Enhancing ENIGMA Given Clause Guidance

Jan Jakubův[1] Josef Urban[1]

AITP'18, Aussois, 29th March 2018

[1]Czech Technical University in Prague

Outline

ATPs & Given Clauses

Enigma Models

Enhanced Features

Experiments with Boosting & Looping

Outline

Given Clause Loop Paradigm

Problem representation

- first order clauses (ex. "$x = 0 \vee \neg P(f(x, x))$")
- posed for proof by contradiction

Given an initial set C of clauses and a set of inference rules, find a derivation of the *empty clause* (for example, by the resolution of clauses with conflicting literals L and $\neg L$).

Basic Loop

```
Proc = {}
Unproc = all available clauses
while (no proof found)
{
   select a given clause C from Unproc
   move C from Unproc to Proc
   apply inference rules to C and Proc
   put inferred clauses to Unproc
}
```

Clause Selection Heuristics in E Prover

- E Prover has several pre-defined clause weight functions. (and others can be easily implemented)
- Each weight function assigns a real number to a clause.
- Clause with the smallest weight is selected.

E Prover Strategy

- E strategy = E parameters influencing proof search (term ordering, literal selection, clause splitting, ...)
- Weight functions to guide given clause selection.
- Several clause weight functions can be combined together:

```
(10  *  ClauseWeight1(10,0.1,...),
 1   *  ClauseWeight2(...),
 20  *  ClauseWeight3(...))
```

Outline

Enigma Basics

- **Idea:** Use fast linear classifier to guide given clause selection!
- **ENIGMA** stands for...

Enigma Basics

- **Idea:** Use fast linear classifier to guide given clause selection!
- **ENIGMA** stands for...

Efficient learNing-based Inference Guiding MAchine

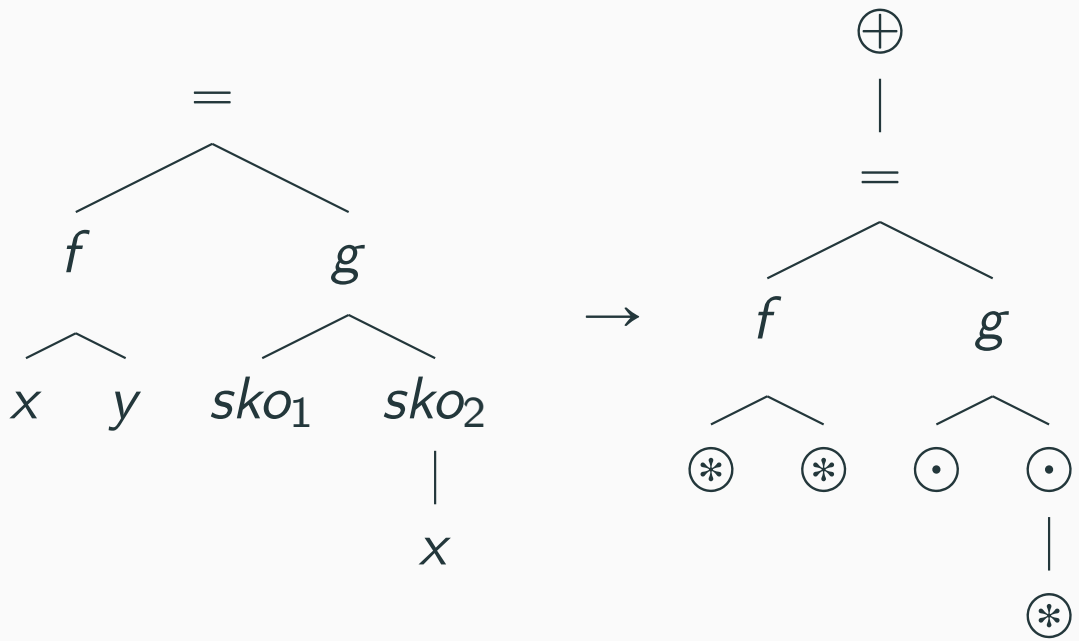
LIBLINEAR: Linear Classifier

- LIBLINEAR: open source library[1]
- **input:** positive and negative examples (float vectors)
- **output:** model ($\sim$ a vector of weights)
- **evaluation** of a generic vector: dot product with the model

