universität innsbruck

Computational Logic

Reinforcement Learning of Theorem Proving

Cezary Kaliszyk

Upscale, LRI

October 9, 2018

Math as the next AI challenge

AI can win in easy and some hard games

- We believe we can extrapolate this to other games

Math as the next AI challenge

AI can win in easy and some hard games

- We believe we can extrapolate this to other games

Some higher math is too hard for humans

- Computer proven theorems

$$n(n(x)+y)+n(n(x)+n(y))=x \iff n(n(x+y)+n(x+n(y)))=x$$

- Computer assisted proofs

Kepler, 4 colors, Feit-Thompson Sorting nets

Math as the next AI challenge

AI can win in easy and some hard games

- We believe we can extrapolate this to other games

Some higher math is too hard for humans

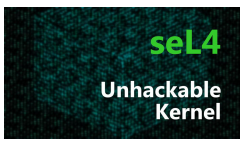
- Computer proven theorems

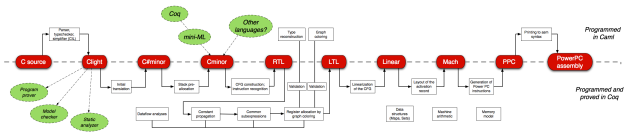
$$n(n(x)+y)+n(n(x)+n(y))=x \iff n(n(x+y)+n(x+n(y)))=x$$

- Computer assisted proofs

Kepler, 4 colors, Feit-Thompson Sorting nets

Proving properties of computer programs





CAKEML
A Verified Implementation of ML

Machine learning in proof techniques already

Relevant Knowledge Selection

- Automated Reasoning
- Important component for human-computer interaction

Conjecturing / Theory Exploration

- Statistical and Generative approaches

...

More human-like proof (divide and conquer) (?)

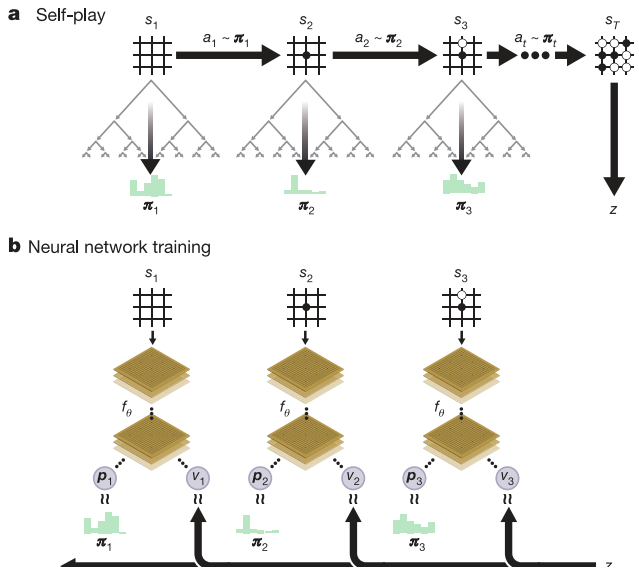
Auto-formalization (?)

Learn reasoning step selection

What does AlphaZero do?

[*Silver et al.*]

- Self-play (learned strategy)
- Repeated strategy improvement
- Search game tree wisely
- Estimate moves and states

