Prover9 Unleashed: Automated Configuration for Enhanced Proof Discovery

Kristina Aleksandrova[2] Jan Jakubův[1,2] Cezary Kaliszyk[2]

AITP'24, Aussois, France, 3rd September 2024

[1]Czech Technical University in Prague, Prague, Czech Republic
[2]University of Innsbruck, Innsbruck, Austria

Outline

Outline

Prover9: Legendary Theorem Prover

- Prover9 is an automated theorem prover for first-order logic
- . . . based on resolution and paramodulation.

Prover9: Legendary Theorem Prover

- Prover9 is an automated theorem prover for first-order logic
- . . . based on resolution and paramodulation.
- Not developed since 2009 but . . .
- . . . still actively and successfully used by its users.
- . . . still used in the CASC competition (as a reference).

Our Goal

- Prover9 contains a rich language for strategy configuration.
- As opposed to the state-of-the-art theorem provers,

  . . . does not have an automated strategy selection mechanism.

Our Goal

- Prover9 contains a rich language for strategy configuration.
- As opposed to the state-of-the-art theorem provers,

  . . . does not have an automated strategy selection mechanism.
- Goal: Use Grackle automated strategy invention on Prover9.

Our Goal

- Prover9 contains a rich language for strategy configuration.
- As opposed to the state-of-the-art theorem provers,
  . . . does not have an automated strategy selection mechanism.
- Goal: Use Grackle automated strategy invention on Prover9.
- Question: Can Prover9 still compete with state-of-the-art?

Prover9: Proof Search Strategy Configuration

- Supports more than 100 basic options
  . . . (booleans, numerical, categorical).

```
set(binary_resolution).
set(expand_relational_defs).
clear(back_subsume).
assign(order, kbo).
assign(not_weight, -2).
assign(or_weight, 200).
assign(prop_atom_weight, 9999).
```

Prover9: Given Clause Selection Configuration

- Prover9 processes one given clause a time.
- Allows to define which clauses should be preferred,
  . . . using priority queues where clauses are sorted (age/weight)

```
list(given_selection).
   part(queue1, high, age   ,  weight < 500 & -horn  ) = 50.
   part(queue2, high, weight,  depth < 15            ) = 50.
   part(queue3, low , weight,  -false                ) = 5.
   part(rest  , low , random,  all                   ) = 1.
end_of_list.
```

Prover9: Action Rules

- action rules: Change options values on the fly,
  . . . when certain event occur.

```
list(actions).
  generated=5000  -> assign(pick_given_ratio, 4).
end_of_list.
```

Prover9: Keep/Delete Rules

- keep/delete rules: Delete generated clauses (incomplete),
  . . . satisfying a delete condition unless kept is satisfied.

```
list(delete).
    weight > 20 & variables > 4.
    weight > 30.
end_of_list.

list(keep).
    horn & variables<3.
end_of_list.
```

Outline

Grackle: Algorithm Motivation

- Grackles are passerine birds native to North and South America.

Grackle: Algorithm Motivation

- Grackles are passerine birds native to North and South America.
- During evolution, different grackle species developed different bill sizes to be able to feed on different types of nutrients.

Grackle: Algorithm Motivation

- Grackles are passerine birds native to North and South America.
- During evolution, different grackle species developed different bill sizes to be able to feed on different types of nutrients.
- This decreases competition between different species and increases the chances of their survival.

Grackle: Algorithm Overview

- Successor/generalization of BliStr/Tune tools.
- Grackle utilizes existing algorithm configuration frameworks
  - ParamILS: Iterative Local Search (Hutter et al.)
  - SMAC3: Bayesian Optimization (Lindauer et al.)

  to improve a strategy on a given set of problems.

