Encoding Dependency Pair Techniques and Control Strategies for Maximal Completion*

Haruhiko Sato and Sarah Winkler

Graduate School of Information Science and Technology, Hokkaido University, Sapporo, Japan
Institute of Computer Science, University of Innsbruck, Innsbruck, Austria
{haru@complex.ist.hokudai.ac.jp,sarah.winkler@uibk.ac.at}

Abstract. This paper describes two advancements of SAT-based Knuth-Bendix completion as implemented in Maxcomp. (1) Termination techniques using the dependency pair framework are encoded as satisfiability problems, including dependency graph and reduction pair processors. (2) Instead of relying on pure maximal completion, different SAT-encoded control strategies are exploited.
Experiments show that these developments let Maxcomp improve over other automatic completion tools, and produce novel complete systems.

Keywords: term rewriting, completion, SAT encoding, dependency pairs

1 Introduction

Recently, some impressive progress was been achieved by exploiting SAT/SMT solvers in theorem proving [6]. Maximal completion is a simple yet highly efficient Knuth-Bendix completion approach which relies on MaxSAT solving [5]. It is hence inherently limited to compute complete term rewrite systems (TRSs) whose termination can be expressed as a SAT problem. The maximal completion tool Maxcomp restricts to LPO and KBO, which naturally narrows the range of possible completions. For instance, in the following presentation of CGE_2 the last equation cannot be oriented:

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

In general, Maxcomp cannot complete CGE problems, which describe commuting group endomorphisms as occurring in the theory of uninterpreted functions [10]. Another potential limitation of Maxcomp is given by the fact that its exclusive search strategy is to orient as many equations as possible.

This paper presents two advancements of Maxcomp. (1) Our abstract framework for SMT encodings allows to first switch from a termination problem to a dependency pair (DP) problem and subsequently apply an arbitrary sequential combination of dependency pair processors. We give encodings for different

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

estimations of the dependency graph (DG), and show how to apply reduction pair processors in this context. Though encoding termination of a TRS as a satisfiability problem has become common practice, to the best of our knowledge all previous encodings restrict to a specific reduction order or interpretations into a particular domain. (2) The original version of Maxcomp always tried to generate a complete TRS by orienting as many equations as possible. However, this control strategy is not always optimal to guide the proof search. We devised satisfiability encodings for a number of alternative control strategies, and compared them experimentally.

Our results show that these enhancements allow Maxcomp to not only complete CGE problems but in general improve over previous automatic completion tools. Though we described preliminary results on DP encodings in [7], our recent work on control strategies greatly enhanced the tool's power and scalability.

The remainder of this paper is structured as follows. Section 2 collects some preliminaries before our encodings for dependency pair techniques are outlined in Section 3. Section 4 presents the developed control strategies. Some further implementation issues are described in Section 5, and experimental results are presented in Section 6.

2 Preliminaries

We assume familiarity with term rewriting [1]. Knuth-Bendix completion aims to transform an equational system (ES) $\mathcal{E}_0$ into a TRS $\mathcal{R}$ which is complete for $\mathcal{E}_0$, i.e., terminating, confluent and equivalent to $\mathcal{E}_0$. The set of critical pairs $\mathrm{CP}(\ell_1 \to r_1, \ell_2 \to r_2)$ denotes all equations $\ell_2\sigma[r_1\sigma]_p \approx r_2\sigma$ such that p is a function symbol position in ℓ_2, $\ell_2|_p$ and ℓ_1 are unifiable with mgu σ, and if $p = \epsilon$ then the two rules are not variants. We write $\mathrm{CP}(\mathcal{R})$ for the set of critical pairs among rules from a TRS $\mathcal{R}$. The relation $\downarrow_{\mathcal{R}}$ denotes $\to_{\mathcal{R}}^* \cdot {}_{\mathcal{R}}^*\!\leftarrow$. We also write $(s \approx t)\downarrow_{\mathcal{R}}$ for an equation $s' \approx t'$ such that s' and t' are some $\mathcal{R}$-normal forms of s and t, respectively, and mean the natural extension to sets of equations $\mathcal{E}$ when writing $\mathcal{E}\downarrow_{\mathcal{R}}$. For an ES $\mathcal{E}$ we write $\widetilde{\mathcal{E}}$ to denote the set of all equations $\ell \approx r$ such that $\ell \approx r \in \mathcal{E} \cup \mathcal{E}^{-1}$ and $\ell \to r$ is a valid rewrite rule.

