Solving Hard Mizar Problems with Instantiation and Strategy Invention

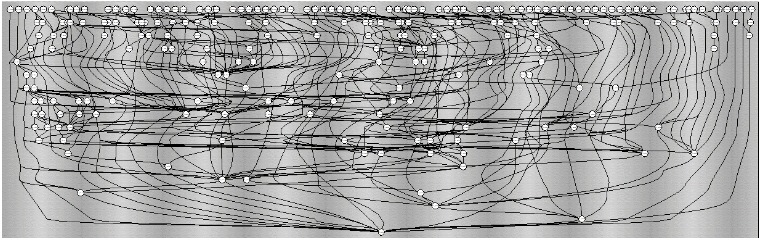
Jan Jakubův, Mikoláš Janota, Josef Urban

Czech Technical University in Prague

September 5th / AGI 2024

Background - MML and MPTP

- Mizar Mathematical Library (MML): Large library of formal mathematics developed since 1989
- 1465 math articles and 3.7M lines of human-readable proofs in 2024
- In 2003: **MPTP**: Mizar Problems for Theorem Proving
- export MML for automated theorem provers (ATPs)
- Used since for AITP research (MPTP20 talk: `https://t.ly/SFdPA`)
- 2006: the $100 MPTP Challenges (`https://t.ly/clXXe`)
- bushy (easier, smaller) vs chainy (large, *hammer*) MPTP problems



ATP timeline on MPTP problems

- 2010: Vampire solved 40% of bushy (easier) problems
- 2014: about 40% of chainy (hammer) problems solved by AI/TP methods (also done for Flyspeck)
- **2021**: about 60% of chainy solved with many AI/TP methods:
- E/ENIGMA and Vampire/Deepire (Mizar60 paper at ITP23)
- In total: 75.5% proved (union of bushy and chainy, higher times)
- See `https://github.com/ai4reason/ATP_Proofs` for about 200 interesting proofs found in those experiments
- Our goal here: Solve more of the remaining 14163 *hard* Mizar problems (and thus progress towards my 2014 AITP Challenges)

AITP Challenges/Bets from 2014

- 3 AITP bets for 10k EUR from my 2014 talk at Institut Henri Poincare (`tinyurl.com/yb55b3jv`)
- In 20 years, 80% of Mizar and Flyspeck toplevel theorems will be provable automatically (same hardware, same libraries as in 2014 - about 40% then)
- In 10 years: 60% (DONE already in 2021 - 3 years ahead of schedule)
- 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

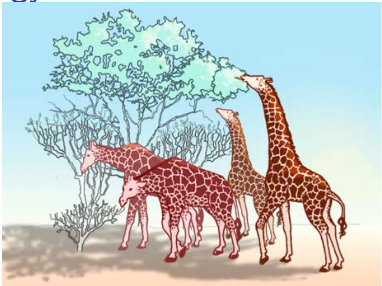
Our Main Results and Methods

- Solved 3,021 (21.3%) of remaining 14,163 hard Mizar problems
- Thus increased percentage of ATP-proved Mizar problems from 75.5% to 80.7%
- We used instantiation-based methods, particularly cvc5 SMT solver
- Note that we did not use any special decision procedures in cvc5
- We invented stronger cvc5 strategies using our Grackle system
- Further improved by different clausification and premise selection
- This has surprisingly high impact on instantiation-based methods

Overview of Instantiation-Based ATP/SMT Methods

- Herbrand (1930): a set of clauses is unsat iff finitely ground-unsat
- Gilmore's procedure (1960) - generate ground instances and check for ground unsat (decidable, inefficient in 1960)
- Efficient SAT/UNSAT: DPLL (1960/61), CDCL (1996, revolutionary)
- 2005: John Harrison: "People now say that problems are NP-easy"
- Since 2000s: renewed development of inst-based methods:
- iProver, Darwin, Equinox, SMTs like Z3, CVC, veriT, etc.
- Satallax (higher-order ATP), AVATAR (Vampire), etc.
- cvc5: SMT solver using instantiation for quantifiers
- Alternates between ground solver and instantiation module
- Generates lemmas by instantiating quantified formulas
- Uses various instantiation heuristics (e-matching, model-based, enumeration, etc.)
- Quite different from saturation-based ATPs; add ML guidance?

