Termination Analysis for Innermost Rewriting via Transformations

Park Netrakom (1510213)

School of Information Science,
Japan Advanced Institute of Science and Technology

August 4, 2017

Keywords: tern rewriting, innermost termination, transformation.

Aim: Innermost Termination of Transformed Systems

Term rewriting is one of simple and powerful Turing-complete computational models, which underlies automated theorem proving (e.g. Vampire, Agda, Coq) and declarative programming languages (e.g. CafeOBJ, OCaml, Haskell). In these applications, computation is performed with evaluation strategies. So-called eager and lazy evaluations in functional programming are modeled as the innermost and outermost strategies in term rewriting, respectively.

Since computation needs not to be terminating, most software is concerned about the *termination* property. There are many research attempts to prove termination of term rewrite systems *automatically*. In contrast, there are fewer techniques for termination of the innermost strategy, even much fewer for that of the outermost strategy. In order to address lack of techniques for outermost termination proofs recently transformational approaches have emerged. In this methodology we reduce the outermost termination problem to the innermost termination problem by transforming a given term rewrite system.

In this research we focus on Thiemann's transformation [6], which is one of powerful techniques to reduce outermost termination to innermost termination. With a small example we illustrate the transformation.

Example 1. For the running example, we consider the term rewrite system $\mathcal{R}$ over the signature $\{\mathsf{f}^{(1)}, \mathsf{g}^{(1)}, \mathsf{b}^{(0)}\}$:

$$\begin{aligned} 1: &\quad \mathsf{f}(\mathsf{f}(\mathsf{g}(x))) \to x \\ 2: &\quad \mathsf{g}(\mathsf{b}) \to \mathsf{f}(\mathsf{g}(\mathsf{b})) \end{aligned}$$

The system is neither *terminating* nor *innermost terminating*, as witnessed by the infinite rewrite sequence starting from $\mathsf{g}(\mathsf{b})$:

$$\underline{\mathsf{g}(\mathsf{b})} \to_{\mathcal{R}} \mathsf{f}(\underline{\mathsf{g}(\mathsf{b})}) \to_{\mathcal{R}} \mathsf{f}(\mathsf{f}(\underline{\mathsf{g}(\mathsf{b})})) \to_{\mathcal{R}} \mathsf{f}(\mathsf{f}(\mathsf{f}(\underline{\mathsf{g}(\mathsf{b})}))) \to_{\mathcal{R}} \cdots$$

Here the underlined parts indicate the subterms rewritten by rule 2. However, if we employ the outermost strategy, rewriting of the term terminates:

$$\underline{\mathsf{g}(\mathsf{b})} \to_{\mathcal{R}} \mathsf{f}(\underline{\mathsf{g}(\mathsf{b})}) \to_{\mathcal{R}} \underline{\mathsf{f}(\mathsf{f}(\mathsf{g}(\mathsf{b})))} \to_{\mathcal{R}} \mathsf{b}$$

In fact $\mathcal{R}$ has the outermost termination property. We prove it by using Thiemann's transformation. The transformation yields the following rewrite system $\mathcal{R}^T$ over the signature $\{\mathsf{b}^{(0)}, \mathsf{f}^{(1)}, \mathsf{g}^{(1)}, \mathsf{f}_1^{(1)}, \mathsf{g}_1^{(1)}, \underline{\mathsf{f}}^{(1)}, \underline{\mathsf{g}}^{(1)}, \triangledown^{(1)}, \triangle^{(1)}, \blacktriangledown_{\mathsf{f}}^{(1)}, \blacktriangledown_{\mathsf{g}}^{(1)}, \blacktriangle^{(1)}, \mathsf{top}^{(1)}\}$:

$$
\begin{array}{lrcllrcl}
1: & \triangledown(\mathsf{f}(x)) & \to & \blacktriangledown_{\mathsf{f}}(\underline{\mathsf{f}}(x)) & 2: & \triangledown(\mathsf{g}(x)) & \to & \blacktriangledown_{\mathsf{g}}(\underline{\mathsf{g}}(x)) \\
3: & \blacktriangledown_{\mathsf{f}}(\underline{\mathsf{f}}(x)) & \to & \mathsf{f}_1(\triangledown(x)) & 4: & \blacktriangledown_{\mathsf{g}}(\underline{\mathsf{g}}(x)) & \to & \mathsf{g}_1(\triangledown(x)) \\
5: & \underline{\mathsf{f}}(\mathsf{f}(\mathsf{g}(x))) & \to & \blacktriangle(x) & 6: & \underline{\mathsf{g}}(\mathsf{b}) & \to & \blacktriangle(\mathsf{f}(\mathsf{g}(\mathsf{b}))) \\
7: & \blacktriangledown_{\mathsf{f}}(\blacktriangle(x)) & \to & \triangle(x) & 8: & \blacktriangledown_{\mathsf{g}}(\blacktriangle(x)) & \to & \triangle(x) \\
9: & \mathsf{f}_1(\triangle(x)) & \to & \triangle(\mathsf{f}(x)) & 10: & \mathsf{g}_1(\triangle(x)) & \to & \triangle(\mathsf{g}(x)) \\
11: & \mathsf{top}(\triangle(x)) & \to & \mathsf{top}(\triangledown(x)) & & & &
\end{array}
$$

