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 \( (t) \): \( X, c, f(t_1, \cdots, t_n), \text{true}, \text{false} \)
- 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

\[ P := \emptyset \]  \hspace{2cm} (processed)
\[ U := \{ \text{clausified axioms and a negated conjecture} \} \] \hspace{2cm} (unprocessed)

while \( U \neq \emptyset \) do
  if \( \emptyset \in U \cup P \) then return \textit{Unsatisfiable}
  \[ g := \text{select}(U) \]
  \[ P := P \cup \{g\} \]
  \[ U := U \setminus \{g\} \]
  \[ U := U \cup \{ \text{clauses inferred with } g \text{ and one from } P \} \]
done

return \textit{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
  - $C = \{ f(X) \neq a, p(a) \neq true \}$
  - $sym(C) = [f, X, a, p, a, true]$
  - $weight(C) = \sum_{s \in sym(C)} weight(s)$
- 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: \( \text{prio} : \text{Clause} \rightarrow \text{Long} \)
  - parametrized weight function: \( \text{weight} : \text{Clause} \times \text{args} \rightarrow \text{Double} \)
  - weight function parameters (\( \text{args} \) above)
- unprocessed clauses are pre-sorted by \( \text{prio} \)
- the smaller priority/weight the better
- clauses are stored in a priority queue
  - clause with the smallest \( (\text{prio}(C), \text{weight}(C)) \) 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.

```plaintext
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
```

Jan Jakubův, Josef Urban

Machine Learning of Given Clause Selection in E
Combining Different Heuristics

A reference scheme (EXP) for experiments

A user can combine several heuristics:

\[
(1 \times \text{ConjectureRelativeSymbolWeight}(
    \text{SimulateSOS}, 0.5, 100, 100, 100, 100, 1.5, 1.5, 1.5, 1),
4 \times \text{ConjectureRelativeSymbolWeight}(
    \text{ConstPrio}, 0.1, 100, 100, 100, 100, 1.5, 1.5, 1.5, 1.5),
1 \times \text{FIFOWeight}(\text{PreferProcessed}),
1 \times \text{ConjectureRelativeSymbolWeight}(
    \text{PreferNonGoals}, 0.5, 100, 100, 100, 100, 1.5, 1.5, 1.5, 1),
4 \times \text{Refinedweight}(\text{SimulateSOS}, 3, 2, 2, 1.5, 2))
\]
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 $\nu \in \{\star, \alpha\}$
  - $\star$: all variables unified
  - $\alpha$: $\alpha$-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** \( r \in \{ \text{ter, sub, top, gen} \} \)
  - \((\text{ter})\) All conjecture terms
  - \((\text{sub})\) All conjecture subterms
  - \((\text{top})\) Subterms and top-level generalizations
    (for any conjecture symbol \( f \) add \( f(X_1, \cdots, X_n) \))
  - \((\text{gen})\) Generalizations of all conjecture subterms
Common parameters

Term weight extension

- each heuristic defines a simple term weight function \( \text{weight}_1 \)
- this is extended to term weight “weight(t)”
- controlled by parameter \( e \in \{1, \Sigma, \lor\} \)

1. use \( \text{weight}(t) = \text{weight}_1(t) \) directly
2. sum \( \text{weight}_1(s) \) for each subterm

\[
\text{weight}(t) = \sum_{s \in \text{subterms}(t)} \text{weight}_1(s)
\]

3. get maximum of all subterms

\[
\text{weight}(t) = \max_{s \in \text{subterms}(t)} \text{weight}_1(s)
\]
Common parameters

Clause weight extension

- term weight is extended to clause weight
- \( \text{weight}(C) = \text{weight}(\{t_1 = t_2, t_3 \neq t_4, \ldots \}) = \sum_i \text{weight}(t_i) \)
- appropriately multiplied by
  - \( \gamma_{\text{maxl}} \) maximal literal multiplier
  - \( \gamma_{\text{pos}} \) positive equality multiplier
  - \( \gamma_{\text{maxt}} \) maximal term multiplier
Term: Conjecture Subterm Weight

- favor related terms
- \( \text{Term}(P, v, r, \gamma_{\text{conj}}, \delta_f, \delta_c, \delta_p, \delta_v, e, \gamma_{\text{max}t}, \gamma_{\text{max}l}, \gamma_{\text{pos}}) \)
- \( \text{weight}_1(t) = \begin{cases} \gamma_{\text{conj}} \cdot \delta & \text{iff } t \in \text{RelatedTerms} \\ \delta & \text{otherwise} \end{cases} \)

where \( \delta \in \{\delta_f, \delta_c, \delta_p, \delta_v\} \) accordingly to the top-level symbol of \( t \)
TfIdf: Conjecture Frequency Weight

