Learning-Assisted Reasoning within Interactive Theorem Provers

Thibault Gauthier

April 30, 2020

What is interactive theorem proving?

Goal: Provides a **formal proof** of a theorem

Human: High-level proof plan

Automation: fills the gap in the proof.

What is it useful for?

- Verifying programs (CompCert, SEL4, CakeML)
- Verifying mathematical statements: 4-color, Kepler

What it should be useful for?

- Help discover new mathematical proofs

Plan

1) HOL4 interactive theorem prover

2) HOL(y)Hammer automation

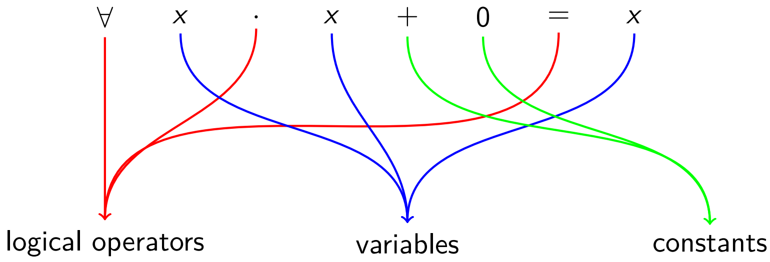
3) TacticToe automation

Interactive Theorem Provers	Theorems	Constants
Mizar	51086	9172
Coq	23320	4841
HOL4	16476	2247
HOL Light	16191	820
Isabelle/HOL	14814	1076
Matita	1712	629

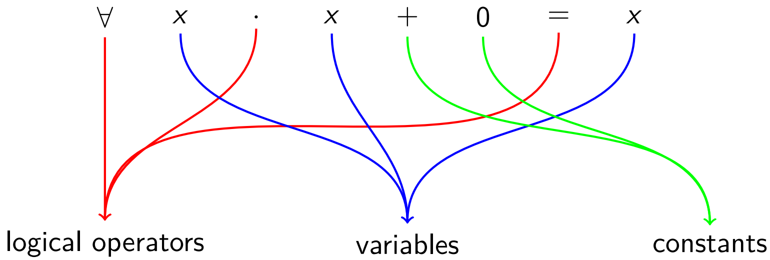
Philosophy of HOL4 logic

Set theory vs Type theory

How to represent a formula in HOL4 ?



How to represent a formula in HOL4 ?



$$\forall\ (\lambda x.\ (=\ ((+\ x)\ 0)\ x))$$

HOL4 calculus (basic rules)

Natural deduction presented in sequents with one conclusion:

- Rules for logical connectives

- Rules for equality

- Rules for functions

- 4 additional axioms

- Definitions of new functions.

Secure: only using these rules, one can derive new theorems.

Programming new rules from the basic rules

Examples of non-trivial theorem producing procedures

1) Transitive closure checker:
Proves that two formulas are equal using a set of equalities.

2) Simplifier:
Simplify a theorem using set of rewriting rules

Programming tactics

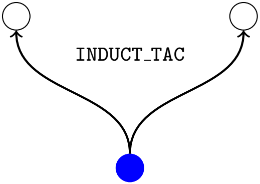
A **tactic** takes a goal g and produces new goals and a validation. The validation takes the proven new goals and proves g.

A **goal** is a sequent that is not proven. The set of assumption of the sequent is often empty, so we can often consider the goal to just be a formula.

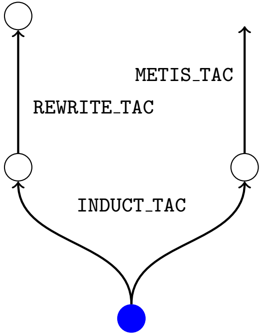
Common tactics: `INDUCT_TAC`, `REWRITE_TAC`, `METIS_TAC`

`THENL` tactical composes the effect of tactics.

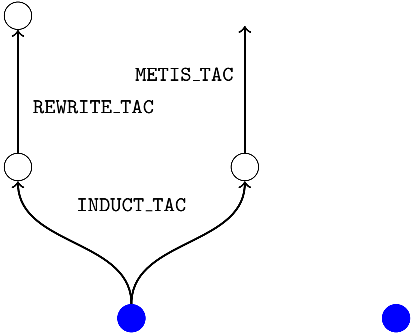
THENL tactical composes the effect of tactics.



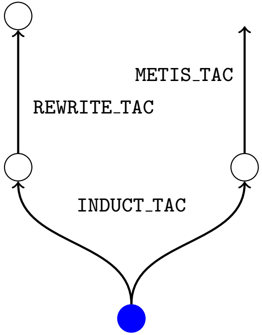
THENL tactical composes the effect of tactics.

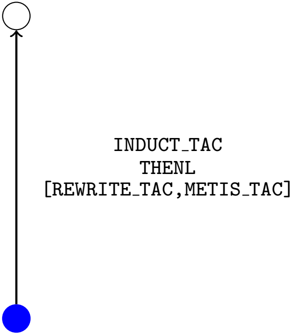


THENL tactical composes the effect of tactics.



THENL tactical composes the effect of tactics.



