ProofWatch++: Watchlist Guidance for E prover (and some Machine Learning)

Zarathustra Goertzel Jan Jakubův Josef Urban
Stephan Schulz

Czech Technical University in Prague

DHBW Stuttgart

ITP'18 and BUMERANG'18, 24th September 2018

Learning From Mizar Proofs

We work with

- Mizar Mathematical Library

```
theorem :: WAYBEL_1:85
  for H being non empty lower-bounded RelStr st H is Heyting holds
  for a, b being Element of H holds 'not' (a "/\" b) >= ('not' a) "\/" ('not' b)
```

```
theorem Th36: :: YELLOW_5:36
  for L being non empty Boolean RelStr
  for a, b being Element of L holds
  ( 'not' (a "\/" b) = ('not' a) "/\" ('not' b) & 'not' (a "/\" b) = ('not' a) "\/" ('not' b) )
```

Learning From Mizar Proofs: WAYBEL_1 85

```
theorem :: WAYBEL_1:85
  for H being non empty lower-bounded RelStr st H is Heyting holds
  for a, b being Element of H holds 'not' (a "/\" b) >= ('not' a) "\/" ('not' b)
```

```
  let H be non empty lower-bounded RelStr ; :: thesis:
  assume A1: H is Heyting ; :: thesis:
  then A2: Bottom H <= Bottom H by ORDERS_2:1;
  let a, b be Element of H; :: thesis:
  A3: 'not' a <= 'not' a by A1, ORDERS_2:1;
  A4: 'not' b <= 'not' b by A1, ORDERS_2:1;
  (a "/\" b) "/\" (('not' a) "\/" ('not' b)) = ((a "/\" b) "/\" ('not' a)) "\/" ((a
  "/\" b) "/\" ('not' b)) by A1, Def3
  .= ((b "/\" a) "/\" ('not' a)) "\/" ((a "/\" b) "/\" ('not' b)) by A1, LATTICE3:15
  .= (b "/\" (a "/\" ('not' a))) "\/" ((a "/\" b) "/\" ('not' b)) by A1, LATTICE3:16
  .= (b "/\" (a "/\" ('not' a))) "\/" (a "/\" (b "/\" ('not' b))) by A1, LATTICE3:16
  .= (b "/\" (Bottom H)) "\/" (a "/\" (b "/\" ('not' b))) by A1, A3, Th82
  .= (b "/\" (Bottom H)) "\/" (a "/\" (Bottom H)) by A1, A4, Th82
  .= ((Bottom H) "/\" b) "\/" (a "/\" (Bottom H)) by A1, LATTICE3:15
  .= ((Bottom H) "/\" b) "\/" ((Bottom H) "/\" a) by A1, LATTICE3:15
  .= (Bottom H) "\/" ((Bottom H) "/\" a) by A1, Th3
  .= (Bottom H) "\/" (Bottom H) by A1, Th3
  .= Bottom H by A1, A2, YELLOW_0:24 ;
  hence 'not' (a "/\" b) >= ('not' a) "\/" ('not' b) by A1, Th82; :: thesis:
```

Learning From Mizar Proofs: YELLOW_5 36

```
let L be non empty Boolean RelStr ; :: thesis:
let a, b be Element of L; :: thesis:
A1: (a "\/" b) "/\" (('not' a) "/\" ('not' b)) = ((a "\/" b) "/\" ('not' a)) "/\"
('not' b) by LATTICE3:16
.= ((('not' a) "/\" a) "\/" (('not' a) "/\" b)) "/\" ('not' b) by WAYBEL_1:def 3
.= ((Bottom L) "\/" (('not' a) "/\" b)) "/\" ('not' b) by Th34
.= (('not' a) "/\" b) "/\" ('not' b) by WAYBEL_1:3
.= ('not' a) "/\" (b "/\" ('not' b)) by LATTICE3:16
.= ('not' a) "/\" (Bottom L) by Th34
.= Bottom L by WAYBEL_1:3 ;
(a "\/" b) "\/" (('not' a) "/\" ('not' b)) = a "\/" (b "\/" (('not' a) "/\"
('not' b))) by LATTICE3:14
.= a "\/" ((b "\/" ('not' a)) "/\" (b "\/" ('not' b))) by Th17
.= a "\/" ((b "\/" ('not' a)) "/\" (Top L)) by Th34
.= a "\/" (b "\/" ('not' a)) by WAYBEL_1:4
.= (a "\/" ('not' a)) "\/" b by LATTICE3:14
.= (Top L) "\/" b by Th34
.= Top L by WAYBEL_1:4 ;
then ('not' a) "/\" ('not' b) is_a_complement_of a "\/" b by A1, WAYBEL_1:def 23;
hence 'not' (a "\/" b) = ('not' a) "/\" ('not' b) by Th11; :: thesis:
A2: (a "/\" b) "/\" (('not' a) "\/" ('not' b)) = a "/\" (b "/\" (('not' a) "\/"
('not' b))) by LATTICE3:16
.= a "/\" ((b "/\" ('not' a)) "\/" (b "/\" ('not' b))) by WAYBEL_1:def 3
.= a "/\" ((b "/\" ('not' a)) "\/" (Bottom L)) by Th34
.= a "/\" (b "/\" ('not' a)) by WAYBEL_1:3
.= (a "/\" ('not' a)) "/\" b by LATTICE3:16
.= (Bottom L) "/\" b by Th34
.= Bottom L by WAYBEL_1:3 ;
(a "/\" b) "\/" (('not' a) "\/" ('not' b)) = ((a "/\" b) "\/" ('not' a)) "\/"
('not' b) by LATTICE3:14
.= ((('not' a) "\/" a) "/\" (('not' a) "\/" b)) "\/" ('not' b) by Th17
.= ((Top L) "/\" (('not' a) "\/" b)) "\/" ('not' b) by Th34
.= (('not' a) "\/" b) "\/" ('not' b) by WAYBEL_1:4
.= ('not' a) "\/" (b "\/" ('not' b)) by LATTICE3:14
.= ('not' a) "\/" (Top L) by Th34
.= Top L by WAYBEL_1:4 ;
then ('not' a) "\/" ('not' b) is_a_complement_of a "/\" b by A2, WAYBEL_1:def 23;
hence 'not' (a "/\" b) = ('not' a) "\/" ('not' b) by Th11; :: thesis:
```

Learning From Mizar Proofs

We work with

- Mizar Mathematical Library
- E prover
  - can be a hammer for theorem proving (ITP)
  - uses first order formula (fof) form of Mizar:

```
% Mizar problem: t85_waybel_1,waybel_1,3140,5
fof(t85_waybel_1, conjecture,  (! [A] :  ( ( ~ (v2_struct_0(A))  &  (v1_yellow_0(A) & l1_orders_2
(A)) )  =>  (v9_waybel_1(A) =>  (! [B] :  (m1_subset_1(B, u1_struct_0(A)) =>  (! [C] :  (m1_subse
t_1(C, u1_struct_0(A)) => r1_orders_2(A, k10_lattice3(A, k7_waybel_1(A, B), k7_waybel_1(A, C)), k
7_waybel_1(A, k11_lattice3(A, B, C)))) ) ) ) ) ) ) ).
fof(antisymmetry_r2_hidden, axiom,  (! [A, B] :  (r2_hidden(A, B) =>  ~ (r2_hidden(B, A)) ) ) ).
fof(cc10_struct_0, axiom,  (! [A] :  (l1_struct_0(A) =>  ( ( ~ (v2_struct_0(A))  & v7_struct_0(A)
)  => v13_struct_0(A, 1)) ) ) ).
```

Learning From Mizar Proofs

We work with

- Mizar Mathematical Library
- E prover
  - can be a hammer for theorem proving (ITP)
  - uses first order formula (FOF) form of Mizar
  - learns from E proof clauses in conjunctive normal form:

```
cnf(c_0_63,plain,(v2_struct_0(X1)|v1_lattice3(X1)|~l1_orders_2(X1)|~v9_waybel_1(X1))).# trainpos
cnf(c_0_52,plain,(v2_struct_0(X1)|v5_orders_2(X1)|~l1_orders_2(X1)|~v9_waybel_1(X1))).# trainpos
cnf(c_0_136,plain,(v2_struct_0(X1)|r1_orders_2(X1,X3,k7_waybel_1(X1,X2))|k11_lattice3(X1,X2,X3)
        !=k3_yellow_0(X1)|~v1_yellow_0(X1)|~l1_orders_2(X1)|~v9_waybel_1(X1)
        |~m1_subset_1(X3,u1_struct_0(X1))|~m1_subset_1(X2,u1_struct_0(X1)))).# trainpos
cnf(c_0_139,negated_conjecture,(m1_subset_1(k11_lattice3(esk1_0,esk2_0,esk3_0),u1_struct_0(esk1_0)))).# trainpos
```

ProofWatch: Results Overview

To prove YELLOW_5 Theorem 36, E considers

- 6550 given clauses without learning from proofs
- 5218 given clauses with learning from proofs (such as WAYBEL_1 85)

ProofWatch: Results Overview

On our benchmarks of 57987 Mizar theorems, E proves

- 21670 (37.3%) conjectures without learning from proofs
- 23192 (39.9%) conjectures with learning from proofs

This is a 7% improvement (and there are obviously improvements to try).

ProofWatch: Results Overview

- ProofWatch provides a proof-state vector that characterizes the conjecture's mathematical context.

