On Some Implementation Aspects of VMTL

Felix Schernhammer, Bernhard Gramlich
TU Wien, Austria, {felixs,gramlich}@logic.at

1 Introduction

VMTL (Vienna Modular Termination Laboratory [12]) is a program designed to (semi-) automatically prove termination of various classes of term rewriting systems (TRSs). In particular, conditional and context-sensitive TRSs (CTRSs and CSRSs) are supported. Based on the well-known dependency pair framework of [1, 7, 4], VMTL is extensible by new dependency pair processors and transformations (from (C)TRSs/CSRSs to (C)TRSs/CSRSs). An important design goal of VMTL is to make extension as easy as possible. Moreover, VMTL provides implementations of several standard and new dependency pair processors as well as of transformations (from CTRSs to CSRSs).

In this work we briefly discuss some implementation details of the DP processors of VMTL, focussing in particular on the non-standard ones. The processors currently available in VMTL are a *dependency graph processor*, used for decomposition of dependency pair problems (based on [1, Section 4.1]), a *subterm-criterion* processor, based on [9], a *reduction pair processor using recursive path orderings with status* implemented using an external SAT solver, based on [13], a *reduction pair processor using polynomial orderings* with coefficients from $\mathbb{N}$ and constants from $\mathbb{Z}$, based on [6], *narrowing and instantiation processors* based on [7, 11] and a *size change principle processor*, based on [14].

The inclusion of the (compared to the other ones) more “exotic” size-change principle processor is a tribute to the commitment of VMTL to specialize to a certain extent on the termination analysis of conditional TRSs.

The size-change principle processor is able to trace (the size of) argument terms over several function calls not necessarily comparing arguments at the same positions. This may be advantageous in the setting of TRSs obtained by transforming CTRSs, since there argument terms of auxiliary function symbols change positions frequently.

In the following we discuss some of these processors and their particular implementation in some detail. We assume familiarity with the basic notions and notations of term rewriting, context-sensitive rewriting and the dependency pair framework (cf. e.g. [5, 10, 7]).

2 Dependency Pair Processors of VMTL

2.1 Dependency Graph Processor

For the computation of the estimated dependency graph we basically use the estimation defined in [1, Section 4.1]. This estimation relies on the functions cap^{μ} and ren^{μ} that map terms to terms. $cap^{\mu}(s)$ replaces maximal active subterms of the term s, rooted by defined symbols with fresh variables. $ren^{\mu}(s)$ replaces each active variable of s by a fresh one.

It turned out that for the termination analysis of rewrite systems with a particular class of so-called *forbidden patterns* (cf. [8]) we obtain systems with rules of the form $C[l] \to C[r]$ through transformations. These rules motivate the definition of a modified notion of constructors and defined symbols, respectively.

Definition 1 (Quasi-defined symbols). *Let $\mathscr{R} = (\Sigma, R)$ be TRS. For each rule $l \to r$ (where $l \neq r$) the maximal context C is identified, such that $l = C[s]$ and $r = C[t]$ (note that C might be empty) and $root(s) \in \Sigma$. Then $root(s)$ is a* quasi-defined symbol.

We denote the set of quasi-defined symbols of a TRS (R,Σ) by D^q and the corresponding set of quasi-constructors (i.e., $\Sigma \setminus D^q$) as C^q. Observe that quasi-defined function symbols and quasi-constructors are incomparable with the usual notions of defined functions and constructors.[1] Like ordinary constructors, our modified constructors have the important property that the "top" part of a term - if it is constructed only from such constructors - cannot be modified by reduction, i.e., whenever $s = C[s_1,\dots,s_n]$ and C consists only of functions from C^q and variables, then for all terms t with $s \to^* t$ we have $t = C[t_1,\dots,t_n]$ for some terms $t_1,\dots,t_n$.

Now, we define $\widetilde{cap}^\mu$ to replace all maximal active subterms rooted by functions from D^q by fresh variables.

Note that the estimated dependency graph obtained by using $\widetilde{cap}^\mu$ is in general not comparable with the one obtained by using cap^μ (for the reason mentioned above). However, in practice the intersection of both can be used.

Example 1. *Consider the following (sub-)TRS from [8, Example 4]:*

$$\begin{array}{rr}
2nd(inf(x)) \to 2nd(x : inf(s(x))) & 2nd(x : (y : zs)) \to y \\
s(inf(x)) \to s(x : inf(s(x))) & inf(inf(x)) \to inf(x : inf(s(x))) \\
inf(x) : y \to (x : inf(s(x))) : y & (x' : inf(x)) : y \to (x' : (x : inf(s(x)))) : y
\end{array}$$

As this system is context-free, we use a replacement map μ with $\mu(f) = \{1,\dots ar(f)\}$ for all $f \in \Sigma$. Here, $: \in C^q$, while $:$ is not a constructor (in the traditional sense). Thus, consider for instance the rule (which corresponds to a dependency pair) $inf(x) : y \to (x : inf(s(x))) : y$ (i.e., $C[inf(x)] \to C[x : inf(s(x))]$)). Then $\widetilde{cap}^\mu(ren^\mu((x : inf(s(x))) : y)) = (x : z) : y$ which is not unifiable with $inf(x) : y$. Hence, there is no arc in the estimated dependency graph from this dependency pair to itself. Yet, such an arc would be present if the original cap^μ function were used instead of $\widetilde{cap}^\mu$.

