Implementation of Lambda-Free Higher-Order Superposition

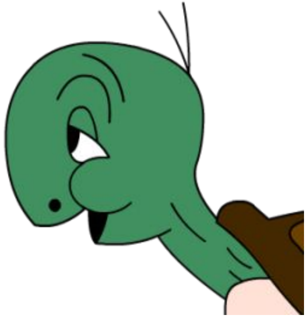
Petar Vukmirović

VRIJE UNIVERSITEIT AMSTERDAM

Automatic theorem proving – state of the art

FOL

HOL

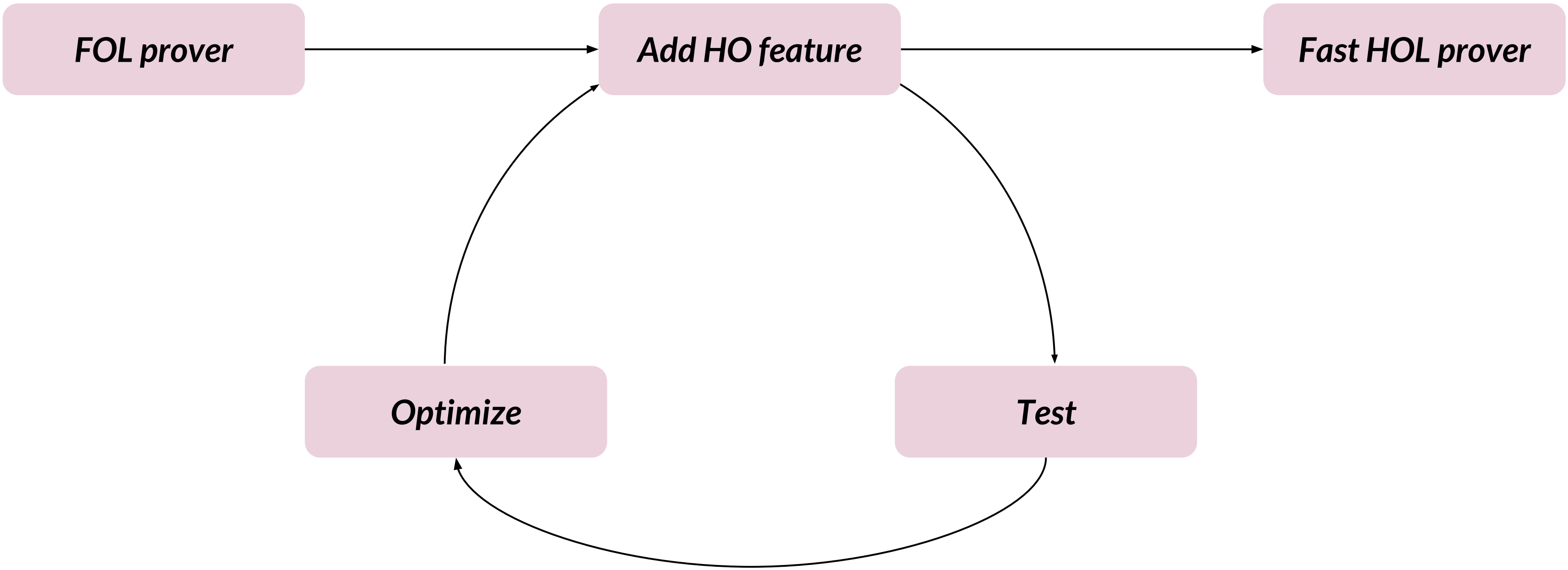


Automatic theorem proving – challenge

HOL

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

My approach



Syntax

Types:

$$\tau ::= a \mid \tau \to \tau$$

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

+

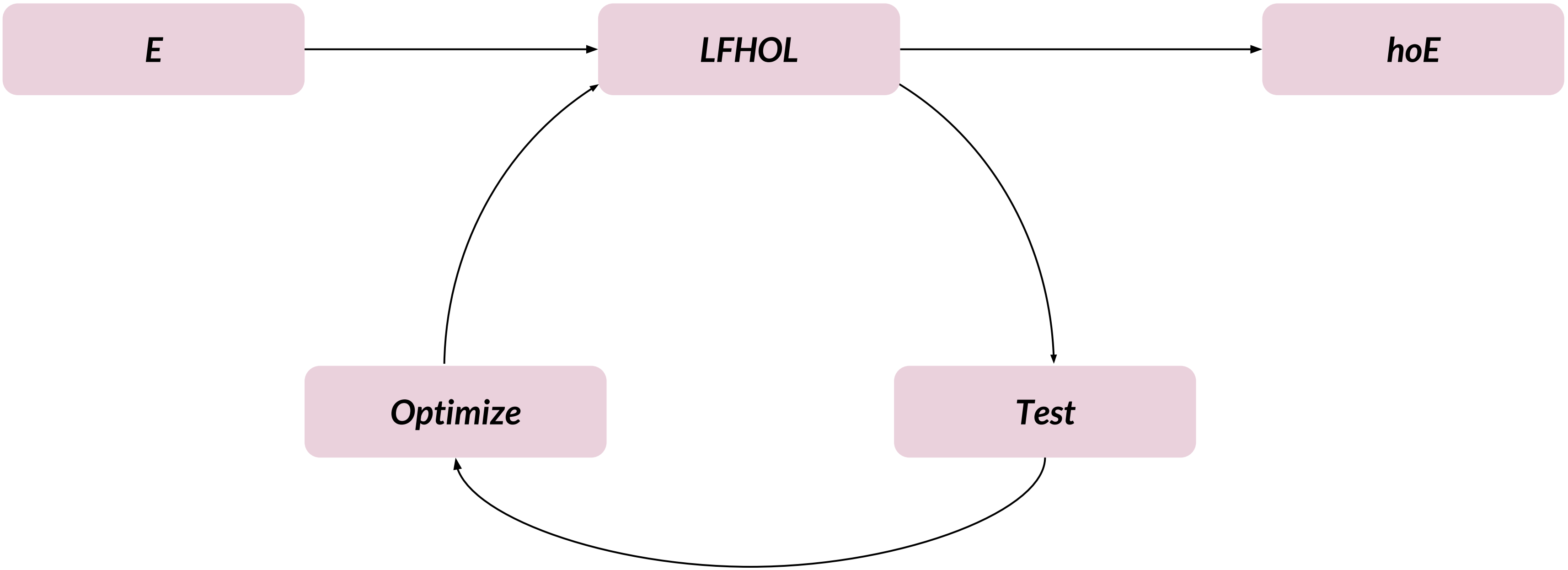
Partial application

=

Lambda-Free

Higher-Order Logic

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
- ✓ Works well with E heuristics