ProofWatch: Watchlist History

- Clauses from related (E) proofs are put on the **Watchlist**.
  - E feature implemented by Stephan Schulz.
  - E checks if each clause it considers matches a watchlist clause.

```
cnf(c_0_63,plain,(v2_struct_0(X1)|v1_lattice3(X1)|~l1_orders_2(X1)|~v9_waybel_1(X1))).# trainpos
cnf(c_0_52,plain,(v2_struct_0(X1)|v5_orders_2(X1)|~l1_orders_2(X1)|~v9_waybel_1(X1))).# trainpos
cnf(c_0_136,plain,(v2_struct_0(X1)|r1_orders_2(X1,X3,k7_waybel_1(X1,X2))|k11_lattice3(X1,X2,X3)
        !=k3_yellow_0(X1)|~v1_yellow_0(X1)|~l1_orders_2(X1)|~v9_waybel_1(X1)
        |~m1_subset_1(X3,u1_struct_0(X1))|~m1_subset_1(X2,u1_struct_0(X1)))).# trainpos
cnf(c_0_139,negated_conjecture,(m1_subset_1(k11_lattice3(esk1_0,esk2_0,esk3_0),u1_struct_0(esk1_0)))).# trainpos
```

ProofWatch: Watchlist History

- E proof clauses are put on the **Watchlist**.
  - E feature implemented by Stephan Schulz.
  - E checks if each clause it considers matches a watchlist clause.
- Hint lists (Veroff 96)
  - provide a similar feature in Prover9 and Otter
  - used for proving extensions of AIM (Abelian Inner Mapping) conjecture in loop theory by Bob Veroff
  - have obtained very long proofs (1000+ steps)

ProofWatch: Watchlist Usage

What goes on the watchlist?

- Related or similar proofs' clauses
- Proof sketches
- Conjectured lemmas/sub-goals
- Generally useful clauses or "mathematical tricks"

ProofWatch: Watchlist Usage

1. Matching clauses (subsumption) is **symbolic** AI.
2. Matching with proof completion is **statistical** AI.
3. The vectorial *proofstate* characterization can work with standard ML methods.
4. Watchlist provides logical interface between AI, human, or ITP context and the ATP:

ProofWatch: Experiments

We do the first benchmark of E's Watchlist. Specifically:

- how to include the watchlist in E strategies?
- how to build the watchlist?
- *matching* **watchlist clauses** modulo skolem names (e.g., $esk(A, B) \sqsubseteq eskfoobar(A, B)$)
- dynamic weighting of **hints** by proof completion (aka 'relevance')
- k-NN recommendation of prior proofs based on conjecture features

E Basics: Goal

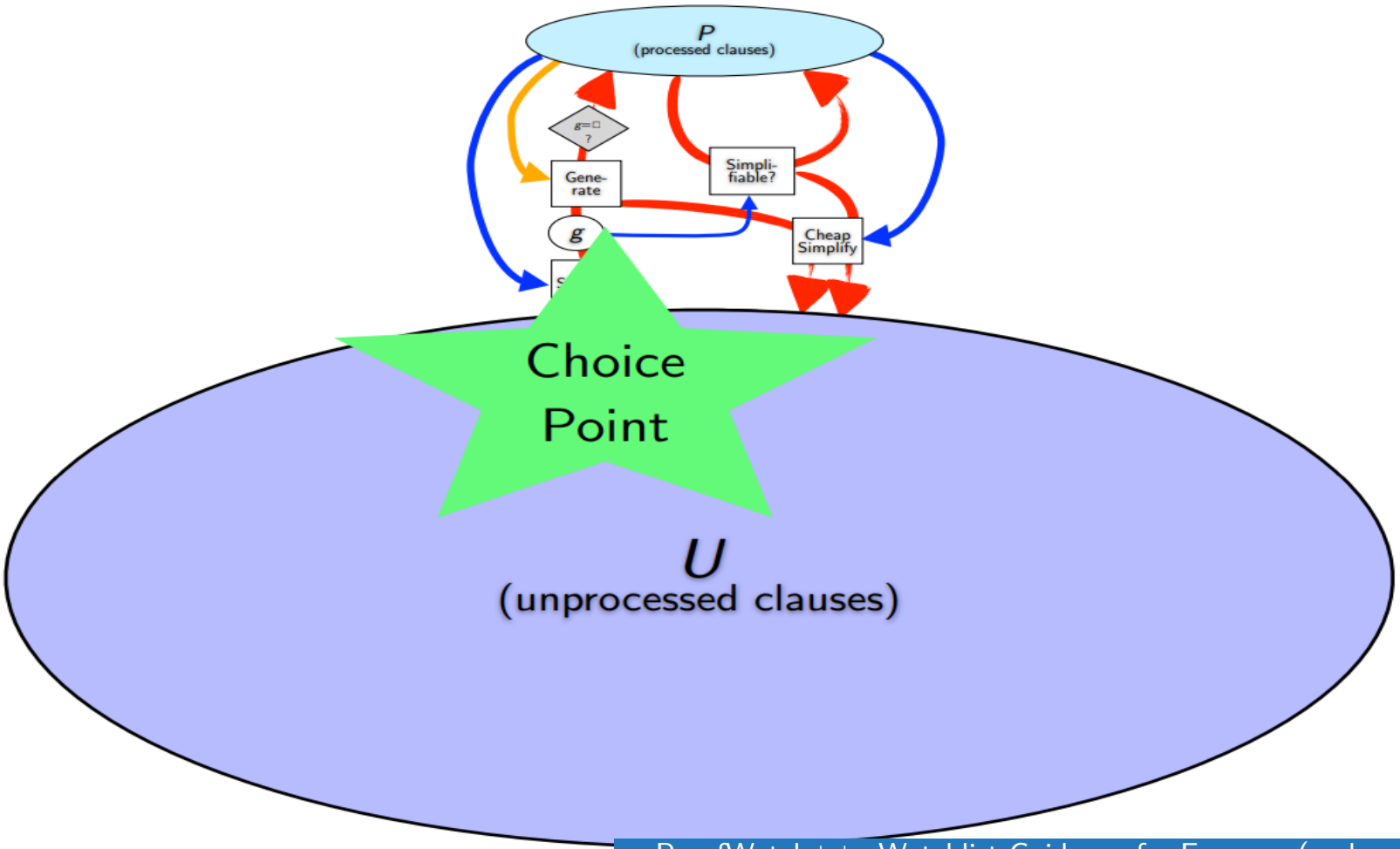
- E's a saturation-based refutational theorem prover (using superposition calculus)
- To prove P, E attempts to prove $\{\neg P \cup \text{Premises}\} \rightarrow \bot$

E Basics: Given Clause Loop

- E has two sets of clauses:
  - Processed clauses
  - Unprocessed clauses
- E's modus operandi is the **given clause loop**:

```
while (no proof found)
{
  select a given clauses
  apply inference rules to selected clauses
  process inferred clauses
}
```

Motivation


P
(processed clauses)
g=□ ?
Gene-rate
Simpli-fiable?
g
Cheap Simplify
Choice Point
U
(unprocessed clauses)

Basic Watchlist Mechanism: Given Clause Selection

- **watchlist**: list of clauses, hints.
- Check if each clause subsumes a **hint** (clause $\sqsubseteq$ hint)
- If yes, boost clause priority (and thus score), making selection more likely.

Basic Watchlist Mechanism: Clause Evaluation Functions

- E chooses given clauses by weighted round-robbin selection of clause evaluation functions (CEFs),
- Clause evaluation functions
  - partition clauses into queues based on *priority*
  - order queues based on clause weight

Basic Watchlist Mechanism: Clause Evaluation Functions

An example E strategy:

```
-tKBO -H(2*Clauseweight(PreferWatchlist,20,9999,4)
        ,4*FIFOWeight(PreferUnit))
```

Where

- **Priority** functions: PreferWatchlist, PreferUnit
- **Weight** functions: Clauseweight, FIFOWeight

Baseline 02: An Actual Evolved Strategy File

```
--definitional-cnf=24 --split-aggressive --simul-par
amod --forward-context-sr --destructive-er-aggressive
--destructive-er --prefer-initial-clauses -tKBO -winv
freqrank -c1 -Ginvfreq -F1 --delete-bad-limit=150000000
-WSelectMaxLComplexAvoidPosPred
-H(1*ConjectureTermPrefixWeight(DeferSOS,1,3,0.1,5,0,0.1,1
,1*ConjectureTermPrefixWeight(DeferSOS,1,3,0.5,100,0,0.2,0
,1*Refinedweight(PreferWatchlist,4,300,4,4,0.7)
,1*RelevanceLevelWeight2(PreferProcessed,0,1,2,1,1,1,200,20
,1*StaggeredWeight(DeferSOS,1)
,1*SymbolTypeweight(DeferSOS,18,7,-2,5,9999.9,2,1.5)
,2*Clauseweight(PreferWatchlist,20,9999,4)
,2*ConjectureSymbolWeight(DeferSOS,9999,20,50,-1,50,3,3,0.5
,2*StaggeredWeight(Defer
```

Dynamic Watchlist Mechanism

- Multiple watchlists: one for each loaded proof.
- Keep track of which proofs **hints** come from.
- Count what % of each proof has been covered (subsumed)
- Assign more priority for higher completion %.
  - –> closer to contradiction
  - –> closer to current proof
- Can represent proof state.
- The **PreferWatchlistRelevant** priority function.

Dynamic Watchlist Mechanism: Relevance Inheriting

Probably a good idea to keep following a good hint:

- *Inherit watchlist priority from parents*

Priority queues are absolute, so

- Decrease inheritance each generation with multiplicative decay.
- Drop to zero below a threshold

Dynamic Watchlist Mechanism: Relevance Calculation

For a clause C matching at least one watchlist W_i:

$$relevance_0(C) = \max_{W \in \{W_i : C \sqsubseteq W_i\}} \left(\frac{progress(W)}{|W|} \right)$$

Where $progress(W)$ is how many clauses in watchlist W have been subsumed.

Dynamic Watchlist Mechanism: Inheritance Calculation

With decay, for $\delta < 1$,

$$relevance_1(C) = relevance_0(C) + \delta * \operatorname*{avg}_{D \in parents(C)} \Big(relevance_1(D) \Big)$$

Aditionally, reset to 0 if $relevance_1(C) < \alpha$ and $\frac{relevance_1(C)}{length(C)} < \beta$ for *threshold* parameters $\alpha, \beta > 0$.

