Machine Learning of Given Clause Selection in E

Jan Jakubův, Josef Urban

Automated Reasoning Group
Czech Technical University in Prague

AITP16, Obergurgl, 4th April 2016

Terminology

- (FOL) Term (t): X, c, $f(t_1, \cdots, t_n)$, \$true, \$false
- Predicate (p): basically a term
- Literal: $t_1 = t_2$ or $t_1 \neq t_2$ (spec. $p(X) = \$\text{true}$, $p(X) \neq \$\text{true}$)
- Clause: a set of literals (implicitly OR-ed)
- Axioms: a set of clauses
- Conjecture: a clause

Saturation Based Theorem Proving

Basic variant without subsumption

$P := \emptyset$ *(processed)*
$U := \{\textit{clausified axioms and a negated conjecture}\}$ *(unprocessed)*
while $(U \neq \emptyset)$ do
  if $(\emptyset \in U \cup P)$ then return *Unsatisfiable*
  $g := \mathrm{select}(U)$
  $P := P \cup \{g\}$
  $U := U \setminus \{g\}$
  $U := U \cup \{\textit{clauses inferred with } g \textit{ and one from } P\}$
done
return *Satisfiable*

Basic Clause Selection Methods

Selecting the “right” clause is crucial:

- pick the shortest clause from U
- pick the oldest clause from U
- count symbols with different weights
  - $C = \{f(X) \neq a, p(a) \neq \$\mathrm{true}\}$
  - $sym(C) = [f, X, a, p, a, \$\mathrm{true}]$
  - $\mathrm{weight}(C) = \sum_{s \in sym(C)} \mathrm{weight}(s)$
- e.g. use lower weights for conjecture symbols
- various combinations

Clause Selection in E

How it is done in E

- Clause evaluation is done by selecting:
  - unparameterized priority function: $\mathit{prio} : \mathit{Clause} \to \mathit{Long}$
  - parametrized weight function: $\mathit{weight} : \mathit{Clause} \times \mathit{args} \to \mathit{Double}$
  - weight function parameters (*args* above)
- unprocessed clauses are pre-sorted by *prio*
- the smaller priority/weight the better
- clauses are stored in a priority queue
  - clause with the smallest $(\mathit{prio}(C), \mathrm{weight}(C))$ is selected

Example Clause Selection Heuristic

A reference heuristic (SYM) for experiments

- A user can choose a heuristic by
  - combining built-in priority and weight functions, and
  - selecting weight function parameters.

```
ConjectureRelativeSymbolWeight(   // weight function
   ConstPrio,  // priority function
   0.1,        // conjecture symbol weight multiplier
   100,        // weight of function symbols
   100,        // weight of constants
   100,        // weight of predicate symbols
   100,        // weight of variables
   1.5,        // maximal term multiplier
   1.5,        // maximal literal multiplier
   1.5)        // positive equality multiplier
```

Combining Different Heuristics

A reference scheme (EXP) for experiments

A user can combine several heuristics:

```
(1*ConjectureRelativeSymbolWeight(
      SimulateSOS,0.5,100,100,100,100,1.5,1.5,1),
 4*ConjectureRelativeSymbolWeight(
      ConstPrio,0.1,100,100,100,100,1.5,1.5,1.5),
 1*FIFOWeight(PreferProcessed),
 1*ConjectureRelativeSymbolWeight(
      PreferNonGoals,0.5,100,100,100,100,1.5,1.5,1),
 4*Refinedweight(SimulateSOS,3,2,2,1.5,2))
```

6 Conjecture Weight Functions

- different ways of relating a clause to the conjecture
- each is determined by a term weight function
- term weight determines a similarity of a term with a conjecture
- all heuristics share some common parameters

Common parameters

Variable normalization

- controlled by parameter $v \in \{\star, \alpha\}$
  - $(\star)$ all variables unified
  - (α) α-normalized variables (in a term)

Common parameters

Related terms

- term weight measures a similarity of a term with some term from the set of related terms (RelatedTerms)
- controlled by parameter $r \in \{\mathrm{ter}, \mathrm{sub}, \mathrm{top}, \mathrm{gen}\}$