Grackle: Algorithm Overview

- Successor/generalization of BliStr/Tune tools.
- Grackle utilizes existing algorithm configuration frameworks
    - ParamILS: Iterative Local Search (Hutter et al.)
    - SMAC3: Bayesian Optimization (Lindauer et al.)

  to improve a strategy on a given set of problems.
- Grackle input:
    - initial set of strategies
    - input problems
    - strategy space parametrization: parameters and their values
    - solver wrapper

Grackle: Algorithm Overview

- Successor/generalization of BliStr/Tune tools.
- Grackle utilizes existing algorithm configuration frameworks
    - ParamILS: Iterative Local Search (Hutter et al.)
    - SMAC3: Bayesian Optimization (Lindauer et al.)

  to improve a strategy on a given set of problems.
- Grackle input:
    - initial set of strategies
    - input problems
    - strategy space parametrization: parameters and their values
    - solver wrapper
- Grackle output:
    - portfolio of strategies complementary on input problems

Grackle: Invent Portfolio of Strategies

Repeat the following:

1. Evaluate all strategies on all problems P
2. Select one strategy S to be improved
3. Specialize strategy S for the problems where it performs best
4. Go to 1

Terminate when:

- all strategies has been improved, or . . .
- time limit is reached.

Grackle: Encoding advanced Prover9 features

To describe a given clause selection queue:

```
part(queue1, high, age , weight < 500 & -horn ) = 50.
```

we introduce a set of parameters

while fixing the maximal count of parts in the condition (j)

Grackle: Encoding advanced Prover9 features

To describe a given clause selection queue:

```
part(queue1, high, age , weight < 500 & -horn ) = 50.
```

we introduce a set of parameters

while fixing the maximal count of parts in the condition (j)

1. H^{order}: select the sorting criterion (`age`,`weight`,`random`)

Grackle: Encoding advanced Prover9 features

To describe a given clause selection queue:

```
part(queue1, high, age , weight < 500 & -horn ) = 50.
```

we introduce a set of parameters

while fixing the maximal count of parts in the condition (j)

1. H^{order}: select the sorting criterion (`age`,`weight`,`random`)
2. H_j^{cond}: the property of the j-th part in the condition

Grackle: Encoding advanced Prover9 features

To describe a given clause selection queue:

```
part(queue1, high, age , weight < 500 & -horn ) = 50.
```

we introduce a set of parameters

while fixing the maximal count of parts in the condition (j)

1. H^{order}: select the sorting criterion (`age`,`weight`,`random`)
2. H_j^{cond}: the property of the j-th part in the condition
3. $H_j^{connect}$: the logical connective between condition parts

Grackle: Encoding advanced Prover9 features

To describe a given clause selection queue:

```
part(queue1, high, age , weight < 500 & -horn ) = 50.
```

we introduce a set of parameters

while fixing the maximal count of parts in the condition (j)

1. H^{order}: select the sorting criterion (`age`,`weight`,`random`)
2. H_j^{cond}: the property of the j-th part in the condition
3. $H_j^{connect}$: the logical connective between condition parts
4. H_j^{val}: the numeric value to compare the property with

Grackle: Encoding advanced Prover9 features

To describe a given clause selection queue:

```
part(queue1, high, age , weight < 500 & -horn ) = 50.
```

we introduce a set of parameters

while fixing the maximal count of parts in the condition (j)

1. H^{order}: select the sorting criterion (`age`,`weight`,`random`)
2. H_j^{cond}: the property of the j-th part in the condition
3. $H_j^{connect}$: the logical connective between condition parts
4. H_j^{val}: the numeric value to compare the property with
5. H_j^{neg}: is the condition negated (`yes` or `no`)

Grackle: Encoding advanced Prover9 features

To describe a given clause selection queue:

```
part(queue1, high, age , weight < 500 & -horn ) = 50.
```

we introduce a set of parameters

while fixing the maximal count of parts in the condition (j)

1. H^{order}: select the sorting criterion (`age`,`weight`,`random`)
2. H_j^{cond}: the property of the j-th part in the condition
3. $H_j^{connect}$: the logical connective between condition parts
4. H_j^{val}: the numeric value to compare the property with
5. H_j^{neg}: is the condition negated (`yes` or `no`)
6. H^{ratio}: the queue selection ratio

Grackle: Encoding advanced Prover9 features

Similarly, we encode other Prover9 advanced features:

- keep/delete rules: deleting generated clauses (incomplete)
- actions: dynamic changing of parameter values

Grackle: Encoding advanced Prover9 features

Similarly, we encode other Prover9 advanced features:

- keep/delete rules: deleting generated clauses (incomplete)
- actions: dynamic changing of parameter values

This gives rise to 2 Grackle domains:

- default domain: only basic options (value-based)
- full domain: basic + advanced features

Outline

Experiments Setup

- AIM benchmark used in the CASC 2016 ATP competition
  ...divided into 1020 training and 200 testing problems
  ...from a large theorem proving project in loop theory.

