Hints for AVATAR (and some more)

Martin Suda

Czech Technical University in Prague, Czech Republic

PIWo 2019, Prague, October 2019

“Interactive Theorem Proving” with ATPs

Some people actually use ATPs to do math!

"Interactive Theorem Proving" with ATPs

Some people actually use ATPs to do math!

- e.g., Bob Veroff and Michael Kinyon
- using Otter, Prover9, Mace4
- questions from algebra: axioms bases for boolean algebras, ortho-lattices, loop theory
- targeting open problems (e.g. the AIM conjecture)

"Interactive Theorem Proving" with ATPs

Some people actually use ATPs to do math!

- e.g., Bob Veroff and Michael Kinyon
- using Otter, Prover9, Mace4
- questions from algebra: axioms bases for boolean algebras, ortho-lattices, loop theory
- targeting open problems (e.g. the AIM conjecture)

In what sense interactive?

- a single proof attempt (ATP call) usually does not solve it
- trying different formulations / axiomatizations
- trying various additional assumptions and learning from them

"Interactive Theorem Proving" with ATPs

Some people actually use ATPs to do math!

- e.g., Bob Veroff and Michael Kinyon
- using Otter, Prover9, Mace4
- questions from algebra: axioms bases for boolean algebras, ortho-lattices, loop theory
- targeting open problems (e.g. the AIM conjecture)

In what sense interactive?

- a single proof attempt (ATP call) usually does not solve it
- trying different formulations / axiomatizations
- trying various additional assumptions and learning from them

➥ By the way, these attempts may run for weeks!

Hints

What is a hint?

- a clause supplied by the user as part of the input
- whenever a newly derived clause C subsumes a hint clause, this C is prioritized for selection

Hints

What is a hint?

- a clause supplied by the user as part of the input
- whenever a newly derived clause C subsumes a hint clause, this C is prioritized for selection

➥ Hints are a means for steering the proof search!

Hints

What is a hint?

- a clause supplied by the user as part of the input
- whenever a newly derived clause C subsumes a hint clause, this C is prioritized for selection

➥ Hints are a means for steering the proof search!

Where do hints come from?

- the (expert) user just thinks of some

Hints

What is a hint?

- a clause supplied by the user as part of the input
- whenever a newly derived clause C subsumes a hint clause, this C is prioritized for selection

➥ Hints are a means for steering the proof search!

Where do hints come from?

- the (expert) user just thinks of some
- more realistically: clauses from proofs of similar theorems or of the same theorem but under different assumptions

Hints

What is a hint?

- a clause supplied by the user as part of the input
- whenever a newly derived clause C subsumes a hint clause, this C is prioritized for selection

➥ Hints are a means for steering the proof search!

Where do hints come from?

- the (expert) user just thinks of some
- more realistically: clauses from proofs of similar theorems or of the same theorem but under different assumptions

➥ Hope that similar theorems can be proved using similar intermediate steps.

Hints

What is a hint?

- a clause supplied by the user as part of the input
- whenever a newly derived clause C subsumes a hint clause, this C is prioritized for selection

➥ Hints are a means for steering the proof search!

Where do hints come from?

- the (expert) user just thinks of some
- more realistically: clauses from proofs of similar theorems or of the same theorem but under different assumptions

➥ Hope that similar theorems can be proved using similar intermediate steps.

How to come up with hints automatically?