For example, the outermost rewrite step of $\mathcal{R}$

$$\mathsf{f}(\mathsf{f}(\mathsf{g}(\mathsf{b}))) \to_{\mathcal{R}} \mathsf{b}$$

corresponds to the four innermost rewrite steps of $\mathcal{R}^T$:

$$
\begin{aligned}
\mathsf{top}(\triangledown(\mathsf{f}(\mathsf{f}(\mathsf{g}(\mathsf{b})))))) &\to_{\mathcal{R}^T} \mathsf{top}(\blacktriangledown_{\mathsf{f}}(\underline{\mathsf{f}}(\mathsf{f}(\mathsf{g}(\mathsf{b})))))) \\
&\to_{\mathcal{R}^T} \mathsf{top}(\blacktriangledown_{\mathsf{f}}(\blacktriangle(\mathsf{b}))) \\
&\to_{\mathcal{R}^T} \mathsf{top}(\triangle(\mathsf{b})) \\
&\to_{\mathcal{R}^T} \mathsf{top}(\triangledown(\mathsf{b}))
\end{aligned}
$$

Since this correspondence generally holds, we can show outermost termination of the original system $\mathcal{R}$ by proving innermost termination of the transformed system $\mathcal{R}^T$. It is also known that the transformation is *complete*, meaning that if $\mathcal{R}^T$ is innermost terminating then $\mathcal{R}$ is outermost terminating.

Now the remaining question is whether one can show innermost termination of such a transformed rewrite system. The above example clearly reveals a major problem of the approach: This kind of transformations significantly increases the complexity of term structure in rewrite rules. Unfortunately, even state-of-the-art termination provers tend to fail as proof techniques cannot analyze the complex term and rewriting structures. For example, AProVE and TTT2, the 1st and 2nd places on the termination competition in 2015, fail to prove innermost termination of the above system $\mathcal{R}^T$.

Approach: Transformation and Type Information

The aim of this research is to establish techniques for showing innermost termination of systems resulting from Thiemann's transformation. There are various transformation techniques for termination with specific strategies [2, 1]. Most of them result in rewrite systems similar to those of Thiemann's transformation.

There are two major problems of transformed systems. The first problem is that one rewrite step becomes many rewrite steps as seen in Example 1. Termination proofs are usually established by detecting decreasing parameters. However, the intermediate steps obfuscate the decreasingness. The second problem originates from the nature of innermost rewriting. Majority of existing termination techniques directly or indirectly employ the notion of reduction order, which does not fit for innermost termination proofs when the system is non-terminating.

In order to address these problems we develop new *transformation* techniques. Exploiting *type information*, we resolve the first problem about complexity of term structure. There is a technique to introduce many-sorts to untyped rewrite systems. As proved in the main part of the thesis, all rewrite systems induced by Thiemann's transformation admit (proper) many-sorted signatures. Based on the sort information we can perform *type-based reachability analysis* which can be integrated for various termination techniques, such as dependency graphs (Giesl, Arts, and Ohlebusch 2002) [5], usable rules (Hirokawa and Middeldorp 2007, Thiemann et al. 2008) [3, 7], and simple freezing (Hirokawa et al. 2013) [4].

For handling the second problem we introduce a transformation technique, dubbed *pattern separation*. Instantiating rewrite rules, this transformation fills in the gap between ordinary rewrite step and innermost rewrite step, the latter of which lacks the closure under substitutions. By using the aforementioned type introduction technique, pattern separation can be further improved.

Illustration

Here we illustrate these techniques, contributions of this thesis. We start with *type-based reachability analysis*.

Example 2 (continued from Example 1)**.** The next sort information can be attached to the transformed system $\mathcal{R}^T$.

$$\left\{\begin{array}{cccc} \nabla : \alpha \to \beta & \triangle : \alpha \to \beta & \underline{\mathsf{f}} : \alpha \to \delta & \underline{\mathsf{g}} : \alpha \to \delta \\ \blacktriangledown_{\mathsf{f}} : \delta \to \beta & \blacktriangledown_{\mathsf{g}} : \delta \to \beta & \blacktriangle : \alpha \to \delta & \underline{\mathsf{b}} : \alpha \\ \mathsf{f} : \alpha \to \alpha & \mathsf{g} : \alpha \to \alpha & \mathsf{f}_1 : \beta \to \beta & \mathsf{g}_1 : \beta \to \beta \\ \mathsf{top} : \beta \to \gamma & & & \end{array}\right\}$$

