Conditional Complexity*

Cynthia Kop, Aart Middeldorp, and Thomas Sternagel

Institute of Computer Science
University of Innsbruck, Austria
{cynthia.kop|aart.middeldorp|thomas.sternagel}@uibk.ac.at

Abstract

We propose a notion of complexity for oriented conditional term rewrite systems. This notion is realistic in the sense that it measures not only successful computations but also partial computations that result in a failed rule application. A transformation to unconditional context-sensitive rewrite systems is presented which reflects this complexity notion, as well as a technique to derive runtime and derivational complexity bounds for the latter.

1998 ACM Subject Classification F.4.2 Grammars and Other Rewriting Systems

Keywords and phrases conditional term rewriting, complexity

Digital Object Identifier 10.4230/LIPIcs.RTA.2015.223

1 Introduction

Conditional term rewriting is a well-known computational paradigm. First studied in the eighties and early nineties of the previous century, in more recent years transformation techniques have received a lot of attention and automatic tools for (operational) termination [8, 16, 25] as well as confluence [27] were developed.

In this paper we are concerned with the following question: What is the length of a longest derivation to normal form in terms of the size of the starting term? For unconditional rewrite systems this question has been investigated extensively and numerous techniques have been developed that provide an upper bound on the resulting notions of derivational and runtime complexity (e.g. [5, 11, 12, 19, 20]). Tools that support complexity methods ([2, 22, 30]) are under active development and compete annually in the complexity competition.[1]

We are not aware of any techniques or tools for conditional (derivational and runtime) complexity—or indeed, even of a *definition* for conditional complexity. This may be for a good reason, as it is not obvious what such a definition should be. Of course, simply counting (top-level) steps will not do. Taking the conditions into account when counting successful rewrite steps is a natural idea and transformations from conditional term rewrite systems to unconditional ones exist (e.g., unravelings [24]) that do justice to this two-dimensional view [15, 16]. However, we will argue that this still gives rise to an unrealistic notion of complexity. Modern rewrite engines like Maude [6] that support conditional rewriting can spend significant resources on evaluating conditions that in the end prove to be useless for rewriting the term at hand. This should be taken into account when defining complexity.

Contribution. We propose a new notion of conditional complexity for a large class of reasonably well-behaved conditional term rewrite systems. This notion aims to capture the maximal number of rewrite steps that can be performed when reducing a term to normal

* This research is supported by the Austrian Science Fund (FWF) project I963.
[1] http://cbr.uibk.ac.at/competition/

26th International Conference on Rewriting Techniques and Applications (RTA'15).
Editor: Maribel Fernández; pp. 223–240

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

form, including the steps that were computed but ultimately not useful. In order to reuse existing methodology, we present a transformation into unconditional rewrite systems that can be used to estimate the conditional complexity. The transformed system is context-sensitive (Lucas [13, 14]), which is not yet supported by current complexity tools, but ignoring the corresponding restrictions, we still obtain an upper bound on the conditional complexity.

Organization. The remainder of the paper is organized as follows. In the next section we recall some preliminaries. Based on the analysis of conditional complexity in Section 3, we introduce our new notion formally in Section 4. Section 5 presents a transformation to context-sensitive rewrite systems, and in Section 6 we present an interpretation-based method targeting the resulting systems. Even though we are far removed from tool support, examples are given to illustrate that manual computations are feasible. Related work is discussed in Section 7 before we conclude in Section 8 with suggestions for future work.

2 Preliminaries

We assume familiarity with (conditional) term rewriting and all that (e.g., [4, 28, 24]) and only shortly recall important notions that are used in the following.

In this paper we consider *oriented* conditional term rewrite systems (CTRSs for short). Given a CTRS $\mathcal{R}$, a substitution σ, and a list of conditions $c\colon s_1 \approx t_1, \dots, s_k \approx t_k$, let $\mathcal{R} \vdash c\sigma$ denote $s_i\sigma \to_{\mathcal{R}}^* t_i\sigma$ for all $1 \leqslant i \leqslant k$. We have $s \to_{\mathcal{R}} t$ if there exist a position p in s, a rule $\ell \to r \Leftarrow c$ in $\mathcal{R}$, and a substitution σ such that $s|_p = \ell\sigma$, $t = s[r\sigma]_p$, and $\mathcal{R} \vdash c\sigma$. We may write $s \xrightarrow{\epsilon} t$ for a rewrite step at the root position and $s \xrightarrow{>\epsilon} t$ for a non-root step.

Given a (C)TRS $\mathcal{R}$ over a signature $\mathcal{F}$, the root symbols of left-hand sides of rules in $\mathcal{R}$ are called *defined* and every other symbol in $\mathcal{F}$ is a *constructor*. These sets are denoted by $\mathcal{F}_{\mathcal{D}}$ and $\mathcal{F}_{\mathcal{C}}$, respectively. For a defined symbol f, we write $\mathcal{R}{\restriction}f$ for the set of rules in $\mathcal{R}$ that define f. A constructor term consists of constructors and variables. A basic term is a term $f(t_1, \dots, t_n)$ with $f \in \mathcal{F}_{\mathcal{D}}$ and constructor terms $t_1, \dots, t_n$.

Context-sensitive rewriting, as used in Section 5, restricts the positions in a term where rewriting is allowed. A (C)TRS is combined with a *replacement map* μ, which assigns to every n-ary symbol $f \in \mathcal{F}$ a subset $\mu(f) \subseteq \{1, \dots, n\}$. A position p is *active* in a term t if either $p = \epsilon$, or $p = i\,q$, $t = f(t_1, \dots, t_n)$, $i \in \mu(f)$, and q is active in t_i. The set of active positions in a term t is denoted by $\mathcal{P}\mathsf{os}_\mu(t)$, and t may only be reduced at active positions.

Given a terminating and finitely branching TRS $\mathcal{R}$ over a signature $\mathcal{F}$, the *derivation height* of a term t is defined as $\mathsf{dh}(t) = \max\{n \mid t \to^n u \text{ for some term } u\}$. This leads to the notion of *derivational complexity* $\mathsf{dc}_{\mathcal{R}}(n) = \max\{\mathsf{dh}(t) \mid |t| \leqslant n\}$. If we restrict the definition to basic terms t we get the notion of *runtime complexity* $\mathsf{rc}_{\mathcal{R}}(n)$ [10].

Rewrite rules $\ell \to r \Leftarrow c$ of CTRSs are classified according to the distribution of variables among ℓ, r, and c. In this paper we consider 3-CTRSs, where the rules satisfy $\mathcal{V}\mathsf{ar}(r) \subseteq \mathcal{V}\mathsf{ar}(\ell, c)$. A CTRS $\mathcal{R}$ is *deterministic* if for every rule $\ell \to r \Leftarrow s_1 \approx t_1, \dots, s_k \approx t_k$ in $\mathcal{R}$ we have $\mathcal{V}\mathsf{ar}(s_i) \subseteq \mathcal{V}\mathsf{ar}(\ell, t_1, \dots, t_{i-1})$ for $1 \leqslant i \leqslant k$. A deterministic 3-CTRS $\mathcal{R}$ is *quasi-decreasing* if there exists a well-founded order $>$ with the subterm property that extends $\to_{\mathcal{R}}$, such that $\ell\sigma > s_i\sigma$ for all $\ell \to r \Leftarrow s_1 \approx t_1, \dots, s_k \approx t_k \in \mathcal{R}$, $1 \leqslant i \leqslant k$, and substitutions σ with $s_j\sigma \to_{\mathcal{R}}^* t_j\sigma$ for $1 \leqslant j < i$. Quasi-decreasingness ensures termination and, for finite CTRSs, computability of the rewrite relation. Quasi-decreasingness coincides with *operational termination* [15]. We call a CTRS *constructor-based* if the right-hand sides of conditions as well as the arguments of left-hand sides of rules are constructor terms.

Limitations. We restrict ourselves to left-linear constructor-based deterministic 3-CTRSs, where moreover all right-hand sides of conditions are linear, and use only variables not occurring in the left-hand side or in earlier conditions. That is, for every rule $f(\ell_1, \dots, \ell_n) \to r \Leftarrow s_1 \approx t_1, \dots, s_k \approx t_k \in \mathcal{R}$:

- $\ell_1, \dots, \ell_n, t_1, \dots, t_k$ are linear constructor terms without common variables,
- $\mathcal{V}\mathsf{ar}(s_i) \subseteq \mathcal{V}\mathsf{ar}(\ell_1, \dots, \ell_n, t_1, \dots, t_{i-1})$ for $1 \leqslant i \leqslant k$ and $\mathcal{V}\mathsf{ar}(r) \subseteq \mathcal{V}\mathsf{ar}(\ell_1, \dots, \ell_n, t_1, \dots, t_k)$.

We will call such systems *CCTRSs* in the sequel. Furthermore, we restrict our attention to *quasi-decreasing* and *confluent* CCTRSs. While these latter restrictions are not needed for the formal development in this paper, without them the complexity notion that we propose is either undefined or not meaningful, as argued below.

To appreciate the limitations, note that in CTRSs which are not deterministic, 3-CTRSs or quasi-decreasing, the rewrite relation is undecidable in general, which makes it hard to define what complexity means. The restriction to linear constructor-TRSs is common in rewriting, and the restrictions on the conditions are a natural extension of this. Most importantly, with these restrictions computation is unambiguous: To evaluate whether a term $\ell\sigma$ reduces with a rule $\ell \to r \Leftarrow s_1 \approx t_1, \dots, s_k \approx t_k$, we start by reducing $s_1\sigma$ and, finding an instance of t_1, extend σ to the new variables in t_1 resulting in σ', continue with $s_2\sigma'$, and so on. If any extension of σ satisfies all conditions then this procedure will find one, no matter how we reduce. However, if confluence, quasi-decreasingness or any of the restrictions on the conditions were dropped, this would no longer be the case and we might be unable to verify whether a rule applied without enumerating all possible reducts of its conditions. The restrictions on the ℓ_i are needed to obtain Lemma 5, which will be essential to justify the way we handle failure.

▶ **Example 1.** The CTRS $\mathcal{R}$ consisting of the rewrite rules

$$\begin{aligned} 0 + y &\to y & \mathsf{fib}(0) &\to \langle 0, \mathsf{s}(0)\rangle \\ \mathsf{s}(x) + y &\to \mathsf{s}(x + y) & \mathsf{fib}(\mathsf{s}(x)) &\to \langle z, w\rangle \;\Leftarrow\; \mathsf{fib}(x) \approx \langle y, z\rangle,\; y + z \approx w \end{aligned}$$

is a quasi-deterministic and confluent CCTRS. The requirements for quasi-decreasingness are satisfied (e.g.) by the lexicographic path order with precedence $\mathsf{fib} > \langle\cdot,\cdot\rangle > + > \mathsf{s}$.

3 Analysis

We start our analysis with a deceivingly simple CCTRS to illustrate that the notion of complexity for conditional systems is not obvious.

