MizAR 60 for Mizar 50

J. Jakubův[1] K. Chvalovský[1] Z. Goertzel[1] C. Kaliszyk[2] M. Olšák[3] B. Piotrowski[1] S. Schulz[4] M. Suda[1] J. Urban[1]

AITP'23, Aussois, France, 5th September 2023

[1]Czech Technical University in Prague, Prague, Czech Republic
[2]University of Innsbruck, Austria and INDRC, Prague, Czech Republic
[3]Institut des Hautes tudes Scientifiques, Paris, France
[4]DHBW Stuttgart, Stuttgart, Germany

Outline

Introduction: Mizar, MML, Hammers and AITP

ENIGMA: ATP Guidance and Related Technologies

Learning Premise Selection From the MML

Experiments and Results

Overview of the Methods and Experiments

- MML: Mizar Mathematical Library
  = large corpus of formalized mathematics

Overview of the Methods and Experiments

- MML: Mizar Mathematical Library
  = large corpus of formalized mathematics
- MPTP: Mizar Problems for Theorem Provers
  = translation of MML theorems to First-Order Logic (FOL)

Overview of the Methods and Experiments

- MML: Mizar Mathematical Library
  = large corpus of formalized mathematics
- MPTP: Mizar Problems for Theorem Provers
  = translation of MML theorems to First-Order Logic (FOL)
- ATP: Automated Theorem Provers for FOL
- MizAR: Mizar + ATPs (*Automated Reasoning*)

Overview of the Methods and Experiments

- MML: Mizar Mathematical Library
  = large corpus of formalized mathematics
- MPTP: Mizar Problems for Theorem Provers
  = translation of MML theorems to First-Order Logic (FOL)
- ATP: Automated Theorem Provers for FOL
- MizAR: Mizar + ATPs (*Automated Reasoning*)
- Evaluation in two premise selection schemes:
  1. **Bushy**: Use the premises extracted from user-written dependencies in Mizar files.
  2. **Hammer**: Use all theorems available in MML (at the given time) as premises.

Overview of the Methods and Experiments

- MML: Mizar Mathematical Library
  = large corpus of formalized mathematics
- MPTP: Mizar Problems for Theorem Provers
  = translation of MML theorems to First-Order Logic (FOL)
- ATP: Automated Theorem Provers for FOL
- MizAR: Mizar + ATPs (*Automated Reasoning*)
- Evaluation in two premise selection schemes:
  1. **Bushy**: Use the premises extracted from user-written dependencies in Mizar files.
  2. **Hammer**: Use all theorems available in MML (at the given time) as premises.
- Challenge: Use machine learning (ML) and ATPs to improve.
- Previously at Mizar40: 56% (bushy) / 40.6% (hammer)

AITP Challenges/Bets from 2014

- 3 AITP bets from JU's 2014 talk at Institut Henri Poincare
  - In 20 years, 80% of Mizar and Flyspeck toplevel theorems will be provable automatically (same hardware, same libraries as in 2014 - about 40% then)
  - The same in 30 years - I'll give you 2:1. In 10 years: 60% (i.e. a 50% more than Mizar40/Flyspeck)
  - In 25 years, 50% of the toplevel statements in LaTeX-written Msc-level math curriculum textbooks will be parsed automatically and with correct formal semantics
  - No betting: all this could be today done in 5 years with reasonable resources
  - Hurry up: I will only accept bets up to 10k EUR total
- The 50% improvement promised in my 2014 ERC proposal
- TacticToe has done 66%-69% on HOL already in 2017/18
- Hence we were left with the Mizar 60% challenge

ENIGMA: Machine Learning for ATPs

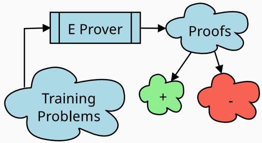
- ENIGMA: *Efficient Learning Based Inference Guiding Machine*
- Machine learning for given clause selection guidance in ATPs.
- Reported at IJCAR'20: Learning-guided ATP that improves over state-of-the-art ATPs (IJCAR'20).
- ENIGMA and other ML-based methods are used on Mizar.
- Experiments typically involve train/evaluation loop.