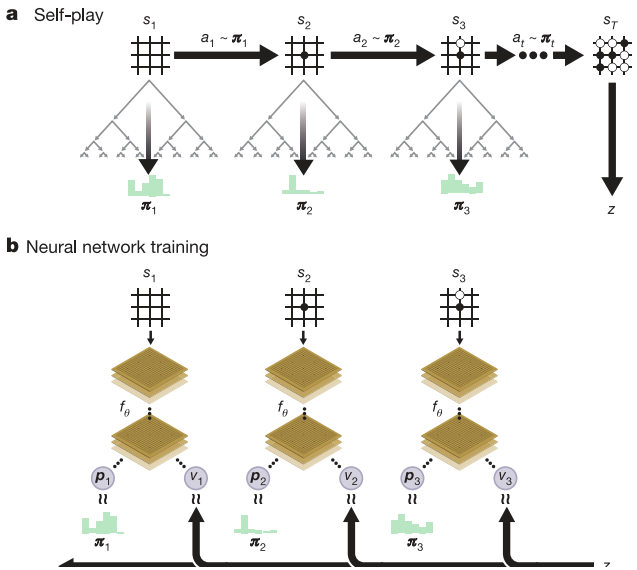


Learn reasoning step selection

What does AlphaZero do?

[*Silver et al.*]

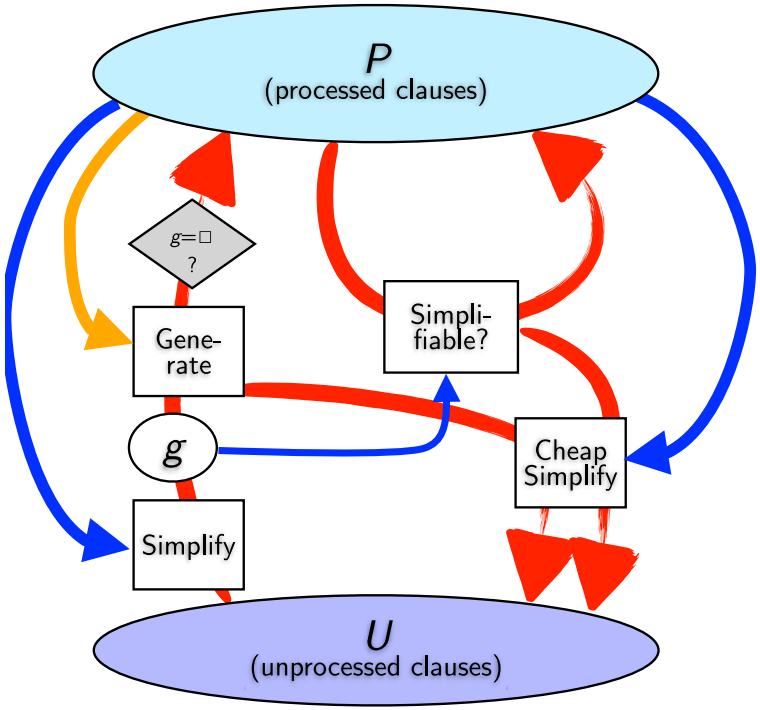
- Self-play (learned strategy)
- Repeated strategy improvement
- Search game tree wisely
- Estimate moves and states



Deep learning for theorem proving?

Deep Learning Queue in E Prover

[*LPAR'17*]