▶ **Example 2.** The CCTRS $\mathcal{R}_{\mathsf{even}}$ consists of the following six rewrite rules:

$$\begin{aligned} \mathsf{even}(0) &\to \mathsf{true} && (1) & \mathsf{odd}(0) &\to \mathsf{false} && (4)\\ \mathsf{even}(\mathsf{s}(x)) &\to \mathsf{true} \Leftarrow \mathsf{odd}(x) \approx \mathsf{true} && (2) & \mathsf{odd}(\mathsf{s}(x)) &\to \mathsf{true} \Leftarrow \mathsf{even}(x) \approx \mathsf{true} && (5)\\ \mathsf{even}(\mathsf{s}(x)) &\to \mathsf{false} \Leftarrow \mathsf{even}(x) \approx \mathsf{true} && (3) & \mathsf{odd}(\mathsf{s}(x)) &\to \mathsf{false} \Leftarrow \mathsf{odd}(x) \approx \mathsf{true} && (6) \end{aligned}$$

If, like in the unconditional case, we count the number of steps needed to normalize a term, then a term $t_n = \mathsf{even}(\mathsf{s}^n(0))$ has derivation height 1, since $t_n \to \mathsf{false}$ in a single step. To reflect actual computation, the rewrite steps to verify the condition should be taken into account. Viewed like this, normalizing t_n takes $n + 1$ rewrite steps.

Table 1 Number of steps required to normalize $\mathsf{even}(\mathsf{s}^n(\mathsf{0}))$ and $\mathsf{odd}(\mathsf{s}^n(\mathsf{0}))$ in Maude.

n	0	1	2	3	4	5	6	7	8	9	10	11	12
$2^{n+1}-1$	1	3	7	15	31	63	127	255	511	1023	2047	4095	8191
$\mathsf{even}(\mathsf{s}^n(\mathsf{0}))$	1	3	3	11	5	37	7	135	9	521	11	2059	13
$\mathsf{odd}(\mathsf{s}^n(\mathsf{0}))$	1	2	6	4	20	6	70	8	264	10	1034	12	4108
$\mathsf{even}(\mathsf{s}^n(\mathsf{0}))$	1	2	7	8	31	32	127	128	511	512	2047	2048	8191
$\mathsf{odd}(\mathsf{s}^n(\mathsf{0}))$	1	3	4	15	16	63	64	255	256	1023	1024	4095	4096

However, this still seems unrealistic, since a rewriting engine cannot know in advance which rule to attempt first. For example, when rewriting t_9, rule (2) may be tried first, which requires normalizing $\mathsf{odd}(\mathsf{s}^8(\mathsf{0}))$ to verify the condition. After finding that the condition fails, rule (3) is attempted. Thus, for $\mathcal{R}_{\mathsf{even}}$, a realistic engine would select a rule with a failing condition about half the time. If we assume a worst possible selection strategy and count all rewrite steps performed during the computation, we need $2^{n+1}-1$ steps to normalize t_n.

Although this exponential upper bound may come as a surprise, a powerful rewrite engine like Maude [6] does not perform much better, as can be seen from the data in Table 1. For rows three and four we presented the rules to Maude in the order given in Example 2. Changing the order to (4), (6), (5), (1), (3), (2) we obtain the last two rows. For no order on the rules is the optimal linear bound on the number of steps obtained for all tested terms.

From the above we conclude that *a realistic definition of conditional complexity should take failed computations into account.* This opens new questions, which are best illustrated on a different (admittedly artificial) CTRS.

▶ **Example 3.** The CCTRS $\mathcal{R}_{\mathsf{fg}}$ consists of the following two rewrite rules:

$$\mathsf{f}(x) \to x \qquad\qquad \mathsf{g}(x) \to \mathsf{a} \Leftarrow x \approx \mathsf{b}$$

How many steps does it take to normalize $t_{n,m} = \mathsf{f}^n(\mathsf{g}(\mathsf{f}^m(\mathsf{a})))$? As we have not imposed an evaluation strategy, one approach for evaluating this term could be as follows. We use the second rule on the subterm $\mathsf{g}(\mathsf{f}^m(\mathsf{a}))$. This fails in m steps. With the first rule at the root position we obtain $t_{n-1,m}$. We again attempt the second rule, failing in m steps. Repeating this scenario results in $n \cdot m$ rewrite steps before we reach the term $t_{0,m}$.

In the above example we keep attempting—and failing—to rewrite an unmodified copy of a subterm we tried before, with the same rule. Even though the position of the subterm $\mathsf{g}(\mathsf{f}^m(\mathsf{a}))$ changes, we already know that this reduction will fail. Hence it is reasonable to assume that once we fail a conditional rule on given subterms, we should not try the same rule again on (copies of) the same subterms. This idea will be made formal in Section 4.

▶ **Example 4.** Continuing with the term $t_{0,m}$ from the preceding example, we could try to use the second rule, which fails in m steps. Next, the first rule is applied on a subterm, and we obtain $t_{0,m-1}$. Again we try the second rule, failing after executing $m-1$ steps. Repeating this alternation results eventually in the normal form $t_{0,0}$, but not before computing $\frac{1}{2}(m^2+3m)$ rewrite steps in total.

Like in Example 3, we keep coming back to a subterm which we have already tried before in an unsuccessful attempt. The difference is that the subterm has been rewritten between successive attempts. According to the following general result, we do not need to reconsider a failed attempt to apply a conditional rewrite rule if only the arguments were changed.

▶ **Lemma 5.** *Given a CCTRS $\mathcal{R}$, suppose $s \xrightarrow{>\epsilon}{}^* t$ and let $\rho\colon \ell \to r \Leftarrow c$ be a rule such that s is an instance of ℓ. If $t \xrightarrow{\epsilon}_\rho u$ then there exists a term v such that $s \xrightarrow{\epsilon}_\rho v$ and $v \to^* u$.*

So if we can rewrite a term at the root position eventually, and the term already matches the left-hand side of the rule with which we can do so, then we can rewrite the term with this rule immediately and obtain the same result.

Proof. Let σ be a substitution such that $s = \ell\sigma$ and $\mathsf{dom}(\sigma) \subseteq \mathcal{V}\mathsf{ar}(\ell)$. Because ℓ is a basic term, all steps in $s \xrightarrow{>\epsilon}{}^* t$ take place in the substitution part σ of $\ell\sigma$. Since ℓ is a linear term, we have $t = \ell\tau$ for some substitution τ such that $\mathsf{dom}(\tau) \subseteq \mathcal{V}\mathsf{ar}(\ell)$ and $\sigma \to^* \tau$. Because the rule ρ applies to t at the root position, there exists an extension τ' of τ such that $\mathcal{R} \vdash c\tau'$. We have $u = r\tau'$. Define the substitution σ' as follows:

$$\sigma'(x) = \begin{cases} \sigma(x) & \text{if } x \in \mathcal{V}\mathsf{ar}(\ell) \\ \tau'(x) & \text{if } x \notin \mathcal{V}\mathsf{ar}(\ell) \end{cases}$$

We have $s = \ell\sigma = \ell\sigma'$ and $\sigma' \to^* \tau'$. Let $a \approx b$ be a condition in c. From $\mathcal{V}\mathsf{ar}(b) \cap \mathcal{V}\mathsf{ar}(\ell) = \varnothing$ we infer $a\sigma' \to^* a\tau' \to^* b\tau' = b\sigma'$. It follows that $\mathcal{R} \vdash c\sigma'$ and thus $s \xrightarrow{\epsilon}_\rho r\sigma'$. Hence we can take $v = r\sigma'$ as $r\sigma' \to^* r\tau' = u$. ◀

From these observations we see that we can mark occurrences of defined symbols with the rules we have tried without success or, symmetrically, with the rules we have yet to try.

4 Conditional Complexity

To formalize the ideas from Section 3, we label defined function symbols by subsets of the rules used to define them.

▶ **Definition 6.** Let $\mathcal{R}$ be a CCTRS over a signature $\mathcal{F}$. The labeled signature $\mathcal{F}'$ is defined as $\mathcal{F}_\mathcal{C} \cup \{f_R \mid f \in \mathcal{F}_\mathcal{D} \text{ and } R \subseteq \mathcal{R}\restriction f\}$. A labeled term is a term in $\mathcal{T}(\mathcal{F}', \mathcal{V})$.

Intuitively, the label R in f_R records the defining rules for f which have not yet been attempted.

▶ **Definition 7.** Let $\mathcal{R}$ be a CCTRS over a signature $\mathcal{F}$. The mapping $\mathsf{label}\colon \mathcal{T}(\mathcal{F}, \mathcal{V}) \to \mathcal{T}(\mathcal{F}', \mathcal{V})$ labels every defined symbol f with $\mathcal{R}\restriction f$. The mapping $\mathsf{erase}\colon \mathcal{T}(\mathcal{F}', \mathcal{V}) \to \mathcal{T}(\mathcal{F}, \mathcal{V})$ removes the labels of defined symbols.

We obviously have $\mathsf{erase}(\mathsf{label}(t)) = t$ for every $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$. The identity $\mathsf{label}(\mathsf{erase}(t)) = t$ holds for constructor terms t but not for arbitrary terms $t \in \mathcal{T}(\mathcal{F}', \mathcal{V})$.

▶ **Definition 8.** A *labeled normal form* is a term in $\mathcal{T}(\mathcal{F}_\mathcal{C} \cup \{f_\varnothing \mid f \in \mathcal{F}_\mathcal{D}\}, \mathcal{V})$.

The relation $\rightharpoonup$ is designed in such a way that a ground labeled term can be reduced if and only if it is not a labeled normal form. First, with Definition 9 we can remove a rule from a label if that rule will never be applicable due to an impossible matching problem.

▶ **Definition 9.** We write $s \xrightarrow{\perp} t$ if there exist a position $p \in \mathcal{P}\mathsf{os}(s)$ and a rewrite rule $\rho\colon f(\ell_1, \ldots, \ell_n) \to r \Leftarrow c$ such that

1. $s|_p = f_R(s_1, \ldots, s_n)$ with $\rho \in R$,
2. $t = s[f_{R\setminus\{\rho\}}(s_1, \ldots, s_n)]_p$, and
3. there exist a linear labeled normal form u with fresh variables, a substitution σ, and an index $1 \leqslant i \leqslant n$ such that $s_i = u\sigma$ and $\mathsf{erase}(u)$ does not unify with ℓ_i.

The last item ensures that rewriting s strictly below position p cannot give a reduct that matches ℓ, since $s_i = u\sigma$ can only reduce to instances $u\sigma'$ of u and thus not to an instance of ℓ_i. Furthermore, by the linearity of $\ell = f(\ell_1, \ldots, \ell_n)$ we also have that, if $s_1, \ldots, s_n$ are labeled normal forms then either $f(s_1, \ldots, s_n)$ is an instance of ℓ or $\xrightarrow{\perp}$ applies.

Second, Definition 10 describes how to "reduce" labeled terms in general.