ENIGMA: Training & Evaluation Loop

1. Evaluate problems using a prover.

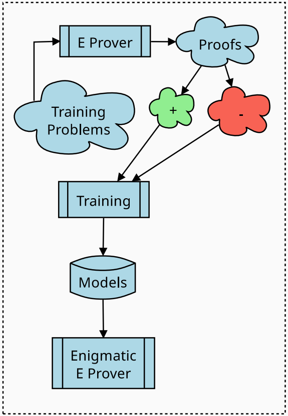
ENIGMA: Training & Evaluation Loop

1. Evaluate problems using a prover.
2. Obtain train samples from proofs:
   - positive = proof clauses
   - negative = other processed



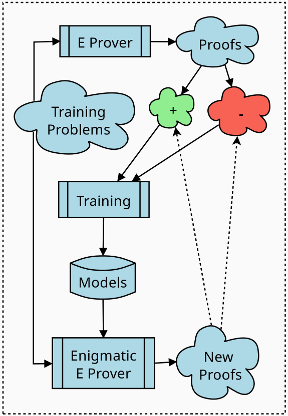
ENIGMA: Training & Evaluation Loop

1. Evaluate problems using a prover.
2. Obtain train samples from proofs:
   - positive = proof clauses
   - negative = other processed
3. Train a classifier model.
4. Use the model inside the prover.



ENIGMA: Training & Evaluation Loop

1. Evaluate problems using a prover.
2. Obtain train samples from proofs:
   - positive = proof clauses
   - negative = other processed
3. Train a classifier model.
4. Use the model inside the prover.
5. Obtain new proofs and go to 2.



Outline

Saturation Theorem Proving Meets Machine Learning

- ATPs typically perform proof search by contradiction.
- Statements are translated to clauses (literal disjuction).

Saturation Theorem Proving Meets Machine Learning

- ATPs typically perform proof search by contradiction.
- Statements are translated to clauses (literal disjuction).
- Proof search is guided by the *given clause algorithm*.
- Inference rules allow to infer a new clause.

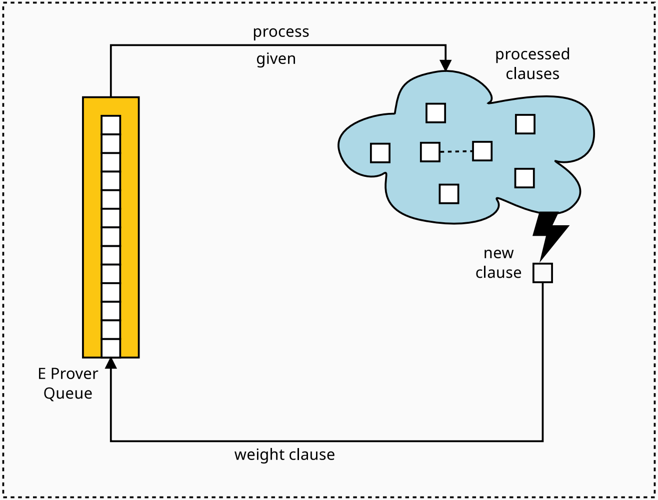
Saturation Theorem Proving Meets Machine Learning

- ATPs typically perform proof search by contradiction.
- Statements are translated to clauses (literal disjuction).
- Proof search is guided by the *given clause algorithm*.
- Inference rules allow to infer a new clause.
- The proof state consists of two subsets
    1. processed clauses: mutually inferred
    2. unprocessed clauses: sorted in a queue heuristically