Approach 2:
Applicative encoding

- ✓ Easy to implement

Unification problem

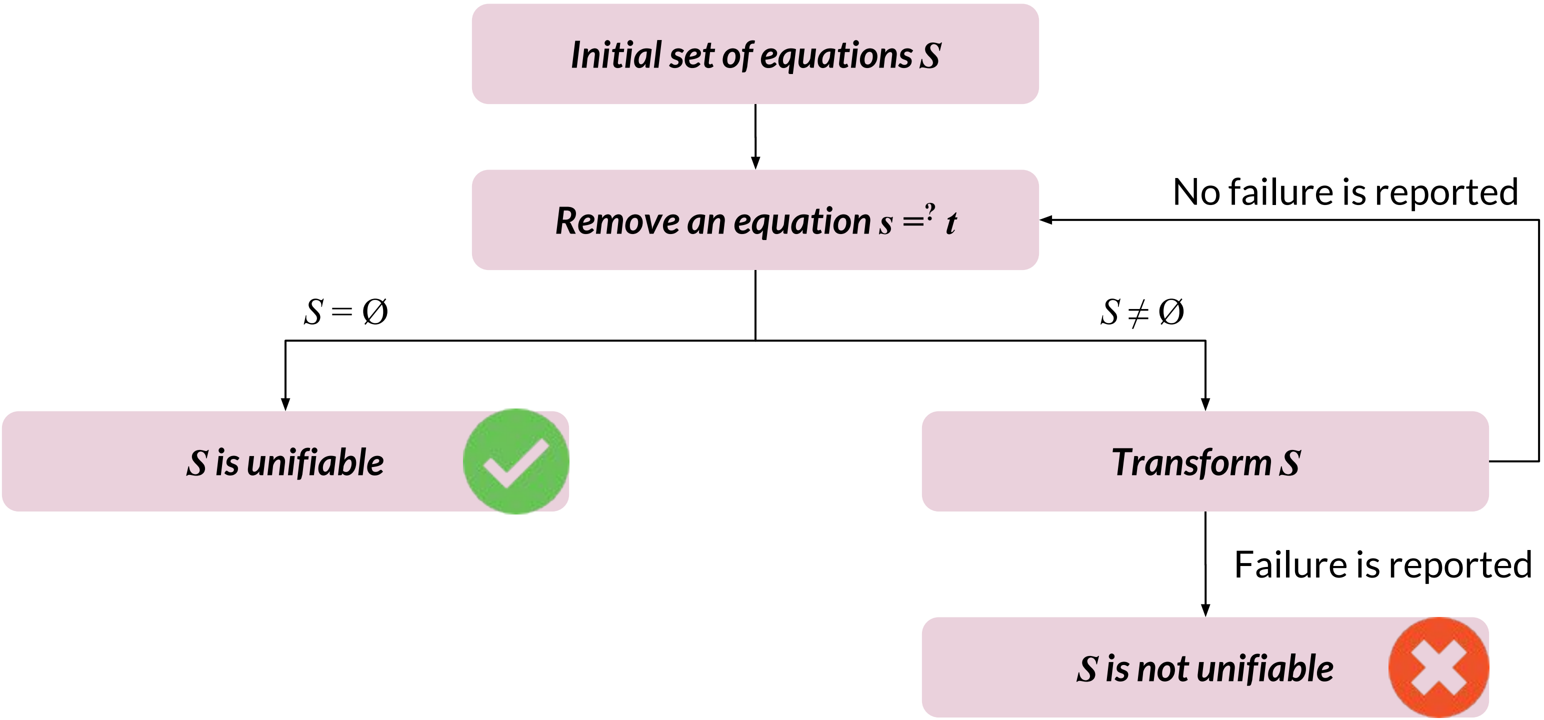
Given the set of equations

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

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

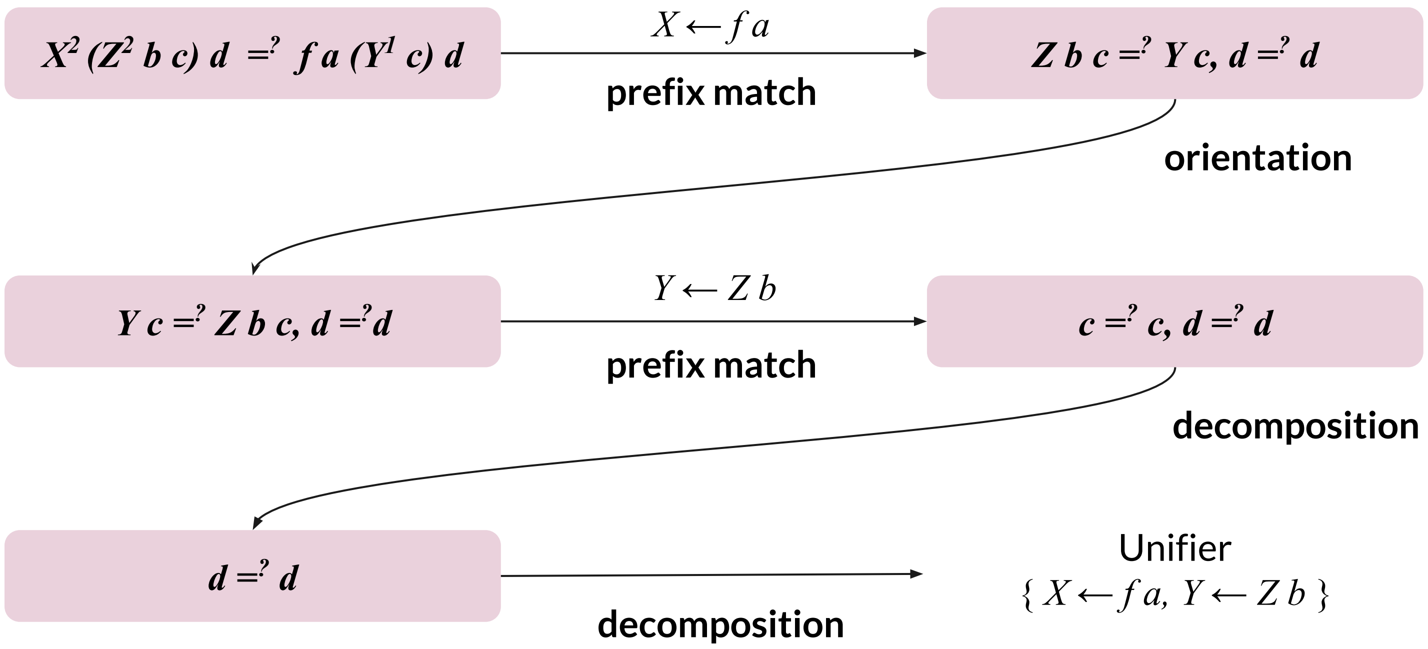
Match $s =^? t$

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

FOL algorithm fails on LFHOL terms

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

Example


$X^2\ (Z^2\ b\ c)\ d \ =^? \ f\ a\ (Y^1\ c)\ d$
$X \leftarrow f\ a$
prefix match
$Z\ b\ c =^? Y\ c,\ d =^? d$
orientation
$Y\ c =^? Z\ b\ c,\ d =^? d$
$Y \leftarrow Z\ b$
prefix match
$c =^? c,\ d =^? d$
decomposition
$d =^? d$
decomposition
Unifier
$\{\ X \leftarrow f\ a,\ Y \leftarrow Z\ b\ \}$

