Document downloaded from:

http://hdl.handle.net/10251/81584

This paper must be cited as:

Lucas Alba, S.; Meseguer, J. (2016). Normal forms and normal theories in conditional rewriting. Journal of Logical and Algebraic Methods in Programming. 85(1):67-97. doi:10.1016/j.jlamp.2015.06.001.

The final publication is available at

http://dx.doi.org/10.1016/j.jlamp.2015.06.001

Copyright Elsevier

Additional Information

this is the author's version of a work that was accepted for publication in Journal of Logical and Algebraic Methods in Programming. Changes resulting from the publishing process, such as peer review, editing, corrections, structural formatting, and other quality control mechanisms may not be reflected in this document. Changes may have been made to this work since it was submitted for publication. A definitive version was subsequently published in Journal of Logical and Algebraic Methods in Programming vol. 85 (2016) DOI 10.1016/j.jlamp.2015.06.001

Normal forms and normal theories in conditional rewriting

Salvador Lucas[a,b], José Meseguer[a]

[a] *CS Dept., University of Illinois at Urbana-Champaign*
[b] *DSIC, Universitat Politècnica de València*

Abstract

We present several new concepts and results on conditional term rewriting within the general framework of order-sorted rewrite theories (OSRTs), which support types, subtypes and rewriting modulo axioms, and contains the more restricted framework of conditional term rewriting systems (CTRSs) as a special case. The concepts shed light on several subtle issues about conditional rewriting and conditional termination. We point out that the notions of *irreducible* term and of *normal form*, which coincide for unconditional rewriting, have been conflated for conditional rewriting but are in fact *totally different* notions. Normal form is a *stronger* concept. We call any rewrite theory where all irreducible terms are normal forms a *normal* theory. We argue that normality is essential to have good executability and computability properties. Therefore we call all other theories *abnormal*, freaks of nature to be avoided. The distinction between irreducible terms and normal forms helps in clarifying various notions of strong and weak termination. We show that abnormal theories can be terminating in various, equally abnormal ways; and argue that any computationally meaningful notion of strong or weak conditional termination should be a property of normal theories. In particular we define the notion of a *weakly operationally terminating* (or *weakly normalizing*) OSRT, discuss several evaluation mechanisms to compute normal forms in such theories, and investigate general conditions under which the rewriting-based operational semantics and the initial algebra semantics of a confluent, weakly normalizing OSRT coincide thanks to a notion of *canonical term algebra*. Finally, we investigate appropriate conditions and proof methods to ensure that a rewrite theory is normal; and characterize the stronger property of a rewrite theory being operationally terminating in terms of a natural generalization of the notion of quasi-decreasing order.

Keywords: Conditional term rewriting, normal forms, normal theory, operational termination, rewriting logic, Maude.

1. Introduction

This paper presents several new contributions to conditional term rewriting and to the semantics of declarative, rewriting-based languages. Conditional

rewriting is considered within the general and highly expressive framework of *order-sorted rewrite theories* (OSRTs), that is, theories $\mathcal{R} = (\Sigma, B, R)$, where (Σ, B) is an order-sorted equational theory [17, 7] with equational axioms B, and R is a collection of conditional rewrite rules with oriented conditions of the form: $\ell \to r \Leftarrow s_1 \to t_1, \ldots, s_n \to t_n$, which are applied *modulo* B. All the results are *in particular* new results for *Conditional Term Rewriting Systems* (CTRSs); that is, for order-sorted rewrite theories of the special form $\mathcal{R} = (\Sigma, \varnothing, R)$, with $B = \varnothing$ and Σ unsorted, i.e., having a *single* sort. The point of using OSRTs is, of course, that not only the CTRS-based syntactic rewriting, but the more general rewriting modulo axioms B is thus supported; and that, for obvious reasons of expressiveness, all well-known rule-based languages, e.g., [8, 23, 6, 3], support not only rewriting modulo axioms, but also types and, often, subtypes. Therefore, the greater generality of OSRTs is not a caprice, but an absolute necessity for making formal specification and declarative programming expressive and practical.

Our contributions consist in asking and providing detailed answers to the following, innocent-sounding questions:

1. What is the right notion of *normal form* for an OSRT?
2. What is the right notion of *weak operational termination* for an OSRT?
3. Under what conditions can OSRTs be used as *declarative programs* having a well-behaved semantics? How can we *execute* such programs? How can their executability conditions be checked in practice?
4. Under what conditions does a confluent OSRT have a *canonical term algebra* that can be effectively computed and that provides a *complete agreement* between the operational semantics of the OSRT as a functional program, and its mathematical, initial algebra semantics?
5. Can the operational termination of OSRTs be characterized in terms of orders?

Surprisingly enough, some of these questions seem to never have been asked. At best, the issues involved seem to have remained implicit as not well-understood, anomalous features in the literature. Consider, for example, question (1) above, which asks about the notion of normal form. For unconditional term rewriting the notion is absolutely clear and unproblematic: a normal form is a term t that is *irreducible*, that is, such that there is no t' with $t \to_{\mathcal{R}} t'$. To the best of our knowledge, all the CTRS literature is unanimous in identifying normal forms with irreducible terms also in the conditional case. That is, the terms "normal form" and "irreducible term" are used with the *same* meaning by all authors.[1] However, for an OSRT, and in particular for a CTRS, the notion of normal

[1] Nevertheless, in the seminal paper [2], for $\mathcal{R}$ a confluent and orthogonal CTRS a distinction is made between the set $\mathsf{Irr}(\mathcal{R})$ of its irreducible terms (called there "normal forms"), and a subset $\mathsf{Irr}_f(\mathcal{R}) \subseteq \mathsf{Irr}(\mathcal{R})$ of irreducible terms of "finite order." However, as we further explain in Section 3.1, the notion of "irreducible term of finite order" is too weak to capture the intuitive notion of normal form, namely, a term that is the *result* of the term normalization process.

form is actually highly problematic. The big problem is that for an OSRT there can be terms t that are irreducible in the above sense, i.e., there is no t' with $t \rightarrow_{\mathcal{R}} t'$, but such that when we give t to a rewrite engine for evaluation such an engine loops! For a trivial example, consider the single conditional rewrite rule $a \rightarrow b \Leftarrow a \rightarrow c$. Since the rewrite relation defined by this conditional rule is the empty set, the constant a is trivially irreducible; but the proof tree associated to the normalization of a using the CTRS inference system is *infinite* [13], and a rewrite engine that tries to evaluate a will loop when trying to satisfy the rule's condition. Therefore, calling a a *normal form* is a very bad joke, since, intuitively, a term is considered to be a normal form if it is "fully normalized," that is, if it is the *result* of fully evaluating some input term by rewriting; but this is precisely what a in the above example is *not*.

Our answer to this puzzle is to introduce a precise distinction (fully articulated in the paper) between irreducible terms and normal forms: every term in normal form is irreducible, but, as the above example shows, not every irreducible term is a normal form. We call an OSRT *normal*[2] iff every irreducible term is a normal form, and call it *abnormal* otherwise. Abnormal theories, like the one above, are hopeless for executability purposes and should be viewed as monsters in the menagerie of CTRSs and OSRTs.

Termination is quite a subtle issue for OSRTs in general and CTRSs in particular. The distinction between irreducible terms and normal forms helps in clarifying various notions of termination. We show that abnormal theories can be terminating in various, equally abnormal ways; and argue that any computationally meaningful notion of strong or weak conditional termination should be a property of normal theories. Many notions of conditional termination have been proposed (see e.g., [20]), but it is by now well-understood that the most satisfactory notion from a computational point of view is that of *operational termination* [13] (more on this later). Here we ask and answer several questions, further developing this notion. One question is (5) above. For the case of deterministic 3-CTRS we proved in [13] that operational termination is *equivalent* to the order-based notion of *quasi-decreasingness*. In Section 6 we generalize this result to a similar result characterizing operational termination of OSRTs in terms of an (axiom-compatible) term ordering.

Another related question is question (2), which could be more simply rephrased as follows: what is the right notion of weak termination/normalization for OSRTs? Although notions of weak CTRS termination go back at least to [2], as we further explain in Section 3.1 they can be very misleading; that is, they can violate one's most basic intuitions about what termination *means*. Our

[2] Note that this meaning of "normal" is in open conflict with the definition of a *normal CTRS* (see, e.g., [20]) as a CTRS whose rewrite rules R are all of the form $l \rightarrow r$ *if* $\bigwedge_{i=1..n} u_i \rightarrow v_i$ with each v_j ground and, not only R-irreducible, but, furthermore, R_u-irreducible, where R_u is the set of unconditional rules obtained from R by dropping all conditions. Since: (i) the terminology "normal" in, e.g., [20], is not universally accepted (e.g., with an extra orthogonality condition they are instead called type III_n in [2]), and (ii) these two meanings of "normal" are so different, no confusion should arise.

distinction between irreducible terms and normal forms shows that there are in fact *two* possible notions, a computationally ill-behaved one (*weak termination*: every term has a terminating rewrite sequence ending in an irreducible term), and a computationally well-behaved one (*weak operational termination*: every term has a normal form).

The notion of normal OSRT is closely related to question (3), namely, that of executability conditions for declarative, conditional rule-based programs, and their evaluation methods, i.e., their operational semantics. As we explain in Section 4, there are *several* evaluation methods, which become increasingly more efficient as we impose further conditions on the OSRT which we use as our program, and depending on the *meaning* that such a program has.

A rewrite theory $\mathcal{R} = (\Sigma, B, R)$, can have *two* different meanings: (i) a functional meaning, in which a function symbol f is understood as a *function* defined by rules of the form $R = \vec{E}$, which *orient* conditional equations E and are executed modulo B; and (ii) a non-functional meaning, in which a term t is understood (modulo B) as a *state* in a concurrent system, whose local concurrent transitions are specified by the rules R [16]. In the Maude language, the first meaning is supported by *functional modules* of the form $\texttt{fmod}\,(\Sigma, B, E)\,\texttt{endfm}$, where the oriented conditional equations $\vec{E}$ are confluent; and the second meaning is supported by *system modules* of the form[3] $\texttt{mod}\,(\Sigma, B, R)\,\texttt{endm}$.

For *functional* programs specified by an OSRT, that is, for rewrite theories of the form $(\Sigma, B, \vec{E})$, with $\vec{E}$ conditional equations oriented as rewrite rules and executed modulo B, the issue raised in question (4) is not just one of having good executability conditions, but actually of *correctness*. More precisely, of *semantic agreement* between an abstract *initial algebra semantics* when the rules are viewed as equations, and an *operational semantics* based on rewriting, where the computed *values* —that is, the normal forms— give rise to a very intuitive algebra, the *canonical term algebra*, which under the assumptions of confluence, strict coherence, determinism, sort-decreasingness and operational termination is *isomorphic* to the initial algebra of the specification.

Question (4) above asks, essentially: what are the *most general* conditions ensuring this isomorphism and keeping both computability of the canonical term algebra and an exact agreement between mathematical and operational semantics? That is, what are the right conditions for this semantic agreement when we relax the operational termination condition? This is answered in Section 4.4. Last but not least, in Sections 3 and 5 we investigate appropriate *conditions* and *proof methods* to ensure that a theory has good executability properties such as being normal, and evaluation to normal form defining a total recursive function.

This paper is a substantial extension of our conference paper [14]. Besides having been fully reorganized, having more thorough discussions of the key concepts and of related literature, and containing both detailed proofs of all

[3] More generally of the form $\texttt{mod}\,(\Sigma, B \uplus E, R)\,\texttt{endm}$, with (Σ, B, E) satisfying the requirements of a functional module. For the sake of a simpler exposition, in this paper we treat only the case where the only equations involved are the axioms B.

the theorems stated in [14] and more examples, additional new contributions include: (i) a detailed description of the operational semantics of computing normal forms in deterministic normal rewrite theories based on their proof theory; (ii) a discussion of conditions under which such an operational semantics is supported by the Maude system; (iii) a detailed study of computability properties; and (iv) a more detailed treatment of canonical term algebras for normal, strongly deterministic, confluent, sort-decreasing, and weakly normalizing rewrite theories.

2. Preliminaries

To make the paper self-contained, we recall here some basic notions of order-sorted algebra, order-sorted rewrite theories, strict coherence, and operational termination.

Order-Sorted Algebra. We summarize here material from [7, 17] on order-sorted algebra. We start with a partially ordered set $(S, \leq)$ of *sorts*, where $s \leq s'$ is interpreted as *subsort inclusion*. The *connected components* of $(S, \leq)$ are the equivalence classes $[s] \in S/\equiv_{\leq}$, where $\equiv_{\leq} = (\leq \cup \geq)^+$, i.e., $\equiv_{\leq}$ is the smallest equivalence relation containing $\leq$. We also define $\lfloor s \rfloor = \{s' \in S \mid s' \leq s\}$, i.e., the sorts in S which are smaller than or equal to s. When $[s]$ has a top element we denote it by $\top_{[s]}$. An *order-sorted signature* $(\Sigma, S, \leq)$ consists of a poset of sorts $(S, \leq)$ and an $S^* \times S$-indexed family of sets $\Sigma = \{\Sigma_{w,s}\}_{(w,s)\in S^*\times S}$, which are *function symbols* with a given string of argument sorts and a result sort. If $f \in \Sigma_{s_1 \ldots s_n, s}$, we display f as $f : s_1 \cdots s_n \to s$. This is called a *rank* declaration for symbol f. Some of these symbols f can be *subsort-overloaded*, i.e., they can have several rank declarations related in the $\leq$ ordering [7]. To avoid ambiguous terms, we assume that Σ is *sensible*, meaning that if $f : s_1 \cdots s_n \to s$ and $f : s'_1 \cdots s'_n \to s'$ are such that $[s_i] = [s'_i]$, $1 \leq i \leq n$, then $[s] = [s']$. And to make both equational and rewriting deduction simpler we will further assume that Σ is also *kind complete*, meaning that: (i) each connected component $[s]$ has a *top sort*, denoted $\top_{[s]} \in [s]$ and called the *kind* of $[s]$, such that $s' \leq \top_{[s]}$ for each $s' \in [s]$, and (ii) whenever $(f : s_1 \cdots s_n \to s) \in \Sigma$, then we also have a typing of f at the kind level of the form: $(f : \top_{[s_1]} \cdots \top_{[s_n]} \to \top_{[s]}) \in \Sigma$. The kind completeness assumption involves no real loss of generality, since each Σ can be extended into a kind-complete $\Sigma^{\Box}$ so that both equational deduction and rewriting in Σ are are *conservatively extended* to the level of $\Sigma^{\Box}$ [19]. Throughout this paper, Σ will always be assumed *sensible and kind-complete*.

Given an S-sorted set $\mathcal{X} = \{\mathcal{X}_s\}_{s\in S}$ of *mutually disjoint* sets of variables, the family of sets $\mathcal{T}_\Sigma(\mathcal{X}) = \{\mathcal{T}_\Sigma(\mathcal{X})_s\}_{s\in S}$ defines for each $s \in S$ the set $\mathcal{T}_\Sigma(\mathcal{X})_s$ of Σ-*terms* of sort s as the least set such that (i) $\mathcal{X}_s \subseteq \mathcal{T}_\Sigma(\mathcal{X})_s$, (ii) if $s \geq s'$ then $\mathcal{T}_\Sigma(\mathcal{X})_s \supseteq \mathcal{T}_\Sigma(\mathcal{X})_{s'}$, and (iii) for each $f : s_1 \cdots s_n \to s$ and $t_i \in \mathcal{T}_\Sigma(\mathcal{X})_{s_i}$, $1 \leq i \leq n$, $f(t_1, \ldots, t_n) \in \mathcal{T}_\Sigma(\mathcal{X})_s$. The assumption that Σ is sensible ensures that if $[s] \neq [s']$, then $\mathcal{T}_\Sigma(\mathcal{X})_s \cap \mathcal{T}_\Sigma(\mathcal{X})_{s'} = \varnothing$. And kind-completeness implies that $\mathcal{T}_\Sigma(\mathcal{X})_{\top_{[s]}} = \bigcup_{s'\in[s]} \mathcal{T}_\Sigma(\mathcal{X})_{s'}$.

