Reinforcement Learning of Theorem Proving

a paper by
Cezary Kaliszyk, Josef Urban, Henryk Michalewski and Miroslav Olšák

presented by
Dobrik Georgiev

Overview

How tableau provers work

Clauses:

$$
\begin{aligned}
c_1 &: P(X) \\
c_2 &: R(X,Y) \lor \neg P(X) \lor Q(Y) \\
c_3 &: S(X) \lor \neg Q(b) \\
c_4 &: \neg S(X) \lor \neg Q(X) \\
c_5 &: \neg Q(X) \lor \neg R(a,X) \\
c_6 &: \neg R(a,X) \lor Q(X)
\end{aligned}
$$

How tableau provers work

Clauses:

$c_1 \; : \; P(X)$

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

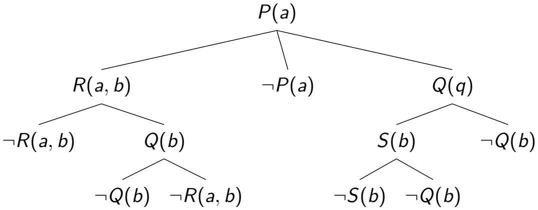
$c_3 \; : \; S(X) \vee \neg Q(b)$

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

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

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

Figure: A closed tableau



How tableau provers work

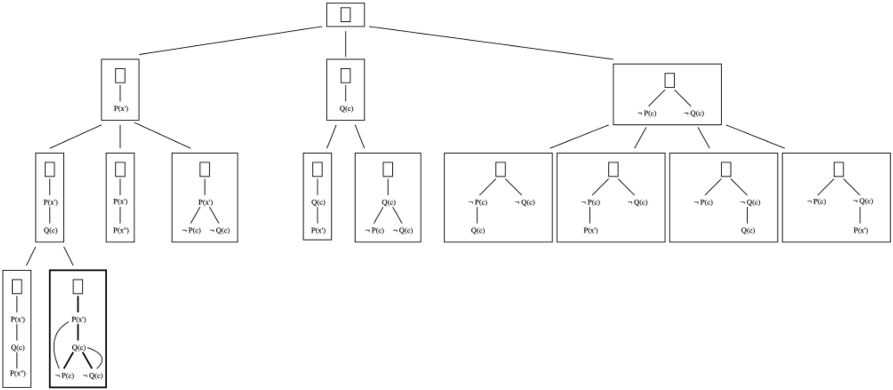


Figure: The search tree of a (non-connection) tableau based TP

RL Basics

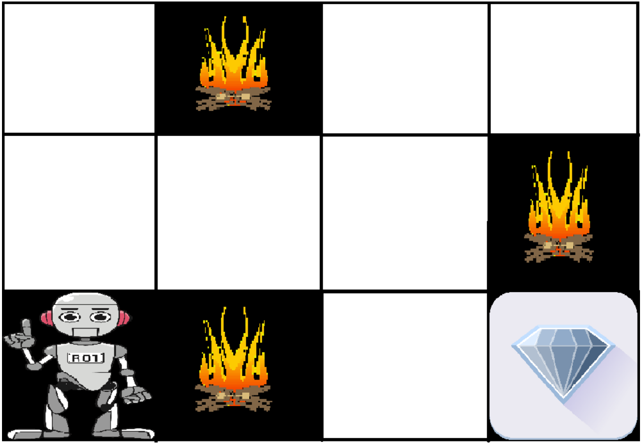


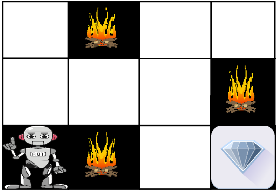
Figure: An agent has to reach a reward without burning

- policy learning
- value learning

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 – 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 – 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 $\{(\oplus, =, f) : 1, (=, f, \circledast) : 2, ...\}$

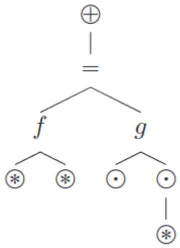


Figure: Feature Tree for L

[3] Example 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 – # 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 – # 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
  - $r_a =$ how often a occurs at i
  - $p_i = softmax(r_a, R)$

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
  - $r_a =$ how often a occurs at i
  - $p_i = softmax(r_a, R)$
- Associate node state feature with value
  - 0 if node not a proof
  - $0.99^{proof\ \ depth}$ otherwise
- Apply regression on the logits to learn prior probability and value

Results

Results from 2003 problems of the Mizar Mathematical Library (Grabowski et al., 2010) with limit of 2×10^6 inferences.

Iteration	1	5	10	15	20
Proved	1037	1182	1210	1223	1235

Table: Proved problems per iterations of learning

Methodology	**Proved**	**IPS**
Heuristics	876	64K
RL	1235	16K

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

Summary

- Reinforcement Learning can be applied to tableau based provers
- Many new problems solved
- RL (and ML methods in general) slow down provers

Questions?

Bibliography

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.