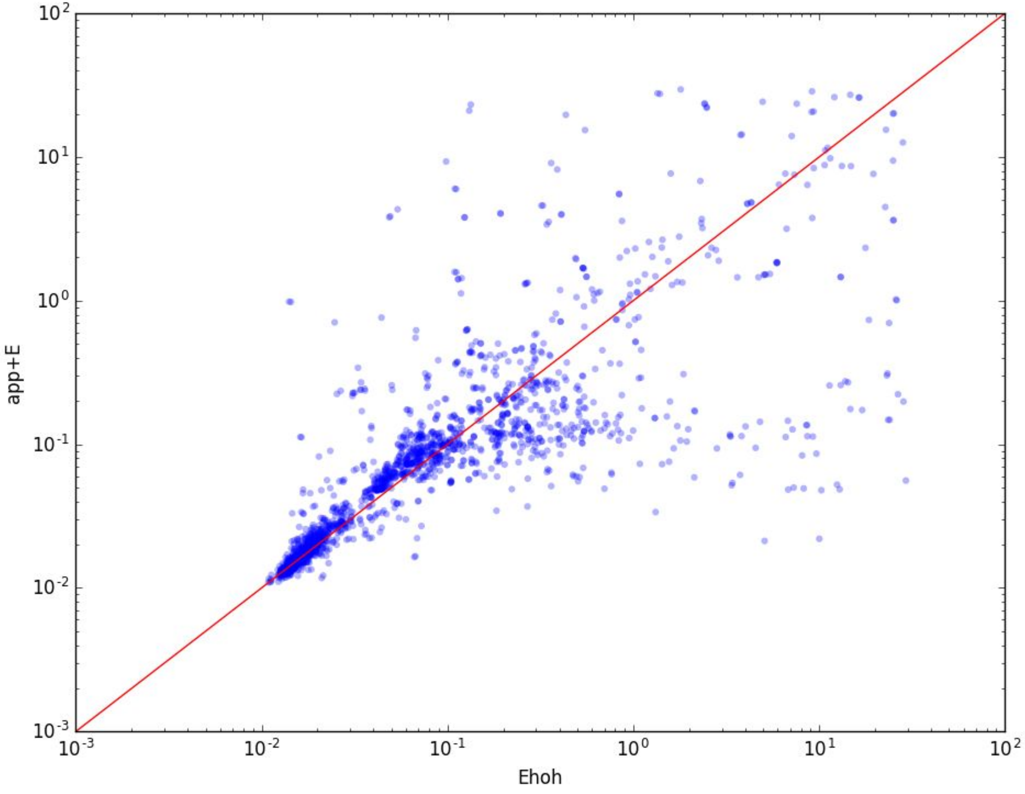


5012 problems in THF format

Partial application and applied variables are **not** encoded

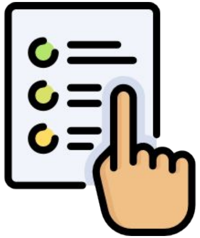
30s timeout

Sledgehammer benchmarks



Performance improvements

Take symbol type into account for symbol weight or precedence generation



Prefer clauses that have no applied variables

Implemented, but full evaluation is pending

Summary

Engineering viewpoint

- New type module
- Native term representation
- Elegant algorithm extensions
- Prefix optimizations

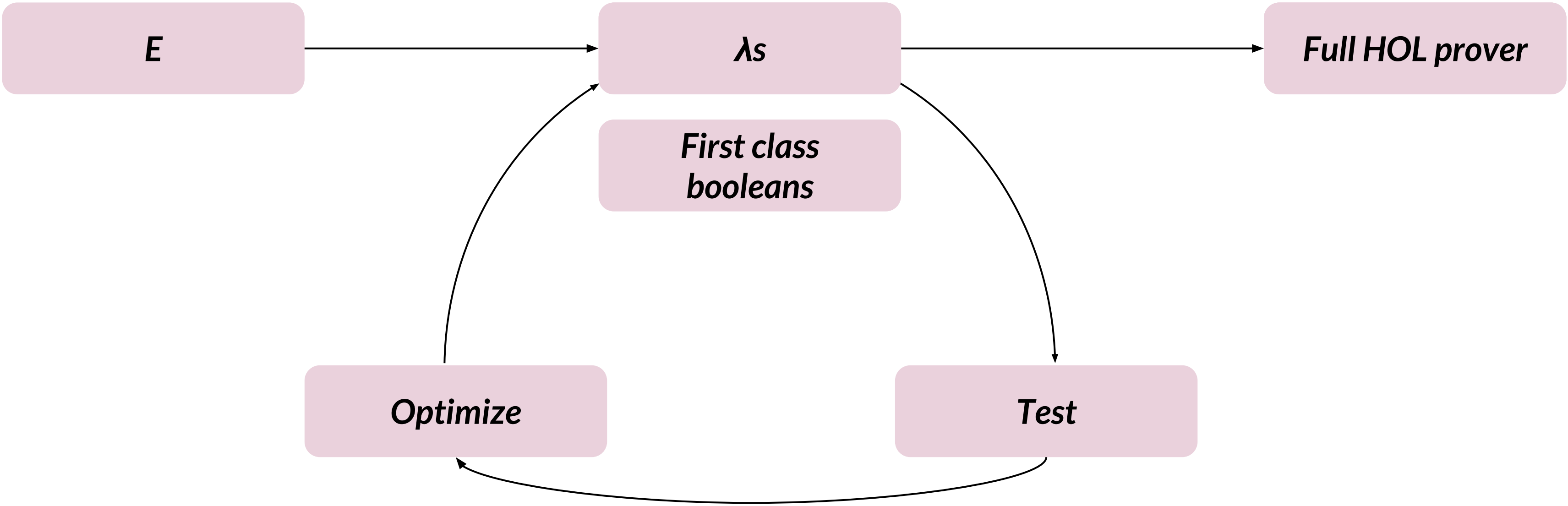
Theoretical viewpoint

- Graceful algorithm extension
- Graceful data structures extension

Future work

Integration with official E

New features



Implementation of Lambda-Free Higher-Order Superposition

Petar Vukmirović

VRIJE UNIVERSITEIT AMSTERDAM