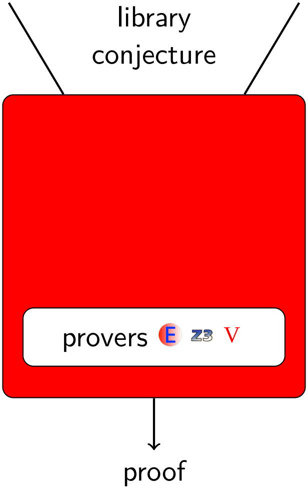


HolyHammer

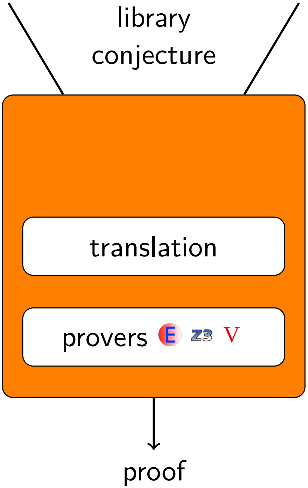
library
conjecture

proof

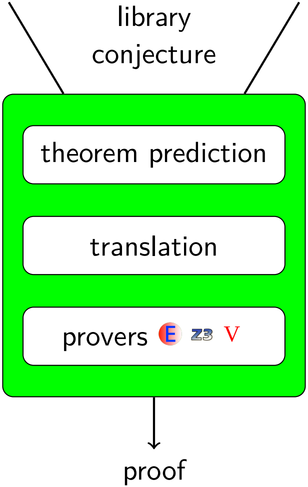
HolyHammer



HolyHammer



HolyHammer



library
conjecture
theorem prediction
translation
provers
E
Z3
V
proof

Theorem prediction

Which theorems are useful for the prove a goal (conjecture) g?

1) Theorems that are similar to g.

2) Theorems that were used in proofs of theorems similar to g.

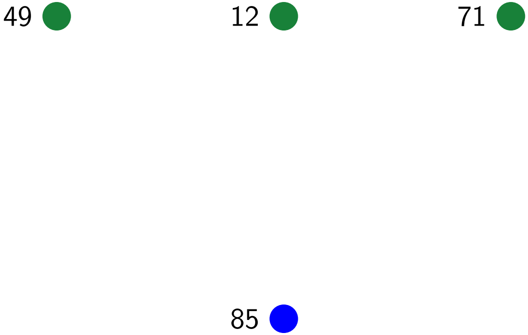
Theorem prediction: similar theorems

	Formula	Syntactic features
Conjecture	$\forall x, y.\ (x+y) \times (x-y) = x^2 - y^2$	
	$\forall x, y, z.\ x \times (y+z) = x \times y + x \times z$	
	$\forall x, y.\ x+y = y+x$	
Library	$\forall x, y.\ x \times y = y \times x$	
	$e^{i\pi} + 1 = 0$	
	$(x^2)' = 2 \times x$	

Theorem prediction: similar theorems

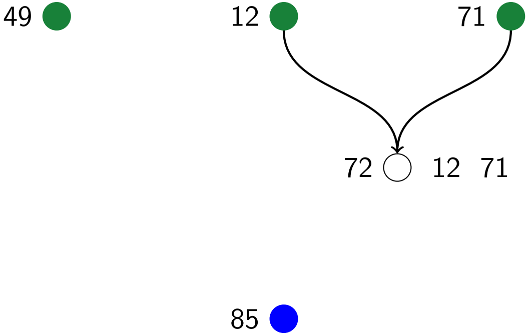
	Formula	Syntactic features
Conjecture	$\forall x, y.\ (x+y)\times(x-y) = x^2 - y^2$	$+, \times, \ ^2$
	$\forall x, y, z.\ x \times (y+z) = x \times y + x \times z$	$\times, +$
	$\forall x, y.\ x + y = y + x$	$+$
Library	$\forall x, y.\ x \times y = y \times x$	$\times$
	$e^{i\pi} + 1 = 0$	$e, i, \times, \pi, +, 1, 0$
	$(x^2)' = 2 \times x$	$', 2, \times, \ ^2$

Theorem prediction: dependencies



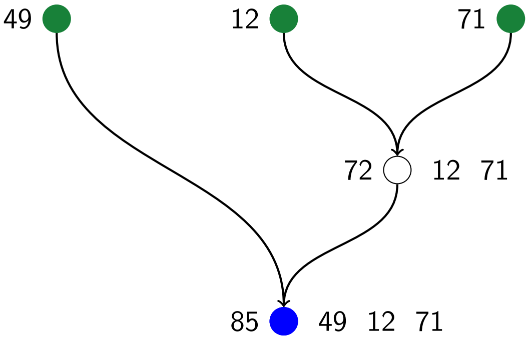
● conjecture
● theorem
$\rightarrow$ rule
○ lemma

Theorem prediction: dependencies



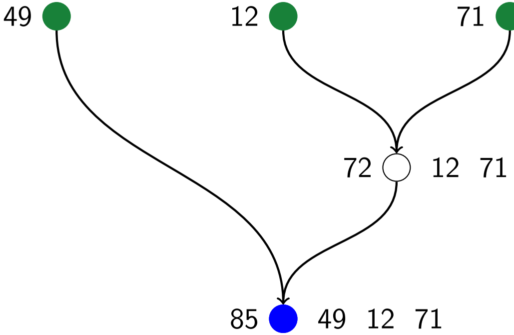
- conjecture
- theorem
- $\rightarrow$ rule
- ○ lemma

Theorem prediction: dependencies



- conjecture
- theorem
- $\rightarrow$ rule
- lemma

Theorem prediction: dependencies



● conjecture
● theorem
→ rule
○ lemma

Theorem	Dependencies
85	49 12 71
102	51 45 86 12
...	...