[1] `http://www.csie.ntu.edu.tw/~cjlin/liblinear/`

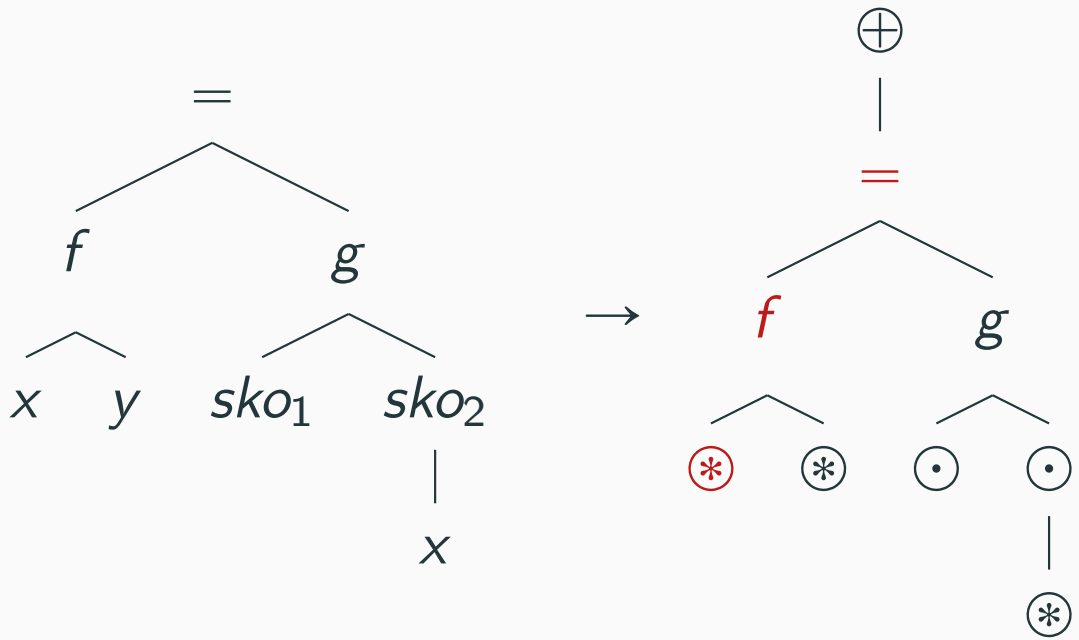
Clauses as Feature Vectors

Consider the literal as a tree and simplify (sign, vars, skolems).



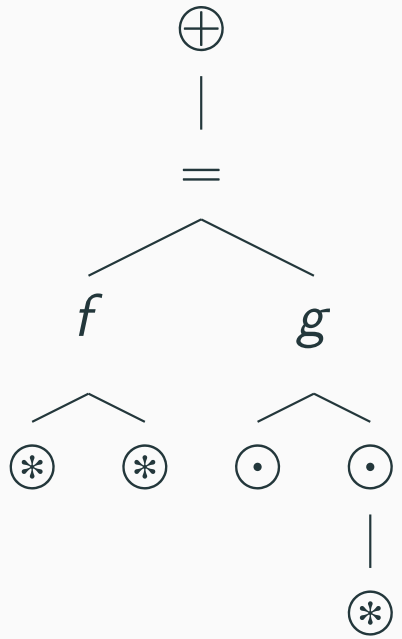
Clauses as Feature Vectors

Features are descending paths of length 3 (triples of symbols).



Clauses as Feature Vectors

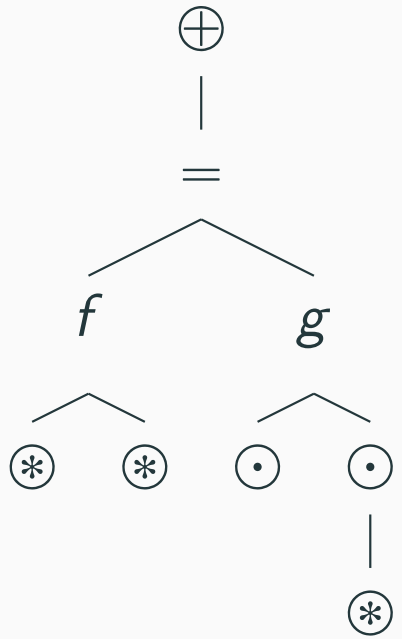
Collect and enumerate all the features. Count the clause features.



#	feature	count
1	(⊕,=,a)	0
⋮	⋮	⋮
11	(⊕,=,f)	1
12	(⊕,=,g)	1
13	(=,f,⊛)	2
14	(=,g,⊙)	2
15	(g,⊙,⊛)	1
⋮	⋮	⋮

Clauses as Feature Vectors

Take the counts as a feature vector.