LFHOL equation set transformation

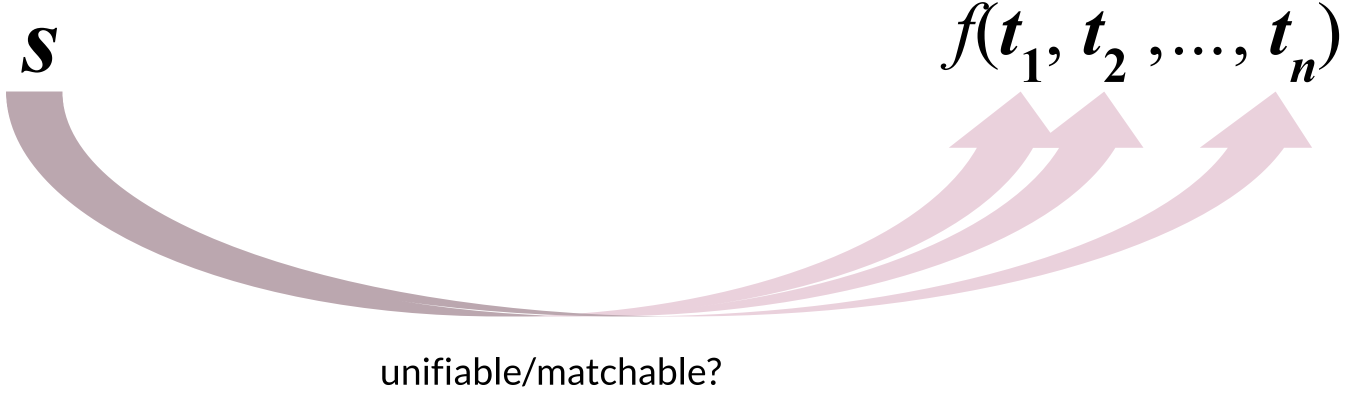
Match $s =^? t$

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

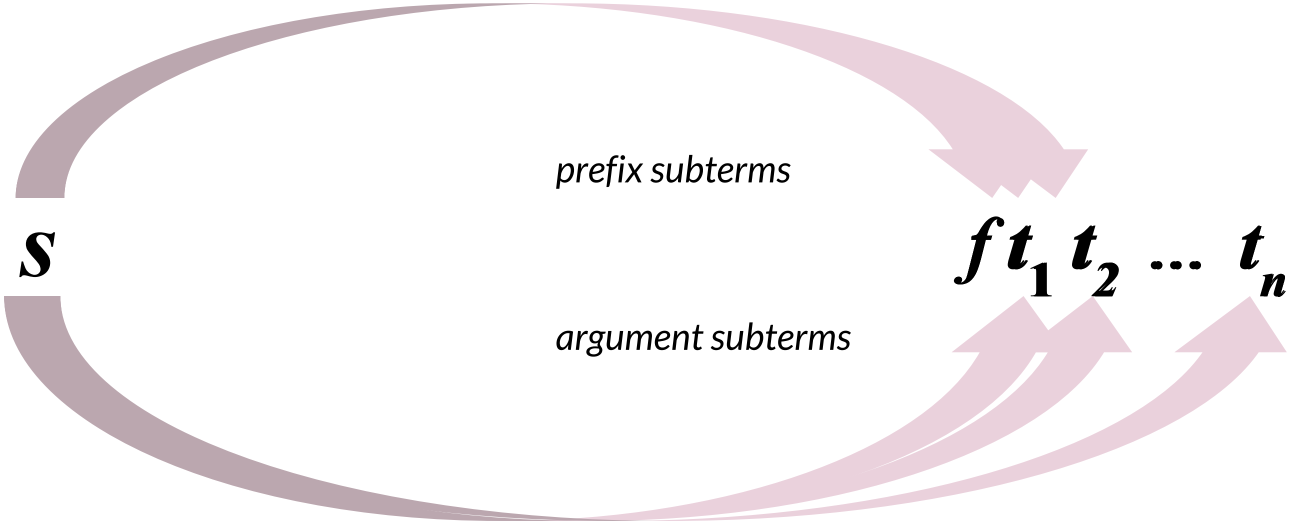
Standard FOL operations

unifiable/matchable?