Maximal completion is a simple completion approach based on MaxSAT solving. For an input ES $\mathcal{E}_0$, it tries to compute $\Phi_{\mathcal{E}_0}(\mathcal{E}_0)$ as follows:

Definition 1. *Let $\mathcal{E}$ be a fixed ES. For any ES $\mathcal{C}$, $\Phi_{\mathcal{E}}$ is defined by*

$$\Phi_{\mathcal{E}}(\mathcal{C}) = \begin{cases} \mathcal{R} & \text{if } \mathcal{E} \cup \mathrm{CP}(\mathcal{R}) \subseteq \downarrow_{\mathcal{R}} \text{ for some } \mathcal{R} \in \mathfrak{R}(\mathcal{C}) \\ \Phi_{\mathcal{E}}(\mathcal{C} \cup S(\mathcal{C})) & \text{otherwise} \end{cases} \tag{1}$$

where $\mathfrak{R}(\mathcal{C})$ consists of terminating TRSs $\mathcal{R}$ such that $\mathcal{R} \subseteq \widetilde{\mathcal{C}}$, and $S(\mathcal{C}) \subseteq \leftrightarrow_{\mathcal{C}}^*$.

Theorem 1 ([5]). *The TRS $\Phi_{\mathcal{E}_0}(\mathcal{E}_0)$ is complete for $\mathcal{E}_0$ 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

$>$, and $S(\mathcal{C})$ is a subset of $\bigcup_{\mathcal{R}\in\mathfrak{R}(\mathcal{C})} (\mathrm{CP}(\mathcal{R}) \cup \mathcal{E}_0)\downarrow_{\mathcal{R}}$.

In this paper we use the dependency pair (DP) framework to show termination of TRSs [4]. 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 $\mathcal{C}$, let the set of dependency pair candidates $\mathrm{DPC}(\mathcal{C})$ be all rules $F(t_1, \ldots, t_n) \to G(u_1, \ldots, u_n)$ such that $f(t_1, \ldots, t_n) \approx r \in \widetilde{\mathcal{C}}$, $r \trianglerighteq g(u_1, \ldots, u_n)$ but $\ell \not\trianglerighteq g(u_1, \ldots, u_n)$, and F, G are fresh function symbols.

3 Encodings

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

Example 1. Suppose one wants to encode termination according to the following strategy in three stages: first dependency pairs are computed, then two reduction pair processors based on the reduction pairs $(>^{\mathsf{poly}}, \geqslant^{\mathsf{poly}})$ and $(>^{\mathsf{lpo}}, \geqslant^{\mathsf{lpo}})$ are applied, using monotonic polynomial interpretations and LPO, respectively. Let $[\cdot >^{\mathsf{poly}} \cdot]$ denote a SAT encoding of $>^{\mathsf{poly}}$, and similar for the other relations.

Suppose that the current set of equations contains a potential rule $\alpha\colon \mathsf{aa} \to \mathsf{ba}$. (To enhance readability we here use string notation and write aa to denote the term $\mathsf{a}(\mathsf{a}(x))$, etc.) Note that this rule gives rise to the dependency pair $\beta\colon \mathsf{Aa} \to \mathsf{Ba}$ if b is a defined symbol. Rule α gives rise to the following constraints:

$$S^0_\alpha \to W^1_\alpha \wedge X^{\mathrm{def}}_{\mathsf{a}} \wedge (X^{\mathrm{def}}_{\mathsf{b}} \to S^1_\beta) \tag{a}$$

$$S^1_\beta \to [\mathsf{Aa} \geqslant^{\mathsf{poly}} \mathsf{Ba}] \wedge (\neg[\mathsf{Aa} >^{\mathsf{poly}} \mathsf{Ba}] \to S^2_\beta) \tag{b}$$

$$S^2_\beta \to [\mathsf{Aa} \geqslant^{\mathsf{lpo}} \mathsf{Ba}] \wedge (\neg[\mathsf{Aa} >^{\mathsf{lpo}} \mathsf{Ba}] \to S^3_\beta)$$

$$W^1_\alpha \to [\mathsf{aa} \geqslant^{\mathsf{poly}} \mathsf{ba}] \wedge (\neg[\mathsf{aa} >^{\mathsf{poly}} \mathsf{ab}] \to W^2_\alpha) \tag{c}$$

