A Premise Selection Algorithm for **Apia**

Alejandro Calle-Saldarriaga

Universidad EAFIT

acalles@eafit.edu.co

Logic and Computation Seminar

September 13, 2016

Content

Abstract

Automathic theorem provers need to recieve a reasonably small number of premises in order for them to be able to prove a given conjecture with limited processor time. In large theories this is not always possible, as many irrelevant clauses are added to the premises. In order to solve this problem, premise selection algorithms have emerged in the past few years, some using non-learning methods and others using learning ones. Our goal in this project is to implement a non-learning premise selection algorithm for Apia, in order to further link the interactive theorem prover Agda with Authomatic theorem provers.

Proof Assistants

- Software tool developed that assists in the development of formal proofs.
- It checks proofs, i.e., it automatically verifies mathematical theories by formalizing the definitions, the axioms and the proofs, and then checks the well-formedness of the definitions and the correctness of the proofs within a given logic (Barendregt and Geuvers, 2001).
- Many proof assistants: **Agda**[1], **Coq**[2], **Isabelle**[3], **Hol Light**[4], etc.

[1] http://wiki.portal.chalmers.se/agda/pmwiki.php
[2] https://coq.inria.fr/
[3] `https://isabelle.in.tum.de/`
[4] `http://www.cl.cam.ac.uk/~jrh13/hol-light/`

Example

```
module Test where

data _∨_ (A B : Set) : Set where
  inj₁ : A → A ∨ B
  inj₂ : B → A ∨ B

∨-comm : {A B : Set} → (A ∨ B) → (B ∨ A)
∨-comm (inj₁ p) = inj₂ p
∨-comm (inj₂ q) = inj₁ q
```

Automatic Theorem Provers

- ATPs are computer programs that prove mathematical theorems.
- They show that the conjecture is the logical consequence of a set of statements (the axioms).
- There are many ATPs: **E**[5], **Vampire**[6], etc.
- Usually solve problems written in TPTP format.

[5] http://wwwlehre.dhbw-stuttgart.de/ sschulz/E/E.html
[6] http://www.vprover.org/

Example

```
$ cat problem.p
fof(def1,axiom,
    a = foo(a,b)).

fof(def2, axiom,
   b = foo(a,b)).

fof(prove,conjecture,
    foo(a,b) = foo(b,a)).

$ eprover --auto --tptp3-format problem.p
```

A Quick Introduction to Hammers

- Link between proof assistants and ATPs.
- Proof assistant → Premise selector → ATP → Proof assistant.
- Three main components: premise selector, translation module and proof reconstruction module.
- Some hammers: **Sledgehammer**[7] (**Isabelle**) and **Hol(y) Hammer**[8] (**Hol Light**).

[7] `http://isabelle.in.tum.de/website-Isabelle2009-1/sledgehammer.html`
[8] `http://cl-informatik.uibk.ac.at/software/hh/`

Premise Selection

- Goal: identify n premises that may be relevant to prove a given conjecture.
- Removes irrelevant clauses and helps the ATPs be faster.
- Different type of methods: non-learning (rely on human-constructed heuristics) and learning (uses machine learning on available proofs).

Non-learning Methods

- **MePo** (Meng and Paulson, 2009) algorithm. Keeps track of a set of relevant symbols and ranks facts by their amount of relevant symbols. It computes each fact scores by $r/(r+i)$, where r is the number of relevant symbols and i the number of irrelevant symbols. It selects all perfect scoring facts and some top scoring facts, and add their symbols to the set of relevant symbols.
- Kryštof and Voronokov (2011) proposed the **SiNE** algorithm. It uses global frequencies of symbols to define their generality and build a relation between each symbol s and all formulas ϕ. Premise selection is done by recursively following this defined relation, starting with the conjecture's symbols.

Learning Methods

- More sophisticated. They have a training phase in which the algorithm searches for a function that explains the training data. In the ATP context they train on all existing proofs.
- Naive Bayes: It uses an strong independence assumption between proofs. It computes the probability that a fact f is used to prove a conjecture c. Can be found in Kühlwein et al. (2013).
- The k nearest neighbors method computes the k nearest previous example given a defined distance. It ranks them based on distance, meaning that a premise needed for proofs of nearby theorems ranks higher. Kaliszyk and Urban (2013) use a weighted modification.

Apia

- **Apia** proves first-order theorem written in **Agda**.
- It translates formulas to TPTP language so ATPs can use them.
- Potential hammer.
- It does not have a premise selector. Actually, they are no premise selectors for type-theory based proof assistants like **Agda**!
- You need to use the ATP pragma in your .agda file.

ATP Pragma in **Apia**

Example

```
module Test2 where

data _∧_ (A : Set) (B : Set) : Set where
  ∧-inj : A → B → (A ∧ B)

_⇔_ : (P : Set) → (Q : Set) → Set
A ⇔ B = (A → B) ∧ (B → A)

postulate
  A B C : Set
  ∧-assoc : ((A ∧ B) ∧ C) ⇔  (A ∧ (B ∧ C))
{-# ATP prove ∧-assoc  #-}
```

Example

```
$ agda Test2.agda
$ apia --atp=e Test2.agda
Proving the conjecture in /tmp/Test2/11-8743-assoc.fof
E 1.9 Sourenee proved the conjecture
```

Objectives

- General: implement an **Apia** module for premise selection.
- Specific: compare learning and non-learning methods, save time in the selection of premises for ATPs, and help towards the construction of a hammer for **Agda**.

Scope

- Implement a simple premise selection algorithm for **Apia** using non-learning methods.
- In future work we would like to implement one using learning methods.

References

Barendregt, H. and Geuvers, H. (2001). Proof-assistants using Dependent Type Systems. In Robinson, A. and Voronokov, A., editors, *Handbook of Automated Reasoning*, chapter 18, pages 1151–1238. Elsevier Science Publishers, Elsevier.

Kaliszyk, C. and Urban, J. (2013). Stronger Automation for Flyspeck by Feature Weighting and Strategy Evolution. In Blanchette, J. C. and Urban, J., editors, *PxTP 2013. Third International Workshop on Proof Exchange for Theorem Proving*, volume 14 of *EPiC Series in Computing*, pages 87–95. EasyChair.

Kryštof, H. and Voronokov, A. (2011). Sine Qua Non for Large Theory Reasoning. In Bjørner, N. and Sofronie-Stokkermans, V., editors, *Automated Deduction – CADE-23*, chapter 23, pages 299–314. Springer Berlin Heidelberg, Springer.

Kühlwein, D., Blanchette, J. C., Kaliszyk, C., and Urban, J. (2013). MaSh: Machine Learning for Sledgehammer. In Blazy, S., Pauline-Mohring, C., and Pichardie, D., editors, *Interactive Theorem Proving*, chapter 5, pages 35–50. Springer Berlin Heidelberg, Springer.

Meng, J. and Paulson, L. C. (2009). Lightweight Relevance Filtering for Machine-Generated Resolution Problems. *Journal of Applied Logic*, 7(1):41–57.

Thanks!