… 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

$f\ X\ Y$ $\qquad\qquad$ $f\ a\ b\ c$

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

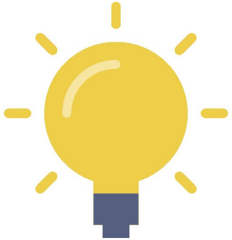
E’s indexing data structures

Discrimination trees

Fingerprint indexing

Feature vector indexing

Discrimination trees



Factor out operations common for many terms

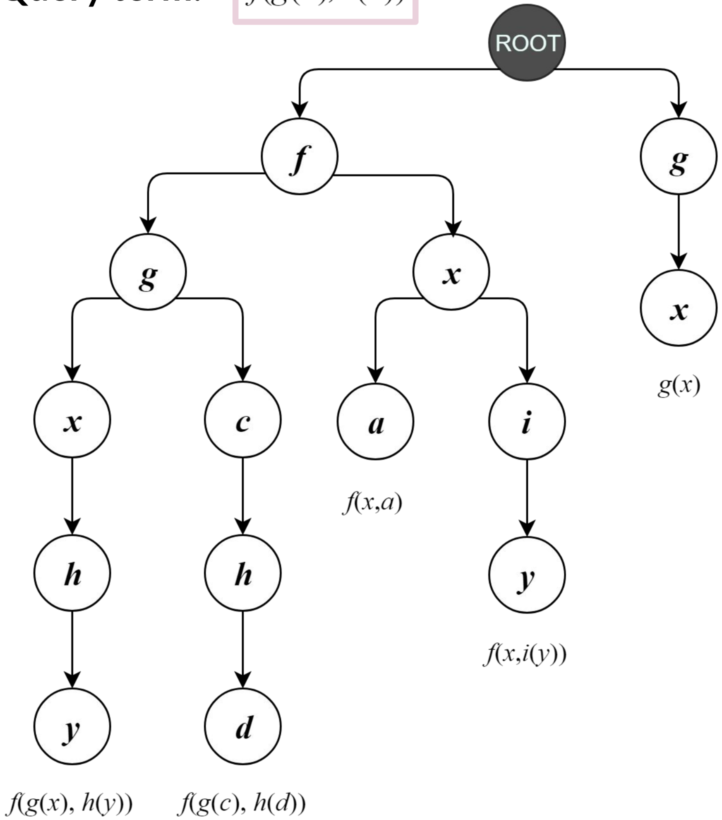
Flatten the term and use it as a key

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

Flattening: f x f h x y

