Artificial Intelligence in Theorem Proving

Cezary Kaliszyk

VTSA

Overview

Last Lecture

- theorem proving problems
- premise selection
- deep learning for theorem proving
- state estimation

Today

- automated reasoning
- learning in classical ATPs
- learning for tableaux
- reinforcement learning in TP
- longer proofs

What about ATPs

Proof by contradiction

- Assume that the conjecture does not hold
- Derive that axioms and negated conjecture imply $\bot$

Saturation

- Convert problem to CNF
- Enumerate the consequences of the available clauses
- Goal: get to the empty clause

Redundancies

Simplify or eliminate some clauses (contract)

Calculus

Resolution

$$\frac{C \lor A \quad D \lor \neg B}{(C \lor D)\sigma} \qquad \frac{C \lor A \lor B}{(C \lor A)\sigma}$$

Calculus

Ordered Resolution

$$\frac{C \vee A \quad D \vee \neg B}{(C \vee D)\sigma} \qquad \frac{C \vee A \vee B}{(C \vee A)\sigma}$$

$A\sigma$ strictly maximal wrt $C\sigma$ and B maximal wrt $D\sigma$.

Calculus

Ordered Resolution

$$\frac{C \vee A \quad D \vee \neg B}{(C \vee D)\sigma} \qquad \frac{C \vee A \vee B}{(C \vee A)\sigma}$$

$A\sigma$ strictly maximal wrt $C\sigma$ and B maximal wrt $D\sigma$.

Equality axioms?

Calculus

Ordered Resolution

$$\frac{C \vee A \quad D \vee \neg B}{(C \vee D)\sigma} \qquad \frac{C \vee A \vee B}{(C \vee A)\sigma}$$

$A\sigma$ strictly maximal wrt $C\sigma$ and B maximal wrt $D\sigma$.

Equality axioms?

Ordered Paramodulation

$$\frac{C \vee s \neq s'}{C\sigma'} \qquad \frac{C \vee s = t \quad D \vee L[s']}{(C \vee D \vee L[t])\sigma'}$$

Calculus

Ordered Resolution

$$\frac{C \vee A \quad D \vee \neg B}{(C \vee D)\sigma} \qquad \frac{C \vee A \vee B}{(C \vee A)\sigma}$$

$A\sigma$ strictly maximal wrt $C\sigma$ and B maximal wrt $D\sigma$.

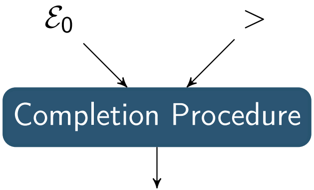
Equality axioms?

Ordered Paramodulation

$$\frac{C \vee s \neq s'}{C\sigma'} \qquad \frac{C \vee s = t \quad D \vee L[s']}{(C \vee D \vee L[t])\sigma'}$$

$(s = t)\sigma$ and $L[s']\sigma'$ maximal in their clauses.

Completion



$$(\mathcal{E}_0, \varnothing) \vdash (\mathcal{E}_1, \mathcal{R}_1) \vdash (\mathcal{E}_2, \mathcal{R}_2) \vdash (\mathcal{E}_3, \mathcal{R}_3) \vdash \cdots$$

deduce $\dfrac{(\mathcal{E}, \mathcal{R})}{(\mathcal{E} \cup \{s \approx t\}, \mathcal{R})}$ if $s \;{}_{\mathcal{R}}\!\!\leftarrow u \rightarrow_{\mathcal{R}} t$

delete $\dfrac{(\mathcal{E} \cup \{s \approx s\}, R)}{(\mathcal{E}, \mathcal{R})}$

orient $\dfrac{(\mathcal{E} \cup \{s \dot{\approx} t\}, \mathcal{R})}{(\mathcal{E}, \mathcal{R} \cup \{s \rightarrow t\})}$ if $s > t$

compose $\dfrac{(\mathcal{E}, \mathcal{R} \cup \{s \rightarrow t\})}{(\mathcal{E}, \mathcal{R} \cup \{s \rightarrow u\})}$ if $t \rightarrow_{\mathcal{R}} u$

simplify $\dfrac{(\mathcal{E} \cup \{s \dot{\approx} t\}, \mathcal{R})}{(\mathcal{E} \cup \{u \dot{\approx} t\}, \mathcal{R})}$ if $s \rightarrow_{\mathcal{R}} u$

collapse $\dfrac{(\mathcal{E}, \mathcal{R} \cup \{s \rightarrow t\})}{(\mathcal{E} \cup \{u \approx t\}, \mathcal{R})}$ if $s \overset{\sqsupset}{\rightarrow}_{\mathcal{R}} u$

Superposition Calculus

$$\frac{C \vee A \quad D \vee \neg B}{(C \vee D)\sigma} \qquad \frac{C \vee A \vee B}{(C \vee A)\sigma}$$

$$\frac{C \vee s = t \quad D \vee \neg A[s']}{(C \vee D \vee \neg A[t])\sigma} \qquad \frac{C \vee s = t \quad D \vee A[s']}{(C \vee D \vee A[t])\sigma}$$

$$\frac{C \vee s = t \quad D \vee u[s'] \neq v}{(C \vee D \vee u[t] \neq v)\sigma} \qquad \frac{C \vee s = t \quad D \vee u[s'] = v}{(C \vee D \vee u[t] = v)\sigma}$$

$$\frac{C \vee s \neq t}{C\sigma} \qquad \frac{C \vee u = v \vee s = t}{(C \vee v \neq t \vee u = t)\sigma}$$

Basis of

- E, Vampire, Spass, Prover9, ≈Metis

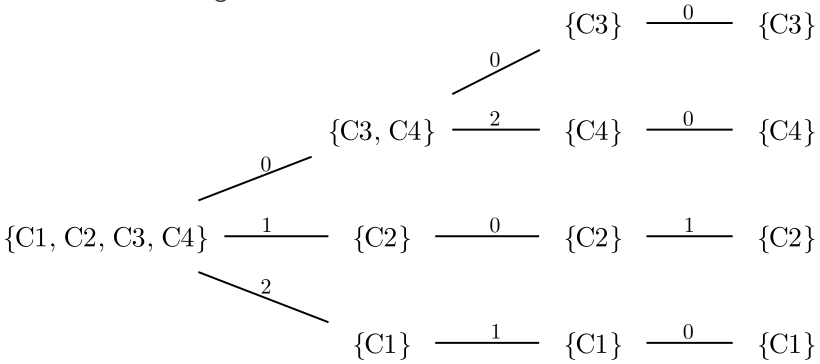
Beyond the Calculus

Tautology Deletion

$$a \vee b \vee \neg a \vee d$$

Subsumption (forward and backward)

