Compression of Rewriting Systems for Termination Analysis

Johannes Waldmann[1]

July 12, 2012

[1] Fakultät IMN, HTWK Leipzig, Germany

Motivation

- rewriting system:
  set of pairs of terms with variables
- linear interpretation:
  mapping of function symbol f to linear function
  $[f] : (x_1, \ldots, x_k) \mapsto f_0 + f_1 \cdot x_1 + \ldots + f_k \cdot x_k$
- goal: given the interpretation, compute efficiently (e.g., using fewest multiplications) the interpretations of the set of terms occuring as lhs and rhs of rules.
- application: efficient (symbolic) computation
  $\Rightarrow$ small constraint program for the coefficients,
  $\Rightarrow$ constraint solver can solve this fast.

Example (no compression needed)

signature $\Sigma = \{a/1, b/1\}$
rewriting system $\{a(b(x)) \to b(a(x))\}$
symbolic linear interpretation (over $\mathbb{N}$)
$[a] = y \mapsto a_0 + a_1 \cdot y, [b] = y \mapsto b_0 + b_1 \cdot y$
interpretation of lhs and rhs:
$[ab] = y \mapsto a_0 + a_1 \cdot b_0 + a_1 \cdot b_1 \cdot y,$
$[ba] = y \mapsto b_0 + b_1 \cdot a_0 + b_1 \cdot a_1 \cdot y$
constraints for termination:

- monotonicity $a_0 \geq 0, a_1 \geq 1, b_0 \geq 0, b_1 \geq 1$
- compatibility $a_0 + a_1 b_0 > b_0 + b_1 a_0$, $a_1 b_1 \geq b_1 a_1$

one solution: $a_0 = 0, a_1 = 2, b_0 = 1, b_1 = 1$

Example (where compression helps)

- signature $\Sigma = \{a/1, b/1\}$
  rewriting system $\{aabb(x) \to bbbaaa(x)\}$
- naively, $3 + 5 = 8$ substitutions
  (each with 2 multiplications, 1 addition)
- should compute
  $c = aa, d = bb, \text{lhs} = cd, \text{rhs} = bdca$
  $1 + 1 + 1 + 3 = 6$ substitutions
- the constraint system has a solution in the domain $\mathbb{N}^4$ (that is, $a_1, b_1 \in \mathbb{N}^{4\times 4}$; $a_0, b_0 \in \mathbb{N}^{4\times 1}$)

A Model for Compression

- linear, straight-line (singleton) context-free tree grammar in Chomsky normal form:
  rules have the form $h(\dots) \to f(\dots, g(\dots), \dots)$
- (Larsson, Moffat (for strings) 2000;
  Lohrey, Maneth, Mennicke (for trees) 2010)
- can achieve exponential compression:
  $a^8 \Rightarrow (a^2)^4 \Rightarrow ((a^2)^2)^2$
- "$\exists G$ with $L(G) = \{t\}$ and $|G| \leq B$?" $\in$ NPc
- efficient (linear-time) approximation algorithm

The Tree Re-Pair Algorithm

construct SCFTG by repeatedly substituting maximal set of non-overlapping occurences of a pair of function symbols

$$f(\dots, g(\dots), \dots) \Rightarrow h(\dots, \dots, \dots)$$

$w = ooxoooxo$, cost (# of multiplications): 7
patterns/no-occurrences: $oo : 2, ox : 2, xo : 2$
replace $\stackrel{oo \to 1}{\longrightarrow} 1xo1xo \stackrel{xo \to 2}{\longrightarrow} 1212 \stackrel{12 \to 3}{\longrightarrow} 33.$

result: $(33, 3 \to 12, 1 \to oo, 2 \to xo)$, cost: 4

Tree Re-Pair properties

- linear time
  clever update of store of occurences of pairs
- non-deterministic
  choice of pair,
  choice of overlapping substitution
- guaranteed approximation ratio?

Tree Re-Pair Approximation Quality

$w = ooxoooxo$, cost (# of multiplications): 7
pattern occurrances: $oo : 2, ox : 2, xo : 2$
should replace $\stackrel{oo \to 1}{\longrightarrow} 1xo1xo \stackrel{xo \to 2}{\longrightarrow} 1212 \stackrel{12 \to 3}{\longrightarrow} 33$.
cost: 4

substitute "wrong" instance of oo:
$oox\underline{oo}oxo \stackrel{oo \to 1}{\longrightarrow} 1x1oxo$
no more repeated pairs.
cost: 6

cf. Charikar et al (2005): *The smallest grammar problem* (Sect. D, G) $O((n/\log n)^{2/3}) \cap \Omega(\sqrt{\log n})$

Experimental Data

- a tree re-pair algorithm is used in the termination prover Matchbox

  Endrullis, Waldmann, Zantema: *Matrix Interpretations for Proving Termination of Term Rewriting.* IJCAR06, JAR08

  as a preprocessing step
- on "average" termination problems (from TPDB), this cuts size of constraint system in half, and also the time for the solver.

  (no complete and exact measurements)
- the implementation is naive (= quadratic), cannot handle some large benchmarks
- the underlying cost function is naive (= wrong)

What is the Cost of a Tree?

ranked signature Σ, $t \in \text{Term}(\Sigma, V)$.
denote $|t|_V :=$ no. of diff. variables in t.

linear interpretation $\Sigma \to (D^* \to D)$ where $D = B^n$
$[f] : (x_1, \ldots, x_k) \mapsto f_0 + f_1 \cdot x_1 + \ldots + f_k \cdot x_k$
maps $t \in \text{Term}(\Sigma, V)$ to $|t|_V$-ary function

compute bottom-up: $\text{cost}(f(t_1, \ldots, t_k)) =$ sum of

- absolute part: k multiplications (matrix $\times$ vector), k additions (vector)
- linear coefficients: $\sum\{|t_i|_V : t_i \notin V\}$ multiplications (m. $\times$ m.), some additions (m.)

cost is dominated by (matrix $\times$ matrix) multipl. (?)

What is the Cost of a Grammar?

$=$ sum of costs of the right-hand sides of productions

... and what are the implications?

- cost of tree
  depends on (TRS) variables in subtrees
- cost change cause by (inverse) application of a production depends on the position
  example: pattern $(f, 2, g)$
  saves a lot of work in $f(x, g(h(y_1, y_2, \ldots, y_k)))$,
  but nothing in $f(x, g(h'(z)))$.

Conclusion

Observation:

- tree compression is helpful in implementations of automated termination analysis

Suggestion:

- XML compressors should be run on TPDB `http://www.termination-portal.org/wiki/TPDB` (both termination and certification benchmarks) contains huge and deeply nested trees

Work to do:

- modify tree re-pair for our cost function
- bound the approximation ratio