The operations $f : (t_1, \ldots, t_n) \mapsto f(t_1, \ldots, t_n)$, for each $f \in \Sigma_{s_1 \ldots s_n, s}$, on the family $\mathcal{T}_\Sigma(\mathcal{X}) = \{\mathcal{T}_\Sigma(\mathcal{X})_s\}_{s \in S}$ define an order-sorted Σ-algebra called the *free* algebra on $\mathcal{X}$ and denoted, by abuse of language, $\mathcal{T}_\Sigma(\mathcal{X})$. When $\mathcal{X} = \varnothing$, $\mathcal{T}_\Sigma = \mathcal{T}_\Sigma(\varnothing)$ denotes the *initial* Σ-algebra. By further abuse of language, $\mathcal{T}_\Sigma(\mathcal{X})$ also denotes the set of all *well-formed* Σ-terms, that is, the set-theoretic union $\mathcal{T}_\Sigma(\mathcal{X}) = \cup_{s \in S} \mathcal{T}_\Sigma(\mathcal{X})_s$. A simple syntactic condition on $(\Sigma, S, \leq)$ called *preregularity* [7] ensures that each well-formed term t has always a *least sort* possible among all sorts in S, which is denoted $LS(t)$. A *substitution* σ is an S-sorted mapping $\sigma = \{\sigma : \mathcal{X}_s \to \mathcal{T}_\Sigma(\mathcal{X})_s\}_{s \in S}$ from variables to terms. $Subst(\mathcal{T}_\Sigma(\mathcal{X}))$ denotes the set of all such substitutions. The *application* of σ to t (denoted $\sigma(t)$) consists of simultaneously replacing the variables occurring in t terms according to the mapping σ. The *domain* of σ is the set $dom(\sigma) = \{x \in \mathcal{X} \mid \sigma(x) \neq x\}$. A *specialization* ν is an injective substitution that maps a variable x of sort s to a variable x' of sort $s' \leq s$.

A Σ-*equation* has the form $t = t'$, with the sorts of t and t' in the same connected component. A *conditional* Σ-*equation* is an implication $\bigwedge_{1 \leq i \leq n} u_i = v_i \Rightarrow t = t'$, with $n \geq 0$. If $n = 0$ we identify such a conditional equation with the equation $t = t'$. An order-sorted conditional *theory* is a pair (Σ, E) with E a set of conditional Σ-equations. The *models* of (Σ, E) are precisely the order-sorted Σ-*algebras* that satisfy the conditional equations E [7, 17]. Order-sorted conditional equational logic has a sound and complete inference system [17]. Furthermore, the category of Σ-algebras satisfying E has an initial algebra denoted $\mathcal{T}_{\Sigma/E}$. Although Σ-equations need to be explicitly quantified [7, 17], by assuming that Σ has *non-empty sorts*, that is, that for each sort s we have $\mathcal{T}_{\Sigma,s} \neq \varnothing$ we can leave such quantification implicit. For simplicity we will assume throughout that Σ has non-empty sorts. The relation $=_E$ denotes provable equality modulo E, that is, $t =_E t'$ iff $E \vdash t = t'$, where $\vdash$ is the provability relation for order-sorted conditional equational logic [17].

Equality modulo E extends naturally to substitutions: we write $\sigma =_E \tau$ iff for all variables x we have $\sigma(x) =_E \tau(x)$. Given Σ-terms t', t we say that t' E-matches t (with E-match σ) iff there is a substitution σ such that $t' =_E \sigma(t)$. We then call t' an E-*instance* of t. E is said to have a *finitary* E-*matching algorithm* iff $=_E$ is a decidable relation and there is an algorithm that, given any two Σ-terms t', t, generates a finite, complete set of E-matches of t' as an E-instance of t, denoted $Match_E(t', t)$, so that for any other such E-match τ there is a $\sigma \in Match_E(t', t)$ such that $\tau =_E \sigma$.

Order-Sorted Rewrite Theories. An (order-sorted) rewrite rule is an ordered pair (l, r), written $l \to r$, with $l, r \in \mathcal{T}_\Sigma(\mathcal{X})$, and $LS(l) \equiv_\leq LS(r)$.

An *order-sorted rewrite theory* (OSRT) is a triple $\mathcal{R} = (\Sigma, B, R)$, where Σ is an order-sorted signature, B is a set of Σ-equations, and R is a collection of conditional rewrite rules with oriented conditions of the form $\ell \to r \Leftarrow s_1 \to t_1, \ldots, s_n \to t_n$, where $\ell \to r$ and the $s_i \to t_i$ are order-sorted rewrite rules (with $\ell \notin \mathcal{X}_s$ for all $s \in S$), and where the conditions $s_i \to t_i$ are intended to express the reachability of (instances of) t_i from (instances of) s_i.

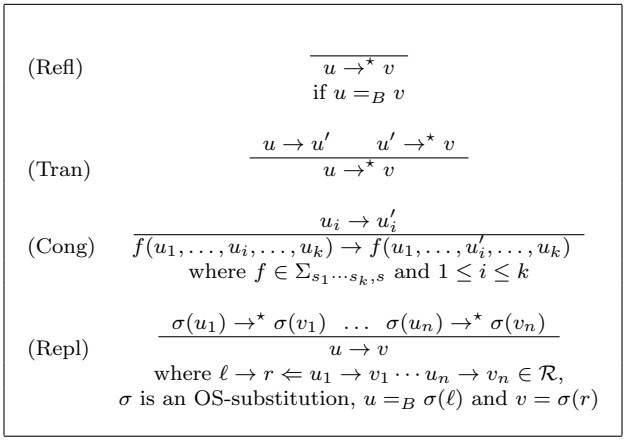


Figure 1: Inference rules for order-sorted rewrite theories

Remark 1. *Throughout this paper we assume the equations $(u = v) \in B$ are:*

1. regular *(i.e., $\mathcal{V}ar(u) = \mathcal{V}ar(v)$);*
2. linear *(i.e., no repeated variables in either u or v);*
3. *having a finitary B-matching algorithm;*
4. sort-preserving *(i.e., for each specialization ν, $LS(\nu(u)) = LS(\nu(v))$); and*
5. *with* top typing*, i.e., each variable x in $\mathcal{V}ar(u) = \mathcal{V}ar(v)$ has top sort $\top_{[s]}$ for some connected component of sorts $[s]$.*

Examples of axioms B satisfying (1)–(3) include combinations of associativity and/or commutativity and/or identity axioms. Maude supports rewriting modulo such axioms and also checks automatically properties (4) (it actually checks a somewhat weaker condition for identity axioms that still ensures a least sort for each B-equivalence class), and (5).

Rewrite rules $\ell \to r \Leftarrow c$ in OSRTs are classified according to the distribution of variables among ℓ, r, and c, as follows: type 1, if $\mathcal{V}ar(r) \cup \mathcal{V}ar(c) \subseteq \mathcal{V}ar(\ell)$; type 2, if $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(\ell)$; type 3, if $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(\ell) \cup \mathcal{V}ar(c)$; and type 4, if no restriction is given. An n-OSRT contains only rewrite rules of types $m \le n$. A 3-OSRT $\mathcal{R}$ is called *deterministic* if for each rule $l \to r \Leftarrow s_1 \to t_1, \ldots, s_n \to t_n$ in $\mathcal{R}$ and each $1 \le i \le n$, we have $\mathcal{V}ar(s_i) \subseteq \mathcal{V}ar(l) \cup \bigcup_{j=1}^{i-1} \mathcal{V}ar(t_j)$. If for all specializations ν we have $LS(\nu(\ell)) \ge LS(\nu(r))$ we then say that the OS-rule $\ell \to r \Leftarrow c$ is *sort-decreasing*. We call an OSRT $\mathcal{R} = (\Sigma, B, R)$ *sort-decreasing* if all rules in R are so.

We write $\mathcal{R} \vdash t \to u$, abbreviated $t \to_{\mathcal{R}} u$, (resp. $\mathcal{R} \vdash t \to^\star u$, abbreviated $t \to^\star_{\mathcal{R}} u$) iff there is a closed proof tree (i.e., with no pending open goals) for $t \to u$ (resp. $t \to^\star u$) from $\mathcal{R}$ using the inference system in Figure 1. We write $\mathcal{R} \nvdash t \to u$, abbreviated $t \not\to_{\mathcal{R}} u$ (resp. $\mathcal{R} \nvdash t \to^\star u$, abbreviated $t \not\to^\star_{\mathcal{R}} u$) if such a closed proof tree does *not* exist. $t \to_{\mathcal{R}} u$ is a more compact notation for the usual notation $t \to_{R,B} u$, which we shall also use. That is, the above inference

system uses matching modulo B at some subterm position to perform a rewrite step. Note, however, the subtle notational distinction between the LaTeX `\ast` symbol ($*$), and the `\star` symbol ($\star$). This has been done on purpose to stress that $t \to^{\star}_{\mathcal{R}} u$ is *not* the reflexive-transitive closure of $t \to_{\mathcal{R}} u$, but only a closely-related relation. What we actually have (see [18]) is: $t \to^{\star}_{\mathcal{R}} u$ iff either: (i) $t =_B u$, or (ii) there is an $n \geq 1$ such that $t \to_{\mathcal{R}} t_1 \to_{\mathcal{R}} \ldots \to_{\mathcal{R}} t_n =_B u$. Since we assume that there is a finitary B-matching algorithm, as we explain in more detail in Section 4, the relations $\to_{\mathcal{R}}$ and $\to^{\star}_{\mathcal{R}}$ can be easily implemented for deterministic OSRTs.

An OSRT $\mathcal{R}$ is called *confluent* iff for all terms t, u, v, $t \to^{\star}_{\mathcal{R}} u$ and $t \to^{\star}_{\mathcal{R}} v$ imply that there is a term w such that $u \to^{\star}_{\mathcal{R}} w$ and $v \to^{\star}_{\mathcal{R}} w$.

An OSRT $\mathcal{R}$ is called *terminating* if there is no infinite rewrite sequence $t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} \cdots$ Besides this *classical* notion, which is also important in our development, we discuss below the notion of *operational termination* of OSRTs.

Strict Coherence. The intention, of course, is to achieve with $\to_{\mathcal{R}}$ the effect of rewriting in B-equivalence classes; that is, the effect of a relation usually denoted $\to_{R/B}$, which can be much harder and much less efficient to implement (Section 4 discusses implementation issues in more detail). However, for $=_B$ to be an actual *bisimulation* between $\to_{R,B}$ and $\to_{R/B}$, so that the more efficient relation $\to_{R,B}$ defined in Figure 1 can be used, the rules R should be *closed* under B-*extensions*, which makes them *strictly coherent* in the sense of [18]. Strict coherence precisely means that if $t \to_{\mathcal{R}} u$ and $t =_B t'$, then there is a term u' such that $t' \to_{\mathcal{R}} u'$ and $u =_B u'$. Furthermore, if the rules R are closed under B-extensions, they satisfy the following, even stronger property (see [18]), which implies strict coherence as a corollary:

For each instance of **Replacement** (Repl) of the form:

$$\frac{\theta(u_1) \to^{\star} \theta(v_1) \quad \ldots \quad \theta(u_n) \to^{\star} \theta(v_n)}{u \to v}$$

and each B-equality $u =_B u'$ there is another instance of (Repl) of the form

$$\frac{\theta'(u_1) \to^{\star} \theta'(v_1) \quad \ldots \quad \theta'(u_n) \to^{\star} \theta'(v_n)}{u' \to v'}$$

with $v =_B v'$ and $\theta(x) = \theta'(x)$ for each $x \in \mathcal{V}ar(u_1 \to v_1 \ \wedge \ldots \wedge \ u_n \to v_n)$.

Strict coherence is a stronger and simpler notion than the usual notion of coherence [10], but it is all we need when the equations B are regular and linear. For B any combination of associativity, commutativity and identity axioms, Maude automatically computes the B-extensions of the user-given rules R (see Sect. 4.8 in [3]), so that strict coherence, and with it the effect of rewriting in B-equivalence classes, is achieved. Since strict B-coherence is essential for this effect to be achieved, and therefore for *correct* executability, from now on we will assume that all OSRTs are closed under B extensions and are therefore strictly B-coherent.

Operational Termination. Given a logic $\mathcal{L}$ (defined by its inference rules), one has the notion of a *theory* or *specification* $\mathcal{S}$ in such a logic, so that $\mathcal{L}$'s inference system becomes specialized to each such specification $\mathcal{S}$ to derive its provable theorems φ. Assume that we have an *interpreter* for the logic $\mathcal{L}$, that is, an *inference machine* that, given a theory $\mathcal{S}$ and a goal formula φ will try to incrementally build a proof tree for φ. Intuitively, we would say that $\mathcal{S}$ is *terminating* if for any φ the interpreter either finds a proof in finite time, or fails in all possible attempts also in finite time, see Definition 4 below for a precise statement of the property. The notion of *operational termination* captures this fact, meaning that, given an initial goal, an interpreter will either succeed in finite time in producing a closed proof tree, or will fail in finite time, not being able to close or extend further any of the possible proof trees, after exhaustively searching for all such proof trees. We summarize below some of the main concepts in [13, 4].

Theories $\mathcal{S}$ in a logic $\mathcal{L}$ belong to a set of theories $Th_{\mathcal{L}}$. Each theory $\mathcal{S} \in Th_{\mathcal{L}}$ has an associated set $Form_{\mathcal{L}}(\mathcal{S})$ of its *formulas*, and an associated set of *inference rules* $\mathcal{I}_{\mathcal{L}}(\mathcal{S})$, where each inference rule $\iota \in \mathcal{I}_{\mathcal{L}}(\mathcal{S})$ is a scheme specifying a (possibly infinite) set of pairs $(\vec{\varphi}, \varphi)$, called its *instances*, and denoted $\frac{\vec{\varphi}}{\varphi}$, where $\vec{\varphi} \in Form_{\mathcal{L}}(\mathcal{S})^*$ and $\varphi \in Form_{\mathcal{L}}(\mathcal{S})$. The key proof-theoretic notion in a logic $\mathcal{L}$ is that of a *proof tree*.

Definition 1. *The set of (finite) proof trees for a theory $\mathcal{S}$ in a logic $\mathcal{L}$ and the root of a proof tree are defined inductively as follows. A* proof tree *is*

- *either an* open goal*, simply denoted as φ, where φ is a formula for $\mathcal{S}$; then, we define $root(\varphi) = \varphi$,*
- *or a* non-atomic *tree with φ as its root, denoted as*

$$\frac{T_1 \quad \cdots \quad T_n}{\varphi} \ (\iota)$$

where φ is a formula for $\mathcal{S}$, ι is an inference rule in $\mathcal{I}_{\mathcal{L}}(\mathcal{S})$, and $T_1, \ldots, T_n$ are proof trees such that

$$\frac{root(T_1) \quad \cdots \quad root(T_n)}{\varphi}$$

is an instance of ι.

We say that a proof tree is closed *whenever it is finite and contains no open goals. If T is a closed proof tree for $\mathcal{S}$, we then write $\mathcal{S} \vdash root(T)$, and call $root(T)$ a* theorem *of $\mathcal{S}$.*

Notice the difference between φ, an open goal, and $\overline{\varphi}$, a goal closed by a rule ι without premises. An occurrence of a goal G in a proof tree T is at level n in T if the length of the path from G to the root of T is n.

Increasing chains of (finite) proof trees can give rise to infinite proof trees.

Definition 2. *A proof tree T is a* proper prefix *of a proof tree T' if there are one or more open goals $\varphi_1, \ldots, \varphi_n$ in T such that T' is obtained from T by replacing each φ_i by a non-atomic proof tree T_i having φ_i as its root. We denote the proper prefix relation as $T \subset T'$.*

An infinite proof tree *is an infinite increasing chain of finite trees, that is, a sequence $\{T_i\}_{i\in\mathbb{N}}$ such that for all i, $T_i \subset T_{i+1}$.*

We characterize the proof trees with computational meaning (those which are computed by an *interpreter* for $\mathcal{L}$ which solves goals in a proof tree bottom-up and from left to right), by means of the notion of well-formed proof tree.