Experiments Setup

- AIM benchmark used in the CASC 2016 ATP competition
  ...divided into 1020 training and 200 testing problems
  ...from a large theorem proving project in loop theory.
- We start with 5 simple Prover9 strategies.

Experiments Setup

- AIM benchmark used in the CASC 2016 ATP competition
  . . . divided into 1020 training and 200 testing problems
  . . . from a large theorem proving project in loop theory.
- We start with 5 simple Prover9 strategies.
- We run Grackle for 24 hours on training problems.

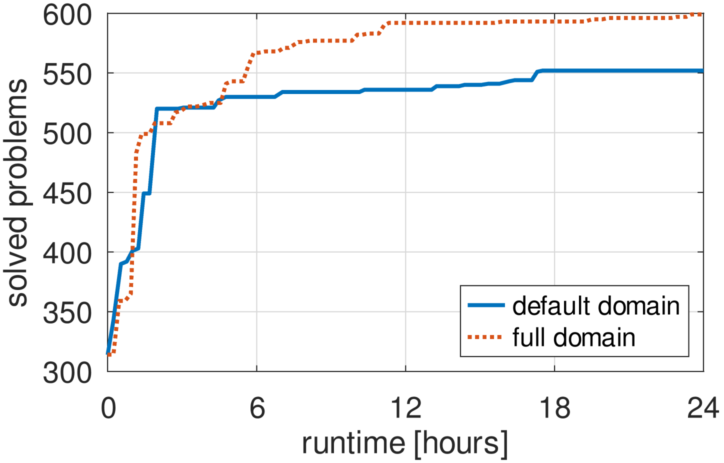
Experiments Setup

- AIM benchmark used in the CASC 2016 ATP competition
  ... divided into 1020 training and 200 testing problems
  ... from a large theorem proving project in loop theory.
- We start with 5 simple Prover9 strategies.
- We run Grackle for 24 hours on training problems.
- The 10 best strategies are evaluated on testing problems,
  ... each with a time limit of 30 seconds ($30 * 10 = 300$).

Experiments Setup

- AIM benchmark used in the CASC 2016 ATP competition
  ...divided into 1020 training and 200 testing problems
  ...from a large theorem proving project in loop theory.
- We start with 5 simple Prover9 strategies.
- We run Grackle for 24 hours on training problems.
- The 10 best strategies are evaluated on testing problems,
  ...each with a time limit of 30 seconds ($30 * 10 = 300$).
- This portfolio is compared with E and Vampire's portfolios
  ...with a compatible time limit of 300 seconds.

Grackle run on AIM train problems



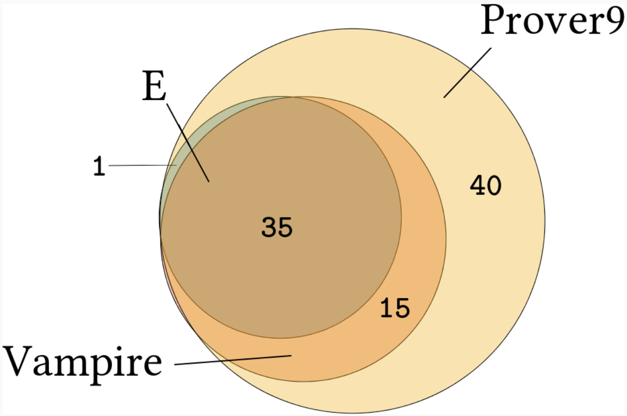
Results on AIM test problems

prover	*solves*
Prover9/Grackle	**91**
Vampire/casc-mode	50
E/auto-schedule	36

Results on AIM test problems

prover	*solves*
Prover9/Grackle	**91**
Vampire/casc-mode	50
E/auto-schedule	36
Prover9/auto	*41*