Automated Strategy Invention: BliStr and Grackle



- Dawkins: The Blind Watchmaker
- Grow diverse strategies by iterative local search and evolution
- ATP strategies are programs specified in rich DSLs - can be evolved
- The ATP strategies are like giraffes, the problems are their food
- The better the giraffe specializes for eating problems unsolvable by others, the more it gets fed and further evolved
- fast "inductive" training phase, followed (if successful) by a slower "hard thinking" phase, in which the newly trained strategies attempt to solve some more problems, making them into further training data

BliStr: Blind Strategymaker (2012)

- Used for automated invention of saturation-based ATP strategies
- The E strategy with longest specification in Jan 2012

```
G-E--_029_K18_F1_PI_AE_SU_R4_CS_SP_S0Y:

4 * ConjectureGeneralSymbolWeight(
      SimulateSOS,100,100,100,50,50,10,50,1.5,1.5,1),
3 * ConjectureGeneralSymbolWeight(
      PreferNonGoals,200,100,200,50,50,1,100,1.5,1.5,1),
1 * Clauseweight(PreferProcessed,1,1,1),
1 * FIFOWeight(PreferProcessed)
```

The Longest E Strategy After BliStr Evolution

Evolutionarily designed Franken-strategy (29 heuristics combined):

```
6 * ConjectureGeneralSymbolWeight(PreferNonGoals,100,100,100,50,50,1000,100,1.5,1.5
8 * ConjectureGeneralSymbolWeight(PreferNonGoals,200,100,200,50,50,1,100,1.5,1.5,1)
8 * ConjectureGeneralSymbolWeight(SimulateSOS,100,100,100,50,50,50,50,1.5,1.5,1)
4 * ConjectureRelativeSymbolWeight(ConstPrio,0.1, 100, 100, 100, 100, 1.5, 1.5, 1.5
10 * ConjectureRelativeSymbolWeight(PreferNonGoals,0.5, 100, 100, 100, 100, 1.5, 1.
2 * ConjectureRelativeSymbolWeight(SimulateSOS,0.5, 100, 100, 100, 100, 1.5, 1.5, 1
10 * ConjectureSymbolWeight(ConstPrio,10,10,5,5,5,1.5,1.5,1.5)
1 * Clauseweight(ByCreationDate,2,1,0.8)
1 * Clauseweight(ConstPrio,3,1,1)
6 * Clauseweight(ConstPrio,1,1,1)
2 * Clauseweight(PreferProcessed,1,1,1)
6 * FIFOWeight(ByNegLitDist)
1 * FIFOWeight(ConstPrio)
2 * FIFOWeight(SimulateSOS)
8 * OrientLMaxWeight(ConstPrio,2,1,2,1,1)
2 * PNRefinedweight(PreferGoals,1,1,1,2,2,2,0.5)
10 * RelevanceLevelWeight(ConstPrio,2,2,0,2,100,100,100,100,1.5,1.5,1)
8 * RelevanceLevelWeight2(PreferNonGoals,0,2,1,2,100,100,100,400,1.5,1.5,1)
2 * RelevanceLevelWeight2(PreferGoals,1,2,1,2,100,100,100,400,1.5,1.5,1)
6 * RelevanceLevelWeight2(SimulateSOS,0,2,1,2,100,100,100,400,1.5,1.5,1)
8 * RelevanceLevelWeight2(SimulateSOS,1,2,0,2,100,100,100,400,1.5,1.5,1)
5 * rweight21_g
3 * Refinedweight(PreferNonGoals,1,1,2,1.5,1.5)
1 * Refinedweight(PreferNonGoals,2,1,2,2,2)
2 * Refinedweight(PreferNonGoals,2,1,2,3,0.8)
```

Grackle (2022, CICM)

- Successor/generalization of BliStr
- Grackles: birds that evolved different bill sizes for different food
- Uses existing algorithm configuration frameworks
  - ParamILS: Iterative Local Search (Hutter et al.)
  - SMAC3: Bayesian Optimization (Lindauer et al.)

  to improve a strategy on a given set of problems
- Grackle input:
  - initial set of strategies
  - input problems
  - strategy space parametrization: parameters and their values
  - solver wrapper
- Grackle output:
  - portfolio of strategies complementary on input problems

Grackle: Invent Portfolio of Strategies

Repeat the following:

1. Evaluate all strategies on all problems P
2. Select one strategy S to be improved
3. Specialize strategy S for the problems where it performs best
4. Go to 1

Terminate when:

- all strategies has been improved, or . . .
- time limit is reached.

