UNIVERSITY OF WATERLOO

Reinforcement Learning of Theorem Proving

Cezary Kaliszyk, Josef Urban, Henryk Michalewski, and Mirek Olsak

NIPS 2018

Presented by Zheng Ma

Motivation

- Automated reasoning over mathematical proof was a major motivation for the development of computer science.
- In principle, Automated theorem provers (ATPs) can in principle be used to attack any formally stated mathematical problem.
- As of today, ATPs are far weaker than trained mathematicians.
- In recent years, machine learning has been used to improve the performance of ATPs

First Order Logic

- Recall that first order logic consists of variables, functions, predicates, logical connectives, and quantifiers.
- For example, the following first order formula states the fact that "if natural number x is prime, then $\sqrt{x}$ is irrational".

$$\forall x\ Prime(x)\ \wedge\ Natural(x) \rightarrow Irrational\big(sqrt(x)\big)$$

- With proper mathematical axioms/theorems as premises, we can prove this formula.
- Note that the provability of first order formulas are undecidable.

Connection Calculi

- leanCoP is an automated theorem prover based on connection calculi.
- Suppose we are trying to prove that formula $P\land R$ is a logical consequence of $\left(\exists x\ Q(x) \lor \neg Q(c)\right) \rightarrow P$ and $\mathrm{P} \rightarrow (\exists y\ Q(y) \lor R)$. Formally stated as
- $\left(\exists x\ Q(x) \lor \neg Q(c)\right) \rightarrow P, \mathrm{P} \rightarrow (\exists y\ Q(y) \lor R) \vdash P\land R$
- Connection calculi first transform this problem into a Skolemized DNF formula

$$(P\land R) \lor \left(\neg P\land Q(x')\right) \lor (\underbrace{\neg Q(b)}_{\text{Literal}}\land P) \lor (\neg \underbrace{Q(c)}_{\text{Atom}}\land\neg P) \lor (\underbrace{P\land\neg R}_{\text{Clause}})$$

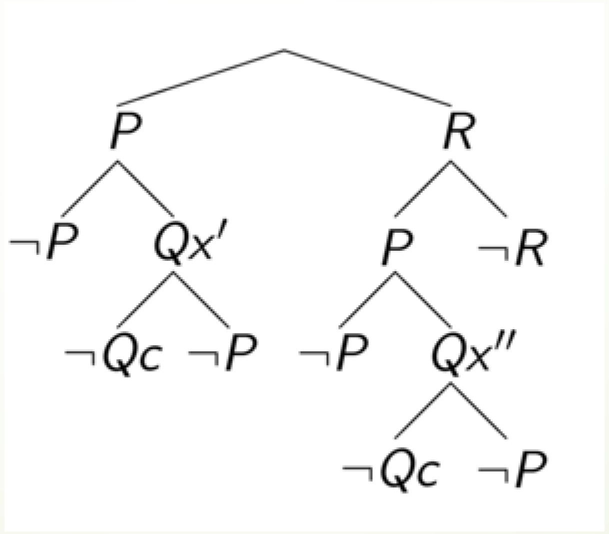
- This formula is called the "matrix"

$$\left\{\{P, R\}, \{\neg P, Q(x')\}, \{\neg Q(b), P\}, \{\neg Q(c), \neg P\}, \{P, \neg R\}\right\}$$

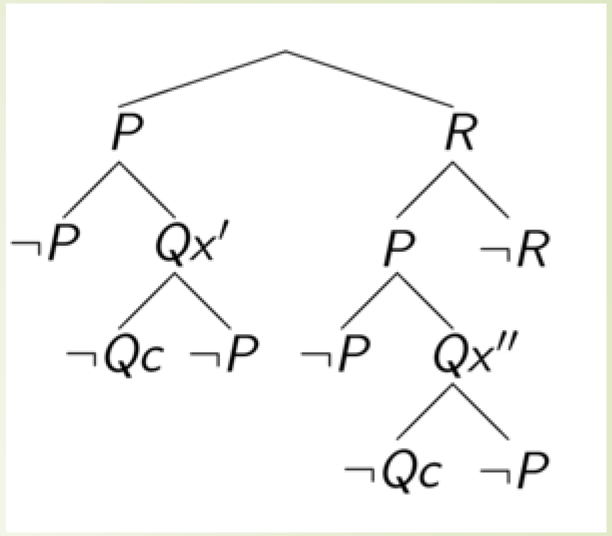
Connection Calculi

- Then we construct a tableau from the matrix.

