System Feature Description: Importing Refutations into the GAPT Framework

PxTP Workshop, Manchester

Cvetan Dunchev, Alexander Leitsch, Tomer Libal, Martin Riener, Mikheil Rukhaia, Daniel Weller and Bruno Woltzenlogel-Paleo

June 30, 2012

Introduction

Context: GAPT Framework

GAPT = General Architecture for Proofs and Theorems

provides (in different stages of development):

- Languages
  (Typed Lambda Calculus, First Order Logic, Higher Order Logic)
- Calculi (various Sequent Calculi, Resolution)
- Algorithms (Unification, Matching, . . .)
- Interactive theorem prover (TAP)
- Proof Transformations
  (Proof Skolemization, Cut-elimination by Resolution, Herbrand Sequent Extraction)

Short overview of the CERES method

- Preprocessing of the input Sequent Calculus proof (Skolemization, Regularization)
- Extraction of the characteristic clause set
- Refutation of the characteristic clause set by an external resolution theorem prover
- Constructing proof projections to clauses from the characteristic clause set
- Constructing a proof in atomic-cut normal form from the refutation and the projections

Importing a Proof

Problems with Proof Parsing

- Variable renaming
- Substitutions not given
- Variety of inference rules
- Contraction of several inferences into one
- Incomplete or outdated documentation of the inference rules

Example

Clause set: $\{\ \vdash P(a); P(x) \vdash P(f(x));\ \vdash f(x) = g(x); P(g(a)) \vdash\ \}$

Refutation:

$$\dfrac{\dfrac{\dfrac{\vdash P(a) \qquad P(x) \vdash P(f(x))}{\vdash P(f(a))}\, Res\ \sigma = \{x \mapsto a\} \qquad \vdash f(x) = g(x)}{\vdash P(g(a))}\, Para\ \sigma = \{x \mapsto a\} \qquad P(g(a)) \vdash}{\vdash}\, Res$$

Example

Prover9:

```
1 f(x) = g(x). [assumption].
2 -P(x) | P(f(x)). [assumption].
3 -P(x) | P(g(x)). [copy(2),rewrite([1(2)])].
4 P(a). [assumption].
5 -P(g(a)). [assumption].
6 P(g(a)). [hyper(3,a,4,a)].
7 $F. [resolve(6,a,5,a)].
```

Example

SPASS:

```
1[0:Inp] || -> equal(g(U),f(U))**.
2[0:Inp] P(U) || -> P(f(U))*.
3[0:Inp] || -> P(a)*.
4[0:Inp] || P(g(a))* -> .
5[0:Rew:1.0,4.0] || P(f(a))* -> .
7[0:Res:2.1,5.0] P(a) || -> .
8[0:SSi:7.0,3.0] || -> .
```

Example

Vampire:

```
7. $false (1:0) [subsumption resolution 6,3]
3. 'P'(a) (0:2) [input]
6. ~'P'(a) (1:2) [resolution 5,4]
4. ~'P'(g(a)) (0:3) [input]
5. 'P'(g(X0)) | ~'P'(X0) (0:5) [definition unfolding 2,1]
1. f(X0) = g(X0) (0:5) [input]
2. 'P'(f(X0)) | ~'P'(X0) (0:5) [input]
```

Example

Vampire TPTP output:

```
fof(f7,plain,($false),
  inference(subsumption_resolution,[],[f6,f3])).
fof(f3,axiom,('P'(a)),
  file('simple.tptp',unknown)).
fof(f6,plain,(~'P'(a)),
  inference(resolution,[],[f5,f4])).
fof(f4,axiom,(~'P'(g(a))),
  file('simple.tptp',unknown)).
fof(f5,plain,(( ! [X0] : ('P'(g(X0)) | ~'P'(X0)) )),
  inference(definition_unfolding,[],[f2,f1])).
fof(f1,axiom,(( ! [X0] : (f(X0) = g(X0)) )),
  file('simple.tptp',unknown)).
fof(f2,axiom,(( ! [X0] : ('P'(f(X0)) | ~'P'(X0)) )),
  file('simple.tptp',unknown)).
```

