Checkable Proofs for First-Order Theorem Proving

Giles Reger[1], Martin Suda[2]

[1]School of Computer Science, University of Manchester, UK
[2]TU Wien, Vienna, Austria

ARCADE 2017 – Gothenburg, August 6, 2017

Why do we want proofs?

- (explain the result to humans)

Why do we want proofs?

- (explain the result to humans)
- certify correctness
  - increase confidence in the result
  - debugging of the prover itself

Why do we want proofs?

- (explain the result to humans)
- certify correctness
  - increase confidence in the result
  - debugging of the prover itself
- primary output to be further processed
  - visualisation, interpolation, ...
  - combination of systems

Why do we want proofs?

- (explain the result to humans)
- certify correctness
  - increase confidence in the result
  - debugging of the prover itself
- primary output to be further processed
  - visualisation, interpolation, ...
  - combination of systems
- hammers

  = interactive theorem prover tactics employing an ATP

Why do we want proofs?

- (explain the result to humans)
- certify correctness
  - increase confidence in the result
  - debugging of the prover itself
- primary output to be further processed
  - visualisation, interpolation, ...
  - combination of systems
- hammers

  = interactive theorem prover tactics employing an ATP

Automatically checkable / with formal semantics

Current situation in the ATP world

Thousands of Solutions from Theorem Provers

```
fof(c_0_11,plain,(
    ! [X2] :
      ( ~ lives(X2)
      | X2 = agatha
      | X2 = butler
      | X2 = charles ) ),
    inference(variable_rename,[status(thm)],[inference(fof_nnf,[status(thm)],[pel55_3])])).

fof(c_0_12,plain,
    ( lives(esk1_0)
    & killed(esk1_0,agatha) ),
    inference(skolemize,[status(esa)],[inference(variable_rename,[status(thm)],[pel55_1])])).
```

- TPTP syntax and fixed conventions

Current situation in the ATP world

Thousands of Solutions from Theorem Provers

```
fof(c_0_11,plain,(
    ! [X2] :
      ( ~ lives(X2)
      | X2 = agatha
      | X2 = butler
      | X2 = charles ) ),
    inference(variable_rename,[status(thm)],[inference(fof_nnf,[status(thm)],[pel55_3])])).

fof(c_0_12,plain,
    ( lives(esk1_0)
    & killed(esk1_0,agatha) ),
    inference(skolemize,[status(esa)],[inference(variable_rename,[status(thm)],[pel55_1])])).
```

- TPTP syntax and fixed conventions
- lack of formal semantics precludes reliable proof-checking

Current situation in the ATP world

Thousands of Solutions from Theorem Provers

```
fof(c_0_11,plain,(
    ! [X2] :
      ( ~ lives(X2)
      | X2 = agatha
      | X2 = butler
      | X2 = charles ) ),
    inference(variable_rename,[status(thm)],[inference(fof_nnf,[status(thm)],[pel55_3])])).

fof(c_0_12,plain,
    ( lives(esk1_0)
    & killed(esk1_0,agatha) ),
    inference(skolemize,[status(esa)],[inference(variable_rename,[status(thm)],[pel55_1])])).
```

- TPTP syntax and fixed conventions
- lack of formal semantics precludes reliable proof-checking
- proof reconstruction in hammers may fail for various reasons

Current situation in the ATP world

Thousands of Solutions from Theorem Provers

```
fof(c_0_11,plain,(
    ! [X2] :
      ( ~ lives(X2)
      | X2 = agatha
      | X2 = butler
      | X2 = charles ) ),
    inference(variable_rename,[status(thm)],[inference(fof_nnf,[status(thm)],[pel55_3])])).

fof(c_0_12,plain,
    ( lives(esk1_0)
    & killed(esk1_0,agatha) ),
    inference(skolemize,[status(esa)],[inference(variable_rename,[status(thm)],[pel55_1])])).
```

- TPTP syntax and fixed conventions
- lack of formal semantics precludes reliable proof-checking
- proof reconstruction in hammers may fail for various reasons

Independent reproving of logical entailments is still very useful

An ideal proof format

An ideal proof format

General accommodates all known techniques: superposition, InstGen, ...