(ter) All conjecture terms

(sub) All conjecture subterms

(top) Subterms and top-level generalizations (for any conjecture symbol f add $f(X_1, \cdots, X_n)$)

(gen) Generalizations of all conjecture subterms

Common parameters

Term weight extension

- each heuristic defines a simple term weight function weight_1
- this is extended to term weight "weight(t)"
- controlled by parameter $e \in \{1, \Sigma, \vee\}$

(1) use $\mathrm{weight}(t) = \mathrm{weight}_1(t)$ directly

(Σ) sum $\mathrm{weight}_1(s)$ for each subterm

$$\mathrm{weight}(t) = \sum_{s \in \mathrm{subterms}(t)} \mathrm{weight}_1(s)$$

($\vee$) get maximum of all subterms

$$\mathrm{weight}(t) = \max_{s \in \mathrm{subterms}(t)} \mathrm{weight}_1(s)$$

Common parameters

Clause weight extension

- term weight is extended to clause weight
- $\text{weight}(C) = \text{weight}(\{t_1 = t_2, t_3 \neq t_4, \cdots\}) = \sum_i \text{weight}(t_i)$
- appropriately multiplied by

  (γ_{maxl}) maximal literal multiplier
  (γ_{pos}) positive equality multiplier
  (γ_{maxt}) maximal term multiplier

Term: Conjecture Subterm Weight

- favor related terms
- $\texttt{Term}(P, v, r, \gamma_{\mathsf{conj}}, \delta_{\mathsf{f}}, \delta_{\mathsf{c}}, \delta_{\mathsf{p}}, \delta_{\mathsf{v}}, e, \gamma_{\mathsf{maxt}}, \gamma_{\mathsf{maxl}}, \gamma_{\mathsf{pos}})$
- $\mathsf{weight}_1(t) = \begin{cases} \gamma_{\mathsf{conj}} * \delta & \text{iff } t \in \mathsf{RelatedTerms} \\ \delta & \text{otherwise} \end{cases}$

  where $\delta \in \{\delta_{\mathsf{f}}, \delta_{\mathsf{c}}, \delta_{\mathsf{p}}, \delta_{\mathsf{v}}\}$ accordingly to the top-level symbol of t

TfIdf: Conjecture Frequency Weight

- favor infrequent related terms
- $\texttt{TfIdf}(P, v, r, \delta_{\text{doc}}, e, \gamma_{\text{maxt}}, \gamma_{\text{maxl}}, \gamma_{\text{pos}})$
- $\text{tf}(t)$ = "number of occurrences of t in RelatedTerms"
- $\text{df}(t)$ = "number of Documents which contain t" where
  - Documents are axioms (if $\delta_{\text{doc}} = \text{ax}$)
  - Documents are all processed clauses (if $\delta_{\text{doc}} = \text{pro}$)

$$\text{tfidf}(t) = \text{tf}(t) * \log \frac{1 + |\text{Documents}|}{1 + \text{df}(t)} \qquad \text{weight}_1(t) = \frac{1}{1 + \text{tfidf}(t)}$$

Pref: Conjecture Term Prefix Weight

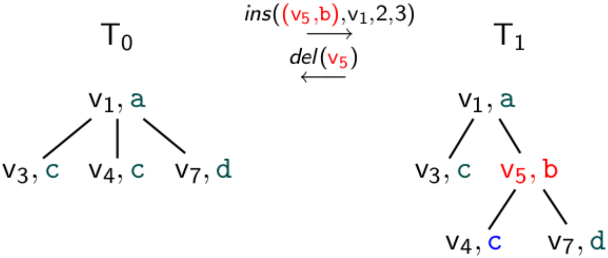
- favor terms which share a prefix with a related term
- $\texttt{Pref}(P, v, r, \delta_{\text{match}}, \delta_{\text{miss}}, e, \gamma_{\text{maxt}}, \gamma_{\text{maxl}}, \gamma_{\text{pos}})$
- $\text{max-pref}(t) =$ "the longest prefix shared with RelatedTerms"
- $\text{weight}_1(t) = \delta_{\text{match}} * |\text{max-pref}(t)| + \delta_{\text{miss}} * (|t| - |\text{max-pref}(t)|)$