$$\{\{P, R\}, \{\neg P, Q(x')\}, \{\neg Q(b), P\}, \{\neg Q(c), \neg P\}, \{P, \neg R\}\}$$

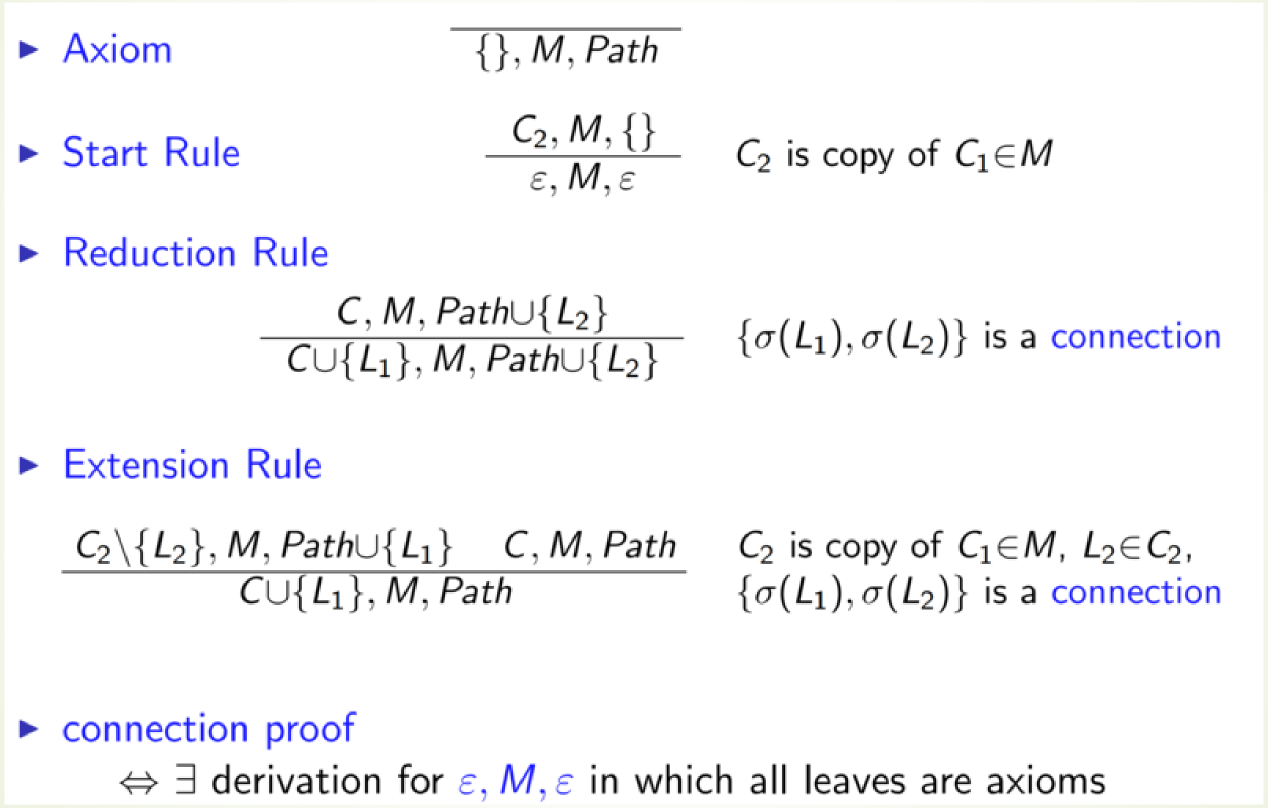


Connection Calculi



- Two literals form a connection if:
  - They have opposite signs
  - Their atoms are unifiable
- If a branch contains a connection, this branch is closed.
- If every branch is closed, the tableau is closed.
- Theorem: $P \vdash C$ iff exists a closed tableau from the corresponding matrix.
  - P is a set of logical formulas, the premises
  - C is a logical formula, the conjecture

Formal Connection Calculi



- Axiom $\frac{}{\{\}, M, Path}$
- Start Rule $\frac{C_2, M, \{\}}{\varepsilon, M, \varepsilon}$ C_2 is copy of $C_1 \in M$
- Reduction Rule

$$\frac{C, M, Path \cup \{L_2\}}{C \cup \{L_1\}, M, Path \cup \{L_2\}} \qquad \{\sigma(L_1), \sigma(L_2)\} \text{ is a connection}$$

