Transformations of Conditional Term Rewrite Systems

Karl Gmeiner and Bernhard Gramlich (Faculty Mentor)
Institute of Computer Languages
Vienna University of Technology
Email: {gmeiner,gramlich}@logic.at

Abstract — *Term rewriting is a well understood and commonly used framework in theoretical computer science for program analysis and program verification. An intuitive extension of term rewrite systems are conditional term rewrite systems. Yet, although they are more natural for certain tasks, they turn out to be more complex to analyze. In order to better understand and implement conditional term rewrite systems, various transformations of such systems into unconditional ones have been developed. In our work we analyzed existing approaches for such transformations, presented a unifying framework for them, and developed a new transformation that has certain advantages as compared to previous ones.*

I. Introduction

A. Motivation

Term rewriting is a widely accepted framework in theoretical computer science that derives from equational reasoning. It is used for program verification and program analysis, especially for functional-logic programming languages. Conditional term rewriting is a more expressive extension of rewriting that appears naturally in many practical applications.

Due to the inherent recursion in the conditions, conditional term rewrite systems (CTRSs) are considerably more complicated to analyze. In order to be able to use the much better understood setting of unconditional term rewrite systems (TRSs) to reason about CTRSs, several transformations have been defined (e.g. in [1]) in order to eliminate conditions while preserving desirable properties of the original CTRS.

In our work in [2], we analyzed existing approaches and compared them to each other. Partially based on [2], in [3] we introduced a unifying framework for a more coherent description of definitions and properties of such transformations, and presented a novel transformation that has various advantages as compared to previous approaches.

In the following we will briefly introduce CTRSs and transformations of CTRSs via an example and then sketch the ideas underlying our novel transformation.

B. Preliminaries

CTRSs consist of directed equations (*rules*) that have some application conditions. For instance, a logical *not* can be defined as a CTRS with two rules δ_1, δ_2 in the following way:

$$\mathcal{R}_{\mathrm{not}} = \left\{ \begin{array}{llll} \delta_1\colon \mathrm{not}(x) & \to & \mathrm{tt} & \Leftarrow \quad x = \mathrm{ff} \\ \delta_2\colon \mathrm{not}(x) & \to & \mathrm{ff} & \Leftarrow \quad x = \mathrm{tt} \end{array} \right\}$$

Equality in the conditions can be interpreted in many ways. We will only consider here *normal 1-CTRSs* $\mathcal{R}$, i.e., in all rules $l \to r \Leftarrow s_1 = t_1, \ldots, s_n = t_n$ of $\mathcal{R}$ equality in the conditions is interpreted as the transitive, reflexive closure $\to^*$ of the rewrite relation. Furthermore, all right-hand sides of the conditions $t_1, \ldots, t_n$ are *irreducible (w.r.t.* $\mathcal{R}_u = \{l \to r \mid l \to r \Leftarrow c \in \mathcal{R}\}$*) ground terms*, and all variables in the conditions and the right-hand side r also occur in the left-hand side l.

II. Transformation of CTRSs Revisited

In most transformations, conditional rules are split up into two or more unconditional rules. For instance, using so-called *unravelings* introduced in [1] the CTRS $\mathcal{R}_{\mathrm{not}}$ is transformed into

$$U(\mathcal{R}_{\mathrm{not}}) = \left\{ \begin{array}{rcl} \mathrm{not}(x) & \to & U_{\delta_1}(x, x) \\ U_{\delta_1}(\mathrm{ff}, x) & \to & \mathrm{tt} \\ \mathrm{not}(x) & \to & U_{\delta_2}(x, x) \\ U_{\delta_2}(\mathrm{tt}, x) & \to & \mathrm{ff} \end{array} \right\}$$

where condition evaluation is now done explicitly in the unconditional system,

Using the transformation of [4] the arity of the *not*-symbol is increased such that the conditions of both conditional rules can be encoded in it. The additional arguments initially contain $\bot$ to indicate an uninitialized condition:

$$\mathcal{R}'_{\mathrm{not}} = \left\{ \begin{array}{rcl} \mathrm{not}'(x, \bot, z) & \to & \mathrm{not}(x, \langle x \rangle, z) \\ \mathrm{not}'(x, \langle \mathrm{ff} \rangle, z) & \to & \mathrm{tt} \\ \mathrm{not}'(x, y, \bot) & \to & \mathrm{not}'(x, y, \langle x \rangle) \\ \mathrm{not}'(x, y, \langle \mathrm{tt} \rangle) & \to & \mathrm{ff} \end{array} \right\}$$

The CTRS $\mathcal{R}_{\mathrm{not}}$ is confluent and terminating. This implies that on all possible rewrite paths, we always obtain a unique normal form. In order to simulate conditional rewrite sequences in a transformed TRS, we also want to obtain unique unique normal forms in the transformed TRS that correspond to normal forms in the original system.

This property is not satisfied by the unraveling $U(\mathcal{R}_{\mathrm{not}})$. Consider the term $\mathrm{not}(\mathrm{tt})$. In $\mathcal{R}_{\mathrm{not}}$, for δ_1 the condition $\mathrm{tt} \to^* \mathrm{ff}$ is not satisfied so that we can only apply δ_2. Therefore $\mathrm{not}(\mathrm{tt})$ only rewrites to ff. In $U(\mathcal{R}_{\mathrm{not}})$, $\mathrm{not}(\mathrm{tt})$ rewrites to ff, but also to $U_{\delta_1}(\mathrm{tt}, \mathrm{tt})$ which is irreducible. We would now have to *backtrack* from $U_{\delta_1}(\mathrm{tt}, \mathrm{tt})$ to $\mathrm{not}(\mathrm{tt})$ and try different rewrite paths.