▶ **Definition 10.** A *complexity-conscious reduction* is a sequence $t_1 \rightharpoonup t_2 \rightharpoonup \cdots \rightharpoonup t_m$ of labeled terms where $s \rightharpoonup t$ if either $s \xrightarrow{\perp} t$ or there exist a position $p \in \mathcal{P}\mathsf{os}(s)$, rewrite rule $\rho\colon f(\ell_1, \ldots, \ell_n) \to r \Leftarrow a_1 \approx b_1, \ldots, a_k \approx b_k$, substitution σ, and index $1 \leqslant j \leqslant k$ such that

1. $s|_p = f_R(s_1, \ldots, s_n)$ with $\rho \in R$ and $s_i = \ell_i\sigma$ for all $1 \leqslant i \leqslant n$,
2. $\mathsf{label}(a_i)\sigma \rightharpoonup^* b_i\sigma$ for all $1 \leqslant i \leqslant j$,

and either

3. $j = k$ and $t = s[\mathsf{label}(r)\sigma]_p$

in which case we speak of a *successful* step, or

4. $j < k$ and there exist a linear labeled normal form u and a substitution τ such that $\mathsf{label}(a_{j+1})\sigma \rightharpoonup^* u\tau$, u does not unify with b_{j+1}, and $t = s[f_{R\setminus\{\rho\}}(s_1, \ldots, s_n)]_p$,

which is a *failed* step.

It is easy to see that for all ground labeled terms s which are not labeled normal forms, a term t exists such that $s \xrightarrow{\perp} t$ or there are p, ρ, σ such that $s|_p$ "matches" ρ in the sense that the first requirement in Definition 10 is satisfied. As all b_i are linear constructor terms on fresh variables and conditions are evaluated from left to right, $\mathsf{label}(a_i)\sigma \rightharpoonup b_i\sigma$ simply indicates that $a_i\sigma$—with labels added to allow reducing defined symbols in a_i—reduces to an instance of b_i.

A successful reduction occurs when we manage to reduce each $\mathsf{label}(a_i)\sigma$ to $b_i\sigma$. A failed reduction happens when we start reducing $\mathsf{label}(a_i)\sigma$ and obtain a term that will never reduce to an instance of b_i. As discussed after Definition 9, this is what happens in case 4.

▶ **Definition 11.** The *cost* of a complexity-conscious reduction is the sum of the costs of its steps. The cost of a step $s \rightharpoonup t$ is 0 if $s \xrightarrow{\perp} t$,

$$1 + \sum_{i=1}^{k} \mathsf{cost}(\mathsf{label}(a_i)\sigma \rightharpoonup^* b_i\sigma)$$

in case of a successful step $s \rightharpoonup t$, and

$$\sum_{i=1}^{j} \mathsf{cost}(\mathsf{label}(a_i)\sigma \rightharpoonup^* b_i\sigma) + \mathsf{cost}(\mathsf{label}(a_{j+1})\sigma \rightharpoonup^* u\tau)$$

in case of a failed step $s \rightharpoonup t$. The *conditional derivational complexity* of a CCTRS $\mathcal{R}$ is defined as $\mathsf{cdc}_{\mathcal{R}}(n) = \max\{\mathsf{cost}(t \rightharpoonup^* u) \mid |t| \leqslant n \text{ and } t \rightharpoonup^* u \text{ for some term } u\}$. If we restrict t to basic terms we arrive at the *conditional runtime complexity* $\mathsf{crc}_{\mathcal{R}}(n)$.

Note that the cost of a failed step is the cost to evaluate its conditions and conclude failure, while for a successful step we add one for the step itself.

The following result connects the relations $\rightharpoonup$ and $\to$ to each other.

▶ **Lemma 12.** *Let $\mathcal{R}$ be a CCTRS.*

1. *If* $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ *and* $s \to^* t$ *then* $\mathsf{label}(s) \rightharpoonup^* \mathsf{label}(t)$.
2. *If* $s, t \in \mathcal{T}(\mathcal{F}', \mathcal{V})$ *and* $s \rightharpoonup^* t$ *then* $\mathsf{erase}(s) \to^* \mathsf{erase}(t)$.

Proof. 1. We use induction on the number of rewrite steps needed to derive $s \to^* t$. If $s = t$ then the result is obvious, so let $s \to u \to^* t$. The induction hypothesis yields $\mathsf{label}(u) \rightharpoonup^* \mathsf{label}(t)$, so it suffices to show $\mathsf{label}(s) \rightharpoonup^* \mathsf{label}(u)$. There exist a position $p \in \mathcal{P}\mathsf{os}(s)$, a rule $\rho\colon \ell \to r \Leftarrow a_1 \approx b_1, \dots, a_k \approx b_k$, and a substitution σ such that $s|_p = \ell\sigma$, $u = s[r\sigma]_p$, and $a_i\sigma \to^* b_i\sigma$ for all $1 \leqslant i \leqslant n$. Let σ' be the (labeled) substitution $\mathsf{label} \circ \sigma$. Fix $1 \leqslant i \leqslant k$. We have $\mathsf{label}(a_i\sigma) = \mathsf{label}(a_i)\sigma'$ and $\mathsf{label}(b_i\sigma) = b_i\sigma'$ (as b_i is a constructor term). Because $a_i\sigma \to^* b_i\sigma$ is used in the derivation of $s \to^* t$ we can apply the induction hypothesis, resulting in $\mathsf{label}(a_i\sigma) \rightharpoonup^* \mathsf{label}(b_i\sigma)$. Furthermore, writing $\ell = f(\ell_1, \dots, \ell_n)$, we obtain $\mathsf{label}(\ell) = f_{\mathcal{R}\restriction f}(\ell_1, \dots, \ell_n)$. Hence

$$\mathsf{label}(s) = \mathsf{label}(s)[\mathsf{label}(\ell)\sigma']_p \rightharpoonup \mathsf{label}(s)[\mathsf{label}(r)\sigma']_p = \mathsf{label}(u)$$

because conditions (1)–(4) in Definition 10 are satisfied.

2. We use induction on the pair $(\mathsf{cost}(s \rightharpoonup^* t), \|s\|)$ where $\|s\|$ denotes the sum of the sizes of the labels of defined symbols in s, ordered lexicographically. The result is obvious if $s = t$, so let $s \rightharpoonup u \rightharpoonup^* t$. Clearly $\mathsf{cost}(s \rightharpoonup^* t) \geqslant \mathsf{cost}(u \rightharpoonup^* t)$. We distinguish two cases.

   - Suppose $s \xrightarrow{\perp} u$ or $s \rightharpoonup u$ by a failed step. In either case we have $\mathsf{erase}(s) = \mathsf{erase}(u)$ and $\|s\| = \|u\| + 1$. The induction hypothesis yields $\mathsf{erase}(u) \to^* \mathsf{erase}(t)$.
   - Suppose $s \rightharpoonup u$ is a successful step. So there exist a position $p \in \mathcal{P}\mathsf{os}(s)$, a rule $\rho\colon \ell \to r \Leftarrow a_1 \approx b_1, \dots, a_k \approx b_k$ in $\mathcal{R}$, a substitution σ, and terms $\ell', a'_1, \dots, a'_k$ such that $s|_p = \ell'\sigma$ with $\mathsf{erase}(\ell') = \ell$, $a'_i\sigma \rightharpoonup^* b_i\sigma$ with $\mathsf{erase}(a'_i) = a_i$ for all $1 \leqslant i \leqslant k$, and $u = s[\mathsf{label}(r)\sigma]_p$. Let σ' be the (unlabeled) substitution $\mathsf{erase} \circ \sigma$. We have $\mathsf{erase}(s) = \mathsf{erase}(s)[\ell\sigma']_p$ and $\mathsf{erase}(u) = \mathsf{erase}(s)[r\sigma']_p$. Since $\mathsf{cost}(s \rightharpoonup u) > \mathsf{cost}(a'_i\sigma \rightharpoonup^* b_i\sigma)$ we obtain $a_i\sigma' = \mathsf{erase}(a'_i\sigma) \to^* \mathsf{erase}(b_i\sigma) = b_i\sigma'$ from the induction hypothesis, for all $1 \leqslant i \leqslant k$. Hence $\mathsf{erase}(s) \to \mathsf{erase}(u)$. Finally, $\mathsf{erase}(u) \to^* \mathsf{erase}(t)$ by another application of the induction hypothesis. ◀

5 Complexity Transformation

The notion of complexity introduced in the preceding section has the downside that we cannot easily reuse existing complexity results and tools. Therefore, we will consider a transformation to unconditional rewriting where, rather than tracking rules in the labels of the defined function symbols, we will keep track of them in separate arguments, but restrict reduction by adopting a suitable context-sensitive replacement map.

▶ **Definition 13.** Let $\mathcal{R}$ be a CCTRS over a signature $\mathcal{F}$. For $f \in \mathcal{F}_\mathcal{D}$, let m_f be the number of rules in $\mathcal{R}\restriction f$. The context-sensitive signature $(\mathcal{G}, \mu)$ is defined as follows:

- $\mathcal{G}$ contains two constants $\perp$ and $\top$,
- for every constructor symbol $g \in \mathcal{F}_\mathcal{C}$ of arity n, $\mathcal{G}$ contains the symbol g with the same arity and $\mu(g) = \{1, \dots, n\}$,
- for every defined symbol $f \in \mathcal{F}_\mathcal{D}$ of arity n, $\mathcal{G}$ contains two symbols f and f_a of arity $n + m_f$ with $\mu(f) = \{1, \dots, n\}$ and $\mu(f_\mathsf{a}) = \{n+1, \dots, n+m_f\}$,
- for every defined symbol $f \in \mathcal{F}_\mathcal{D}$ of arity n, rewrite rule $\rho\colon \ell \to r \Leftarrow c_1, \dots, c_k$ in $\mathcal{R}\restriction f$, and $1 \leqslant i \leqslant k$, $\mathcal{G}$ contains a symbol c^i_ρ of arity $n + i$ with $\mu(c^i_\rho) = \{n + i\}$.

Fixing an order $\mathcal{R}\restriction f = \{\rho_1, \ldots, \rho_{m_f}\}$, terms in $\mathcal{T}(\mathcal{G}, \mathcal{V})$ that are involved in reducing $f(s_1, \ldots, s_n) \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ will have one of two forms: $f(s_1, \ldots, s_n, t_1, \ldots, t_{m_f})$ with each $t_i \in \{\top, \bot\}$, indicating that rule ρ_i has been attempted (and failed) if and only if $t_i = \bot$, and $f_{\mathsf{a}}(s_1, \ldots, s_n, t_1, \ldots, c^{j+1}_{\rho_i}(s_1, \ldots, s_n, b_1, \ldots, b_j, u_{j+1}), \ldots, t_{m_f})$ indicating that rule ρ_i is currently being evaluated and the first j conditions of ρ_i have succeeded. The reason for passing the terms $s_1, \ldots, s_n$ to $c^{j+1}_{\rho_i}$ is that it allows for easier complexity methods.