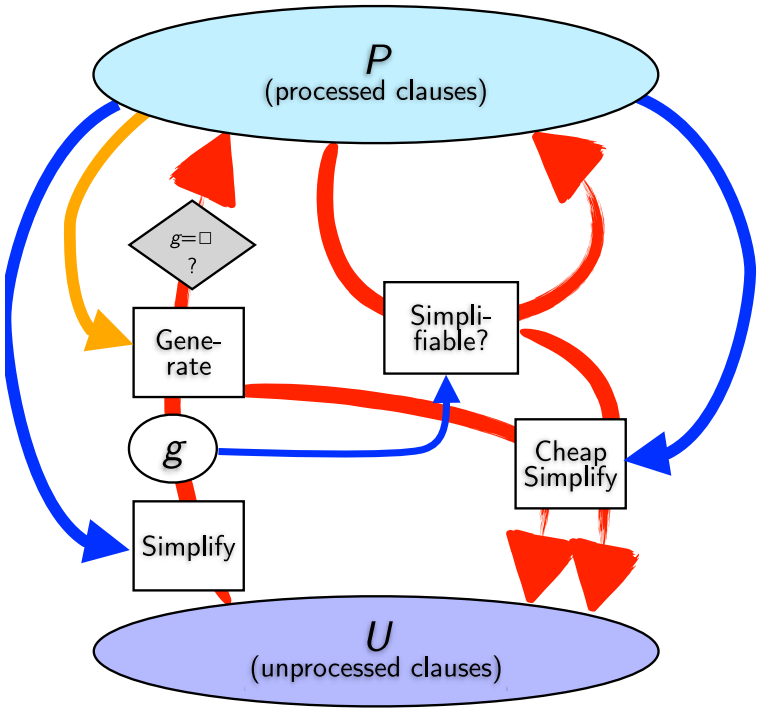


[Schulz'15]

Deep Learning Queue in E Prover

[*LPAR'17*]

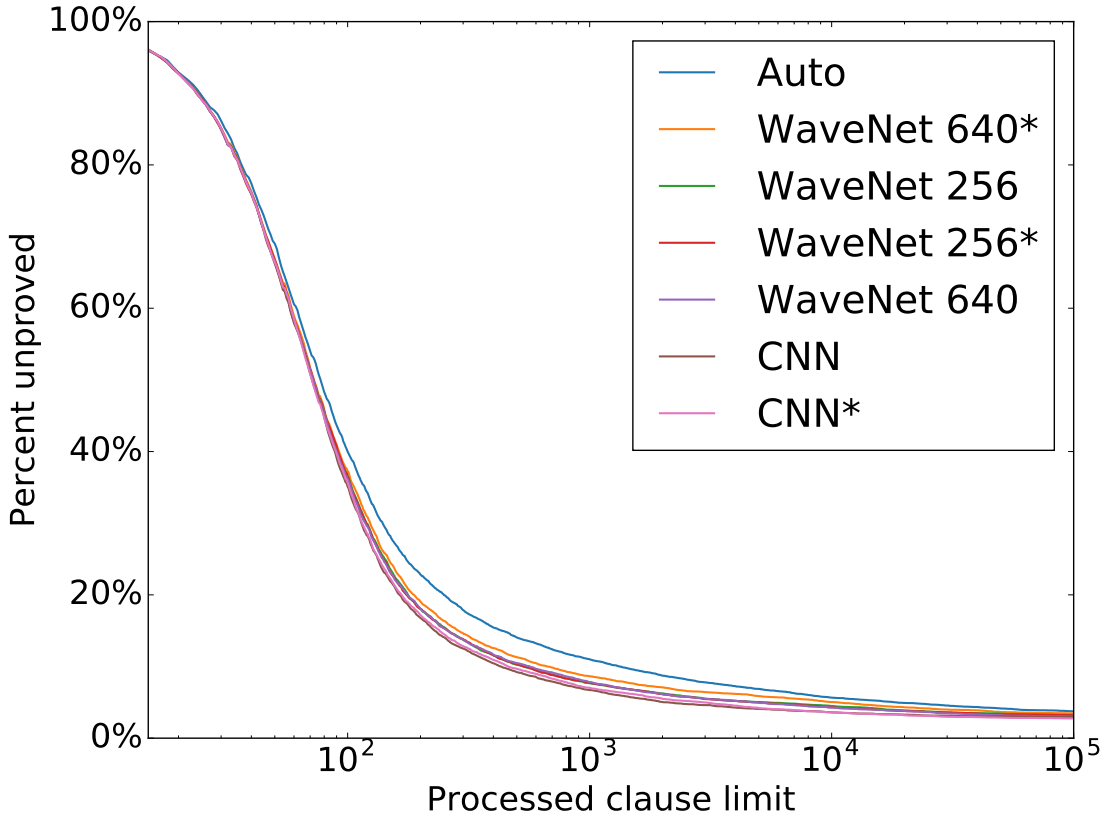
- Choice: unprocessed clause selection
- Batching clauses to evaluate together
- Switch to `auto` after 1000 steps
- Hybrid with `auto`