$$W^2_\alpha \to [\mathsf{aa} \geqslant^{\mathsf{lpo}} \mathsf{ba}]$$

$$\neg S^3_\beta \tag{d}$$

Here the boolean variables $S^0_\alpha, W^1_\alpha, \ldots, W^3_\alpha, S^1_\beta, \ldots, S^3_\beta$ express strict/weak orientation of α and β in different proof stages, and $X^{\mathrm{def}}_{\mathsf{a}}$, $X^{\mathrm{def}}_{\mathsf{b}}$ express whether a and b are defined. The constraint (a) triggers the DP β and "moves" rule α to the weak component. Constraint (b) expresses that if the DP β is not oriented it remains to be considered, both for stage 2 and 3. Constraint (c) ensures that rule α is weakly oriented. Since monotonic polynomial interpretations allow for rule removal, α can be removed after stage 2 if it was strictly oriented. Finally, (d) demands that the DP β needs no consideration after stage 3.

The following paragraphs transfer standard notions of the DP framework to our satisfiability setting.

Definition 2. *A* DP problem encoding *is a tuple* $\mathcal{D} = (\mathcal{S}, \mathcal{W}, \varphi)$ *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*. We call an assignment* α finite for *a DP problem encoding* $(\mathcal{S}, \mathcal{W}, \varphi)$ *if* $\alpha(\varphi) = \top$ *and the DP problem* $(\mathcal{P}^{\mathcal{S}}_{\alpha}, \mathcal{R}^{\mathcal{W}}_{\alpha})$ *is finite, given by the TRSs* $\mathcal{P}^{\mathcal{S}}_{\alpha} = \{\ell \to r \mid S_{\ell\to r} \in \mathcal{S},\ \alpha(S_{\ell\to r}) = \top\}$ *and* $\mathcal{R}^{\mathcal{W}}_{\alpha} = \{\ell \to r \mid W_{\ell\to r} \in \mathcal{W},\ \alpha(W_{\ell\to r}) = \top\}$.

Definition 3. *A* DP processor encoding Proc *maps a DP problem encoding* $\mathcal{D} = (\mathcal{S}, \mathcal{W}, \varphi)$ *to a finite set of DP problem encodings* $\mathrm{Proc}(\mathcal{D}) = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$. *The encoding* Proc *is* sound *if for any* $\mathcal{D}$ *such that* $\mathrm{Proc}(\mathcal{D}) = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ *and any assignment* α *that is finite for all* $\mathcal{D}_i$*, it also holds that* α *is finite for* $\mathcal{D}$.

Definition 4. *The set of* initial variables *for an ES* $\mathcal{C}$ *is* $\mathcal{I}_{\mathcal{C}} = \{I_{\ell\to r} \mid \ell \approx r \in \widetilde{\mathcal{C}}\}$. *For an ES* $\mathcal{C}$ *the* initial DP problem encoding *is given by* $\mathcal{D}_{\mathcal{C}} = (\mathcal{S}, \mathcal{W}, \varphi)$ *where* $\mathcal{S} = \{S_{\ell\to r} \mid \ell \to r \in \mathrm{DPC}(\mathcal{C})\}$, $\mathcal{W} = \{W_{\ell\to r} \mid \ell \approx r \in \widetilde{\mathcal{C}}\}$ *and*

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

Lemma 1. *Let* $\mathcal{C}$ *be an ES. Suppose there is a tree whose nodes are DP problem encodings satisfying the following conditions:*

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

Let the leaves be $\{(\mathcal{S}_i, \mathcal{W}_i, \varphi_i) \mid 1 \le i \le k\}$. *If an assignment* α *satisfies*

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

then the TRS $\mathcal{R} = \{\ell \to r \mid \ell \approx r \in \widetilde{\mathcal{C}},\ \alpha(I_{\ell\to r}) = \top\}$ *is terminating.*

Proof. By induction on the tree structure, α is finite for all nodes. Termination of $\mathcal{R}$ follows from finiteness of α for the root $\mathcal{D}_{\mathcal{C}}$. □

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 $(\mathcal{S}, \mathcal{W}, \varphi)$ *is mapped to* $\{(\mathcal{S}', \mathcal{W}', \varphi \wedge T_S \wedge T_W)\}$ *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*

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

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

Here all boolean variables in $\mathcal{S}'$ and $\mathcal{W}'$ are assumed to be fresh. 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 with argument filterings and usable rules—are well-studied, see for instance [8, 15, 3, 14, 13].