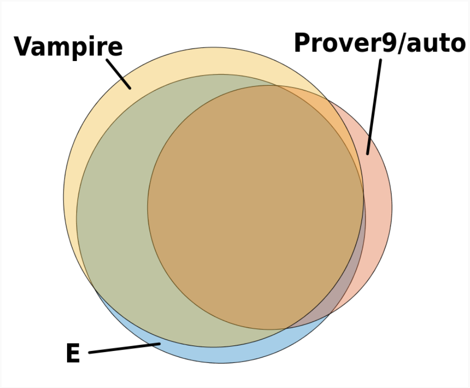
Results on 200 AIM testing problems



Outline

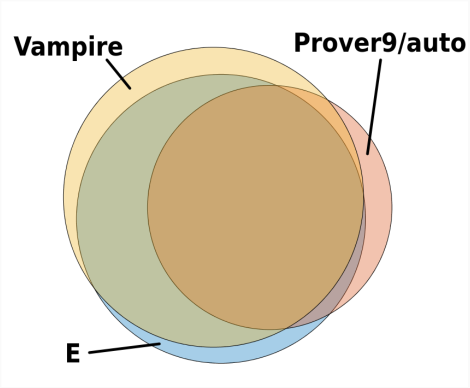
Initial Evaluation on Whole TPTP

- Q: On which TPTP category Prover9/auto performs best?



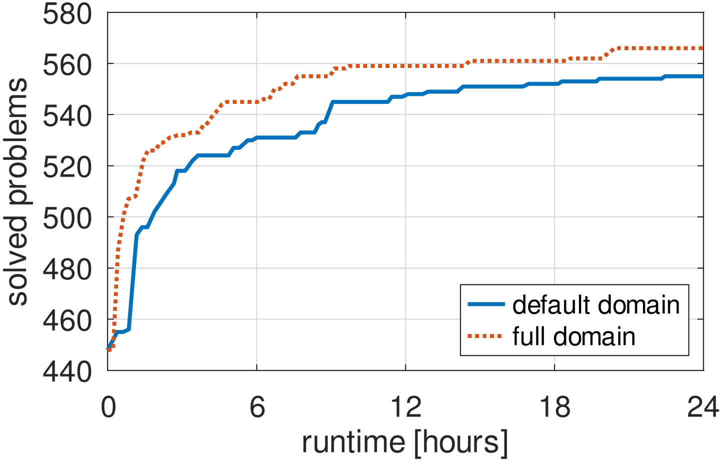
Initial Evaluation on Whole TPTP

- Q: On which TPTP category Prover9/auto performs best?



- A: On TPTP/NUM category.

Grackle run on TPTP/NUM problems



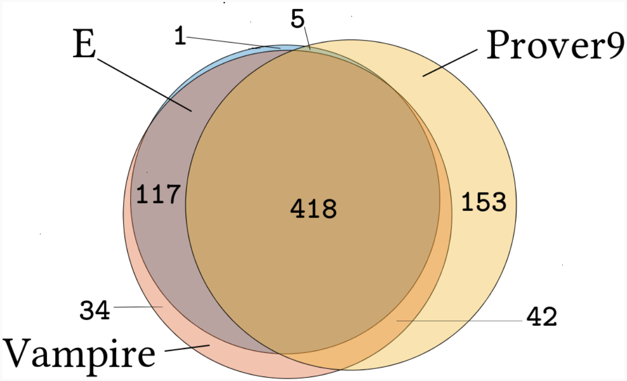
Results on TPTP/NUM problems

prover	*solves*
Prover9/Grackle	**618**
Vampire/casc-mode	611
E/auto-schedule	541

Results on TPTP/NUM problems

prover	*solves*
Prover9/Grackle	**618**
Vampire/casc-mode	611
E/auto-schedule	541
Prover9/auto	*519*

Results on TPTP/NUM problems



Thank you for listening.

Future work:

- Run Grackle for Prover9 on other TPTP categories.
- Intelligent problem-based strategy selection.
- Grackle strategy invention with Prover9 hints.

Grackle is available at GitHub:

`http://github.com/ai4reason/grackle`

Questions?