Clause Evaluation Heuristics for the E Prover

Jan Jakubův

Automated Reasoning Group
Czech Technical University in Prague

ARG Seminar, 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. $p(X) = \$true$, $p(X) \neq \$true$)
- 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)*
$U := \{$*clausified axioms and a negated conjecture*$\}$ *(unprocessed)*
while $(U \neq \emptyset)$ do
  if $(\emptyset \in U \cup P)$ then return *Unsatisfiable*
  $g :=$ select(U)
  $P := P \cup \{g\}$
  $U := U \setminus \{g\}$
  $U := U \cup \{$*clauses inferred with* g *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
  - $C = \{f(X) \neq a, p(a) \neq \$true\}$
  - $sym(C) = [f, X, a, p, a, \$true]$
  - $weight(C) = \sum_{s \in sym(C)} weight(s)$
- 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: $prio : Clause \rightarrow Long$
  - parametrized weight function: $weight : Clause \times args \rightarrow Double$
  - 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 $(prio(C), weight(C))$ 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 (TALK) 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))
```

New Weight Functions

- several new weight functions have been implemented
- all of them have variants which work with
  - a universal variable, that is, all variables unified (uni)
  - α-normalized variables (alf)
- all of them relate clause terms/subterms to either
  - all conjecture terms (ter)
  - all conjecture subterms (sub)
  - all conjecture subterms and "toplevel" generalizations (top) (for each conjecture symbol f consider also $f(X_1, \cdots, X_n)$)
  - all conjecture subterms and all their generalizations (gen)

Conjecture Related Term Weight

syntax and arguments

Extends `ConjectureRelativeSymbolWeight` to subterms.

```
ConjectureRelativeTermWeight(   // t...
   prio,       // priority function
   norm,       // normalization style (uni,alf)
   related,    // related terms (ter,sub,top,gen)
   rmult,      // related term multiplier
   tw,         // weight of terms
   cw,         // weight of constants
   pw,         // weight of predicates
   vw,         // weight of variables
   mtmult,     // maximal term multiplier
   mlmult,     // maximal literal multiplier
   pemult)     // positive equality multiplier
```

Conjecture Related Term Weight

weight function preliminaries

- $weight_0(X) = \mathtt{vw}$
- $weight_0(t) = \begin{cases} \mathtt{rmult} * \mathtt{w} & \text{iff } t \in \mathtt{related} \\ \mathtt{w} & \text{otherwise} \end{cases}$

  where $\mathtt{w} \in \{\mathtt{tw}, \mathtt{cw}, \mathtt{pw}\}$ accordingly to the top-level symbol of t

Conjecture Related Term Weight

weight function

- $subterms(t) = \{\text{all subterms of } t \text{ (including } t)\}$
- $weight(t) = \sum_{t_0 \in subterms(t)} weight_0(t_0)$
- $weight(C) = weight(\{t_1 = t_2, t_3 \neq t_4, \cdots\}) = \sum_i weight(t_i)$
  (appropriately multiplied by `mtmult`,`mlmult`, `pemult`)

Conjecture Related Term Weight

implementation notes

- quite efficient implementation using *shared term banks*
- computation of $weight_0(t)$ is in $O(1)$ (amortized)
- computation of $weight(t)$ is in $O(|t|)$ (amortized)

Conjecture Term Prefix Weight

syntax and arguments

Computes the longest prefix shared with some related term.

```
ConjectureTermPrefixWeight(   // p...
   prio,       // priority function
   norm,       // normalization style (uni,alf)
   related,    // related terms (ter,sub,top,gen)
   match,      // cost of symbol match
   miss        // cost of symbol mismatch
   mtmult,     // maximal term multiplier
   mlmult,     // maximal literal multiplier
   pemult)     // positive equality multiplier
```

Conjecture Term Prefix Weight

weight function

- $weight(t) = len * \texttt{match} + (|t| - len) * \texttt{miss}$
  where len is the length of the longest prefix t shares with some term from `related`
- $weight(C)$ is computed using $weight(t)$ as before

Conjecture Term Prefix Weight

implementation notes

- quite efficient implementation using *discrimination trees*
- computation of $weight(t)$ is in $O(|t|)$
- TODO: sum over subterms like in the previous heuristic

Conjecture Levenshtein Distance Weight

syntax and arguments

Computes Levenshtein distance from related terms.