- Extension Rule

$$\frac{C_2 \backslash \{L_2\}, M, Path \cup \{L_1\} \quad C, M, Path}{C \cup \{L_1\}, M, Path} \qquad \begin{array}{l} C_2 \text{ is copy of } C_1 \in M,\ L_2 \in C_2, \\ \{\sigma(L_1), \sigma(L_2)\} \text{ is a connection} \end{array}$$

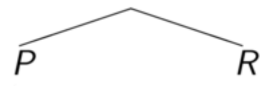
- connection proof
  $\Leftrightarrow \exists$ derivation for $\varepsilon, M, \varepsilon$ in which all leaves are axioms

Formal Connection Calculi

$$M = \{\, \{P, R\}, \{\neg P, Qx\}, \{\neg Qb, P\}, \{\neg Qc, \neg P\}, \{P, \neg R\} \,\}$$

$$\dfrac{\{P, R\}, M, \{\}}{\varepsilon, \{\{P, R\}, \{\neg P, Qx\}, \{\neg Qb, P\}, \{\neg Qc, \neg P\}, \{P, \neg R\}\}, \varepsilon}\ \textit{start rule}$$

(e: extension rule; r: reduction rule; a: axiom)

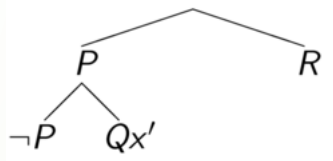


Extension rule

$$M = \{\, \{P, R\}, \{\neg P, Qx\}, \{\neg Qb, P\}, \{\neg Qc, \neg P\}, \{P, \neg R\} \,\}$$

$$\dfrac{\dfrac{\{Qx'\}, M, \{P\} \qquad \{R\}, M, \{\}}{\{P, R\}, M, \{\}}\ e}{\varepsilon, \{\{P, R\}, \{\neg P, Qx\}, \{\neg Qb, P\}, \{\neg Qc, \neg P\}, \{P, \neg R\}\}, \varepsilon}\ \textit{start rule}$$

(e: extension rule; r: reduction rule; a: axiom)



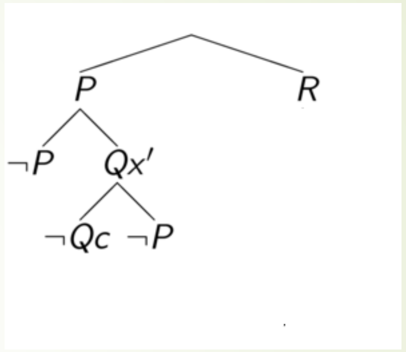
Formal Connection Calculi

Extension rule

$$M = \{\, \{P, R\}, \{\neg P, Qx\}, \{\neg Qb, P\}, \{\neg Qc, \neg P\}, \{P, \neg R\} \,\}$$

$$\dfrac{\dfrac{\dfrac{\{\neg P\}, M, \{P, \neg Qx'\} \quad \{\}, M, \{P\}}{\{Qx'\}, M, \{P\}}\,e \qquad \{R\}, M, \{\}}{\{P, R\}, M, \{\}}\,e}{\varepsilon, \{\{P, R\}, \{\neg P, Qx\}, \{\neg Qb, P\}, \{\neg Qc, \neg P\}, \{P, \neg R\}\}, \varepsilon}\,\textit{start rule}$$

(e: extension rule; r: reduction rule; a: axiom)



Reduction rule

$$M = \{\, \{P, R\}, \{\neg P, Qx\}, \{\neg Qb, P\}, \{\neg Qc, \neg P\}, \{P, \neg R\} \,\}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\{\}, M, \{P, \neg Qx'\}}\,a}{\{\neg P\}, M, \{P, \neg Qx'\}}\,r \quad \dfrac{}{\{\}, M, \{P\}}\,a}{\{Qx'\}, M, \{P\}}\,e \qquad \{R\}, M, \{\}}{\{P, R\}, M, \{\}}\,e}{\varepsilon, \{\{P, R\}, \{\neg P, Qx\}, \{\neg Qb, P\}, \{\neg Qc, \neg P\}, \{P, \neg R\}\}, \varepsilon}\,\textit{start rule}$$