Lev: Conjecture Levenshtein Distance Weight

- measure the Levenshtein distance from related terms
- $\mathtt{Lev}(P, v, r, \delta_{\text{ins}}, \delta_{\text{del}}, \delta_{\text{ch}}, e, \gamma_{\text{maxt}}, \gamma_{\text{maxl}}, \gamma_{\text{pos}})$
- variable costs of insert/delete/change operations
- $\text{weight}_1(t) = \min_{s \in \text{RelatedTerms}} \Delta_{\text{Lev}}(t, s)$

Ted: Conjecture Tree Distance Weight

- measure the tree edit distance from related terms
- $\mathtt{Ted}(P, v, r, \delta_{\text{ins}}, \delta_{\text{del}}, \delta_{\text{ch}}, e, \gamma_{\text{maxt}}, \gamma_{\text{maxl}}, \gamma_{\text{pos}})$
- *Ref.* Zhang, Shasha: Simple Fast Algorithms for the Editing Distance Between Trees and Related Problems, 1989.



Struc: Conjecture Structural Distance Weight

- Computes “structural” distance from related terms
- $\mathtt{Struc}(P, v, r, \delta_{\text{miss}}, \gamma_{\text{inst}}, \gamma_{\text{gen}}, e, \gamma_{\text{maxt}}, \gamma_{\text{maxl}}, \gamma_{\text{pos}})$

$$\Delta_{\text{Struc}}(x, y) = \begin{cases} 0 & \text{iff } x = y \\ \delta_{\text{miss}} & \text{otherwise} \end{cases} \qquad \begin{array}{l} \Delta_{\text{Struc}}(x, t) = \gamma_{\text{inst}} * |t| \\ \Delta_{\text{Struc}}(t, x) = \gamma_{\text{gen}} * |t| \end{array}$$

$$\Delta_{\text{Struc}}(f(t_1, \ldots, t_n), f(s_1, \ldots, s_n)) = \sum_{i=1}^{n} \Delta_{\text{Struc}}(t_i, s_i)$$

$$\Delta_{\text{Struc}}(t, s) = \gamma_{\text{gen}} * |t| + \gamma_{\text{inst}} * |s| \qquad \text{(otherwise)}$$

Experimental Evaluation

- Evaluation on 2078 MPTP bushy problems (Mizar)
- SYM heuristic is used as a reference
- EXP heuristic scheme is used to measure diversity (2*EXP+)
- All the experiments were run with $\gamma_{\mathsf{maxt}} = \gamma_{\mathsf{maxl}} = \gamma_{\mathsf{pos}} = 1.5$
- ... with a constant priority function
- ... with 5 seconds time limit
- ... manually chosen parameters

Bests on 2078 MPTP Problems (by solved)

heuristic	δ	solved	speed	%SYM+
Lev	$\star$-gen-1	**841**	2.4	18.3
Struc	$\star$-ter-1	**833**	3.9	17.2
Ted	α-gen-1	**797**	1.2	12.1
Pref	α-gen-Σ	**788**	4.0	10.8
Term	$\star$-gen-Σ	**749**	5.6	5.3
TfIdf	α-gen-Σ	**738**	3.1	3.8
SYM		**711**	3.4	0.0

Bests on 2078 MPTP Problems (by 2*EXP+)

heuristic	δ	2*EXP+	speed	%SYM+
Lev	$\star$-gen-1	**41**	2.4	18.3
Ted	α-gen-1	**33**	1.3	12.1
Struc	$\star$-sub-Σ	**32**	2.9	17.0
Pref	α-gen-Σ	**21**	4.0	10.8
Term	α-gen-1	**20**	4.4	-0.7
TfIdf	$\star$-sub-Σ	**17**	3.5	0.3
SYM		**7**	3.4	0.0

Conclusions

- Lev and Struc work best
- in many case unified variables equal to α-normalization
- "exact match" heuristics perform best with $e = \Sigma$
- different parameters (e.g. costs) matter
- often $r =$ gen is best but not always

Future Work

- Apply machine learning (ParamILS, BliStr) to find
- . . . best parameters for each heuristic
- . . . best combinations with priority functions
- . . . “orthogonal” heuristics with maximal coverage
- Incorporate machine learning directly into given clause selection