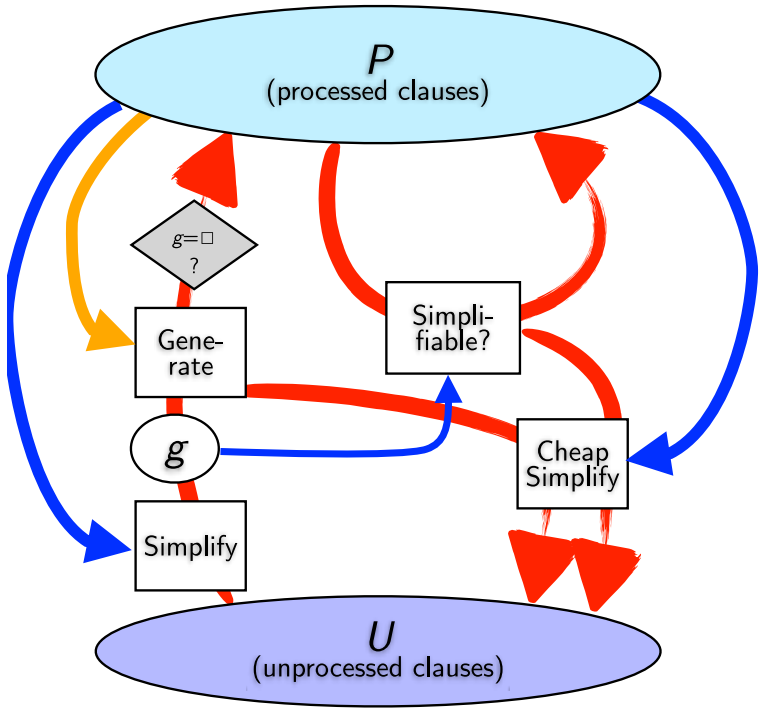


[Schulz'15]

Deep Learning Queue in E Prover

[*LPAR'17*]





[Schulz'15]

leanCoP: Lean Connection Prover

[*Otten 2010*]

Connected tableaux calculus

- Goal oriented, good for large theories

Regularly beats Metis and Prover9 in CASC (CADE ATP competition)

- despite their much larger implementation

Compact Prolog implementation, easy to modify

- Variants for other foundations: iLeanCoP, mLeanCoP
- First experiments with machine learning: MaLeCoP

Easy to imitate

- leanCoP tactic in HOL Light

Lean connection Tableaux

Very simple rules:

- Extension unifies the current literal with a copy of a clause
- Reduction unifies the current literal with a literal on the path

$$\frac{}{\{\}, M, Path} \quad Axiom$$

$$\frac{C, M, Path \cup \{L_2\}}{C \cup \{L_1\}, M, Path \cup \{L_2\}} \quad Reduction$$

$$\frac{C_2 \backslash \{L_2\}, M, Path \cup \{L_1\} \qquad C, M, Path}{C \cup \{L_1\}, M, Path} \quad Extension$$

Example lean connection proof

Clauses:

$c_1 : P(x)$

$c_2 : R(x,y) \vee \neg P(x) \vee Q(y)$

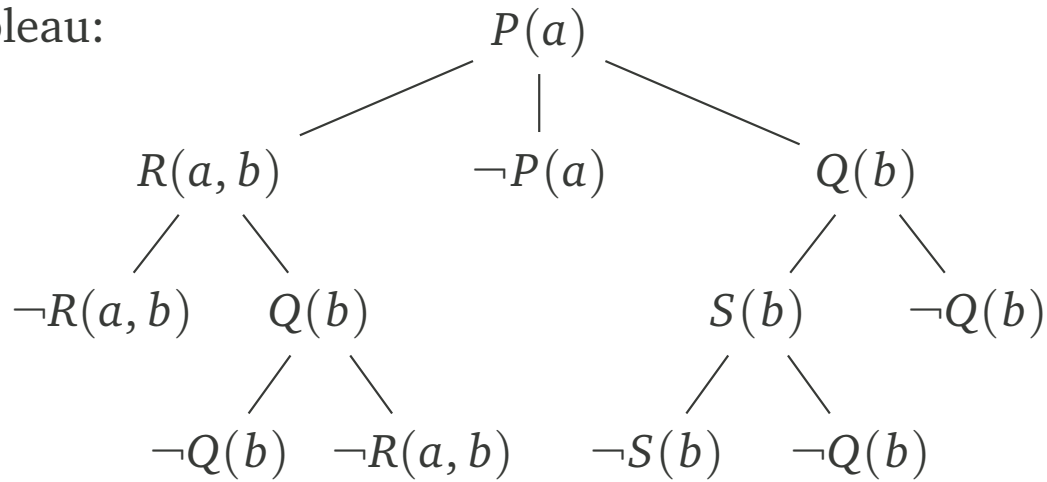
$c_3 : S(x) \vee \neg Q(b)$

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

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

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

Tableau:



First experiment: MaLeCoP in Prolog

[*Tableaux 2011*]

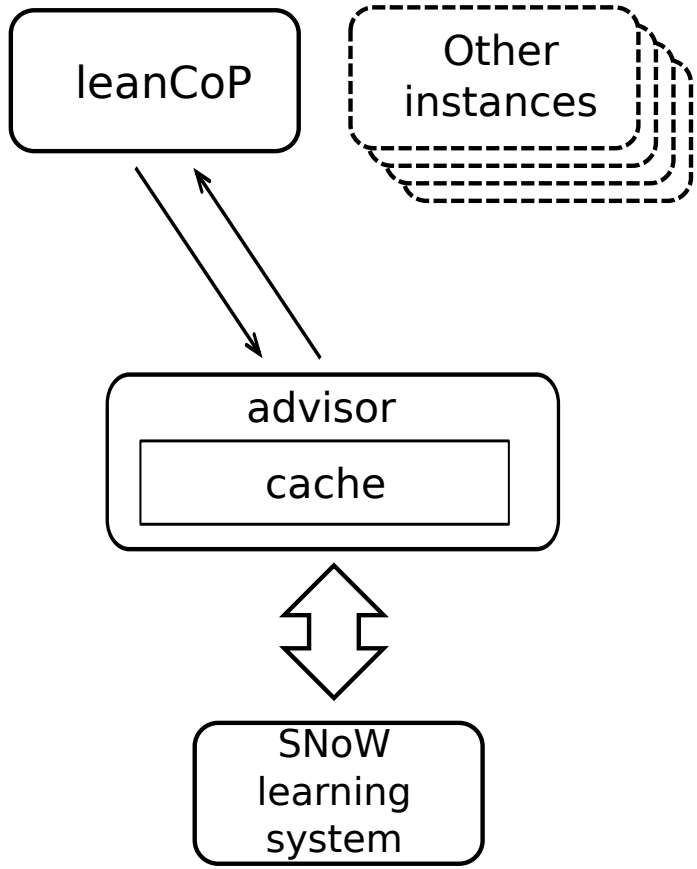
Select extension steps

- Using external advice

Slow implementation

- 1000 times less inf per second

Can avoid 90% inferences!