cvc5 Strategy Space

- Defined by cvc5's command line options and values
- cvc5 distinguishes regular and expert (experimental) options
- Regular parametrization: 98 parameters, $\sim 10^{35}$ strategies
- Full parametrization: 168 parameters, $\sim 10^{58}$ strategies
- We focused on options relevant to uninterpreted functions with quantifiers

Dataset

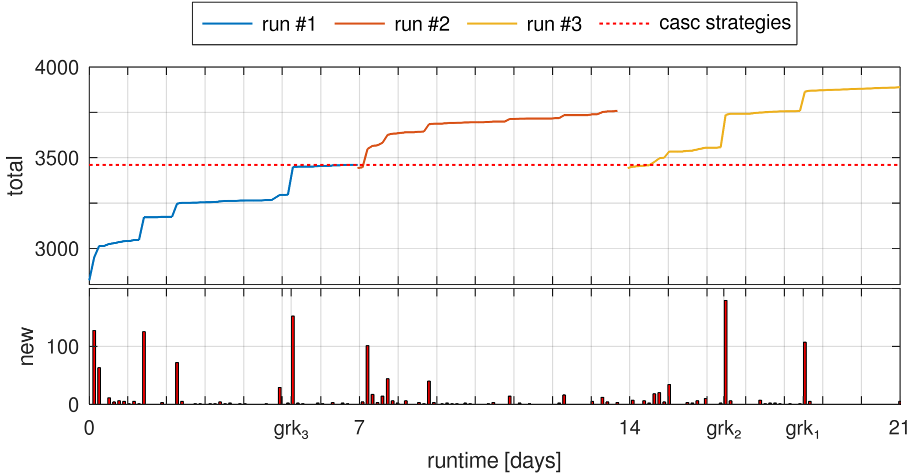
- 14,163 previously ATP-unproved Mizar *bushy* problems
- Extended with 4,283 hard problems proved only in latest ATP experiments
- This is done to give Grackle a bit easier problems to start inventing on
- We also used heuristically premise-minimized versions
- Total of 16,861 hard problems for doing cvc5 strategy development

Grackle Runs

- Three 7-day Grackle runs
- Run #1: regular space, starts with 2 CASC strategies
- Run #2: regular space, starts with 6 best strategies from Run #1
- Run #3: full strategy space, starts with the same as Run #2
- 30 second time limit per problem, 30 minutes per strategy invention
- Run #1: a proof of concept run starting with a weaker portfolio, 345 new probs
- Run #2: more serious, 485 new probs
- Run #3: measure the effect of expert options, 629 new probs

Progress of Three Grackle Runs

Progress in time of problems cumulatively solved by each Grackle run:



Grackle Strategy Invention Results

- 143 new strategies invented
- Best single strategy: 2,796 problems (11.5% improvement)
- Best 16 strategies: 4,039 problems (16.7% improvement)
- Total solved: 4,113 problems

Higher Time Limits

- Evaluated best strategies with 600 second time limit
- Best Grackle strategy: 3,496 problems
- Best CASC strategy: 3,059 problems
- 14.3% improvement for single best strategy
- cvc5 (single strategy grk_1) solves almost 50% more problems when the time limit is increased from 60 to 600 seconds.
- E Prover (auto mode / single strategy) solves only 10% more with the same time limit increase.

Reformulation Experiments

- External clausification using E prover
  - Two variants: default (cnf1) and aggressive definition introduction (cnf2)
  - cnf2: Halved average number of literals, 60% symbols
  - Added 369 newly solved problems
- Tested different premise selection methods:
  - Bushy (original premises)
  - GNN (Graph Neural Networks)
  - LightGBM (Gradient Boosting Decision Trees)
- Highly complementary to other methods
- Added 1,065 newly solved problems

Top 10 Strategies from Greedy Cover

version	*strategy*	*addon*		*total*	*alone*	*new*
min_{fof}	grk_1	+3496	-	3496	3496	1243
min_{cnf1}	grk_2	+738	+21.11%	4234	3231	1192
gnn	grk_1	+535	+12.64%	4769	1215	432
bushy	grk_1	+311	+6.52%	5080	1441	553
min_{fof}	grk_3	+298	+5.87%	5378	3220	1146
lgbm	grk_1	+233	+4.33%	5611	1512	541
min_{cnf1}	grk_3	+161	+2.87%	5772	3223	1092
min_{cnf1}	casc_{10}	+112	+1.94%	5884	3125	999
min_{fof}	grk_2	+90	+1.53%	5974	3146	1131
min_{cnf2}	grk_2	+62	+1.04%	6036	2949	1045

