Connection Calculus and Its (Reinforcement) Learning

Josef Urban

Czech Technical University in Prague

April 27, 2020

European Research Council
Established by the European Commission

Automated Theorem Proving

Historical dispute: Gentzen and Hilbert

- Today two communities: Resolution (-style) and Tableaux
- But also more: instantiation-based, Satisfiability Modulo Theories (SMT), Higher-order, etc.

Possible answer: What is better in practice?

- Say the CASC competition or ITP assistance?
- Since the late 90s: resolution (superposition)

But ATP is still far from human performance

- Tableaux may be better for ML methods
- ML methods may be the decisive factor in ATP in the next years

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 and its Guidance

Clauses:

$c_1 : P(x)$

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

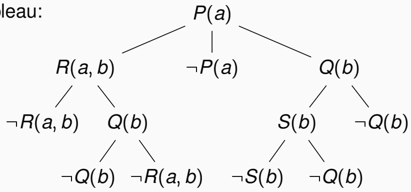
$c_3 : S(x) \lor \neg Q(b)$

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

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

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

Closed Connection Tableau:



- learn guidance of every clausal inference in connection tableau (leanCoP)
- set of first-order clauses, *extension* and *reduction* steps
- proof finished when all branches are closed
- a lot of nondeterminism, requires backtracking
- good for learning – the tableau compactly represents the proof state

leanCoP calculus

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

axiom: $\dfrac{}{\{\}, M, \mathit{Path}}$

reduction rule: $\dfrac{C, M, \mathit{Path} \cup \{L_2\}}{C \cup \{L_1\}, M, \mathit{Path} \cup \{L_2\}}$

where there exists a unification substitution σ such that $\sigma(L_1) = \sigma(\overline{L_2})$

extension rule: $\dfrac{C' \setminus \{L_2\}, M, \mathit{Path} \cup \{L_1\} \qquad C, M, \mathit{Path}}{C \cup \{L_1\}, M, \mathit{Path}}$

where C' is a fresh copy of some $C'' \in M$ such that $L_2 \in C'$ and $\sigma(L_1) = \sigma(\overline{L_2})$ where σ is unification substitution.

Prolog code for the core of leanCoP

```
%   prove(Cla,Path)
prove([Lit|Cla],Path) :-
        (-NegLit=Lit;-Lit=NegLit) ->
        (
          member(NegL,Path),
          unify_with_occurs_check(NegL,NegLit)
        ;
          lit(NegLit,NegL,Cla1,Grnd1),
          unify_with_occurs_check(NegL,NegLit),
          prove(Cla1,[Lit|Path])
        ),
        prove(Cla,Path).
prove([],_).
```

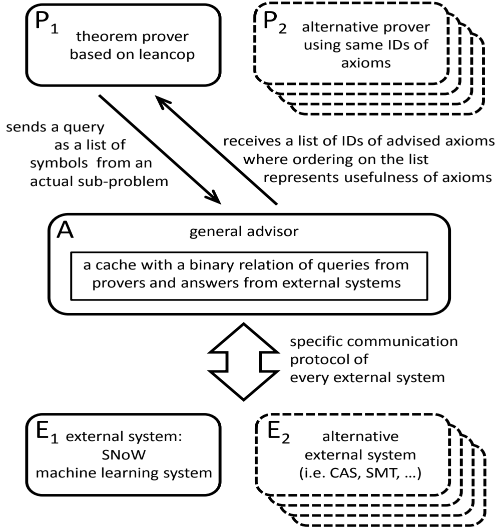
More detailed Prolog code of leanCoP

```
prove([Lit|Cla],Path,PathLim,Lem,Set) :-
  \+ (member(LitC,[Lit|Cla]), member(LitP,Path),LitC==LitP),
  (-NegLit=Lit;-Lit=NegLit) -> (
      member(LitL,Lem), Lit==LitL
      ;
      member(NegL,Path),
      unify_with_occurs_check(NegL,NegLit)
      ;
      lit(NegLit,NegL,Cla1,Grnd1),
      unify_with_occurs_check(NegL,NegLit),
        ( Grnd1=g -> true ;
          length(Path,K), K<PathLim -> true ;
          \+ pathlim -> assert(pathlim), fail ),
      prove(Cla1,[Lit|Path],PathLim,Lem,Set)
    ), ( member(cut,Set) -> ! ; true ),
    prove(Cla,Path,PathLim,[Lit|Lem],Set).
prove([],_,_,_,_,[]).
```