2.2 Subterm Criterion Processor

The subterm-criterion processor is based on [9] and adapted for context-sensitive dependency pair problems by substituting $\rhd$ by $\rhd_\mu$. Then, the correctness of the processor is shown analogously to the proof of Theorem 11 in [9], exploiting that $\rhd_\mu$ is closed under substitutions and $\rhd_\mu \circ \to_\mu \subseteq \to_\mu \circ \rhd_\mu$.

Note that this generalization of the subterm-criterion processor has already been proposed in [2].

2.3 Narrowing Processor

VMTL's narrowing processors are based on results from [11, Section 6]. There, given a (context-sensitive) dependency pair problem $(P = \{s \to t\} \uplus P', R, \mu)$, and provided that

(1) t does not unify with the left-hand side of any other dependency pair from P,

(2) t is linear, and

(3) there are no migrating variables in $s \to t$, i.e., there are no variables which occur active in t and non-active in s,

the (forward) narrowing processor yields a new DP problem $(\overline{P}, R, \mu)$ where $\overline{P} = (P - \{s \to t\}) \cup \{s_k\theta_k \to t_k\}$ such that t_k are all the context-sensitive one-step narrowings of t_k and θ_k are the corresponding mgu's.

In fact, in some cases the set of the narrowings that need to be included in the resulting DP problem can be restricted further (see [11] for more details). Note that Condition (3) is not mentioned in [3, Theorem 5.3]. However, it is essential as the following example shows.

[1] In $\{g(f(x)) \to g(f(a))\}$, f is quasi-defined, but not defined, and g is defined, but not quasi-defined.

Example 2. *Consider the CSRS*

$$t(f(x)) \to t(h(x)) \qquad t(i(b)) \to t(f(a)) \qquad a \to b \qquad h(x) \to i(x)$$

with $\mu(t) = \mu(h) = \{1\}$ and $\mu(i) = \mu(f) = \emptyset$. The context-sensitive dependency pairs (according to the approach of [1] also used in VMTL) are

$$\begin{array}{lll} T(f(x)) \to T(h(x)) & T(i(b)) \to T(f(a)) & T(f(x)) \to H(x) \\ U(a) \to A & T(f(x)) \to U(x) & \end{array}$$

One might be tempted to narrow $T(h(x))$ to $T(i(x))$ and thus replace the first dependency pair by $T(f(x)) \to T(i(x)))$. However, while the initial DP problem is infinite (and the CSRS indeed μ-non-terminating), the transformed problem is finite, thus the transformation is unsound.

We suggest another slight variation of the narrowing processor defined above. Preconditions (1), (2) and (3) can be replaced by

(1') $ren(t)$ does not unify with the left-hand side of any dependency pair from P, and

(2') for all possible narrowing steps of t, i.e., with $t|_p\theta = l\theta$ for some rule $l \to r$, we have that t_p is linear, l matches t_p and $l \to r$ does not contain migrating variables.

This formulation has the particular advantage that it may also be applicable to non-right-linear dependency pairs.

Example 3. *Consider the example*

$$from(x) \to cons(x, cons(nil, from(s(x)))) \qquad cons(s(x), xs) \to nil$$

taken from the TPDB outermost *section. The corresponding DP problem contains a dependency pair $FROM(x) \to CONS(x, cons(nil, from(s(x))))$. According to our modified definition (one can easily see that the (linearized) right-hand side of this rule does not unify with any left-hand side of any dependency pair) the rule can be narrowed to $FROM(x) \to CONS(x, cons(nil, cons(s(x), cons(nil, from(s(s(x)))))))$. This gain in structure may be useful in a subsequent (outermost) termination analysis.*

2.4 Instantiation Processor

In VMTL we use a modified version of the instantiation processors proposed in [7], because the use of our processors turned out to be fruitful in the termination analysis of CSRSs obtained by transforming CTRSs. In particular, we do not only allow propagation of constructors, but also of defined symbols under certain circumstances (cf. [11, Definitions 12 and 13] for details).