Note that Definition 5 can easily be modified to admit rule removal by adding clauses $(\neg[\ell >_\pi r] \to W'_{\ell\to r})$ to the conjunction definiting T_W, similar as for T_S.

Definition 6 (Dependency graph processor). *A DP problem encoding of the form* $(\mathcal{S}, \mathcal{W}, \varphi)$ *is mapped to* $\{(\mathcal{S}', \mathcal{W}', \psi)\}$ *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* $\psi = \varphi \wedge T_S \wedge T_W$ *where*

$$T_S = \bigwedge_{S_p, S_{p'} \in \mathcal{S}} S_p \wedge S_{p'} \wedge [p \Rightarrow p'] \wedge \neg S'_p \wedge \neg S'_{p'} \to X^{\mathrm{w}}_p > X^{\mathrm{w}}_{p'}$$

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

Here T_S encodes cycle analysis of the graph in the sense that a cycle $p_1 \Rightarrow p_2 \Rightarrow \cdots \Rightarrow p_n \Rightarrow p_1$ 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 $[s \to t \Rightarrow u \to v]$ 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 present it 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* $\mathcal{D} = (\mathcal{S}, \mathcal{W}, \varphi)$ *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* $\mathcal{S}_i = \{S_{i,\ell\to r} \mid S_{\ell\to r} \in \mathcal{S}\}$, $\mathcal{W}_i = \{W_{i,\ell\to r} \mid S_{\ell\to r} \in \mathcal{S}\} \cup \{W_{i,\ell\to r} \mid W_{\ell\to r} \in \mathcal{W}\}$, $\psi_i = \varphi \wedge T_{\mathrm{scc}}(k) \wedge T_S(i) \wedge T_W(i)$, *and*

$$T_{\mathrm{scc}}(k) = \bigwedge_{S_p \in \mathcal{S}} 1 \le X^{\mathrm{scc}}_p \le k \wedge \bigwedge_{S_p, S_{p'} \in \mathcal{S}} S_p \wedge S_{p'} \wedge [p \Rightarrow p'] \to X^{\Rightarrow}_{p,p'} \wedge X^{\mathrm{scc}}_p \ge X^{\mathrm{scc}}_{p'}$$

$$T_S(i) = \bigwedge_{S_p, S_{p'} \in \mathcal{S}} X^{\Rightarrow}_{p,p'} \wedge X^{\mathrm{scc}}_p = i \wedge X^{\mathrm{scc}}_{p'} = i \wedge \neg S_{i,p} \wedge \neg S_{i,p'} \to X^{\mathrm{w}}_p > X^{\mathrm{w}}_{p'}$$

$$T_W(i) = \bigwedge_{S_p \in \mathcal{S}} S_p \wedge X^{\mathrm{scc}}_p = i \to W_{i,p} \wedge \bigwedge_{W_p \in \mathcal{W}} W_p \to W_{i,p}$$

Here $X^{\Rightarrow}_{p_1,p_2}$ 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 [4], but we omit the proofs here due to lack of space.

4 Control Strategies

In its original version, Maxcomp generated terminating TRSs $\mathfrak{R}(\mathcal{C})$ by orienting as many equations in $\mathcal{C}$ as possible. This was motivated by the following observation: whenever a TRS $\mathcal{R}$ is complete for $\mathcal{E}_0$, then any terminating TRS $\mathcal{R}'$ satisfying $\mathcal{R} \subseteq \mathcal{R}' \subseteq \leftrightarrow^*_{\mathcal{E}_0}$ is complete for $\mathcal{E}_0$ as well. However, this choice

of $\mathfrak{R}(\mathcal{C})$ has drawbacks in the case where the selected TRS is *not* yet complete: In case of multiple possibilities, the search is not guided towards "more useful" TRSs. Moreover, the chosen TRSs are large such that critical pair generation and normalization tend to be inefficient.

We therefore experimented with different components of control strategies which can be combined in a variety of ways. The following desirable properties of TRSs $\mathcal{R} \in \mathfrak{R}(\mathcal{C})$ constitute the basis of the below definitions, reflecting the aim to eventually derive a complete TRS for the axioms $\mathcal{E}_0$.

