Learning to Advise an Equational Prover

Chad E. Brown[1], Bartosz Piotrowski[1,2], Josef Urban[1]

[1]Czech Technical University
[2]University of Warsaw

AITP
17 September 2020
Aussois

Introduction

- `aimleap` is a simple prover for solving equations like this one:
  `T(T(L(x,y,z),w),L(x,y,z)\x) = T((L(x,y,z)\x)\x,w)`.
- `aimleap` can benefit from an advisor which can estimate lengths of proofs of equations $s = t$.
- In this work we provide a machine-learned advisor to `aimleap`.
- We use data coming from the AIM project.

Search procedure in `aimleap` prover

Initial parameters:

- $s = t$ – an equation to be proven,
- $\mathcal{A}$ – a set of known equations; we fixed a set of 87 equations,
- n – a maximum allowed distance; we set it to 10,

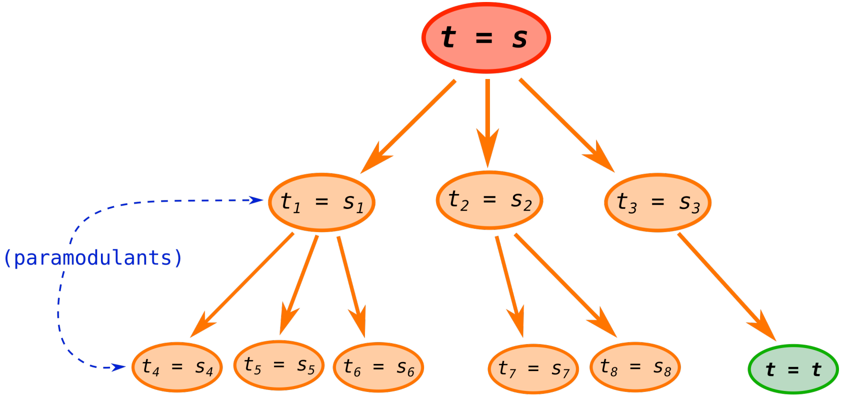
Procedure:

1. If s and t are unifiable, then report success.
2. If $n = 0$, then report failure.
3. Compute a finite set of *paramodulants* $s_i = t_i$. These are defined as rewrites of $s = t$ by a single equation from $\mathcal{A}$.
4. Order these paramodulants using an advisor, filtering out those which the advisor deems to require more than $n - 1$ paramodulation steps to complete the proof, and for each one ask if $s_i = t_i$ is provable in $n - 1$ steps.

Another constraint:

- m – abstract time limit (# of recursive calls); we set it to 100.

Search procedure in aimleap prover



87 basic equations

```
(Loop Axioms)
lid : e * x = x
rid : x * e = x
b1  : x \ (x * y) = y
b2  : x * (x \ y) = y
s1  : (x * y) / y = x
s2  : (x / y) * y = x

(Definitions)
a(x,y,z) := (x*(y*z))\((x*y)*z)
K(x,y) := (y*x)\(x*y)
T(u,x) := x\(u*x)
L(u,x,y) := (y*x)\(y*(x*u))
R(u,x,y) := ((u*x)*y)/(x*y)

(AIM Axioms)
TT: T(T(u,x),y) = T(T(u,y),x)
TL: T(L(u,x,y),z) = L(T(u,z),x,y)
TR: T(R(u,x,y),z) = R(T(u,z),x,y)
LR: L(R(u,x,y),z,w) = R(L(u,z,w),x,y)
LL: L(L(u,x,y),z,w) = L(L(u,z,w),x,y)
RR: R(R(u,x,y),z,w) = R(R(u,z,w),x,y)
```

```
(70 additional equations)
x / x = e
e \ x = x
x / e = x
x \ x = e
(y / x) \ y = x
x * T(y,x) = y * x
T(x / y,y) = y \ x
(x * T(y,x)) / x = y
(x * y) * K(y,x) = y * x
T(x,x \ y) = (x \ y) \ y
x*T(T(y,x),z) = T(y,z)*x
T(T(x/y,z),y) = T(y\x,z)
(x*y)*L(z,y,x) = x*(y*z)
L(x\y,x,z) = (z*x)\(z*y)
R(x,y,z)*(y*z) = (x*y)*z
R(x/y,y,z) = (x*z)/(y*z)
x*((x\e)*y) = L(y,x\e,x)
(x\e)*y = x\L(y,x\e,x)
        .
        .
        .
```

Data set

- Veroff obtained a large number of AIM proofs using Prover9.
- We extracted 3468 equations from them.
- Each equation $s = t$ has recorded *distance* between s and t.