#	feature	count
1	(⊕,=,a)	0
⋮	⋮	⋮
11	(⊕,=,f)	1
12	(⊕,=,g)	1
13	(=,f,⊛)	2
14	(=,g,⊙)	2
15	(g,⊙,⊛)	1
⋮	⋮	⋮

Enigma Model Construction

1. Collect training examples from E runs (useful/useless clauses).
2. Enumerate all the features (π :: feature $\rightarrow$ int).
3. Translate clauses to feature vectors.
4. Train a LIBLINEAR classifier (w :: float$^{|\mathsf{dom}(\pi)|}$).
5. Enigma model is $\mathcal{E} = (\pi, w)$.

Given Clause Selection by Enigma

We have Enigma model $\mathcal{E} = (\pi, w)$ and a generated clause C.

1. Translate C to feature vector Φ_C using π.
2. Compute prediction:

$$\text{weight}_0(C) = \begin{cases} 1 & \text{iff } w \cdot \Phi_C > 0 \\ 10 & \text{otherwise} \end{cases}$$

3. Combine prediction with clause length:

$$\text{weight}(C) = \text{weight}_0(C) + \delta * |C|$$

Enigma Given Clause Selection

- We have implemented `Enigma` weight function in E.
- Enigma model can be used alone to select a given clause:

$$(1 \quad * \quad \texttt{Enigma}(\mathcal{E},\delta))$$

- or in combination with other E weight functions:

$$\begin{aligned}(23 \quad &* \quad \texttt{Enigma}(\mathcal{E},\delta),\\ 3 \quad &* \quad \texttt{StandardWeight}(\ldots),\\ 20 \quad &* \quad \texttt{StephanWeight}(\ldots))\end{aligned}$$

Outline

Conjecture Features

- Enigma classifier $\mathcal{E}$ is independent on the goal conjecture!
- Improvement: Extend Φ_C with goal conjecture features.
- Instead of vector Φ_C take vector (Φ_C, Φ_G).

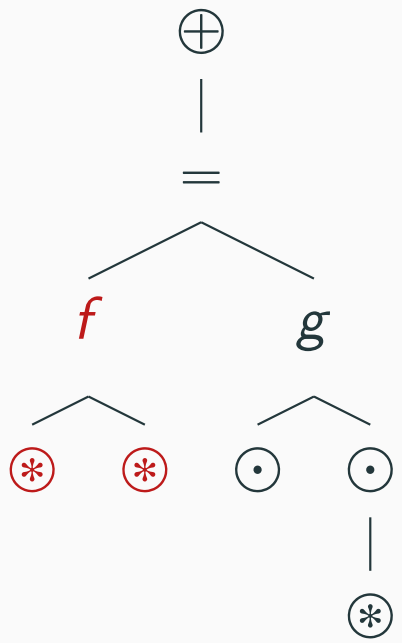
Conjecture Features and Prediction Rates (%)

AIM	train accuracy		10-fold cross-val	
data	noconj	conj	noconj	conj
simple	84.7	84.6	84.6	84.5
50-50	76.3	78.0	76.3	77.8

MZR	train accuracy		10-fold cross-val	
data	noconj	conj	noconj	conj
simple	92.2	95.0	90.8	93.9
50-50	89.2	91.9	88.8	91.5

Horizontal Features

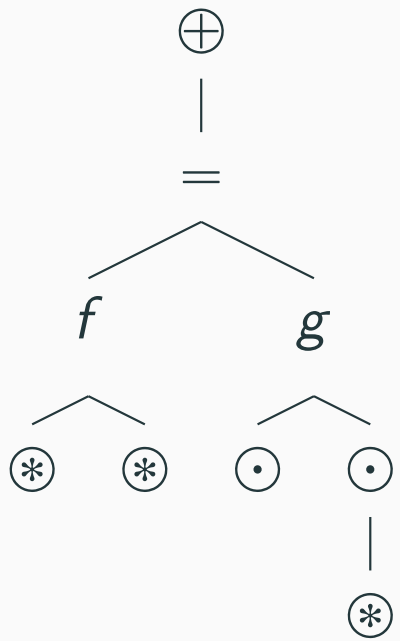
Function applications and arguments top-level symbols.