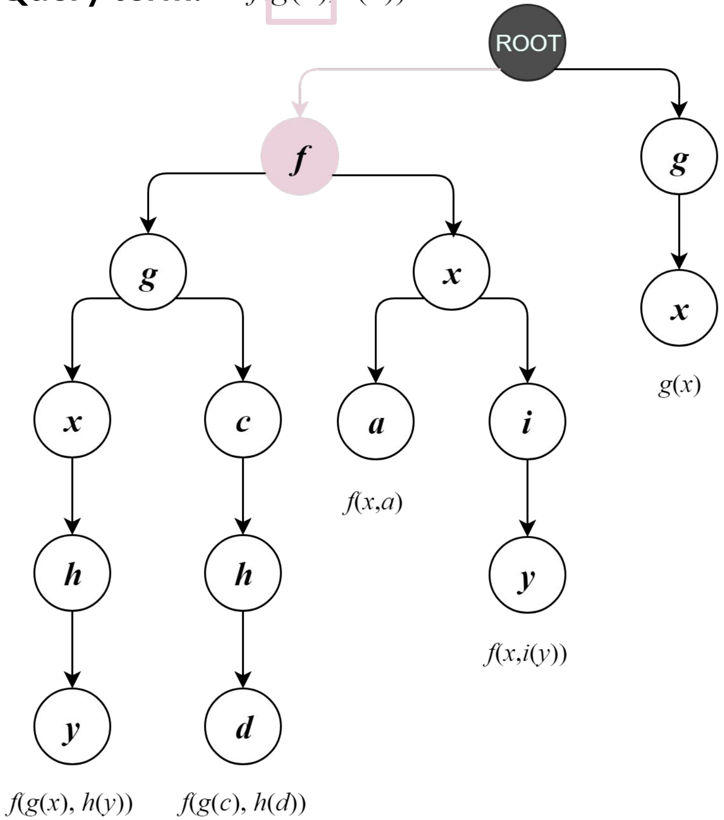
Example

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



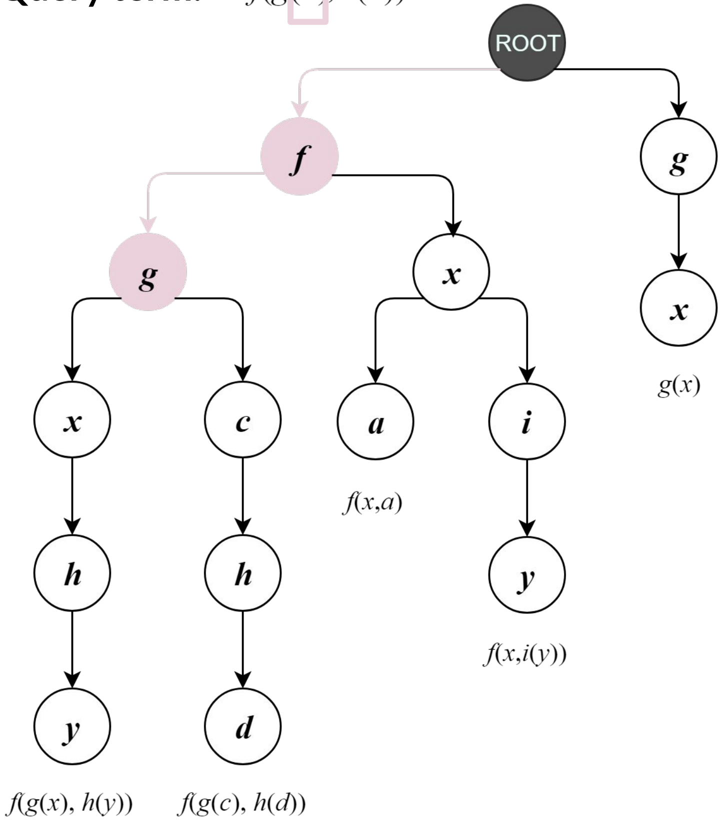
Example

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

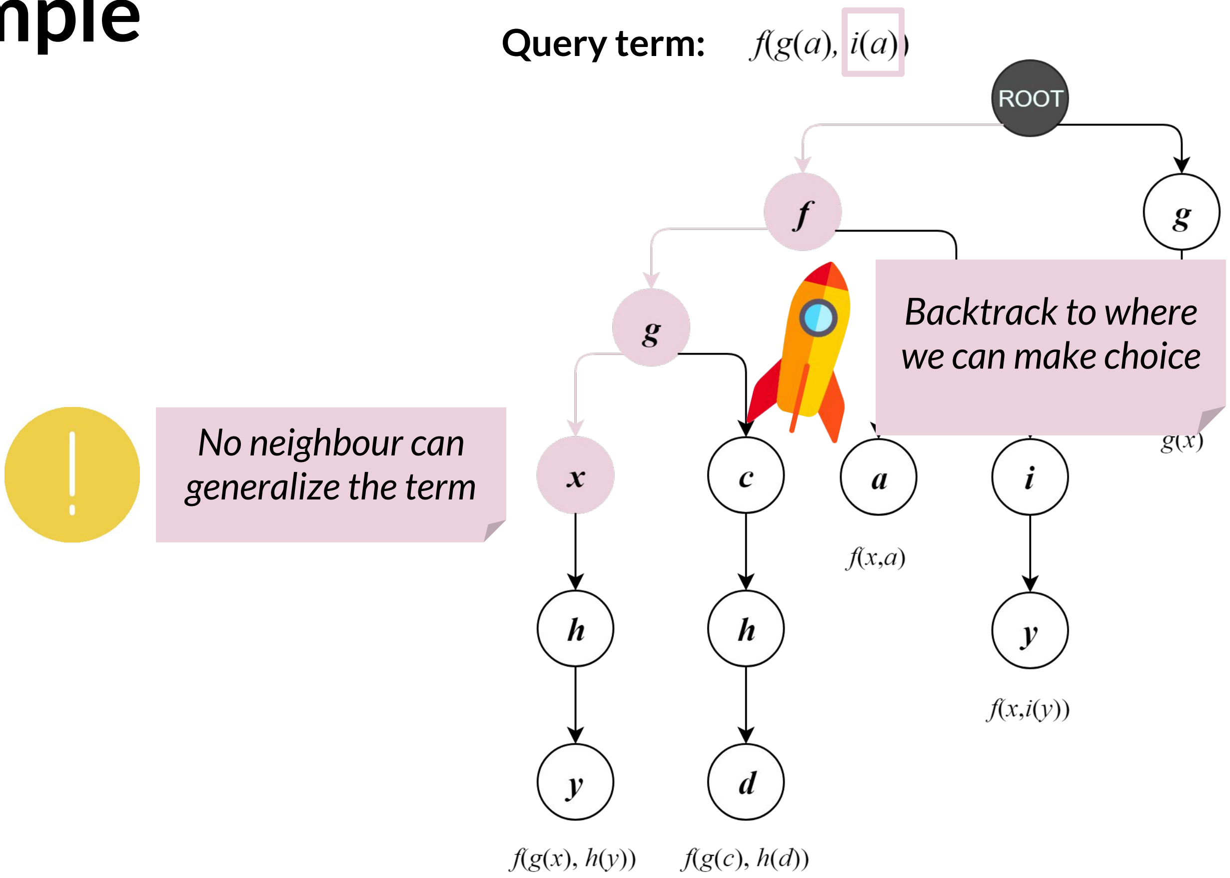


Example

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

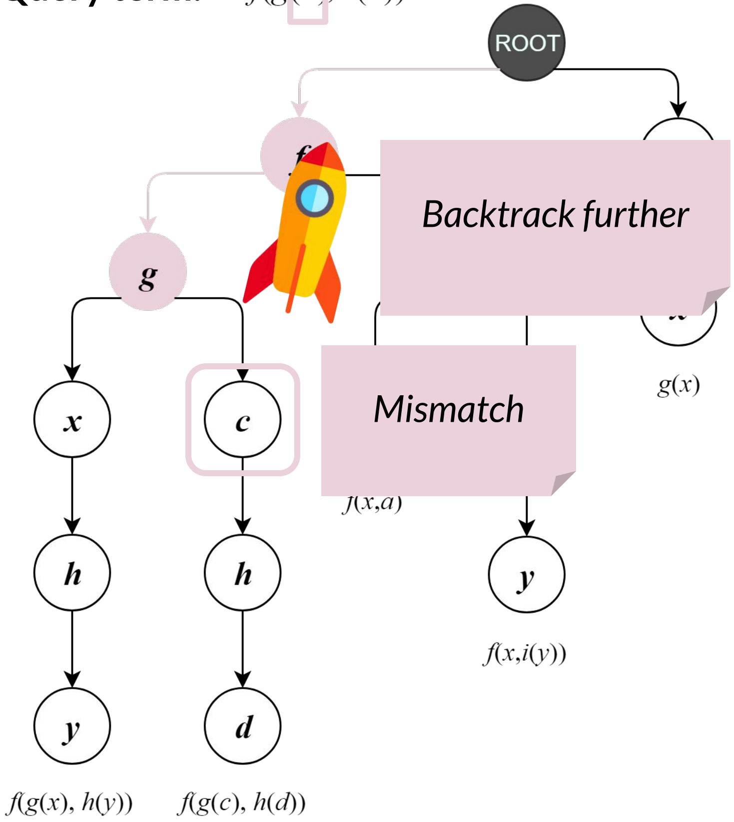


Example


Query term: f(g(a), i(a))
ROOT
f
g
g
Backtrack to where we can make choice
No neighbour can generalize the term
x
c
a
i
g(x)
f(x,a)
h
h
y
y
d
f(x,i(y))
f(g(x), h(y))
f(g(c), h(d))