(1) All nontrivial equations in $\mathcal{C}$ should be reducible by $\mathcal{R}$.
(2) The axioms $\mathcal{E}_0$ should be derivable from $\mathcal{R}$.
(3) Preferably, the critical pairs of $\mathcal{R}$ should be joinable.

We use sets of constraints cs and mc, where constraints in cs have to be always satisfied, whereas the number of satisfied constraints from mc is to be maximized. To determine cs and mc, the following options c and mc are considered:

$$\mathsf{c} ::= \mathsf{Red} \mid \mathsf{Comp} \qquad \mathsf{mc} ::= \mathsf{None} \mid \mathsf{MaxRed} \mid \mathsf{CPRed} \mid \mathsf{Oriented} \mid \mathsf{NotOriented}$$

Here Red ensures property (1) by demanding that $\varphi_{red}(\mathcal{C})$ is satisfied.

$$\varphi_{red}(\mathcal{C}) = \bigwedge_{s \approx t \in \mathcal{C}} \varphi_{red}(s \approx t) \quad \varphi_{red}(s \approx t) = \bigvee \{ I_{\ell \to r} \mid \ell \to r \in \widetilde{\mathcal{C}} \text{ reduces } s \text{ or } t \}$$

Option Comp ensures property (2) by demanding that $\varphi_{comp}(\mathcal{C})$ is satisfied. To that end, every equation $\ell \approx r \in \mathcal{C}$ is associated with a fresh boolean variable $E_{\ell \approx r}$ and a fresh integer variable $w_{\ell \approx r}$.

$$\varphi_{comp}(\mathcal{C}) = \bigwedge_{\ell \approx r \in \mathcal{E}_0} E_{\ell \approx r} \wedge \bigwedge_{\ell \approx r \in \mathcal{C}} E_{\ell \approx r} \to \ell = r \vee I_{\ell \to r} \vee I_{r \to \ell} \vee \varphi_{\leftrightarrow}(\ell \approx r)$$

$$\varphi_{\leftrightarrow}(\ell \approx r) = \bigvee_{\mathcal{E} \in \mathfrak{D}_{\ell \approx r}} \bigwedge_{e' \in \mathcal{E}} E_{e'} \wedge w_{\ell \approx r} > w_{e'}$$

Here $\mathfrak{D}_{\ell \approx r}$ consists of ESs $\mathcal{E} \subseteq \mathcal{C}$ satisfying $\ell \leftrightarrow_{\mathcal{E}}^* r$, and $w_{\ell \approx r}$ avoids cyclic dependencies among equations by requiring that equations are only derived from equations associated with smaller values. Suitable sets $\mathfrak{D}_e$ can be collected when rewriting equations: Whenever an equation e is simplified to e' using rules $\mathcal{R}$, $\{e'\} \cup \mathcal{R}$ is added to $\mathfrak{D}_e$, and $\{e\} \cup \mathcal{R}$ is added to $\mathfrak{D}_{e'}$.

Concerning options for mc, None requires nothing, MaxRed maximizes the number of clauses $\varphi_{red}(s \approx t)$ for $s \approx t \in \mathcal{C}$, Oriented maximizes the number of oriented equations in $\mathcal{C}$, and NotOriented maximizes the number of unoriented equations in $\mathcal{C}$. The option CPRed tries to reduce as many critical pairs of $\mathcal{R} \in \mathfrak{R}(\mathcal{C})$ as possible by maximizing the number of satisfied clauses in $\varphi_{CPred}(\mathcal{C})$. Here $\mathcal{R}(\mathcal{C})$ consists of all rewrite rules $\ell \to r$ such that $\ell \approx r \in \widetilde{\mathcal{C}}$.

$$\varphi_{CPred}(\mathcal{C}) = \{ I_{r_1} \wedge I_{r_2} \to \varphi_{red}(s \approx t) \mid r_1, r_2 \in \mathcal{R}(\mathcal{C}) \text{ and } s \approx t \in \mathrm{CP}(r_1, r_2) \}$$

5 Implementation

We next describe some further implementation details of our extension of Maxcomp, which will in the sequel be referred to as MaxcompDP. The general layout

```
function maxcomp(C, S)
  R(C), S' := max_k(C, S, k)
  for all R ∈ R(C)
    if CP(R) ∪ E0 ⊆↓R then R
    else C := C ∪ select(n, (CP(R) ∪ C)↓R)
  maxcomp(C, S')
```