Baseline Strategies

- We use 57897 Mizar40 conjectures with premises pre-selected.
- We previously evolved 32 strategies with a coverage of 24702 proofs (in 5s each).
- We use a *top 5 greedy cover* of strategies:
  Strategies A-E.
  In 10s they together solve 21670 problems.
  In 5 s they together solve 21122 problems.

Using the Watchlist

E native option:

- Stephan Schulz's -**xUseWatchlistEvo** (*evo*)

Using the Watchlist

E native option:

- Stephan Schulz's **-xUseWatchlistEvo** (*evo*)

Some options (modifying baseline strategies):

- All **priority functions** -> **PreferWatchlist** (*pref*)
- Always PreferWatchlist, default to other priority function if not on watchilst (*uwl*).
- Match Skolems by name and arity only (on top of *pref*) (*ska*).
- Priority functions -> **PreferWatchlistRelevant** (*dyn*).
- Add relevance inheritance to *dyn* (*dyndec*).

Using the Watchlist

E native option:

- Stephan Schulz's -**xUseWatchlistEvo** (*evo*)

Some options (modifying baseline strategies):

- All **priority functions** -> **PreferWatchlist** (*pref*)
- Always PreferWatchlist, default to other priority function if not on watchilst (*uwl*).
- Match Skolems by name and arity only (on top of *pref*) (*ska*).
- Priority functions -> **PreferWatchlistRelevant** (*dyn*).
- Add relevance inheritance to *dyn* (*dyndec*).

And for comparison:

- Priority functions -> **ConstPrio** (*const*)

Creating the Watchlist

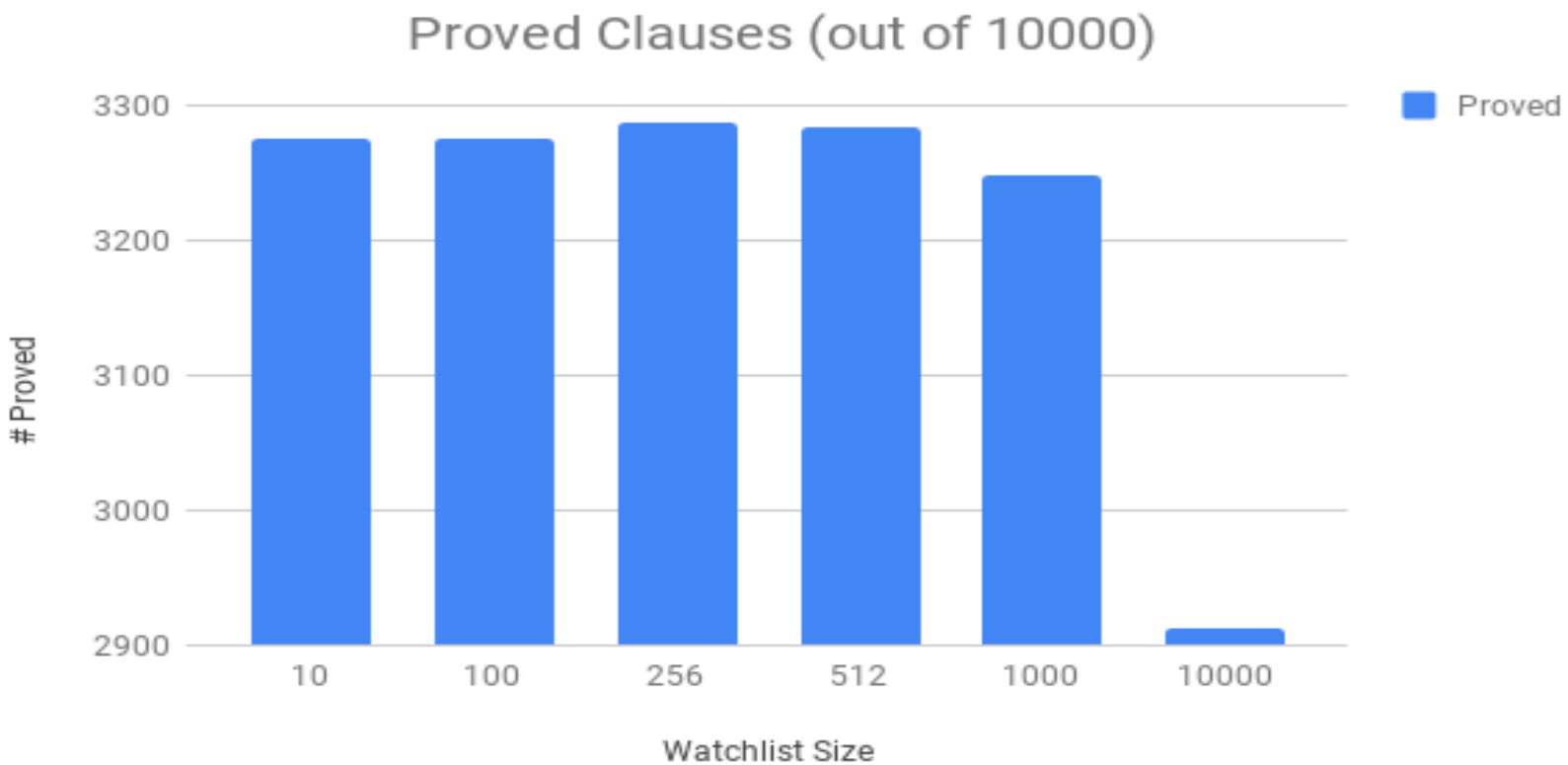
- Use most frequent clause
- Use proof clauses from the conjecture's Mizar article (size 1-4000)
- Use k-NN to suggest clauses
- Use k-NN to suggest proofs

Informal/Preliminary Testing

- Clauses from proofs by same strategy seem more useful.
- Not removing matched **hints** helps with small watchlists.
- Defaultweight(PreferWatchlist) sucks: **weight** functions are good.

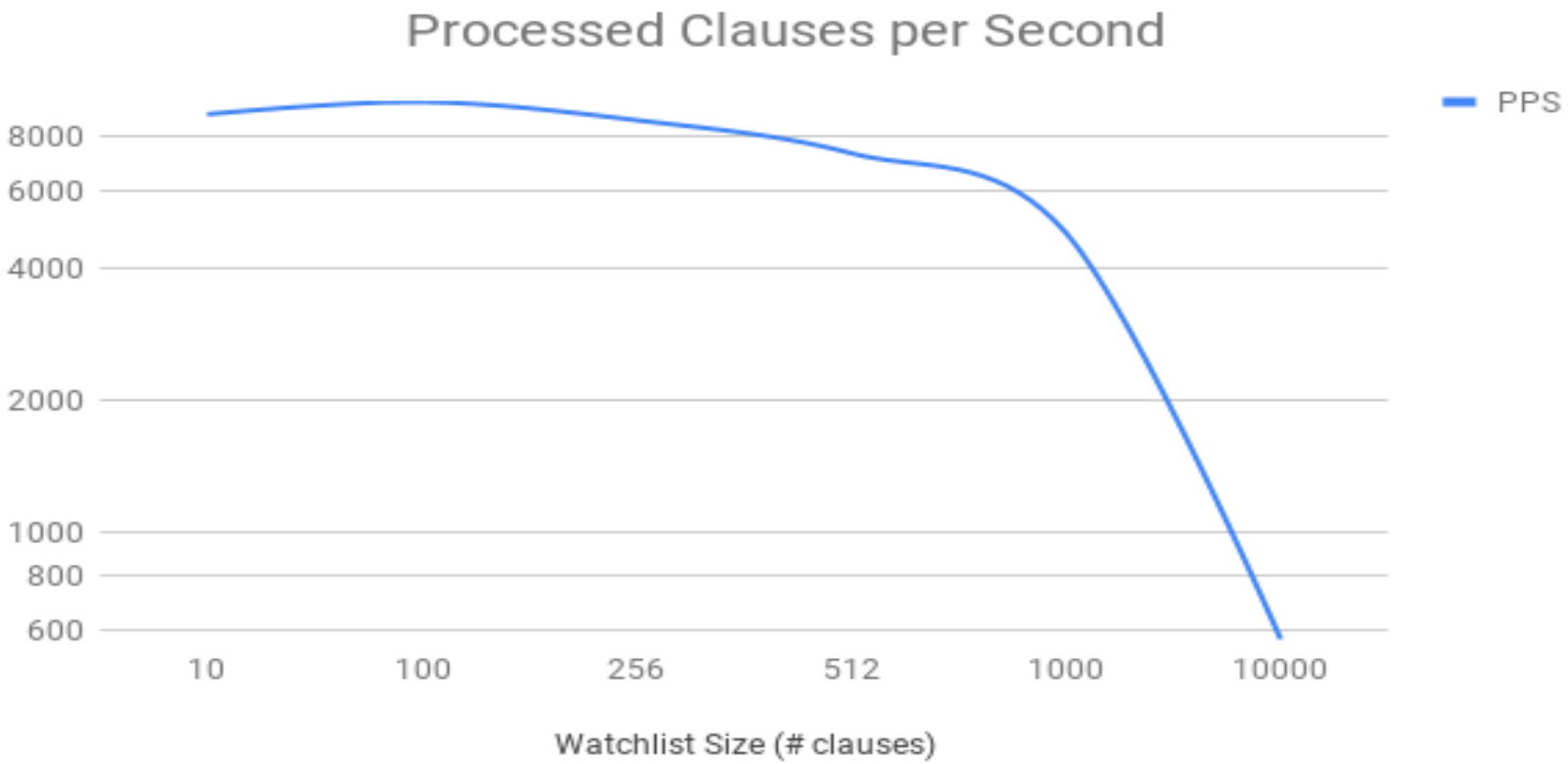
Small Most Frequent Clauses Test

Results of *prefA* run with **hints** taken from all articles.