Common Structure

- Inference label by: clause id, premise ids, clause, rule name

Problem

- Parse proof of an external resolution prover
- Fill in missing information
- Normalize proof to use only resolution and paramodulation

Approach

- Extract premises and target clause from proof step
- Use internal prover TAP to reprove each single step (forward resolution)
- Construct full refutation from the steps

The TAP Prover

- Simple resolution prover
- Intended for interactive use and experiments
- Commands based

TAP Internals

- Configuration: State + Commands Queue + Data
- State: clause set + guidance map
- Command: Function from configuration to list of successor states (possibly empty)
- Data: information passed only to following command, not stored in state

Commands for original use (interactive theorem prover)

- Resolve
- Paramodulation
- Factor
- Variants
- DeterministicAnd

- SetStream
- SetTargetClause
- InsertResolvent
- RefutationReached

Changes for Replaying

Changes

- Store resolution proofs instead of clauses
- Add new commands: Prover9Init, Replay, guidance commands

Prover9Init

- Pass clause set to theorem prover and parse result
- Schedule InsertResolvent and AddGuidedInitialClause command for each assumption
- Schedule Factor command for each factoring inference
- Schedule Replay command for every other inference step

Changes for Replaying

Replay

- Create new TAP instance
- Get proofs for premise clauses from guidance map
- Schedule SetClauseWithProof command for the premise clauses
- Schedule SetTargetClause command for the target clause
- Initialize prover to use Resolution and Paramodulation for proof search
- Start proof search
- Add proof found to guidance map and schedule InsertResolvent command for proof of target clause

Changes for Replaying

Guidance Map Management

- SetClauseWithProof
- AddGuidedInitialClause
- AddGuidedClauses
- GetGuidedClauses
- IsGuidedNotFound

Example

Command Queue after Prover9Init

AddGuidedInitialClause(1, List($=(f(x), g(x))$))
InsertResolvent
AddGuidedInitialClause(2, List($\neg P(x), P(f(x))$))
InsertResolvent
Replay(List(0, 2, 1))
AddGuidedInitialClause(4, List($P(a)$))
InsertResolvent
AddGuidedInitialClause(5, List($\neg P(g(a))$))
InsertResolvent
Replay(List(0, 3, 4))
Replay(List(0, 6, 5))

Replayed Example

$$
\dfrac{
\vdash P(a)
\qquad
\dfrac{
\dfrac{
\dfrac{\dfrac{\vdash f(x) = g(x)}{\vdash f(x_6) = g(x_6)}\ \textit{Variant}
\qquad
\dfrac{P(x) \vdash P(f(x))}{P(x_5) \vdash P(f(x_5))}\ \textit{Variant}}
{P(x_5) \vdash P(g(x_5))}\ \textit{Para}\ \sigma = \{x_6 \mapsto x_5\}
}{P(x_{10}) \vdash P(g(x_{10}))}\ \textit{Variant}
}{}
}{\vdash P(g(a))}\ \textit{Res}\ \sigma = \{x_{10} \mapsto a\}
$$

$$
\dfrac{\vdash P(g(a)) \qquad P(g(a)) \vdash}{\vdash}\ \textit{Res}
$$

Pitfalls

Pitfalls

- Forward reasoning prevents some strategies
  - Factorization can not only be applied after an inference step
  - No reflexivity rule: add reflexivity axiom or unfold rule
  - Equations might get flipped
  - Expectation that a single inference is provable in few steps not met

Conclusion and Future Work

- Normalized proof with instantiations needed for cut-elimination and Herbrand sequent extraction
- Replay of Prover9 proofs works for small examples, performance issues for larger ones
- Macro rules with large numbers of premises need specialized handling (necessary for Vampire/SPASS/E/... integration)

Thanks for the attention!