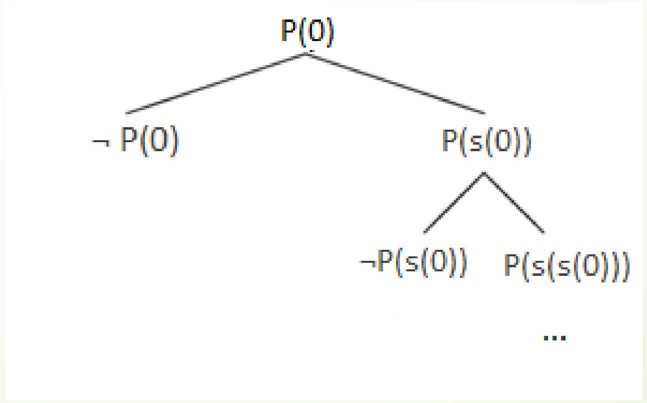
(e: extension rule; r: reduction rule; a: axiom)

Bare Prover: Underlying Prover of This Paper

- The function *prove(subgoal, M, path)* is stated as follows:
  - If *subgoal* is empty
    - return *TRUE*
  - If reduction is possible
    - Perform reduction, generating *new_subgoal*, *new_path*
    - success = *prove(new_subgoal, M, new_path)*
    - If success == TRUE return TRUE
  - For all clauses in *M*
    - If a clause can do extension with *subgoal*
    - Perform extension, generating *new_subgoal1*, *new_path*, *new_subgoal2*
    - success = *prove(new_subgoal1, M, new_path)* and *prove(new_subgoal2, M, path)*
    - If success == TRUE return TRUE
  - return *FALSE*

Why do we need machine learning?

- Pathological example 1: $\{\{P\},\{\neg P\}, \{\neg P, Q\}\}$. Suppose we start with $\{P\}$, then choose $\{\neg P, Q\}$ to do extension, we will fail.
- Pathological example 2: $\{\{P(0)\},\{\neg P(x), P(s(x))\},\{\neg P(0)\}\}$. Suppose we start with $\{P(0)\}$, and choose $\{\neg P(x), P(s(x))\}$ for extension, we will end up with tableau:

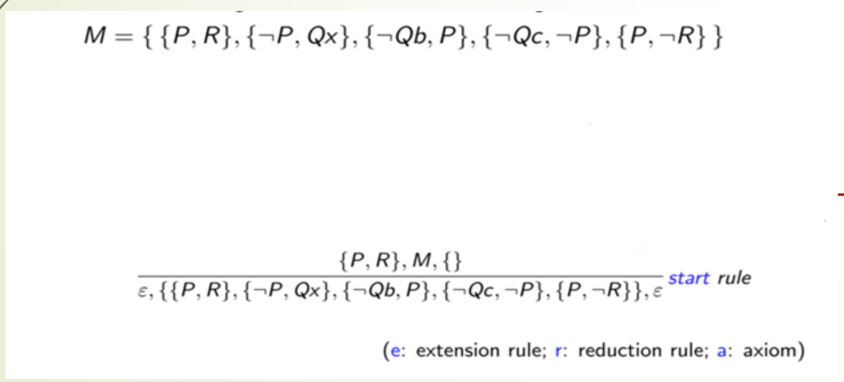


Completeness

- From example 2, we can see that the bare prover is not complete.
- To ensure completeness: enforce iterative deepening.
- Bare prover + iterative deepening = leanCoP / mlCoP
- It is the task of reinforcement learning to guide the prover in such scenarios towards a successful proof.

State and Action

- Proof state: the whole tableau
- Action: an inference step (extension or reduction), along with the chosen clause, the chosen literal

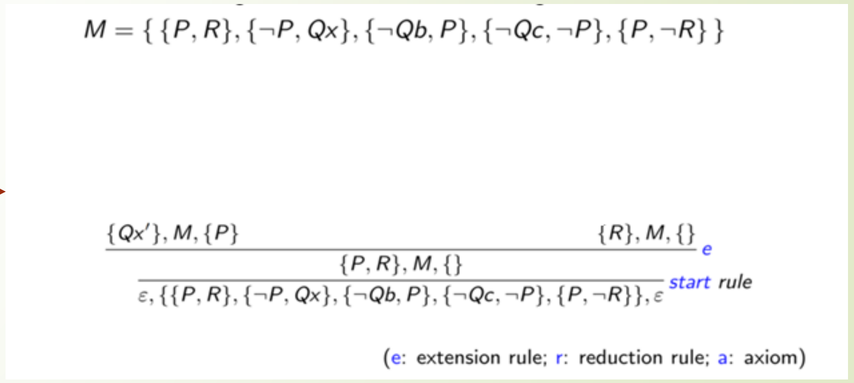


$$M = \{\, \{P, R\}, \{\neg P, Qx\}, \{\neg Qb, P\}, \{\neg Qc, \neg P\}, \{P, \neg R\} \,\}$$

$$\dfrac{\{P, R\}, M, \{\}}{\varepsilon, \{\{P, R\}, \{\neg P, Qx\}, \{\neg Qb, P\}, \{\neg Qc, \neg P\}, \{P, \neg R\}\}, \varepsilon}\ \textit{start rule}$$