An ideal proof format

General accommodates all known techniques:
superposition, InstGen, ...

Ideally "open-ended" = extendable

An ideal proof format

General accommodates all known techniques:
superposition, InstGen, ...

Ideally "open-ended" = extendable

Not just entailments preprocessing and "unsound" steps:
Skolemization, naming, symmetry breaking, ...

An ideal proof format

General accommodates all known techniques:
superposition, InstGen, ...

Ideally "open-ended" = extendable

Not just entailments preprocessing and "unsound" steps:
Skolemization, naming, symmetry breaking, ...

Efficiency of checking
ideally low order poly-time

An ideal proof format

General accommodates all known techniques:
superposition, InstGen, ...

Ideally "open-ended" = extendable

Not just entailments preprocessing and "unsound" steps:
Skolemization, naming, symmetry breaking, ...

Efficiency of checking
ideally low order poly-time

Easy implementation and low runtime overhead

An ideal proof format

General accommodates all known techniques:
superposition, InstGen, ...

Ideally "open-ended" = extendable

Not just entailments preprocessing and "unsound" steps:
Skolemization, naming, symmetry breaking, ...

Efficiency of checking
ideally low order poly-time

Easy implementation and low runtime overhead

General adoption
accepted by the community, supported by major tools

Other communities: previous and related work

Other communities: previous and related work

DRAT — propositional SAT

- surprisingly general / beyond entailment
- small set of rules / efficient checking

Other communities: previous and related work

DRAT propositional SAT

- surprisingly general / beyond entailment
- small set of rules / efficient checking

CeTA termination community

- translation to higher-order formalism (Isabelle/HOL)
- extendable (IsaFoR library)
- efficient checking (via code generation support)

Other communities: previous and related work

DRAT propositional SAT

- surprisingly general / beyond entailment
- small set of rules / efficient checking

CeTA termination community

- translation to higher-order formalism (Isabelle/HOL)
- extendable (IsaFoR library)
- efficient checking (via code generation support)

Dedukti "A universal proof checker"

- target logic: $\lambda\Pi$-*calculus modulo*
- proof checker, translator
- already used to encode superposition and resolution

What is happening at the SMT side?

What is happening at the SMT side?

LFSC [Stump et al.]

- LF $\approx \lambda\Pi$-calculus
- SC = Side Conditions
  (small custom programming language)
- used by CVC4

What is happening at the SMT side?

LFSC [Stump et al.]

- LF $\approx \lambda\Pi$-calculus
- SC = Side Conditions
  (small custom programming language)
- used by CVC4

A Flexible Proof Format for SMT [Besson et al. 11]

- syntax by the SMT-LIB 2.0
- veriT
- framework for formula processing [CADE17]

What is happening at the SMT side?

LFSC [Stump et al.]

- LF $\approx \lambda\Pi$-calculus
- SC = Side Conditions
  (small custom programming language)
- used by CVC4

A Flexible Proof Format for SMT [Besson et al. 11]

- syntax by the SMT-LIB 2.0
- veriT
- framework for formula processing [CADE17]

Proofs and refutations, and Z3 [de Moura & Bjørner 08]

- reports on memory overhead / performance slowdown
- proof reconstruction is challenging [Böhme 09]

Is it a technical problem?

Why don't we have the nice proofs yet? [BMF15]:

1. low priority of the proof output effort amongst other development tasks,
2. differences of opinion on what features should be included in the standard,
3. and the overhead connected with switching from the currently adopted approach to a different one.

Wrapping up

Two kinds of obstacles:

- technical
- societal

Which obstacle is bigger?

Wrapping up

Two kinds of obstacles:

- technical
- societal

Which obstacle is bigger?

> Competitions help!
>
> — Common knowledge

Could they help more? Should competitions require checkable proofs?

Can ARCADE help? We need a community-led approach, you are the community.

Soliciting expert opinion

Q1

What are the main hurdles preventing us from having *Checkable Proofs for First-Order Theorem Proving*?

Q2

What should be the next steps to see this challenge realized in the near future?

Q3

Is more research on the theoretical side needed, or are we simply struggling because too many people would need to agree on too many details and commit to the subsequently?