Definition 3. *We say that a proof tree T is* well-formed *if it is either an open goal, or a closed proof tree, or a proof tree of the form*

$$\frac{T_1 \quad \cdots \quad T_n}{\varphi} \quad (\iota)$$

where for each j T_j is itself well-formed, and there is $i \leq n$ such that T_i is not closed, for any $j < i$ T_j is closed, and each of the T_{i+1} ,...,T_n is an open goal. An infinite proof tree is well-formed *if it is an ascending chain of well-formed finite proof trees.*

Definition 4 (Operational termination). *A theory $\mathcal{S}$ in a logic $\mathcal{L}$ is called* operationally terminating *if no infinite well-formed proof tree for $\mathcal{S}$ exists.*

In particular, we say that an OSRT $\mathcal{R}$ (viewed as a theory of the logic $\mathcal{L}$ for OSRTs whose inference system is given in Figure 1) is operationally terminating by applying Definition 4 to $\mathcal{R}$ and $\mathcal{L}$.

In the same vein, we can say that a predicate π (for instance, $\rightarrow$ or $\rightarrow^\star$ in the inference system of Figure 1) is operationally terminating if for any goal φ such that $\varphi = \pi(t_1, \ldots, t_k)$ for terms $t_1, \ldots, t_k$, φ is operationally terminating. In the following, we speak about *operational* 1-*termination* of an OSRT as the operational termination of $\rightarrow$ (with respect to the inference system of Figure 1). Since by the (Tran) rule the operational non-termination of $\rightarrow$ implies that of $\rightarrow^\star$, the operational termination of an OSRT is equivalent to the operational termination of $\rightarrow^\star$. Similarly, we say that a term t is operationally terminating (resp. operationally 1-terminating) iff every goal $t \rightarrow^\star u$ (resp. $t \rightarrow u$) is operationally terminating for all terms u. We call $\mathcal{R}$ *ground operationally terminating* (resp. *ground operationally* 1-*terminating*) iff all $t \in \mathcal{T}_\Sigma$ are so.

3. Normal and Abnormal Rewrite Theories

Normal forms and normal theories are defined. The relationship between normal forms and "irreducible terms of finite order" in the sense of [2] is also explained.

3.1. Normal Forms and a Taxonomy of Theories

Definition 5 (Irreducible terms and weak termination). *For $\mathcal{R}$ an OSRT and t, v terms, we say that t is* irreducible *iff for any term v, $t \not\to_{\mathcal{R}} v$. $\mathsf{Irr}(\mathcal{R})$ (resp. $\mathsf{GIrr}(\mathcal{R})$) is the set of irreducible terms (resp. ground irreducible terms) of $\mathcal{R}$. A substitution σ is called* irreducible *iff $\sigma(x) \in \mathsf{Irr}(\mathcal{R})$ for all variables x.*

If t rewrites to an irreducible term v —i.e., $t \to^{\star}_{\mathcal{R}} v$ and $v \in \mathsf{Irr}(\mathcal{R})$— we say that t has a (not necessarily unique) irreducible form *v, denoted $t{\twoheadrightarrow_{\mathcal{R}}}v$ or just $t{\twoheadrightarrow}v$ if no confusion arises. If every term t has an irreducible form, i.e., $t{\twoheadrightarrow_{\mathcal{R}}}v$ for some irreducible v, then $\mathcal{R}$ is called* weakly terminating.

Any terminating OSRT is weakly terminating, but in general, the converse is not true. Of course the very notions of terminating and weakly terminating CTRS can be highly misleading, since when the OSRT is abnormal (for the precise definition, see Definition 7 below) such notions can violate one's basic intuitions about termination. Consider, for example, the following "abnormal" weakly terminating CTRS:

Example 1. *Let $\mathcal{R}$ be the CTRS with constants a, b, c, unary function symbol f, and rules R:*

$$\begin{aligned} f(x) &\to f(f(x)) && (1)\\ f(x) &\to a && (2)\\ a &\to b \Leftarrow a \to c && (3) \end{aligned}$$

Since every Σ-term different from b or c can be rewritten to a, and since a, b, c are $\mathcal{R}$-irreducible, $\mathcal{R}$, though not terminating, is weakly terminating. However, a user evaluating Σ-terms other than b or c in an interpreter to their irreducible form will never *see any results, unless the interpreter implements some reachability checking (which is well-known to be undecidable in general). That is, as far as the user is concerned, except for inputs b and c,* nothing *will* ever *terminate in the most obvious and basic sense of getting back a result from the interpreter. So, even calling $\mathcal{R}$ "weakly terminating" requires large doses of irony and can be highly misleading for the unwary. The moral of this little tale is that* abnormal OSRTs have equally abnormal notions of termination*, which, computationally speaking, are as useless as the abnormal OSRTs themselves.*

Definition 6 (Normal form, weak normalization). *A term v is called a* normal form *iff it is irreducible and operationally 1-terminating. Let $\mathsf{NF}(\mathcal{R})$ (resp. $\mathsf{GNF}(\mathcal{R})$) be the set of normal forms (resp. ground normal forms) of $\mathcal{R}$. A substitution σ is called* normalized *iff $\sigma(x) \in \mathsf{NF}(\mathcal{R})$ for all variables x.*

If $t{\twoheadrightarrow_{\mathcal{R}}}v$ and v is a normal form, we then write $t \to^{!}_{\mathcal{R}} v$ and call v a normal form *of t. If every (ground) term t has a normal form, i.e., $t \to^{!}_{\mathcal{R}} v$ for some normal form v, then $\mathcal{R}$ is called* weakly (ground) operationally terminating *(or* weakly (ground) normalizing*), abbreviated WOT (resp. GWOT).*

Note that $\mathsf{NF}(\mathcal{R}) \subseteq \mathsf{Irr}(\mathcal{R})$ and $\to^{!}_{\mathcal{R}} \subseteq {\twoheadrightarrow_{\mathcal{R}}}$ (these inclusions can be strict!).

Remark 2 (Notation). *If $\mathcal{R}$ is confluent and weakly terminating (resp. WOT) and $t \twoheadrightarrow_{\mathcal{R}} v$ (resp. $t \to^{!}_{\mathcal{R}} v$), then v is called the* irreducible *(resp.* normal *or* canonical*)* form *of t, denoted $v = t\!\downarrow_{\mathcal{R}}$ (resp. $v = t!_{\mathcal{R}}$), which is unique up to B-equality.*

Example 2. *The one-step rewrite relation for $a \to b \Leftarrow a \to c$ (a single rule OSRT) is empty. Hence, this OSRT is terminating and a is irreducible. However, a is* not *a normal form: every attempt to prove a reduction step on a starts an infinite proof tree. Note, however, that b and c* are *normal forms.*

There can also be *reducible* terms that are *not* operationally 1-terminating.

Example 3. *Term $f(a)$ is not operationally 1-terminating in the 2-CTRS $\mathcal{R}$:*

$$\begin{aligned} g(a) &\to c(b) && (4)\\ b &\to f(a) && (5)\\ f(x) &\to x \Leftarrow g(x) \to c(y) && (6) \end{aligned}$$

Since $g(a) \to c(b)$, we have $f(a) \to a$ by means of a finite proof tree. However, since the evaluation of the condition could continue *beyond $c(b)$*

$$g(a) \to c(\underline{b}) \to c(f(a))$$

and the term $f(a)$ can start a new (deep) proof tree, we also have an infinite (well-formed) proof tree for the goal $f(a) \to u$ with u arbitrary.

Remark 3. *Note that $\mathcal{R}$ in Example 3 is terminating. This is easy to see, because the* underlying TRS $\mathcal{R}_u = \{\ell \to r \mid \ell \to r \Leftarrow c \in \mathcal{R}\}$ *is clearly terminating.*

Definition 7 (Normal rewrite theory). *A deterministic OSRT $\mathcal{R}$ is called* normal *(resp.* ground normal*) if the set $\mathsf{Irr}(\mathcal{R})$ (resp. the set $\mathsf{GIrr}(\mathcal{R})$) is operationally terminating, i.e., every irreducible (ground) term is a (ground) normal form: $\mathsf{Irr}(\mathcal{R}) = \mathsf{NF}(\mathcal{R})$ (resp. $\mathsf{GIrr}(\mathcal{R}) = \mathsf{GNF}(\mathcal{R})$). A deterministic OSRT $\mathcal{R}$ is called* abnormal *iff it is not normal.*

Corollary 1. *Any operationally 1-terminating deterministic OSRT $\mathcal{R}$ is normal.*

Remark 4. *Clearly, Examples 1–2 are abnormal CTRSs.*

The strict B-coherence assumption then gives us:

Lemma 1. *Let $\mathcal{R} = (\Sigma, B, R)$ be a deterministic OSRT whose axioms B satisfy requirements (i)–(v) in Section 2 and where the rules R are closed under B-extensions. Then the sets $\mathsf{Irr}(\mathcal{R})$ and $\mathsf{NF}(\mathcal{R})$ are both closed under B-equality.*

PROOF. Let $u \in \mathsf{Irr}(\mathcal{R})$ and let $v =_B u$. We can reason by contradiction. Suppose $v \to_{\mathcal{R}} w$. This exactly means that there is a proof tree proving $v \to w$, but by strict coherence this also means that there is a proof tree proving $u \to w'$ for some $w' =_B w$, contradicting $u \in \mathsf{Irr}(\mathcal{R})$.

Let $u \in \mathsf{NF}(\mathcal{R})$ and let $v =_B u$. Suppose that $v \notin \mathsf{NF}(\mathcal{R})$. By the above reasoning, $v \in \mathsf{Irr}(\mathcal{R})$, so this means that v is irreducible but not operationally 1-terminating. That is, there is an infinite well-formed proof tree T with goal $v \to_{\mathcal{R}} w$. By strict coherence, there is then a goal $u \to w'$ for some $w' =_B w$, and an infinite well-formed proof tree T' which is identical to T, except by changing its goal from $v \to_{\mathcal{R}} w$ to $u \to w'$, contradicting $u \in \mathsf{NF}(\mathcal{R})$. □

Corollary 2. *Let $\mathcal{R} = (\Sigma, B, R)$ be a weakly normalizing deterministic OSRT whose axioms B satisfy requirements (i)–(v) in Section 2 and where the rules R are closed under B-extensions. Then $\mathcal{R}$ is a normal theory.*

PROOF. Suppose $u \in \mathsf{Irr}(\mathcal{R}) - \mathsf{NF}(\mathcal{R})$. Since $\mathcal{R}$ is weakly normalizing, there is a $v \in \mathsf{NF}(\mathcal{R})$ such that $u \to^{!}_{\mathcal{R}} v$. But this forces $u =_B v$, which by Lemma 1 forces $u \in \mathsf{NF}(\mathcal{R})$, contradicting $u \in \mathsf{Irr}(\mathcal{R}) - \mathsf{NF}(\mathcal{R})$. □

Example 4. *The CTRS $\mathcal{R}$ in Example 3 is weakly normalizing (see below) and therefore normal, but, as already shown, not operationally 1-terminating. To see that it is normal, assume to the contrary that $\mathsf{Irr}(\mathcal{R}) - \mathsf{NF}(\mathcal{R}) \neq \varnothing$ and choose a minimal irreducible term $s \in \mathsf{Irr}(\mathcal{R}) - \mathsf{NF}(\mathcal{R})$ so that all its strict subterms are normal forms. Since f is the only symbol defined by a conditional rule, we must have $s = f(t)$ for some normal form t. Since $f(t)$ is irreducible, the evaluation of the condition in the rule cannot succeed, i.e., $g(t)$ must be irreducible. Since t is a normal form, $g(t)$ cannot start any infinite well-formed tree. Contradiction!*

Furthermore, we can describe in detail the set $\mathsf{NF}(\mathcal{R})$. Since all operators are unary, using Polish notation we can express all terms as strings. It is then not hard to check that, denoting by X the set of variables, the normal forms can be described by means of the regular expression:

$$\mathsf{NF}(\mathcal{R}) = a \cup X \cup (f \cup g)^* c^+ (X \cup a).$$

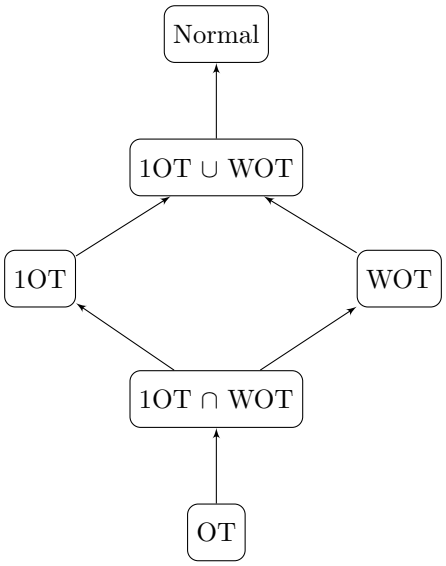
Weak normalization can be shown as follows: b has normal form a, and for any $h \in \{f, g, c\}$ and term t, we can prove by structural induction and case analysis that $h(t)$ has a normal form by assuming that t has a normal form $u \in \mathsf{NF}(\mathcal{R})$.

Note that $\mathcal{R}$ is: (i) terminating (see Remark 3); and (ii) weakly operationally terminating. However, as pointed out in Example 3, $\mathcal{R}$ is not *operationally 1-terminating and, a fortiori,* not *operationally terminating.*

Example 4 and Corollaries 1–2, help us gain a more complete picture about various subclasses of normal rewrite theories. Let **Normal**, **1OT**, **WOT**, and **OT** denote, respectively, the classes of normal, operationally 1-terminating, weakly operationally terminating, and operationally terminating OSRTs. By Corollaries 1–2 we have inclusions:

$$\mathbf{OT} \subseteq \mathbf{1OT}, \mathbf{WOT} \subseteq \mathbf{Normal}.$$

From Example 4 we know that $\mathbf{WOT} \not\subseteq \mathbf{1OT}$. Furthermore, the 1-rule TRS $\{f(x) \to f(f(x))\}$ shows that $\mathbf{1OT} \not\subseteq \mathbf{WOT}$. And the two-rules TRS $\{f(x) \to f(f(x)), f(x) \to a\}$ shows that we have a *strict* inclusion $\mathbf{OT} \subset \mathbf{1OT} \cap \mathbf{WOT}$. The strict inclusion $\mathbf{1OT} \cup \mathbf{WOT} \subset \mathbf{Normal}$ is also easy to check by adding to the CTRS in Example 3 the rewrite rule $h(x) \to h(h(x))$, which breaks weak operational termination but does not change the set of normal forms. Therefore, all inclusions (indicated by arrows) in the taxonomy below are strict.



The following notions of strongly and super-strongly deterministic OSRT are very useful for executability purposes and will be extensively used later in this paper.

Definition 8 (Strongly and super-strongly deterministic rewrite theory). *A deterministic OSRT $\mathcal{R} = (\Sigma, B, R)$ is called* strongly deterministic *(resp.* super-strongly deterministic*) if for each $\ell \to r \Leftarrow s_1 \to t_1, \ldots, s_n \to t_n$ in R, and each irreducible substitution θ we have: $\theta(t_1), \ldots, \theta(t_n) \in \mathsf{Irr}(\mathcal{R})$ (resp. and each normalized substitution θ we have: $\theta(t_1), \ldots, \theta(t_n) \in \mathsf{NF}(\mathcal{R})$).*

The CTRS $\mathcal{R}$ in Example 3 is super strongly deterministic. Since irreducible and normal forms coincide for normal OSRTs, strongly deterministic normal OSRTs are obviously super strongly deterministic. But not all strongly or super-strongly deterministic rewrite theories are normal. For an example of an abnormal super-strongly deterministic rewrite theory, see the conditional CTRS extending combinatory logic in Section 5.5 of [22], which is a variation on the CTRS in Prop. 4.4 of [2].