e.g. E uses Feature Vector Indexing



Still...

```
fof(6, axiom,![X1]:![X2]:![X4]:gg(X1,sup_sup(X1,X2,X4)),file('i/f/1/goal_138__Q_Restricted_Rewriting.qrstep
fof(32, axiom,![X1]:![X2]:gg(set(product_prod(X1,X1)),transitive_rtrancl(X1,X2)),file('i/f/1/goal_138__Q_Re
fof(55, axiom,![X1]:![X19]:![X20]:(member(product_prod(X1,X1),X19,X20)=>member(product_prod(X1,X1),X19,tran
fof(68, axiom,![X1]:![X5]:![X3]:![X36]:![X20]:![X37]:![X16]:(ord_less_eq(set(product_prod(X1,X3)),X36,X20)=
fof(70, axiom,![X1]:![X20]:transitive_rtrancl(X1,transitive_rtrancl(X1,X20))=transitive_rtrancl(X1,X20),fil
fof(74, axiom,![X1]:![X24]:![X34]:![X33]:((~(member(X1,X24,X34))=>member(X1,X24,X33))=>member(X1,X24,sup_su
fof(78, axiom,![X1]:![X11]:![X13]:transitive_rtrancl(X1,sup_sup(set(product_prod(X1,X1)),transitive_rtrancl
fof(79, axiom,![X1]:![X22]:![X39]:(member(X1,X22,collect(X1,X39))<=>pp(aa(X1,bool,X39,X22))),file('i/f/1/go
fof(85, axiom,![X1]:(semilattice_sup(X1)=>![X23]:![X24]:![X22]:(ord_less_eq(X1,sup_sup(X1,X23,X24),X22)<=>(
fof(86, axiom,![X1]:![X11]:relcomp(X1,X1,X1,transitive_rtrancl(X1,X11),transitive_rtrancl(X1,X11))=transiti
fof(98, axiom,![X1]:![X33]:![X34]:(gg(set(X1),X34)=>(ord_less_eq(set(X1),X33,X34)<=>sup_sup(set(X1),X33,X34
fof(99, axiom,![X1]:![X33]:![X34]:ord_less_eq(set(X1),X33,sup_sup(set(X1),X33,X34)),file('i/f/1/goal_138__Q
fof(100, axiom,![X3]:![X1]:supteq(X1,X3)=sup_sup(set(product_prod(term(X1,X3),term(X1,X3))),supt(X1,X3),id(
fof(102, axiom,![X1]:![X34]:![X33]:ord_less_eq(set(X1),X34,sup_sup(set(X1),X33,X34)),file('i/f/1/goal_138__
fof(103, axiom,![X1]:![X33]:![X18]:![X34]:(ord_less_eq(set(X1),X33,X18)=>(ord_less_eq(set(X1),X34,X18)=>ord
fof(109, axiom,![X1]:![X34]:![X33]:(gg(set(X1),X33)=>(ord_less_eq(set(X1),X34,X33)=>sup_sup(set(X1),X33,X34
fof(114, axiom,![X1]:![X33]:![X18]:![X34]:![X48]:(ord_less_eq(set(X1),X33,X18)=>(ord_less_eq(set(X1),X34,X4
fof(116, axiom,![X1]:![X33]:ord_less_eq(set(X1),X33,X33),file('i/f/1/goal_138__Q_Restricted_Rewriting.qrste
fof(125, axiom,![X1]:![X24]:![X33]:![X34]:(member(X1,X24,X33)=>(~(member(X1,X24,X34))=>member(X1,X24,minus_
fof(127, axiom,![X1]:![X24]:![X33]:![X34]:(member(X1,X24,minus_minus(set(X1),X33,X34))=>~((member(X1,X24,X3
fof(131, axiom,![X1]:![X33]:(gg(set(X1),X33)=>collect(X1,aTP_Lamp_a(set(X1),fun(X1,bool),X33))=X33),file('i
fof(134, axiom,![X1]:(order(X1)=>![X35]:![X49]:((gg(X1,X35)&gg(X1,X49))=>(ord_less_eq(X1,X35,X49)=>(ord_les
fof(136, axiom,![X1]:(preorder(X1)=>![X35]:![X49]:![X50]:(ord_less_eq(X1,X35,X49)=>(ord_less_eq(X1,X49,X50)
fof(143, axiom,![X1]:![X33]:![X34]:(ord_less_eq(set(X1),X33,X34)<=>![X52]:(gg(X1,X52)=>(member(X1,X52,X33)=
fof(160, axiom,![X1]:![X39]:![X35]:![X33]:(pp(aa(X1,bool,X39,X35))=>(member(X1,X35,X33)=>?[X30]:(gg(X1,X30)
fof(171, axiom,![X1]:![X65]:![X66]:(pp(aa(X1,bool,aTP_Lamp_a(set(X1),fun(X1,bool),X65),X66))<=>member(X1,X6
fof(186, axiom,![X67]:semilattice_sup(set(X67)),file('i/f/1/goal_138__Q_Restricted_Rewriting.qrsteps_comp_s
```

Still the search space is huge: What can we learn?

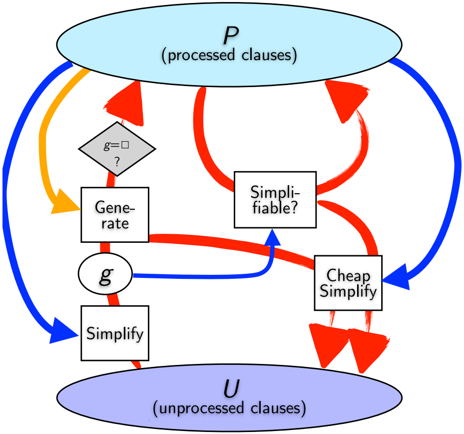
What has been learned

- CASC: Strategies
- AIM: Hints
- Hammers: Premises

What can be chosen in Superposition calculus

- Term ordering
- (Negative) literal selection
- Clause selection

E-Prover given-clause loop



Most important choice: unprocessed clause selection

[*Schulz 2015*]

Learning for E: Data Collection

Mizar top-level theorems

[*Urban 2006*]

- Encoded in FOF

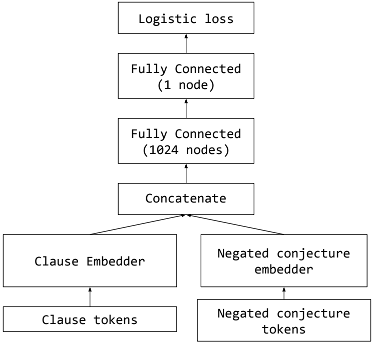
32,521 Mizar theorems with ≥ 1 proof

- training-validation split (90%-10%)
- replay with one strategy

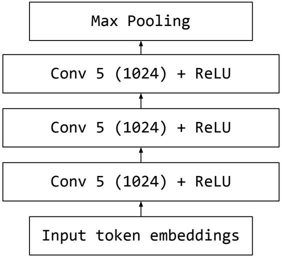
Collect all CNF intermediate steps

- and unprocessed clauses when proof is found

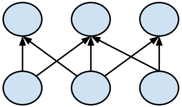
Deep Network Architectures

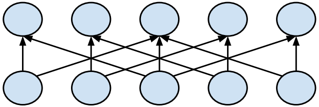


Overall network



Convolutional Embedding





Recursive Neural Networks

- Curried representation of first-order statements
- Separate nodes for `apply`, `or`, `and`, `not`
- Layer weights learned jointly for the same formula
- Embeddings of symbols learned with rest of network
- Tree-RNN and Tree-LSTM models[1]

[1] Relation to graphs

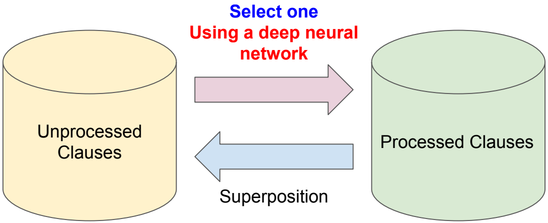
Model accuracy

Model	Embedding Size	Accuracy: 50-50% split
Tree-RNN-256×2	256	77.5%
Tree-RNN-512×1	256	78.1%
Tree-LSTM-256×2	256	77.0%
Tree-LSTM-256×3	256	77.0%
Tree-LSTM-512×2	256	77.9%
CNN-1024×3	256	80.3%
⋆CNN-1024×3	256	78.7%
CNN-1024×3	512	79.7%
CNN-1024×3	1024	79.8%
WaveNet-256×3×7	256	79.9%
⋆WaveNet-256×3×7	256	79.9%
WaveNet-1024×3×7	1024	81.0%
WaveNet-640×3×7(20%)	640	**81.5%**
⋆WaveNet-640×3×7(20%)	640	79.9%

⋆ = train on unprocessed clauses as negative examples

Improving Proof Search inside E

Overview



Problem

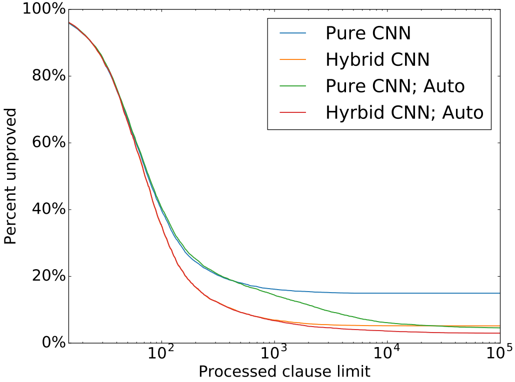
- Deep neural network evaluation is slow
- Slower than combining selected clause with all processed clauses[2]

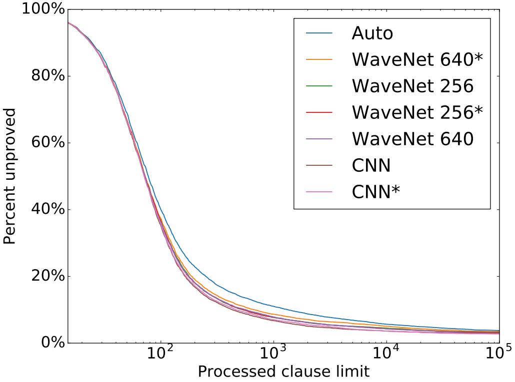
[2] State of 2016

Hybrid heuristic

Optimizations for performance

- Batching
- Combining TF with auto





Harder Mizar top-level statements

Model	DeepMath 1	DeepMath 2	Union of 1 and 2
Auto	578	581	674
⋆WaveNet 640	644	612	767
⋆WaveNet 256	692	712	864
WaveNet 640	629	685	997
⋆CNN	905	812	1,057
CNN	839	935	1,101
Total (unique)	1,451	1,458	1,712