The transformed TRS $\mathcal{R}'_{\mathrm{not}}$ does not require backtracking, because as for $\mathcal{R}_{\mathrm{not}}$ we get unique normal forms: $\mathrm{not}'(\mathrm{tt}, \bot, \bot) \to \mathrm{not}'(\mathrm{tt}, \langle \mathrm{tt} \rangle, \bot) \to \mathrm{not}'(\mathrm{tt}, \langle \mathrm{tt} \rangle, \langle \mathrm{tt} \rangle) \to \mathrm{ff}$

The transformation of [4] yields satisfactory results for *left-linear confluent constructor CTRSs*, but for other confluent CTRSs it may still result in non-unique normal forms:

$$\mathcal{R} = \left\{ \begin{array}{lcll} f(g(x)) & \to & x & \Leftarrow \quad x = 0 \\ g(s(x)) & \to & g(x) & \end{array} \right\}$$

is transformed into

$$\mathcal{R}' = \left\{ \begin{array}{rcl} f'(g(x), \bot) & \to & f'(g(x), \langle x \rangle) \\ f'(g(x), \langle 0 \rangle) & \to & x \\ g(s(x)) & \to & g(x) \end{array} \right\}$$

The term $f(g(s(0)))$ (that corresponds to $f'(g(s(0)), \bot)$ in $\mathcal{R}'$) only has the normal form 0 in $\mathcal{R}$, but in $\mathcal{R}'$ we also obtain $f'(g(0), \langle s(0) \rangle)$. Observe that in the latter term the matcher for x in the g-subterm (0) is different from the matcher for x used in the conditional argument $(s(0))$.

Later, in [5] a transformation was defined that guarantees unique normal forms for all confluent CTRSs:

$$\mathcal{R}'' = \left\{ \begin{array}{rcl} f'(g(x), \bot) & \to & f'(g(x), \{x\}) \\ f'(g(x), \{0\}) & \to & \{x\} \\ g(s(x)) & \to & \{g(x)\} \\ f'(\{x\}, z) & \to & \{f'(x, \bot)\} \\ g(\{x\}) & \to & \{g(x)\} \\ s(\{x\}) & \to & \{s(x)\} \\ \{\{x\}\} & \to & \{x\} \end{array} \right\}$$

This transformation is syntactically rather complex and furthermore our experiments also showed drawbacks concerning efficiency: The purpose of the circum-fix symbol $\{.\}$ is to "reset" conditions that may block further rewrite steps. The term $f'(g(s(0)), \bot)$ rewrites to $f'(\{g(0)\}, \langle s(0) \rangle)$ which can be further rewritten to $\{f'(g(0), \bot)\}$ and $\{0\}$. A drawback of $\{.\}$ is its complexity and the fact that it resets conditional arguments "too often".

In [3] finally, we developed a new transformation that is sound for preserving normal forms for a large class of confluent CTRSs without the specific drawbacks of [5]. In our approach, we try to avoid that conditions get "out of sync" with other arguments. We therefore encode conditions in all subterms of a conditional rule, that *overlap* with another rule. Using this novel approach, our example system $\mathcal{R}$ is transformed into

$$\mathcal{R}''' = \left\{ \begin{array}{rcl} f(g'(x, \bot)) & \to & f(g'(x, \langle x \rangle)) \\ f(g'(x, \langle 0 \rangle)) & \to & x \\ g'(s(x), z) & \to & g'(x, \bot) \end{array} \right\}$$

Now the term $f(g'(s(0), \bot))$ yields the rewrite sequence

$$f(g'(s(0), \bot)) \to f(g'(s(0), \langle s(0) \rangle)) \to f(g'(0, \bot))$$

and finally we obtain the only normal form 0, as in $\mathcal{R}$.

III. Conclusion and Perspectives

In our work we have analyzed existing transformations from CTRSs into unconditional systems and, based on our findings, developed a new transformation with good syntactical and also practical properties.

In our future PhD research we plan to extend our approach to larger classes of CTRSs, to implement and evaluate it, and to use it for interesting application areas, e.g. for inversion of equational / functional programs via CTRSs.

Acknowledgments

The first author's research is funded by a grant of the Vienna PhD School of Informatics.

References

[1] M. Marchiori. Unravelings and ultra-properties. In M. Hanus and M. Rodríguez-Artalejo, eds., *Proc. 5th Int. Conf. on Algebraic and Logic Programming (ALP 1996), Aachen, Germany*, LNCS 1139, pp. 107–121. Springer, September 1996.

[2] K. Gmeiner. Transformation of conditional term rewriting systems. Master's thesis, Fakultät für Informatik, TU Wien, Austria, 1997.

[3] K. Gmeiner and B. Gramlich. Transformations of conditional rewrite systems revisited. In A. Corradini and U. Montanari, eds., *Recent Trends in Algebraic Development Techniques (WADT 2008) – Selected Papers*, LNCS 5486, pp. 166–186. Springer, 2009.

[4] S. Antoy, B. Brassel, and M. Hanus. Conditional narrowing without conditions. In *Proc. 5th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming (PPDP 2003), Uppsala, Sweden*, pp. 20–31. ACM Press, 2003.

[5] T.-F. Şerbănuţă and G. Roşu. Computationally equivalent elimination of conditions. In F. Pfenning, ed., *Proc. 17th International Conference on Rewriting Techniques and Applications (RTA 2006), Seattle, WA, USA*, LNCS 4098, pp. 19–34. Springer, 2006.