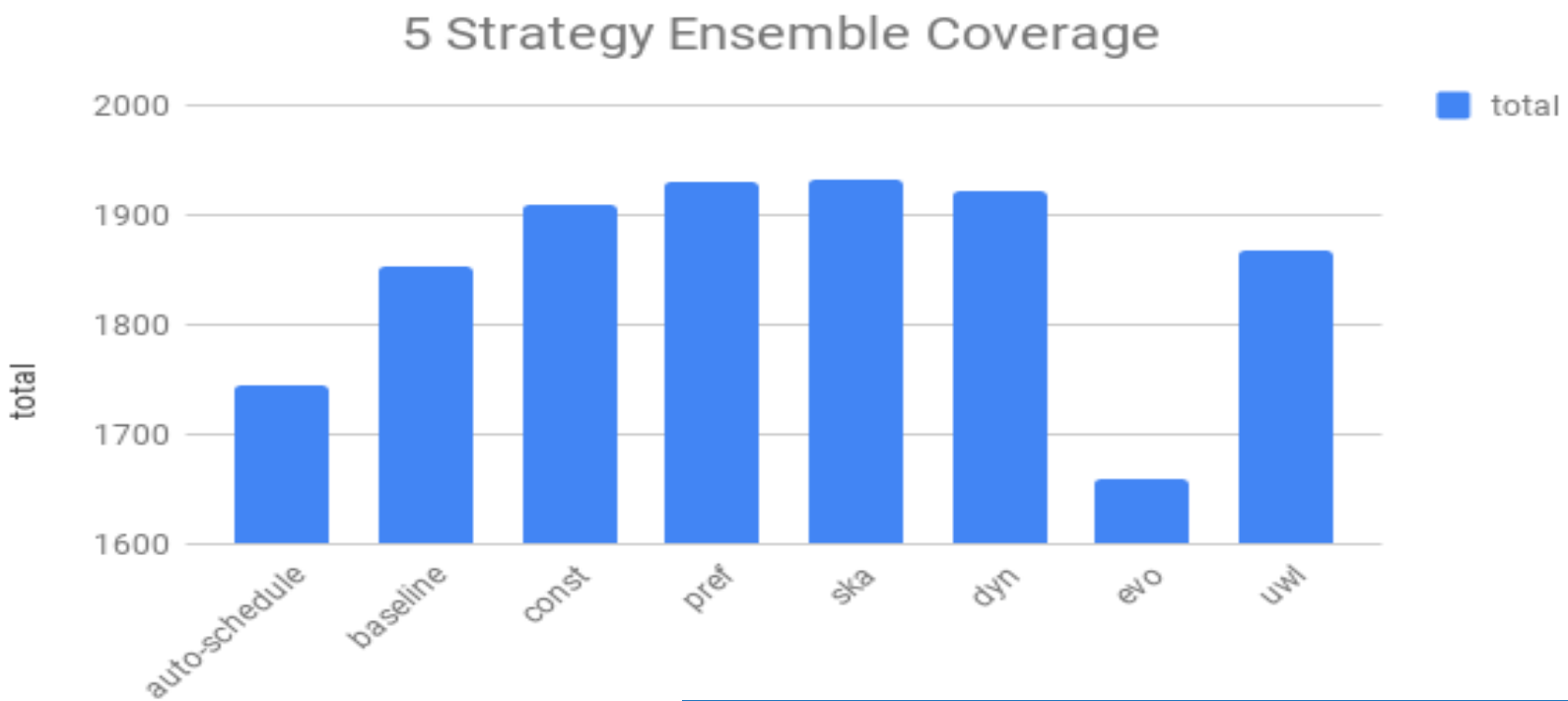
Small Most Frequent Clauses Test

Results of *prefA* run with **hints** taken from all articles.

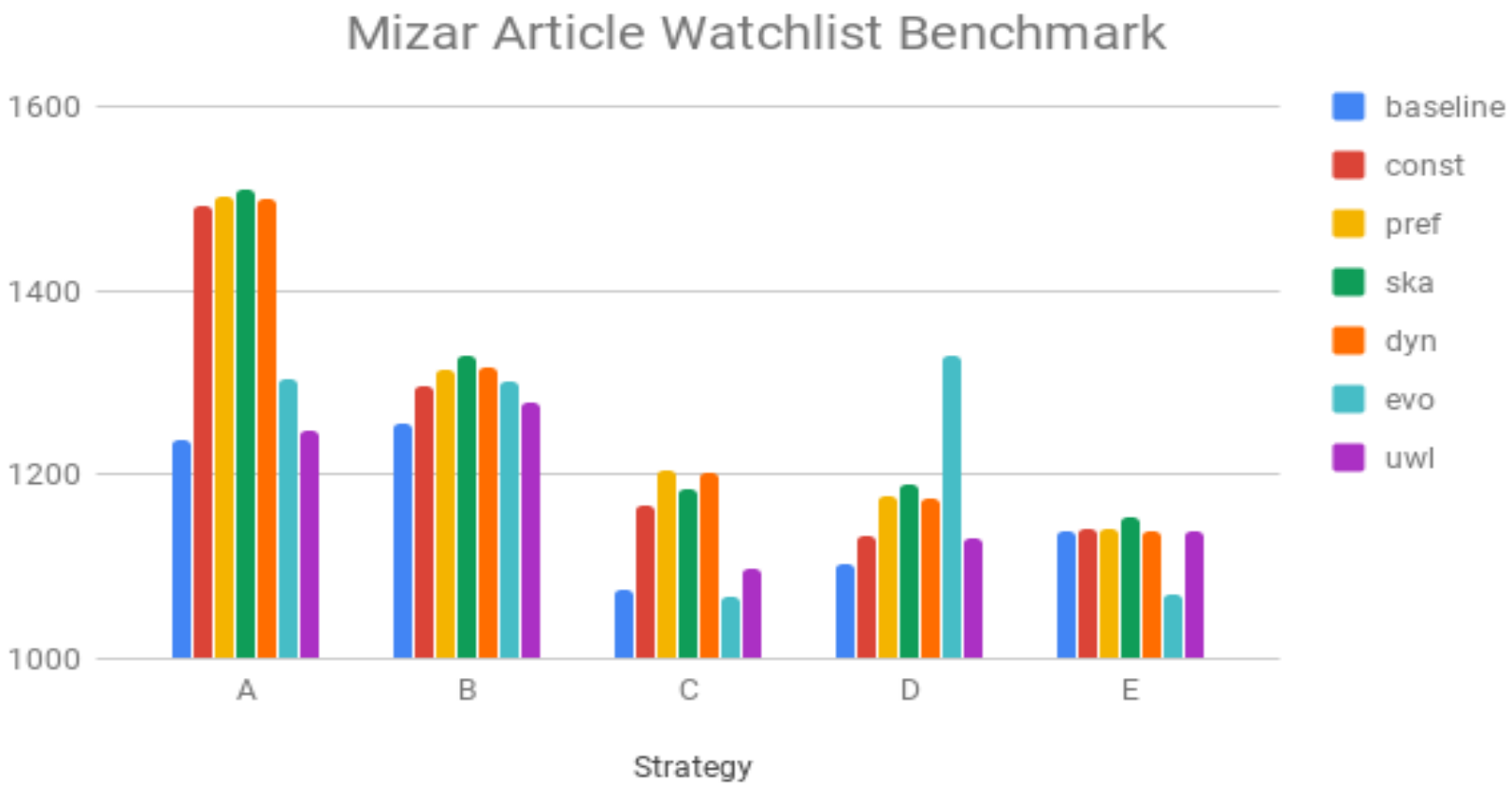


Mizar-article Watchlist Benchmark

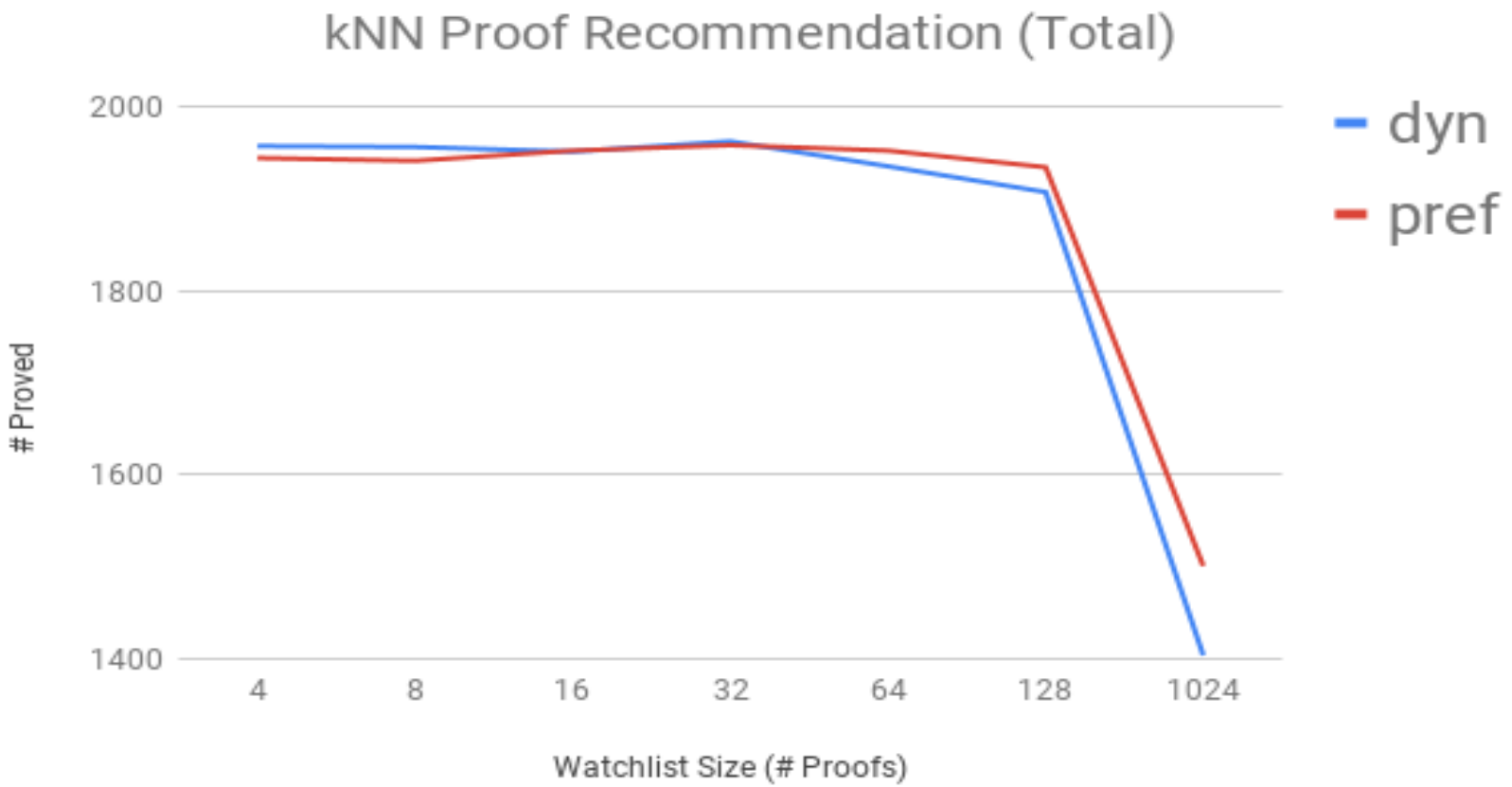
The strategies are run for 10s each on a random sample of 5000 conjectures.