Overall proved 7.4% of the harder statements

Harder Mizar top-level statements

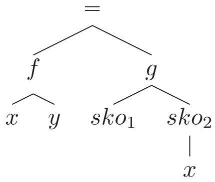
Model	DeepMath 1	DeepMath 2	Union of 1 and 2
Auto	578	581	674
⋆WaveNet 640	644	612	767
⋆WaveNet 256	692	712	864
WaveNet 640	629	685	997
⋆CNN	905	812	1,057
CNN	839	935	1,101
Total (unique)	1,451	1,458	1,712

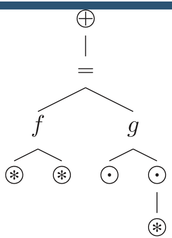
Overall proved 7.4% of the harder statements

- Batching and hybrid necessary
- Model accuracy unsatisfactory

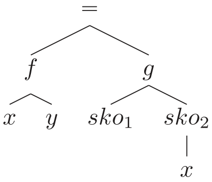
ENIGMA

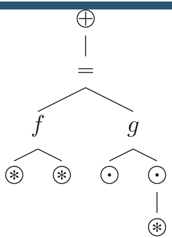
[*Jakubuv,Urban 2017*]





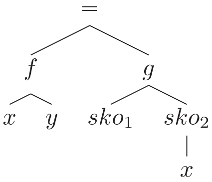
ENIGMA

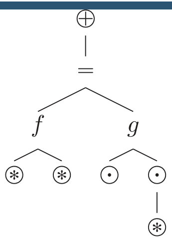




- Evaluation on AIM
- E's auto-schedule: 261
- Single best strategy: 239

ENIGMA





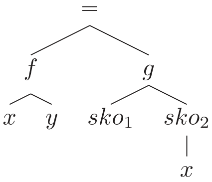
- Evaluation on AIM
- E's auto-schedule: 261
- Single best strategy: 239

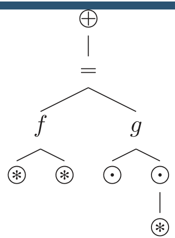
$$\textit{predict-weight}(C, \mathcal{M}) = \begin{cases} 1 & \text{iff } \textit{predict}(C, \mathcal{M}) = \boxplus \\ 10 & \text{otherwise} \end{cases}$$

$$\textit{weight}(C, \mathcal{M}) = \gamma \cdot \textit{length}(C) + \textit{predict-weight}(C, \mathcal{M})$$

ENIGMA

[*Jakubuv,Urban 2017*]





- Evaluation on AIM
- E's auto-schedule: 261
- Single best strategy: 239

$$\textit{predict-weight}(C, \mathcal{M}) = \begin{cases} 1 & \text{iff } \textit{predict}(C, \mathcal{M}) = \boxplus \\ 10 & \text{otherwise} \end{cases}$$

$$\textit{weight}(C, \mathcal{M}) = \gamma \cdot \textit{length}(C) + \textit{predict-weight}(C, \mathcal{M})$$

- Different trained models: 337
- Accuracy: 97.6%
- Looping and boosting
- Still in 30s: best trained strategy: 318

Automated Theorem Proving

Historical dispute: Gentzen and Hilbert

- Today two communities: Resolution (-style) and Tableaux

Possible answer: What is better in practice?

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

But still so far from humans?

- We can do learning much better for Tableaux
- And with ML beating brute force search in games, maybe?

leanCoP: Lean Connection Prover

[*Otten 2010*]

Connected tableaux calculus

- Goal oriented, good for large theories

Regularly beats Metis and Prover9 in CASC (ATP Systems 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

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

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

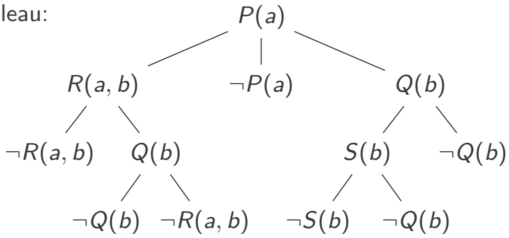
$$\frac{C_2 \setminus \{L_2\},\ M,\ Path \cup \{L_1\} \qquad C,\ M,\ Path}{C \cup \{L_1\},\ M,\ Path} \quad \textit{Extension}$$

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:



- Formula to prove:

$$(((\exists x Q(x) \vee \neg Q(c)) \Rightarrow P) \wedge (P \Rightarrow (\exists y Q(y) \wedge R))) \Rightarrow (P \wedge R)$$

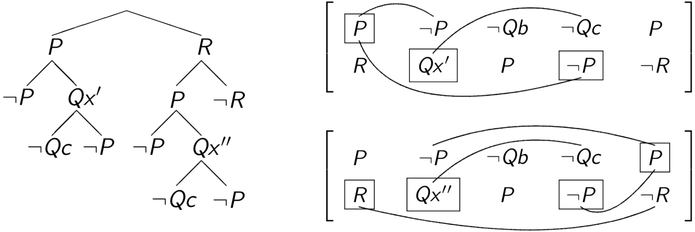
- DNF:

$$(P \wedge R) \vee (\neg P \wedge Qx) \vee (\neg Qb \wedge P) \vee (\neg Qc \wedge \neg P) \vee (P \wedge \neg R)$$

- Matrix:

$$\left[\begin{bmatrix} P \\ R \end{bmatrix} \begin{bmatrix} \neg P \\ Qx \end{bmatrix} \begin{bmatrix} \neg Qb \\ P \end{bmatrix} \begin{bmatrix} \neg Qc \\ \neg P \end{bmatrix} \begin{bmatrix} P \\ \neg R \end{bmatrix} \right]$$

- Tableaux:



leanCoP: Basic Code

```
prove([Lit|Cla],Path,PathLim,Lem,Set) :-

  (-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],PathLim,Lem,Set)
    ),

    prove(Cla,Path,PathLim,Lem,Set).
prove([],_,_,_,_).
```

leanCoP: Actual Code (Optimizations, No history)

```
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([],_,_,_,_).
```

First experiment: MaLeCoP in Prolog

[*Tableaux 2011*]

Select extension steps

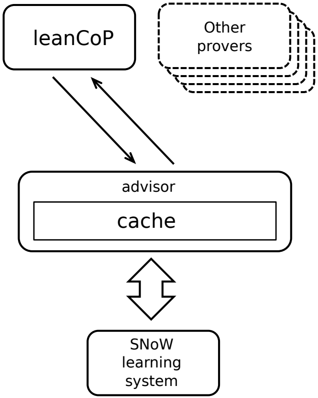
- Using external advice

Slow implementation

- 1000 less inf per second

Can avoid 90% inferences!

Important: Strategies



What about efficiency: FEMaLeCoP

[*LPAR 2015*]

Advise the:

- selection of clause for every tableau extension step

Proof state: weighted vector of symbols (or terms)

- extracted from all the literals on the active path
- Frequency-based weighting (IDF)
- Simple decay factor (using maximum)

Consistent clausification

- formula `?[X]: p(X)` becomes `p('skolem(?[A]:p(A),1)')`

Predictor: Custom sparse naive Bayes

- association of the features of the proof states
- with contrapositives used for the successful extension steps

FEMaLeCoP: Data Collection and Indexing

Extension of the saved proofs

- Training Data: pairs (path, used extension step)

External Data Indexing (incremental)

- `te_num`: number of training examples
- `pf_no`: map from features to number of occurrences $\in \mathbb{Q}$
- `cn_no`: map from contrapositives to numbers of occurrences
- `cn_pf_no`: map of maps of cn/pf co-occurrences

Problem Specific Data

- Upon start FEMaLeCoP reads
  - **only current-problem relevant** parts of the training data
- cn_no and cn_pf_no filtered by contrapositives in the problem
- pf_no and cn_pf_no filtered by possible features in the problem

Efficient Relevance (1/2)

Estimate the relevance of each contrapositive φ by

$$\text{P}(\varphi \text{ is used in a proof in state } \psi \mid \psi \text{ has features } \text{F}(\gamma))$$

where $F(\gamma)$ are the features of the current path.

Efficient Relevance (1/2)

Estimate the relevance of each contrapositive φ by

$$\text{P}(\varphi \text{ is used in a proof in state } \psi \mid \psi \text{ has features F}(\gamma))$$

where $F(\gamma)$ are the features of the current path.