(e: extension rule; r: reduction rule; a: axiom)

$$\xrightarrow{\text{Extension}\ \{\neg P, Q(x)\}\ P}$$



$$M = \{\, \{P, R\}, \{\neg P, Qx\}, \{\neg Qb, P\}, \{\neg Qc, \neg P\}, \{P, \neg R\} \,\}$$

$$\dfrac{\dfrac{\{Qx'\}, M, \{P\} \qquad \{R\}, M, \{\}}{\{P, R\}, M, \{\}}\ e}{\varepsilon, \{\{P, R\}, \{\neg P, Qx\}, \{\neg Qb, P\}, \{\neg Qc, \neg P\}, \{P, \neg R\}\}, \varepsilon}\ \textit{start rule}$$

(e: extension rule; r: reduction rule; a: axiom)

Monte Carlo Guidance

- Each proof state i needs to maintain three parameters, its prior probability p_i, its total reward w_i, and number of its visits n_i. If no policy learning is used, the prior probabilities are all equal to one.
- Heuristic to estimate the future reward of leaf states: suppose leaf state i has g_i open subgoals, the reward is computed as 0.95^{g_i}. This will be replaced once value learning is implemented.
- The standard UCT formula is chosen to select the next actions in the playouts.

$$\frac{w_i}{n_i} + 2 \cdot p_i \cdot \sqrt{\frac{\log N}{n_i}}$$

Monte Carlo Guidance

- Play b playouts of length d from the empty tableau, each playout backpropagates the values of proof states it visits
- After these b playouts a special action (inference) is made, corresponding to an actual move, resulting in a new bigstep tableau.
- The next b playouts will start from this tableau, followed by another bigstep, etc.

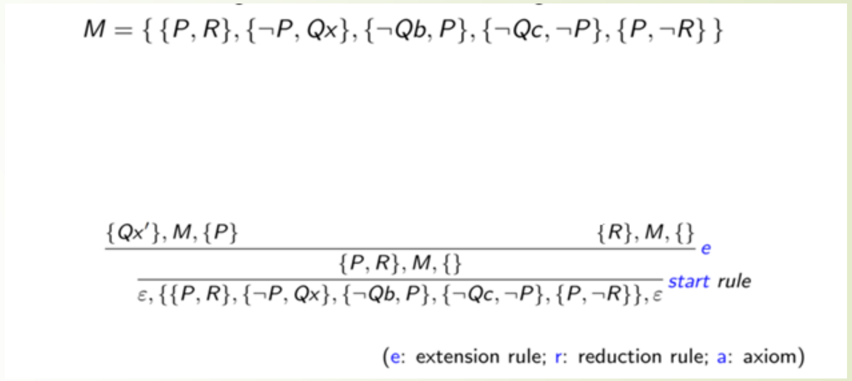
Policy Learning and Guidance

- From many runs of MCT, we will know the prior probability of actions in particular proof states, we can extract the frequency of each action a, and normalize it by dividing with the average action frequency at that state, resulting in a relative proportion $r_a \in (0, \infty)$.
- Characterize proof state into feature vector f_s, action into feature vector f_a.
- (f_s, f_a) is regressed against the associated r_a.
- Model output is used for p_i in the next MCT run.

Value Learning and Guidance

- For a proof state s, if it corresponds to a successful proof, the value is assigned as $v_s = 1$.
- If it corresponds to a failed proof, the value is assigned as $v_s = 0$.
- For other scenarios, denote the distance between state s and a successful state as d_s, then the value is assigned as $v_s = 0.99^{d_s}$
- State feature vector f_s is regressed against v_s.
- During the proof search, the reward of leaf states are computed from this prediction.