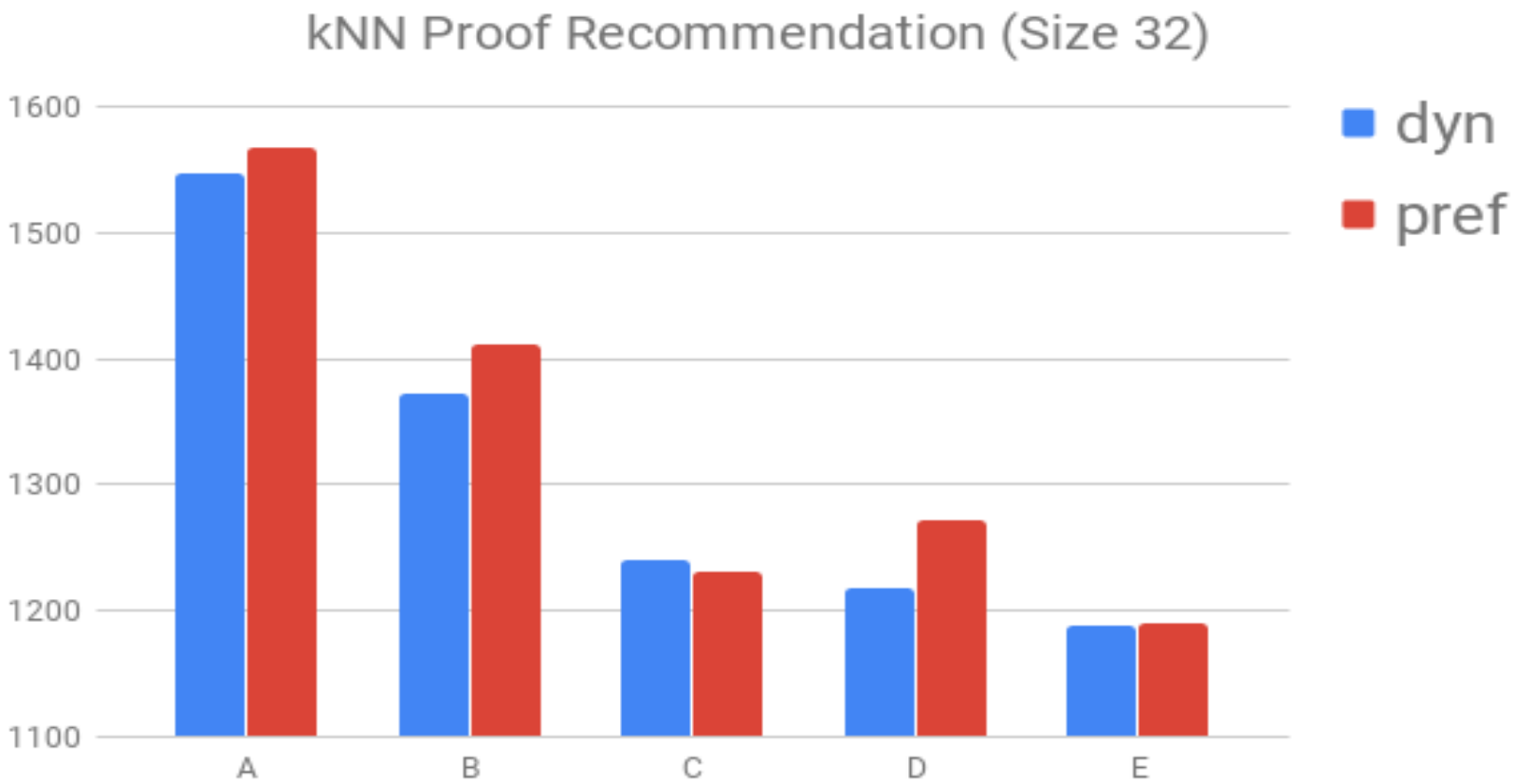
Mizar-article Watchlist Benchmark



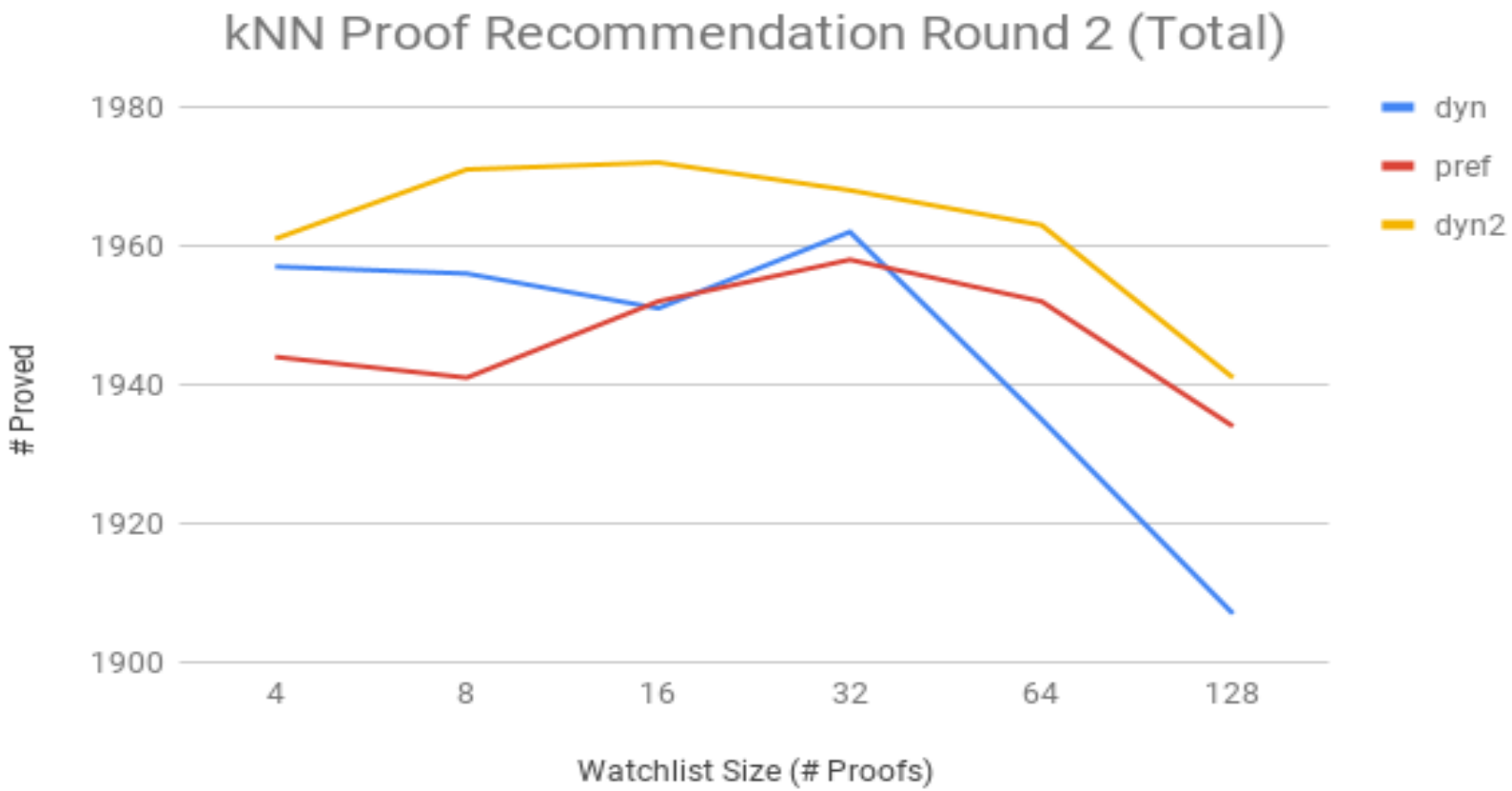
k-NN Dynamic Watchlist Round 1



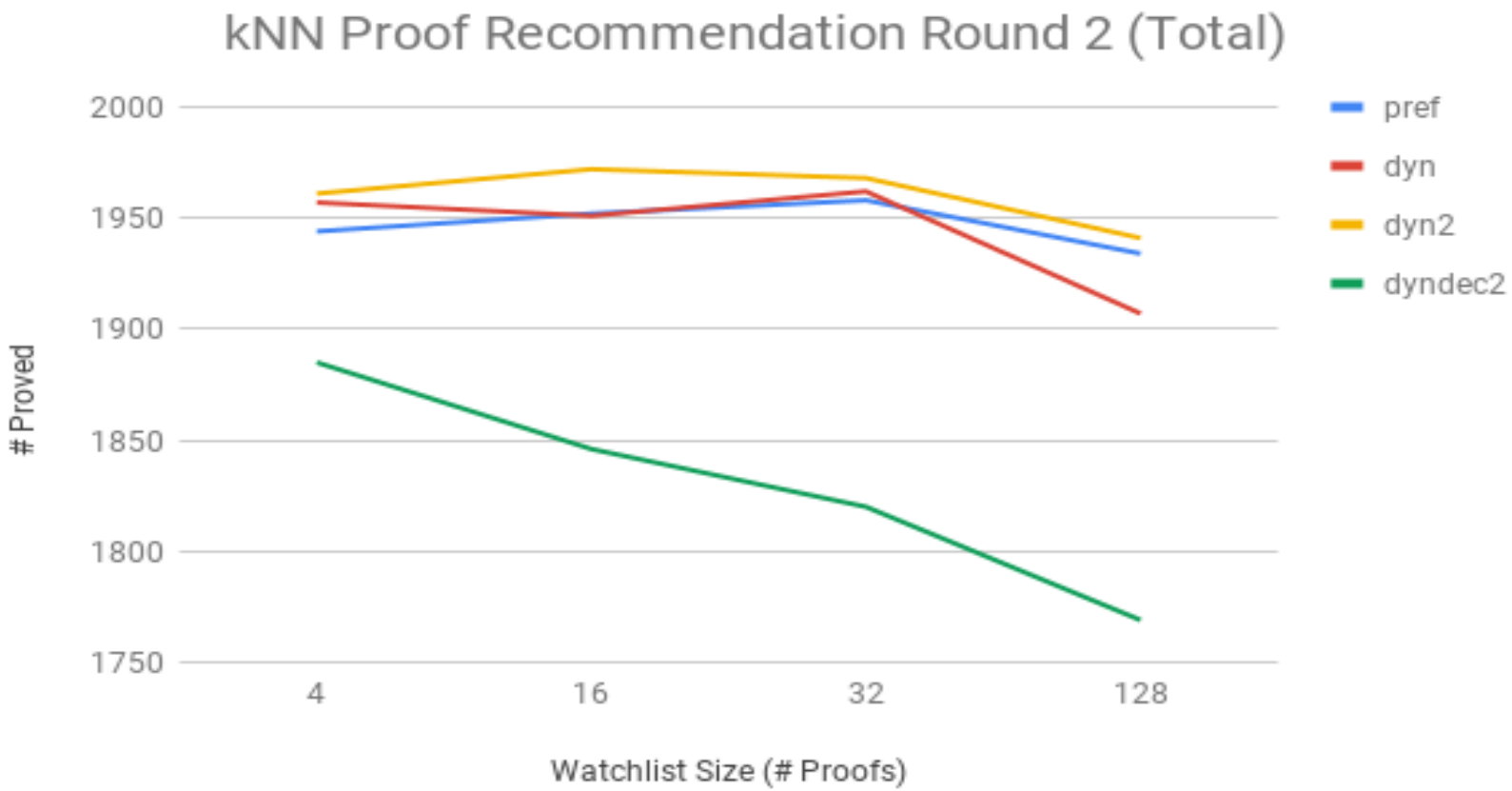
k-NN Dynamic Watchlist Round 1



k-NN Dynamic Watchlist Round 2

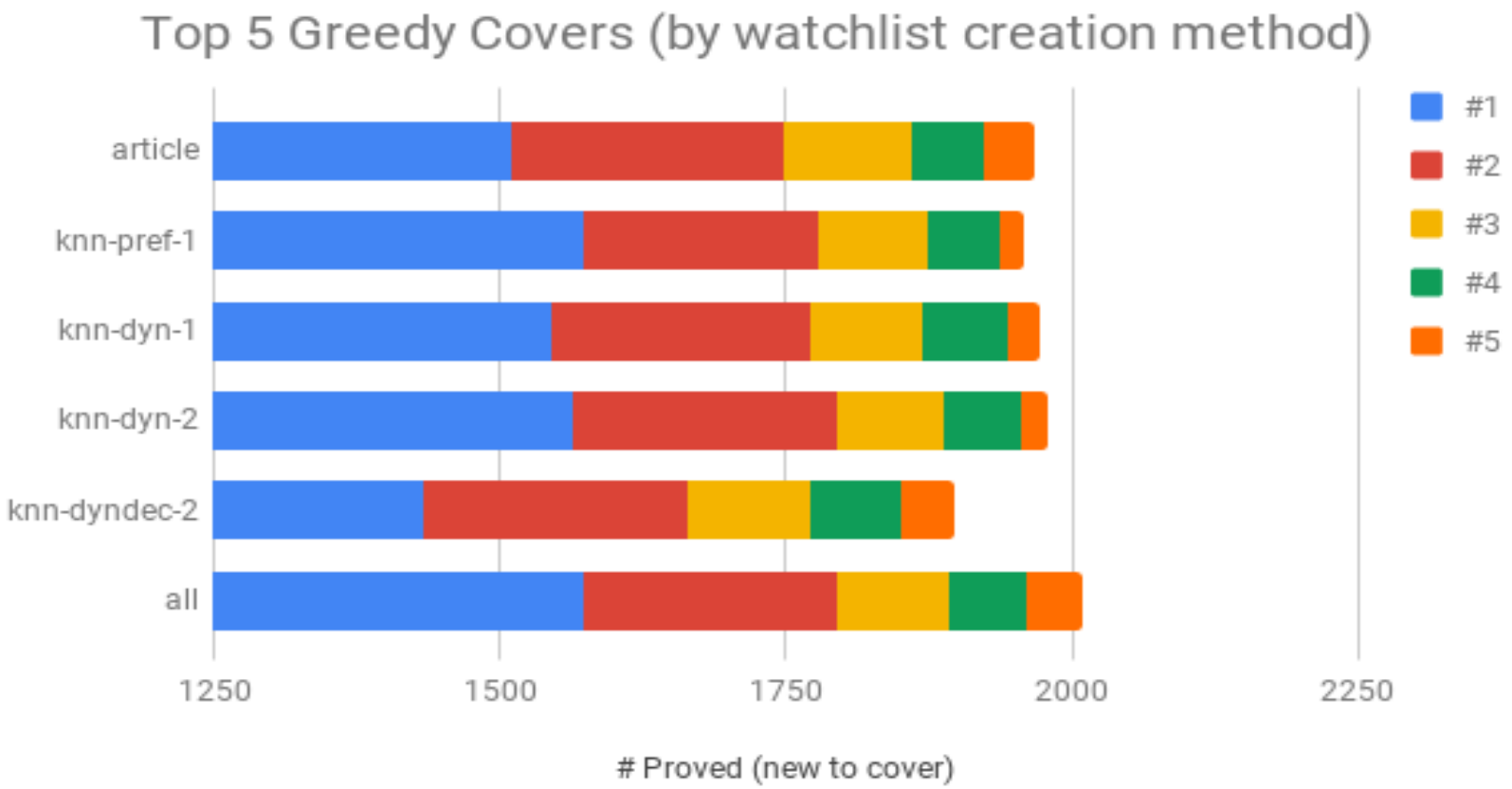


k-NN Dynamic Inheritance Watchlist



Divergence from baseline

Grand Total Results



Grand Total Results

total	knn-prefA_16	dyn2C_16	dyndecD_16	dynE_4	dyndecA_128
2007	1565	230	97	68	47
23192	18583	2553	1050	584	422
23192	18583	14486	14514	13532	15916

Table: Top: Cumulative sum of the 5000 test set greedy cover. Recall the baseline covered 1853 and *const* 1910. Bottom: Greedy cover run on the full dataset, cumulative and total proved. The top 5 baseline strategies covered 21657. Thus there's an improvement of 7%.

De Morgan's laws for Boolean lattices

```
theorem Th36: :: YELLOW_5:36
  for L being non empty Boolean RelStr
  for a, b being Element of L holds
  ( 'not' (a "\/" b) = ('not' a) "/\" ('not' b) & 'not' (a "/\" b) = ('not' a) "\/" ('not' b) )
```

helped significantly by

```
theorem :: WAYBEL_1:85
  for H being non empty lower-bounded RelStr st H is Heyting holds
  for a, b being Element of H holds 'not' (a "/\" b) >= ('not' a) "\/" ('not' b)
```