Assuming the features are independent, this is:

$$\begin{aligned}
&P(\varphi \text{ is used in } \psi\text{'s proof}) \\
&\quad \cdot \prod_{f \in F(\gamma) \cap F(\varphi)} P\big(\psi \text{ has feature } f \mid \varphi \text{ is used in } \psi\text{'s proof}\big) \\
&\quad \cdot \prod_{f \in F(\gamma) - F(\varphi)} P\big(\psi \text{ has feature } f \mid \varphi \text{ is not used in } \psi\text{'s proof}\big) \\
&\quad \cdot \prod_{f \in F(\varphi) - F(\gamma)} P\big(\psi \text{ does not have } f \mid \varphi \text{ is used in } \psi\text{'s proof}\big)
\end{aligned}$$

Efficient Relevance (2/2)

All these probabilities can be estimated (using training examples):

$$\sigma_1 \ln t + \sum_{f \in (\overline{f} \cap \overline{s})} i(f) \ln \frac{\sigma_2 s(f)}{t} + \sigma_3 \sum_{f \in (\overline{f} - \overline{s})} i(f) + \sigma_4 \sum_{f \in (\overline{s} - \overline{f})} i(f) \ln(1 - \frac{s(f)}{t})$$

where

- $\overline{f}$ are the features of the path
- $\overline{s}$ are the features that co-occurred with φ
- $t = cn\_no(\varphi)$
- $s = cn\_fp\_no(\varphi)$
- i is the IDF
- σ_* are experimentally chosen parameters

Inference speed ... drops to about 40%

Prover	Proved (%)
OCaml-leanCoP	574 (27.6%)
FEMaLeCoP	635 (30.6%)
together	664 (32.0%)

(evaluation on MPTP bushy problems, 60 s)

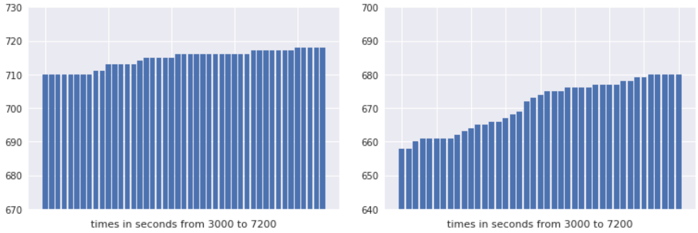
On various datasets, 3–15% problems more solved than leanCoP
(run for double the time)

What about stronger learning?

Yes, but...

[*Michalewski 2017*]

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



NBayes vs XGBoost on 2h timeout

Preliminary experiments with deep learning

[*Olšak 2017*]

- So far too slow

Is theorem proving just a maze search?

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.

Can we provide a tree search like interface?

- Two functions suffice

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

$$\text{action} : \text{action} \rightarrow \text{state}$$

- where

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

Is it ok to change the tree?

Most learning for games sticks to game dynamics

- Only tell it how to do the moves

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!)

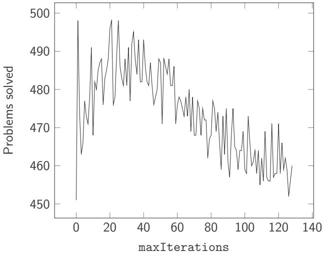
But with random playouts: depth hundreds of thousands!

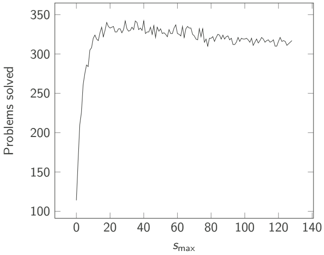
- Just unlikely to find a proof $\rightarrow$ learning

Monte Carlo First Try: MONTECOP

Use Monte Carlo playouts to guide restricted backtracking

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





“Simple” learning in leanCoP

FEMaLeCoP: Speed: 40%

On various datasets, 3–15% problems more solved than leanCoP

XGBoost: Speed: 8%

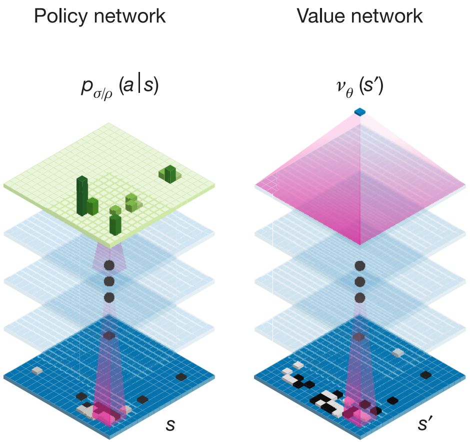
But more precise and again small improvement

Monte Carlo

- Improves on leanCoP, but not by a big margin
- Change in game moves
- More inspiration from games?

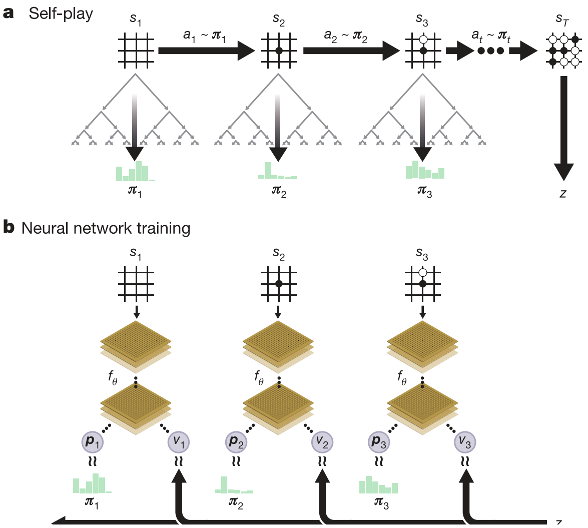
AlphaZero (1/3)

[*Silver et al.*]

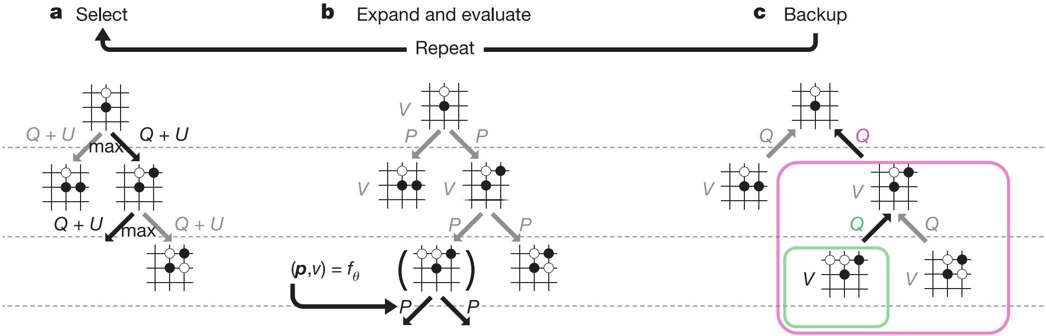


AlphaZero (2/3)

[*Silver et al.*]



[Silver et al.]


a Select
b Expand and evaluate
c Backup
Repeat
Q + U
max
Q + U
Q + U
max
Q + U
V
P
P
V
V
P
P
(p,v) = f_θ
P
P
Q
Q
V
V
Q
Q
V
V

How to select the best action?

[*Szepesvari 2006*]

Intuition

- Given some prior probabilities
- And having done some experiments
- Which action to take?
- (later extended to sequences of actions in a tree)

How to select the best action?

[Szepesvari 2006]

Intuition

- Given some prior probabilities
- And having done some experiments
- Which action to take?
- (later extended to sequences of actions in a tree)

$\frac{w_i}{n_i}$ average reward $\qquad p_i$ action i prior

N number of experiments $\qquad n_i$ action i experiments

How to select the best action?

[*Szepesvari 2006*]

Intuition

- Given some prior probabilities
- And having done some experiments
- Which action to take?
- (later extended to sequences of actions in a tree)

Monte Carlo Tree Search with Upper Confidence Bounds for Trees

- Select node n maximizing

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

- where

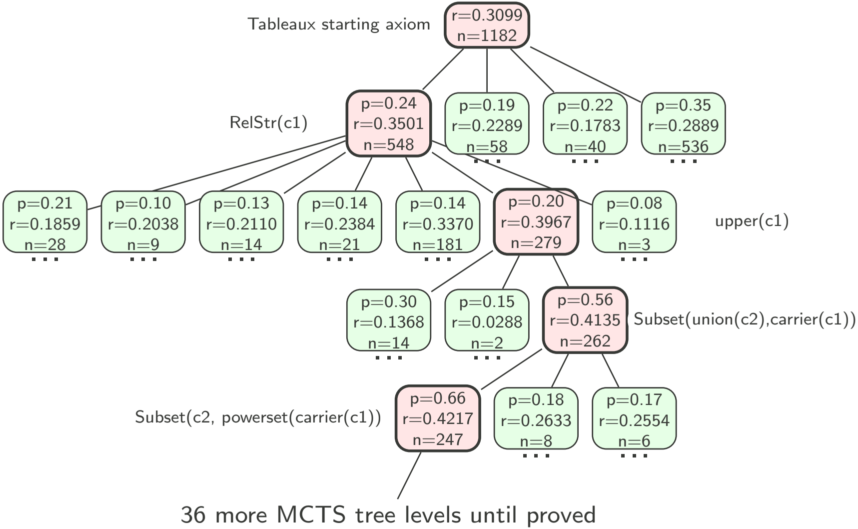
$\frac{w_i}{n_i}$ average reward

p_i action i prior

N number of experiments

n_i action i experiments

MCTS tree for WAYBEL_0:28



Learn Policy and Value

Policy: Which actions to take?