▶ **Definition 14.** The maps $\xi_\star \colon \mathcal{T}(\mathcal{F}, \mathcal{V}) \to \mathcal{T}(\mathcal{G}, \mathcal{V})$ with $\star \in \{\bot, \top\}$ are inductively defined:

$$\xi_\star(t) = \begin{cases} t & \text{if } t \text{ is a variable,} \\ f(\xi_\star(t_1), \ldots, \xi_\star(t_n)) & \text{if } t = f(t_1, \ldots, t_n) \text{ and } f \text{ is a constructor symbol,} \\ f(\xi_\star(t_1), \ldots, \xi_\star(t_n), \star, \ldots, \star) & \text{if } t = f(t_1, \ldots, t_n) \text{ and } f \text{ is a defined symbol.} \end{cases}$$

Linear terms in the set $\{\xi_\bot(t) \mid t \in \mathcal{T}(\mathcal{F}, \mathcal{V})\}$ are called $\bot$-*patterns*.

In the transformed system that we will define, a ground term is in normal form if and only if it is a $\bot$-pattern. This allows for syntactic "normal form" tests. Most importantly, it allows for purely syntactic anti-matching tests: If s does not reduce to an instance of some linear constructor term t, then $s \to^* u\sigma$ for some substitution σ and $\bot$-pattern u that does not unify with t. What is more, we only need to consider a finite number of $\bot$-patterns u.

▶ **Definition 15.** Let t be a linear constructor term. The set of *anti-patterns* $\mathsf{AP}(t)$ is inductively defined as follows. If t is a variable then $\mathsf{AP}(t) = \varnothing$. If $t = f(t_1, \ldots, t_n)$ then $\mathsf{AP}(t)$ consists of the following $\bot$-patterns:

- $g(x_1, \ldots, x_m)$ for every m-ary constructor symbol g different from f,
- $g(x_1, \ldots, x_m, \bot, \ldots, \bot)$ for every defined symbol g of arity m in $\mathcal{F}$, and
- $f(x_1, \ldots, x_{i-1}, u, x_{i+1}, \ldots, x_n)$ for all $1 \leqslant i \leqslant n$ and $u \in \mathsf{AP}(t_i)$.

Here $x_1, \ldots, x_{m(n)}$ are fresh and pairwise distinct variables.

▶ **Example 16.** Consider the CCTRS of Example 1. The set $\mathsf{AP}(\langle z, w\rangle)$ consists of the $\bot$-patterns $\mathsf{0}$, $\mathsf{s}(x)$, $\mathsf{fib}(x, \bot, \bot)$, and $+(x, y, \bot, \bot)$.

The straightforward proof of the following lemma is omitted.

▶ **Lemma 17.** *Let s be a $\bot$-pattern and t a linear constructor term with $\mathcal{V}\mathsf{ar}(s) \cap \mathcal{V}\mathsf{ar}(t) = \varnothing$. If s and t are not unifiable then s is an instance of an anti-pattern in $\mathsf{AP}(t)$.*

We are now ready to define the transformation from a CCTRS $(\mathcal{F}, \mathcal{R})$ to a context-sensitive TRS $(\mathcal{G}, \mu, \Xi(\mathcal{R}))$. Here, we will use the notation $\langle t_1, \ldots, t_n\rangle[u]_i$ to denote the sequence $t_1, \ldots, t_{i-1}, u, t_{i+1}, \ldots, t_n$ and we occasionally write $\vec{t}$ for a sequence $t_1, \ldots, t_n$.

▶ **Definition 18.** Let $\mathcal{R}$ be a CCTRS over a signature $\mathcal{F}$. For every defined symbol $f \in \mathcal{F}_\mathcal{D}$ we fix an order on the m_f rules that define f: $\mathcal{R}\restriction f = \{\rho_1, \ldots, \rho_{m_f}\}$. The context-sensitive TRS $\Xi(\mathcal{R})$ is defined over the signature $(\mathcal{G}, \mu)$ as follows. Let $\rho \colon f(\ell_1, \ldots, \ell_n) \to r \Leftarrow a_1 \approx b_1, \ldots, a_k \approx b_k$ be the i-th rule in $\mathcal{R}\restriction f$.

- If $k = 0$ then $\Xi(\mathcal{R})$ contains the rule

$$f(\ell_1, \ldots, \ell_n, \langle x_1, \ldots, x_{m_f}\rangle[\top]_i) \to \xi_\top(r) \tag{1_ρ}$$

- If $k > 0$ then $\Xi(\mathcal{R})$ contains the rules

$$f(\vec{\ell}, \langle x_1, \dots, x_{m_f}\rangle[\top]_i) \to f_{\mathsf{a}}(\vec{\ell}, \langle x_1, \dots, x_{m_f}\rangle[c^1_\rho(\vec{\ell}, \xi_\top(a_1))]_i) \tag{2_ρ}$$
$$f_{\mathsf{a}}(\vec{\ell}, \langle x_1, \dots, x_{m_f}\rangle[c^k_\rho(\vec{y}, b_1, \dots, b_k)]_i) \to \xi_\top(r) \tag{3_ρ}$$

  and the rules

$$f_{\mathsf{a}}(\vec{\ell}, \langle x_1, \dots, x_{m_f}\rangle[c^j_\rho(\vec{y}, b_1, \dots, b_j)]_i) \to f_{\mathsf{a}}(\vec{\ell}, \langle x_1, \dots, x_{m_f}\rangle[c^{j+1}_\rho(\vec{y}, b_1, \dots, b_j, \xi_\top(a_{j+1}))]_i) \tag{4_ρ}$$

  for all $1 \leqslant j < k$, and the rules

$$f_{\mathsf{a}}(\vec{\ell}, \langle x_1, \dots, x_{m_f}\rangle[c^j_\rho(\vec{y}, b_1, \dots, b_{j-1}, v)]_i) \to f(\vec{\ell}, \langle x_1, \dots, x_{m_f}\rangle[\bot]_i) \tag{5_ρ}$$

  for all $1 \leqslant j \leqslant k$ and $v \in \mathsf{AP}(b_j)$.
- Regardless of k, $\Xi(\mathcal{R})$ contains the rules

$$f(\langle y_1, \dots, y_n\rangle[v]_j, \langle x_1, \dots, x_{m_f}\rangle[\top]_i) \to f(\langle y_1, \dots, y_n\rangle[v]_j, \langle x_1, \dots, x_{m_f}\rangle[\bot]_i) \tag{6_ρ}$$

  for all $1 \leqslant j \leqslant n$ and $v \in \mathsf{AP}(\ell_j)$.

Here $x_1, \dots, x_{m_f}, y_1, \dots, y_n$ are fresh and pairwise distinct variables. A step using rule (1_ρ) or rule (3_ρ) has cost 1; other rules—also called *administrative rules*—have cost 0.

Rule (1_ρ) simply adds the $\top$ labels to the right-hand sides of unconditional rules. To apply a conditional rule ρ, we "activate" the current function symbol with rule (2_ρ) and start evaluating the first condition of ρ by steps inside the last argument of c^1_ρ. With rules (4_ρ) we move to the next condition and, after all conditions have succeeded, an application of rule (3_ρ) results in the right-hand side with $\top$ labels. If a condition fails (5_ρ) or the left-hand side of the rule does not match and will never match (6_ρ), then we simply replace the label for ρ by $\bot$, indicating that we do not need to try it again.

These rules carry some redundant information. For example, all c^i_ρ are passed the parameters $\ell_1, \dots, \ell_n$ of the corresponding rule. This is done to make it easier to orient the resulting rules with interpretations, as we will see in Section 6. Also, instead of passing $b_1, \dots, b_j$ to each c^{j+1}_ρ, and $\ell_1, \dots, \ell_n$ to f_{a}, it would suffice to pass along their *variables*. This was left in the current form to simplify the presentation.

Note that the rules that do not produce the right-hand side of the originating conditional rewrite rule are administrative and hence do not contribute to the cost of a reduction. The anti-pattern sets result in many rules (5_ρ) and (6_ρ), but all of these are simple. We could generalize the system by replacing each $\star_i$ by a fresh variable; the complexity of the resulting (smaller) TRS gives an upper bound for the original complexity.

▶ **Example 19.** The (context-sensitive) TRS $\Xi(\mathcal{R}_{\mathsf{even}})$ consists of the following rules:

$$\mathsf{even}(\mathsf{0}, \top, y, z) \to \mathsf{true} \tag{1_1}$$
$$\mathsf{even}(\star_1, \top, y, z) \to \mathsf{even}(\star_1, \bot, y, z) \tag{6_1}$$
$$\mathsf{even}(\mathsf{s}(x), y, \top, z) \to \mathsf{even}_{\mathsf{a}}(\mathsf{s}(x), y, \mathsf{c}^1_2(\mathsf{s}(x), \mathsf{odd}(x, \top, \top, \top)), z) \tag{2_2}$$
$$\mathsf{even}_{\mathsf{a}}(\mathsf{s}(x), y, \mathsf{c}^1_2(y', \mathsf{true}), z) \to \mathsf{true} \tag{3_2}$$
$$\mathsf{even}_{\mathsf{a}}(\mathsf{s}(x), y, \mathsf{c}^1_2(y', \star_2), z) \to \mathsf{even}(\mathsf{s}(x), y, \bot, z) \tag{5_2}$$
$$\mathsf{even}(\star_3, y, \top, z) \to \mathsf{even}(\star_3, y, \bot, z) \tag{6_2}$$

$$\mathsf{even}(\mathsf{s}(x),y,z,\top) \to \mathsf{even}_\mathsf{a}(\mathsf{s}(x),y,z,\mathsf{c}^1_3(\mathsf{s}(x),\mathsf{even}(x,\top,\top,\top))) \tag{2_3}$$
$$\mathsf{even}_\mathsf{a}(\mathsf{s}(x),y,z,\mathsf{c}^1_3(y',\mathsf{true})) \to \mathsf{false} \tag{3_3}$$
$$\mathsf{even}_\mathsf{a}(\mathsf{s}(x),y,z,\mathsf{c}^1_3(y',\star_2)) \to \mathsf{even}(\mathsf{s}(x),y,z,\bot) \tag{5_3}$$
$$\mathsf{even}(\star_3,y,z,\top) \to \mathsf{even}(\star_3,y,z,\bot) \tag{6_3}$$
$$\mathsf{odd}(0,\top,y,z) \to \mathsf{false} \tag{1_4}$$
$$\mathsf{odd}(\star_1,\top,y,z) \to \mathsf{odd}(\star_1,\bot,y,z) \tag{6_4}$$
$$\mathsf{odd}(\mathsf{s}(x),y,\top,z) \to \mathsf{odd}_\mathsf{a}(\mathsf{s}(x),y,\mathsf{c}^1_5(\mathsf{s}(x),\mathsf{odd}(x,\top,\top,\top)),z) \tag{2_5}$$
$$\mathsf{odd}_\mathsf{a}(\mathsf{s}(x),y,\mathsf{c}^1_5(y',\mathsf{true}),z) \to \mathsf{false} \tag{3_5}$$
$$\mathsf{odd}_\mathsf{a}(\mathsf{s}(x),y,\mathsf{c}^1_5(y',\star_2),z) \to \mathsf{odd}(\mathsf{s}(x),y,\bot,z) \tag{5_5}$$
$$\mathsf{odd}(\star_3,y,\top,z) \to \mathsf{odd}(\star_3,y,\bot,z) \tag{6_5}$$
$$\mathsf{odd}(\mathsf{s}(x),y,z,\top) \to \mathsf{odd}_\mathsf{a}(\mathsf{s}(x),y,z,\mathsf{c}^1_6(\mathsf{s}(x),\mathsf{even}(x,\top,\top,\top))) \tag{2_6}$$
$$\mathsf{odd}_\mathsf{a}(\mathsf{s}(x),y,z,\mathsf{c}^1_6(y',\mathsf{true})) \to \mathsf{true} \tag{3_6}$$
$$\mathsf{odd}_\mathsf{a}(\mathsf{s}(x),y,z,\mathsf{c}^1_6(y',\star_2)) \to \mathsf{odd}(\mathsf{s}(x),y,z,\bot) \tag{5_6}$$
$$\mathsf{odd}(\star_3,y,z,\top) \to \mathsf{odd}(\star_3,y,z,\bot) \tag{6_6}$$

for all

$$\star_1 \in \mathsf{AP}(0) = \{\mathsf{true},\mathsf{false},\mathsf{s}(x),\mathsf{even}(x,\bot,\bot,\bot),\mathsf{odd}(x,\bot,\bot,\bot)\}$$
$$\star_2 \in \mathsf{AP}(\mathsf{true}) = \{\mathsf{false},0,\mathsf{s}(x),\mathsf{even}(x,\bot,\bot,\bot),\mathsf{odd}(x,\bot,\bot,\bot)\}$$
$$\star_3 \in \mathsf{AP}(\mathsf{s}(x)) = \{\mathsf{true},\mathsf{false},0,\mathsf{even}(x,\bot,\bot,\bot),\mathsf{odd}(x,\bot,\bot,\bot)\}$$