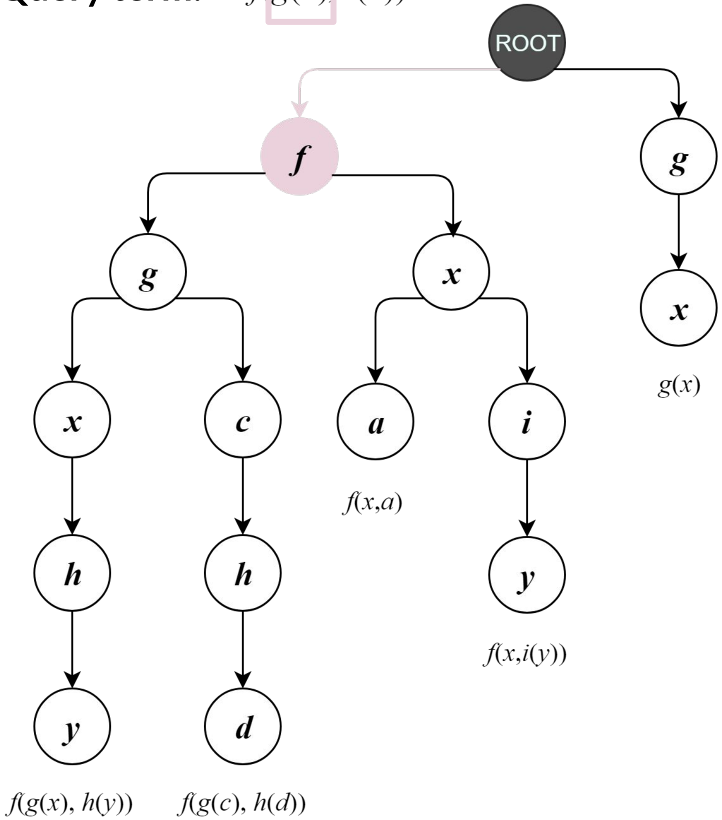
Example

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



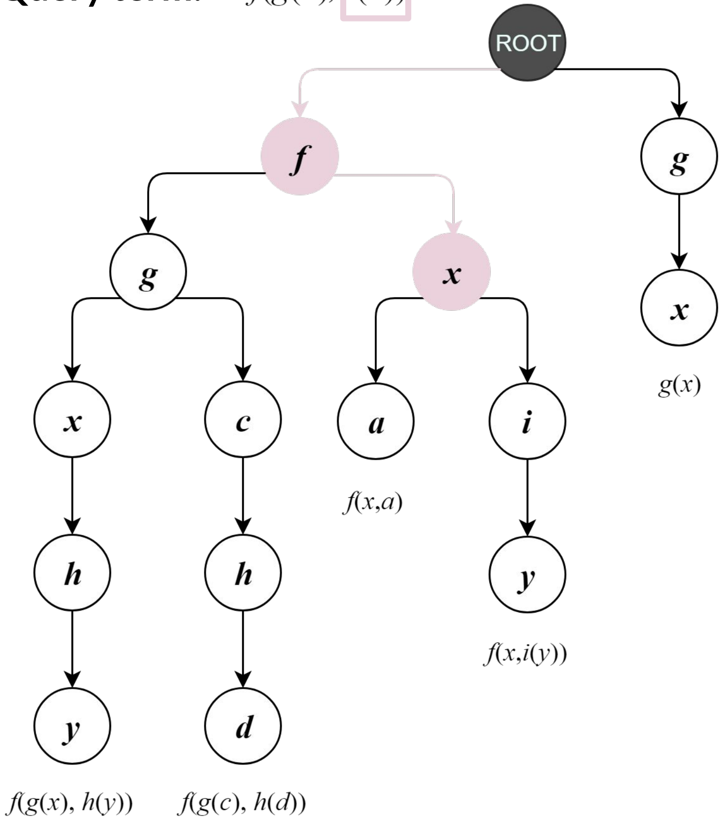
Example

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



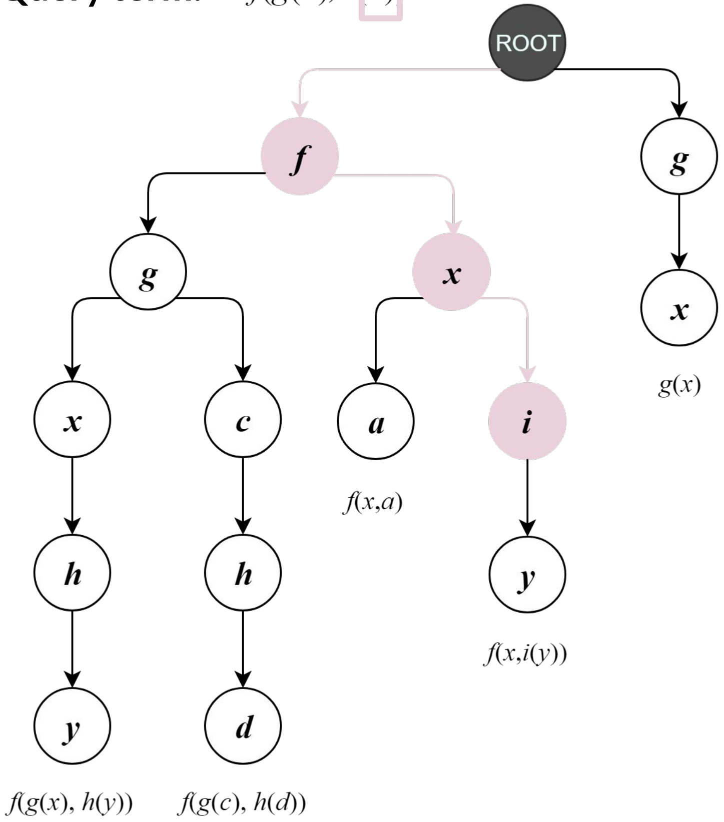
Example

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



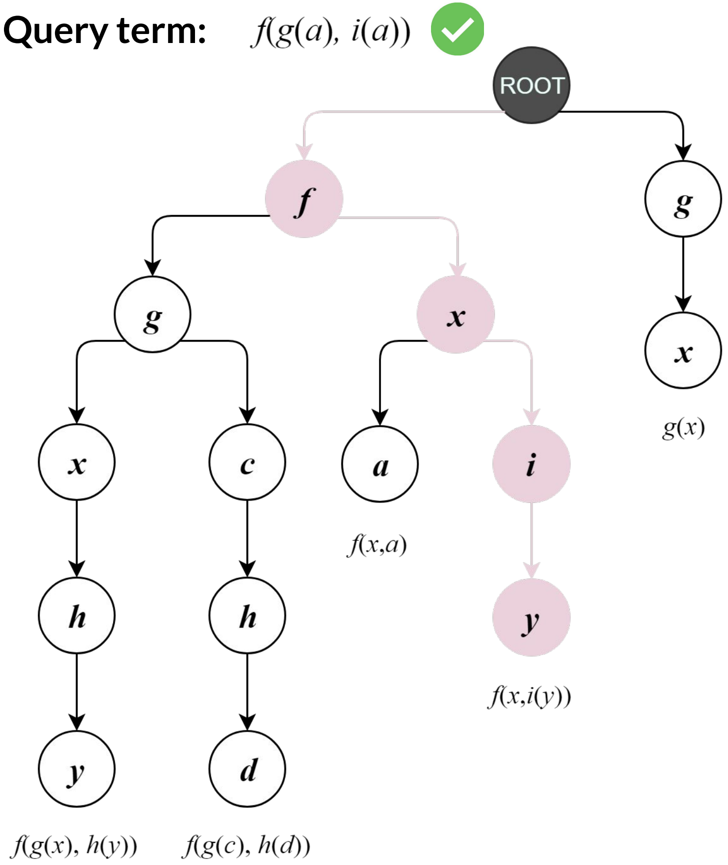
Example

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

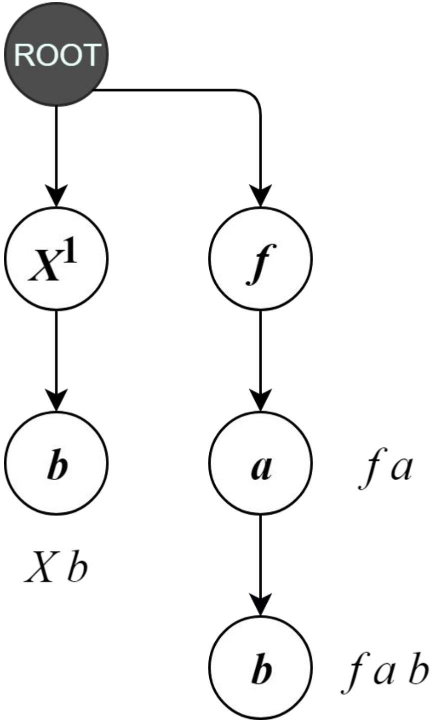


Example

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

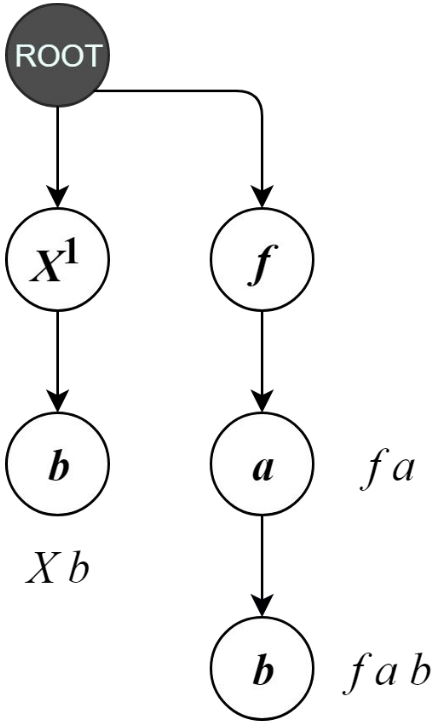


LFHOL challenges



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

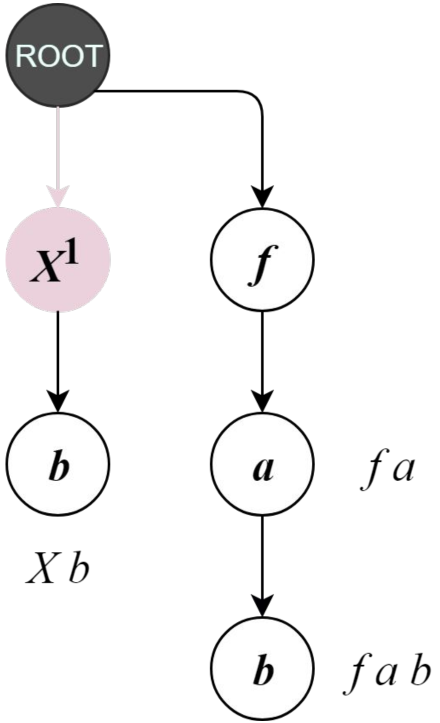
LFHOL challenges



Query term: $g\ a\ b$

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

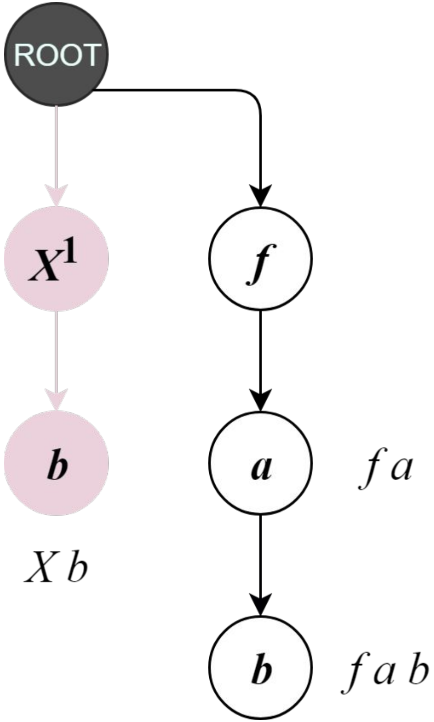
LFHOL challenges



Query term: $g\ a\ b$

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

