A Satisfiability Encoding of Dependency Pair Techniques for Maximal Completion*

Haruhiko Sato[1] and Sarah Winkler[2]

1 **Graduate School of Information Science and Technology, Hokkaido University, Sapporo, Japan**
2 **Institute of Computer Science, University of Innsbruck, Innsbruck, Austria**

Abstract

We present a general approach to encode termination in the dependency pair framework as a satisfiability problem, and include encodings of dependency graph and reduction pair processors. We use our encodings to increase the power of the completion tool Maxcomp.

1 Introduction

Maximal completion [4] is a simple yet efficient Knuth-Bendix completion approach which relies on MaxSAT solving. It can thus only compute convergent term rewrite systems (TRSs) whose termination can be expressed by satisfiability constraints.

Encoding termination techniques for TRSs via satisfiability problems has become common practice. However, to the best of our knowledge all previous encodings restrict to a single termination technique such as a specific reduction order or interpretations into a particular domain. Hence the maximal completion tool Maxcomp was so far restricted to LPO and KBO, and could not handle input problems such as CGE_2 [6], which describes two commuting group endomorphisms as given by the following set of equations $\mathcal{E}$:

$$\begin{array}{lll} \mathsf{e} \cdot x \approx x & \mathsf{f}(x \cdot y) \approx \mathsf{f}(x) \cdot \mathsf{f}(y) & x \cdot (y \cdot z) \approx (x \cdot y) \cdot z \\ \mathsf{i}(x) \cdot x \approx \mathsf{e} & \mathsf{g}(x \cdot y) \approx \mathsf{g}(x) \cdot \mathsf{g}(y) & \mathsf{f}(x) \cdot \mathsf{g}(y) \approx \mathsf{g}(y) \cdot \mathsf{f}(x) \end{array}$$

In this paper we present a uniform layout for an SMT encoding of compound termination strategies that combine different techniques from the dependency pair framework. We give encodings of dependency pairs, a rule removal processor, and two versions of dependency graph approximations. We implemented our encodings on top of Maxcomp. Our experimental results show that this allows Maxcomp to complete problems like CGE_2, and boosts its power beyond simple termination.

2 Preliminaries

We assume familiarity with term rewriting [1]. Knuth-Bendix completion aims to transform an equational system (ES) $\mathcal{E}$ into a TRS $\mathcal{R}$ which is convergent for $\mathcal{E}$, i.e., terminating, confluent and equivalent to $\mathcal{E}$. We write $\mathrm{CP}(\mathcal{R})$ for the set of critical pairs of a TRS $\mathcal{R}$, and $\downarrow_{\mathcal{R}}$ for $\to_{\mathcal{R}}^* \cdot {}_{\mathcal{R}}^*\!\leftarrow$. Maximal completion is a simple completion approach based on MaxSAT solving. For an input ES $\mathcal{E}$, it tries to compute $\varphi(\mathcal{E})$ where φ is defined as

$$\varphi(\mathcal{C}) = \begin{cases} \mathcal{R} & \text{if } \mathcal{E} \cup \mathrm{CP}(\mathcal{R}) \subseteq {\downarrow_{\mathcal{R}}} \text{ for some } \mathcal{R} \in \mathfrak{R}(\mathcal{C}) \\ \varphi(\mathcal{C} \cup S(\mathcal{C})) & \text{otherwise} \end{cases}$$

$\mathfrak{R}(\mathcal{C})$ consists of terminating TRSs $\mathcal{R}$ such that $\mathcal{R} \subseteq \mathcal{C} \cup \mathcal{C}^{-1}$, and $S(\mathcal{C}) \subseteq \bigcup_{\mathcal{R} \in \mathfrak{R}(\mathcal{C})} \mathrm{CP}(\mathcal{R})$.

* This research was supported by the Austrian Science Fund project I963.