Translation

HOL4 **higher-order** formulas vs ATPs **first-order** formulas.

Translation

HOL4 **higher-order** formulas vs ATPs **first-order** formulas.

There exist higher-order ATPs: Satallax, Leo-3.

Translation: Lambda-lifting

Higher-order:

$$\forall k.\ \text{linear}\ (\lambda x.\ k \times x)$$

Higher-order (lambda-free):

$$\forall k\ x.\ \mathbf{f}\ k\ x =_{def} k \times x$$

$$\forall k.\ \text{linear}\ (\mathbf{f}\ k)$$

Translation: Apply operator

Higher-order:

$$\forall f.\ (I\ f)\ x = f\ x$$

First-order:

$$\forall f.\ \mathrm{ap}(\mathrm{ap}(I, f), x) = \mathrm{ap}(f, x)$$

Translation: Apply operator

Higher-order:

$$\forall f.\ (I\ f)\ x = f\ x$$

First-order:

$$\forall f.\ \mathsf{ap}(\mathsf{ap}(I, f), x) = \mathsf{ap}(f, x)$$

Optional axiom: $f_1(x) = \mathsf{ap}(f, x)$

Type encoding

$$\forall x\,{:}\,\mathit{real}.\ I(x) = x$$

FOF guards:

$$\forall x\,{:}\,\mathit{set}.\ x \in \mathit{real} \Rightarrow I(x) = x$$

Additional axiom: $\forall y.\ y \in \mathit{real} \Rightarrow I(y) \in \mathit{real}$

FOF tags:

$$\forall x\,{:}\,\mathit{set}.\ \mathrm{s}(\mathit{real}, I(\mathrm{s}(\mathit{real}, x))) = \mathrm{s}(\mathit{real}, x)$$

Polymorphism? Type variables as term variables.

Re-proving

Tested library	Hammer	Benchmark	Success
	MizAR	standard library	40% (60%?)
	SledgeHammer	judgement day	77%
	Hol(y)Hammer	flyspeck	39%
	Hol(y)Hammer	standard library	50%
	CoqHammer	standard library	41%

Summary

HOL(y)Hammer can solve a given goal automatically by:

- select relevant theorems among tens of thousands of theorems

- translating those theorems and the goal to ATPs

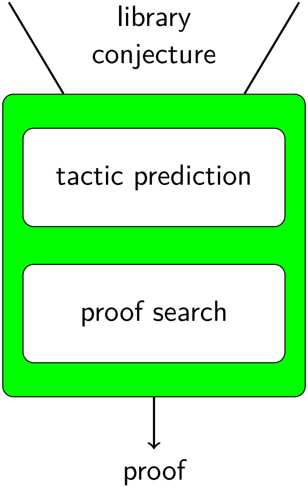
TacticToe

library

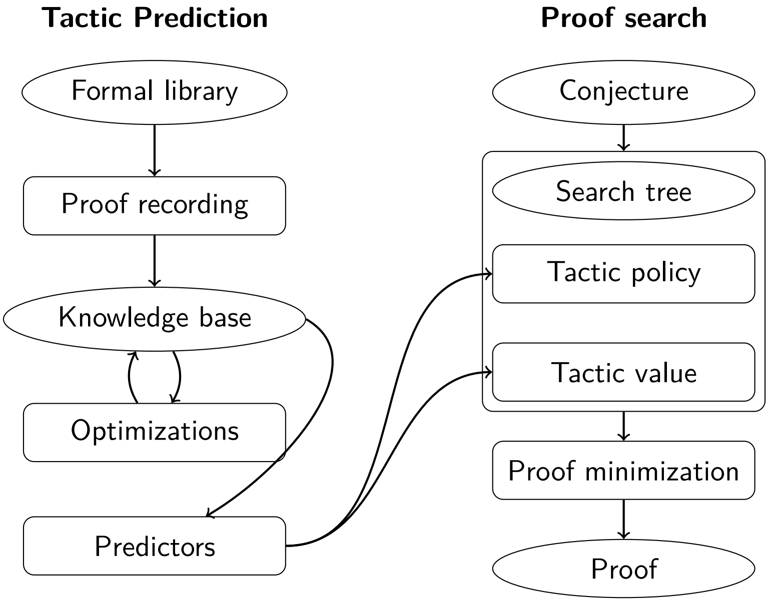
conjecture

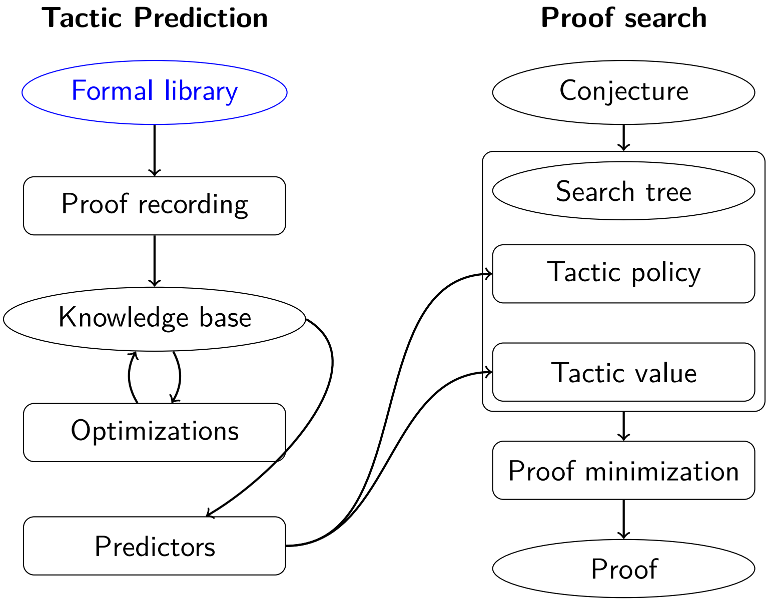
proof

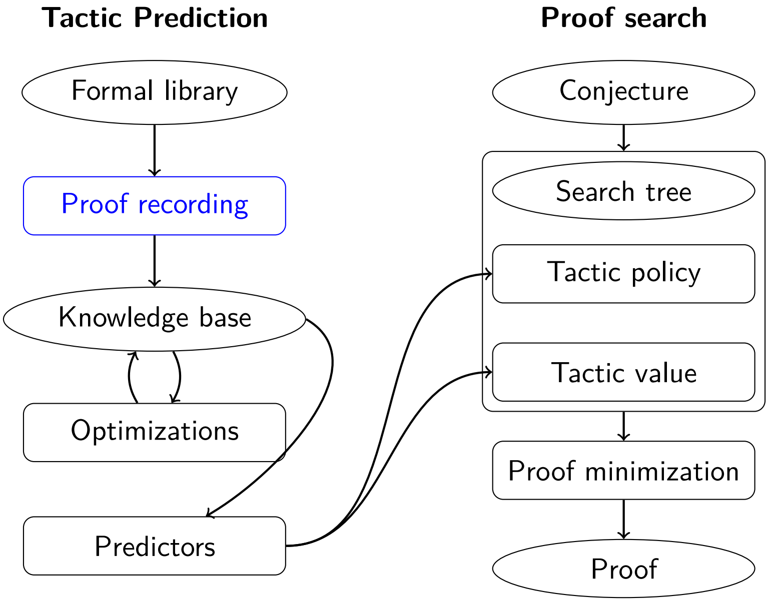
TacticToe



Tactics	Useful for
Solvers	linear system, differential equations
Simplifiers	reducing fractions, differentiation
Induction	natural numbers, lists, trees


Tactic Prediction
Proof search
Formal library
Proof recording
Knowledge base
Optimizations
Predictors
Conjecture
Search tree
Tactic policy
Tactic value
Proof minimization
Proof


Tactic Prediction
Formal library
Proof recording
Knowledge base
Optimizations
Predictors
Proof search
Conjecture
Search tree
Tactic policy
Tactic value
Proof minimization
Proof


Tactic Prediction
Proof search
Formal library
Proof recording
Knowledge base
Optimizations
Predictors
Conjecture
Search tree
Tactic policy
Tactic value
Proof minimization
Proof

Proof recording

Original proof:

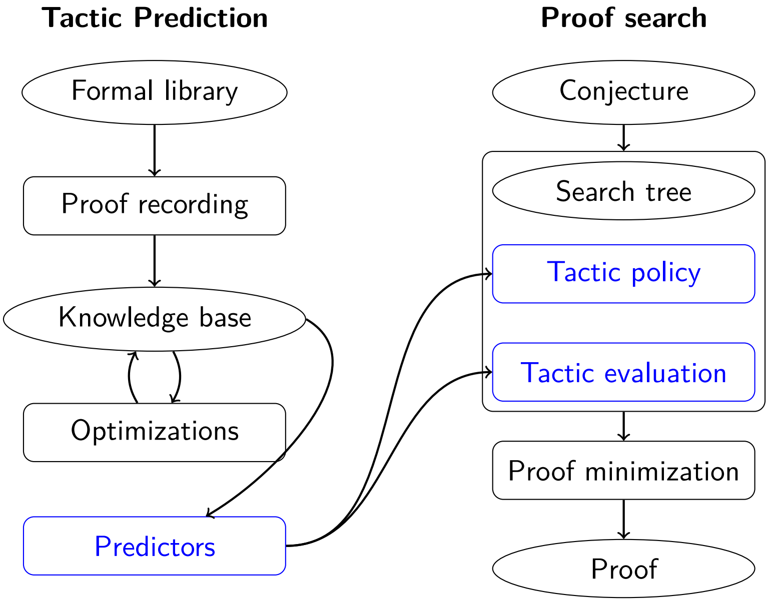
```
INDUCT_TAC THENL [REWRITE_TAC, METIS_TAC]
```

Modified proof:

```
(R numLib.INDUCT_TAC) THENL
  [R boolLib.REWRITE_TAC, R metisLib.METIS_TAC]
```

Database of tactics:

```
R (f n) (f (SUC n)) ⇒ transitive R: INDUCT_TAC
n ∗ m ≤ n ∗ p ⇒ (n = 0) ∨ m ≤ p    : REWRITE_TAC
INJ f U(:num) s ⇒ INFINITE s       : METIS_TAC
...
```


Tactic Prediction
Proof search
Formal library
Proof recording
Knowledge base
Optimizations
Predictors
Conjecture
Search tree
Tactic policy
Tactic evaluation
Proof minimization
Proof

Policy prediction algorithm

Given a new goal g, which tactics lead towards a proof?