Note the weaker assumptions on H: just lower bounded and Heyting. Yet, 62 (80.5%) of the 77 clauses from the proof of `WAYBEL_1:85` are matched.

De Morgan's laws for Boolean lattices - Statistics

- 32 related proofs results in 2220 clauses on the watchlists.
- The dynamically guided proof search takes 5218 (nontrivial) given clause loops
- The ATP proof is 436 inferences long
- 194 given clauses match the watchlist in the proof search
- 120 (61.8%) of them end up being part of the proof
- 27.5% of the proof steps guided by the watchlist
- The proof search without the watchlist takes 6550 nontrivial given clause loops (25.5% more)

Final proof progress

0	0.438	42/96	1	0.727	56/77	2	0.865	45/52	3	0.360	9/25
4	0.750	51/68	5	0.259	7/27	6	**0.805**	**62/77**	7	0.302	73/242
8	0.652	15/23	9	0.286	8/28	10	0.259	7/27	11	0.338	24/71
12	0.680	17/25	13	0.509	27/53	14	0.357	10/28	15	0.568	25/44
16	0.703	52/74	17	0.029	8/272	18	0.379	33/87	19	0.424	14/33
20	0.471	16/34	21	0.323	20/62	22	0.333	7/21	23	0.520	26/50
24	0.524	22/42	25	0.523	45/86	26	0.462	6/13	27	0.370	20/54
28	0.411	30/73	29	0.364	20/55	30	0.571	16/28	31	0.357	10/28

Table: Final state of the proof progress for the (serially numbered) 32 proofs loaded to guide the proof of `YELLOW_5:36`. We show the computed ratio and the number of matched and all clauses.