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

▶ **Theorem 1** ([4]). *The TRS $\varphi(\mathcal{E})$ is convergent for $\mathcal{E}$ if it is defined.*

In the maximal completion tool Maxcomp, $\mathfrak{R}(\mathcal{C})$ is computed by maximizing the number of satisfied clauses in $\bigvee_{s\approx t\in\mathcal{C}}[s > t] \vee [t > s]$, subject to the side constraints implied by the SAT/SMT encoding $[\cdot > \cdot]$ of some reduction order $>$.

In this paper we use the dependency pair (DP) framework to show termination of TRSs [3]. A DP problem is a pair of two TRSs $(\mathcal{P}, \mathcal{R})$, it is finite if it does not admit an infinite chain. A DP processor Proc is a function which maps a DP problem to either a set of DP problems or "no". It is sound if a DP problem d is finite whenever $\mathrm{Proc}(d) = \{d_1, \ldots, d_n\}$ and all of d_i are finite.

For an ES $\mathcal{C}$, we define the set of *dependency pair candidates* $\mathrm{DPC}(\mathcal{C})$ as all rules $\ell^\# \to u^\#$ such that $\ell \approx r \in \mathcal{C}$, $\ell \to r$ is a rewrite rule, and $r \trianglerighteq u$ but $\ell \not\trianglerighteq u$.

3 Encodings

We first illustrate the idea of our encodings by means of an example.

▶ **Example 2.** Suppose we want to orient a maximal number of equations from the ES $\mathcal{E}$ given in the introduction, where termination is to be shown by computing dependency pairs, applying a reduction pair processor based on a polynomial interpretation and finally a reduction pair processor based on LPO with argument filterings.

Let $\mathcal{P} = \mathrm{DPC}(\mathcal{E} \cup \mathcal{E}^{-1})$. For all equations $s \approx t$ in $\mathcal{C} = \mathcal{E} \cup \mathcal{E}^{-1} \cup \mathcal{P}$ we use *strict* variables $S^i_{s\to t}$ as well as *weak* variables $W^i_{s\to t}$ for all $0 \le i \le 3$. Moreover, boolean variables X^{def}_f encode whether f is a defined symbol. We maximize the number of satisfied clauses in the disjunction $\bigvee_{s\approx t\in\mathcal{E}} S^0_{s\to t} \vee S^0_{t\to s}$ subject to the following constraints:

$$\bigwedge_{s\approx t\in\mathcal{E}\cup\mathcal{E}^{-1}} S^0_{s\to t} \to (W^1_{s\to t} \wedge X^{\mathrm{def}}_{\mathsf{root}(s)} \wedge \bigwedge_{\ell\to r\in\mathrm{DPC}(s\to t)} X^{\mathrm{def}}_{\mathsf{root}(r)} \to S^1_{\ell\to r}) \tag{a}$$

$$\bigwedge_{i=2}^{3} \bigwedge_{\ell\to r\in\mathcal{P}} (S^{i-1}_{\ell\to r} \to [\ell \geqslant^i r]) \wedge (\neg[\ell >^i r] \to S^i_{\ell\to r}) \tag{b}$$

$$\bigwedge_{i=2}^{3} \bigwedge_{s\approx t\in\mathcal{E}\cup\mathcal{E}^{-1}} W^{i-1}_{s\to t} \to (W^i_{s\to t} \wedge [s \geqslant^i t]) \tag{c}$$

$$\bigwedge_{\ell\to r\in\mathcal{P}} \neg S^3_{\ell\to r} \tag{d}$$

Clauses (a) trigger DPs and 'move' rules to the weak component, (b) expresses that if a DP is not oriented it remains to be considered, (c) requires rules to be weakly oriented, and (d) demands that finally no DP remains unoriented. Here $[\ell >^i r]$ ($[\ell \geqslant^i r]$) refers to strict (weak) orientation constraints imposed by polynomial interpretations for $i = 2$ and LPO with argument filterings for $i = 3$.