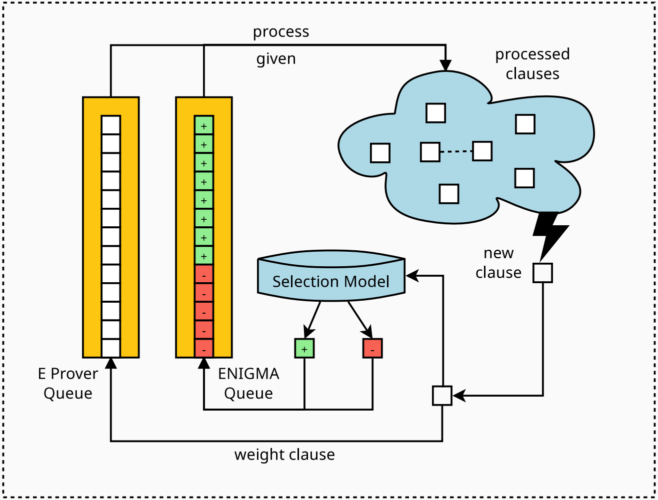
Saturation Theorem Proving Meets Machine Learning

- ATPs typically perform proof search by contradiction.
- Statements are translated to clauses (literal disjuction).
- Proof search is guided by the *given clause algorithm*.
- Inference rules allow to infer a new clause.
- The proof state consists of two subsets
    1. processed clauses: mutually inferred
    2. unprocessed clauses: sorted in a queue heuristically
- In the main loop, the best unprocessed clauses is picked and moved to processed, giving rise to new unprocessed clauses.

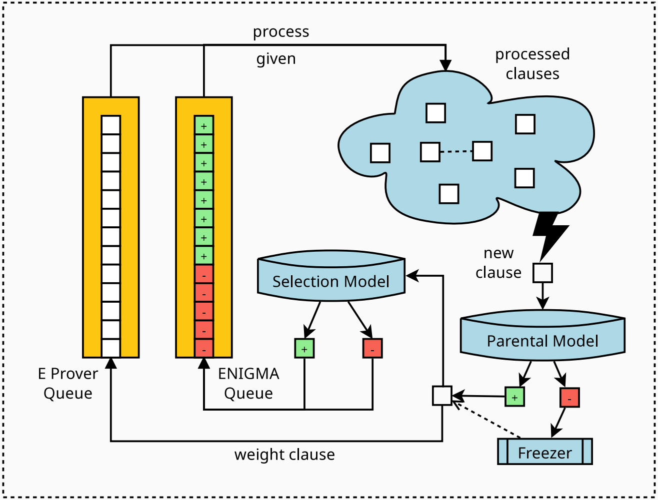
E Prover: Given Clause Loop



ENIGMA: Given Clause Guidance

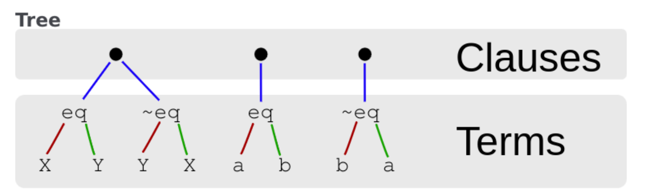


ENIGMA: Parental Generation Filter



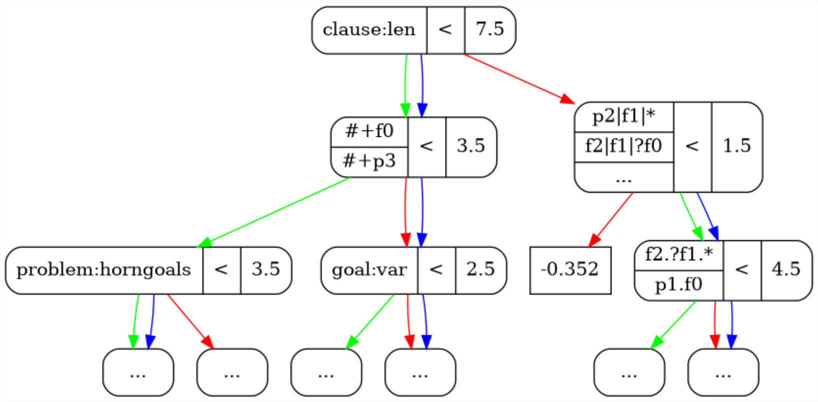
Gradient Boosted Decision Tree Classifiers and Features

Represent clauses as feature vectors and use LightGBM.