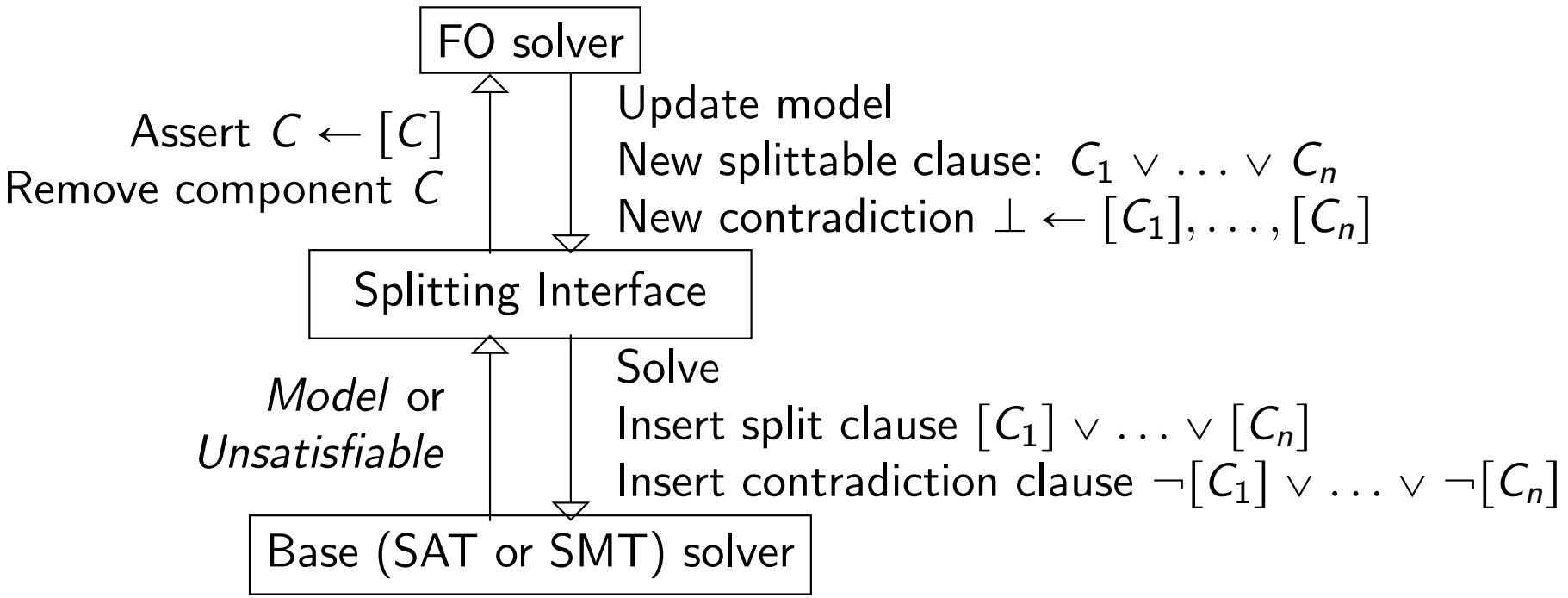
AVATAR: a reminder

AVATAR [Voronkov'14]

- modern architecture of first order theorem provers
- integrates saturation with a SAT solver (or an SMT solver)
- efficient realization of the *clause splitting rule*
- instead of one monolithic proof search
  a sequence of proof searches on (much) smaller sub-problems

- implemented in theorem prover Vampire
- shown highly successful in practice

AVATAR architecture overview



Boosting AVATAR with hints

Instead of waiting for the user to supply hints for problem P . . .

. . . attempt P using AVATAR and collect as hints the first-order parts of the clauses appearing in the sub-proofs of the so far derived contradiction clauses

Boosting AVATAR with hints

Instead of waiting for the user to supply hints for problem P . . .

. . . attempt P using AVATAR and collect as hints the first-order parts of the clauses appearing in the sub-proofs of the so far derived contradiction clauses

DEMO!

Outline

1. Hints for AVATAR
2. An Experiment
3. What is a Significant Improvement?

Outline

1 Hints for AVATAR

2 An Experiment

3 What is a Significant Improvement?

Experimental setup

Vampire setup:

- `--saturation_algorithm discount` (for stability)
- `--age_weight_ratio 1:10` (works well with discount)
- `--time_limit 10` (reasonable time to finish)

Experimental setup

Vampire setup:

- `--saturation_algorithm discount` (for stability)
- `--age_weight_ratio 1:10` (works well with discount)
- `--time_limit 10` (reasonable time to finish)

Computers:

- either Starexec
- or CTU's (slurm) cluster

Experimental setup

Vampire setup:

- `--saturation_algorithm discount` (for stability)
- `--age_weight_ratio 1:10` (works well with discount)
- `--time_limit 10` (reasonable time to finish)

Computers:

- either Starexec
- or CTU's (slurm) cluster

The benchmark:

- TPTP v 7.2.0
- 17573 eligible first-order problems

Results

(on Starexec)

configuration	solved	uniques	additional
base	7914	0	7914
base+hints	7882	2	62
sac	8100	13	299
sac+hints	8106	13	23

base = `-sa discount -awr 10 -t 10`

sac = `--split_at_activation on`

Results

(on Starexec)

configuration	solved	uniques	additional
base	7914	0	7914
base+hints	7882	2	62
sac	8100	13	299
sac+hints	8106	13	23

base = `-sa discount -awr 10 -t 10`
sac = `--split_at_activation on`

Experimented with AVATAR flushing; also not very interesting

Let's try a different benchmark . . .

MIZAR bushy "small"

- 57 880 problems translated from the MIZAR library

Let's try a different benchmark . . .

MIZAR bushy "small"

- 57 880 problems translated from the MIZAR library

(base: `-sa discount -awr 10 -t 10 -sac on`)

Let's try a different benchmark ...

MIZAR bushy "small"

- 57 880 problems translated from the MIZAR library

(base: `-sa discount -awr 10 -t 10 -sac on`)

Results

configuration	solved	uniques
base	14843	184
base+hints	14873	214

Let's try a different benchmark . . .

MIZAR bushy "small"

- 57 880 problems translated from the MIZAR library

(base: `-sa discount -awr 10 -t 10 -sac on`)

Results

configuration	solved	uniques
base	14843	184
base+hints	14873	214

(30 problems is approx. 0.5‰ of the benchmark size)

So, should we be sad and abandon the idea?

So, should we be sad and abandon the idea?

Maybe, but . . .

So, should we be sad and abandon the idea?

Maybe, but . . .

- maybe it only gets interesting with really hard problems!

So, should we be sad and abandon the idea?

Maybe, but . . .

- maybe it only gets interesting with really hard problems!
- maybe we should have a smarter notion of similarity!
  - demodulate hints?

So, should we be sad and abandon the idea?

Maybe, but . . .

- maybe it only gets interesting with really hard problems!
- maybe we should have a smarter notion of similarity!
  - demodulate hints?
- maybe we need restarts to prevent the prover from choking

So, should we be sad and abandon the idea?

Maybe, but . . .

- maybe it only gets interesting with really hard problems!
- maybe we should have a smarter notion of similarity!
  - demodulate hints?
- maybe we need restarts to prevent the prover from choking

- we should also try strengthening the theory with reasonable additional assumptions, as routinely done by Veroff et al.

So, should we be sad and abandon the idea?

Maybe, but . . .

- maybe it only gets interesting with really hard problems!
- maybe we should have a smarter notion of similarity!
  - demodulate hints?
- maybe we need restarts to prevent the prover from choking

- we should also try strengthening the theory with reasonable additional assumptions, as routinely done by Veroff et al.

➥ Ongoing and future work!

Outline

1 Hints for AVATAR

2 An Experiment

3 What is a Significant Improvement?

A Methodology Question

When should we get excited about a new technique?

A Methodology Question

When should we get excited about a new technique?

1. The idea looks clever and sophisticated

A Methodology Question

When should we get excited about a new technique?

1. The idea looks clever and sophisticated
   ➥ Could aim for a pure theory paper at CADE!

A Methodology Question

When should we get excited about a new technique?

1. The idea looks clever and sophisticated
   ➥ Could aim for a pure theory paper at CADE!

2. Solves more problems than baseline

A Methodology Question

When should we get excited about a new technique?

1. The idea looks clever and sophisticated
   ➥ Could aim for a pure theory paper at CADE!

2. Solves more problems than baseline
   ➥ Obviously, this gives us more power!

A Methodology Question

When should we get excited about a new technique?

1. The idea looks clever and sophisticated
   ➥ Could aim for a pure theory paper at CADE!

2. Solves more problems than baseline
   ➥ Obviously, this gives us more power!

3. The solution set differs enough from baseline

A Methodology Question

When should we get excited about a new technique?

