Equational Reasoning for Termination of Rewriting*

Harald Zankl and Aart Middeldorp
Institute of Computer Science
University of Innsbruck
6020 Innsbruck, Austria
{harald.zankl,aart.middeldorp}@uibk.ac.at

Abstract

In this note we suggest to use equational reasoning for obtaining finer approximations of dependency graphs.

1 Introduction

The dependency graph is an essential ingredient to make the dependency pair framework modular and powerful. This graph is not computable in general but sound approximations exist [1, 5, 7, 8, 10]. In this note we propose a new method based on equational reasoning to further improve existing approximations. We first recall the basic concepts of the dependency pair framework in Section 2. Then the relationship to equational reasoning is sketched in Section 3 before experimental results are reported in Section 4. Related ideas and possible future work is discussed in Section 5.

2 Dependency Pair Framework

We assume familiarity with term rewriting [2, 11] and the termination competition.[1] Next we recall the key concepts that are essential for the remainder of this note.

Definition 1. The nodes of the *dependency graph* $\mathsf{DG}(\mathcal{R})$ of a TRS $\mathcal{R}$ are the dependency pairs of $\mathcal{R}$ and there is an edge from node $s \to t$ to node $u \to v$ if there exist substitutions σ and τ such that $t\sigma \to_{\mathcal{R}}^* u\tau$.

To make use of this definition a corresponding DP processor is formulated.

Theorem 2. *The following processor [6, 7] is sound and complete. For a DP problem $(\mathcal{P}, \mathcal{R})$ the processor returns $\{(\mathcal{P}_1, \mathcal{R}), \ldots, (\mathcal{P}_n, \mathcal{R})\}$ where $\mathcal{P}_1, \ldots, \mathcal{P}_n$ are the SCCs of $\mathsf{DG}(\mathcal{R})$.* □

The condition in Definition 1 is not decidable but sound approximations exist [1, 5, 7, 8, 10]. In the sequel we show that Waldmeister [4] allows a finer approximation of the condition from Definition 1.

3 Equational Reasoning

Next we sketch how Waldmeister can be used to get finer approximations of the dependency graph: The existence of substitutions σ and τ such that for two terms t and u with $t\sigma \to_{\mathcal{R}}^* u\tau$ clearly implies $t\sigma \leftrightarrow_{\mathcal{R}}^* u\tau$. The latter condition can be tested with Waldmeister.

Let $\mathcal{E}$ denote the set of equations resulting from removing the orientation of rules in $\mathcal{R}$. To test an edge $(s \to t, u \to v)$ from $\mathsf{DG}(\mathcal{R})$, the input for Waldmeister consists of $\mathcal{E}$ and the equation (called conclusion) $t = u$. The theorem prover performs a variant of ordered completion [3] until it succeeds in proving or refuting $t\sigma \leftrightarrow_{\mathcal{R}}^* u\tau$. Thus an edge $(s \to t, u \to v)$ in the dependency graph can be removed if Waldmeister manages to refute $t\sigma \leftrightarrow_{\mathcal{R}}^* u\tau$. In all other cases (Waldmeister proves the claim or does not terminate) nothing can be said about the edge.

*This research is supported by FWF (Austrian Science Fund) project P18763.

[1] http://termcomp.uibk.ac.at

```
NAME         various11
MODE         PROOF
SORTS        ANY
SIGNATURE    g: ANY ANY -> ANY
             1: -> ANY
             0: -> ANY
             h: ANY -> ANY
             F: ANY ANY ANY -> ANY
             f: ANY ANY ANY -> ANY
ORDERING     LPO g > 1 > 0 > h > F > f
VARIABLES    z,w,y,x : ANY
EQUATIONS    f(0, 1, x) = f(h(x), h(x), x)
             h(0) = 0
             h(g(x, y)) = y
             F(x, x, x) = F(x, x, x)
CONCLUSION   F(h(z), h(z), z) = F(0, 1, w)
```

Listing 1: Input to Waldmeister

```
Initial equations:
------------------

(    1)   x1 = h(g(x2,x1))
(    2)   h(0) = 0
(    3)   f(h(x1),h(x1),x1) = f(0,1,x1)
(    4)   F(x1,x1,x1) = F(x1,x1,x1)
(    5)   eq(x1,x1) = true
(    6)   eq(F(h(x1),h(x1),x1),F(0,1,x2)) = false

Goals:
------

(    1)   true ?= false
          using narrowing to prove F(h(x1),h(x1),x1) ?= F(0,1,x2)

Detected structure: Orkus
********************************************************************************
****************************** COMPLETION - PROOF ******************************
********************************************************************************
new rule:                 1   F(h(x1),h(x1),x1) ? F(0,1,x2)
new rule:                 2   h(g(x1,x2)) -> x2
new rule:                 3   F(x1,x1,g(x2,x1)) ? F(0,1,x3)
new rule:                 4   h(0) -> 0
new rule:                 5   F(0,0,0) ? F(0,1,x1)
new rule:                 6   f(h(x1),h(x1),x1) -> f(0,1,x1)
new rule:                 7   eq(x1,x1) -> true
new rule:                 8   f(0,0,0) -> f(0,1,0)
new equation:             1   f(0,1,g(x1,x2)) = f(x2,x2,g(x1,x2))

Refuted Goals:
No.   1: true ?= false, current true ?= false
          using narrowing to prove F(h(x1),h(x1),x1) ?= F(0,1,x2)

1 goal was specified, which was refuted.

Waldmeister states: System completed.
```