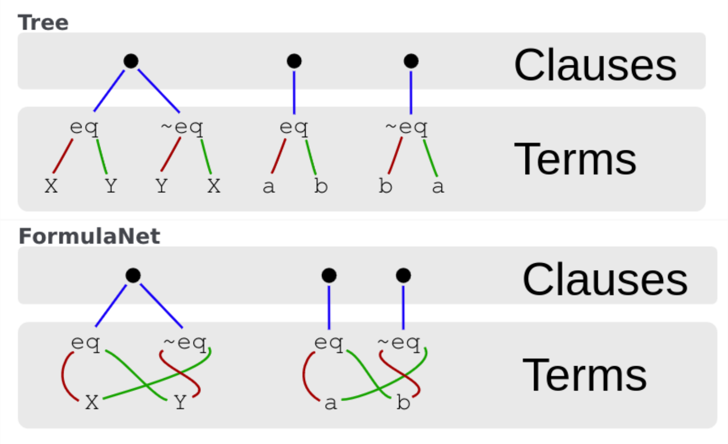


- **Vertical / Horizontal Features**: count symbol n-tuples
- **Conjecture Features**: embed conjecture features
- **Feature Hashing**: hash feature unique string identifiers
- **Parent Features**: embed features of parent clauses
- **Anonymization**: forget names except arity ($plus \Rightarrow f2$)

Gradient Boosted Decision Tree Classifiers and Features

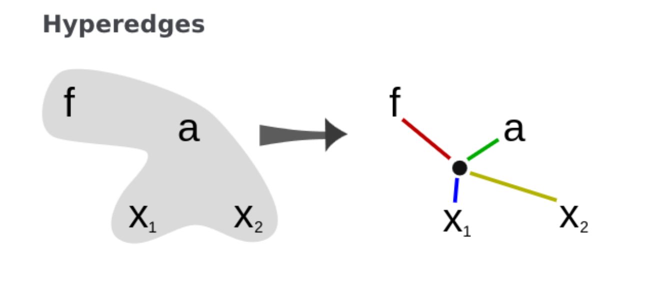


Graph Neural Network (GNN) Classifiers



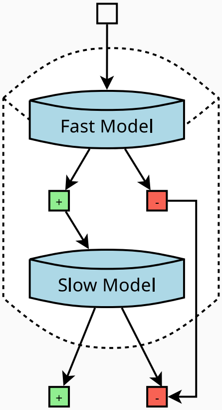
Graph Neural Network (GNN) Classifiers

- A set of clauses is represented by a hypergraph with three kinds of nodes for clauses, subterms/literals, and symbols.
- Graph hyperedges represent relationships among the objects.
- GNN layers perform message passing across the edges.



Multi-phase ENIGMA: Fast and Slow Models

- **Multi-phase ENIGMA:** Introduced to deal with computationally expensive (*slow*) ML models, like graph neural networks (GNNs).
- Fast model (GBDT) is used for preliminary clause filtering.
- Fast model over-approximates on positive classes.
- Only clauses classified with high confidence as negatives are rejected.



Additional Related Techniques

- **GPU Server Mode:** Reduce overhead of GPU loading.
    - GPU server with preloaded models on several GPUs.
    - Provers communicate with the server via a network socket.

Additional Related Techniques

- **GPU Server Mode:** Reduce overhead of GPU loading.
  - GPU server with preloaded models on several GPUs.
  - Provers communicate with the server via a network socket.
- **Leapfrogging:** Filter the evolving set of generated clauses.
  (1) Stop the prover after some time, filter clauses, and continue.
  (2) Split clauses into components, solve separately, and merge.

Additional Related Techniques

- **GPU Server Mode:** Reduce overhead of GPU loading.
  - GPU server with preloaded models on several GPUs.
  - Provers communicate with the server via a network socket.
- **Leapfrogging:** Filter the evolving set of generated clauses.
  (1) Stop the prover after some time, filter clauses, and continue.
  (2) Split clauses into components, solve separately, and merge.
- **BliStr/Tune:** Automated invention of ATP strategies.
  - interleaves targeted parameter search on problem clusters...
  - with a large-scale evaluation of the invented strategies