Tactics that were useful for goals similar to g.

Policy prediction algorithm

Given a new goal g, which tactics lead towards a proof?

Tactics that were useful for goals similar to g.

Similarity determined using the nearest neighbor algorithm.

Policy prediction algorithm

Database of tactics is a map from goals to tactics.

```
R (f n) (f (SUC n)) ⇒ transitive R: INDUCT_TAC
n * m ≤ n * p ⇒ (n = 0) ∨ m ≤ p   : REWRITE_TAC
INJ f U(:num) s ⇒ INFINITE s      : METIS_TAC
```

New goal:

```
LENGTH (MAP f l) = LENGTH l
```

Policy for the new goal:

```
Rank Tactic      Policy
1    REWRITE_TAC 0.5
2    METIS_TAC   0.25
...
4    INDUCT_TAC  0.0625
...
```

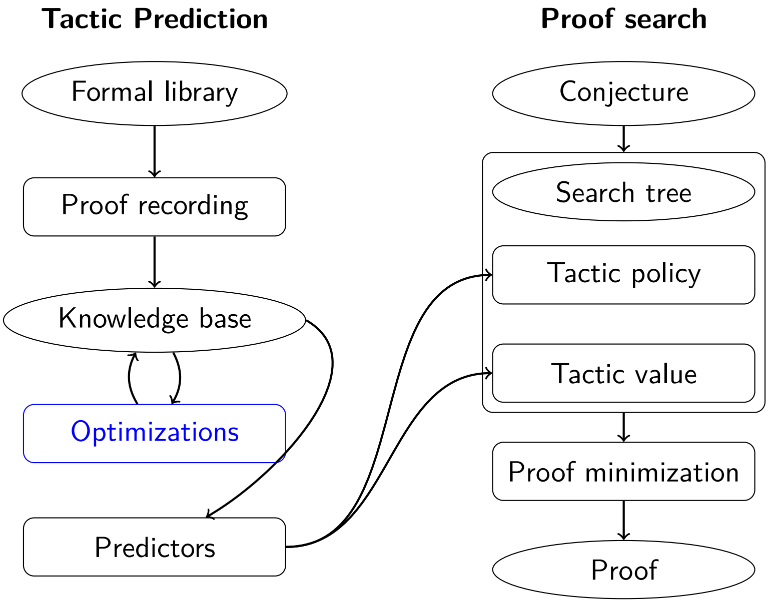
Predicting the value of a goal (or list of goals)

Database of goals:

- Positive examples: produced during TacticToe search and appear in the final proof.
- Negative examples: produced during TacticToe search but do not appear in the final proof.

Future idea: tactic modeling using the value.

Plan


Tactic Prediction
Proof search
Formal library
Proof recording
Knowledge base
Optimizations
Predictors
Conjecture
Search tree
Tactic policy
Tactic value
Proof minimization
Proof

Optimizations

Improve recorded data to create better predictions during search.

Optimizations: orthogonalization

Issue: Many tactics are doing the same job on a goal g.

Solution: Competition for g where the most popular tactic wins.

Optimizations: orthogonalization

Recorded goal-tactic pair:

`LENGTH (MAP f l) = LENGTH l: INDUCT_TAC`

Competition:

	Progress	Coverage
INDUCT_TAC	Yes	136
REWRITE_TAC	No	2567
METIS_TAC	Yes	694

Added to the database:

`LENGTH (MAP f l) = LENGTH l: METIS_TAC`

Result: 6 % improvement.

Optimizations: abstraction

Issue: Some theorems are never used inside tactics.

Solution: Abstract all lists of theorems in a tactic
and instantiate them depending on the target goal.

Optimizations: abstraction

Abstraction algorithm:

```
Original      : REWRITE_TAC [T1,T2]
Abstraction   : REWRITE_TAC X
Instantiation: REWRITE_TAC [T67, T1, T43, ..]
```

Question: Dow we keep the original or the abstraction ?

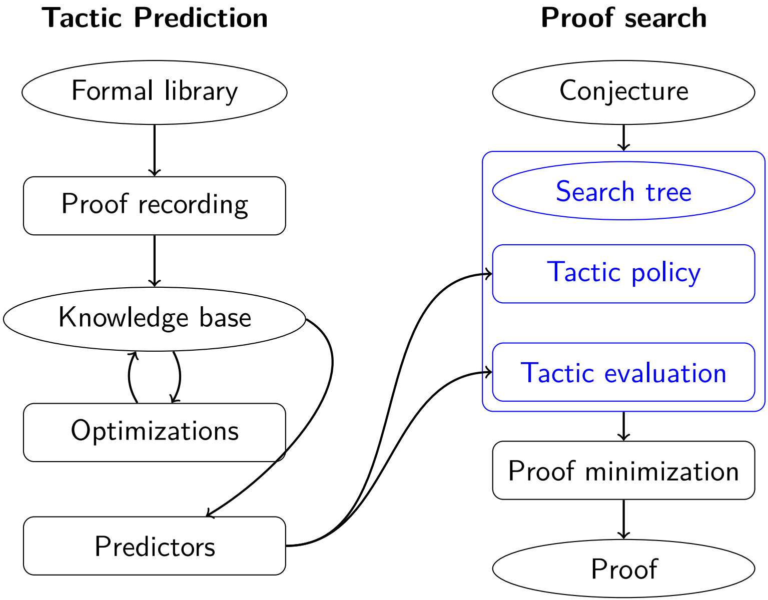
Answer: Let them compete during orthogonalization.