ProofWatch Only Example

```
theorem :: BOOLEALG:68
  for L being B_Lattice
  for X, Y being Element of L holds (X \+\ Y) ` = (X "/\" Y) "\/" ((X `) "/\" (Y `))
proof end;
```

using 272/285, 283/350, and 155/233 clauses from:

```
theorem Th33: :: BOOLEALG:33
  for L being B_Lattice
  for X, Y being Element of L holds X \ (X "/\" Y) = X \ Y
proof end;

theorem :: BOOLEALG:67
  for L being B_Lattice
  for X, Y, Z being Element of L holds (X \+\ Y) \+\ Z = X \+\ (Y \+\ Z)
proof end;

theorem Th23: :: LATTICES:23
  for L being B_Lattice
  for a, b being Element of L holds (a "/\" b) ` = (a `) "\/" (b `)
proof end;
```

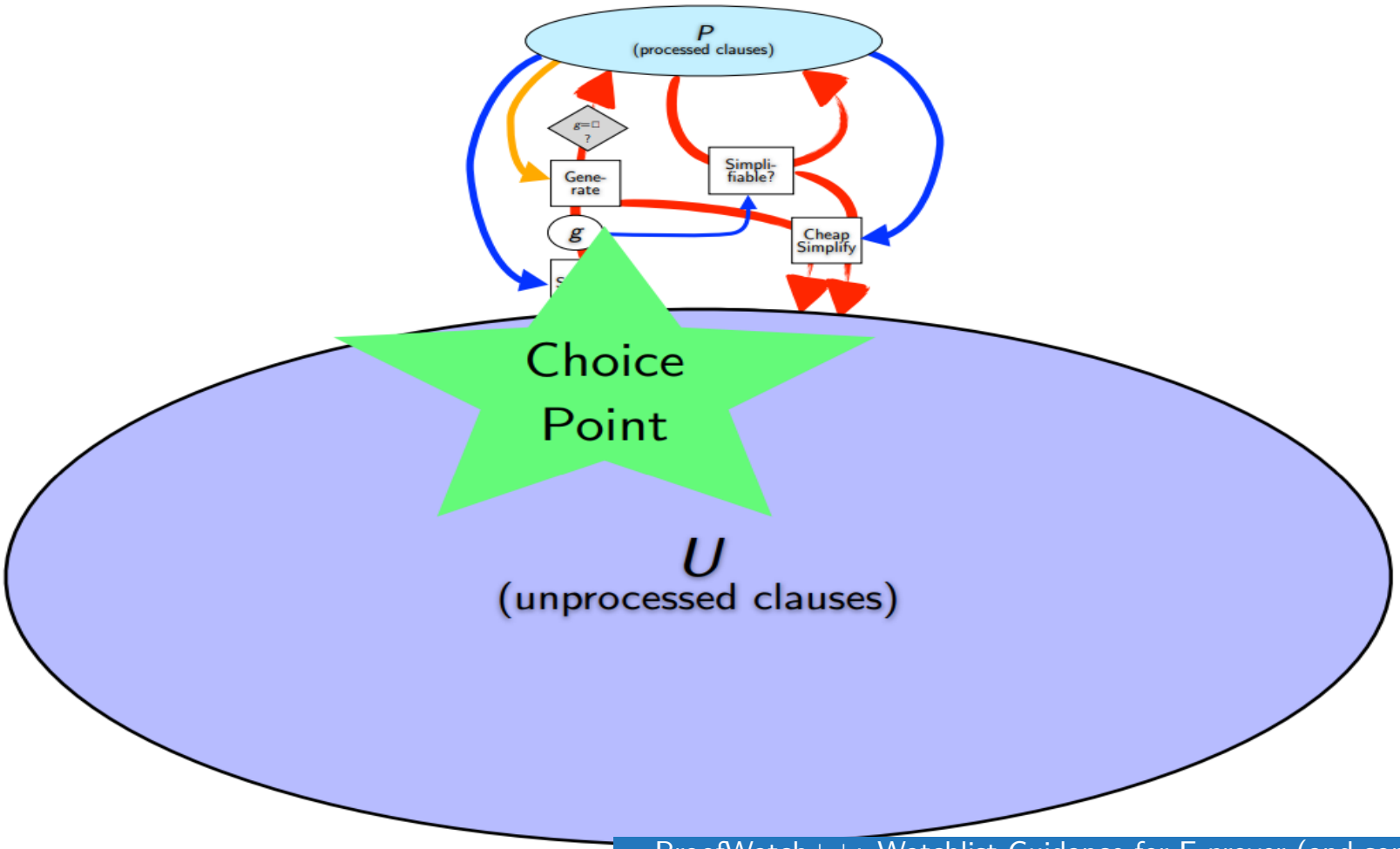
Proof Progress

```
#   watchlist    0: 0.714 (       5 /       7)
#   watchlist    1: 0.500 (      19 /      38)
#   watchlist    2: 0.649 (      72 /     111)
#   watchlist    3: 0.945 (      69 /      73)
#   watchlist    4: 0.618 (      21 /      34)
#   watchlist    5: 0.954 (     272 /     285)
#   watchlist    6: 0.565 (      13 /      23)
#   watchlist    7: 0.765 (      13 /      17)
#   watchlist    8: 0.484 (      30 /      62)
#   watchlist    9: 0.639 (      69 /     108)
#   watchlist   10: 0.809 (     283 /     350)
#   watchlist   11: 0.460 (      40 /      87)
#   watchlist   12: 0.667 (      52 /      78)
#   watchlist   13: 0.765 (      78 /     102)
#   watchlist   14: 0.833 (     110 /     132)
#   watchlist   15: 0.665 (     155 /     233)
```

Conclusion (of ProofWatch talk)

- Upgraded E's watchlist to include dynamic proof-completion and relevance inheritance.
  - and to output a proof-state vector with contextual information.
- Benchmarked E's watchlist feature on Mizar Mathematical Library.
- Demonstrated k-NN proof recommendation as effective for watchlist creation.
  - in successive rounds
- ProofWatch obtains 7% improvement over strong non-watchlist E strategies.

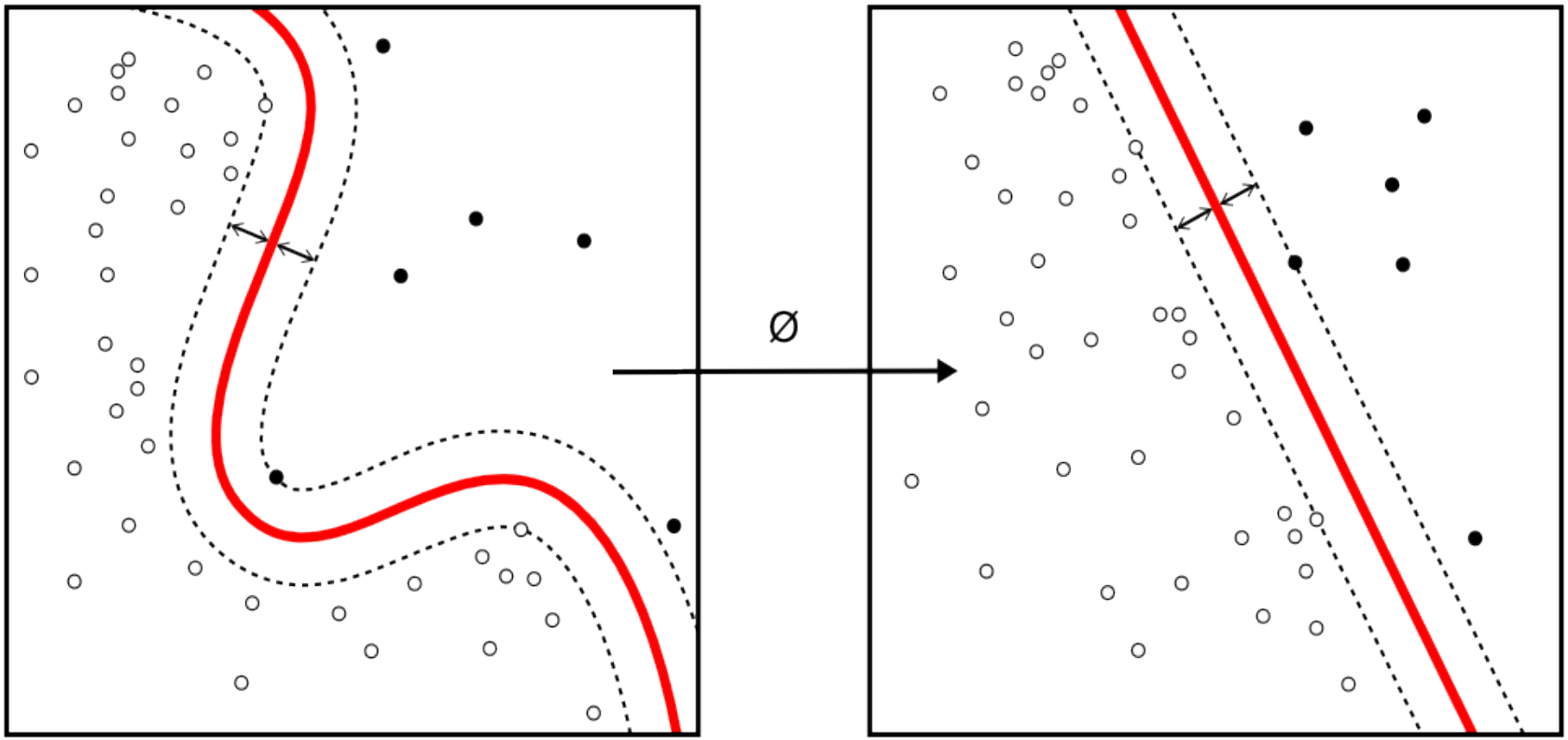
Motivation: Use Machine Learning!