- *addon* = addition to the portfolio; *total* = partial portfolio performance
- *alone* = standalone strategy performance (600 seconds time limit)
- *new* = hard Mizar problems newly solved by each strategy
- Grackle-invented strategies dominate the greedy cover
- The results also transfer to a new (unseen) version of MML

Analysis of Invented Strategies

Best CASC strategies:

$casc_7$	`full-saturate-quant`	`multi-trigger-priority`	`multi-trigger-when-single`
$casc_{10}$	`full-saturate-quant`	`enum-inst-interleave`	`decision=internal`
$casc_{14}$	`full-saturate-quant`	`cbqi-vo-exp`	

Best Grackle strategies:

grk_1	`full-saturate-quant cbqi-vo-exp relational-triggers cond-var-split-quant=agg`
grk_2	`full-saturate-quant cbqi-vo-exp relevant-triggers multi-trigger-priority ieval=off no-static-learning miniscope-quant=off`
grk_3	`full-saturate-quant multi-trigger-priority multi-trigger-when-single term-db-mode=relevant`

- Focus on changing behavior of quantifier instantiation module
- Best strategies combine enumerative instantiations with appropriate trigger selection for e-matching
- grk_1 and grk_2 extend $casc_{14}$; grk_3 extends $casc_7$
- repo with the invented strategies and problems solved: `https://github.com/ai4reason/cvc5_grackle_mizar`

Interesting Solved Problems

- KURATO_1:6: Kuratowski's closure-complement problem
  - 131 lines in Mizar
  - Combination of equational reasoning and a large case split (14 cases) That likely makes it hard for the superposition-based systems
  - SMT-style congruence closure likely useful when a more complex term equal to a less complex term
- ASYMPT_1:18: Big O relation for modulo functions
  - functions $f(n) = n \bmod 2$ and $g(n) = n + 1 \bmod 2$ are not in the Big O relation (in any direction).
  - 122 lines in Mizar
  - Only provable with a single Grackle-invented strategy grk_2 and external clausification, taking 62 s.
  - case splits related to the mod 2 values; triggers seems to play a big role
- ROBBINS4:3: Equivalent condition for ortholattices
  - 145 lines in Mizar
  - a lot of equational reasoning (should be good for E/Vampire!)
  - possibly large multi-literal clauses make this hard for saturation systems

Interesting Solved Problems

```
definition let T be non empty TopSpace; let A be Subset of T;
func Kurat14Set A -> Subset-Family of T equals
{ A, A-, A-`, A-`-, A-`-`, A-`-`-, A-`-`-` } \/
{ A`, A`-, A`-`, A`-`-, A`-`-`, A`-`-`-, A`-`-`-` };
end;
theorem :: KURATO_1:6:
for T being non empty TopSpace
for A, Q being Subset of T st Q in Kurat14Set A holds
Q` in Kurat14Set A & Q- in Kurat14Set A;

theorem :: ASYMPT_1:18
 for f,g being Real_Sequence st
   (for n holds f.n = n mod 2) & (for n holds g.n = n+1 mod 2)
  holds ex s,s1 being eventually-nonnegative Real_Sequence
  st s = f & s1 = g & not s in Big_Oh(s1) & not s1 in Big_Oh(s)

theorem :: ROBBINS4:3
for L being non empty OrthoLattStr holds L is Ortholattice iff
  (for a, b, c being Element of L holds
     (a "\/" b) "\/" c = (c` "/\" b`)` "\/" a)
& (for a, b being Element of L holds a = a "/\" (a "\/" b))
& for a, b being Element of L holds a = a "\/" (b "/\" b`)
```

Conclusions

- Significant progress on hard Mizar problems
- Instantiation-based methods today surprisingly good
- Strategy invention (Grackle) very useful for cvc5
- High impact of problem reformulation: different clausifications, premise selection
- Interesting competition (also within our Prague group) between saturation-based (Vampire/Deepire, E/ENIGMA) and instantiation-based (cvc5, iProver, Satallax) ATPs

Future Work

- Apply strategy invention to other problem sets (e.g. TPTP, Isabelle)
- Further explore problem reformulation techniques (rewarding here)
- More learning for guiding instantiation:
  - neural (GNN - LPAR'24)
  - fast non-neural (ECAI'24)
  - choosing formulas, variables, instances ...
  - end-to-end ML-style guessing of instances?