Here we suppose that with other termination methods we succeeded to eliminate rules 5 and 6 from $\mathcal{R}^T$. Terms of form $\mathsf{top}(\nabla(s))$ no longer reaches $\mathsf{top}(\triangle(t))$ for any terms s and t. Our type-based reachability analysis can detect this unreachability in the following way: We interpret each term to a set of function symbols that may appear in reachable terms. This can be computed by using the rewrite system $\|\mathcal{R}^T\|$ on sets:

$$\begin{array}{llll} 1: & \{\nabla\} \rightsquigarrow \{\blacktriangledown_{\mathsf{f}}, \underline{\mathsf{f}}\} & 2: & \{\nabla\} \rightsquigarrow \{\blacktriangledown_{\mathsf{g}}, \underline{\mathsf{g}}\} \\ 3: & \{\blacktriangledown_{\mathsf{f}}, \underline{\mathsf{f}}\} \rightsquigarrow \{\mathsf{f}_1, \nabla\} & 4: & \{\blacktriangledown_{\mathsf{g}}, \underline{\mathsf{g}}\} \rightsquigarrow \{\mathsf{g}_1, \nabla\} \\ & & & \\ 7: & \{\blacktriangledown_{\mathsf{f}}, \blacktriangle\} \rightsquigarrow \{\triangle\} & 8: & \{\blacktriangledown_{\mathsf{g}}, \blacktriangle\} \rightsquigarrow \{\triangle\} \\ 9: & \{\mathsf{f}_1, \triangle\} \rightsquigarrow \{\triangle\} & 10: & \{\mathsf{g}_1, \triangle\} \rightsquigarrow \{\triangle\} \\ 11: & \{\mathsf{top}, \triangle\} \rightsquigarrow \{\mathsf{top}, \nabla\} & & \end{array}$$

Because our terms are sorted, the interpretation of $\mathsf{top}(\nabla(s))$, say A, does not contain the symbol $\triangle$, and moreover the set A cannot reach a set containing $\triangle$ by using $\|\mathcal{R}^T\|$. This is sufficient to conclude the announced unreachability. The information of unreachability is used for the computation of the *dependency graph*. Although we omit its explanation here, this technique now shows innermost termination of $\mathcal{R}^T$.

Our running example can also be handled by pattern separation, which is our another contribution.

Example 3 (continued from Example 1)**.** Pattern separation replaces rule 4 of $\mathcal{R}^T$ by its instantiated versions:

$$4.1: \quad \blacktriangledown_{\mathsf{g}}(\underline{\mathsf{g}}(\mathsf{f}(x))) \to \mathsf{g}_1(\triangledown(\mathsf{f}(x))) \quad 4.2: \quad \blacktriangledown_{\mathsf{g}}(\underline{\mathsf{g}}(\mathsf{g}(x))) \to \mathsf{g}_1(\triangledown(\mathsf{g}(x)))$$
$$4.3: \quad \blacktriangledown_{\mathsf{g}}(\underline{\mathsf{g}}(\mathsf{h}(x))) \to \mathsf{g}_1(\triangledown(\mathsf{h}(x))) \quad 4.4: \quad \blacktriangledown_{\mathsf{g}}(\underline{\mathsf{g}}(\mathsf{c})) \to \mathsf{g}_1(\triangledown(\mathsf{c}))$$

Innermost termination of the resulting system can be shown by existing termination provers (such as AProVE and TTT2). We want to stress that the character of innermost rewrite step has been changed by the separation: As shown in the thesis, the final system has even the *termination* property.

Contribution

Here is the list of our contributions:

- analysis of types for Thiemann's transformation,
- type-based reachability analysis,
- a pattern separation technique, and
- the optimized version of pattern separation.

References

[1] J. Endrullis and D. Hendriks. From outermost to context-sensitive rewriting. In *Proceedings of the 20th International Conference on Rewriting Techniques and Applications*, volume 5595 of *Lecture Notes in Computer Science*, pages 305–319, 2009.

[2] J. Giesl and A. Middeldorp. Transformation techniques for context-sensitive rewrite systems. *Journal of Functional Programming*, 14:329–427, 2004.

[3] N. Hirokawa and A. Middeldorp. Tyrolean termination tool: Techniques and features. *Information and Computation*, 205:474–511, 2007.

[4] N. Hirokawa, A. Middeldorp, and H. Zankl. Uncurrying for termination and complexity. *Journal of Automated Reasoning*, 50:279315, 1990.

[5] N. Hirokawa, A. Middeldorp, and H. Zankl. Modular termination proofs for rewriting using dependency pairs. *Journal of Symbolic Computation*, 34:21–58, 2002.

[6] R. Thiemann. From outermost termination to innermost termination. In *Proceedings of the 35th Conference on Current Trends in Theory and Practice of Computer Science*, volume 5404 of *Lecture Notes in Computer Science*, pages 533–545, 2009.

[7] R. Thiemann and A. Middeldorp. Innermost termination of rewrite systems by labeling. In *Proceedings of the 7th International Workshop on Reduction Strategies in Rewriting and Programming*, volume 204 of *Electronic Notes in Theoretical Computer Science*, pages 3–19, 2008.