- Proportions predicted based on proportions in similar states

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

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

Mizar TPTP problems: train (29272) and test (3252) sets

Baseline

System	leanCoP	playouts	UCT
Test	**1143**	431	804

Mizar TPTP problems: train (29272) and test (3252) sets

Baseline

System	leanCoP	playouts	UCT
Test	**1143**	431	804

10 iterations

Iteration	1	2	3	4	5
Test	1354	1519	1566	1595	**1624**

Mizar TPTP problems: 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	9	10
Train	12325	13749	14155	14363	14403	14431	14342	**14498**	14481	14487
Test	1354	1519	1566	1595	**1624**	1586	1582	1591	1577	1621

Mizar TPTP problems: train (29272) and test (3252) sets

Baseline

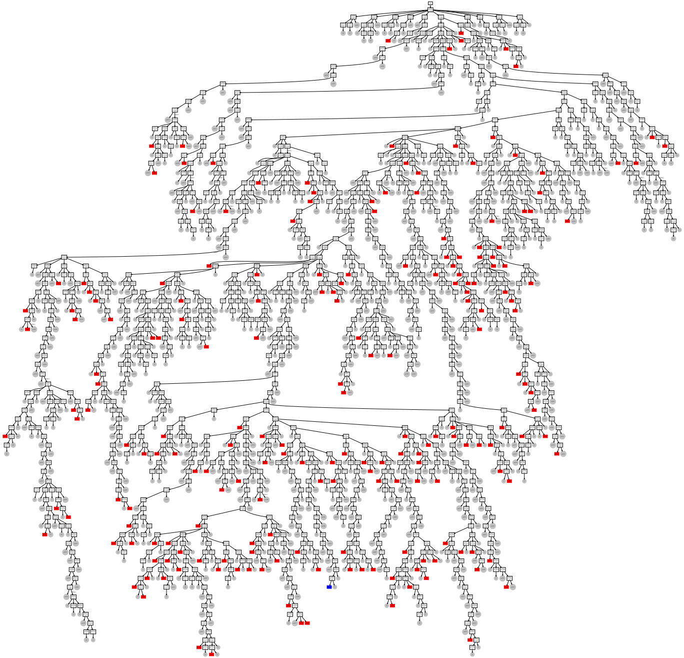
System	leanCoP	playouts	UCT
Train	10438	4184	7348
Test	**1143**	431	804

10 iterations

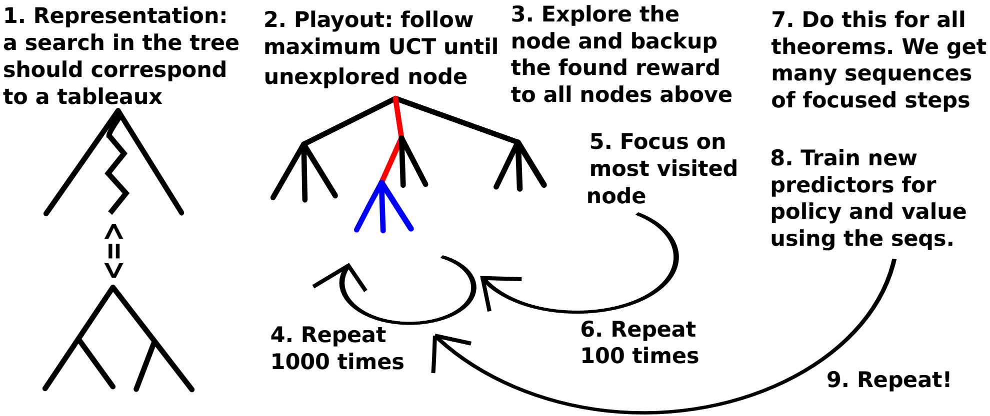
Iteration	1	2	3	4	5	6	7	8	9	10
Train	12325	13749	14155	14363	14403	14431	14342	**14498**	14481	14487
Test	1354	1519	1566	1595	**1624**	1586	1582	1591	1577	1621

More Time

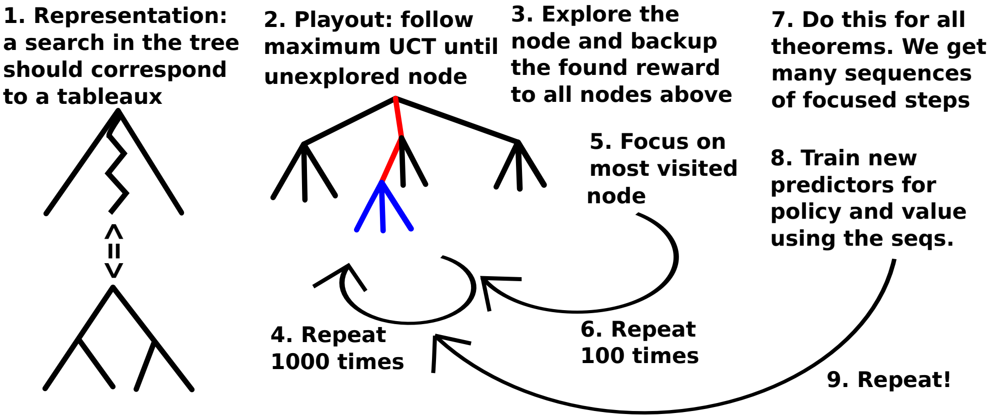
leanCoP, 4M inferences, strategies	1396
rlCoP union	1839



RL-CoP setup summary



RL-CoP setup summary

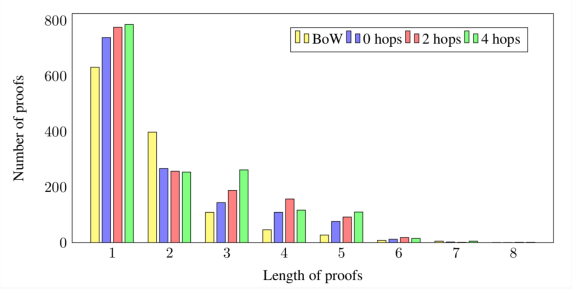


Theorem proving requiring significant hardware

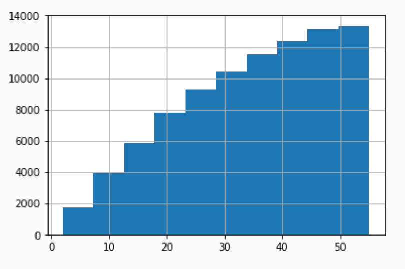
ATP versus learned ATP

ATPs tend to find short proofs.

- Learning helps only minimally



Graph Representations for Higher-Order Logic and Theorem Proving [*A. Paliwal et.al., 2019*]



Cumulative proof lengths of rlCoP on the Mizar Mathematical Library [*NeurIPS 2018*]

Main aims of FLoP

[*Zombori'2019*]

- Build an internal guidance system that can find long proofs
- Find a domain that is simple enough to analyse the inner workings of the prover
- At first, try to learn from very few problems (with given or without given proofs)
- Try to generalize to harder problems (longer proofs) with a similar structure

Domain: Robinson Arithmetic

Name	Axiom
zeroSuccessor	$\forall X : \neg(o = X)$
diffSuccessor	$\forall X, Y : (s(X) = s(Y)) \Rightarrow (X = Y)$
addZero	$\forall X : plus(X, o) = X$
addSuccessor	$\forall X, Y : plus(X, s(Y)) = s(plus(X, Y))$
mulZero	$\forall X : mul(X, o) = o$
mulSuccessor	$\forall X, Y : mul(X, s(Y)) = plus(mul(X, Y), X)$

- Prove simple ground equalities
- Proofs are non trivial, but have a strong shared structure
- Proof lengths can get very long as numbers increase
- See how little supervision is required to learn some proof types

Challenges for RL for TP

- Theorem proving as a 1 person game
- Meta-Learning task: each problem is a new maze: train on some, evaluate on others
- Sparse, binary rewards
- Defining good features
- Action space not fixed: different across steps and across problems

Challenges for RL for TP

- Theorem proving as a 1 person game
- Meta-Learning task: each problem is a new maze: train on some, evaluate on others
- Sparse, binary rewards
- Defining good features
- Action space not fixed: different across steps and across problems