Fig. 1. Main control function.

of Maxcomp based on the control loop shown in Figure 1 was kept. As input parameters, the function `maxcomp` obtains a set of equations $\mathcal{C}$ and an overall strategy $\mathcal{S}$ (described below). The ES $\mathcal{C}$ is initialized with $\mathcal{E}_0$. In each recursive call, the `max_k` function tries to find k terminating TRSs $\mathfrak{R}(\mathcal{C})$ according to the strategy $\mathcal{S}$, and it returns a possibly modified strategy $\mathcal{S}'$. For each $\mathcal{R} \in \mathfrak{R}(\mathcal{C})$, if $\mathcal{R}$ is confluent and joins $\mathcal{E}_0$ then `maxcomp` succeeds; otherwise n new equations are selected and added to $\mathcal{C}$. In order to find $\mathfrak{R}(\mathcal{C})$, `max_k` uses (MAX)SAT calls to Yices [2]. In some important aspects MaxcompDP deviates from Maxcomp, the next paragraphs describe these changes.

Termination Strategies. Termination of TRSs $\mathfrak{R}(\mathcal{C})$ is encoded according to a certain termination strategy. Besides LPO and KBO as used in Maxcomp, MaxcompDP now also provides as base orders simple linear polynomial interpretations of the shape $x_1+\ldots+x_n+c$ and the instance MPOL of the weighted path order [13]. We implemented the DP techniques presented in Section 3, and also support argument filterings for LPO and KBO. Overall, termination strategies can thus be constructed according to the following grammar:

$$\mathsf{o} ::= \mathsf{LPO} \mid \mathsf{KBO} \mid \mathsf{MPol} \mid \mathsf{LPol} \qquad \mathsf{t} ::= \mathsf{o}s \mid \mathsf{DP}(\mathsf{o}s) \mid \mathsf{DG}(\mathsf{o}s) \mid \mathsf{DGk}(\mathit{int}, \mathsf{o}s)$$

where $\mathsf{o}s$ abbreviates $\mathsf{o}\ \mathit{list}$, which is interpreted as lexicographic combination of the associated reduction orders/reduction pairs. DP switches to a DP problem using Definition 4, while DG and DGk use DG encodings according to Definitions 6 and 7, respectively. In the sequel we consider the strategies $t_{lpo} := [\mathsf{LPO}]$, $t_{dp} := \mathsf{DP}([\mathsf{LPol}, \mathsf{LPO}])$, $t_{dg} := \mathsf{DG}([\mathsf{LPol}, \mathsf{LPO}])$ and $t_{dg2} := \mathsf{DGk}(2, [\mathsf{LPol}, \mathsf{LPO}])$.

Overall Strategies. An overall strategy $\mathcal{S}$ for the `max_k` function combines termination and control strategies and has the type $(\mathsf{t}, \mathsf{c}\ \mathit{set}, \mathsf{mc})\ \mathit{list}$. Such a strategy is used as follows: If $\mathcal{S}$ is the empty list, `max_k` fails. If $\mathcal{S}$ is a nonempty list $(t, cs, mc) :: \mathcal{S}'$, `max_k` tries to find a TRS $\mathcal{R}_i$ by satisfying the constraints t and cs, and maximizing the satisfied constraints of mc, such that $\mathcal{R}_i$ is different from $\mathcal{R}_1, \ldots, \mathcal{R}_{i-1}$. If `max_k` can find k TRSs in this way, it returns $\mathcal{R}_1, \ldots, \mathcal{R}_k$ and $\mathcal{S}$. If it fails to do so for $\mathcal{R}_i$, it tries to find the remaining TRSs using strategy $\mathcal{S}'$, and returns $\mathcal{S}'$. This allows to change to a more appropriate termination and/or control strategy if the current one can, e.g., not orient