Distance	Number of problems
2	1641 (47.3%)
3	869 (25.0%)
4	353 (10.2%)
5	284 (8.2%)
6–10	372 (10.5%)

- Additionally, we created 10000 synthetic equations.
- The extracted examples are used for testing, the synthetic ones – for training.

Data set – examples

Bunch of training examples of form $(s = t, \mathit{dist})$:

s	*t*	*dist*
T(T(T(T(x,y),z),x),w	T(T(T(T(x,y),x),z),w	1
T((e/x)*y,z	T(((e/x)*x)\((e/x)*y),z	2
T(e\((e/x)*y),z	L(T(x\y,z),x,e/x	3
x*L(x\(x/y),z,w	((x/y)*y)*L(L(y\e,z,w),y,x/y	4
(X*Y)/L(x\Y,x,(y*z)/(w*z))	R(y/w,w,z)*x	5
K((x\y)\y,z)*T(x,x\y	X/((K((x\y)\y,z)*((x\y)\y))\X	6
(x/((y\e)*x))*T(z,R(y,y\e,x)	z*R(X/(y\X),y\e,x	9

Rote learner

- As a sanity check an *oracle advisor* aka *rote learner* was used:
  - for all (sub)goals seen in the proofs it returns the true distance,
  - for unseen goals it returns 50 (effectively prunning them out).
- The `aimleap` prover with the oracle advisor can reprove all the 3465 problems (with no backtracking).
- We tested the rote learner in a cross-validation scenario:
  - data split into 10 parts,
  - the rote learner tested on one part can use knowledge only from the remaining 9 parts.
- Success rate in that setting: 21.9% (800 problems solved).

Constant distance

- We tested an advisor giving simply constant distance c for each equation $s = t$ for which s is not equal t, or 0 otherwise.
- The results:

Constant	Solved problems
0	0
1 – 7	135 (3.9%)
8	138 (4.0%)
9	1739 (50.1%)
10	132 (3.8%)

- Constant distance 9 performs so well because it makes the search more breadth-first-like and the prover easily solves all the goals with distances 1 and 2 ($\approx$ 50% of the problems).

Training the advisor

- For providing machine-learned advice we used XGBoost.
- Training examples were fed into the model as features of pairs of terms and the corresponding distance between them.
- We used ENIGMA-style features, i.e., paths of lengths 1–3 from the term's parse tree, with numbers of their occurrences.
- Hyperparameters of XGBoost were: objective function – *mean squared error*, number of boosting rounds – 1000, maximal depth of a decision tree – 10, learning rate – 0.1.
- The advisor was trained on a separate set of 10000 synthetized examples.

Accuracy and search results of the advisor

- On a cross-validation split the performance metrics of the trained advisor were:
    - root mean square error: 1.1,
    - accuracy: 59%.
- `aimleap` with the advisor plugged-in and an additional constraint of 60 second time limit could solve 299 problems out of 3468 testing problems (only 9% ...)
- But: there were 135 problems not solved by the rote learner and 18 problems not solved with any constant-distance advice.

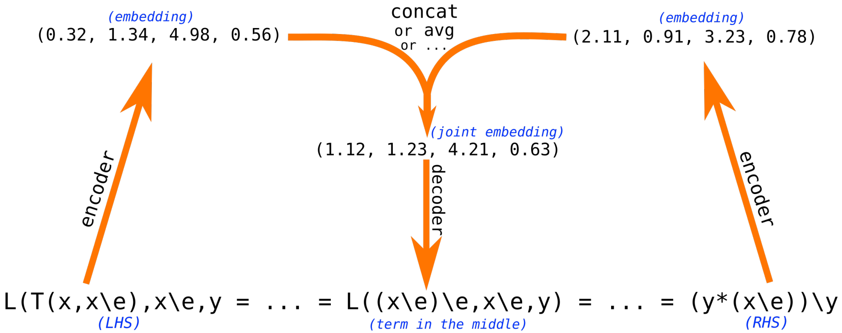
First-order automated provers

- For further comparison we gave the problems to three automated provers: Prover9, Waldmeister and E.
- For all of them a timeout of 60 seconds was used.

Prover	Solved problems	Not solved but solved by `aimleap`
E	1342 (38.6%) or 2684 (77.4%)	113
Prover9	2037 (58.7%)	49
Waldmeister	2170 (62.6%)	92

Next experiment: synthetizing term in the middle

- Try to guess <u>*term-in-the-middle*</u>:



- Having produced the term, try to prove:
  <u>*LHS = term-in-the-middle*</u> and <u>*term-in-the-middle = RHS*</u>.