Statistical Guidance of Connection Tableau

- **MaLeCoP** (2011): first prototype Machine Learning Connection Prover
- extension rules chosen by naive Bayes trained on good decisions
- training examples: tableau features plus the name of the chosen clause
- initially slow: off-the-shelf learner 1000 times slower than raw leanCoP
- 20-time search shortening on the MPTP Challenge
- second version: 2015, with C. Kaliszyk
- both prover and naive Bayes in OCAML, fast indexing
- Fairly Efficient MaLeCoP = **FEMaLeCoP**
- 15% improvement over untrained leanCoP on the MPTP2078 problems
- using iterative deepening - enumerate shorter proofs before longer ones

General Advising Design



LeanCoP modifications

- Consistent clausification across many problems needed for consistent learning/advice
- Options like definition introduction need to be fixed
- Providing training data for external advising systems
- Mechanisms for taking advice from external system(s)
- Profiling mechanisms
- External advice is quite slow: number of strategies defined trading advice for speed

Statistical Guidance of Connection Tableau – rlCoP

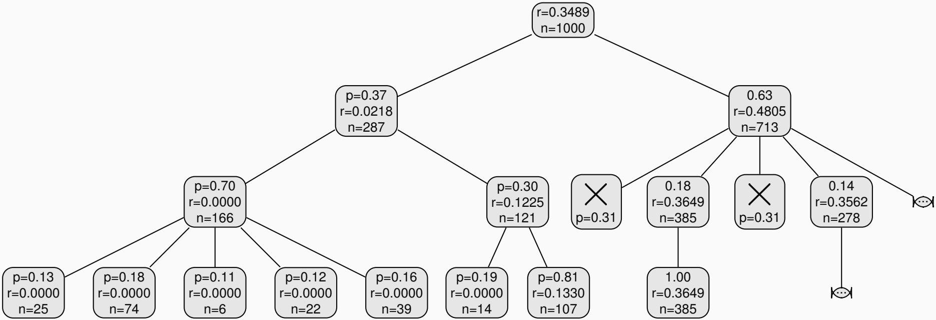
- 2018: stronger learners via C interface to OCAML (boosted trees)
- remove iterative deepening, the prover can go arbitrarily deep
- added Monte-Carlo Tree Search (MCTS) – AlphaGo/Zero
- MCTS search nodes are sequences of clause application
- a good heuristic to explore new vs exploit good nodes:

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

(UCT - Kocsis, Szepesvari 2006)

- learning both *policy* (clause selection) and *value* (state evaluation)
- clauses represented not by names but also by features (generalize!)
- binary learning setting used: | proof state | clause features |
- mostly term walks of length 3 (trigrams), hashed into small integers
- many iterations of proving and learning

Tree Example



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

Reinforcement from scratch – 2003 problems

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**

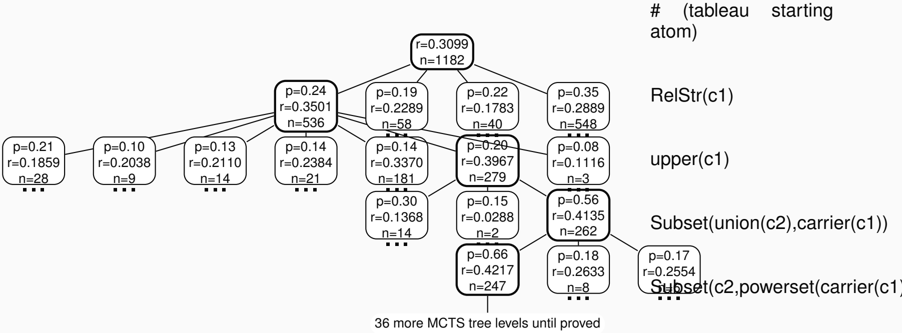
rlCoP on 2003 Mizar problems – Policy and Value only

System	leanCoP	bare prover	rlCoP without policy/value (UCT only)
Problems proved	876	434	770

Iteration	1	2	3	4	5	6	7	8	9	10
Proved	974	1008	1028	1053	1066	1054	1058	1059	1075	1070
Iteration	11	12	13	14	15	16	17	18	19	20
Proved	1074	1079	1077	1080	1075	1075	**1087**	1071	1076	1075

Iteration	1	2	3	4	5	6	7	8	9	10
Proved	809	818	821	821	818	824	**856**	831	842	826
Iteration	11	12	13	14	15	16	17	18	19	20
Proved	832	830	825	832	828	820	825	825	831	815