```
ConjectureLevenshteinDistanceWeight(   // l...
   prio,       // priority function
   norm,       // normalization style (uni,alf)
   related,    // related terms (ter,sub,top,gen)
   ins,        // cost of insert operation
   del         // cost of delete operation
   ch,         // cost of change operation
   mtmult,     // maximal term multiplier
   mlmult,     // maximal literal multiplier
   pemult)     // positive equality multiplier
```

Conjecture Levenshtein Distance Weight

weight function

- $weight(t) = min_{t_0 \in \texttt{related}} \Delta(t, t_0)$
  where Δ is the Levenshtein distance of two terms (symbol sequences) using appropriate operation costs
- $weight(C)$ is computed using $weight(t)$ as before

Conjecture Levenshtein Distance Weight

implementation notes

- not very efficient implementation!
- time complexity of $\Delta(t, t_0)$ is in $O(|t| * |t_0|)$
- computation of $weight(t)$ is in $O(|t| * R)$
  where R is the sum of lengths of terms in `related`
- some time might be saved by caching results
- TODO: sum over subterms like in the first heuristic

Conjecture Structural Distance Weight

syntax and arguments

Computes "structural" distance from related terms.

```
ConjectureStructuralDistanceWeight(   // s...
   prio,       // priority function
   norm,       // normalization style (uni,alf)
   related,    // related terms (ter,sub,top,gen)
   vmiss,      // variable mismatch penalty
   inst        // instantiation factor
   gen,        // generalization factor
   mtmult,     // maximal term multiplier
   mlmult,     // maximal literal multiplier
   pemult)     // positive equality multiplier
```

Conjecture Structural Distance Weight

weight function

- $\Delta(X, Y) = \begin{cases} 0 & \text{iff } X = Y \\ \texttt{vmiss} & \text{otherwise} \end{cases}$
- $\Delta(X, t) = \texttt{inst} * |t|$ (instantiate X to become t)
- $\Delta(t, X) = \texttt{gen} * |t|$ (generalize t to become X)
- $\Delta(f(t\cdots), g(s\cdots)) = \texttt{gen} * |f(t\cdots)| + \texttt{inst} * |g(t\cdots)|$
- $\Delta(f(t_0\cdots), f(s_0\cdots)) = \sum_i \Delta(t_i, s_i)$
- $\mathit{weight}(t) = \mathit{min}_{t_0 \in \texttt{related}} \Delta(t, t_0)$

Conjecture Structural Distance Weight

implementation notes

- little bit better than Levenshtein
- time complexity of $\Delta(t, t_0)$ is in $O(max(|t|, |t_0|))$
- computation of $weight(t)$ is in $O(m * |\texttt{related}|)$ where m is the maximal length of a term from $\texttt{related} \cup \{t\}$
- some time might be saved by caching results
- TODO: find better case for top symbol mismatch (consider $f(a, b)$ and $g(a, b)$)
- TODO: sum over subterms like in the first heuristic

Experimental Evaluation

benchmark sets

- We use 3 benchmark sets with 5 seconds time limit:
  1. BUSHY: 2078 "bushy" problems from MTPT challenge
  2. TPTP-CNF: 5078 TPTP problems in CNF format
  3. TPTP-FOF: 7974 TPTP problems in FOF format

Experimental Evaluation

heuristic code names

- Heuristic names have shape *h* + *norm* + *rel* where
- *h* is a single letter encoding weight function
  - t for Conjecture Relative Term Weight
  - p for Conjecture Term Prefix Weight
  - l for Conjecture Levenshtein Distance Weight
  - s for Conjecture Structural Distance Weight
- *norm* denote variable normalization style (uni, alf)