Additional Related Techniques

- **GPU Server Mode:** Reduce overhead of GPU loading.
  - GPU server with preloaded models on several GPUs.
  - Provers communicate with the server via a network socket.
- **Leapfrogging:** Filter the evolving set of generated clauses.
  1. Stop the prover after some time, filter clauses, and continue.
  2. Split clauses into components, solve separately, and merge.
- **BliStr/Tune:** Automated invention of ATP strategies.
  - interleaves targeted parameter search on problem clusters...
  - with a large-scale evaluation of the invented strategies
- **Deepire:** Machine learning in Vampire
  - uses recursive neural networks
  - classifies generated clauses based on their derivation

Robust Portfolio Construction

- Machine learning methods typically overfit on the trains.
- We use train/devel/holdout split to prevent overfitting.
- Portfolio construction: Construct a sequence of strategies to be run sequentially, each with a given time limit.
- For compatibility with Mizar40: portfolio runtime = 420s
- **Random split greedy cover construction:**
  (1) Split the devel set D into halves: $D = D_1 \cup D_2$.
  (2) Construct greedy cover C_1 on D_1 and evaluate C_1 on D_2.
  (3) Goto (1): repeat this process n times ($n = 1000$)
  (4) Select the less overfitting portfolio C_1 from (2).

Outline

Premise Selection in MML

- When proving a new conjecture, only a small fraction of the large ITP library is typically needed.
- Too many redundant significantly reduces ATP performance.
- Several premise selection approaches:
- **Bushy Premises ($\mathcal{B}$)**: Estimate premises based on human-written premises in the Mizar proof.
- Additional *data-driven* and *machine-learning* approaches.

Multilabel Premise Selection

Simple but fast ML methods used already in Mizar40 evaluation:

Multilabel Premise Selection

Simple but fast ML methods used already in Mizar40 evaluation:

- **k-NN ($\mathcal{K}$):** The k-nearest neighbours algorithm

  Choose k facts closest to the conjecture in the feature space and select their dependencies.

Multilabel Premise Selection

Simple but fast ML methods used already in Mizar40 evaluation:

- **k-NN ($\mathcal{K}$):** The k-nearest neighbours algorithm

  Choose k facts closest to the conjecture in the feature space and select their dependencies.

- **Naive Bayes ($\mathcal{N}$):** The sparse Naive Bayes algorithm

  Estimates the relevance of a fact F by the conditional probability of F being useful (statistically) under the condition of the features being present in the conjecture.

Premise Selection as Binary Classification

The models for clause selection can be used for premise selection.

- **Gradient Boosted Decision Trees ($\mathcal{L}$):**
  - faster to train than the deep learning methods
  - perform well with unbalanced training sets
  - handle well sparse features
  - scores the pairwise relevance of the conjecture and a premise

Premise Selection as Binary Classification

The models for clause selection can be used for premise selection.

- **Gradient Boosted Decision Trees ($\mathcal{L}$):**
  - faster to train than the deep learning methods
  - perform well with unbalanced training sets
  - handle well sparse features
  - scores the pairwise relevance of the conjecture and a premise
- **Graph Neural Networks ($\mathcal{G}$):**
  - the same GNN architecture as for clause selection
  - scores premises relevance to the conjecture in a single query
  - combined with a simpler k-NN to preselect 512 facts

Ensemble Methods for Premise Selection

- Combine several premise selection methods into one.
- Prediction scores of different methods might be incomparable.

Ensemble Methods for Premise Selection

- Combine several premise selection methods into one.
- Prediction scores of different methods might be incomparable.
- Solution: Consider rankings produced by different methods.
- Combine several rankings into one (mean, minimum, ...).

Ensemble Methods for Premise Selection

- Combine several premise selection methods into one.
- Prediction scores of different methods might be incomparable.
- Solution: Consider rankings produced by different methods.
- Combine several rankings into one (mean, minimum, ...).
- Method weights to set preferred ensemble methods:

$$\mathcal{E}^{\mathcal{K},\mathcal{N},\mathcal{G}}_{0.25,0.25,0.5}$$

Subproblem Based Premise Minimization

- Mizar proof consists of a series of natural deduction steps (subproblems) that have to be justified.
- ATP proofs of subproblems can be used to prune the (overapproximated) set of human-written premises in Mizar.