Result: 15% improvement

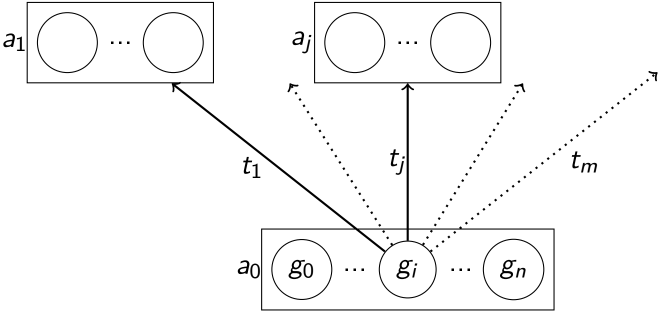
Optimizations: preselection

Issue: Predictions are too slow during proof search.

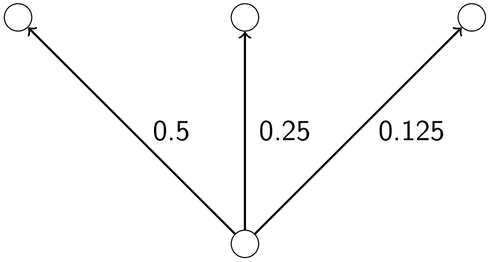
Solution: Preselect 500 suitable tactics by importing proofs (many tactics) from related goals.


Tactic Prediction
Proof search
Formal library
Proof recording
Knowledge base
Optimizations
Predictors
Conjecture
Search tree
Tactic policy
Tactic evaluation
Proof minimization
Proof

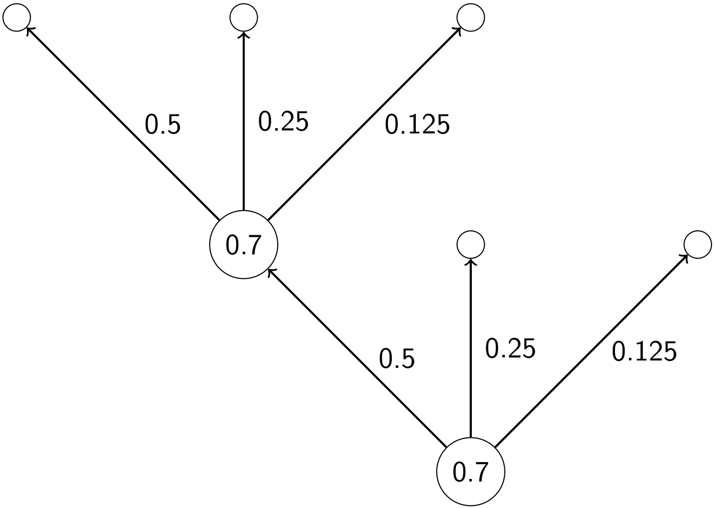
Proof search: search tree



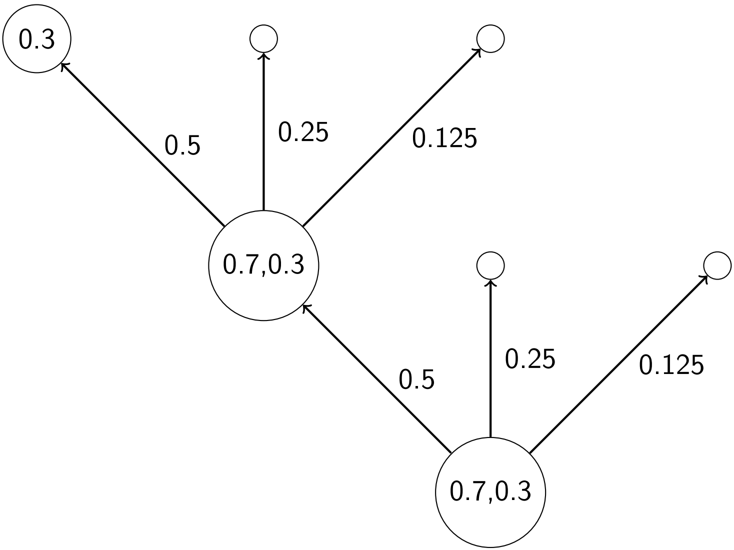
Proof search: advanced tree search



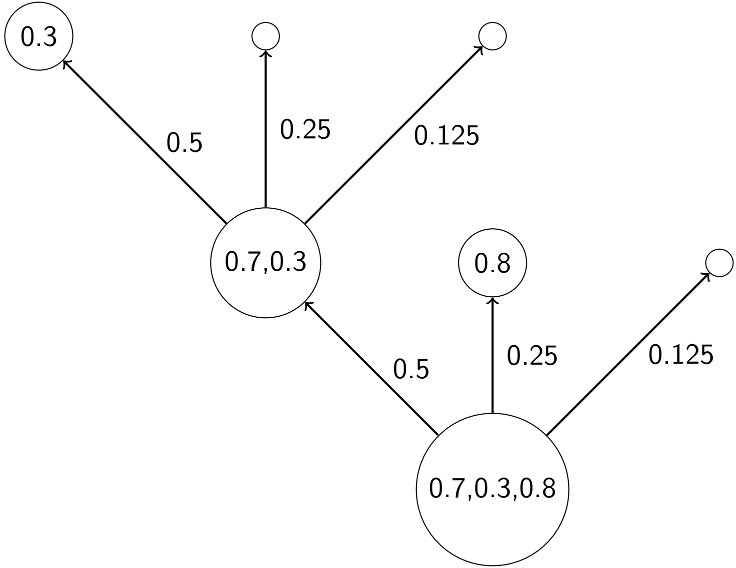
Proof search: advanced tree search



Proof search: advanced tree search



Proof search: advanced tree search



Proof search: advanced tree search (PUCT)