#	feature	count
1	(⊕,=,a)	0
⋮	⋮	⋮
100	$= (f, g)$	1
101	$f(\circledast, \circledast)$	1
102	$g(\odot, \odot)$	1
103	$\odot(\circledast)$	1
⋮	⋮	⋮

Static Clause Features

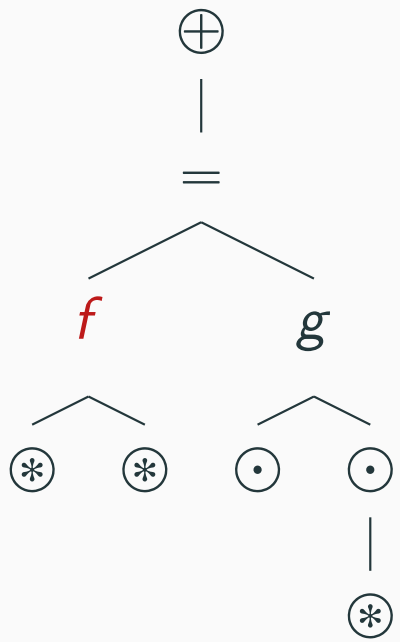
For a clause, its length and the number of pos./neg. literals.



#	feature	count/val
103	⊙(⊛)	1
⋮	⋮	⋮
200	len	9
201	pos	1
202	neg	0
⋮	⋮	⋮

Static Symbol Features

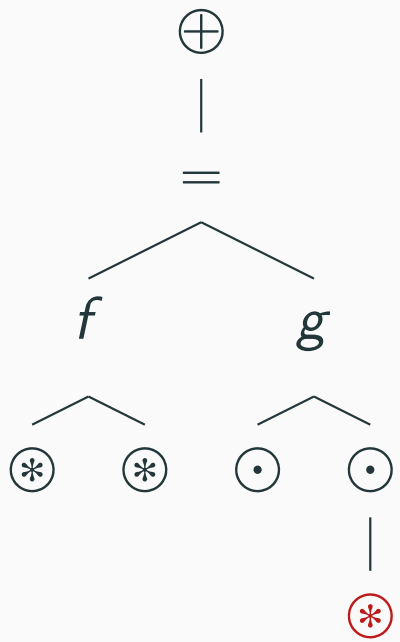
For each symbol, its count and maximum depth.



#	feature	count/val
202	neg	0
⋮	⋮	⋮
300	$\#_{\oplus}(f)$	1
301	$\#_{\ominus}(f)$	0
⋮	⋮	⋮
310	$\%_{\oplus}(\circledast)$	4
311	$\%_{\ominus}(\circledast)$	0
⋮	⋮	⋮

Static Symbol Features

For each symbol, its count and maximum depth.



#	feature	count/val
202	neg	0
⋮	⋮	⋮
300	$\#_{\oplus}(f)$	1
301	$\#_{\ominus}(f)$	0
⋮	⋮	⋮
310	$\%_{\oplus}(\circledast)$	4
311	$\%_{\ominus}(\circledast)$	0
⋮	⋮	⋮

Outline

Boosting

- Training data are uneven.
- Usually we have more negative examples (cca 10 times).
- Previously: Repeat positive examples 10 times.

Smarter Boosting

1. Collect training data.
2. Create classifier $\mathcal{E} = (\pi, w)$.
3. Compute prediction accuracy on the training data (using w).
4. **If** $(acc^+ > acc^-)$ **then** finish.
5. Repeat misclassified positive clauses in the training data.
6. **Goto** 2.

Looping

1. Run E prover with strategy S on problems P.
2. Collect/extend training data.
3. Create classifier $\mathcal{E} = (\pi, w)$ from the training data.
4. Construct strategies $S^0_{\mathcal{E}}$ and $S_{\mathcal{E}}$.
5. Evaluate $S^0_{\mathcal{E}}$ and $S_{\mathcal{E}}$ on problems P.
6. **Goto** 2.

Experiments with Clustering

- MPTP benchmarks (2079 problems from Mizar).
- 10 E Prover strategies from auto mode (autos).
- Problems are clustered into 33 articles/categories.
- We train Enigma separately on all articles (for each S).
- We take best-performing strategies on each article.

Best Enigma Models per Article

article	total	best autos	best Enigma	Enigma+
compts	23	7	7	+0.0%
enumset1	96	86	86	+0.0%
pre	37	21	22	+4.8%
relset	32	20	22	+10.0%
funct	235	160	185	+15.6%
filter	65	6	7	+16.7%
orders	61	28	36	+28.6%
wellord1	59	27	35	+29.6%
waybel	174	42	74	+76.2%

Total Portfolio Improvement on MPTP

portfolio	solved	autos%	autos+	autos-
E (auto-schedule)	1343	-3.8%	+25	-79
autos (10)	1397	+0%	+0	-0
Enigmas (62)	1450	+3.7%	+103	-50

Thank you.