Given a DP problem $(P = \{s \to t\} \uplus P', R, \mu)$ and provided that

- all subterms of s containing variables are rooted by constructor symbols,
- $s \to t$ does not contain migrating variables and
- all $l \to r \in P$ where $cap^{\mu}(ren^{\mu}(r))$ unifies with s (this subset of P is denoted by $Pred_{s \to t}(P)$), satisfy that $s\sigma = r$ for some substitution σ,

the (backward) instantiation processor yields a new dependency pair problem $(P' \cup \{s\sigma \to t\sigma \mid l \to r \in Pred_{s \to t}(P) \wedge r = s\sigma\}, R, \mu)$.

Example 4. *Consider a DP Problem given by*

$$P = \left\{ \begin{array}{ccc} F(x) & \rightarrow & G(x) \\ G(b) & \rightarrow & F(a) \end{array} \right. \qquad R = \left\{ \begin{array}{ccc} f(x) & \rightarrow & g(x) \\ g(b) & \rightarrow & f(a) \\ a & \rightarrow & b \end{array} \right.$$

and $\mu(f) = \{1, \dots ar(f)\}$ *for all* $f \in \Sigma$. $Pred_{F(x)\rightarrow G(a)}(P)$ *is* $\{G(b) \rightarrow F(a)\}$ *and* $F(x)$ *matches* $F(a)$. *Hence, the instantiation processor yields a new DP problem* $(\{F(a) \rightarrow G(a), G(b) \rightarrow F(a)\}, R, \mu)$.

See e.g., [11, Examples 10, 11 and 12] for more complex examples where the application of the instantiation processor is shown.

3 Summary and Conclusion

We have discussed some details and particularities of the dependency pair processors in VMTL. Note that at the time of writing not all proposed features are already present in the publicly available version of VMTL. A more comprehensive presentation of the VMTL system itself can be found in [12]. VMTL is available as stand-alone command-line executable (as a java jar file) and through a webinterface at the VMTL homepage (`http://www.logic.at/vmtl/`). There, also more details regarding extensibility and benchmarks of VMTL on the set of TRSs from the TPDB in various categories can be found.

References

[1] B. Alarcón, F. Emmes, C. Fuhs, J. Giesl, R. Gutiérrez, S. Lucas, P. Schneider-Kamp, and R. Thiemann. Improving context-sensitive dependency pairs. In: I. Cervesato, H. Veith, and A. Voronkov, eds., Proc. LPAR'08, Doha, Qatar, November 22–27, 2008, LNCS 5330, pp. 636–651. Springer, Nov. 2008.

[2] B. Alarcón, R. Gutiérrez, and S. Lucas. Context-sensitive dependency pairs. In: S. Arun-Kumar and N. Garg, eds., Proc. (FST&TCS'06), Kolkata, India, Dec. 13–15, 2006, LNCS 4337, pp. 297–308, Springer, 2006.

[3] B. Alarcón, R. Gutiérrez, and S. Lucas. Improving the Context-Sensitive Dependency Graph. *Electronic Notes in Theoretical Computer Science*, 188:91–103, 2007.

[4] T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236(1-2):133–178, 2000.

[5] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[6] C. Fuhs, J. Giesl, A. Middeldorp, P. Schneider-Kamp, R. Thiemann, and H. Zankl. SAT solving for termination analysis with polynomial interpretations. In: Proc. SAT'07, LNCS 4501, pp. 340-354. Springer, 2007.

[7] J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Mechanizing and improving dependency pairs. *Journal of Automated Reasoning*, 37(3):155–203, 2006.

[8] B. Gramlich and F. Schernhammer. Extending context-sensitivity in term rewriting. Draft version (submitted for publication), available at `http://www.logic.at/people/schernhammer/`, 2009.

[9] N. Hirokawa and A. Middeldorp. Dependency pairs revisited. In: V. van Oostrom, ed., Proc. RTA'04, Aachen, Germany, June 3 – June 5, 2004, LNCS 3091, pp. 249-268, Springer, June 2004.

[10] S. Lucas. Context-sensitive computations in functional and functional logic programs. *Journal of Functional and Logic Programming*, 1998(1), Jan. 1998. The MIT Press.

[11] F. Schernhammer and B. Gramlich. Characterizing and proving operational termination of deterministic conditional term rewriting systems. Technical Report E1852-2009-01, TU Wien, Austria, Mar. 2009.

[12] F. Schernhammer and B. Gramlich. VMTL – A modular termination laboratory. In: R. Treinen, ed., Proc. RTA'09, Brasilia, Brazil, June 29 – July 1, 2009, LNCS, Springer, June 2009. To appear.

[13] P. Schneider-Kamp, R. Thiemann, E. Annov, M. Codish, and J. Giesl. Proving termination using recursive path orders and SAT solving. In: Proc. FroCoS'07, LNCS 4720, pp. 267–282, Springer, 2007.

[14] R. Thiemann and J. Giesl. Size-change termination for term rewriting. In: R. Nieuwenhuis, ed., RTA'03, Valencia, Spain, June 9 – June 11, 2003, LNCS 2706, pp. 264-278, Springer, June 2003.