- *rel* denote related terms set (ter, sub, top, gen)
- lalfgen = Levenshtein + α-normalization + generalizations
- Levenshtein's might have addition suffix for op. costs
  for example 551 means `ins`=5, `del`=5, `ch`=1

Experimental Evaluation

columns legend

- column prover: prover code name
- column solved: count of problems solved
- column %: percentage of problems solved
- %SYM+: gain on SYM in percentage of solved problems
- TALK+: how many solved problems were not solved by TALK (that is, how many "hard" problems were solved)
- SYM and TALK (slides 6 and 7) are used as references
- Note: TALK is a scheme (not directly comparable)

Top 10 on BUSHY Sorted by solved

prover	solved	%	%SYM+	TALK+
salfter	839	40.38	6.30	46
suniter	837	40.28	6.21	45
lalfgen155	834	40.13	6.06	50
lunigen155	833	40.09	6.02	50
sunisub	822	39.56	5.49	35
salfsub	820	39.46	5.39	34
lalfgen115	819	39.41	5.34	38
lalfgen151	815	39.22	5.15	34
sunitop	814	39.17	5.10	31
salftop	813	39.12	5.05	30
SYM	708	34.07	0.00	7
TALK	954	45.91	11.84	0

Top 10 on BUSHY Sorted by TALK+

prover	solved	%	%SYM+	TALK+
lalfgen155	834	40.13	6.06	50
lunigen155	833	40.09	6.02	50
salfter	839	40.38	6.30	46
suniter	837	40.28	6.21	45
lalfgen115	819	39.41	5.34	38
sunisub	822	39.56	5.49	35
salfsub	820	39.46	5.39	34
lunigen115	813	39.12	5.05	34
lalfgen151	815	39.22	5.15	34
lunigen151	807	38.84	4.76	32
SYM	708	34.07	0.00	7
TALK	954	45.91	11.84	0

Top 10 on TPTP-CNF Sorted by solved

prover	solved	%	%SYM+	TALK+
luniter511	2213	43.58	3.37	81
lalfter511	2205	43.42	3.21	80
lalftop	2199	43.30	3.09	83
lunitop	2196	43.25	3.03	88
luniter	2194	43.21	2.99	82
lunisub511	2193	43.19	2.97	78
lunisub	2193	43.19	2.97	86
tunitop	2186	43.05	2.84	46
puniter	2186	43.05	2.84	61
lalfsub511	2183	42.99	2.78	75
SYM	2042	40.21	0.00	42
TALK	2627	51.73	11.52	0

Top 10 on TPTP-CNF Sorted by TALK+

prover	solved	%	%SYM+	TALK+
sunisub	2126	41.87	1.65	101
salfsub	2129	41.93	1.71	101
punigen	2145	42.24	2.03	96
sunitop	2101	41.37	1.16	95
salftop	2097	41.30	1.08	94
salfter	2131	41.97	1.75	91
suniter	2129	41.93	1.71	90
lunitop	2196	43.25	3.03	88
luniter151	2167	42.67	2.46	88
palfgen	2132	41.99	1.77	86
SYM	2042	40.21	0.00	42
TALK	2627	51.73	11.52	0

Top 10 on TPTP-FOF Sorted by solved

prover	solved	%	%SYM+	TALK+
lalfter	3029	37.99	1.58	103
lalfsub	3018	37.85	1.44	102
lalfter151	3015	37.81	1.40	100
luniter	2984	37.42	1.02	103
lalfsub151	2982	37.40	0.99	89
lunisub	2978	37.35	0.94	103
lalfter511	2975	37.31	0.90	92
tunigen	2974	37.30	0.89	74
luniter151	2973	37.28	0.88	102
talfgen	2971	37.26	0.85	74
SYM	2903	36.41	0.00	56
TALK	3637	45.61	9.20	0

Top 10 on TPTP-FOF Sorted by TALK+

prover	solved	%	%SYM+	TALK+
salfter	2869	35.98	-0.43	116
suniter	2868	35.97	-0.44	116
salfsub	2828	35.47	-0.94	114
sunisub	2822	35.39	-1.02	113
lunisub	2978	37.35	0.94	103
lalfter	3029	37.99	1.58	103
luniter	2984	37.42	1.02	103
lalfsub	3018	37.85	1.44	102
luniter151	2973	37.28	0.88	102
salftop	2752	34.51	-1.89	101
SYM	2903	36.41	0.00	56
TALK	3637	45.61	9.20	0

Conclusions & Future Work

- term distance metrics seem to work best
- in many case unified variables equal to α-normalization (but unified variables are easier to implement)
- TODO: automated way to find reasonable parameters
- TODO: dynamic set of related terms?
- TODO: learning