Algorithm 1 FLoP: Curriculum Learning on Proofs

Input: problem set $\mathcal{P}$, policy π, progress threshold $\in [0..1]$
train steps $\in \mathbb{N}$, episodes between updates: k $\in \mathbb{N}$
Output: trained policy π, possible new proofs for problems in $\mathcal{P}$

steps $\leftarrow$ 0
curriculum $\leftarrow$ *1*
while steps < train steps **do**
  successes $\leftarrow$ 0
  for j in 1..k **do**
    $p \leftarrow$ random problem from problem set $\mathcal{P}$ ▷ An episode corresponds to a problem
    if p has stored proof **then** ▷ Determine initial state
      Take proof steps according to stored proof until *curriculum* number of steps remain
      $s_0 \leftarrow$ state of problem p after initial proof steps taken
    else
      $s_0 \leftarrow$ starting state of problem p
    while not episode over **do**
      Take action according to policy $\pi(a_i|s_i)$, observe next state s_{i+1} and reward r_{i+1}
      steps $\leftarrow$ steps + 1
    if proof is found for p **then**
      successes $\leftarrow$ successes + 1
      if found proof is shorter than previous proof **then**
        store proof as new proof for p
      if no proof of p was known before **then**
        curriculum $\leftarrow$ 1 ▷ Restart curriculum learning
  Update policy π
  success rate $\leftarrow$ successes / k
  if success rate > progress threshold **then**
    curriculum $\leftarrow$ *curriculum* + 1 ▷ Advance curriculum

FLoP

External guidance based on RL

- Theorem Prover encapsulated as an environment
- Use curriculum learning

Applicable when we know the proof of a problem

- More efficient use of training signals
- Start learning from the end of the proof
- Gradually move starting step towards the beginning of proof

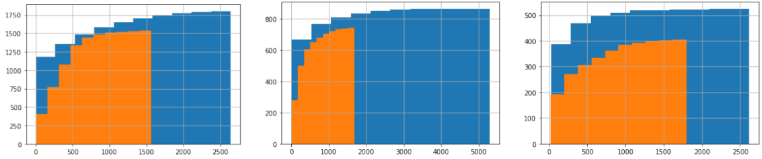
Proximal Policy Optimization (PPO)

- Actor learns a policy (what steps to take)
- Critic learns a value (how promising is a proof state)
- Actor is confined to change slowly to increase stability

Datasets

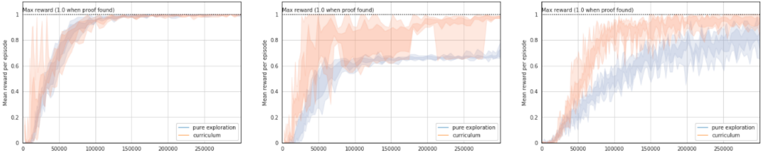
Stage	Set	Size	Description
Stage 1	Train	2	$1+1=2$, $1\cdot 1=1$
	Eval	1800	Expressions of the form $N_1+N_2=N_3$, $N_1\cdot N_2=N_3$, where $0\le N_i<30$. (Examples: $3{+}4{=}7$ or $5{\cdot}12{=}60$)
Stage 2	Train	3	$1+1=2$, $1\cdot 1=1$, $1\cdot 1\cdot 1=1$
	Eval	1000	$T=N$, where $0\le N$, and T is a random expression with 3 operators and operands N_i such that $0\le N_i<10$. (E.g.: $((3{+}4){\cdot}2){+}6{=}20$)
Stage 3	Train	810	$T_1=T_2$, where T_1 and T_2 are random expressions with 3 operators and operands N_i such that $0\le N_i<2$.
	Eval	1000	$T_1=T_2$, where T_1 and T_2 are random expressions with 3 operators and operands N_i such that $2\le N_i<10$. (E.g. $((3{+}4){\cdot}2){+}6{=}((1{+}1){\cdot}5){\cdot}2$)

Evaluation



(a) (b) (c)

Figure 4: *Comparing the length of proofs found by* FLoP *(blue) and* rlCoP *(orange) on the Robinson Arithmetic dataset. All figures are cumulative histograms, vertical axes show the number of proofs, horizontal axes show the length of proofs. Best models are shown for both* FLoP *and* rlCoP*. Figures (a), (b), (c) correspond to Stage 1, 2, 3 respectively.* FLoP *found more proofs in all stages.*



(a) (b) (c)

Figure 5: *(a)-(c) – Stages 1-3, training graphs centered at the mean reward, darker bars are delimited by quantiles at 0.25 and 0.75, lighter bars extending from min to max; in total 36 models, 6 models per graph, 20M samples per experiment. Curriculum helps in Stages 2 and 3.*

Learning for ATPs: Summary and next steps

For some calculi major improvement

- Learning for Resolution-style systems open

Learn features

RL prefers shorter proofs

- but they may not be the ones that generalize best

Evaluate with backtracking

Scale to more interesting domains

- Bolzano–Weierstrass

Communication with a Proof Assistant

The prover does not get what I mean

- Completely clear things need to be fully expanded
- Even if I said it 100 times, I have to say it again
  - (or implement the expansion)

Compared to a student

- Proof assistant does not get what I mean
- Cannot repeat a simple action

Proof assistant – assistant

Given some text, the assistant can say

- What you wrote
- What you wanted to write
  - (What I think you meant)
- Does it make sense
  - Can I be convinced of this
  - (Can I prove this*)

Proof assistant – assistant

Given some text, the assistant can say

- What you wrote
- What you wanted to write
  - (What I think you meant)
- Does it make sense
  - Can I be convinced of this
  - (Can I prove this*)

Tasks

- Understand LaTeX formulas, as well as some text
- Translate it to logic (of a/the proof assistant)
- Report on the success

Questions

- Can we (a computer) learn how to state lemmas formally?
- Can we (a computer) learn to prove?

Demo

Strong-semantics probabilistic parser for HOL Light

Input the formula to parse. Separate symbols with spaces:

sin 0 = cos pi / 2 [Submit]

debug: cache→ decode → 18 bigram&trigram features → 1024 nearest neighbours → 16 nyk parses → 12 distinct terms

Conjecture as HOL Light term:	Type info:	Automatically Provable?		Time
sin (&0) = cos pi / &2		disproved		(6.74s)
sin (&0) = cos (pi / &2)		yes	REWRITE_TAC [SIN_0; COS_PI2]	(0.87s)
csin (Cx (&0)) = Cx (cos (pi / &2))		yes	REWRITE_TAC [CSIN_0; COS_PI2]	(0.74s)
csin (Cx (&0)) = ccos (Cx (pi / &2))		yes	MESON_TAC [NUMERAL; CX_COS; CSIN_0; COS_PI2]	(0.76s)
Cx (sin (&0)) = ccos (Cx (pi / &2))		yes	MESON_TAC [SIN_0; NUMERAL; CX_COS; COS_PI2]	(0.70s)
Cx (sin (&0)) = Cx (cos (pi / &2))		yes	REWRITE_TAC [SIN_0; COS_PI2]	(0.80s)
csin (Cx (&0)) = ccos (Cx pi / Cx (&2))		yes	MESON_TAC [NUMERAL; CX_DIV; CX_COS; CSIN_0; COS_PI2]	(0.93s)
csin (Cx (&0)) = ccos (Cx pi) / Cx (&2)		no advice		
csin (Cx (&0)) = Cx (cos pi) / Cx (&2)		no advice		
Cx (sin (&0)) = ccos (Cx pi / Cx (&2))		yes	MESON_TAC [SIN_0; NUMERAL; CX_DIV; CX_COS; COS_PI2]	(1.23s)

Why don't we have this? (1/2)

Claus Zinn and others tried and have not arrived very far because:

- lack of background knowledge
- lack of powerful automated reasoning
- lack of self-adapting translation

Why don't we have this? (1/2)

Claus Zinn and others tried and have not arrived very far because:

- lack of background knowledge
- lack of powerful automated reasoning
- lack of self-adapting translation

But huge machine learning progress

Why don't we have this? (2/2)

Controlled languages

- Ranging from Naproche and MathLang to Mizar

Easy start but huge number of patterns

100 most frequent patterns cover half of 42,931 ProofWiki sentences [CICM'14]

```
5829  Let $?$ be [?].
2688  Let $?$.
774   Then $?$ is [?].
736   Let $?$ be [?] of $?$.
724   Let $?$ and $?$ be [?].
578   Let $?$ be the [?] of $?$.
555   Let $?$ be the [?].
```

But can go very far

- Thousands of manually entered patterns [Matsuzaki+'16,'17]
- Better than humans on university entrance exams (some domains) [Arai+'18]

Learning data: Aligned corpora

- Dense Sphere Packings: A Blueprint for Formal Proofs
  - 400 theorems and 200 concepts mapped [*Hales13*]
- IsaFoR [*SternagelThiemann14*]
  - most of "Term Rewriting and All That" [*BaaderNipkow*]
- Compendium of Continuous Lattices (CCL)
  - 60% formalized in Mizar [*BancerekRudnicki02*]
  - high-level concepts and theorems aligned
- Feit-Thompson theorem by Gonthier [*Gonthier13*]
  - Two graduate books
- detailed proofs and symbol linking in Wikipedia, ProofWiki, PlanetMath, ...

Aligned corpora: Kepler Example

Informal Formal

Definition of [fan, blade] DSKAGVP (fan) [fan ↔ FAN]

Let (V, E) be a pair consisting of a set $V \subset \mathbb{R}^3$ and a set E of unordered pairs of distinct elements of V. The pair is said to be a *fan* if the following properties hold.