What about efficiency: FEMaLeCoP

[*LPAR 2015*]

Very simple but very fast classifier built-in

- Naive Bayes (with optimizations)

Approximate features and multi-level indexing

- Offline indexing
- Indexing for the current problem
- Discrimination tree stores NB data

Consistent clausification and skolemization

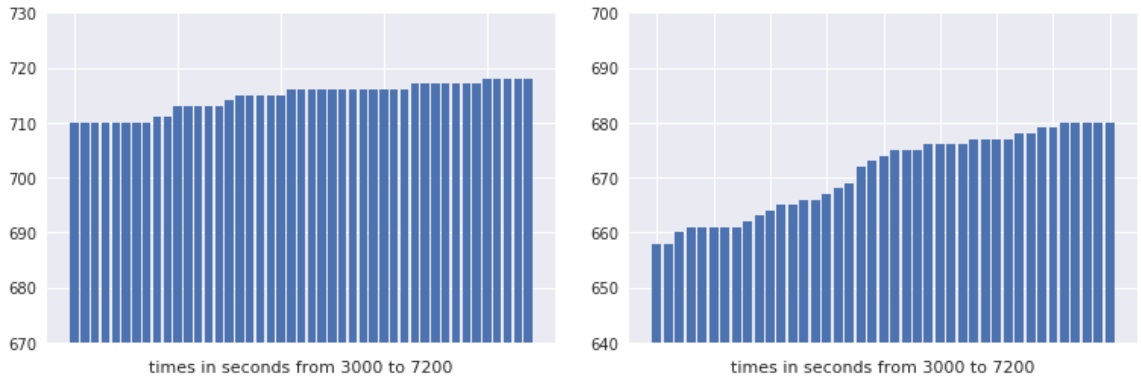
Performance is about 40% of non-learning leanCoP speed

- A few more theorems proved (3–15%)

What about stronger learning?

Yes, but...

- If put directly, huge times needed
- Still improvement small



NBayes vs XGBoost on 2h timeout

Preliminary experiments with deep learning

- So far quite slow

Could we give our tableaux to an AI / game engine?

Could we give our tableaux to an AI / game engine?

Is theorem proving just a maze search?

Yes and NO!

- The proof search tree is not the same as the tableau tree!
- Unification can cause other branches to disappear.

Provide an external interface to proof search

- Usable in OCaml, C++, and Python
- Two functions suffice

$$\text{start} : \text{problem} \to \text{state}$$

$$\text{go} : \text{action} \to \text{state}$$

- where

$$\text{state} = \langle \text{avail action list} \times \text{remaining goal-paths} \rangle$$

Is it ok to change the tree?

Most learning for games sticks to game dynamics

- Only tell it how to do the moves

Is it ok to change the tree?

Most learning for games sticks to game dynamics

- Only tell it how to do the moves

Why is it ok to skip other branches

- Theoretically ATP calculi are complete
- Practically most ATP strategies incomplete

In usual 30s – 300s runs

- Depth of proofs with backtracking: 5–7 (complete)
- Depth with restricted backtracking: 7–10 (more proofs found!)

But with random playouts: depth hundreds of thousands!

- Just unlikely to find a proof → learning

Monte Carlo First Try: MONTECOP [*CADE'17*]

Use Monte Carlo playouts to guide restricted backtracking

- Improves on leanCoP, but not by a big margin
- Potential still limited by depth

What could we do more?

- Learn both policy and value
- Unfold only useful branches
- Do not backtrack
- Arbitrarily long playouts
- Learn both proofs and lack thereof

How to search a tree?

[*Szepesvari 2006*]

How to search a tree?

[*Szepesvari 2006*]