Below we relate complexity-conscious reductions with $\mathcal{R}$ to context-sensitive reductions in $\Xi(\mathcal{R})$. The following definition explains how we map terms in $\mathcal{T}(\mathcal{F}',\mathcal{V})$ to terms in $\mathcal{T}(\mathcal{G},\mathcal{V})$. It resembles the earlier definition of $\xi_\star$.

▶ **Definition 20.** For $t \in \mathcal{T}(\mathcal{F}',\mathcal{V})$ we define

$$\zeta(t) = \begin{cases} t & \text{if } t \in \mathcal{V}, \\ f(\zeta(t_1),\dots,\zeta(t_n)) & \text{if } t = f(t_1,\dots,t_n) \text{ with } f \text{ a constructor symbol}, \\ f(\zeta(t_1),\dots,\zeta(t_n),c_1,\dots,c_{m_f}) & \text{if } t = f_R(t_1,\dots,t_n) \text{ with } R \subseteq \mathcal{R}{\restriction}f \end{cases}$$

where $c_i = \top$ if the i-th rule of $\mathcal{R}{\restriction}f$ belongs to R and $c_i = \bot$ otherwise, for $1 \leqslant i \leqslant m_f$. For a substitution $\sigma \in \Sigma(\mathcal{F}',\mathcal{V})$ we denote the substitution $\zeta \circ \sigma$ by σ_ζ.

It is easy to see that $p \in \mathcal{P}\mathsf{os}_\mu(\zeta(t))$ if and only if $p \in \mathcal{P}\mathsf{os}(t)$ if and only if $\zeta(t)|_p \notin \{\bot,\top\}$, for any $t \in \mathcal{T}(\mathcal{F}',\mathcal{V})$. The easy induction proof of the following lemma is omitted.

▶ **Lemma 21.** *If $t \in \mathcal{T}(\mathcal{F},\mathcal{V})$ then $\zeta(\mathsf{label}(t)) = \xi_\top(t)$. If $t \in \mathcal{T}(\mathcal{F}',\mathcal{V})$ and $\sigma \in \Sigma(\mathcal{F}',\mathcal{V})$ then $\zeta(t\sigma) = \zeta(t)\sigma_\zeta$. Moreover, if t is a labeled normal form then $\zeta(t) = \xi_\bot(\mathsf{erase}(t))$.*

▶ **Theorem 22.** *Let $\mathcal{R}$ be a CCTRS. If $s \rightharpoonup^* t$ is a complexity-conscious reduction with cost N then there exists a context-sensitive reduction $\zeta(s) \to^*_{\Xi(\mathcal{R}),\mu} \zeta(t)$ with cost N.*

Proof. We use induction on the number of steps in $s \rightharpoonup^* t$. The result is obvious when this number is zero, so let $s \rightharpoonup u \rightharpoonup^* t$ with M the cost of the step $s \rightharpoonup u$ and $N - M$ the cost of $u \rightharpoonup^* t$. The induction hypothesis yields a context-sensitive reduction $\zeta(u) \to^*_{\Xi(\mathcal{R}),\mu} \zeta(t)$ of cost $N - M$ and so it remains to show that there exists a context-sensitive reduction $\zeta(s) \to^*_{\Xi(\mathcal{R}),\mu} \zeta(u)$ of cost M. Let $\rho\colon f(\ell_1,\dots,\ell_n) \to r \Leftarrow a_1 \approx b_1,\dots,a_k \approx b_k$ be the rule

in $\mathcal{R}$ that gives rise to the step $s \rightharpoonup u$ and let i be its index in $\mathcal{R}\restriction f$. There exist a position $p \in \mathcal{P}\mathsf{os}(s)$, terms $s_1, \dots, s_n$, and a subset $R \subseteq \mathcal{R}\restriction f$ such that $s|_p = f_R(s_1, \dots, s_n)$ and $\rho \in R$. We have $\zeta(s)|_p = \zeta(s|_p) = f_R(\zeta(s_1), \dots, \zeta(s_n), c_1, \dots, c_{m_f})$ where $c_j = \top$ if the j-th rule of $\mathcal{R}\restriction f$ belongs to R and $c_j = \bot$ otherwise, for $1 \leqslant j \leqslant m_f$. In particular, $c_i = \top$. Note that p is an active position in $\zeta(s)$. We distinguish three cases.

- First suppose that $s \xrightarrow{\bot} u$. So $M = 0$, $u = s[f_{R\setminus\{\rho\}}(s_1, \dots, s_n)]_p$, and there exist a linear labeled normal form v, a substitution σ, and an index $1 \leqslant j \leqslant n$ such that $s_j = v\sigma$ and $\mathsf{erase}(v)$ does not unify with ℓ_j. By Lemma 21, $\zeta(s_j) = \zeta(v\sigma) = \zeta(v)\sigma_\zeta = \xi_\bot(\mathsf{erase}(v))\sigma_\zeta$. By definition, $\xi_\bot(\mathsf{erase}(v))$ is a $\bot$-pattern, which cannot unify with ℓ_j because $\mathsf{erase}(v)$ does not. From Lemma 17 we obtain an anti-pattern $v' \in \mathsf{AP}(\ell_j)$ such that $\xi_\bot(\mathsf{erase}(v))$ is an instance of v'. Hence $\zeta(s) = \zeta(s)[f(\zeta(s_1), \dots, \zeta(s_n), c_1, \dots, c_{m_f})]_p$ with $\zeta(s_j)$ an instance of $v' \in \mathsf{AP}(\ell_j)$ and $c_i = \top$. Consequently, $\zeta(s)$ reduces to $\zeta(s)[f(\zeta(s_1), \dots, \zeta(s_n), \langle c_1, \dots, c_{m_f}\rangle[\bot]_i)]_p$ by an application of rule (6_ρ), which has cost zero. The latter term equals $\zeta(s[f_{R\setminus\{\rho\}}(s_1, \dots, s_n)]_p) = \zeta(u)$, and hence we are done.
- Next suppose that $s \rightharpoonup u$ is a successful step. So there exists a substitution σ such that $\mathsf{label}(a_i)\sigma \rightharpoonup^* b_i\sigma$ with cost M_i for all $1 \leqslant i \leqslant k$, and $M = 1 + M_1 + \cdots + M_k$. The induction hypothesis yields reductions $\zeta(\mathsf{label}(a_i)\sigma) \to^*_{\Xi(\mathcal{R}),\mu} \zeta(b_i\sigma)$ with cost M_i. By Lemma 21, $\zeta(\mathsf{label}(a_i)\sigma) = \zeta(\mathsf{label}(a_i))\sigma_\zeta = \xi_\top(a_i)\sigma_\zeta$ and $\zeta(b_i\sigma) = b_i\sigma_\zeta$. Moreover, $\zeta(s)|_p = \zeta(s|_p) = f(\vec{\ell}, \langle c_1, \dots, c_{m_f}\rangle[\top]_i)\sigma_\zeta$ and $\zeta(u) = \zeta(s)[\zeta(\mathsf{label}(r)\sigma)]_p$ with $\zeta(\mathsf{label}(r))\sigma_\zeta = \xi_\top(r)\sigma_\zeta$ by Lemma 21. So it suffices if $f(\vec{\ell}, \langle c_1, \dots, c_{m_f}\rangle[\top]_i)\sigma_\zeta \to^*_{\Xi(\mathcal{R}),\mu} \xi_\top(r)\sigma_\zeta$ with cost M. If $k = 0$ we can use rule (1_ρ). Otherwise, we use the reductions $\xi_\top(a_i)\sigma_\zeta \to^*_{\Xi(\mathcal{R}),\mu} b_i\sigma_\zeta$, rules (2_ρ) and (3_ρ), and $k-1$ times a rule of type (4_ρ) to obtain

$$\begin{aligned}
f(\vec{\ell}, \langle c_1, \dots, c_{m_f}\rangle[\top]_i)\sigma_\zeta &\to_{\Xi(\mathcal{R}),\mu} f_{\mathsf{a}}(\vec{\ell}, \langle c_1, \dots, c_{m_f}\rangle[c^1_\rho(\vec{\ell}, \xi_\top(a_1))]_i)\sigma_\zeta \\
&\to^*_{\Xi(\mathcal{R}),\mu} f_{\mathsf{a}}(\vec{\ell}, \langle c_1, \dots, c_{m_f}\rangle[c^1_\rho(\vec{\ell}, b_1)]_i)\sigma_\zeta \\
&\to_{\Xi(\mathcal{R}),\mu} f_{\mathsf{a}}(\vec{\ell}, \langle c_1, \dots, c_{m_f}\rangle[c^2_\rho(\vec{\ell}, b_1, \xi_\top(a_2))]_i)\sigma_\zeta \\
&\to^*_{\Xi(\mathcal{R}),\mu} \cdots \\
&\to_{\Xi(\mathcal{R}),\mu} f_{\mathsf{a}}(\vec{\ell}, \langle c_1, \dots, c_{m_f}\rangle[c^k_\rho(\vec{\ell}, b_1, \dots, b_k)]_i)\sigma_\zeta \\
&\to_{\Xi(\mathcal{R}),\mu} \xi_\top(r)\sigma_\zeta
\end{aligned}$$