1. The idea looks clever and sophisticated
   ➥ Could aim for a pure theory paper at CADE!

2. Solves more problems than baseline
   ➥ Obviously, this gives us more power!

3. The solution set differs enough from baseline
   ➥ To have a chance to improve strategy schedule . . .

Focusing on the Third Point

Proof search is known to be very fragile. Even a small change will "stir" it and create a different solution set.

Focusing on the Third Point

Proof search is known to be very fragile. Even a small change will "stir" it and create a different solution set.

What does it mean to differ enough from baseline?

Focusing on the Third Point

Proof search is known to be very fragile. Even a small change will "stir" it and create a different solution set.

What does it mean to differ enough from baseline?

- Keep a database of problems known to be solvable by some strategy and compare against that.

Focusing on the Third Point

Proof search is known to be very fragile. Even a small change will "stir" it and create a different solution set.

What does it mean to differ enough from baseline?

- Keep a database of problems known to be solvable by some strategy and compare against that.
  ➥ Computationally expensive, open ended, but YES!

Focusing on the Third Point

Proof search is known to be very fragile. Even a small change will "stir" it and create a different solution set.

What does it mean to differ enough from baseline?

- Keep a database of problems known to be solvable by some strategy and compare against that.
  ➥ Computationally expensive, open ended, but YES!

While the database is being built . . .

Focusing on the Third Point

Proof search is known to be very fragile. Even a small change will "stir" it and create a different solution set.

What does it mean to differ enough from baseline?

- Keep a database of problems known to be solvable by some strategy and compare against that.
  ➥ Computationally expensive, open ended, but YES!

While the database is being built . . .

Let's have some random fun!

Randomly permuting the input problem

Use `tptp4X -trandomize` from the TPTP toolset to:

- randomize the order of commutative logical operations
- randomize the order of formulas

Randomly permuting the input problem

Use `tptp4X -trandomize` from the TPTP toolset to:

- randomize the order of commutative logical operations
- randomize the order of formulas

Can we solve more problems?

Randomly permuting the input problem

Use `tptp4X -trandomize` from the TPTP toolset to:

- randomize the order of commutative logical operations
- randomize the order of formulas

Can we solve more problems?

configuration	solved	uniques	additional
straight	8612	53	8612
shuffled1	8773	60	345
shuffled2	8788	85	128
shuffled3	8775	48	48

(now on the CTU cluster)

Randomly permuting the input problem

Use `tptp4X -trandomize` from the TPTP toolset to:

- randomize the order of commutative logical operations
- randomize the order of formulas

Can we solve more problems?

configuration	solved	uniques	additional
straight	8612	53	8612
shuffled1	8773	60	345
shuffled2	8788	85	128
shuffled3	8775	48	48

(now on the CTU cluster)

➥ recalling **randoCoP (Raths, Otten; 2008)**

One more experiment with randomness

Clause Selection and Age-weight Ratio

Vampire alternates between selecting the next given clause by age (old first) and by weight (light first) under a given ratio.

One more experiment with randomness

Clause Selection and Age-weight Ratio

Vampire alternates between selecting the next given clause by age (old first) and by weight (light first) under a given ratio.

Normally, this alternation is regular. What if we change it to probabilistic?

One more experiment with randomness

Clause Selection and Age-weight Ratio

Vampire alternates between selecting the next given clause by age (old first) and by weight (light first) under a given ratio.

Normally, this alternation is regular. What if we change it to probabilistic?

configuration	solved	uniques	additional
base	8725	12	8725
rnd1	8747	8	91
rnd2	8744	16	37
rnd3	8768	23	37
rnd4	8735	14	21
rnd5	8741	16	16

base = `-sa discount -awr 1:1 -t 10`

Summary

Empirical research with an ATP:

1. have a new idea
2. implement (and debug)
3. conduct experiments

Summary

Empirical research with an ATP:

1. have a new idea
2. implement (and debug)
3. conduct experiments

When are the results significant?

- improving overall performance (high total solved)
- solving hard problems ("the uniques")

Summary

Empirical research with an ATP:

1. have a new idea
2. implement (and debug)
3. conduct experiments

When are the results significant?

- improving overall performance (high total solved)
- solving hard problems ("the uniques")

Why don't we use (carefully seeded) randomness to prove more theorems (without much actual extra thinking)?