

Reinforcement Learning of Theorem Proving

a paper by

Cezary Kaliszyk, Josef Urban, Henryk Michalewski and Miroslav Olšák

presented by

Dobrik Georgiev

Overview

- 1 How tableau provers work
- 2 Reinforcement Learning and Application to TP
 - Basics
 - Application to TP
 - Results
- 3 Summary

1 How tableau provers work

2 Reinforcement Learning and Application to TP

- Basics
- Application to TP
- Results

3 Summary

How tableau provers work

Clauses:

$$c_1 : P(X)$$

$$c_2 : R(X, Y) \vee \neg P(X) \vee Q(Y)$$

$$c_3 : S(X) \vee \neg Q(b)$$

$$c_4 : \neg S(X) \vee \neg Q(X)$$

$$c_5 : \neg Q(X) \vee \neg R(a, X)$$

$$c_6 : \neg R(a, X) \vee Q(X)$$

How tableau provers work

Clauses:

$$c_1 : P(X)$$

$$c_2 : R(X, Y) \vee \neg P(X) \vee Q(Y)$$

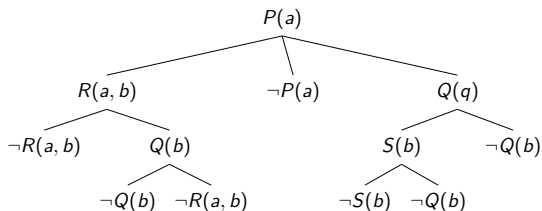
$$c_3 : S(X) \vee \neg Q(b)$$

$$c_4 : \neg S(X) \vee \neg Q(X)$$

$$c_5 : \neg Q(X) \vee \neg R(a, X)$$

$$c_6 : \neg R(a, X) \vee Q(X)$$

Figure: A closed tableau



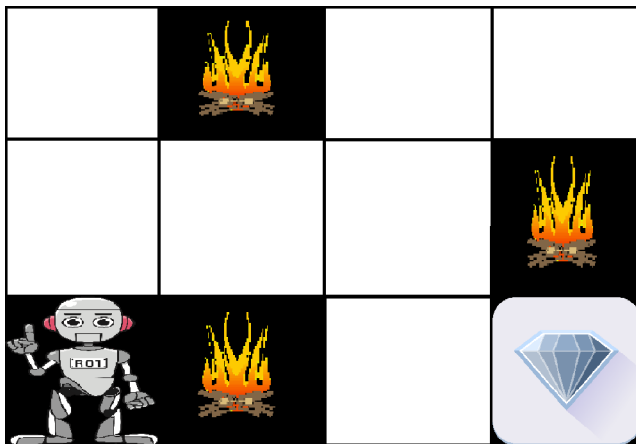
1 How tableau provers work

2 Reinforcement Learning and Application to TP

- Basics
- Application to TP
- Results

3 Summary

RL Basics



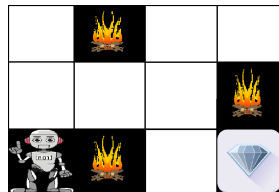
- policy learning
- value learning

Figure: An agent has to reach a reward without burning

Application to TP

RL to TP mapping:

- agent \leftrightarrow TP
- environment \leftrightarrow search tree
- actions \leftrightarrow extending search tree
- reward \leftrightarrow finding a closed tableau



Application to TP – the UCT formula

Tree search with RL – use the UCT formula!

For each node i :

$$f_i = \frac{w_i}{n_i} + c \cdot p_i \cdot \sqrt{\frac{\ln N_i}{n_i}}$$

On every step, take the node with highest f_i .

w_i : **total reward**

n_i : number node of visits

c : hyperparameter

p_i : **prior probability**

N_i : total parent visits

Application to TP – Extracting features (Literals)

- For each Literal L , e.g.
 $f(X, Y) = g(sk_1, sk_2(X))$
- Build it's feature tree
- Count term walks of length 3
- E.g. for L we get $\{(\oplus, =, f) : 1, (=, f, \otimes) : 2, \dots\}$

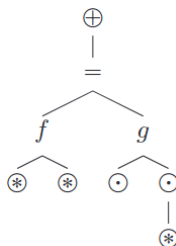


Figure: Feature Tree for L

³Example and picture from Jakubuv and Urban (2017)

Application to TP – Extracting features

- Features for a *clause* – union of features for literals

Application to TP – Extracting features

- Features for a *clause* – union of features for literals
- Feature vector for a *state*:
 - Features of clauses and goals
 - additional metadata – No. of open goals, depth of node, etc.