We consider the following approaches ($\mathcal{M}$):

(1) Only premises of ATP-proved subproblems.
(ignoring unproved)

(2) Add to (1) all explicit Mizar premises of the theorem.

(3) Add to (2) also the (semi-explicit) definitional expansions detected by the natural deduction module.

Outline

Bushy Experiments and Timeline

solved	[%]	*date*	*premises*	*methods/notes*
$\sim$ 32k	56.00	2012	$\mathcal{B}$	Mizar40
$\sim$ 38k	65.65	Jun 2020	$\mathcal{B}$	ENIGMA @ IJCAR'20
40 268	69.57	Oct 2020	$\mathcal{B}$	ENIGMA after IJCAR
40 994	70.83	Nov 12	$\mathcal{M}$	subproblem minimization
41 169	71.13	Nov 12	$\mathcal{M}$	Vampire with 300 s
42 206	72.92	Dec 7	many	E/ENIGMA/Vampire
42 471	73.38	Jan 2021	$\mathcal{G}$, $\mathcal{E}$	BliStr/Tune/E strategies
42 826	73.99	May 14	$\mathcal{G}$,$\mathcal{L}$,$\mathcal{K}$	Deepire @ FroCoS'21
43 599	75.33	Aug 26	$\mathcal{M}$,$\mathcal{B}$,$\mathcal{L}$	2,3-phase ENIGMA
43 717	75.53	Sep 2	$\mathcal{M}$	mainly Vampire/Deepire

Hammer Mode: ENIGMA on The Premise Selection Data

ENIGMA GBDT alone can reach 55% in several loops:

loop	*trains*	*devel* (union)	*devel cover* (in 420s)	[%]
init	20 604	1215	-	-
(1)	25 240	1601	1516	52.33
(2)	25 725	1669	1555	53.69
(3)	25 887	1679	1560	53.88
(4)	29 266	1716	1591	54.94
(5)	37 053	1735	1610	55.59

Hammer Mode: ENIGMA and Others

Other ATP methods are reasonably complementary to ENIGMA:

prover (420s)	cover	*pairs*	[%]
E 2.6 (auto-schedule)	1430	14	49.38
Vampire 4.0 (CASC)	1536	14	53.03
BliStr/Tune	1582	210	54.62
ENIGMA/GBDT	1610	42	55.59
ENIGMA/GNN	1670	84	57.66

Final Hammer Portfolio

- The final portfolio is constructed on the devel using the robust portfolio construction described earlier.
- Then evaluated on the holdout set.
- The final 420-second portfolio has 95 slices:
  - solves 1749 (60.4 %) of the devel problems
  - solves 1690 (58.36 %) of the holdout problems
- Compare with Mizar40:
  - **Bushy mode:** from 56.0% to 75.53%
  - **Hammer mode:** from 40.6% to 58.36%

Single Strategy Performance

1. Our strongest single AI/TP method alone now proves in 30 s 40 % of the lemmas in the hammering mode, i.e., reaching the same strength as the full 420 s portfolio in Mizar40.
2. Our strongest *single* AI/TP method now proves in 120 s 60 % of the toplevel lemmas in the human-premises (*bushy*) mode, i.e., outperforming the union of *all* methods developed in Mizar40 (56 %).
3. On *new* (MML v1382) 13 370 theorems (half of them with *new* symbols): this strongest method outperforms standard E by 58.2 %, while this is only 56.1 % on the Mizar40 version (where we trained). Likely thanks to the *anonymous* ML methods that learn only from the structure of the problems (unlike LLMs that struggle on new terminology).

Finally

Visit us online!

- Interesting ATP proofs of Mizar theorems

  `https://github.com/ai4reason/ATP_Proofs`

- VizAR: Mizar ATP Proof Gallery

  `https://ai.ciirc.cvut.cz/vizar/`

- Check extended version on arXiv for additonal discussions

  `https://arxiv.org/abs/2303.06686`

- and its "ML-guided deductive synthesis vs LLMs" section

  `https://ar5iv.labs.arxiv.org/html/2303.06686#A1.SS2`