  Note that all steps take place at active positions, and that the steps with rules 2_ρ and 4_ρ are administrative. Therefore, the cost of this reduction equals M.
- The remaining case is a failed step $s \rightharpoonup u$. So there exist substitutions σ and τ, an index $1 \leqslant j < k$, and a linear labeled normal form v which does not unify with b_{j+1} such that $\mathsf{label}(a_i)\sigma \rightharpoonup^* b_i\sigma$ with cost M_i for all $1 \leqslant i \leqslant j$ and $\mathsf{label}(a_{j+1})\sigma \rightharpoonup^* v\tau$ with cost M_{j+1}. We obtain $\zeta(\mathsf{label}(a_i)\sigma) = \xi_\top(a_i)\sigma_\zeta$, $\zeta(b_i\sigma) = b_i\sigma_\zeta$, and $\zeta(s)|_p = f(\vec{\ell}, \langle c_1, \dots, c_{m_f}\rangle[\top]_i)\sigma_\zeta$ like in the preceding case. Moreover, like in the first case, we obtain an anti-pattern $v' \in \mathsf{AP}(b_{j+1})$ such that $\xi_\bot(\mathsf{erase}(v))$ is an instance of v'. We have $\zeta(v\tau) = \zeta(v)\tau_\zeta = \xi_\bot(\mathsf{erase}(v))\tau_\zeta$ by Lemma 21. Hence $\zeta(v\tau)$ is an instance of v'. Consequently,

$$\begin{aligned}
f(\vec{\ell}, \langle c_1, \dots, c_{m_f}\rangle[\top]_i)\sigma_\zeta &\to^*_{\Xi(\mathcal{R}),\mu} f_{\mathsf{a}}(\vec{\ell}, \langle c_1, \dots, c_{m_f}\rangle[c^{j+1}_\rho(\vec{\ell}, b_1, \dots, b_j, \xi_\top(a_{j+1}))]_i)\sigma_\zeta \\
&\to^*_{\Xi(\mathcal{R}),\mu} f_{\mathsf{a}}(\vec{\ell}, \langle c_1, \dots, c_{m_f}\rangle[c^{j+1}_\rho(\vec{\ell}, b_1, \dots, b_j, \zeta(v\tau))]_i)\sigma_\zeta \\
&\to_{\Xi(\mathcal{R}),\mu} f(\vec{\ell}, \langle c_1, \dots, c_{m_f}\rangle[\bot]_i)\sigma_\zeta
\end{aligned}$$

  where the last step uses an administrative rule of type (5_ρ). Again, all steps take place at active positions. Note that $f(\vec{\ell}, \langle c_1, \dots, c_{m_f}\rangle[\bot]_i)\sigma_\zeta = \zeta(f_{R\setminus\{\rho\}}(s_1, \dots, s_n)) = \zeta(u|_p)$.

Hence $\zeta(s) \to^*_{\Xi(\mathcal{R}),\mu} \zeta(u)$ as desired. The cost of this reduction is $M_1 + \cdots + M_{j+1}$, which coincides with the cost M of the step $s \rightharpoonup u$. ◀

Theorem 22 provides a way to establish conditional complexity: If $\Xi(\mathcal{R})$ has complexity $O(\varphi(n))$ then the conditional complexity of $\mathcal{R}$ is at most $O(\varphi(n))$.[2] Although there are no complexity tools yet which take context-sensitivity into account, we can obtain an upper bound by simply ignoring the replacement map. Similarly, although existing tools do not accommodate administrative rules, we can count all rules equally. Since for every non-administrative step reducing a term $f_R(\cdots)$ at the root position, at most (number of rules) $\times$ (greatest number of conditions + 1) administrative steps at the root position can be done, the difference is only a constant factor. Moreover, these rules are an instance of *relative rewriting*, for which advanced complexity methods do exist. Thus, it is likely that there will be direct tool support in the future.

6 Interpretations in $\mathbb{N}$

A common method to derive complexity bounds for a TRS is to use interpretations in $\mathbb{N}$. Such an interpretation $\mathcal{I}$ maps function symbols of arity n to functions from $\mathbb{N}^n$ to $\mathbb{N}$, giving a value $[t]_\mathcal{I}$ for ground terms t, which is shown to decrease in every reduction step. The method is easily adapted to take context-sensitivity and administrative rules into account.

Unfortunately, standard interpretation techniques like polynomial interpretations are ill equipped to deal with exponential bounds. Furthermore, to handle the interleaving behavior of f and f_a, the natural choice for an interpretation is to map both symbols to the same function. However, compatibility with rule (2_ρ) then gives rise to the constraint $[\top]_\mathcal{I} \geqslant [c^1_\rho(\ell_1, \ldots, \ell_n, \xi_\top(a_1))]^\alpha_\mathcal{I}$ regardless of the assignment α for the variables in a_1, which is virtually impossible to satisfy. Therefore, we propose a new interpretation-based method which is not subject to this weakness.

Let $\mathbb{B} = \{0, 1\}$. We define relations $\geqslant$ and $>$ on $\mathbb{N} \times \mathbb{B}$ as follows: for $\circ \in \{\geqslant, >\}$, $(n', b') \circ (n, b)$ if $n' \circ n$ and $b' \geqslant b$. Moreover, let π_1 and π_2 be the projections $\pi_1((n, b)) = n$ and $\pi_2((n, b)) = b$.

▶ **Definition 23.** A context-sensitive interpretation over $\mathbb{N} \times \mathbb{B}$ is a function $\mathcal{I}$ mapping each symbol $f \in \mathcal{F}$ of arity n to a function $\mathcal{I}_f$ from $(\mathbb{N} \times \mathbb{B})^n$ to $\mathbb{N} \times \mathbb{B}$, such that $\mathcal{I}_f$ is strictly monotone in its i-th argument for all $i \in \mu(f)$. Given a valuation α mapping each variable to an element of $\mathbb{N} \times \mathbb{B}$, the value $[t]^\alpha_\mathcal{I} \in \mathbb{N} \times \mathbb{B}$ of a term t is defined as usual ($[x]^\alpha_\mathcal{I} = \alpha(x)$ and $[f(t_1, \ldots, t_n)]^\alpha_\mathcal{I} = \mathcal{I}_f([t_1]^\alpha_\mathcal{I}, \ldots, [t_n]^\alpha_\mathcal{I})$). We say $\mathcal{I}$ is compatible with $\mathcal{R}$ if for all $\ell \to r \in \mathcal{R}$ and valuations α, $[\ell]^\alpha_\mathcal{I} > [r]^\alpha_\mathcal{I}$ if $\ell \to r \in \mathcal{R}$ is non-administrative and $[\ell]^\alpha_\mathcal{I} \geqslant [r]^\alpha_\mathcal{I}$ otherwise.

The primary purpose of the second component of (n, b) is to allow more sophisticated choices for $\mathcal{I}$. We easily see that if $s \to_{\mathcal{R}\,\mu} t$ then $[s]^\alpha_\mathcal{I} \geqslant [t]^\alpha_\mathcal{I}$ and $[s]^\alpha_\mathcal{I} > [t]^\alpha_\mathcal{I}$ if the employed rule is non-administrative. Consequently, $\mathsf{dh}(s, \to_{\mathcal{R},\mu}) \leqslant \pi_1([s]^\alpha_\mathcal{I})$ for any valuation α.

[2] We suspect that the equivalence goes both ways: If the derivation height of a term s in $\Xi(\mathcal{R})$ is N then a complexity-conscious reduction of length at least N exists starting at $\mathsf{label}(s)$. However, we have not yet confirmed this proposition as the proof—which is based on swapping rule applications at different positions—is non-trivial and Theorem 22 provides the direction which is most important in practice.

▶ **Example 24.** Continuing Example 19, we define

$$\mathcal{I}_\top = (0,1) \qquad \mathcal{I}_\bot = I_{\mathsf{true}} = \mathcal{I}_{\mathsf{false}} = \mathcal{I}_{\mathsf{0}} = (0,0) \qquad \mathcal{I}_{\mathsf{s}}((x,b)) = (x+1,0)$$
$$\mathcal{I}_{\mathsf{even}}((x,b),(y_1,b_1),(y_2,b_2),(y_3,b_3)) = (1+x+y_1+y_2+y_3+b_2\cdot 3^x+b_3\cdot 3^x,0)$$
$$\mathcal{I}_{\mathsf{odd}} = \mathcal{I}_{\mathsf{even_a}} = \mathcal{I}_{\mathsf{odd_a}} = \mathcal{I}_{\mathsf{even}} \qquad \mathcal{I}_{\mathsf{c}_i^1}((x,b),(y,d)) = (y,0) \quad \text{for all } i \in \{2,3,5,6\}$$

One easily checks that $\mathcal{I}$ satisfies the required monotonicity constraints. Moreover, all rewrite rules in $\Xi(\mathcal{R}_{\mathsf{even}})$ are oriented as required. For instance, for rule (2_2) we obtain

$$\begin{aligned}1+(x+1)+y+0+z+1\cdot 3^{x+1}+b_z\cdot 3^{x+1}&\\ \geqslant 1+(x+1)+y+(1+x+0+0+0+3^x+3^x)+z+0+b_z\cdot 3^{x+1}&\end{aligned}$$

which holds for all $x,y,z \in \mathbb{N}$ and $b_z \in \mathbb{B}$. All constructor terms are interpreted by linear polynomials with coefficients in $\{0,1\}$ and hence $\pi_1([s]_\mathcal{I}) \leqslant |t|$ for all ground constructor terms t. Therefore, the conditional runtime complexity $\mathsf{crc}_{\mathcal{R}_{\mathsf{even}}}(n)$ is bound by

$$\begin{aligned}&\max(\{\pi_1(\mathcal{I}_f((x_1,b_1),\dots,(x_4,b_4))) \mid f \in \mathcal{F}_\mathcal{D} \text{ and } x_1+x_2+x_3+x_4 < n\})\\ =\ &\max(\{1+x_1+x_2+x_3+x_4+2\cdot 3^{x_1} \mid x_1+x_2+x_3+x_4 < n\}) \leqslant 3^n\end{aligned}$$

As observed before, the actual runtime complexity for this system is $O(2^n)$. In order to obtain this more realistic bound, we might observe that, starting from a basic term s, if $s \to^* t$ then the first argument of even and odd anywhere in t cannot contain defined symbols. Therefore, $\mathcal{I}_{\mathsf{even}}$ does not need to be monotone in its first argument. This observation is based on a result in [11], which makes it possible to impose a stronger replacement map μ.

As to derivational complexity, we observe that $[t]_\mathcal{I} \leqslant {}^n3$ (tetration, or $3\uparrow\uparrow n$ in Knuth's up-arrow notation) when t is an arbitrary ground term of size n. To obtain a more elementary bound, we will need more sophisticated methods, for instance assigning a compatible sort system and using the fact that all terms of sort int are necessarily constructor terms.