Remark 5. *Note that if $\mathcal{R}$ is normal, then $\mathcal{R}$ is strongly deterministic iff $\mathcal{R}$ is super-strongly deterministic. However, for abnormal theories the concepts are*

different. For example, the following abnormal CTRS is strongly deterministic, but not *super-strongly deterministic:*

$$f(x) \quad \to \quad g(x) \Leftarrow x \to a \tag{7}$$
$$a \quad \to \quad b \Leftarrow a \to c \tag{8}$$

This is because a in the first rule's condition is irreducible but not a normal form, so that rule is not super-strongly deterministic.

The notion of strongly deterministic OSRT generalizes that of strongly deterministic CTRS in the sense of [20]. The conceptual difference between strongly and super-strongly deterministic OSRTs is obscured by the fact that in [20], and as far as we know in the entire CTRS literature, since irreducible terms and normal forms are conflated, irreducible substitutions are called normalized *substitutions.*

3.2. Relationship between Normal Forms and Irreducible Terms of Finite Order

We explain in detail the relationship between our notion of normal form and Bergstra and Klop's notion[4] of "irreducible term of finite order" in [2] (Def. 4.2), already alluded to in Footnote 1. Their notion is defined for orthogonal CTRSs $\mathcal{R}$ with rules satisfying the requirements explained in Footnote 2 (their type III_n CTRSs, see also Footnote 2), which they prove in [2] to always be confluent. Here is their precise definition, modulo the terminological change just remarked in Footnote 4 to avoid unnecessary confusion:

Definition 9. *([2]) Let $\mathcal{R} = (\Sigma, \varnothing, R)$ be a type III_n CTRS in the sense of [2]. We inductively define the following subsets of $\mathsf{Irr}(\mathcal{R})$:*

- $\mathsf{Irr}(\mathcal{R})_0 = \mathsf{Irr}(\mathcal{R}_u)$ *(recall the definition of $\mathcal{R}_u$ in Remark 3).*
- $t \in \mathsf{Irr}(\mathcal{R})_{n+1}$ *iff for all substitutions θ, rules $l \to r \Leftarrow \bigwedge_{i=1..k} u_i \to v_i$ in R, and contexts C such that $t = C[\theta(l)]$, there are $j \leq n$, and $1 \leq i \leq k$, such that $\theta(u_i) \to_{\mathcal{R}}^* u$, $u \in \mathsf{Irr}(\mathcal{R})_j$ with*[5] *$u \not\equiv v_i$.*
- $\mathsf{Irr}(\mathcal{R})_f = \bigcup_{n \in \mathbb{N}} \mathsf{Irr}(\mathcal{R})_n$*, called the set of* irreducible terms of finite order.

Using confluence of $\mathcal{R}$, Bergsta and Klop then prove (Prop. 4.2.1) that $\mathsf{Irr}(\mathcal{R})_f \subseteq \mathsf{Irr}(\mathcal{R})$. Although in general Bergstra and Klop allow type III_n CTRS to have extra variables in their conditions, since the righthand sides v_i of each conjunct in the condition is an R_u-irreducible *ground* term, in a rule application, the substitution for those extra variables *cannot be incrementally computed*, as it can

[4] Since Bergstra and Klop call all irreducible terms "normal forms," they actually use the terminology "normal form of finite order." To avoid needless terminological confusion, in what follows we will instead call their notion "irreducible term of finite order." This expresses their intended meaning exactly, while avoiding any danger of confusion.

[5] Remember that v_i is a ground $\mathcal{R}_u$-normal form in any III_n CTRS. Here $\equiv$ denotes syntactic equality between terms.

for super-strongly deterministic CTRSs, where such substitutions are computed incrementally, as each conjunct in the condition is solved from left to right. That is, such substitutions must be *guessed* among a generally infinite number of possibilities. Since without the use of symbolic methods this makes the execution of such rules by an interpreter unfeasible, from now on we will restrict our discussion to type III_n CTRS *with no extra variables in their conditions*, which are a special case of super-strongly deterministic CTRSs. Also, since this makes some arguments below easier, we also assume the practical case where the set R of rules of the CTRS is finite.

The first key fact is that we have: $\mathsf{Irr}(\mathcal{R})_0 \subseteq \mathsf{NF}(\mathcal{R}) \subseteq \mathsf{Irr}(\mathcal{R})_f \subseteq \mathsf{Irr}(\mathcal{R})$. The second key fact is that we can have $\mathsf{NF}(\mathcal{R}) \subset \mathsf{Irr}(\mathcal{R})_f$; that is, the notion of "irreducible term of finite order" is *strictly weaker* than that of "normal form" in our sense. The first fact is stated in the following proposition.

Proposition 1. *Let $\mathcal{R} = (\Sigma, \varnothing, R)$ be a type III_n CTRS in the sense of [2], whose rules have no extra variables in their conditions and with R finite. Then we have the set-theoretic inclusions:*

$$\mathsf{Irr}(\mathcal{R})_0 \subseteq \mathsf{NF}(\mathcal{R}) \subseteq \mathsf{Irr}(\mathcal{R})_f \subseteq \mathsf{Irr}(\mathcal{R}).$$

Proof. As already mentioned, Bergsta and Klop prove in [2] (Prop. 4.2.1) that $\mathsf{Irr}(\mathcal{R})_f \subseteq \mathsf{Irr}(\mathcal{R})$. The inclusion $\mathsf{Irr}(\mathcal{R})_0 \subseteq \mathsf{NF}(\mathcal{R})$ is trivial. This leaves us with proving the inclusion $\mathsf{NF}(\mathcal{R}) \subseteq \mathsf{Irr}(\mathcal{R})_f$. By definition, $t \in \mathsf{NF}(\mathcal{R})$ iff $t \in \mathsf{Irr}(\mathcal{R})$ and t is operationally 1-terminating. That is, all proof attempts to prove a one-step rewrite $t \to w$ fail in finite time. More precisely, for any such goal $t \to w$ all well-formed proof trees we can build are finite, and none of them is closed. As explained in Section 4.1, we can restrict ourselves to *sensible* well-formed proof trees. If t is operationally 1-terminating and R satisfies the above conditions, then the set of sensible well-formed proof trees having a goal of the form $t \to t'$ is finite and none of them is closed. Call a sensible well-formed, finite proof tree T *maximal* iff it cannot be extended to a bigger sensible well-formed tree T' with $T' \supset T$. Call a sensible non-closed maximal well-formed tree T a *failure* tree, since it represents a clearly failed proof attempt. Let $FT(\mathcal{R})$ denote the set of failure trees of $\mathcal{R}$. For simplicity, in what follows we will assume that all well-formed proof trees are built using the inference sytem in Figure 2.

We define a function $tc : \mathsf{NF}(\mathcal{R}) \to \mathbb{N}$, called the *tree cost* of t, and prove that $t \in \mathsf{NF}(\mathcal{R})$ implies $t \in \mathsf{Irr}(\mathcal{R})_f$ by strong induction on $tc(t)$. Intuitively, $tc(t)$ measures the cost of detecting all the failed rewrite attempts to rewrite t, and is defined as follows. We first define the set $Att_{\mathcal{R}}(t)$ of t's *rewrite attempts* as the set:

$$Att_{\mathcal{R}}(t) = \{(\alpha, p, \theta) \mid \alpha = (\ell \to r \Leftarrow s_1 \to t_1, \ldots, s_n \to t_n) \in R, p \in Pos(t), t|_p = \theta(\ell)\}.$$

And for each such rewrite attempt (α, p, θ), with $\alpha = (\ell \to r \Leftarrow s_1 \to t_1, \ldots, s_n \to t_n)$, we define the set $FT(\alpha, p, \theta)$ of its *failure trees* as:

$$FT(\alpha, p, \theta) = \{(\frac{T_1 \quad \cdots \quad T_n}{t \to t[\theta(r)]_p}) \in FT(\mathcal{R}) \mid goal(T_i) = \theta(s_i) \to^\star \theta(t_i), 1 \leq i \leq n\}$$

where $\alpha = (\ell \to r \Leftarrow s_1 \to t_1, \ldots, s_n \to t_n)$. Then, for each $t \in \mathsf{NF}(\mathcal{R})$ we define $tc(t) \in \mathbb{N}$ as the double sum:

$$tc(t) = \Sigma_{(\alpha,p,\theta)\in Att_{\mathcal{R}}(t)} \Sigma_{T\in FT(\alpha,p,\theta)} |T|$$

where $|T|$ is the size of tree T, that is, its number of nodes.

The base case for the inductive proof is easy, since $tc(t) = 0$ means that no rewrite attempts are possible, i.e., that t is irreducible by $\mathcal{R}_u$ and therefore belongs to $\mathsf{Irr}(\mathcal{R})_0$. Assume that $u \in \mathsf{Irr}(\mathcal{R})_f$ for each normal $tc(u) \leq m$ and suppose $tc(t) = m + 1$. This means that $Att_{\mathcal{R}}(t) \neq \varnothing$. Pick any $(\ell \to r \Leftarrow s_1 \to t_1, \ldots, s_n \to t_n), p, \theta) \in Att_{\mathcal{R}}(t)$ and let k be biggest possible such that $\mathcal{R} \not\vdash \theta(s_k) \to^\star \theta(t_k)$ but $\mathcal{R} \vdash \theta(s_i) \to^\star \theta(t_i)$, $1 \leq i < k$. Obviously, $1 \leq k < n$, and there is a failure tree for this attempt of the form $\frac{T_1 \quad \cdots \quad T_n}{t \to t[\theta(r)]_p}$ with the $T_1, \ldots, T_{k-1}$ closed proof trees, T_k a failure tree, and $T_{k+1}, \ldots, T_m$ single node trees of the form $\theta(s_j) \to^\star \theta(t_j)$, $k + 1 \leq j \leq n$. Furthermore, any other failure tree T' for the same proof attempt $(\ell \to r \Leftarrow s_1 \to t_1, \ldots, s_n \to t_n), p, \theta)$ and having $T'_1 = T_1, \ldots, T'_k = T_{k-1}$ must also have T'_k a failure tree. Since from an infinite well-formed proof tree for a goal $u \to^\star w$ we can easily build an infinite well-formed proof tree for any other goal $u \to^\star w'$, this means that $\theta(s_k)$ is operationally terminating and a fortiori terminating. The maximality of T_k means that there is an irreducible term $v \neq \theta(t_k)$ appearing in a node $v \to^\star \theta(t_k)$ of T_k such that $\theta(s_k) \twoheadrightarrow v$. Furthermore, v must be a normal form, since otherwise $\theta(s_k)$ would be operationally non-terminating. But any failure tree for v is a proper subtree of a failure tree for the attempt $(\ell \to r \Leftarrow s_1 \to t_1, \ldots, s_n \to t_n), p, \theta)$. Therefore, $tc(v) \leq k$, and by the induction hypothesis there is a natural number n_v such that $v \in \mathsf{Irr}(\mathcal{R})_{n_v}$. Therefore, $t \in \mathsf{Irr}(\mathcal{R})_{n_v+1}$, as desired. □

The following type III_n CTRS shows the second fact, i.e., that the inclusion $\mathsf{NF}(\mathcal{R}) \subseteq \mathsf{Irr}(\mathcal{R})_f$ can be proper.

Example 5. *Let $\mathcal{R}$ be the CTRS with rules:*

$$tt \quad \to \quad true \tag{9}$$

$$a \quad \to \quad b \Leftarrow a \to c \wedge tt \to false. \tag{10}$$

Since it is orthogonal and both c and $false$ are $\mathcal{R}_u$-irreducible, it is a type III_n CTRS. Note that we have $\mathsf{Irr}(\mathcal{R})_0 = \mathsf{NF}(\mathcal{R}) = \{b, c, true, false\} \cup \mathcal{X}$. And, since $tt \to true$, $true \in \mathsf{Irr}(\mathcal{R})_0$, and $true \not\equiv false$, $\mathsf{Irr}(\mathcal{R})_1 = \mathsf{Irr}(\mathcal{R})_f = \mathsf{Irr}(\mathcal{R}) = \{a, b, c, true, false\} \cup \mathcal{X}$. In particular, a, which is not *operationally 1-terminating, and therefore not a normal form in our sense, is already in the set $\mathsf{Irr}(\mathcal{R})_1$.*

The above example clearly shows that the notion of "irreducible term of finite order" is too weak to capture the intuitive notion of normal form; that is, a term returned by an interpreter as the result of the term normalization process.

Admittedly, for type III_n CTRSs with no extra variables in their conditions one could build a special-purpose interpreter computing terms in $\mathsf{Irr}(\mathcal{R})_f$ by evaluating in parallel all the left term instances $\sigma(s_1)\ldots\sigma(s_n)$ of the condition in a rule application and then detecting if some $\sigma(s_i)$ reduces to some term $u \in \mathsf{Irr}(\mathcal{R})_k$ for some k, with $u \not\equiv t_i$, by recursively trying to evaluate u in the same manner. But the correctness of this special-purpose interpreter depends crucially on the *confluence* of type III_n CTRSs and breaks down for non-confluent CTRSs, as the following example shows:

Example 6. *Let $\mathcal{R}$ be the CTRS with rules:*

$$[x] \;\to\; x; x \tag{11}$$
$$x; x \;\to\; \mathit{true} \tag{12}$$
$$x; y \;\to\; x; [y] \tag{13}$$
$$a \;\to\; b \Leftarrow [\mathit{tt}] \to \mathit{false}. \tag{14}$$

The point here is that, when trying to apply the last rule to a, $[\mathit{tt}]$ will reduce to $\mathit{true} \in \mathsf{Irr}(\mathcal{R})_0$, and of course $\mathit{false} \in \mathsf{Irr}(\mathcal{R})_0$ and $\mathit{true} \not\equiv \mathit{false}$. But, if an interpreter cannot assume confluence —indeed this example is non-confluent— it cannot jump to the conclusion *that a cannot be rewritten to b in some other way and —since a is not operationally 1-terminating— must in this case loop forever trying to satisfy the condition $[\mathit{tt}] \to \mathit{false}$.*

Furthermore, if we add to the above CTRS the extra rule $x; [x] \to \mathit{false}$, the special-purpose interpreter for type III_n CTRSs described above —that concludes that a conditional rule cannot be applied *to an instance of its lefthand side when the left side of the corresponding instance of one of its conditions can be rewritten to an irreducible term syntactically different from the condition's irreducible righthand side— will return an* incorrect *result, since it will jump to the false conclusion that, since $[\mathit{tt}]$ reduces to true, a is* irreducible*, and will return a as the result of evaluating a, when in fact now a can be reduced to b.*

Besides depending crucially on confluence, even under the confluence assumption, the idea used by the above special-purpose interpreter for type III_n CTRSs of evaluating in parallel all left term instances $\sigma(s_1)\ldots\sigma(s_n)$ of the condition in a conditional rule application attempt cannot be used to evaluate terms for confluent strongly deterministic CTRSs with extra variables in their condition. The reason is that for such CTRSs the substitution σ for those extra variables must be computed *incrementally*, by solving one condition at a time *from left to right*.

In summary: (i) the inclusion $\mathsf{NF}(\mathcal{R}) \subseteq \mathsf{Irr}(\mathcal{R})_f$ can be proper, so that the notion of "irreducible term of finite order" is too weak to capture the intuitive notion of normal form; and (ii) such a notion and its effective computation depend crucially on the confluence assumption and on rules having no extra variables in their condition, so that its evaluation mechanism either becomes incorrect or cannot be used when either of those assumptions is violated.

4. Computing with Deterministic Rewrite Theories

For executability purposes, deterministic OSRTs are the traditional *non plus ultra*. It is certainly possible to execute more general rewrite theories *symbolically*, as done, for example, in the rewriting modulo SMT approach and, as explained in [21], there are compelling system-oriented applications making such symbolic rewriting very useful. But as far as ordinary rewriting is concerned, the big problem with executing rewrite theories that do not satisfy the determinism requirement is the arbitrariness in choosing a substitution σ for the additional variables in a rule, since in general an infinite number of such substitutions can be chosen. Therefore, this section focuses on the operational semantics and computational properties of *deterministic* rewrite theories $\mathcal{R}$ in general, which can be normal or abnormal, with special emphasis on what can be computed in either case.