The following paragraphs transfer standard notions of the DP framework to our satisfiability setting. A *DP problem encoding* is a tuple $\mathcal{D} = (\mathcal{S}, \mathcal{W}, \phi)$ consisting of two sets of boolean variables $\mathcal{S} = \{S_{\ell\to r} \mid \ell \to r \in \mathcal{P}\}$ and $\mathcal{W} = \{W_{\ell\to r} \mid \ell \to r \in \mathcal{R}\}$ for TRSs $\mathcal{P}$ and $\mathcal{R}$, and a formula ϕ. An assignment α is *finite for* a DP problem encoding $\mathcal{D} = (\mathcal{S}, \mathcal{W}, \phi)$ if $\alpha(\phi) = \top$ and the DP problem $(\mathcal{P}^{\mathcal{S}}_\alpha, \mathcal{R}^{\mathcal{W}}_\alpha)$ given by the TRSs

$$\mathcal{P}^{\mathcal{S}}_\alpha = \{\ell \to r \mid S_{\ell\to r} \in \mathcal{S},\ \alpha(S_{\ell\to r}) = \top\} \quad \mathcal{R}^{\mathcal{W}}_\alpha = \{\ell \to r \mid W_{\ell\to r} \in \mathcal{W},\ \alpha(W_{\ell\to r}) = \top\}$$

is finite. A *DP processor encoding* Proc maps a DP problem encoding $\mathcal{D} = (\mathcal{S}, \mathcal{W}, \phi)$ to a finite set of DP problem encodings $\mathrm{Proc}(\mathcal{D}) = \{\mathcal{D}_1, \dots, \mathcal{D}_n\}$. A DP processor encoding Proc is *sound* if for any $\mathcal{D}$ such that $\mathrm{Proc}(\mathcal{D}) = \{\mathcal{D}_1, \dots, \mathcal{D}_n\}$ and any assignment α that is finite for all $\mathcal{D}_i$, it also holds that α is finite for $\mathcal{D}$.

For an ES $\mathcal{C}$ its set of *initial variables* is $\mathcal{I}_\mathcal{C} = \{I_{\ell\to r} \mid \ell \approx r \in \mathcal{C}\}$.

▶ **Definition 3.** For an ES $\mathcal{C}$ with initial variables $\mathcal{I}_\mathcal{C}$ the *initial DP problem encoding* is given by $\mathcal{D}_\mathcal{C} = (\mathcal{S}, \mathcal{W}, \phi)$ where $\mathcal{S} = \{S_{\ell\to r} \mid \ell \to r \in \mathrm{DPC}(\mathcal{C})\}$, $\mathcal{W} = \{W_{\ell\to r} \mid \ell \approx r \in \mathcal{C}\}$ and

$$\phi = \bigwedge_{\ell\approx r\in\mathcal{C}} I_{\ell\to r} \to \left(W_{\ell\to r} \wedge X^{\mathrm{def}}_{\mathsf{root}(\ell)} \wedge \bigwedge_{s\to t\in \mathrm{DPC}(\ell\to r)} X^{\mathrm{def}}_{\mathsf{root}(t)} \to S_{s\to t} \right)$$

▶ **Lemma 4.** *Let $\mathcal{C}$ be an ES. Suppose there is a tree whose nodes are labelled with DP problem encodings satisfying the following conditions:*

- *The root is labelled with the initial DP problem encoding $\mathcal{D}_\mathcal{C}$.*
- *For every non-leaf node labelled $\mathcal{D}$ with n children labelled $\mathcal{D}_1, \dots, \mathcal{D}_n$ there is a sound processor encoding* Proc *such that* $\mathrm{Proc}(\mathcal{D}) = \{\mathcal{D}_1, \dots, \mathcal{D}_n\}$.