1. (CARDINALITY) V is finite and nonempty. [cardinality ↔ fan1]
2. (ORIGIN) $\mathbf{0} \notin V$. [origin ↔ fan2]
3. (NONPARALLEL) If $\{\mathbf{v}, \mathbf{w}\} \in E$, then $\mathbf{v}$ and $\mathbf{w}$ are not parallel. [nonparallel ↔ fan6]
4. (INTERSECTION) For all $\varepsilon, \varepsilon' \in E \cup \{\{\mathbf{v}\} \;:\; \mathbf{v} \in V\}$, [intersection ↔ fan7]

$$C(\varepsilon) \cap C(\varepsilon') = C(\varepsilon \cap \varepsilon').$$

When $\varepsilon \in E$, call $C^0(\varepsilon)$ or $C(\varepsilon)$ a *blade* of the fan.

basic properties

The rest of the chapter develops the properties of fans. We begin with a completely trivial consequence of the definition.

Informal Formal

Lemma [] CTVTAQA (subset-fan)

If (V, E) is a fan, then for every $E' \subset E$, (V, E') is also a fan.

Proof

This proof is elementary.

Informal Formal

Lemma [fan cyclic] XOHLED

$[E(v) \leftrightarrow$ set_of_edge$]$ Let (V, E) be a fan. For each $\mathbf{v} \in V$, the set

$$E(\mathbf{v}) = \{\mathbf{w} \in V \;:\; \{\mathbf{v}, \mathbf{w}\} \in E\}$$

is cyclic with respect to $(\mathbf{0}, \mathbf{v})$.

Proof

If $\mathbf{w} \in E(\mathbf{v})$, then $\mathbf{v}$ and $\mathbf{w}$ are not parallel. Also, if $\mathbf{w} \neq \mathbf{w}' \in E(\mathbf{v})$, then

Informal Formal

```
#DSKAGVP?
let FAN=new_definition`FAN(x,V,E) <=> ((UNIONS E) SUBSET V) /\ graph(E) /\ fan1(x,V,E) /\ fan2(x,V,
fan6(x,V,E)/\ fan7(x,V,E)`;;
```

basic properties

The rest of the chapter develops the properties of fans. We begin with a completely trivial consequence of the definition.

Informal Formal

```
let CTVTAQA=prove(`!(x:real^3) (V:real^3->bool) (E:(real^3->bool)->bool) (E1:(real^3->bool)->bool)
FAN(x,V,E) /\ E1 SUBSET E
==>
FAN(x,V,E1)`,

REPEAT GEN_TAC
THEN REWRITE_TAC[FAN;fan1;fan2;fan6;fan7;graph]
THEN ASM_SET_TAC[]);;
```

Informal Formal

```
let XOHLED=prove(`!(x:real^3) (V:real^3->bool) (E:(real^3->bool)->bool) (v:real^3).
FAN(x,V,E) /\ v IN V
==> cyclic_set (set_of_edge v V E) x v`,

MESON_TAC[CYCLIC_SET_EDGE_FAN]);;
```

Aligned corpora: Kepler Example

596 formulas from the Flyspeck book extracted with LaTeXML

- Translation to HOL Light based on a small table
- 17% same as formal ones

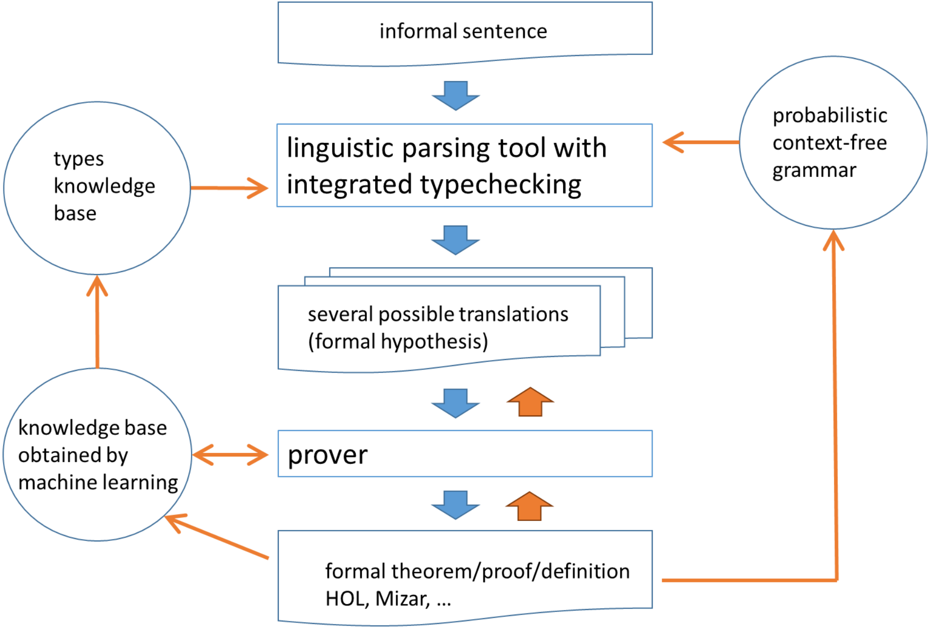
Too hard

- make more precise examples or [*ongoing*]
- start with simpler ones [*ITP'15 +*]

Informalization

22000 Flyspeck statements informalized

- 72 overloaded instances like "+" for `vector_add`
- 108 infix operators
- forget all "prefixes"
  - `real_, int_, vector_, nadd_, hreal_, matrix_, complex_`
  - `ccos, cexp, clog, csin, ...`
  - `vsum, rpow, nsum, list_sum, ...`
- Deleting all brackets, type annotations, and casting functors
  - `Cx` and `real_of_num` (which alone is used 17152 times).


informal sentence
types knowledge base
linguistic parsing tool with integrated typechecking
probabilistic context-free grammar
several possible translations (formal hypothesis)
knowledge base obtained by machine learning
prover
formal theorem/proof/definition HOL, Mizar, ...

CYK and parsing — just a little

Induce PCFG (probabilistic context-free grammar) from term trees

- inner nodes → rules   frequencies → probabilities

Binarize PCFG grammar for efficiency

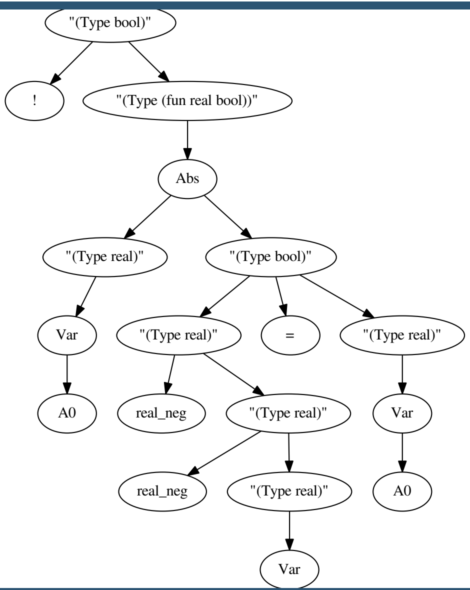
CYK parses ambiguous sentences

- outputs most probable parse trees
- tweak: small probability for each symbol to be a variable

Pruning

- Compatible types for free variables in subtrees
- HOL type-checking
- Hammer

Example tree inducing grammar



Statistics

Just PFG

[ITP'15]

- 39.4% of the Flyspeck sentences parsed correctly
- average rank: 9.34

Problems with PCFG and CYK

$$1 * x + 2 * x$$

Statistics

Just PFG

[ITP'15]

- 39.4% of the Flyspeck sentences parsed correctly
- average rank: 9.34

Problems with PCFG and CYK

$$1 * x + 2 * x$$

Use deeper trees

[ITP 2017]

- semantic pruning + subtree depth 4-8 + substitution trees
- 83% sentences parsed correctly
- average rank: 1.93

Types helped us - what about no types?

Mizar

- Developed by mathematicians for mathematicians
- Many features significantly different from the usual

How would you formalize:

1. Sum of the Result of Operation with Each Element of a Set

For simplicity, we adopt the following convention: X denotes a real unitary space, x, y, y_1, y_2 denote points of X, i, j denote natural numbers, D_1 denotes a non empty set, and p_1, p_2 denote finite sequences of elements of D_1.