So, what do we want to compute? In a functional interpretation of $\mathcal{R}$ we want to compute *unique normal forms*, that is, the unique *values* to which functional expressions, i.e., terms, *reduce* or *normalize* to. Instead, in a concurrent system interpretation of $\mathcal{R}$, although we may also be interested in computing (not necessarily unique) normal forms, which are now interpreted as *final states* of the system, we are more generally interested in solving *reachability problems* for $\mathcal{R}$, that is, in performing some kind of *explicit-state model checking* for $\mathcal{R}$, so that computing final states becomes a special case of this more general reachability analysis.

This is exactly what is supported in Maude [3] for system modules (i.e., rewrite theories with a concurrent system interpretation) through the `search` t `=>*` u command, which asks for all states $\mathcal{R}$-reachable from state t which are substitution instances of the pattern u, and the `search` t `=>!` u command, which asks for all *final states* reachable from state t and matching the pattern u. Of course, if $\mathcal{R}$ is interpreted functionally, the evaluation of a term t by means of Maude's `reduce` command is mathematically equivalent to the (in this case unique if it exists, thanks to confluence) answer of the `search` t `=>!` x command, with x a variable, when the same $\mathcal{R}$ is interpreted non-functionally. But, thanks to confluence, Maude's `reduce` command can be implemented much more efficiently. We explain below the precise meaning of these search commands as ways of computing solutions of *reachability* (resp. *normalization*) *problems*.

Since the reachability problems `search` t `=>*` u we shall consider are solved by *rewriting*, and not by narrowing, the only variables that need be instantiated in a solution to such problems are those of u. This means that *without loss of generality* the term t can be assumed to be a *ground* term. The case when $\mathcal{V}ar(t) = \{x_1, \ldots, x_n\}$ can be reduced to the ground case by working on the extended signature where the $x_1, \ldots, x_n$ have been added as *extra constants*.

Given an OSRT $\mathcal{R} = (\Sigma, B, R)$ and Σ-terms t, u in the same connected component with t a ground term, the *reachability problem from t to u* asks for all substitutions θ with $dom(\theta) = \vec{y} = \mathcal{V}ar(u)$ such that $\mathcal{R} \vdash t \rightarrow^\star v$ and $v =_B \theta(u)$. That is, for all witnesses solving in $\mathcal{R}$ the existential formula

$$(\exists \vec{y})\ t \rightarrow^\star u.$$

In particular, the question of what terms are $\mathcal{R}$-*reachable* from t is the reachability problem from t to x, where if t has sort s, x is a fresh variable of top sort $\top_{[s]}$. That is, the witnesses solving in $\mathcal{R}$ the existential formula $(\exists x)\ t \rightarrow^{\star} x$.

Similarly, the *normalization problem from t to u* asks for all substitutions θ with $dom(\theta) = \vec{y} = \mathcal{V}ar(u)$ such that $\mathcal{R} \vdash t \rightarrow^{!} v$ and $v =_B \theta(u)$. And the *unrestricted normalization problem for t* is the normalization problem from t to x, where if t has sort s, x is a fresh variable of top sort $\top_{[s]}$. That is, its solutions are all substitutions θ which are witnesses solving in $\mathcal{R}$ the existential formula $(\exists x)\ t \rightarrow^{\star} x$ and such that $\theta(x) \in \mathsf{NF}(\mathcal{R})$.

Of course, by changing $t \rightarrow^{!}_{\mathcal{R}} v$ to $t \twoheadrightarrow_{\mathcal{R}} v$, and $\mathsf{NF}(\mathcal{R})$ to $\mathsf{Irr}(\mathcal{R})$, instead of the normalization problem from t to u we would define the *weak termination problem*[6] *from t to u*. The two crucial observations about such a weak termination problem are:

1. If $\mathcal{R}$ is normal, the normalization and weak termination problems *coincide*.
2. If $\mathcal{R}$ is abnormal, such problems do *not* coincide. As explained later, the normalization problem can still be solved by search; but generating a complete set of solutions for the weak termination problem is generally *impossible*, because, as shown in [11, 2] and further discussed in Section 4.3, in general the set $\mathsf{Irr}(\mathcal{R})$ is *not* recursively enumerable.

Of course, when we give a concurrent system interpretation to an abnormal theory $\mathcal{R}$, the set of *all* its *final* states is $\mathsf{Irr}(\mathcal{R})$, and not just $\mathsf{NF}(\mathcal{R})$. However, final states in $\mathsf{Irr}(\mathcal{R}) - \mathsf{NF}(\mathcal{R})$ violate all computational intuitions about what a "final state" is and how to reach it, since we can be already in a final state $u \in \mathsf{Irr}(\mathcal{R}) - \mathsf{NF}(\mathcal{R})$ but never be able to know it, and would then face the Sisyphean task of forever trying to *reach* a final state *from* such a u.

All this clearly suggest that in any concurrent system interpretation of $\mathcal{R}$ *where we need to reason about final states*, deterministic *normal* theories are the *non plus ultra*: abnormal theories are freaks of nature to be avoided at any cost. Likewise, in a functional interpretation of $\mathcal{R}$ normal theories are also the *non plus ultra*.[7] Certainly, under assumptions of strong determinism, confluence, weak termination, and sort-decreasingness, the quotient under $=_B$ of the set of ground irreducible terms $\mathsf{GIrr}(\mathcal{R})$ is, when viewed as an S-sorted family, the carrier of an algebra *isomorphic* to the initial algebra of $\mathcal{R}$ in its equational interpretation.[8] The huge problem, however, is that, in general, such a carrier $\mathsf{GIrr}(\mathcal{R})/=_B$ is not even r.e., and its operations, mapping each ground term to

[6] Typically, when we ask whether a term t weakly terminates we are interested in knowing whether there is *at least one witness* answering positively the problem $(\exists x)\ t \twoheadrightarrow_{\mathcal{R}} x$, with x a fresh variable of top sort $\top_{[s]}$ and $s = LS(t)$. But we can consider more nuanced weak termination problems, like $(\exists \vec{y})\ t \twoheadrightarrow_{\mathcal{R}} v$ for v an irreducible pattern, and we may be interested in enumerating the corresponding answers and not just in knowing that a witness exists.

[7] See Section 4.4 for a more precise formulation of what the *non plus ultra* is for an OSRT $\mathcal{R}$ interpreted functionally when we desire a full agreement between its initial algebra semantics and its canonical term algebra of normal forms.

[8] This easily follows from Theorem 15 in [24] when $\mathcal{R}$ is a CTRS, and from Theorem 5 (the Church-Rosser Theorem) in [18] for $\mathcal{R}$ a general OSRT.

its irreducible form, are *not* computable. That is, the huge problem is that, in general, the mapping $t \mapsto [t\downarrow_{\mathcal{R}}]_B$ is *not* a computable function,[9] making $\mathcal{R}$ useless as a functional program.

Example 7. *Here is a trivial example. Let* Σ *have just constants* a, b, c*, and consider the 1-rule CTRS* $a \to b \Leftarrow a \to c$*, functionally interpreted as the singleton set equation* $E = \{a = b \Leftarrow a = c\}$*. This* $\mathcal{R}$ *is confluent, terminating, and strongly deterministic. Therefore, the Church-Rosser Theorems 15 in [24] and 5 in [18] apply to* $\mathcal{R}$*. In fact, since each ground term is irreducible and* $=_E$ *is the identity relation, the carrier of the initial algebra* $\mathcal{T}_{\Sigma/E}$ *and the set* $\mathsf{GIrr}(\mathcal{R})$ *coincide, so that the isomorphism of initial algebras is the identity. But we only can* reduce *c and b to normal form, since we have* $\mathsf{NF}(\mathcal{R}) = \{b, c\} \subset \{a, b, c\} = \mathsf{GIrr}(\mathcal{R})$*. So* $\mathcal{R}$ *is useless as a functional program.*

The consequences of the above discussion from the practical point of view are that: (i) we can study the operational semantics of solving *reachability problems* for *any* deterministic $\mathcal{R}$, normal or abnormal; but (ii) although we can likewise study the operational semantics of solving *normalization problems* for any deterministic $\mathcal{R}$, the answers to such problems will only be meaningful and practical for *normal* theories. Instead, at the intuitive level of computing final states or evaluating functional expressions, such answers will be essentially and unavoidably *incomplete* (since $\mathsf{Irr}(\mathcal{R}) - \mathsf{NF}(\mathcal{R}) \neq \varnothing$) for *abnormal* theories.

4.1. Solving Reachability Problems in Deterministic OSRTs

In this section we explain in detail how deterministic rewrite theories can be executed in practice, so that the substitution for the extra variables in a rule's condition and righthand side is computed *incrementally*. Our starting point is the inference system for conditional rewriting in Figure 1. However, there is a substantial gap between such a system and a more detailed inference system that can provide an *operational semantics* for deterministic rewrite theories on which an interpreter can be based. The key issue is the need for handling and solving existentially quantified variables in reachability goals $(\exists \vec{y})\ t \to^\star u$.

Our reasoning will be simplified by using the inference system for conditional rewriting in Figure 2, which is slightly different from that in Figure 1, yet easily shown to be equivalent to it. The only difference is that the original (Cong) and (Repl) rules have now been combined into a more powerful (Repl) rule that allows rewrites at any term position p.

The above inference system is certainly quite general. But leaves a number of issues unanswered. The crucial one is how to *guess* the value of the substitution σ for the extra variables in the condition and righthand side when applying a given rule $\ell \to r \Leftarrow u_1 \to v_1 \cdots u_n \to v_n$ in $\mathcal{R}$, which from now on we will always assume is deterministic and strictly B-coherent. Furthermore, the

[9] This result goes back to [11]. See Section 4.3 for a more detailed discussion of computability issues, and Section 4.4 to see how normal theories solve this aporia.

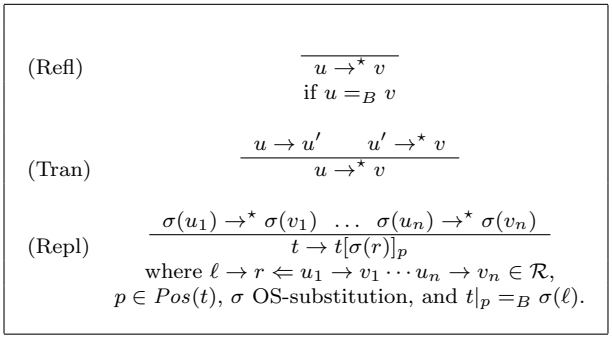


Figure 2: Alternative inference rules for OSRTs

inference rules (Tran) and (Repl) allow useless inference steps that should never be taken by an interpreter implementation:

1. When applying the (Tran) rule, it is useless to select a term u' such that $u \not\to_{\mathcal{R}_u} u'$, since that subgoal will automatically be unprovable.
2. When applying the (Repl) rule, there may be an infinite number of substitutions $\sigma|_{\mathcal{V}ar(\ell)}$ such that $t|_p =_B \sigma|_{\mathcal{V}ar(\ell)}(\ell)$, even though, by the existence of a finitary matching algorithm, there is a *finite* number of matching substitutions in $Match_B(t|_p, \ell)$ that, up to B-equality, represent the possibly infinite number of substitutions of the form $\sigma|_{\mathcal{V}ar(\ell)}$ with $t|_p =_B \sigma|_{\mathcal{V}ar(\ell)}(\ell)$. Because of strict B-coherence, it is useless to choose a $\sigma|_{\mathcal{V}ar(\ell)}$ such that $\sigma|_{\mathcal{V}ar(\ell)} \notin Match_B(t|_p, \ell)$.

Call an application of (Tran) selecting such a hopeless u', or an application of (Repl) with $\sigma|_{\mathcal{V}ar(\ell)} \notin Match_B(t|_p, \ell)$ a "dumb" inference step. Call a well-formed proof tree *sensible* iff it has no dumb inference steps.

Note that the left subgoal $u \to u'$ of any dumb application of (Tran) cannot be extended, i.e., is always a failure goal. Therefore, using strict B-coherence, it is easy to prove that any provable goal of the form $\mathcal{R} \vdash t \to^* t'$ has a closed sensible well-formed proof tree; and that for any provable goal of the form $\mathcal{R} \vdash t \to t'$ there is a provable goal $\mathcal{R} \vdash t \to t''$ having a closed sensible well-formed proof tree and such that $t' =_B t''$.

Therefore, for all practical purposes we can restrict ourselves to sensible well-formed proof trees. The key question now is: if $\mathcal{R}$ is deterministic, strictly B-coherent, with B having a finitary matching algorithm, how can we solve reachability goals of the form $(\exists \vec{y})\ t \to^\star u$? That is, how can we find substitutions σ such that the goal $t \to^\star \sigma(u)$ has a sensible well-formed proof tree? The answer to this question also includes an answer to the question of how the extra variables in a deterministic rule can be solved *incrementally*, and therefore to the further question of specifying a detailed enough operational semantics for deterministic rewrite theories. In essence, these are all the *same* question, which is answered by the more detailed inference system below, which solves goals of

the form $t \rightarrow^\star u$ with t a ground term and u a term possibly with variables, that are understood as *implicitly existentially quantified*.

The inference system takes the form of *meta-level* rewrite rules of the form $G \rightsquigarrow G'$, where G, G' are meta-level terms representing well-formed proof trees. These rules are at the meta-level of the logic because they transform proof tree representations and make use of notions such as term position, matching substitution, and substitution composition. In a reflective language like Maude, they can be implemented as ordinary rewrite rules using its `META-LEVEL` module. Also, although at the meta-level they are indeed rewrite rules, and not narrowing rules, the notation $\rightsquigarrow$ is meant to suggest that they in fact instantiate the variables in u as "logical variables," and therefore have a narrowing-like flavor. As usual, to avoid variable capture, we assume that the variables of all rules in R are *renamed with fresh new variables* in each rule application. The details of such renaming are immaterial. They can be abstracted away by the notion of a standardized apart rule.

Given a rewrite theory $\mathcal{R} = (\Sigma, B, R)$, by a rule of R being *standardized apart*, denoted $(l' \rightarrow r' \Leftarrow C') \ll R$, we mean that there is a variable renaming ρ and a rule $(l \rightarrow r \Leftarrow C) \in R$ such that $(l' \rightarrow r' \Leftarrow C') = \rho(l \rightarrow r \Leftarrow C)$, and, furthermore, the variables $\mathcal{V}ar(l' \rightarrow r' \Leftarrow C')$ are *disjoint* from all the variables previously met during any computation. In our case, the "computations" will be rewrites of the form $G \rightsquigarrow G'$, and their variables will include the variables $\vec{y}$ in the original reachability goal $(\exists \vec{y})\ t \rightarrow^\star u$, implicitly mentioned in the goal $t \rightarrow^\star u$ to be rewritten, plus new variables introduced by conditions.