Features and Learners



$$M = \{\, \{P, R\}, \{\neg P, Qx\}, \{\neg Qb, P\}, \{\neg Qc, \neg P\}, \{P, \neg R\} \,\}$$

$$\dfrac{\dfrac{\{Qx'\}, M, \{P\} \qquad\qquad \{R\}, M, \{\}}{\{P, R\}, M, \{\}}\, e}{\varepsilon, \{\{P, R\}, \{\neg P, Qx\}, \{\neg Qb, P\}, \{\neg Qc, \neg P\}, \{P, \neg R\}\}, \varepsilon}\, \textit{start rule}$$

(e: extension rule; r: reduction rule; a: axiom)

- XGBoost is chosen as the learner.
- Manually engineered features.
- Feature a proof state
  - Symbols: $P, R, Q, \neg, constant, variable$
  - Open goals: $Q(x'), R \rightarrow (0, 1, 1, 0, 0, 1)$
  - Matrix: $(5, 2, 3, 4, 2, 1)$
  - Paths: $P \rightarrow (1, 0, 0, 0, 0, 0)$
  - For each vector, convert to a single integer (by multiplying elements with primes then add them up), and take modulo by a large prime.
- Additional Features: number of goals, total symbol size of all goals, length of active paths, number of current variable instantiations, most common symbols and their frequencies.

Features and Learners

- Feature of action:
  - Action type
  - Clause
  - Literal

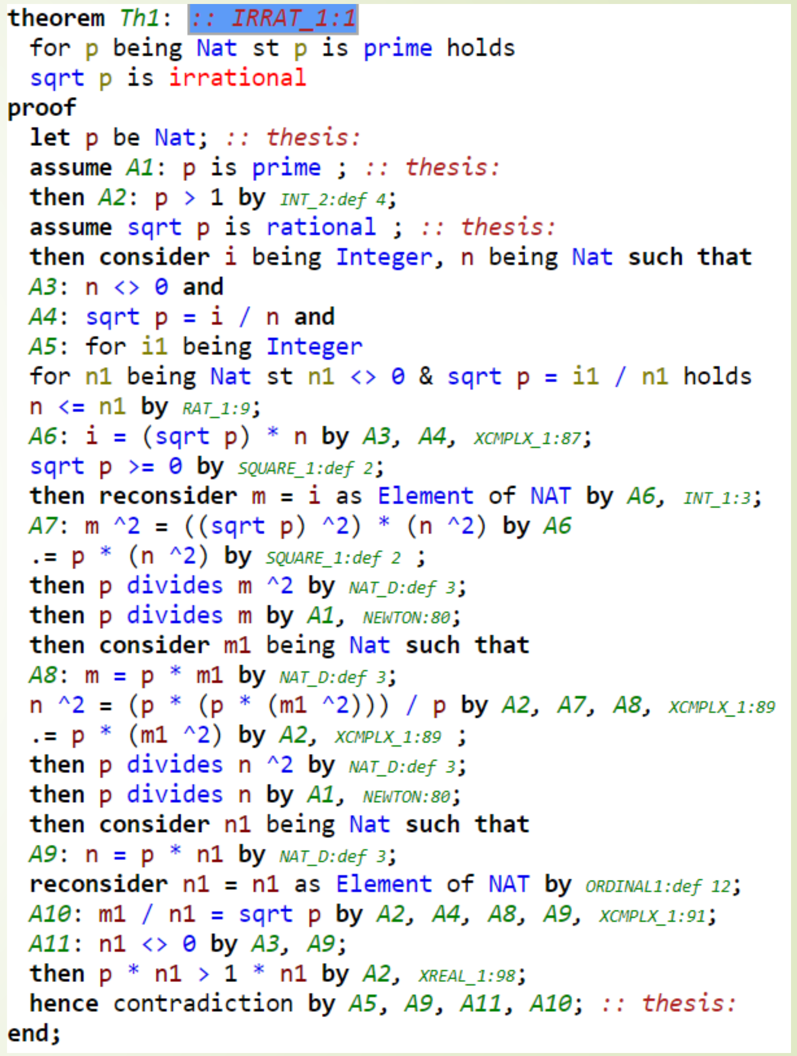
Extension

$\{\neg P, Q(x)\}$

P

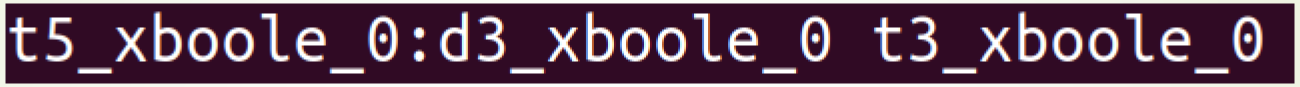
Mizar40 Library

- Mizar Math Library (MML)
  - Based on Tarski-Grothendieck set theory.
  - Written in first-order logic.
- Mizar40: a subset of MML
  - 32,524 theorems with automated proofs
  - Along with many unproven conjectures