Next we state the proposition

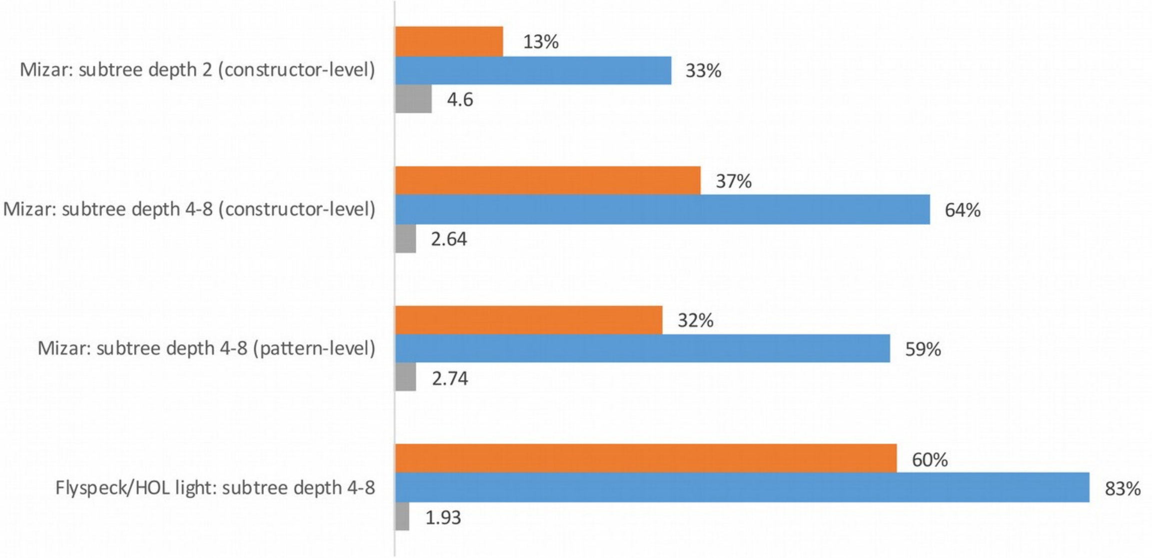
(1) Suppose p_1 is one-to-one and p_2 is one-to-one and $\operatorname{rng} p_1 = \operatorname{rng} p_2$. Then $\operatorname{dom} p_1 = \operatorname{dom} p_2$ and there exists a permutation P of $\operatorname{dom} p_1$ such that $p_2 = p_1 \cdot P$ and $\operatorname{dom} P = \operatorname{dom} p_1$ and $\operatorname{rng} P = \operatorname{dom} p_1$.

Let D_1 be a non empty set and let f be a binary operation on D_1. Let us assume that f is commutative and associative and has a unity. Let Y be a finite subset of D_1. The functor $f \oplus Y$ yields an element of D_1 and is defined as follows:

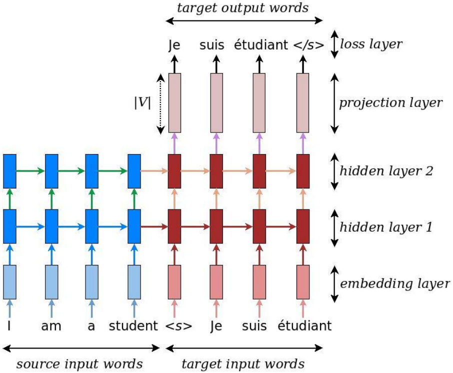
(Def. 1) There exists a finite sequence p of elements of D_1 such that p is one-to-one and $\operatorname{rng} p = Y$ and $f \oplus Y = f \odot p$.

Let us consider X and let Y be a finite subset of the carrier of X. The functor SetopSum(Y, X) is defined as follows:

Mizar Statistics

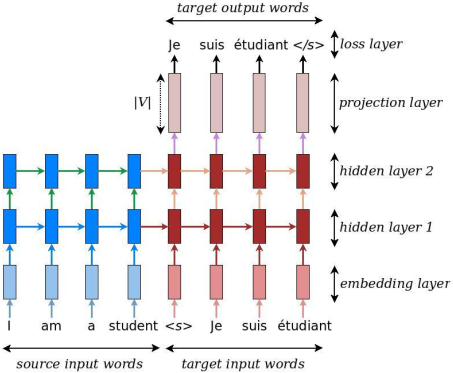


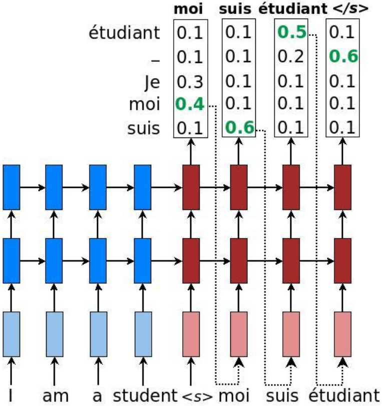
Sequence-to-sequence models: decoder/encoder RNN



[Luong et al'15]

Sequence-to-sequence models: decoder/encoder RNN





[*Luong et al'15*]

Neural Auto-formalization

	Identical Statements	0
Best Model - 1024 Units	69179 (total) 22978 (no-overlap)	65.73% 47.77%

Neural Auto-formalization

	Identical Statements	0
Best Model - 1024 Units	69179 (total) 22978 (no-overlap)	65.73% 47.77%
Top-5 Greedy Cover - 1024 Units - 4-Layer Bi. Res. - 512 Units - 6-Layer Adam Bi. Res. - 2048 Units	78411 (total) 28708 (no-overlap)	74.50% 59.68%
Top-10 Greedy Cover - 1024 Units - 4-Layer Bi. Res. - 512 Units - 6-Layer Adam Bi. Res. - 2048 Units - 2-Layer Adam Bi. Res. - 256 Units - 5-Layer Adam Res. - 6-Layer Adam Res. - 2-Layer Bi. Res.	80922 (total) 30426 (no-overlap)	76.89% 63.25%
Union of All 39 Models	83321 (total) 32083 (no-overlap)	79.17% 66.70%

Neural Auto-formalization

[*CICM'18*]

	Identical Statements	0	≤ 1	≤ 2	≤ 3
Best Model - 1024 Units	69179 (total) 22978 (no-overlap)	65.73% 47.77%	74.58% 59.91%	86.07% 70.26%	88.73% 74.33%
Top-5 Greedy Cover - 1024 Units - 4-Layer Bi. Res. - 512 Units - 6-Layer Adam Bi. Res. - 2048 Units	78411 (total) 28708 (no-overlap)	74.50% 59.68%	82.07% 70.85%	87.27% 78.84%	89.06% 81.76%
Top-10 Greedy Cover - 1024 Units - 4-Layer Bi. Res. - 512 Units - 6-Layer Adam Bi. Res. - 2048 Units - 2-Layer Adam Bi. Res. - 256 Units - 5-Layer Adam Res. - 6-Layer Adam Res. - 2-Layer Bi. Res.	80922 (total) 30426 (no-overlap)	76.89% 63.25%	83.91% 73.74%	88.60% 81.07%	90.24% 83.68%
Union of All 39 Models	83321 (total) 32083 (no-overlap)	79.17% 66.70%	85.57% 76.39%	89.73% 82.88%	91.25% 85.30%

Machine Learning applied to informal LaTeX

For $\bigoplus_{n=1,\dots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ??. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example ??. It may replace S by $X_{spaces,étale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma ??. Namely, by Lemma ?? we see that R is geometrically regular over S.

Lemma 0.1. *Assume (3) and (3) by the construction in the description.*

Suppose $X = \lim |X|$ (by the formal open covering X and a single map $\underline{Proj}_X(\mathcal{A}) = \mathrm{Spec}(B)$ over U compatible with the complex

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If T is surjective we may assume that T is connected with residue fields of S. Moreover there exists a closed subspace $Z \subset X$ of X where U in X' is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem

(1) *f is locally of finite type. Since $S = \mathrm{Spec}(R)$ and $Y = \mathrm{Spec}(R)$.*

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\dots,n} U_i$ be the scheme X over S at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{\mathcal{X},\dots,0}$.

Lemma 0.2. *Let X be a locally Noetherian scheme over S, $E = \mathcal{F}_{X/S}$. Set $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. Since $\mathcal{I}^n \subset \mathcal{I}^n$ are nonzero over $i_0 \leq \mathfrak{p}$ is a subset of $\mathcal{J}_{n,0} \circ \overline{A}_2$ works.*

Lemma 0.3. *In Situation ??. Hence we may assume $\mathfrak{q}' = 0$.*

Proof. We will use the property we see that $\mathfrak{p}$ is the mext functor (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F-algebra where δ_{n+1} is a scheme over S. □

[*Karpathy'16*]

Final Summary / Take Home

- Proofs are hard
- Machine learning key to most powerful proof assistant automation
- Older but very efficient algorithms with significant adjustments
- Many other learning problems and scenarios

Not covered

- Learning strategy selection [*Jakubuv,Urban*]
- Kernel methods [*Kühlwein*]
- Deep Prolog [*Rocktäschel*]
- Semantic Features, Conecturing
- Tactic selection [*Nagashima,...*]
- SVM [*Holden*]
- Adversarial Networks [*Szegedy*]
- Human proof optimization
- Theory exploration [*Bundy+*]
- Concept Alignment [*Gauthier*]
- ...