Monte Carlo Tree Search

Upper Confidence Bounds for Trees

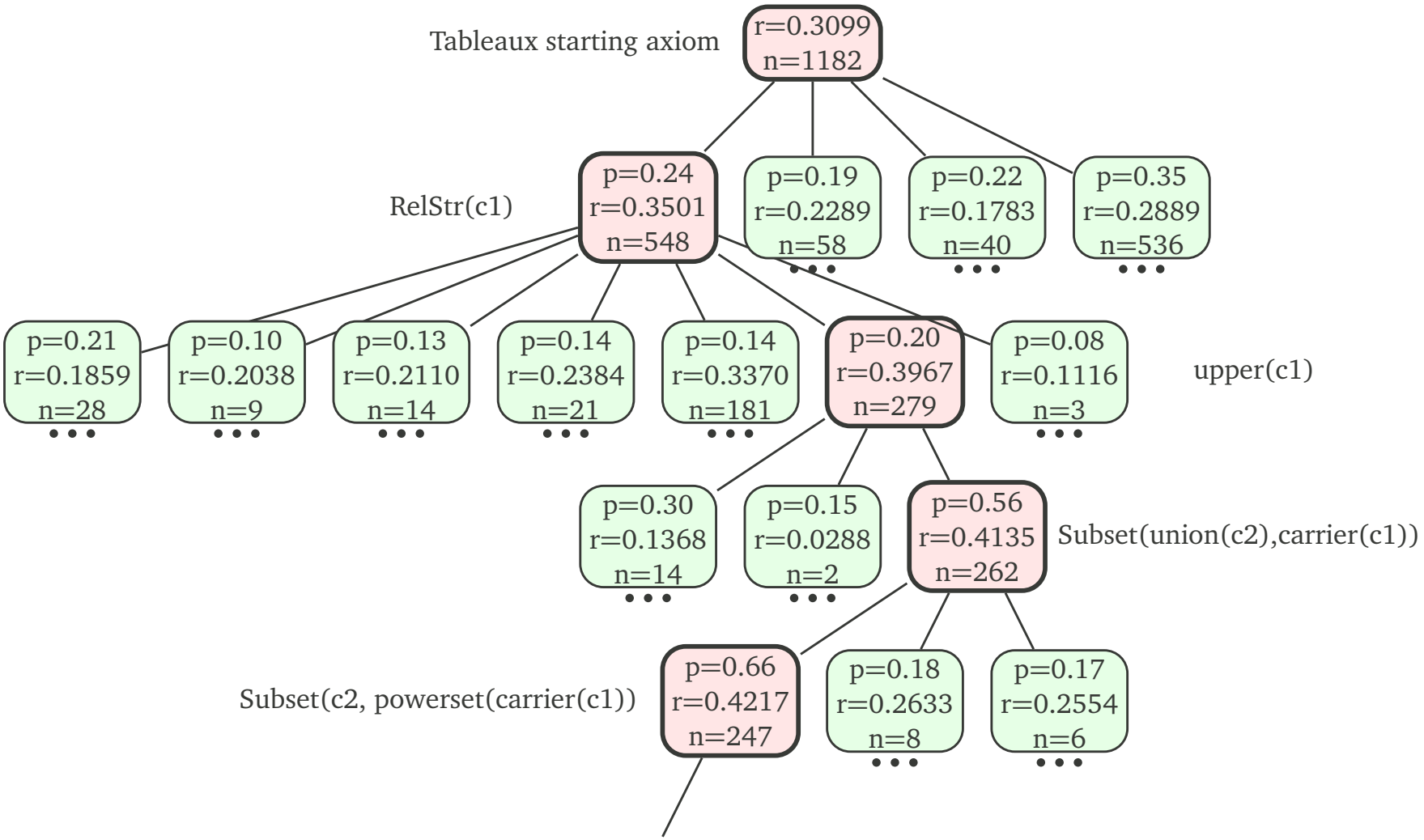
UCT: Select node n maximizing

$$\frac{w_i}{n_i} + c \cdot p_i \cdot \sqrt{\frac{\ln N}{n_i}}$$

Intuition

- Initially proportional to the prior
- Later average reward dominates
- Heuristic replaced by learned priors and rewards

MCTS tree for WAYBEL_0:28



Learn Policy and Value

Policy: Which actions to take?

- Proportions predicted based on proportions in similar states
- Explore less the actions that were “bad” in the past
- Explore more and earlier the actions that were “good”

Learn Policy and Value

Policy: Which actions to take?

- Proportions predicted based on proportions in similar states
- Explore less the actions that were “bad” in the past
- Explore more and earlier the actions that were “good”

Value: How good (close to a proof) is a state?