sufficiently many equations. We experimented with different strategies such as $S_{red} := [(t, \{\mathsf{Red}\})]$, $S_{comp} := [(t, \{\mathsf{Red}, \mathsf{Comp}\})]$, $S_{CPred} := [(t, \{\mathsf{Red}\}, \mathsf{CPRed})]$, $S_{maxcomp} := [(t, \varnothing, \mathsf{Oriented})]$, $S_{notOriented} := [(t, \{\mathsf{Red}, \mathsf{Comp}\}, \mathsf{NotOriented})]$ for different termination strategies t. Here we write (t, cs) for (t, cs, None). The strategy $S_{maxcomp}$ corresponds to the original Maxcomp approach. For a termination strategy t, $s_{full}(t)$ denotes $(t, \{\mathsf{Red}, \mathsf{Comp}\}, \mathsf{CPRed})$, and $S_{full}(t)$ denotes $[s_{full}(t)]$. The S_{auto} strategy turned out particularly useful, it is defined by $[s_{full}(t_{lpo}),\ s_{full}(t_{dp}),\ (t_{lpo}, \{\mathsf{Comp}\}, \mathsf{MaxRed})]$.

The number k has considerable impact; MaxcompDP lets the user control it by an input parameter. By default, $k = 6$ in the first two recursive calls and $k = 2$ afterwards. The rationale behind this choice is that considering a wide variety of orientations in the beginning of a run reduces the risk of getting stuck with an initial, possibly unfortunate orientation.

Selection of New Equations. The `select` function in Figure 1 plays the role of $S(\mathcal{C})$ from Definition 1. Maxcomp by default selected up to 7 equations of size at most 20 from the set $\mathrm{CP}(\mathcal{R})\downarrow_{\mathcal{R}}$ (since it is practically infeasible to add all critical pairs). In contrast, MaxcompDP does not only add critical pairs of a TRS $\mathcal{R}$ to $\mathcal{C}$ but also reduced equations. Therefore the n smallest equations from the set $(\mathrm{CP}(\mathcal{R}) \cup \mathcal{C})\downarrow_{\mathcal{R}}$ are selected, without inducing a size bound. The number n can be controlled by the user, by default $n = 12$.

Incremental Termination Checks. The formulas obtained with our termination encodings easily grow large. However, though in the course of a completion run many satisfiability checks are required, the termination constraint issued for a specific rule does not change. We hence use Yices in an incremental way: whenever a new equation $\ell \approx r$ gives rise to a potential rule $\ell \to r$, its termination constraint $[\ell > r]$ is computed and added to the context of Yices. To find a terminating TRS, we temporarily add the constraints cs and mc according to the current control strategy but backtrack after the SAT check. This allows to use the same Yices context throughout the completion run, and issue termination constraints only once per rule (though new constraints need to be computed if the termination strategy changes).

MaxcompDP as well as all experimental results are available from

`http://cl-informatik.uibk.ac.at/software/maxcompdp`

6 Experiments

Table 1 summarizes our experimental results for the test bed comprising 115 equational systems from the distribution of mkbTT [12], run on a system equipped with an Intel Core i7 with four cores of 2.1GHz each and 7.5 GB of memory. Each ES was given a time limit of 600 seconds, timeouts are marked ∞. The rows labeled (1)–(4) correspond to MaxcompDP using S_{full} with different termination strategies, and (5) applies the automatic mode S_{auto} as described in

Section 5. Rows (DP1)–(DP5) use t_{dp} within different control strategies, and (LPO1) combines t_{lpo} with $S_{maxcomp}$. All runs used the default values for k and n. Finally, we compare with other completion tools that are automatic in that no reduction order is required as input, namely Maxcomp, mkbTT, KBCV [9], and Slothrop [11].

Column # lists the number of successful completions, the next column gives the average time for a completion in seconds. Columns (a)–(d) show the results for some selected systems, namely CGE_2, CGE_5, proofreduction, and equiv_proofs.

		#	avg. time	(a)	(b)	(c)	(d)
(1)	$S_{full}(t_{lpo})$	81	2.2	∞	∞	∞	∞
(2)	$S_{full}(t_{dp})$	89	33.5	17.1	79.5	5.2	3.1
(3)	$S_{full}(t_{dg})$	86	37.3	18.5	155.5	5.7	3.1
(4)	$S_{full}(t_{dg2})$	89	41.0	12.3	254.0	13.2	6.5
(5)	S_{auto}	97	11.6	4.1	104.4	3.6	1.5
(DP1)	$S_{maxcomp}$	56	4.8	157.7	∞	7.5	3.9
(DP2)	S_{red}	81	31.8	568.4	∞	9.8	2.1
(DP3)	S_{comp}	87	45.9	15.5	302.7	3.4	2.2
(DP4)	S_{CPred}	90	29.7	17.1	273.5	9.2	3.4
(DP5)	$S_{notOriented}$	85	15.9	3.6	15.9	20.6	3.8
(LPO1)	$S_{maxcomp}$	77	13.5	∞	∞	∞	∞
	Maxcomp	87	3.8	∞	∞	∞	∞
	mkbTT	85	40.1	33.5	∞	7.3	237.9
	KBCV	88	12.4	∞	∞	∞	∞
	Slothrop	76	65.8	∞	∞	209.4	12.1

Table 1. Experimental Results.