Application to TP – Extracting features

- Features for a *clause* – union of features for literals
- Feature vector for a *state*:
 - Features of clauses and goals
 - additional metadata – No. of open goals, depth of node, etc.
- Features for an *action* contains:
 - features of the clause used
 - features of literal used

Application to TP – Learning the parameters

- Start with unrestricted Monte-Carlo TP runs:

$$f_i = \frac{w_i}{n_i} + c \cdot p_i \cdot \sqrt{\frac{\ln N_i}{n_i}}$$

Application to TP – Learning the parameters

- Start with unrestricted Monte-Carlo TP runs:

$$f_i = \frac{w_i}{n_i} + c \cdot p_i \cdot \sqrt{\frac{\ln N_i}{n_i}}$$

- Learn action a relevance given f_s and f_a

- $r_a = \frac{\text{overall frequency of } a}{\text{action frequency at node}}$
- $r_a \in (0, \infty)$
- $p_i = \text{softmax}(r_a, R)$

Application to TP – Learning the parameters

- Start with unrestricted Monte-Carlo TP runs:

$$f_i = \frac{w_i}{n_i} + c \cdot p_i \cdot \sqrt{\frac{\ln N_i}{n_i}}$$

- Learn action a relevance given f_s and f_a
 - $r_a = \frac{\text{overall frequency of } a}{\text{action frequency at node}}$
 - $r_a \in (0, \infty)$
 - $p_i = \text{softmax}(r_a, R)$
- Associate node state feature with value
 - 0 if node not a proof
 - $0.99^{\text{proof depth}}$ otherwise

Application to TP – Learning the parameters

- Start with unrestricted Monte-Carlo TP runs:

$$f_i = \frac{w_i}{n_i} + c \cdot p_i \cdot \sqrt{\frac{\ln N_i}{n_i}}$$

- Learn action a relevance given f_s and f_a
 - $r_a = \frac{\text{overall frequency of } a}{\text{action frequency at node}}$
 - $r_a \in (0, \infty)$
 - $p_i = \text{softmax}(r_a, R)$
- Associate node state feature with value
 - 0 if node not a proof
 - $0.99^{\text{proof depth}}$ otherwise
- Apply regression on the logits to learn prior probability and value
- **Deduce**, **Learn** from it, and **Loop**

Results

- 90%-10% split on the Mizar Mathematical Library (Grabowski et al., 2010) .
- training set is $\approx 30K$ problems, testing – $\approx 3.2K$

Iteration	1	2	5	8
Training Proved	12325	13749	14403	14498
Testing Proved	1354	1519	1624	1591

Table: Proved problems per iterations of learning

Results

- 90%-10% split on the Mizar Mathematical Library (Grabowski et al., 2010) .
- training set is $\approx 30K$ problems, testing – $\approx 3.2K$

Iteration	1	2	5	8
Training Proved	12325	13749	14403	14498
Testing Proved	1354	1519	1624	1591

Table: Proved problems per iterations of learning

Methodology	Proved	IPS
Heuristics tableaux	1143	64K
RL tableaux	1624	16K

Table: Using RL gives 40% more proves but slows down the inference speed

1 How tableau provers work

2 Reinforcement Learning and Application to TP

- Basics
- Application to TP
- Results

3 Summary

Summary

- ML (RL) application on Theorem Provers
- Helps 'discover' new proofs!
- ... but slows down inference speed

Questions?

- Grabowski, A., Kornilowicz, A., and Naumowicz, A. (2010). Mizar in a nutshell. *Journal of Formalized Reasoning*, 3(2):153–245.
- Jakubuv, J. and Urban, J. (2017). ENIGMA: efficient learning-based inference guiding machine. In *Intelligent Computer Mathematics - 10th International Conference, CICM 2017, Edinburgh, UK, July 17-21, 2017, Proceedings*, pages 292–302.