- Reward states that have few goals
- Reward easy goals

Learn Policy and Value

Policy: Which actions to take?

- Proportions predicted based on proportions in similar states
- Explore less the actions that were “bad” in the past
- Explore more and earlier the actions that were “good”

Value: How good (close to a proof) is a state?

- Reward states that have few goals
- Reward easy goals

Where to get training data?

- Explore 1000 nodes using UCT
- Select the most visited action and focus on it for this proof
- A sequence of selected actions can train both policy and value

Initial comparison: 2000 selected easier Mizar Problems

Baseline, in 200K inferences

leanCoP	random playouts	plain UCT
876	434	770

Initial comparison: 2000 selected easier Mizar Problems

Baseline, in 200K inferences

leanCoP	random playouts	plain UCT
876	434	770

Guided by reinforcement learning from previous iterations (UCT)

Iteration	1	2	3	4	5	6	7	8	9	10
Proved	1037	1110	1166	1179	1182	1198	1196	1193	1212	1210
Iteration	11	12	13	14	15	16	17	18	19	20
Proved	1206	1217	1204	1219	1223	1225	1224	1217	1226	**1235**

Proper Evaluation: Train (29272) and test (3252) sets

Baseline

System	leanCoP	playouts	UCT
Train	10438	4184	7348
Test	**1143**	431	804

Proper Evaluation: Train (29272) and test (3252) sets

Baseline

System	leanCoP	playouts	UCT
Train	10438	4184	7348
Test	**1143**	431	804

10 iterations

Iteration	1	2	3	4	5	6	7	8
Train	12325	13749	14155	14363	14403	14431	14342	**14498**
Test	1354	1519	1566	1595	**1624**	1586	1582	1591

Proper Evaluation: Train (29272) and test (3252) sets

Baseline

System	leanCoP	playouts	UCT
Train	10438	4184	7348
Test	**1143**	431	804

10 iterations

Iteration	1	2	3	4	5	6	7	8
Train	12325	13749	14155	14363	14403	14431	14342	**14498**
Test	1354	1519	1566	1595	**1624**	1586	1582	1591

More Time

leanCoP, 4M inferences, strategies	1396
rlCoP union	1839

Conclusion

Reinforcement learning on Mizar data

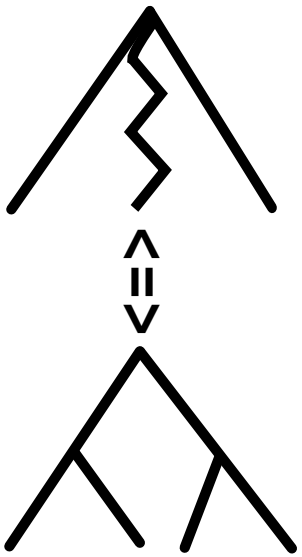
- MCTS, UCT, policy, value work together in connection based setup
- Learning from scratch can work even for a single problem

Lots of things to try

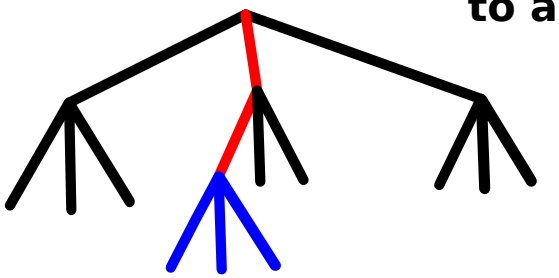
- Other cost functions
- Other learning frameworks
- Larger experiments

Summary

1. Representation: a search in the tree should correspond to a tableaux



2. Playout: follow maximum UCT until unexplored node



3. Explore the node and backup the found reward to all nodes above

4. Repeat 1000 times

5. Focus on most visited node

6. Repeat 100 times

7. Do this for all theorems. We get many sequences of focused steps

8. Train new predictors for policy and value using the seqs.

9. Repeat!