The interpretations in Example 24 may appear somewhat arbitrary, but in fact there is a recipe that we can most likely apply to many TRSs obtained from a CCTRS. The idea is to define the interpretation $\mathcal{I}$ as an extension of a "basic" interpretation $\mathcal{J}$ over $\mathbb{N}$. To do so, we choose for every symbol f of arity n in the original signature $\mathcal{F}$ interpretation functions $\mathcal{J}_f^0,\dots,\mathcal{J}_f^{m_f}\colon \mathbb{N}^n \to \mathbb{N}$ such that $\mathcal{J}_f^0$ is strictly monotone in all its arguments, and the other $\mathcal{J}_f^j$ are weakly monotone. Similarly, for each rule ρ with $k > 0$ conditions we fix interpretation functions $\mathcal{J}_{c,\rho}^1$, …, $\mathcal{J}_{c,\rho}^k$ with $\mathcal{J}_{c,\rho}^i\colon \mathbb{N}^{n+i} \to \mathbb{N}$ if c_ρ^i has arity $n+i$. These functions must be strictly monotone in the last argument. Based on these interpretations, we fix an interpretation for $\mathcal{G}$: $\mathcal{I}_\top = (0,1)$ and $\mathcal{I}_\bot = (0,0)$,

$$\mathcal{I}_f((x_1,b_1),\dots,(x_n,b_n)) = (\mathcal{J}_f^0(x_1,\dots,x_n),0)$$

for every $f \in \mathcal{F}_\mathcal{C}$ of arity n,

$$\begin{aligned}&\mathcal{I}_f((x_1,b_1),\dots,(x_n,b_n),(y_1,d_1),\dots,(y_{m_f},d_{m_f}))\\ &\qquad = (\mathcal{J}_f^0(x_1,\dots,x_n)+y_1+\dots+y_{m_f}+\sum_{i=1}^{m_f}(d_i\cdot\mathcal{J}_f^i(x_1,\dots,x_n)),0)\end{aligned}$$

and $\mathcal{I}_{f_\mathsf{a}} = \mathcal{I}_f$ for every $f \in \mathcal{F}_\mathcal{D}$ of arity n, and

$$\mathcal{I}_{c_\rho^i}((x_1,b_1),\dots,(x_{n+i},b_{n+i})) = (\mathcal{J}_{c,\rho}^i(x_1,\dots,x_{n+i}),0)$$

for every symbol c^i_ρ. It is not hard to see that $\mathcal{I}$ satisfies the monotonicity requirements. In practice, this interpretation assigns to a $\top$ symbol in a term $f(s_1,\dots,s_n,\dots,\top,\dots)$ the value $\mathcal{J}^i_f(s_1,\dots,s_n)$ and ignores the $\mathbb{B}$ component otherwise. Using this interpretation for the rules in Definition 18, the inequalities we obtain can be greatly simplified. Obviously, $[\ell]^\alpha_{\mathcal{I}} \geqslant [r]^\alpha_{\mathcal{I}}$ is satisfied for all rules obtained from clauses (5_ρ) and (6_ρ). For the other clauses we obtain the following requirements for each rule $\rho\colon f(\ell_1,\dots,\ell_n) \to r \Leftarrow a_1 \approx b_1,\dots,a_k \approx b_k$ in the original system $\mathcal{R}$, with ρ the i-th rule in $\mathcal{R}^\uparrow f$:

$$\mathcal{J}^0_f([\ell_1]),\dots,[\ell_n]) + \mathcal{J}^i_f([\ell_1],\dots,[\ell_n]) > \pi_1([\xi_\top(r)]^\alpha_{\mathcal{I}}) \tag{1_ρ}$$

$$\mathcal{J}^i_f([\ell_1],\dots,[\ell_n]) \geqslant \mathcal{J}^1_{c,\rho}([\ell_1],\dots,[\ell_n],\pi_1([\xi_\top(r)]^\alpha_{\mathcal{I}})) \tag{2_ρ}$$

$$\mathcal{J}^0_f([\ell_1],\dots,[\ell_n]) + \mathcal{J}^k_{c,\rho}([\ell_1],\dots,[\ell_n],[b_1],\dots,[b_k]) > \pi_1([\xi_\top(r)]^\alpha_{\mathcal{I}}) \tag{3_ρ}$$

$$\mathcal{J}^j_f([\ell_1],\dots,[\ell_n],[b_1],\dots,[b_j]) \geqslant \mathcal{J}^{j+1}_f([\ell_1],\dots,[\ell_n],[b_1],\dots,[b_j],\pi_1([\xi_\top(r)]^\alpha_{\mathcal{I}})) \tag{4_ρ}$$

for the same cases of k and j as in Definition 18. Here, $[l_j]$ and $[b_j]$ are short-hand notation for $\pi_1([l_j]^\alpha_{\mathcal{I}})$ and $\pi_1([b_j]^\alpha_{\mathcal{I}})$, respectively. Note that $[f(t_1,\dots,t_n)] = \mathcal{J}_f([t_1],\dots,[t_n])$ for constructor terms $f(s_1,\dots,s_n)$. Additionally observing that

$$\pi_1([\xi_\top(f(t_1,\dots,t_n))]^\alpha_{\mathcal{I}}) = \sum_{i=0}^{m_f} \mathcal{J}^i_f(\pi_1([\xi_\top(t_1)]^\alpha_{\mathcal{I}}),\dots,\pi_1([\xi_\top(t_n)]^\alpha_{\mathcal{I}}))$$

we can obtain bounds for the derivation height without ever calculating $\xi_\top(t)$.

▶ **Example 25.** We derive an upper bound for the runtime complexity of $\mathcal{R}_{\mathsf{fib}}$. Following the recipe explained above, we fix $\mathcal{J}^1_+(x,y) = \mathcal{J}^2_+(x,y) = \mathcal{J}^1_{\mathsf{fib}}(x) = 0$. Writing $K = \mathcal{J}^0_{\mathsf{0}}$, $S = \mathcal{J}^0_{\mathsf{s}}$, $P = \mathcal{J}^0_+$, $F = \mathcal{J}^0_{\mathsf{fib}}$, $G = \mathcal{J}^2_{\mathsf{fib}}$, $A = \mathcal{J}^0_{\langle\cdot\rangle}$, $C = \mathcal{J}^1_{c,4}$, and $D = \mathcal{J}^2_{c,4}$, we get the constraints

$$P(K,y) > y \qquad P(S(x),y) > S(P(x,y)) \qquad F(0) > A(0,S(0))$$

for the unconditional rules of $\mathcal{R}_{\mathsf{fib}}$ and

$$G(S(x)) \geqslant C(S(x),F(x)+G(x))$$
$$F(S(x)) + D(S(x),A(y,z),w) > A(z,w)$$
$$C(S(x),A(y,z)) \geqslant D(S(x),A(y,z),P(y,z))$$

for the conditional rule $\mathsf{fib}(\mathsf{s}(x)) \to \langle z,w\rangle \Leftarrow \mathsf{fib}(x) \approx \langle y,z\rangle,\ y+z \approx w$. The functions P, F, A, and S must be strictly monotone in all arguments, whereas for C and D strict monotonicity is required only for the last argument. Choosing $K = 0$, $S(x) = x+1$, $P(x,y) = 2x+y+1$, $A(x,y) = x+y+1$, $C(x,y) = 3y$, and $D(x,y,z) = y+z$ to eliminate as many arguments as possible, the constraints simplify to

$$F(0) > 3 \qquad G(x+1) \geqslant 3\,F(x) + 3\,G(x) \qquad F(x+1) > 0$$

Choosing $F(x) = x+4$ leaves the constraint $G(x+1) \geqslant 3x + 3\,G(x) + 12$, which is satisfied (e.g.) by taking $G(x) = 4^{x+1}$, which results in a conditional runtime complexity of $O(4^x)$.

As in Example 24, we can obtain a more precise bound using [11], by observing that runtime complexity is not altered if we impose a replacement map μ with $\mu(\mathsf{fib}) = \varnothing$, which allows us to choose a non-monotone function for F. More sophisticated methods may lower the bound further.

7 Related Work

We are not aware of any attempt to study the complexity of conditional rewriting, but numerous transformations from CTRSs to TRSs have been proposed in the literature. They can roughly be divided into so-called *unravelings* and *structure-preserving* transformations. The former were coined by Marchiori [17] and have been extensively investigated (e.g. [18, 21, 23, 24, 26]), mainly to establish (operational) termination and confluence of the input CTRS. The latter originate from Viry [29] and improved versions were proposed in [1, 7, 9].

The transformations that are known to transform CTRSs into TRSs such that (simple) termination of the latter implies quasi-decreasingness of the former, are natural candidates for study from a complexity perspective. We observe that unravelings are not suitable in this regard. For instance, the unraveling from [18] transforms the CCTRS $\mathcal{R}_{\mathsf{even}}$ into

$$\begin{array}{lll}
\mathsf{even}(0) \to \mathsf{true} & \mathsf{even}(\mathsf{s}(x)) \to \mathsf{U}_1(\mathsf{odd}(x), x) & \mathsf{U}_1(\mathsf{true}, x) \to \mathsf{true} \\
& \mathsf{even}(\mathsf{s}(x)) \to \mathsf{U}_2(\mathsf{even}(x), x) & \mathsf{U}_2(\mathsf{true}, x) \to \mathsf{false} \\
\mathsf{odd}(0) \to \mathsf{false} & \mathsf{odd}(\mathsf{s}(x)) \to \mathsf{U}_3(\mathsf{odd}(x), x) & \mathsf{U}_3(\mathsf{true}, x) \to \mathsf{false} \\
& \mathsf{odd}(\mathsf{s}(x)) \to \mathsf{U}_4(\mathsf{even}(x), x) & \mathsf{U}_4(\mathsf{true}, x) \to \mathsf{true}
\end{array}$$

This TRS has a linear runtime complexity, which is readily confirmed by a complexity tool like TCT [2]. As the conditional runtime complexity is exponential, the transformation is not suitable for measuring conditional complexity. The same holds for the transformation in [3].

We do not know whether structure-preserving transformations can be used for conditional complexity. If we apply the transformations from [7] and [9] to $\mathcal{R}_{\mathsf{even}}$ we obtain TRSs for which complexity tools fail to establish an upper bound on the runtime and derivational complexities. The latter is also true for the TRS that we obtain from $\Xi(\mathcal{R}_{\mathsf{even}})$ by lifting the context-sensitive restriction, but this is solely due to the (current) lack of support in complexity tools for techniques that yield non-polynomial upper bounds.

8 Conclusion and Future Work

In this paper we have defined a first notion of complexity for conditional term rewriting, which takes failed calculations into account as any automatic rewriting engine would. We have also defined a transformation to unconditional context-sensitive TRSs, and shown how this transformation can be used to find upper bounds for conditional complexity using traditional interpretation-based methods.

There are several possible directions to continue our research.

Weakening restrictions. An obvious direction for future research is to broaden the class of CTRSs we consider. This requires careful consideration. The correctness of the transformation Ξ depends on the limitations that we impose on CCTRSs. However, it may be possible to weaken the restrictions and still obtain at least a sound (if perhaps not complete) transformation. More importantly, though, as discussed in Section 2, the restrictions on the conditions are needed to justify our complexity notion. For the same reason, Lemma 5 (which relies on the left-hand sides being linear basic terms) needs to be preserved.