Let the leaves be labelled $\{(\mathcal{S}_i, \mathcal{W}_i, \phi_i) \mid 1 \le i \le k\}$. If the formula

$$\phi = \bigwedge_{i=1}^{k} \phi_i \wedge \bigwedge_{s\to t\in\mathcal{S}_i} \neg S_{s\to t}$$

is satisfied by an assignment α then the TRS $\mathcal{R} = \{\ell \to r \mid \alpha(I_{\ell\to r}) = \top\}$ is terminating.

Proof. By induction on the tree structure, α is finite for all DP problem encodings occurring as labels. Termination of $\mathcal{R}$ follows from finiteness of α for the root label $\mathcal{D}_\mathcal{C}$. ◀

▶ **Definition 5** (Reduction pair processor)**.** Let $(>, \geqslant)$ be a reduction pair and π an argument filtering, with satisfiability encodings $[\cdot \geqslant_\pi \cdot]$ and $[\cdot >_\pi \cdot]$

A DP problem encoding $(\mathcal{S}, \mathcal{W}, \phi)$ is mapped to $\{(\mathcal{S}', \mathcal{W}', \phi \wedge T_S \wedge T_W)\}$ where $\mathcal{S}' = \{S'_{\ell\to r} \mid S_{\ell\to r} \in \mathcal{S}\}$, $\mathcal{W}' = \{W'_{\ell\to r} \mid W_{\ell\to r} \in \mathcal{W}\}$, and

$$T_S = \bigwedge_{S_{\ell\to r}\in\mathcal{S}} S_{\ell\to r} \to [\ell \geqslant_\pi r] \wedge (\neg[\ell >_\pi r] \to S'_{\ell\to r})$$

$$T_W = \bigwedge_{W_{\ell\to r}\in\mathcal{W}} W_{\ell\to r} \to W'_{\ell\to r} \wedge [\ell \geqslant_\pi r]$$

Concrete encodings $[\cdot \geqslant_\pi \cdot]$ and $[\cdot >_\pi \cdot]$ for LPO/RPO, KBO as well as reduction orders given by polynomial and matrix interpretations—also in combination with argument filterings and usable rules—are well-studied, see for instance [5, 9, 2, 8].

Note that Definition 5 can easily be modified to admit rule removal by setting

$$T_W = \bigwedge_{W_{\ell\to r}\in\mathcal{W}} W_{\ell\to r} \to [\ell \geqslant_\pi r] \wedge (\neg[\ell >_\pi r] \to W'_{\ell\to r})$$

▶ **Definition 6** (Dependency graph processor)**.** A DP problem encoding $(\mathcal{S}, \mathcal{W}, \phi)$ is mapped to the set $\{(\mathcal{S}', \mathcal{W}', \psi)\}$ such that $\mathcal{S}' = \{S'_{\ell\to r} \mid S_{\ell\to r} \in \mathcal{S}\}$, $\mathcal{W}' = \{W'_{\ell\to r} \mid S_{\ell\to r} \in \mathcal{S}\} \cup \{W'_{\ell\to r} \mid W_{\ell\to r} \in \mathcal{W}\}$, and $\psi = \phi \wedge T_S \wedge T_W$ where

$$T_S = \bigwedge_{S_{p_1}, S_{p_2}\in\mathcal{S}} S_{p_1} \wedge S_{p_2} \wedge [p_1 \xrightarrow{\mathsf{edge}} p_2] \wedge \neg S'_{p_1} \wedge \neg S'_{p_2} \to X^{\mathrm{w}}_{p_1} > X^{\mathrm{w}}_{p_2}$$

$$T_W = \left(\bigwedge_{S_{\ell\to r}\in\mathcal{S}} S_{\ell\to r} \to W'_{\ell\to r} \right) \wedge \left(\bigwedge_{W_{\ell\to r}\in\mathcal{W}} W_{\ell\to r} \to W'_{\ell\to r} \right)$$