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

\[
\text{tfidf}(t) = \text{tf}(t) \times \log \frac{1 + |\text{Documents}|}{1 + \text{df}(t)} \quad \text{weight}_1(t) = \frac{1}{1 + \text{tfidf}(t)}
\]
Pref: Conjecture Term Prefix Weight

- favor terms which share a prefix with a related term
- $\text{Pref}(P, v, r, \delta_{\text{match}}, \delta_{\text{miss}}, e, \gamma_{\text{max}}, \gamma_{\text{maxl}}, \gamma_{\text{pos}})$
- $\text{max-pref}(t) =$ “the longest prefix shared with RelatedTerms”
- $\text{weight}_1(t) = \delta_{\text{match}} \times |\text{max-pref}(t)| + \delta_{\text{miss}} \times (|t| - |\text{max-pref}(t)|)$
Lev: Conjecture Levenshtein Distance Weight

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

- measure the **tree edit distance** from related terms
- \( \text{Ted}(P, v, r, \delta_{ins}, \delta_{del}, \delta_{ch}, e, \gamma_{maxt}, \gamma_{maxl}, \gamma_{pos}) \)
Struc: Conjecture Structural Distance Weight

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

\[
\Delta_{\text{Struc}}(x, y) = \begin{cases} 
0 & \text{iff } x = y \\
\delta_{\text{miss}} & \text{otherwise}
\end{cases}
\]

\[
\Delta_{\text{Struc}}(x, t) = \gamma_{\text{inst}} \times |t|
\]

\[
\Delta_{\text{Struc}}(t, x) = \gamma_{\text{gen}} \times |t|
\]

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

\[
\Delta_{\text{Struc}}(t, s) = \gamma_{\text{gen}} \times |t| + \gamma_{\text{inst}} \times |s| \quad \text{(otherwise)}
\]
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 $\gamma_{\text{maxf}} = \gamma_{\text{maxl}} = \gamma_{\text{pos}} = 1.5$
- ... with a constant priority function
- ... with 5 seconds time limit
- ... manually chosen parameters
### Bests on 2078 MPTP Problems (by solved)

<table>
<thead>
<tr>
<th>heuristic</th>
<th>$\delta$</th>
<th>solved</th>
<th>speed</th>
<th>%SYM+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lev</td>
<td>$\star$-gen-1</td>
<td>841</td>
<td>2.4</td>
<td>18.3</td>
</tr>
<tr>
<td>Struc</td>
<td>$\star$-ter-1</td>
<td>833</td>
<td>3.9</td>
<td>17.2</td>
</tr>
<tr>
<td>Ted</td>
<td>$\alpha$-gen-1</td>
<td>797</td>
<td>1.2</td>
<td>12.1</td>
</tr>
<tr>
<td>Pref</td>
<td>$\alpha$-gen-$\Sigma$</td>
<td>788</td>
<td>4.0</td>
<td>10.8</td>
</tr>
<tr>
<td>Term</td>
<td>$\star$-gen-$\Sigma$</td>
<td>749</td>
<td>5.6</td>
<td>5.3</td>
</tr>
<tr>
<td>Tfldf</td>
<td>$\alpha$-gen-$\Sigma$</td>
<td>738</td>
<td>3.1</td>
<td>3.8</td>
</tr>
<tr>
<td>SYM</td>
<td></td>
<td>711</td>
<td>3.4</td>
<td>0.0</td>
</tr>
</tbody>
</table>
# Bests on 2078 MPTP Problems (by 2*EXP+)

<table>
<thead>
<tr>
<th>heuristic</th>
<th>$\delta$</th>
<th>2*EXP+</th>
<th>speed</th>
<th>%SYM+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lev</td>
<td>⋆-gen-1</td>
<td>41</td>
<td>2.4</td>
<td>18.3</td>
</tr>
<tr>
<td>Ted</td>
<td>$\alpha$-gen-1</td>
<td>33</td>
<td>1.3</td>
<td>12.1</td>
</tr>
<tr>
<td>Struc</td>
<td>⋆-sub-$\Sigma$</td>
<td>32</td>
<td>2.9</td>
<td>17.0</td>
</tr>
<tr>
<td>Pref</td>
<td>$\alpha$-gen-$\Sigma$</td>
<td>21</td>
<td>4.0</td>
<td>10.8</td>
</tr>
<tr>
<td>Term</td>
<td>$\alpha$-gen-1</td>
<td>20</td>
<td>4.4</td>
<td>-0.7</td>
</tr>
<tr>
<td>TfIdf</td>
<td>⋆-sub-$\Sigma$</td>
<td>17</td>
<td>3.5</td>
<td>0.3</td>
</tr>
<tr>
<td>SYM</td>
<td></td>
<td>7</td>
<td>3.4</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Jan Jakubův, Josef Urban

Automated Reasoning Group, Czech Technical University in Prague
Conclusions

- Lev and Struc work best
- In many cases, unified variables equal to $\alpha$-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