Here p is the parent node of $a_1, \ldots, a_n$ and PUCT is a heuristic to decide which branch (child) to expand next.

$$Score(a_i) = CurValue(a_i) + c_{exploration} * \frac{PriorPolicy(a_i)}{CurPolicy(a_i)}$$

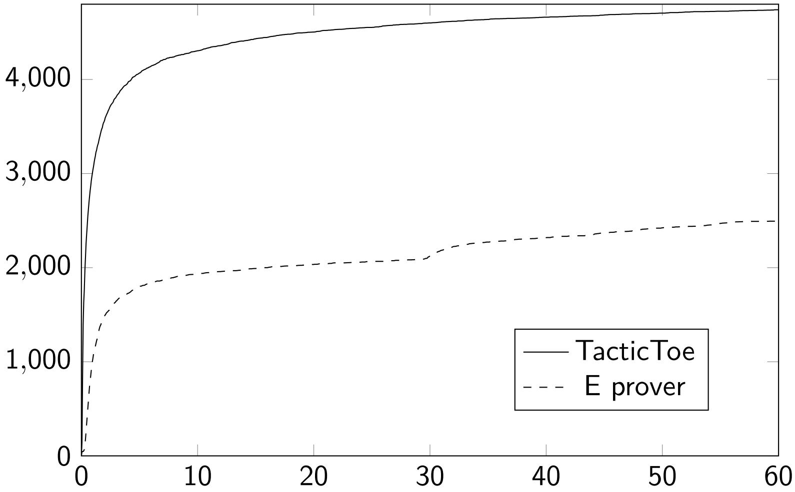
$$CurPolicy(a_i) = \frac{1 + Visit(a_i)}{\sqrt{Visit(p)}}$$

$$CurValue(a_i) = \sum_{a' \in Descendants(a_i)} \frac{PriorValue(a')}{card(Descendants(a_i))}$$

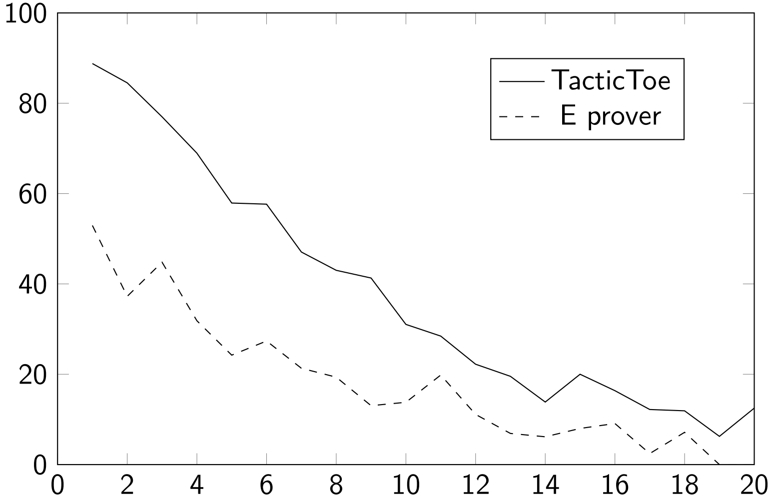
Re-proving

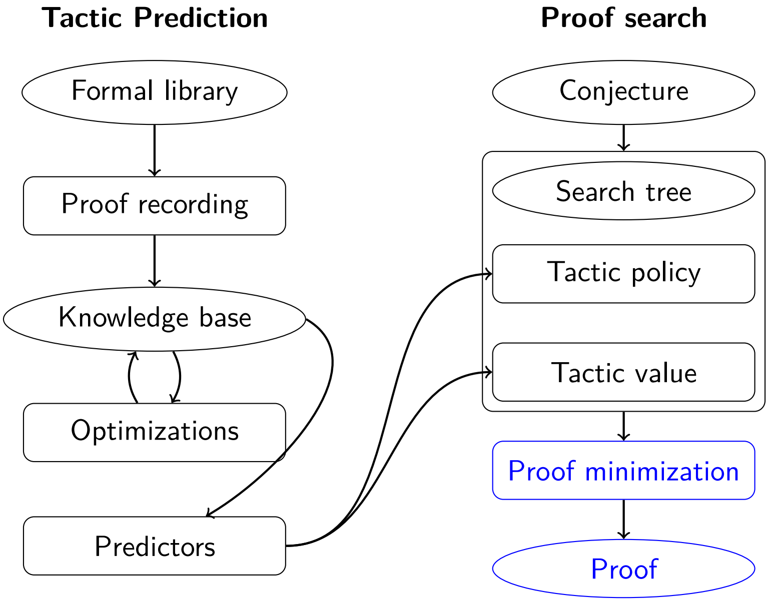
Tested library	Proof automation	Success
		50%
		66%