More trees


r=0.3099 n=1182
p=0.24 r=0.3501 n=536
p=0.19 r=0.2289 n=58
p=0.22 r=0.1783 n=40
p=0.35 r=0.2889 n=548
p=0.21 r=0.1859 n=28
p=0.10 r=0.2038 n=9
p=0.13 r=0.2110 n=14
p=0.14 r=0.2384 n=21
p=0.14 r=0.3370 n=181
p=0.20 r=0.3967 n=279
p=0.08 r=0.1116 n=3
p=0.30 r=0.1368 n=14
p=0.15 r=0.0288 n=2
p=0.56 r=0.4135 n=262
p=0.66 r=0.4217 n=247
p=0.18 r=0.2633 n=8
p=0.17 r=0.2554
36 more MCTS tree levels until proved
# (tableau starting atom)
RelStr(c1)
upper(c1)
Subset(union(c2),carrier(c1))
Subset(c2,powerset(carrier(c1)

rlCoP on 32k Mizar problems

- On 32k Mizar40 problems using 200k inference limit
- nonlearning CoPs:

System	leanCoP	bare prover	rlCoP no policy/value (UCT only)
Training problems proved	10438	4184	7348
Testing problems proved	**1143**	431	804
Total problems proved	11581	4615	8152

- rlCoP with policy/value after 5 proving/learning iters on the training data
- $1624/1143 = 42.1$% improvement over leanCoP on the testing problems

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

Extensions and (re)implementations

- plcop - Prolog setting again
- FloP - longer proof, curriculum learning
- rlcop with GNN - see the paper

Feedback loop for ENIGMA on Mizar data

- Similar to rlCoP - interleave proving and learning of ENIGMA guidance
- Done on 57880 Mizar problems very recently
- Ultimately a 70% improvement over the original strategy

	$\mathcal{S}$	$\mathcal{S} \odot \mathcal{M}_9^0$	$\mathcal{S} \oplus \mathcal{M}_9^0$	$\mathcal{S} \odot \mathcal{M}_9^1$	$\mathcal{S} \oplus \mathcal{M}_9^1$	$\mathcal{S} \odot \mathcal{M}_9^2$	$\mathcal{S} \oplus \mathcal{M}_9^2$	$\mathcal{S} \odot \mathcal{M}$
solved	14933	16574	20366	21564	22839	22413	23467	22910
$\mathcal{S}\%$	+0%	+10.5%	+35.8%	+43.8%	+52.3%	+49.4%	+56.5%	+52.8%
$\mathcal{S}+$	+0	+4364	+6215	+7774	+8414	+8407	+8964	+8822
$\mathcal{S}-$	-0	-2723	-782	-1143	-508	-927	-430	-845

	$\mathcal{S} \odot \mathcal{M}_{12}^3$	$\mathcal{S} \oplus \mathcal{M}_{12}^3$	$\mathcal{S} \odot \mathcal{M}_{16}^3$	$\mathcal{S} \oplus \mathcal{M}_{16}^3$
solved	24159	24701	25100	25397
$\mathcal{S}\%$	+61.1%	+64.8%	+68.0%	+70.0%
$\mathcal{S}+$	+9761	+10063	+10476	+10647
$\mathcal{S}-$	-535	-295	-309	-183

Some References

leanCoP and its variants:

- Jens Otten: leanCoP 2.0 and ileanCoP 1.2: High Performance Lean Theorem Proving in Classical and Intuitionistic Logic (System Descriptions). IJCAR 2008: 283-291
- Jens Otten, Wolfgang Bibel: leanCoP: lean connection-based theorem proving. J. Symb. Comput. 36(1-2): 139-161 (2003)
- Jens Otten: MleanCoP: A Connection Prover for First-Order Modal Logic. IJCAR 2014: 269-276
- Jens Otten: nanoCoP: A Non-clausal Connection Prover. IJCAR 2016: 302-312
- Jens Otten: A Non-clausal Connection Calculus. TABLEAUX 2011: 226-241
- Cezary Kaliszyk, Josef Urban, Jirí Vyskocil: Certified Connection Tableaux Proofs for HOL Light and TPTP. CPP 2015: 59-66

Machine Learning leanCoP-style systems:

- C. Kaliszyk, J. Urban, H. Michalewski, M. Olsak: Reinforcement Learning of Theorem Proving. CoRR abs/1805.07563 (2018)
- Miroslav Olsák, Cezary Kaliszyk, Josef Urban: Property Invariant Embedding for Automated Reasoning. CoRR abs/1911.12073 (2019)
- Zsolt Zombori, Josef Urban, Chad E. Brown: Prolog Technology Reinforcement Learning Prover. CoRR abs/2004.06997 (2020)
- Zsolt Zombori, Adrián Csiszárik, Henryk Michalewski, Cezary Kaliszyk, Josef Urban: Towards Finding Longer Proofs. CoRR abs/1905.13100 (2019)
- Cezary Kaliszyk, Josef Urban: FEMaLeCoP: Fairly Efficient Machine Learning Connection Prover. LPAR 2015: 88-96
- Josef Urban, Jirí Vyskocil, Petr Stepánek: MaLeCoP Machine Learning Connection Prover. TABLEAUX 2011: 263-277