Listing 2: Output of Waldmeister

Example 3. Consider the TRS various/11[2] consisting of the following three rules

$$\begin{aligned} \mathsf{f}(0,1,x) &\to \mathsf{f}(\mathsf{h}(x),\mathsf{h}(x),x) \\ \mathsf{h}(0) &\to 0 \\ \mathsf{h}(\mathsf{g}(x,y)) &\to y \end{aligned}$$

admitting the dependency pairs

$$\begin{aligned} \mathsf{F}(0,1,x) &\to \mathsf{F}(\mathsf{h}(x),\mathsf{h}(x),x) \\ \mathsf{F}(0,1,x) &\to \mathsf{H}(x) \end{aligned}$$

Typical (combinations of) dependency graph approximations [1, 5, 7] (sometimes referred to as edg***) cannot figure out that no instance of $\mathsf{F}(\mathsf{h}(x),\mathsf{h}(x),x)$ rewrites to $\mathsf{F}(0,1,x)$ whereas our approach can. Thus this system can successfully be proved terminating due to an empty dependency graph. The input for above edge to Waldmeister is depicted in Listing 1 most of which is self explanatory. We stress some facts that we learned during our experiments:

- Although Waldmeister is used in its automatic mode, one has to specify a precedence for LPO which is ignored.
- The root function symbols from the **CONCLUSION** must appear among the equations to get consistent results. This explains the peculiar looking $\mathsf{F}(x,x,x) = \mathsf{F}(x,x,x)$.
- Waldmeister does not terminate quite frequently. Thus we impose a time limit of 0.1 seconds per edge.

When calling Waldmeister on the input from Listing 1 it produces the refutation shown in Listing 2.

4 Experimental Results

We integrated our approach into T$_\mathsf{T}$T$_2$ [9] with the help of Waldmeister. In our tests we considered the 1391 TRSs from the Termination Problems Data Base version 5.0. All tests have been performed on a server equipped with eight dual-core AMD Opteron® processors 885 running at a clock rate of 2.6 GHz and on 64 GB of main memory. A time limit of 5 seconds was enforced. Thus the configuration of T$_\mathsf{T}$T$_2$ and the test environment is similar to the latest termination competitions.

For Table 1 we compare the existing approximation (edg***) with our approach (wdg) in different settings. Here wdg just tests edges that could not be removed by edg***. The column none indicates that no additional DP processors are employed and for competition the refinement is integrated within the strategy used by T$_\mathsf{T}$T$_2$ in the 2008 edition of the termination competition. The numbers in the table indicate how many systems could be proved terminating. With edg*** 60 systems can be shown terminating due to a lack of SCCs in the approximated dependency graph. This number can be increased to 71 if additionally wdg is applied. Most remarkably this simple method allows to prove two systems (secret06/tpa01 and various/11) which T$_\mathsf{T}$T$_2$ could not show terminating in the November 2008 competition. These systems have been solved by other tools (AProVE, Jambox, and TPA) before, based on semantic labeling and/or narrowing. The results in the right part of Table 1 do not look so convincing at first since the number is worse for wdg. The reason is due to the small timeout of 5 seconds in T$_\mathsf{T}$T$_2$'s competition mode. Currently we lose three systems (two from the TRCSR directory and various/14)

[2] Systems from the Termination Problems Data Base are written in sans – serif font.

Table 1: Experiments for 1391 TRSs.

	none	competition
edg***	60	779
wdg	71	778

when additionally allowing wdg which could be handled (closely) within 5 seconds beforehand. Since wdg has to call Waldmeister for each single edge in the dependency graph one should not forget about the additional costs. However, we anticipate that further experimentation will provide the necessary insights to build an efficient strategy including wdg.

5 Related and Future Work

Apart from the known approximations for dependency graphs [1, 5, 7, 10] recently some more refinements evolved. Korp and Middeldorp [8] use tree automata techniques to remove edges from the dependency graph. However, they cannot solve systems like various/11 since their approach must first linearize right-hand sides of rules. Zankl and Middeldorp [12] combined (increasing) interpretations with cycle analysis but are also unable to solve Example 3.

Concerning future work we plan to investigate whether other tools than Waldmeister can be used for our needs. Furthermore we want to extend our approach to innermost termination. Here the main benefit is that one can restrict the input to Waldmeister to the usable rules of $\mathcal{R}$ instead of $\mathcal{R}$.

References

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

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

[3] L. Bachmair, N. Dershowitz, and D. Plaisted. *Completion without Failure*, chapter 1 of volume II in Resolution of Equations in Algebraic Structures, pages 1–30. Academic Press, 1989.

[4] J.-M. Gaillourdet, T. Hillenbrand, B. Löchner, and H. Spies. The new WALDMEISTER loop at work. In F. Baader, editor, *CADE*, volume 2741 of *Lecture Notes in Computer Science*, pages 317–321, 2003.

[5] J. Giesl, R. Thiemann, and P. Schneider-Kamp. Proving and disproving termination of higher-order functions. In B. Gramlich, editor, *FroCoS*, volume 3717 of *Lecture Notes in Artificial Intelligence*, pages 216–231, 2005.

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

[7] N. Hirokawa and A. Middeldorp. Automating the dependency pair method. *Information and Computation*, 199(1-2):172–199, 2005.

[8] M. Korp and A. Middeldorp. Beyond dependency graphs. In R. Schmidt, editor, *CADE*, 2009. To appear.

[9] M. Korp, C. Sternagel, H. Zankl, and A. Middeldorp. Tyrolean Termination Tool 2. In R. Treinen, editor, *RTA*, volume 5595 of *Lecture Notes in Computer Science*, pages 295–304, 2009.

[10] A. Middeldorp. Approximations for strategies and termination. *Electronic Notes in Theoretical Computer Science*, 70(6):1–20, 2002.

[11] TeReSe. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, 2003.

[12] H. Zankl and A. Middeldorp. Increasing interpretations. *Annals of Mathematics and Artificial Intelligence*, 2009. To appear.