Re-proving: HOL4 proofs found in less than x seconds



Re-proving: percentage of solved HOL4 proof of size x




Tactic Prediction
Proof search
Formal library
Proof recording
Knowledge base
Optimizations
Predictors
Conjecture
Search tree
Tactic policy
Tactic value
Proof minimization
Proof

Before:

```
boolLib.REWRITE_TAC [DB.fetch "list" "EVERY_CONJ",... ]
  THEN
BasicProvers.Induct_on [HolKernel.QUOTE "l"]
  THENL
  [BasicProvers.SRW_TAC [] [],
   simpLib.ASM_SIMP_TAC (BasicProvers.srw_ss ())
   [boolLib.DISJ_IMP_THM, DB.fetch "list" "MAP",
    DB.fetch "list" "CONS_11", boolLib.FORALL_AND_THM]]
```

After:

```
Induct_on `l` THENL
  [SRW_TAC [] [],
   ASM_SIMP_TAC (srw_ss ())
   [DISJ_IMP_THM, FORALL_AND_THM]]
```

Summary

TacticToe learns from human proofs to solve new goals.

Advantages over ATPs (E prover) for ITP (HOL4) users:

- Includes domain specific automation found in the ITP. (tactics)
- Generated proofs are human-level proofs.
- No translation or reconstruction needed.

Limitations: TacticToe cannot program its own tactics yet.

`http://grid01.ciirc.cvut.cz/~mptp/tactictoe_demo.ogv`

Proof inspection

$$2 \times \sum_{x=0}^{n} x = n \times (n+1)$$

```
Induct_on `n` THENL
  [SRW_TAC [] [] THEN METIS_TAC [SUM_1,I_THM],
   Induct_on `n` THENL
    [ASM_SIMP_TAC arith_ss [SUM_def_compute],
     ASM_SIMP_TAC arith_ss
       [ADD_CLAUSES,SUM_FOLDL,MULT_CLAUSES] THEN
     SRW_TAC [ARITH_ss] [COUNT_LIST_SNOC,FOLDL_SNOC]]]
```

Proof inspection

$$2 \times \sum_{x=0}^{n} x = n \times (n+1)$$

```
Induct_on `n`
```

$$2 \times \sum_{x=0}^{0} x = 0 \times (0+1)$$

$$2 \times \sum_{x=0}^{n} x = n \times (n+1) \Rightarrow 2 \times \sum_{x=0}^{n+1} x = (n+1) \times ((n+1)+1)$$

Proof inspection

$$2 \times \sum_{x=0}^{0} x = 0 \times (0+1)$$

```
SRW_TAC [] []
```

$$\sum_{x=0}^{0} x = 0$$

```
METIS_TAC [SUM_1,I_THM]
```

Proof inspection

$$2 \times \sum_{x=0}^{n} x = n \times (n+1) \Rightarrow 2 \times \sum_{x=0}^{n+1} x = (n+1) \times ((n+1)+1)$$

`Induct_on ‘n‘`

$$2 \times \sum_{x=0}^{0} x = 0 \times (0+1) \Rightarrow 2 \times \sum_{x=0}^{1} x = (0+1) \times ((0+1)+1)$$

$$(2 \times \sum_{x=0}^{n} x = n \times (n+1) \Rightarrow 2 \times \sum_{x=0}^{n+1} x = (n+1) \times ((n+1)+1))$$

$$\Rightarrow$$

$$(2 \times \sum_{x=0}^{n+1} x = (n+1) \times ((n+1)+1) \Rightarrow 2 \times \sum_{x=0}^{(n+1)+1} x = ((n+1)+1) \times (((n+1)+1)+1))$$

Proof inspection

$$2 \times \sum_{x=0}^{0} x = 0 \times (0+1) \Rightarrow 2 \times \sum_{x=0}^{1} x = (0+1) \times ((0+1)+1)$$

```
ASM_SIMP_TAC arith_ss [SUM_def_compute]
```

Proof inspection

$$(2\times\sum_{x=0}^{n} x = n\times(n+1) \Rightarrow 2\times\sum_{x=0}^{n+1} x = (n+1)\times((n+1)+1))$$

$$\Rightarrow$$

$$(2\times\sum_{x=0}^{n+1} x = (n+1)\times((n+1)+1) \Rightarrow 2\times\sum_{x=0}^{(n+1)+1} x = ((n+1)+1)\times(((n+1)+1)+1))$$

```
ASM_SIMP_TAC arith_ss
  [ADD_CLAUSES,SUM_FOLDL,MULT_CLAUSES]
```

$$2\times FOLDL\ (\lambda x\ n'.\ n'+x)\ 0\ (COUNT\_LIST\ ((n+1)+1)) =$$

$$2\times n+(n\times(n+1)+1)+1$$

$$\Rightarrow$$

$$2\times FOLDL\ (\lambda x\ n'.\ n'+x)\ 0\ (COUNT\_LIST\ (((n+1)+1)+1)) =$$

$$4\times n+(n\times(n+1)+2)+1+1+1+1$$

```
SRW_TAC [ARITH_ss] [COUNT_LIST_SNOC, FOLDL_SNOC]
```