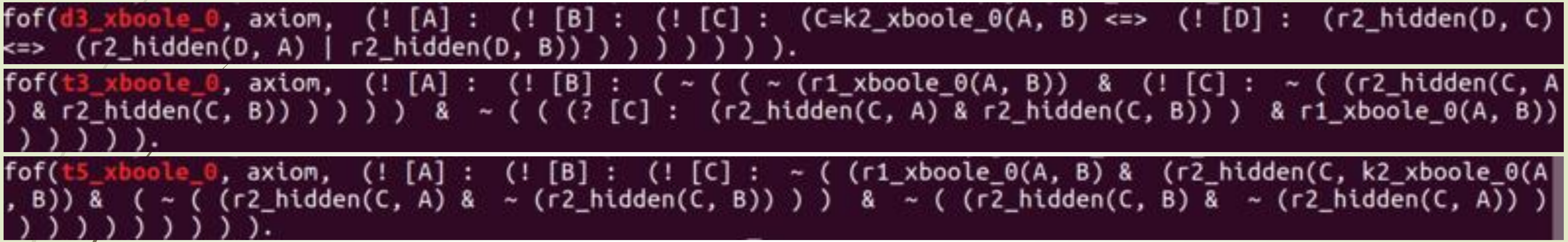


```
theorem Th1: :: IRRAT_1:1
  for p being Nat st p is prime holds
  sqrt p is irrational
proof
  let p be Nat; :: thesis:
  assume A1: p is prime ; :: thesis:
  then A2: p > 1 by INT_2:def 4;
  assume sqrt p is rational ; :: thesis:
  then consider i being Integer, n being Nat such that
  A3: n <> 0 and
  A4: sqrt p = i / n and
  A5: for i1 being Integer
  for n1 being Nat st n1 <> 0 & sqrt p = i1 / n1 holds
  n <= n1 by RAT_1:9;
  A6: i = (sqrt p) * n by A3, A4, XCMPLX_1:87;
  sqrt p >= 0 by SQUARE_1:def 2;
  then reconsider m = i as Element of NAT by A6, INT_1:3;
  A7: m ^2 = ((sqrt p) ^2) * (n ^2) by A6
  .= p * (n ^2) by SQUARE_1:def 2 ;
  then p divides m ^2 by NAT_D:def 3;
  then p divides m by A1, NEWTON:80;
  then consider m1 being Nat such that
  A8: m = p * m1 by NAT_D:def 3;
  n ^2 = (p * (p * (m1 ^2))) / p by A2, A7, A8, XCMPLX_1:89
  .= p * (m1 ^2) by A2, XCMPLX_1:89 ;
  then p divides n ^2 by NAT_D:def 3;
  then p divides n by A1, NEWTON:80;
  then consider n1 being Nat such that
  A9: n = p * n1 by NAT_D:def 3;
  reconsider n1 = n1 as Element of NAT by ORDINAL1:def 12;
  A10: m1 / n1 = sqrt p by A2, A4, A8, A9, XCMPLX_1:91;
  A11: n1 <> 0 by A3, A9;
  then p * n1 > 1 * n1 by A2, XREAL_1:98;
  hence contradiction by A5, A9, A11, A10; :: thesis:
end;
```

Mizar40 Library



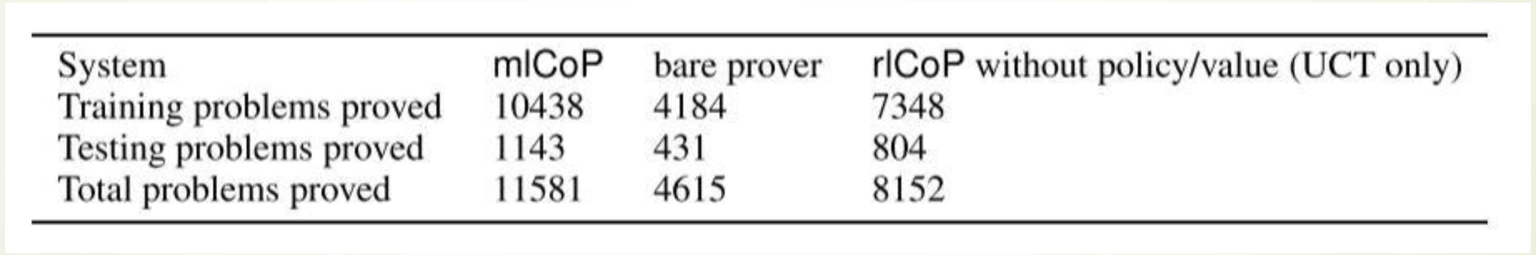
```
t5_xboole_0:d3_xboole_0 t3_xboole_0
```