The DP strategy (2) allows to successfully complete problems (a)–(d), which cannot be completed using LPO or KBO. However, some other systems are lost, compared to the setting using LPO. Typically, these problems require many iterations and/or give rise to many equations. Also, the average time compared to (1) is multiplied. Settings (3) and (4) require more encoding effort such that completion takes even more time than for setting (2). However, the tradeoff between the more complex encoding and the gain in power turns out more beneficial for setting (4), which can complete the same number of problems as (2) but more than (3). Overall S_{auto} proved 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.

Concerning control strategies, a comparison of (1) with (LPO1) and (2) with (DP1) suggests that $S_{maxcomp}$ is by far more suited for plain reduction orders than for complex DP strategies. One reason for that might be that powerful DP strategies can orient more equations such that $S_{maxcomp}$ gives rise to even

larger TRSs. But we also observed that $S_{maxcomp}$ with DPs prefers unfortunate orientations in presence of group theory, which occurs in many problems.

All of S_{red}, S_{comp} and S_{CPred} positively influence the number of completed systems (though at the price of lower efficiency). In the S_{auto} setting, their combination was most successful.

As Table 1 shows, MaxcompDP with strategy S_{auto} can complete more systems than any other automatic completion tool, although the tools are incomparable in the sense that for each tool there is an ES that it can complete, but no other tools can. It manages to complete CGE_5 in 104.4 seconds, whereas for mkbTT it was a major effort requiring more than 35000 seconds. Moreover, MaxcompDP can also complete CGE_6 and CGE_7 in 307 and 362 seconds, respectively (the latter using $n = 18$, though). No other tool could complete these ESs so far.

References

1. F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, Cambridge, 1998.
2. B. Dutertre and L.M. de Moura. A fast linear-arithmetic solver for DPLL(T). In *CAV*, volume 4144 of *LNCS*, pages 81–94, 2006.
3. J. Endrullis, J. Waldmann, and H. Zantema. Matrix interpretations for proving termination of term rewriting. *JAR*, 40(2-3):195–220, 2008.
4. 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.
5. D. Klein and N. Hirokawa. Maximal completion. In *RTA*, volume 10 of *LIPIcs*, pages 71–80, 2011.
6. K. Korovin. Inst-Gen—a modular approach to instantiation-based automated reasoning. In *Programming Logics*, pages 239–270, 2013.
7. H. Sato and S. Winkler. A satisfiability encoding of dependency pair techniques for maximal completion. In *WST*, 2014.
8. 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.
9. T. Sternagel and H. Zankl. KBCV—Knuth-Bendix completion visualizer. In *IJCAR*, volume 7364 of *LNCS*, pages 530–536, 2012.
10. A. Stump and B. Löchner. Knuth-Bendix completion of theories of commuting group endomorphisms. *IPL*, 98(5):195–198, 2006.
11. I. Wehrman, A. Stump, and E.M. Westbrook. Slothrop: Knuth-Bendix completion with a modern termination checker. In *RTA*, volume 4098 of *LNCS*, pages 287–296, 2006.
12. S. Winkler, H. Sato, A. Middeldorp, and M. Kurihara. Multi-completion with termination tools. *JAR*, 50(3):317–354, 2013.
13. A. Yamada, K. Kusakari, and T. Sakabe. A unified ordering for termination proving. *Science of Computer Programming*, 2014. To appear.
14. H. Zankl, N. Hirokawa, and A. Middeldorp. Constraints for argument filterings. In *SOFSEM*, volume 4362 of *LNCS*, pages 579–590, 2007.
15. H. Zankl, N. Hirokawa, and A. Middeldorp. KBO orientability. *JAR*, 43(2):173–201, 2009.