Here T_S encodes cycle analysis of the graph in the sense that a cycle $p_1 \to p_2 \to \cdots \to p_n \to p_1$ issues the unsatisfiable constraint $X^{\mathrm{w}}_{p_1} > X^{\mathrm{w}}_{p_2} > \cdots > X^{\mathrm{w}}_{p_n} > X^{\mathrm{w}}_{p_1}$. For the formula $[s \to t \xrightarrow{\mathsf{edge}} u \to v]$ encoding the presence of an edge from $s \to t$ to $u \to v$ one can simply use $\top$ if $\mathsf{root}(t) = \mathsf{root}(u)$ and $\bot$ otherwise. (We also experimented with an encoding in terms of the unifiability between $\mathsf{REN}(\mathsf{CAP}(t))$ and u, but due to reasons of space do not presented here.)

The above encoding does not allow to use different orderings in SCCs, in contrast to what is commonly done in termination provers. However, it can be modified to consider SCCs by mapping a problem encoding to k independent problem encodings.

▶ **Definition 7** (Dependency graph processor with k SCCs)**.** A DP problem encoding $\mathcal{D} = (\mathcal{S}, \mathcal{W}, \phi)$ is mapped to $\{\mathcal{D}_i\}_{1 \le i \le k} = \{(\mathcal{S}_i, \mathcal{W}_i, \psi_i)\}_{1 \le i \le k}$ where $\mathcal{S}_i = \{S_{i,\ell \to r} \mid S_{\ell \to r} \in \mathcal{S}\}$, $\mathcal{W}_i = \{W_{i,\ell \to r} \mid S_{\ell \to r} \in \mathcal{S}\} \cup \{W_{i,\ell \to r} \mid W_{\ell \to r} \in \mathcal{W}\}$, $\psi_i = \phi \wedge T_{\mathrm{scc}}(k) \wedge T_S(i) \wedge T_W(i)$, and

$$T_{\mathrm{scc}}(k) = \bigwedge_{S_p \in \mathcal{S}} 1 \le X^{\mathrm{scc}}_p \le k \wedge \bigwedge_{S_{p_1}, S_{p_2} \in \mathcal{S}} S_{p_1} \wedge S_{p_2} \wedge [p_1 \xrightarrow{\mathsf{edge}} p_2] \to X^{\mathrm{edge}}_{p_1,p_2} \wedge X^{\mathrm{scc}}_{p_1} \ge X^{\mathrm{scc}}_{p_2}$$

$$T_S(i) = \bigwedge_{S_{p_1}, S_{p_2} \in \mathcal{S}} X^{\mathrm{edge}}_{p_1,p_2} \wedge X^{\mathrm{scc}}_{p_1} = i \wedge X^{\mathrm{scc}}_{p_2} = i \wedge \neg S_{i,p_1} \wedge \neg S_{i,p_2} \to X^{\mathrm{w}}_{p_1} > X^{\mathrm{w}}_{p_2}$$

$$T_W(i) = \bigwedge_{W_p \in \mathcal{W}} W_p \to W_{i,p} \wedge \bigwedge_{S_p \in \mathcal{S}} S_p \wedge X^{\mathrm{scc}}_p = i \wedge \left(\bigvee_{S_{p'} \in \mathcal{S} \setminus \{S_p\}} X^{\mathrm{scc}}_p = X^{\mathrm{scc}}_{p'} \right) \to W_{i,p}$$

Here $X^{\mathrm{edge}}_{p_1,p_2}$ is a boolean variable encoding the presence of both DPs p_1 and p_2 as well as an edge from p_1 to p_2, and X^{scc}_p is an integer variable assigning an SCC number to a DP p. Hence $T_{\mathrm{scc}}(k)$ encodes the separation of the graph into at most k SCCs, and $T_S(i), T_W(i)$ encode conditions to orient the ith SCC.

Soundness of all the above encodings can be shown by relating them to their processor counterparts [3], but we omit the proofs here due to lack of space.