Alternatively, we might consider different strategies for the evaluation of conditions. For example, if all restrictions except for variable freshness in the conditions are satisfied, we could impose the strategy that conditions are always evaluated to normal form. For instance, given a CTRS with a rule

$$\mathsf{f}(y, z) \to r \Leftarrow y \approx \mathsf{g}(x), z \approx x$$

rewriting a term $\mathsf{f}(\mathsf{g}(0+0),0)$ would then bind 0 rather than $0+0$ to x in the first condition (assuming sensible rules for $+$), allowing the second condition to succeed. This approach could also be used to handle non-left-linear rules, for instance transforming a rule $\mathsf{f}(\mathsf{g}(x),x) \to r$ into the left-linear rule above.

It would take a little more effort to handle non-confluent systems in a way that does not allow us to give up on rule applications when it is still possible that their conditions can be satisfied. As a concrete example, consider the CCTRS

$$\mathsf{a}(x) \to 0 \qquad \mathsf{a}(x) \to 1 \qquad \mathsf{f}(x) \to \mathsf{g}(y,y) \Leftarrow \mathsf{a}(x) \approx y \qquad \mathsf{h}(x) \to x \Leftarrow \mathsf{f}(x) \approx \mathsf{g}(0,1)$$

Even though $\mathsf{f}(0) \to^* \mathsf{g}(0,0)$ and $\mathsf{g}(0,0)$ is an instance of $\mathsf{g}(x,0) \in \mathsf{AP}(\mathsf{g}(0,1))$, we should not conclude that $\mathsf{h}(0)$ cannot be reduced. We could instead attempt to calculate all normal forms in order to satisfy conditions, but if we do this for the third rule, we would not find the desired reduction $\mathsf{f}(0) \to \mathsf{g}(\mathsf{a}(0),\mathsf{a}(0))$. Thus, to confirm that $\mathsf{h}(0)$ can be reduced, we need to determine *all* (or at least, all *most general*) reducts of the left-hand sides of conditions. This will likely give very high complexity bounds, however. It would be interesting to investigate how real conditional rewrite engines like Maude handle this problem.

Rules with branching conditions. Consider the following variant of $\mathcal{R}_{\mathsf{even}}$:

$$\mathsf{even}(0) \to \mathsf{true} \quad (1) \qquad \mathsf{odd}(0) \to \mathsf{false} \quad (4)$$
$$\mathsf{even}(\mathsf{s}(x)) \to \mathsf{true} \Leftarrow \mathsf{odd}(x) \approx \mathsf{true} \quad (2) \qquad \mathsf{odd}(\mathsf{s}(x)) \to \mathsf{true} \Leftarrow \mathsf{even}(x) \approx \mathsf{true} \quad (5)$$
$$\mathsf{even}(\mathsf{s}(x)) \to \mathsf{false} \Leftarrow \mathsf{odd}(x) \approx \mathsf{false} \quad (3) \qquad \mathsf{odd}(\mathsf{s}(x)) \to \mathsf{false} \Leftarrow \mathsf{even}(x) \approx \mathsf{false} \quad (6)$$

Evaluating $\mathsf{even}(\mathsf{s}^9(0))$ with rule (2) causes the calculation of the normal form false of $\mathsf{odd}(\mathsf{s}^8(0))$, before concluding that the rule does not apply. In our definitions (of $\rightharpoonup$ and Ξ), and conform to the behavior of Maude, we would dismiss the result and continue trying the next rule. In this case, that means recalculating the normal form of $\mathsf{odd}(\mathsf{s}^8(0))$, but now to verify whether rule (3) applies. There is clearly no advantage in treating the rules (2) and (3) separately. Instead, we could consider rules such as these to be *grouped*; if the left-hand side matches, we try the corresponding condition, and the result determines whether we proceed with (2), (3), or fail. Future definitions of complexity for *sensible* conditional rewriting should take optimizations like these into account.

Improving the transformation. With regard to the transformation Ξ, it should be possible to obtain smaller resulting systems using various optimizations, such as reducing the set AP of anti-patterns using typing considerations, or leaving defined symbols untouched when they are only defined by unconditional rules. As observed in footnote 2, either proving that Ξ preserves complexity, or improving it so that it does, would be interesting.

Complexity methods. While the interpretation recipe from Section 6 has the advantage of immediately eliminating rules (5_ρ) and (6_ρ), it is not strictly necessary to always map f and f_{a} to the same function. With alternative recipes, we may be able to more fully take advantage of the context-sensitivity of the transformed system, and handle different examples.

Besides interpretations into $\mathbb{N}$, there are many other complexity techniques which could possibly be adapted to context-sensitivity and to handle the $\top$ symbols appearing in $\Xi(\mathcal{R})$. As for tool support, we believe that it should not be hard to integrate support for conditional rewriting into existing tools. We hope that, in the future, developers of complexity tools will branch out to context-sensitive rewriting and not shy away from exponential upper bounds.

Acknowledgments. We thank the anonymous reviewers, whose constructive comments have helped to improve the presentation.

References

1 S. Antoy, B. Brassel, and M. Hanus. Conditional narrowing without conditions. In *Proc. 5th PPDP*, pages 20–31, 2003. doi:10.1145/888251.888255.

2 M. Avanzini and G. Moser. Tyrolean complexity tool: Features and usage. In *Proc. 24th RTA*, volume 21 of *Leibniz International Proceedings in Informatics*, pages 71–80, 2013. doi:10.4230/LIPIcs.RTA.2013.71.

3 J. Avenhaus and C. Loría-Sáenz. On conditional rewrite systems with extra variables and deterministic logic programs. In *Proc. 5th LPAR*, volume 822 of *LNAI*, pages 215–229, 1994. doi:10.1007/3-540-58216-9_40.

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

5 G. Bonfante, A. Cichon, J.-Y. Marion, and H. Touzet. Algorithms with polynomial interpretation termination proof. *JFP*, 11(1):33–53, 2001.

6 M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. *All About Maude – A High-Performance Logical Framework*, volume 4350 of *LNCS*. Springer, 2007.

7 T.F. Şerbănuţă and G. Roşu. Computationally equivalent elimination of conditions. In *Proc. 17th RTA*, volume 4098 of *LNCS*, pages 19–34, 2006. doi:10.1007/11805618_3.

8 J. Giesl, M. Brockschmidt, F. Emmes, F. Frohn, C. Fuhs, C. Otto, M. Plücker, P. Schneider-Kamp, S. Swiderski, and R. Thiemann. Proving termination of programs automatically with AProVE. In *Proc. 7th IJCAR*, volume 8562 of *LNCS*, pages 184–191, 2014. doi:10.1007/978-3-319-08587-6_13.

9 K. Gmeiner and N. Nishida. Notes on structure-preserving transformations of conditional term rewrite systems. In *Proc. 1st WPTE*, volume 40 of *OASICS*, pages 3–14, 2014. doi:10.4230/OASIcs.WPTE.2014.3.

10 N. Hirokawa and G. Moser. Automated complexity analysis based on the dependency pair method. In *Proc. 4th IJCAR*, volume 5195 of *LNAI*, pages 364–380, 2008. doi:10.1007/978-3-540-71070-7_32.

11 N. Hirokawa and G. Moser. Automated complexity analysis based on context-sensitive rewriting. In *Proc. Joint 25th RTA and 12th TLCA*, volume 8560 of *LNCS*, pages 257–271, 2014. doi:10.1007/978-3-319-08918-8_18.

12 D. Hofbauer and C. Lautemann. Termination proofs and the length of derivations (preliminary version). In *Proc. 3rd RTA*, volume 355 of *LNCS*, pages 167–177, 1989. doi:10.1007/3-540-51081-8_107.

13 S. Lucas. Context-sensitive computations in functional and functional logic programs. *Journal of Functional and Logic Programming*, 1998(1), 1998.

14 S. Lucas. Context-sensitive rewriting strategies. *Information and Computation*, 178(1):294–343, 2002. doi:10.1006/inco.2002.3176.

15 S. Lucas, C. Marché, and J. Meseguer. Operational termination of conditional term rewriting systems. *Information Processing Letters*, 95(4):446–453, 2005. doi:10.1016/j.ipl.2005.05.002.

16 S. Lucas and J. Meseguer. 2D dependency pairs for proving operational termination of CTRSs. In *Proc. 10th WRLA*, volume 8663 of *LNCS*, pages 195–212, 2014. doi:10.1007/978-3-319-12904-4_11.

17 M. Marchiori. Unravelings and ultra-properties. In M. Hanus and M. Rodríguez-Artalejo, editors, *Proc. 5th ICALP*, volume 1139 of *LNCS*, pages 107–121. Springer, 1996. doi:10.1007/3-540-61735-3_7.

18 M. Marchiori. On deterministic conditional rewriting. Computation Structures Group Memo 405, MIT Laboratory for Computer Science, 1997.

19 A. Middeldorp, G. Moser, F. Neurauter, J. Waldmann, and H. Zankl. Joint spectral radius theory for automated complexity analysis of rewrite systems. In *Proc. 4th CAI*, volume 6742 of *LNCS*, pages 1–20, 2011. doi:10.1007/978-3-642-21493-6_1.

20 G. Moser and A. Schnabl. The derivational complexity induced by the dependency pair method. *Logical Methods in Computer Science*, 7(3), 2011. doi:10.2168/LMCS-7(3:1)2011.

21 N. Nishida, M. Sakai, and T. Sakabe. Soundness of unravelings for conditional term rewriting systems via ultra-properties related to linearity. *Logical Methods in Computer Science*, 8:1–49, 2012. doi:10.2168/LMCS-8(3:4)2012.

22 L. Noschinski, F. Emmes, and J. Giesl. Analyzing innermost runtime complexity of term rewriting by dependency pairs. *Journal of Automated Reasoning*, 51(1):27–56, 2013. doi:10.1007/s10817-013-9277-6.

23 E. Ohlebusch. Transforming conditional rewrite systems with extra variables into unconditional systems. In *Proc. 6th LPAR*, volume 1705 of *LNCS*, pages 111–130, 1999. doi:10.1007/3-540-48242-3_8.

24 E. Ohlebusch. *Advanced Topics in Term Rewriting*. Springer, 2002. doi:10.1007/978-1-4757-3661-8.

25 F. Schernhammer and B. Gramlich. VMTL – a modular termination laboratory. In *Proc. 20th RTA*, volume 5595 of *LNCS*, pages 285–294, 2009. doi:10.1007/978-3-642-02348-4_20.

26 F. Schernhammer and B. Gramlich. Characterizing and proving operational termination of deterministic conditional term rewriting systems. *Journal of Logic and Algebraic Programming*, 79(7):659–688, 2010. doi:10.1016/j.jlap.2009.08.001.

27 T. Sternagel and A. Middeldorp. Conditional confluence (system description). In *Proc. Joint 25th RTA and 12th TLCA*, volume 8560 of *LNCS*, pages 456–465, 2014. doi:10.1007/978-3-319-08918-8_31.

28 Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.

29 P. Viry. Elimination of conditions. *Journal of Symbolic Computation*, 28(3):381–401, 1999. doi:10.1006/jsco.1999.0288.

30 H. Zankl and M. Korp. Modular complexity analysis for term rewriting. *Logical Methods in Computer Science*, 10(1:19):1–33, 2014. doi:10.2168/LMCS-10(1:19)2014.