The rewrite rules will be applied *modulo* associativity and identity of: (i) the conjunction operator $\_ \wedge \_$ with identity $\top$; and (ii) the sequential composition operator $\_;\_$ with identity *nil*. Furthermore, variables ST, SC, SCL, STS, and STL —corresponding to patterns describing already solved parts of the goal, which can be defined either by sorts or by the context-free grammar below— will be used in such rules:

- **Solved Step**: $ST ::= [t \rightarrow u].SC$
- **Solved Condition**: $SC ::= [SCL]$
- **Solved Condition List**: $SCL ::= \top \mid STS \mid STS \wedge SCL$
- **Solved Step Sequence**: $STS ::= [STL\,;\, [t =_B t']]$
- **Solved Step List**: $STL ::= nil \mid ST \mid ST\,;\, STL$

The goal $t \rightarrow^\star u$ (that can be used to represent the evaluation of a *given* term t) will be represented as a term of the form $\{\langle t \rightarrow^\star u \rangle\}$, where $\langle \_ \rangle$ is the *focus symbol*, indicating the *current goal* where meta-level rewriting should take place, and $\{\_\}$ is a *context symbol*, indicating the context in which meta-level rules should be applied, which is needed for the bookkeeping of substitutions (see below). A *solution* of the goal $\{\langle t \rightarrow^\star u \rangle\}$, if it exists, will be of the form: $[STL\,;\, [v_n =_B \theta(u)]].\theta$, i.e., a pair consisting of a solved step sequence and a substitution θ, where STL represents a proof for a sequence of $n \geq 0$ rewrite

steps $t \to v_1 \dots v_{n-1} \to v_n$, and $\theta \in Match_B(v_n, u)$ is the witness proving the existential formula $(\exists \vec{y})\ t \to^\star u$. The meta-level rewrite rules are the following, where id denotes the identity substitution:

- (Refl) $\{STL\,;\, \langle v \to^\star u\rangle\} \rightsquigarrow [STL\,;\, [v =_B \theta(u)]].\theta$, where $\theta \in Match_B(v, u)$.
- (Repl.u) $\{STL\,;\, \langle v \to^\star u\rangle\} \rightsquigarrow$
  $\{STL\,;\, [v \to v[\theta(r)]_p].[\top]\,;\, \{\langle v[\theta(r)]_p \to^\star u\rangle\}\}$, where $\ell \to r \Leftarrow \top \ll R$, $p \in Pos(v)$, $\theta \in Match_B(v|_p, \ell)$.
- (Repl.cnd) $\{STL\,;\, \langle v \to^\star u\rangle\} \rightsquigarrow$
  $\{STL\,;\, \{v \to v[\theta(r)]_p\}.[\![\{\langle \theta(s_1) \to^\star \theta(t_1)\rangle\} \wedge \theta(C)]\!].id\,;\, v[\theta(r)]_p \to^\star u\}$, where $\ell \to r \Leftarrow s_1 \to t_1 \wedge C \ll R$, $p \in Pos(v)$, $\theta \in Match_B(v|_p, \ell)$, and $[\![\_]\!]$ is a context symbol indicating a not yet solved condition,

The above rules exactly mimic the building of a sensible well-formed proof tree using the (Refl) and (Repl) rules, and, implicitly, the (Tran) rule, (which is handled with the sequential composition operator $\_\,;\,\_$). However, rule (Repl.cnd) can produce goals with extra, implicitly existentially quantified variables besides those in the original target term u. This is because in each application of rule (Repl.cnd) the goals in $[\![\langle \theta(s_1) \to^\star \theta(t_1)\rangle \wedge \theta(C)]\!]$ can have (implicitly existentially quantified) fresh variables. To achieve the *incremental* computation of substitutions, so as to make guessing unnecessary in the instantiation of such variables, the following *substitution-handling rules* are added, where α and θ are (meta-)variables ranging over substitutions, $\alpha\theta$ denotes substitution composition, and CND is a (meta-)variable ranging over conditions, that is, over conjunctions of goals of the form $w \to^\star w'$ (note, again, that these rules are applied *modulo* the associativity and identity axioms for $\_ \wedge \_$ and $\_\,;\,\_$):

1. $[\![SCL \wedge STS.\theta \wedge v \to^\star w \wedge CND]\!].\alpha \rightsquigarrow$
   $[\![SCL \wedge STS \wedge \theta(\{\langle v \to^\star w\rangle\} \wedge CND)]\!].\alpha\theta$
2. $[\![SCL \wedge STS.\theta]\!].\alpha \rightsquigarrow [SCL \wedge STS].\alpha\theta$
3. $\{v \to w\}.[SCL].\alpha \rightsquigarrow [v \to \alpha(w)].[SCL].\alpha$
4. $\{STL\,;\, ST.\alpha\,;\, w \to^\star u\} \rightsquigarrow \{STL\,;\, ST\,;\, \langle \alpha(w) \to^\star u\rangle\}$.

Rules (1)–(2) capture the *incremental* computation of the substitution instantiating the extra variables in a deterministic rule's condition. Rule (3) then applies such an incrementally computed substitution to the rule's righthand side. Rule (4) takes care of the fact that rule (Repl.cnd) had created a subgoal of the form $v[\theta(r)]_p \to^\star u$, where r was the righthand side used *in the previous step*, possibly having extra variables that should be instantiated by the substitution α already computed for such a previous step before moving the focus goal to the right.

Example 8. *Consider the following deterministic CTRS for adding natural numbers:*

$$x + 0 \to x \tag{15}$$
$$0 + x \to x \tag{16}$$
$$x + y \to s(s(z + z')) \Leftarrow x \to s(z) \wedge y \to s(z'). \tag{17}$$

The evaluation of the reachability goal $\{\langle s(0)+s(0) \to^\star x' \rangle\}$ *by the above meta-level rewrite rules yields, for example, the following* $\rightsquigarrow$*-irreducible goals, where the sequence of rules labels applied has been recorded (for the sake of readability, we use* n *instead of* $s^n(0)$ *and, if* $n>1$, $s^n(t)$ *instead of* $s(\underbrace{\cdots}_{n-2} s(t) \underbrace{\cdots}_{n-2})$*):*

$$\{\langle 1+1 \to^\star x' \rangle\} \overset{Repl.cnd\, Refl\, 1\, Refl\, 2\, 3\, 4\, Repl.u\, Refl}{\rightsquigarrow\ \rightsquigarrow\ \rightsquigarrow\ \rightsquigarrow\ \rightsquigarrow\ \rightsquigarrow\ \rightsquigarrow\ \rightsquigarrow\ \rightsquigarrow}$$
$$[[1+1 \to s^2(0+0)].[[[1=1]] \wedge [[1=1]]]; [s^2(0+0) \to 2].[\top]; [2=2]].(x'=2)$$

which corresponds to building a closed and sensible well-formed proof tree computing the result $s(s(0))$*. The same goal has the alternative evaluation:*

$$\{\langle 1+1 \to^\star x' \rangle\} \overset{Repl.cnd\, Refl\, 1\, Refl\, 2\, 3\, 4\, Repl.cnd}{\rightsquigarrow\ \rightsquigarrow\ \rightsquigarrow\ \rightsquigarrow\ \rightsquigarrow\ \rightsquigarrow\ \rightsquigarrow\ \rightsquigarrow} G$$

where G *is the goal:*

$$G = \{[1+1 \to s^2(0+0)].[[[1=1]] \wedge [[1=1]]];$$
$$[\![SCL \wedge STS.\theta]\!]\{s^2(0+0) \to s^4(y'+y'')\}.[\![\{\langle 0 \to^\star s(y') \rangle\} \wedge 0 \to^\star s(y'')]\!].id; s^4(y'+y'') \to^\star x'\}$$

which corresponds to building a maximal, open and sensible well-formed proof tree with a failure node $0 \to^\star s(y')$ *which cannot be further extended.*

The above meta-level rewrite rules give a detailed enough specification for building an interpreter to solve reachability goals. Given a reachability goal $\{\langle t \to^\star u \rangle\}$ with t ground, such an interpreter should execute the meta-level rules in a *breadth first* manner to find $\rightsquigarrow$-irreducible goals G in solved form, i.e., goals of the form $STS.\theta$ with STS a solved step sequence and θ the associated witness substitution, yielding a sensible and closed well-formed proof tree for the goal $t \to^\star \theta(u)$.

Note that all $\rightsquigarrow$-irreducible goals yield associated *maximal* sensible well-formed proof trees, which cannot be further extended and are either sensible closed proof trees, or sensible (finite) failure proof trees. We leave for the reader to check that such an interpreter is *sound and complete*, in the sense that it will eventually build all most general[10] and maximal sensible proof trees for the given reachability goal, and only sensible well-formed trees. Of course, unless $\mathcal{R}$ is operationally terminating, in the limit (i.e., as limits of infinite rewrite sequences) the interpreter can also build infinite sensible well-formed proof trees for some reachability goals. Indeed, since the purpose of the meta-level rewrite rules is precisely to simulate the building of sensible well-formed proof trees, it can be shown that lack of operational termination for $\mathcal{R}$ coincides with the existence of infinite meta-level rewrite sequences from some $\mathcal{R}$-reachability goals.

[10] If the sensible well-formed proof tree is closed, then all terms in it will be ground terms. However, as illustrated by goal term G in Example 8, some terms in a sensible well-formed failure proof tree may have variables, so that the associated proof tree could be further instantiated.

4.2. Solving Normalization Problems in Deterministic OSRTs

The above meta-level rewrite rules can also be used to solve normalization problems. The proof of the following three lemmas is left to the reader:[11]

Lemma 2. *Let $\mathcal{R} = (\Sigma, B, R)$ be a deterministic OSRT whose rules are strictly B-coherent. Then a ground*[12] *term v of sort s is a* normal form *iff all meta-level rewrites of the goal $\{\langle v \to^\star x\rangle\}$, with x a variable of top sort $\top_{[s]}$, terminate in failure goals, except for a single solution of the form $[[v =_B v]].(x = v)$, obtained by a single application of the (Refl) meta-level rewrite rule.*

Lemma 3. *Let $\mathcal{R} = (\Sigma, B, R)$ be a deterministic OSRT whose rules are strictly B-coherent. Then a ground term v is a* normal form *of the ground term t of sort s iff the goal $\{\langle t \to^\star x\rangle\}$, with x a variable of top sort $\top_{[s]}$, can be rewritten with the meta-level rules to a goal of the form $\{STL\,;\, \langle v \to^\star x\rangle\}$, and all meta-level rewrites of the goal $\{STL\,;\, \langle v \to^\star x\rangle\}$ terminate in failure goals, except for a single solution of the form $[STL\,;\, [v =_B v]].(x = v)$, obtained by a single application of the (Refl) meta-level rewrite rule.*

Lemma 4. *Let $\mathcal{R} = (\Sigma, B, R)$ be a deterministic OSRT whose rules are strictly B-coherent. Then a ground term v is a normal form and provides solutions for a normalization goal $(\exists x)\ t \to^! u$ iff the goal $\{\langle t \to^\star u\rangle\}$ can be rewritten with the meta-level rules to a goal of the form $\{STL\,;\, \langle v \to^\star u\rangle\}$ such that: (i) $Match_B(v, u) \neq \varnothing$, and (ii) all meta-level rewrites of the goal $\{\langle v \to^\star x\rangle\}$, with x a variable of top sort $\top_{[s]}$, terminate in failure goals, except for a single solution of the form $[[v =_B v]].(x = v)$ (i.e., v is a normal form). Then the set of solutions of the normalization goal provided by the normal form v are: $\{[STL\,;\, [v =_B \theta(u)]].\theta \mid \theta \in Match_B(v, u)\}$, all obtained by single applications of the (Refl) meta-level rewrite rule.*

The above three lemmas immediately suggest a simple strategy to build an interpreter that can solve not only reachability problems of the form $(\exists x)\ t \to^\star u$, but also normalization problems of the form $(\exists x)\ t \to^!\ u$. To solve a normalization problem $(\exists x)\ t \to^!\ u$ the interpreter starts with the goal $\{\langle t \to^\star u\rangle\}$ as for a standard reachability goal; but whenever its breadth first search reaches a goal of the form $\{STL\,;\, \langle v \to^\star u\rangle\}$ such that $Match_B(v, u) \neq \varnothing$, it has to perform the additional check that v is a normal form before it can report the solutions $\{[STL\,;\, [v =_B \theta(u)]].\theta \mid \theta \in Match_B(v, u)\}$ for such a normal form v.

Of course, since the additional check for the normality of v may not terminate and, furthermore, in the case of an abnormal theory $\mathcal{R}$ this non-termination may take place without v being ever rewritten in one step to any other term v', to

[11] We assume that if u has sort s and x has top sort $\top_{[s]}$ in the connected component of s, then $Match_B(u, x) = \{x = u\}$. In principle a B-matching algorithm could return a substitution $x = u'$ with $u' =_B u$, but any practical algorithm will return $x = u$.

[12] Again, the requirement that v is a ground term is immaterial, since otherwise we can always add the variables $\mathcal{V}ar(v)$ to the signature Σ as extra constants.

ensure its *completeness* the interpreter must perform this check for v's normality as a *background task*, which must be interleaved with the further breadth first exploration of all other goals $\rightsquigarrow$-reachable from the original goal $\{\langle t \rightarrow^\star u\rangle\}$.

The strategy such an interpreter can use for solving normalization problems can be greatly simplified under the assumption that $\mathcal{R}$ is operationally 1-terminating, since then the check for v's normality is *decidable*, namely, as the *failure* of all attempts to rewrite v in one step. This normality check can be achieved by: (1) applying in all possible ways either (i) the (Repl-u) or (ii) the (Repl-cnd) meta-level rules to the goal $\{\langle v \rightarrow^\star x\rangle\}$ to obtain goals of the form either (i) $[v \rightarrow v[\theta(r)]_p].[\top]\,;\,\{\langle v[\theta(r)]_p \rightarrow^\star x\rangle\}\}$ (which already show that v is *not* a normal form), or (ii) $\{\{v \rightarrow v[\theta(r)]_p\}.[\![\{\langle \theta(s_1) \rightarrow^\star \theta(t_1)\rangle\} \wedge \theta(C)]\!].id\,;\, v[\theta(r)]_p \rightarrow^\star x\}$; and then (2) checking in case (ii) whether all goals of the form $[\![\{\langle \theta(s_1) \rightarrow^\star \theta(t_1)\rangle\} \wedge \theta(C)]\!].id$ terminate in failure goals, which by the operational 1-termination assumption is decidable. The Maude interpreter assumes that all theories $\mathcal{R}$ on which it is invoked are operationally 1-terminating, and implements in essence this simpler strategy. This means that —since it assumes that the check for normality will always terminate, and therefore performs it before doing any further breath-first search— it is in general *incomplete* for theories not satisfying the operational 1-termination assumption, as shown by the example below.

Example 9. *Maude loops without ever returning a solution for the normalization goal* $(\exists x)\ cnt(0) \rightarrow^! x$ *(the search command* `search` $cnt(0)$ `=>!` x*) when invoked on the following weakly normalizing (thus normal) and strongly deterministic, yet not operationally 1-terminating, CTRS:*

$$cnt(x) \rightarrow s(x) \tag{18}$$
$$cnt(x) \rightarrow cnt(y) \Leftarrow cnt(x) \rightarrow y. \tag{19}$$

Note that $cnt(0)$ *has the infinite set of normal forms:* $\{s^{n+1}(0) \mid n \in \mathbb{N}\}$. *The notation* $cnt(0)$ *is well-chosen, since the operator cnt acts as a* counter. *The terminating states reachable from state* $cnt(s^k(0))$ *are:* $\{s^{n+k+1}(0) \mid n \in \mathbb{N}\}$.

4.3. Computability Properties of Deterministic Rewrite Theories

This section gathers some basic computability results for deterministic OS-RTs. Additional results for confluent theories are presented in Section 4.4.

The distinction between irreducible terms and normal forms can help us better understand the significance of the following well-known result on the in general non r.e. nature of the set of irreducible terms of a CTRS (see, e.g., [11, 2, 22]).

Theorem 1. *[11, 2] There is a deterministic CTRS whose set* $\mathsf{Irr}(\mathcal{R})$ *of irreducible terms is* not recursively enumerable. *Therefore, it is not just undecidable, but not even* semi-decidable *if a term is* $\mathcal{R}$*-irreducible.*

In some sense this theorem shows how wild the beasts in the general menagerie of abnormal OSRTs can be, and illustrates the need for notions such as that of normal theory to obtain reasonable computational behaviors.