4 Experiments

We implemented our DP framework encoding in Maxcomp as described in Section 3. Besides enhancing the previous LPO and KBO implementations with argument filterings, we also added a (restricted version of) linear polynomial interpretations as reduction pair processors. Both versions of the DG processors were included as well.

Table 1 summarizes our experimental results[1] for the test bed comprising 115 equational systems from the distribution of mkbTT [7]. Each ES was given a time limit of 180 seconds, timeouts are marked ∞. Row (1) corresponds to the original Maxcomp using LPO. In setting (2) we use a strategy combining dependency pairs with reduction pair processors applying linear polynomials and LPO. Setting (3) enhances setting (2) with a simple DG processor encoding according to Definition 6, and setting (4) uses Definition 7 with 2 SCCs instead. A simple heuristic is applied by the *automatic mode* (5): one iteration is run with plain LPO and setting (2) in parallel, but afterwards only one strategy (which can orient more of the initial equations) is kept. The column # lists the number of successful completions, the next column gives the average time for a successful completion.

[1] Details available from `http://cl-informatik.uibk.ac.at/software/maxcompdp`

	method	#	avg. time	CGE_2	proofreduction	equiv_proofs
(1)	Maxcomp	85	3.8	∞	∞	∞
(2)	DPs, poly, LPO	83	14.7	6.4	∞	1.6
(3)	DG, poly, LPO	40	2.8	∞	∞	∞
(4)	DG/2SCCs, poly, LPO	25	2.5	∞	∞	∞
(5)	auto	92	10.2	5.6	135.1	1.5

Table 1 Experimental Results.

With the relatively lightweight DP strategy (2) we successfully complete the problems mentioned in Table 1, which cannot be completed using plain LPO or KBO. However, some other systems are lost, compared to Maxcomp using LPO. Typically, these problems require many iterations and/or give rise to many equations. Thus in total (2) completes not quite as many systems as (1), and the average time is tripled. Settings (3) and (4) require considerably more encoding effort and hence succeed on comparatively few systems. For instance, only proving termination of the convergent (unreduced) TRS for equiv_proofs (74 rules) produced in a completion run with setting (2) takes 1.4 seconds for strategy (2) (12K variables, 45K clauses) but 176 seconds with setting (3) (290K variables, 1.2M clauses). Overall the automatic mode turned out to be most powerful since it can often be efficient by applying LPO, but also switch to a more sophisticated strategy in case of unorientable equations. There are even some problems like proofreduction where (5) succeeds but (2) does not—apparently it can be preferable to apply LPO in the beginning before switching to the DP strategy.

References

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

2 J. Endrullis, J. Waldmann, and H. Zantema. Matrix interpretations for proving termination of term rewriting. *JAR*, 40(2-3):195–220, 2008.

3 J. Giesl, R. Thiemann, and P. Schneider-Kamp. The dependency pair framework: Combining techniques for automated termination proofs. In *LPAR*, volume 3452 of *LNCS*, pages 301–331, 2005.

4 D. Klein and N. Hirokawa. Maximal completion. In *RTA*, volume 10 of *LIPIcs*, pages 71–80, 2011.

5 P. Schneider-Kamp, R. Thiemann, E. Annov, M. Codish, and J. Giesl. Proving termination using recursive path orders and SAT solving. In *FroCoS*, volume 4720 of *LNCS (LNAI)*, pages 267–282, 2007.

6 A. Stump and B. Löchner. Knuth-Bendix completion of theories of commuting group endomorphisms. *IPL*, 98(5):195–198, 2006.

7 S. Winkler, H. Sato, A. Middeldorp, and M. Kurihara. Multi-completion with termination tools. *JAR*, 50(3):317–354, 2013.

8 H. Zankl, N. Hirokawa, and A. Middeldorp. Constraints for argument filterings. In *SOFSEM*, volume 4362 of *LNCS*, pages 579–590, 2007.

9 H. Zankl, N. Hirokawa, and A. Middeldorp. KBO orientability. *JAR*, 43(2):173–201, 2009.