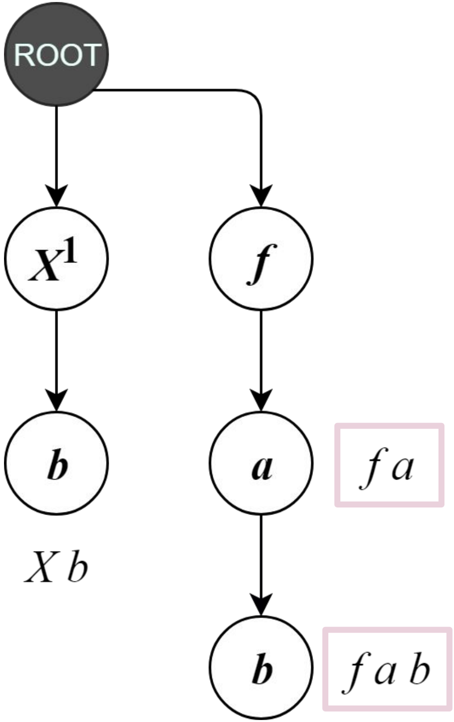
LFHOL challenges



Query term: $g\ a\ b$

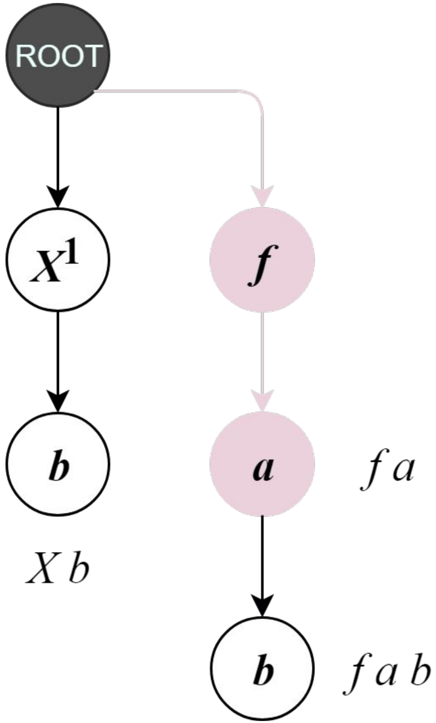
1. ***Applied variables***
   Variable can match a prefix
2. Terms prefixes of one another
3. Prefix optimization

LFHOL challenges



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

LFHOL challenges



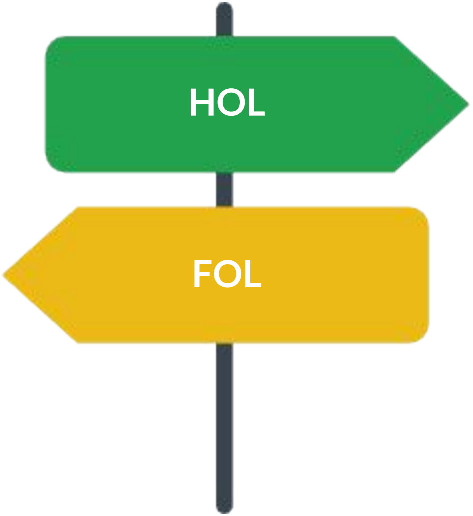
Query term: $f\ a\ \boxed{b}$

1. Applied variables

2. Terms prefixes of one another

3. ***Prefix optimization***
   Prefix matches are allowed

Experimentation results



Two compilation modes:

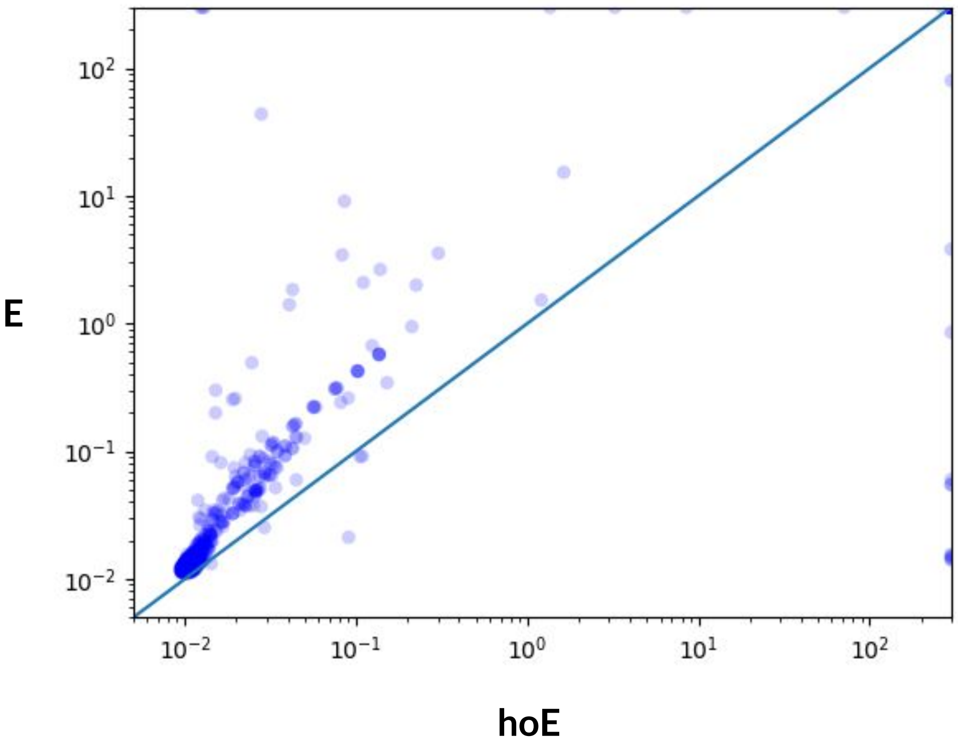
hoE - support for LFHOL

foE - support only for FOL

Gain on LFHOL problems

hoE vs. **original E**

995 (encoded)
LFHOL TPTP
problems



Gain on LFHOL problems

Both finished on **872/995** problems
hoE: **8** unique, E: **11** unique

Mean runtime:

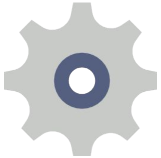
Total runtime:

Overhead on FOL problems

hoE vs. **E**

foE vs. **E**

Minimize the overhead for existing E users



Tested on 7789 FOL TPTP problems

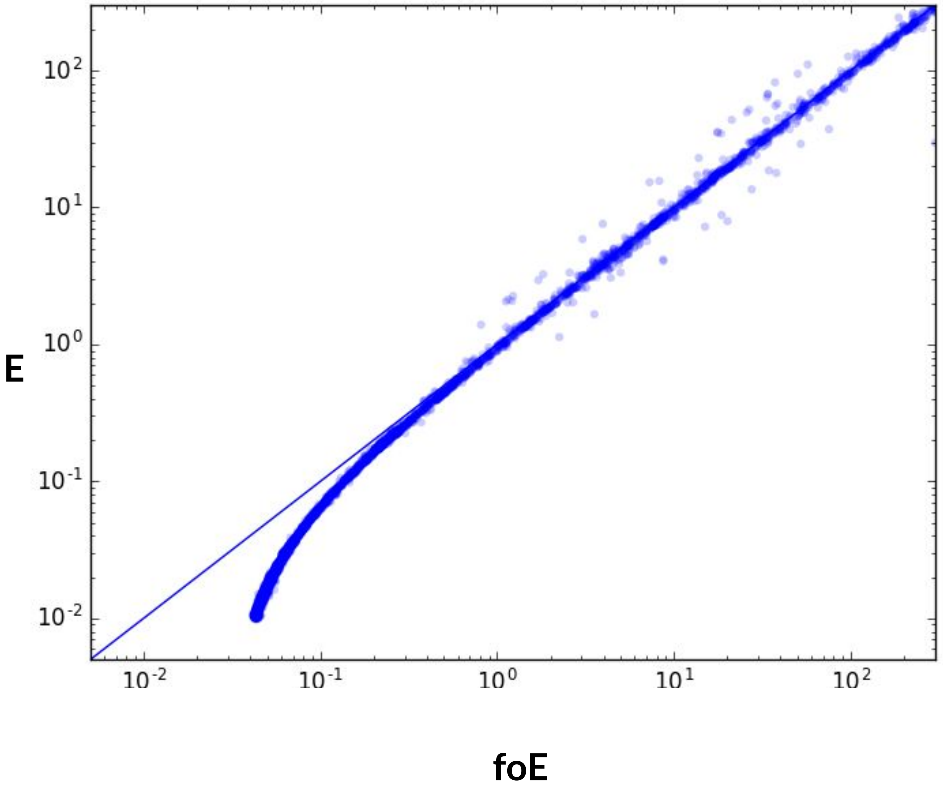
foE vs. E

Median runtime:

foE	1.4s
E	1.3s

Total runtime:

foE	845909s
E	844212s



hoE vs. E

Median runtime:

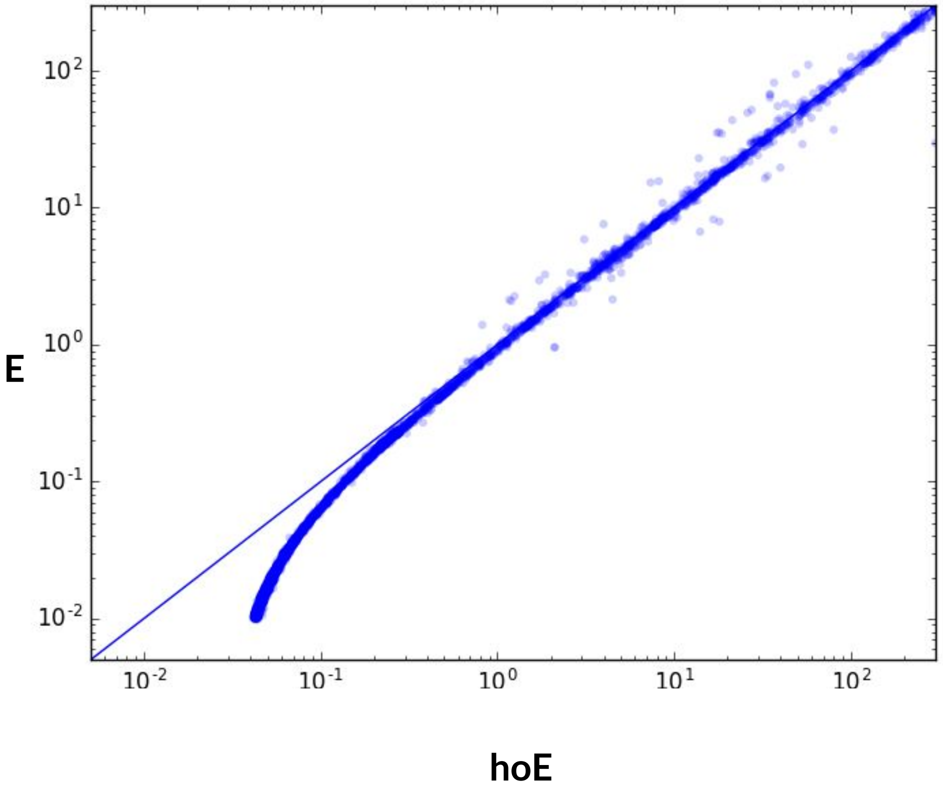
hoE 1.5s

E 1.3s

Total runtime:

hoE 846897s

E 844212s



Summary

Engineering viewpoint

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

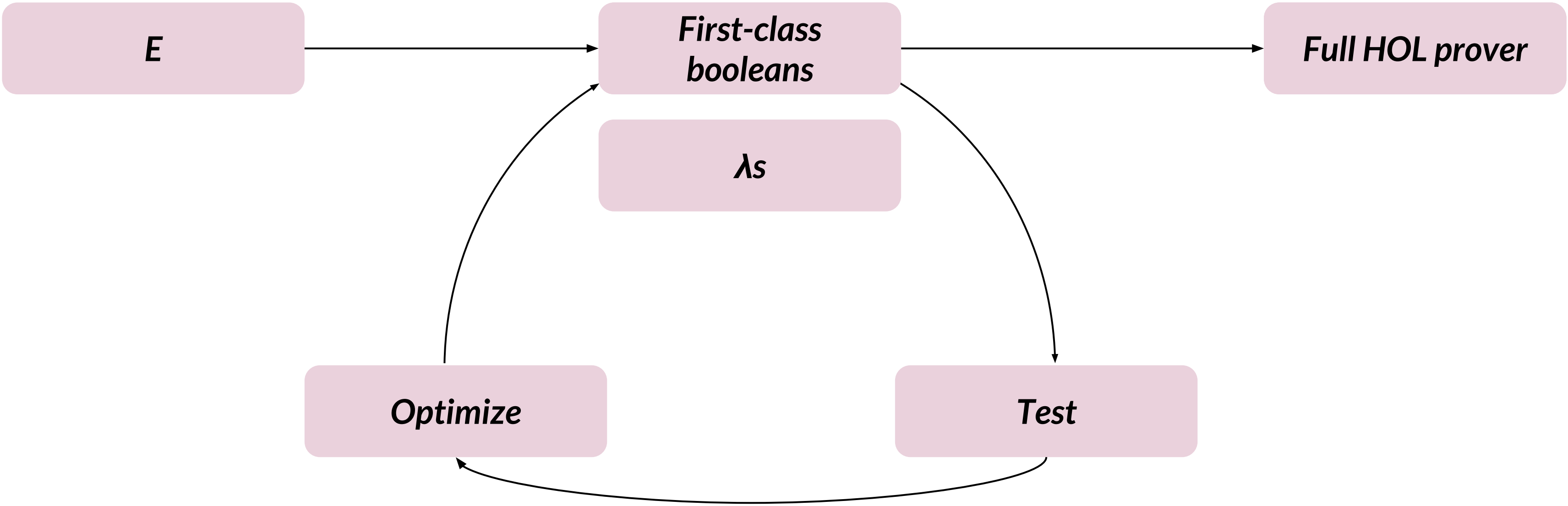
Theoretical viewpoint

- 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