By contrast, our explanations in Sections 4.1 and 4.2 of how reachability and normalization problems for a deterministic OSRT can be solved by an interpreter immediately gives us the following result for *all* deterministic OSRTs with a finite set of rules, including abnormal ones:

Proposition 2. *Given a deterministic OSRT $\mathcal{R} = (\Sigma, B, R)$ with a finitary B-matching algorithm and a strictly B-coherent finite set of rules R, the sets $\mathsf{Red}(\mathcal{R})$ of $\mathcal{R}$-reducible terms, and $\mathsf{NF}(\mathcal{R})$ of $\mathcal{R}$-normal forms are both recursively enumerable and therefore semi-decidable.*

The wild behavior of abnormal, deterministic OSRTs completely disappears for normal ones. In fact, normal theories are much better behaved, since the above proposition —plus the well-known fact that if a set and its complement are r.e., then both subsets are recursive— immediately give us:

Corollary 3. *For a normal deterministic OSRT $\mathcal{R} = (\Sigma, B, R)$ with a finitary B-matching algorithm and strictly B-coherent rules R, the sets $\mathsf{Red}(\mathcal{R})$ of $\mathcal{R}$-reducible terms, and $\mathsf{NF}(\mathcal{R})$ of $\mathcal{R}$-normal forms are both recursive and therefore decidable.*

4.4. Confluent Theories: Church-Rosser, Initiality, and Computability Results

Thanks to the Church-Rosser property, confluent theories support a functional programming style in which oriented equations uniquely define recursive functions on the initial algebra, and term normalization exactly corresponds to function evaluation. Furthermore, initial algebras provide an exact correspondence between the mathematical, initial semantics of the equational theory, and the operational semantics of the oriented equations executed by term rewriting. How well are all these results preserved for conditional theories? Not so well, and not always. The purpose of this section is to explain why normal, confluent and weakly normalizing strongly deterministic OSRTs are crucial for all these results and semantic equivalences to be preserved. In particular, they are essential, and most general possible, for conditional equations to define *total computable functions* by conditional term rewriting.

In an equational style of functional programming, the program is an equational theory. In the conditional case, a *conditional* equational theory. To support types, subtypes, and reasoning modulo built-in axioms, we should consider a very general class of such theories, namely, *order-sorted conditional equational theories* [7, 17] of the form $(\Sigma, E \cup B)$, where B are regular and linear, unconditional equational axioms satisfying all the requirements in Section 2. Of course, the conditional equations are executed by *orienting* the conditional equations

E as conditional rewrite rules[13]

$$\vec{E} = \{t \to t' \Leftarrow u_1 \to v_1, \ldots, u_n \to v_n \mid (t = t' \Leftarrow u_1 = v_1, \ldots, u_n = v_n) \in E\}.$$

The essential property linking the order-sorted equational theory $(\Sigma, E \cup B)$ and the OSRT $(\Sigma, B, \vec{E})$ is the *Church-Rosser property*, stating the equivalence:

$$E \cup B \vdash t = t' \quad \Leftrightarrow \quad t \downarrow_{\vec{E},B} t'$$

where $E \cup B \vdash t = t'$ denotes provability in order-sorted conditional equational logic [7, 17], and where, by definition, $t \downarrow_{\vec{E},B} t'$ iff there is a term u such that $t \to^{\star}_{\vec{E},B} u \wedge t' \to^{\star}_{\vec{E},B} u$. For unconditional equations that are strictly B-coherent, confluence of $\vec{E}$ modulo B is equivalent to the Church-Rosser property. But in general this equivalence fails for conditional equations E, in the sense that, even if the rules $\vec{E}$ are confluent modulo B, they may fail to be Church-Rosser:

Example 10. *[24]: Consider the the conditional equational theory (Σ, E) with signature Σ having just three constants a, b, c and with E having the equations $a = c$ and $b = c \;\Leftarrow\; c = a$. Then the CTRS $(\Sigma, \vec{E})$ has rules $a \to c$ and $b \to c \;\Leftarrow\; c \to a$ and is confluent. We obviously have $E \vdash b = c$. However, c, and therefore b, are in $\vec{E}$-normal form, so that $b \not\downarrow_{\vec{E}} c$. That is, $(\Sigma, \vec{E})$ is confluent but fails to have the Church-Rosser property.*

For OSRTs, the equivalence between confluence and the Church-Rosser property can be recovered under the following conditions, which generalize to OSRTs a similar result for CTRSs in [24]:

Theorem 2. *[18] (Church-Rosser Theorem modulo B). Let $\mathcal{R} = (\Sigma, B, \vec{E})$, associated to an order-sorted conditional equational theory $(\Sigma, E \cup B)$, be such that the rules R are: closed under B-extensions, sort-decreasing, weakly terminating modulo B, and such that for each rewrite rule $l \to r \Leftarrow \bigwedge_{i=1..n} u_i \to v_i$ in $\vec{E}$ and $\vec{E}, B$-irreducible substitution θ, the terms $\theta(v_j)$, $1 \leq j \leq n$, are all $\vec{E}, B$-irreducible. Then $\to_{\vec{E},B}$ is confluent modulo B iff for any Σ-terms t, t' we have the equivalence:*

$$t =_{E \cup B} t' \quad \Leftrightarrow \quad t \downarrow_{\vec{E}/B} t'.$$

Furthermore, if under these assumptions $\to_{\vec{E},B}$ is confluent, then $t =_{E \cup B} t'$ iff $t \downarrow_{\vec{E},B} =_B t' \downarrow_{\vec{E},B}$.

[13] This definition of the conditional rules $\vec{E}$ assumes that an equation $u_i = v_i$ in the condition of a conditional equation is interpreted as a *rewrite condition* $u_i \to v_i$ in the associated conditional rule. This seems to leave out the often considered treatment in which an equational condition $u_i = v_i$ is interpreted as a *joinability condition* $u_i \downarrow v_i$ in the associated rule. But a joinability condition can be viewed as a *special case* of a rewrite condition: just transform $(\Sigma, E \cup B)$ into the theory $(\Sigma^{\equiv}, E^{\equiv} \cup B)$, where $\Sigma^{\equiv}$ extends Σ by adding a fresh new sort *Truth* with a single constant tt and with *Truth* unrelated to the other sorts in Σ, and adding for each connected component $[s]$ in Σ a binary equality operator $\_ \equiv \_ : \top_{[s]}\, \top_{[s]} \to \mathit{Truth}$; and where $E^{\equiv}$: (i) adds, for each connected component $[s]$ of Σ, the equation $x \equiv x = tt$ with x of sort $\top_{[s]}$, and, (ii) modifies each conditional equation $t = t' \Leftarrow u_1 = v_1, \ldots, u_n = v_n$ in E into the semantically equivalent equation $t = t' \Leftarrow u_1 \equiv v_1 = tt, \ldots, u_n \equiv v_n = tt$.

The above theorem can be highly misleading, since under the confluence assumption it provides a desirable result, but utterly fails to provide the crucial computational properties needed for declarative functional programming. The desirable result provided is the existence of a *canonical term algebra* $\mathcal{C}_{\vec{E},B}$ *isomorphic*[14] to the initial algebra $\mathcal{T}_{\Sigma/E\cup B}$, where for each sort s, $\mathcal{C}_{\vec{E},B,s}$ is the set of B-equivalence classes of $\vec{E}, B$-irreducible terms having sort s, and for each operation f in Σ, the function $f_{\mathcal{C}_{\vec{E},B}}$ maps a tuple $([t_1], \ldots, [t_n])$ of B-equivalence classes of $\vec{E}, B$-irreducible terms of the appropriate sorts to the B-equivalence class: $[f(t_1, \ldots, t_n)\downarrow_{\vec{E},B}]$.

However, all our hopes about the good computational properties of $\mathcal{C}_{\vec{E},B}$ are dashed to the ground. The *eval* function mapping each term to its canonical form is precisely the unique, surjective Σ-homomorphism:

$$eval : \mathcal{T}_\Sigma \to \mathcal{C}_{\vec{E},B} : t \mapsto [t\downarrow_{\vec{E},B}].$$

Of course, *eval* is uniquely defined as a *mathematical function* by initiality, but in general is *not* a computable function at all! Recall Theorem 1. In particular, one can define confluent type III_n CTRSs (therefore satisfying all the conditions in above Theorem 2) such that the set $\mathsf{Irr}(\mathcal{R})$ is not r.e. [2, 22]. But $\mathsf{Irr}(\mathcal{R})$ is of course the underlying set of the canonical term algebra $\mathcal{C}_{\vec{E}}$ when the rules R of $\mathcal{R}$ are viewed as oriented equations $\vec{E}$. And this makes it impossible for *eval* to be a *computable* function, since its computability would make $\mathsf{Irr}(\mathcal{R})$ r.e.

What is misleading is of course the optical illusion of thinking that the function $eval : t \mapsto [t\downarrow_{\vec{E},B}]$ is a computational process, something an interpreter could perform, when this is in general impossible. A more algebraic way of expressing this *aporia* is to say that what is misleading is the optical illusion of thinking that the canonical term algebra $\mathcal{C}_{\vec{E},B}$ is a *computable algebra*, since in general it is *not* a computable algebra at all. Therefore, both $\mathcal{C}_{\vec{E},B}$ and its associated *eval* function, while mathematically unobjectionable, are *completely useless* and highly misleading from a computational point of view.

What we need to come out of this nightmare is of course *normality*! Here is the right definition, providing what we claim is the most general way possible for conditional OSRTs to define total computable functions by term rewriting, as well as the desired *computational agreement* between the initial algebra semantics of a functional program and its operational semantics by term normalization:

Definition 10. *[18] A normal, strongly deterministic OSRT $\mathcal{R} = (\Sigma, B, R)$, where B satisfies the assumptions in Remark 1, is called* weakly convergent *iff R is: (i) sort-decreasing; (ii) closed under B-extensions; (iii) confluent modulo B; and (iv) weakly normalizing modulo B. If confluence modulo B holds just for ground terms, $\mathcal{R}$ is called* weakly ground convergent*. A weakly convergent and operationally terminating $\mathcal{R}$ is called* convergent*.*

[14] Recall Footnote 8.

Theorem 3. *[18] (Church Rosser Theorem modulo B with Decidable Equality). Let $\mathcal{R} = (\Sigma, B, \vec{E})$, associated to a conditional order-sorted equational theory $(\Sigma, E \cup B)$, be normal, strongly deterministic, sort-decreasing, closed under B-extensions, and weakly normalizing modulo B. Then $\mathcal{R}$ is confluent modulo B iff for any Σ-terms t, t' we have the equivalence:*

$$t =_{E \cup B} t' \;\Leftrightarrow\; t \downarrow_{\vec{E}/B} t'.$$

Furthermore, for any weakly convergent, and therefore Church-Rosser modulo B, $\mathcal{R} = (\Sigma, B, \vec{E})$, with E finite, the equality relation $t =_{E \cup B} t'$ is decidable *by checking whether $t!_{\vec{E}/B} =_B t'!_{\vec{E}/B}$ holds.*

Corollary 4. *[18] (Agreement between Mathematical and Operational Semantics). Let $\mathcal{R} = (\Sigma, B, \vec{E})$, associated to a conditional order-sorted equational theory $(\Sigma, E \cup B)$, be weakly ground convergent with E finite. Then $\mathcal{C}_{\vec{E},B}$ is a computable algebra*[15] *and we have an isomorphism of algebras:*

$$\mathcal{T}_{\Sigma/E \cup B} \;\cong\; \mathcal{C}_{\vec{E},B}.$$

In summary, for functional programs based on conditional equations whose aim is to define *total* recursive functions, the requirements of normality and weak ground convergence are essential and as general as possible. Any further weakening of the conditions will frustrate the overall aim: (i) lack of either ground confluence or strong determinism can compromise the Church-Rosser property; (ii) lack of sort decreasingness can block some computations and destroy the isomorphism $\mathcal{T}_{\Sigma/E \cup B} \cong \mathcal{C}_{\vec{E},B}$; (iii) lack of weak normalization will make functions partial; and (iv) lack of normality will make the only *computable* evaluation function available, namely, the normalization function $norm : t \mapsto t!_{\mathcal{R}}$ partial, whereas under normality and ground weak convergence we have the functional equality $norm = eval$, so that all anomalies evaporate.

4.5. Some Execution Mechanisms for Computing Normal Forms

Given a (not necessarily normal) deterministic OSRT $\mathcal{R}$ and a term t, how can we enumerate all normal forms of t if they exist? In its fullest generality, this question has been answered in Section 4.2. That is, a breadth first search with the meta-level rules in Section 4.1, combined with the background process to check if a normal form has been reached explained in Section 4.2, will enumerate all normal forms of a term t if they exist; but may loop forever if none exist or if the set of normal forms of t is finite and all of them have already been found.

However, this general-purpose search process can be computationally quite expensive. Are there less costly alternatives? And can the search for all normal forms terminate under some assumptions? Without any pretension of giving a

[15] That is, an algebra where both the operations and the equality predicate are computable functions.

systematic answer to these question, we discuss some execution mechanisms — in several cases supported by the Maude system— that provide partial answers to such questions and can be used in practice to compute normal forms.

The first obvious observation is that if $\mathcal{R}$ has a finite set of rules and is operationally terminating, then: (i) it is a fortiori operationally 1-terminating; and (ii) the set of normal forms of a given term t is finite. This means that the simpler and more efficient interpreter for solving normalization problems sketched out in Section 4.2 and supported by Maude can be used to compute the finite set of normal forms of t. In Maude this is achieved by giving the search command: `search` t `=>!` x, where x has the top sort in the connected component of the sort of t. Of course, if besides being operationally terminating $\mathcal{R}$ is confluent, sort-decreasing, and strongly deterministic, Maude's `reduce` command for functional modules is the most efficient execution mechanism available.

A second useful observation is that if $\mathcal{R}$ is confluent and t has a normal form, only one solution exists for the normalization problem $(\exists x)\ t \rightarrow^{!} x$, and the general purpose search process will terminate yielding such a solution if no more solutions are requested. Furthermore, if $\mathcal{R}$ is in addition operationally 1-terminating, the simpler search mechanism supported by Maude can be invoked; and if $\mathcal{R}$ is also weakly normalizing, the command `search [1]` t `=>!` x, with x having the top sort in the connected component will always terminate yielding the unique normal form.

We can illustrate this last case with the following `WEAK-NORM` Maude functional module, which is ground confluent, sort-decreasing, weakly normalizing and operationally 1-terminating, and where `Nat?` is a *supersort* of the sort `Nat`, so that `f` and `g` are functions of sort `Nat?`.

Recall from the Introduction that a Maude functional module has the form: `fmod` (Σ, B, E) `endfm` and is a functional program associated to the equational theory $(\Sigma, E \cup B)$, whose associated rewrite theory $(\Sigma, B, \vec{E})$ is assumed confluent modulo B and is used to try to evaluate a term t to its unique normal from $t!_{\vec{E},B}$ with Maude's `reduce` command. Maude's `search` command cannot be used for functional modules: only for *system modules* of the form `mod` (Σ, B, R) `endm` which specify *concurrent systems*, so that the rules R need not be confluent. However, as we do below, we can transform a functional module `fmod` (Σ, B, E) `endfm` into a system module `mod` $(\Sigma, B, \vec{E})$ `endm` and use the `search` command on this transformed module.

Note the *matching condition* `true := even(N)` in the last conditional equation. This specifies that in the conditional rules $\vec{E}$ associated to the equations E of the module, the equational condition `true = even(N)` in this conditional equation should be oriented as the rewrite condition `even(N) => true` in its corresponding conditional rule.

```
fmod WEAK-NORM is
  protecting BOOL .
  sorts Nat Nat? .
  subsort Nat < Nat? .
  op 0 : -> Nat .              op s : Nat -> Nat .
```

```
  op _+_ : Nat Nat -> Nat .    op even : Nat -> Bool .
  ops f g : Nat? -> Nat? .
  vars N M : Nat .
  eq N + 0 = N .                eq N + s(M) = s(N + M) .
  eq even(0) = true .           eq even(s(0)) = false .
  eq even(s(s(N))) = even(N) .  eq g(N) = N .
  eq f(N) = N + N .
  ceq f(N) = g(f(N)) if true := even(N) .
endfm
```

Giving to Maude the `reduce` command to evaluate the term `f(0)` leads to non-terminating behavior. That is, Maude's innermost strategy for evaluating operationally terminating, confluent and strongly deterministic theories cannot be relied upon to compute normal forms for this module, because it is not operationally terminating.