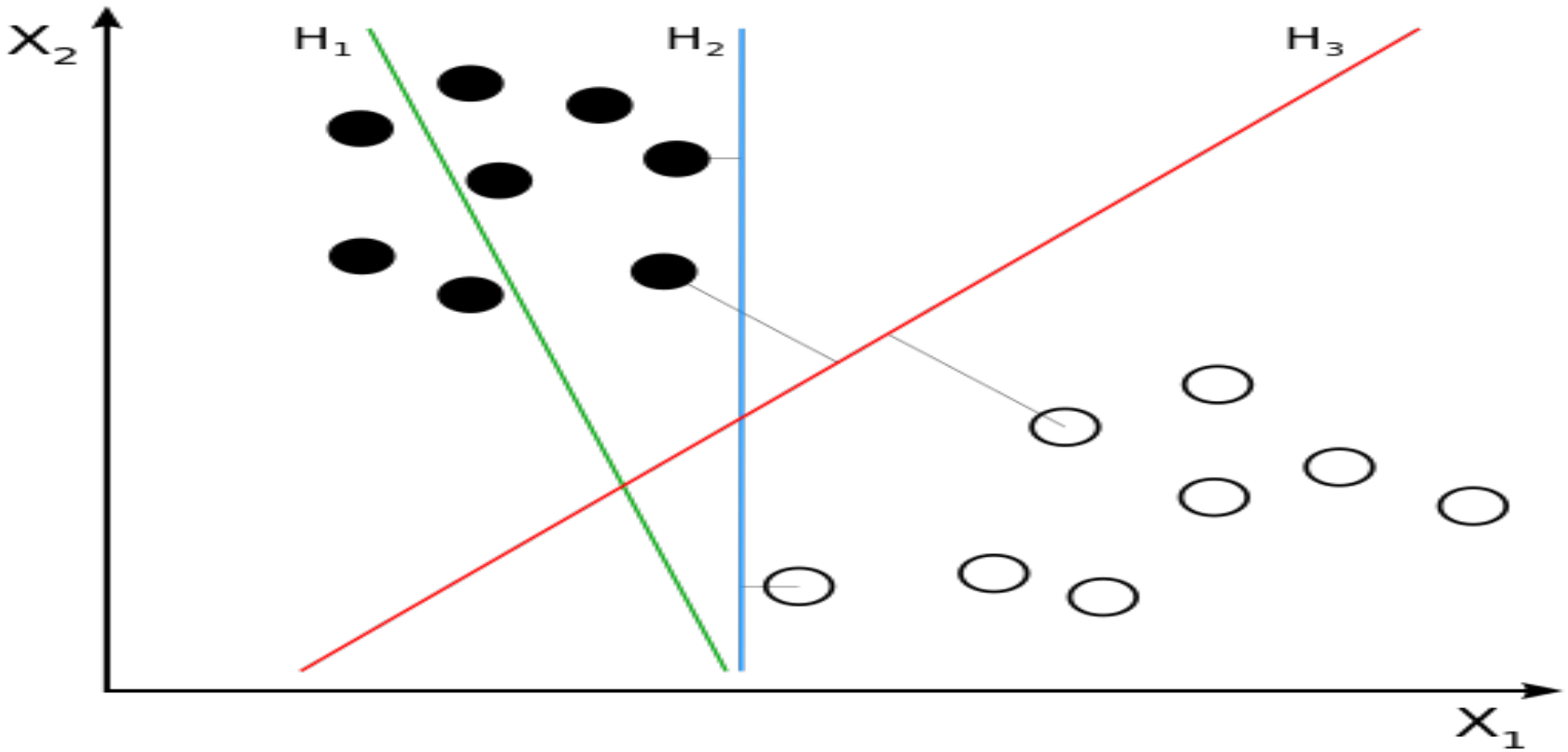
ENIGMA: Efficient learNing-based Inference Guiding MAchine

- Learn a linear model to *classify* clauses as **good** or **bad**.
- Train on proof-data:
  - C_1 : Feature_Vector # trainpos
  - C_2 : Feature_Vector # trainneg
  - . . .
- Turn ENIGMA prediction into weight function (as a CEF):
  - weight$(C, M) = \gamma$length(C) + predict-weight(C, M)
  - Predicted weighst are 1 (positive) or 10 (not positive)

ENIGMA: LIBLINEAR Support Vector Machine



ENIGMA: LIBLINEAR Support Vector Machine



Can we combine ProofWatch + ENIGMA?

- ProofWatch: logically-grounded given clause selection (with some ML)
- ENIGMA: Machine Learned model based given clause selection

Can we combine ProofWatch + ENIGMA?

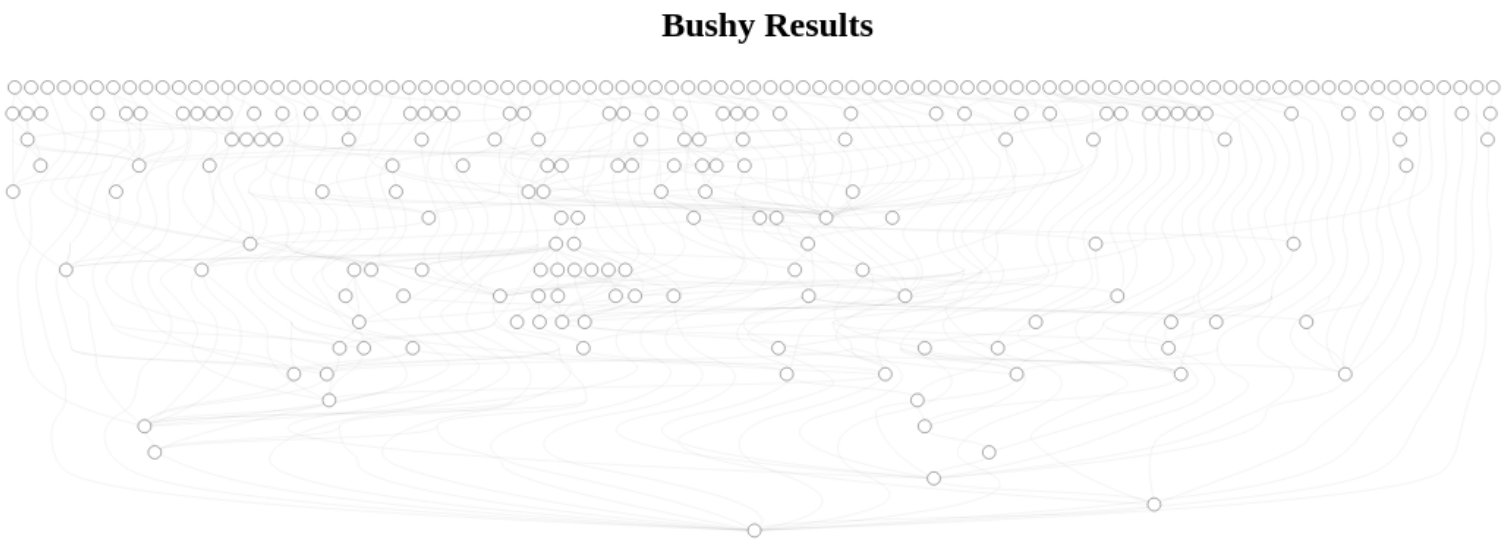
- ProofWatch: logically-grounded given clause selection (with some ML)
- ENIGMA: Machine Learned model based given clause selection
- Key: feed Proof Vectors to ENIGMA.
  - Represents current proof state wrt a set of proofs
  - Add this to ENIGMA features

Proof Progress

```
#   watchlist    0: 0.714 (       5 /       7)
#   watchlist    1: 0.500 (      19 /      38)
#   watchlist    2: 0.649 (      72 /     111)
#   watchlist    3: 0.945 (      69 /      73)
#   watchlist    4: 0.618 (      21 /      34)
#   watchlist    5: 0.954 (     272 /     285)
#   watchlist    6: 0.565 (      13 /      23)
#   watchlist    7: 0.765 (      13 /      17)
#   watchlist    8: 0.484 (      30 /      62)
#   watchlist    9: 0.639 (      69 /     108)
#   watchlist   10: 0.809 (     283 /     350)
#   watchlist   11: 0.460 (      40 /      87)
#   watchlist   12: 0.667 (      52 /      78)
#   watchlist   13: 0.765 (      78 /     102)
#   watchlist   14: 0.833 (     110 /     132)
#   watchlist   15: 0.665 (     155 /     233)
```

First Test: MPTP Challenge

Prove Bolzano-Weierstrass theorem.



MPTP Challenge Results

Run E for 15 seconds or 40,000 given clause loops. With 10 strategies. 252 problems total.

- 195 – Baseline strategies
- 200 – Watchlist strategies (*pref*)
- 202 – ENIGMA
- 205 – ENIGMA + Watchlist strategies

Next challenge: MPTP2078

- Also Bolzano-Weierstrass theorem.
- But all theorems from the 33 relevant Mizar articles.
- Baseline strategies prove 1461: too much for ProofWatch to handle

Next challenge: MPTP2078

- Also Bolzano-Weierstrass theorem.
- But all theorems from the 33 relevant Mizar articles.
- Baseline strategies prove 1461: too much for ProofWatch to handle
- How to construct good *proof vectors*?

Proof Vector Construction Ideas

- Use k-NN as in ProofWatch: large proof vector with many zeros in each case.
- Use K-means to cluster theorems/proofs into sufficiently small sets.
- Seek generic 'basis' vector for all 1461 proofs.
  - Initial k-mediods experiment failed :'(.