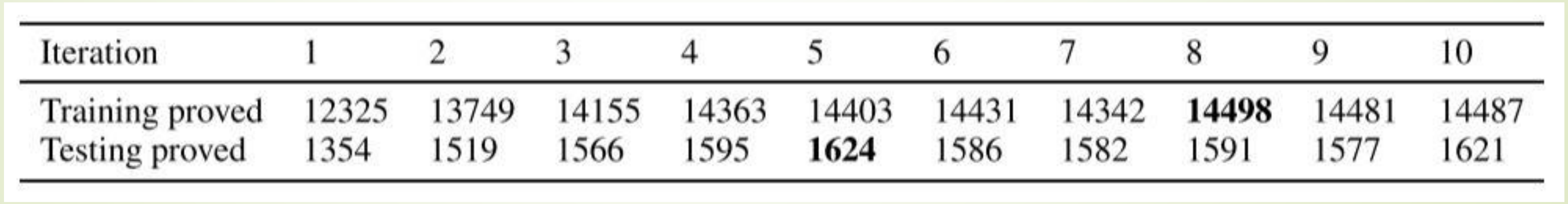
```
fof(d3_xboole_0, axiom,  (! [A] :  (! [B] :  (! [C] :  (C=k2_xboole_0(A, B) <=>  (! [D] :  (r2_hidden(D, C)
<=>  (r2_hidden(D, A) | r2_hidden(D, B)) ) ) ) ) ) ) ).
fof(t3_xboole_0, axiom,  (! [A] :  (! [B] :  ( ~ ( ( ~ (r1_xboole_0(A, B))  &  (! [C] :  ~ ( (r2_hidden(C, A
) & r2_hidden(C, B)) ) ) ) )  &  ~ ( ( (? [C] :  (r2_hidden(C, A) & r2_hidden(C, B)) )  & r1_xboole_0(A, B))
 ) ) ) ) ).
fof(t5_xboole_0, axiom,  (! [A] :  (! [B] :  (! [C] :  ~ ( (r1_xboole_0(A, B) &  (r2_hidden(C, k2_xboole_0(A
, B)) &  ( ~ ( (r2_hidden(C, A) &  ~ (r2_hidden(C, B)) ) )  &  ~ ( (r2_hidden(C, B) &  ~ (r2_hidden(C, A)) )
 ) ) ) ) ) ) ) ) ).
```

Experiment Result

- Split Mizar40 library into 90%/10% training/testing samples.
- 200,000 inferences allowed per proof.
- 10 iterations of policy and value learning are performed.
- 42.1% more theorems proved compared to baseline, mlCoP.



System	mlCoP	bare prover	rlCoP without policy/value (UCT only)
Training problems proved	10438	4184	7348
Testing problems proved	1143	431	804
Total problems proved	11581	4615	8152



Iteration	1	2	3	4	5	6	7	8	9	10
Training proved	12325	13749	14155	14363	14403	14431	14342	**14498**	14481	14487
Testing proved	1354	1519	1566	1595	**1624**	1586	1582	1591	1577	1621

Conclusions

- An automated theorem prover with little domain engineering.
- The author believed that this is a landmark in automated reasoning.
- This paper demonstrated that building general problem solvers by reinforcement learning is a viable approach.

Critiques

- The feature extraction part of this paper is confusing, deep neural networks should do better.
- The policy-value learning iteration is inefficient.
- The experimental design has some flaws. Instead of comparing mlCoP and rlCoP by limiting them to the same number of inference steps, they should limit them to the same amount of time.

Related Work

- A. Alemi, et al. DeepMath-Deep Sequence Models for Premise Selection. NIPS 2016.
- S. Loos, et al. Deep network guided proof search. LPAR-21, 2017.

Thank you!

- Any questions?