However, as mentioned above, the unique normal form of any term in this module can be computed by asking for the first solution of a Maude `search` command. Since Maude's `search` command only applies to system modules, a simple theory transformation, easily definable in Maude's `META-LEVEL` module, has first to be performed to transform the given functional module into its associated system module:

```
mod WEAK-NORM is
  protecting BOOL .
  sorts Nat Nat? .
  subsort Nat < Nat? .
  op 0 : -> Nat .                 op s : Nat -> Nat .
  op _+_ : Nat Nat -> Nat .       op even : Nat -> Bool .
  ops f g : Nat? -> Nat? .
  vars N M : Nat .
  rl N + 0 => N .                 rl N + s(M) => s(N + M) .
  rl even(0) => true .            rl even(s(0)) => false .
  rl even(s(s(N))) => even(N) .   rl g(N) => N .
  rl f(N) => N + N .
  crl f(N) => g(f(N)) if even(N) => true .
endm
```

Unique normal forms can now be found by search, without risk of looping thanks to weak normalization and operational 1-termination:

```
Maude> search [1] f(0) =>! N:Nat .
search in WEAK-NORM : f(0) =>! N .

Solution 1 (state 5)
states: 9  rewrites: 12 in 0ms cpu (0ms real) (44943 rewrites/second)
N --> 0
Maude> search [1] f(s(s(0))) =>! N:Nat .
search in WEAK-NORM : f(s(s(0))) =>! N .

Solution 1 (state 14)
```

```
states: 20  rewrites: 35 in 0ms cpu (0ms real) (55118 rewrites/second)
N --> s(s(s(s(0))))
Maude> search [1] f(s(s(s(s(0))))) =>! N:Nat .
search in WEAK-NORM : f(s(s(s(s(0))))) =>! N .

Solution 1 (state 27)
states: 35  rewrites: 70 in 1ms cpu (1ms real) (57189 rewrites/second)
N --> s(s(s(s(s(s(s(0)))))))
```

Considerably more efficient strategies to compute a unique normal form can be used when $\mathcal{R}$ is confluent and satisfies some additional properties. We discuss here two cases.

4.5.1. Use of rewriting strategies

The first is the case of a type III_n CTRS $\mathcal{R}$ in the sense of Bergstra and Klop [2], who show (Theorem 5.7.1) that the parallel outermost strategy will find the irreducible form of a term t if it exists. This theorem, however, has to be interpreted in the light of Theorem 1, since Bergstra and Klop also prove that the set of $\mathcal{R}$-irreducible terms is not r.e. Indeed, the parallel outermost strategy in Theorem 5.7.1 is not constructive, since it must assume "an oracle deciding whether an $\mathcal{R}_u$-redex is an $\mathcal{R}$-redex." But if the set of $\mathcal{R}$-irreducible terms is not r.e., such an oracle is a pipe dream. 1-termination makes such an oracle available: for each $\mathcal{R}_u$-redex $\sigma(\ell)$ of a rule $\ell \to r \Leftarrow c \in \mathcal{R}$ occurring in a term t, we can then compute the whole set T of proof trees associated to the goal $\sigma(\ell) \to \sigma(r)$ whose first applied rule is (Repl). As discussed in Section 4.1, we consider only *sensible* proof trees. We can then guarantee that T is finite. Operational 1-termination ensures that each proof tree will be obtained in finite time. Then, we can check whether T contains a closed proof tree (meaning that the $\sigma(\ell)$ is an $\mathcal{R}$-redex) or not (meaning that $\sigma(\ell)$ is not an $\mathcal{R}$-redex). Once all $\mathcal{R}$-redexes have been computed, we can obtain the subset of outermost $\mathcal{R}$-redexes. Therefore, we have an effective oracle to compute the parallel outermost strategy for a normal type III_n CTRS $\mathcal{R}$.

4.5.2. Use of replacement restrictions

A second case in which an efficient evaluation strategy can be used to compute the normal form of a term t for a confluent OSRT $\mathcal{R}$ is when $\mathcal{R}$ is operationally μ-terminating for μ a replacement map and satisfies some reasonable assumptions. A *context-sensitive* (CS) [12] OSRT is a four-tuple $\mathcal{R} = (\Sigma, B, R, \mu)$, where (Σ, B, R) is an OSRT, and μ, called the *replacement map*, maps each $f : s_1 \cdots s_n \to s$ in Σ to a subset $\mu(f) \subseteq \{1, \ldots, n\}$ of the argument positions of f under which rewriting is allowed. The operational semantics of context-sensitive OSRTs is defined by restricting the inference system of Figure 1 with the single restriction that, in the (Cong) Rule, i with $1 \leq i \leq k$ must furthermore satisfy $i \in \mu(f)$. The one-step and $(0 \leq n)$-step sequents derived by this, more restricted inference system are denoted $t \to_\mu t'$ and $t \to_\mu^\star t'$. The notions of term in μ-normal form and of the set $\mathsf{NF}_\mu(\mathcal{R})$ of terms in μ-normal form are the straightforward generalizations of the normal form notion

and of $\mathsf{NF}(\mathcal{R})$. The lemma below states the required conditions on a CSOSRT $\mathcal{R} = (\Sigma, B, R, \mu)$ yielding the desired agreement between operational and mathematical semantics. This result relies on the notion of μ-sufficient completeness and of the algebra $\mathcal{C}^{\mu}_{\mathcal{R}}$ of terms in μ-normal form (see [9]). Specifically given a context-sensitive rewrite theory $\mathcal{R} = (\Sigma, B, R, \mu)$, a subsignature $\Omega \subseteq \Sigma$ is called a *subsignature of constructors* for $\mathcal{R}$ —which is then said to be *μ-sufficiently complete* for Ω— iff for each Σ-ground term t there is an Ω-ground term u such that $t \rightarrow^{\star}_{\mu} u$. In addition, if $\mathcal{T}_{\Omega} = \mathsf{NF}_{\mu}(\mathcal{R})$, then Ω is called a signature of *free constructors modulo B* for $\mathcal{R} = (\Sigma, B, R, \mu)$.

Lemma 5. *If $\mathcal{R}$ is a confluent, sort decreasing and super-strongly deterministic context-sensitive 3-OSRT $\mathcal{R} = (\Sigma, B, R, \mu)$, which is μ-operationally terminating and μ-sufficiently complete for $\Omega \subseteq \Sigma$ a subsignature of free constructors modulo B, then:*

1. *$\mathcal{R}$ is ground weakly operationally terminating.*
2. *$\mathcal{C}^{\mu}_{\mathcal{R}} \mid_{\Omega} = \mathcal{T}_{\Omega/B}$.*
3. *For each $t \in \mathcal{T}_{\Sigma}$, $t!_{\mathcal{R},B} = t!^{\mu}_{\mathcal{R},B}$, that is, the normal form and the μ-normal form of t (which can be computed by Maude's* reduce *command)* coincide.
4. *$\mathcal{T}_{\Sigma/E \cup B} \simeq \mathcal{C}^{\mu}_{E/B}$ (agreement between operational and denotational semantics).*

Under the assumptions of Lemma 5 plus the operational 1-μ-termination assumption, we can compute normal forms as follows: since Maude supports the execution of operationally 1-μ-terminating confluent context-sensitive strongly deterministic OSRTs $\mathcal{R} = (\Sigma, B, R, \mu)$ specified as functional modules, we can just use Maude's `reduce` command to compute μ-normal forms, which under the assumptions in Lemma 5 are also ordinary normal forms in the underlying OSRT (Σ, B, R). We can illustrate these ideas with the following example of a context-sensitive strongly deterministic OSRT in Maude:

```
fmod FACTORIAL is
  protecting NAT .
  op monus : Nat Nat -> Nat .
  op _~_ : Nat Nat -> Bool [comm] .
  op [_,_,_] : Bool Nat Nat -> Nat [strat (1 0)] .
  op fact : Nat -> Nat .
  vars N M : Nat .
  eq monus(s(N),s(M)) = monus(N,M) .
  ceq monus(N,M) = N if M := 0 .
  ceq monus(N,M) = 0 if N := 0 .
  eq N ~ N = true .
  eq s(N) ~ s(M) = N ~ M .
  eq 0 ~ s(N) = false .
  eq [true,N,M] = N .
  eq [false,N,M] = M .
  eq fact(N) = [(N ~ 0),s(0),N * fact(monus(N,s(0)))] .
endfm
```

This theory, though ground confluent, is clearly non-terminating because of the last equation. Here, μ does not restrict any argument positions, except for the if-then-else operator $[\_,\_,\_]$, where $\mu([\_,\_,\_]) = \{1\}$, as specified by the `strat` attribute. It is, however, operationally μ-terminating and has `0` and `s`, and `true`, `false` as free constructors. Here are some evaluations:

```
Maude> red fact(2) .
reduce in FACTORIAL : fact(2) .
rewrites: 15 in 0ms cpu (0ms real) (192307 rewrites/second)
result NzNat: 2
Maude> red fact(3) .
reduce in FACTORIAL : fact(3) .
rewrites: 21 in 0ms cpu (0ms real) (10500000 rewrites/second)
result NzNat: 6
Maude> red fact(4) .
reduce in FACTORIAL : fact(4) .
rewrites: 27 in 0ms cpu (0ms real) (692307 rewrites/second)
result NzNat: 24
Maude> red fact(5) .
reduce in FACTORIAL : fact(5) .
rewrites: 33 in 0ms cpu (0ms real) (358695 rewrites/second)
result NzNat: 120
```

Can suitable strategies be used also in the non-confluent case to make the search for normal forms more efficient? Yes. Specifically, we can use context-sensitive rewriting in conjunction with search for non-confluent, deterministic and operationally 1-terminating OSRTs to compute the normal forms of a term, provided the sets of normal forms and of μ-normal forms coincide. In Maude, context-sensitive rewriting is also supported for system modules thanks to the `frozen` operator attribute, which specifies the *complement* of a replacement map. That is, the frozen argument positions are exactly those under which rewriting is forbidden. If no argument positions are specified in a `frozen` declaration, *all* positions under the given operator are frozen. For example, in the module below, the operator `h` has its only argument position frozen.

```
mod FROZENNESS is
  sorts Nat Nat? .
  subsorts Nat < Nat? .
  op 0 : -> Nat .
  op s : Nat -> Nat .
  op _+_ : Nat Nat -> Nat .
  op h : Nat? -> Nat? [frozen] .
  op f : Nat? -> Nat? .
  vars n m : Nat .  var x : Nat? .
  rl n + 0 => n .
  rl n + s(m) => s(n + m) .
  rl f(x) => h(f(x)) .
  rl h(h(x)) => h(x) .
  rl h(f(x)) => h(x) .
```

```
  rl h(n) => 0 .
  crl h(n) => s(m) + s(m) if n => s(m) .
endm
```

By the methods in Section 5.1 it is easy to show that this module is ground normal, and that its set of ground normal forms is the set of natural numbers in Peano notation built with the constructors `0` and `s`. It is also easy to show that its set of ground normal forms coincides with that of its ground μ-normal forms, and that the module is operationally 1-terminating. Since Maude's `search` t `=>!` x command computes exactly the set of μ-normal forms of t for μ as specified by the `frozen` declarations, we can then compute the normal forms of any term in the above module more efficiently using the `frozen` attribute for h. In fact, the above module is furthermore operationally μ-terminating, so that the `search` t `=>!` x command will always teminate. It would, however, loop without the `frozen` declaration. Here are a few sample computations:

```
Maude> search f(h(s(s(0)))) =>! x .
search in FROZENNESS : f(h(s(s(0)))) =>! x .

Solution 1 (state 13)
states: 21  rewrites: 25 in 0ms cpu (2ms real) (56053 rewrites/second)
x --> 0

Solution 2 (state 41)
states: 48  rewrites: 82 in 0ms cpu (3ms real) (89227 rewrites/second)
x --> s(s(s(s(0))))

Solution 3 (state 53)
states: 56  rewrites: 100 in 1ms cpu (3ms real) (88731 rewrites/second)
x --> s(s(s(s(s(s(s(s(0))))))))

No more solutions.

Maude> search f(h(s(s(s(s(0)))))) =>! x .
search in FROZENNESS : f(h(s(s(s(s(0)))))) =>! x .

Solution 1 (state 13)
states: 23  rewrites: 29 in 0ms cpu (2ms real) (36989 rewrites/second)
x --> 0

Solution 2 (state 105)
states: 119  rewrites: 253 in 2ms cpu (5ms real) (88214 rewrites/second)
x --> s(s(s(s(s(s(s(s(0))))))))

Solution 3 (state 141)
states: 146  rewrites: 318 in 4ms cpu (6ms real) (76608 rewrites/second)
x --> s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))))))

No more solutions.
```

5. Proving Order-Sorted Rewrite Theories Normal

An interesting feature in the treatment of innermost termination problems using the dependency pair approach [1] is that, since the variables in the right-hand side of the dependency pairs are in normal form, the rules which can be used to connect contiguous dependency pairs are usually a proper subset of the rules in the TRS. This leads to the notion of *usable rules* [1, Definition 32] which simplifies the proofs of innermost termination of rewriting.

In our analysis of normal rewrite theories we have a similar situation: when an irreducible term $t = f(t_1, \ldots, t_k)$ is tested to see whether it is a normal form, in the nontrivial case there will be a conditional rule $\ell \to r \Leftarrow c$ and a substitution σ such that $t = \sigma(\ell)$ and we have to check whether the evaluation of the conditions $\sigma(c)$ gives rise to an infinite well-formed proof tree (for the goal $\sigma(\ell) \to \sigma(r)$) or not. Therefore, if we single out those rules that *can be* involved in any attempt to evaluate $\sigma(c)$, we can obtain a more precise analysis. In the following, we prepare our main definition (Definition 11 below).

Given an order-sorted signature $(\Sigma, S, \leq)$, we define an unsorted signature $\mathcal{F}$, where $f \in \mathcal{F}$ is of *arity* n if and only if $f \in \Sigma_{s_1 \cdots s_n, s}$ for some $s_1, \ldots, s_n, s \in S$. $\mathcal{T}_{\mathcal{F}}(\mathcal{X})$ denotes the set of unsorted terms from the unsorted signature $\mathcal{F}$ and set of unsorted variables $\mathcal{X}$. The *root symbol* $root(t)$ of a term $t \in \mathcal{T}_{\mathcal{F}}(\mathcal{X})$ is given by $root(x) = x$ if $x \in \mathcal{X}$, and $root(t) = f$ if $t = f(t_1, \ldots, t_n)$ for some $f \in \mathcal{F}$ and $t_1, \ldots, t_n \in \mathcal{T}_{\mathcal{F}}(\mathcal{X})$. The obvious (surjective) mapping $\Theta(\_) : \Sigma \to \mathcal{F}$ that *removes* from a symbol $f \in \Sigma$ all sort information is extended to terms in $\mathcal{T}_{\Sigma}(\mathcal{X})$ by structural induction (variables also get all sort information removed). We let $\Theta(t)$ be the *unsorted version* of $t \in \mathcal{T}_{\Sigma}(\mathcal{X})$, i.e., the term $\Theta(t) \in \mathcal{T}_{\mathcal{F}}(\mathcal{X})$ that is obtained by removing all sort information from all function and variable symbols in t.

$$RULES(\mathcal{R}, t) \quad = \quad \{\ell \to r \Leftarrow c \in R \mid \exists p \in \mathcal{P}os(t), root(\Theta(\ell)) = root(\Theta(t|_p))\}$$

Note that $RULES(\mathcal{R}, t)$ contains the rules of $\mathcal{R}$ which are potentially applicable to the subterms in t.

Definition 11 (Usable rules for a rewrite theory). *Let $\mathcal{R} = (\Sigma, B, R)$ be an OSRT. The set of* usable rules *of $\mathcal{R}$ for t is:*

$$\mathcal{U}(\mathcal{R}, t) = RULES(\mathcal{R}, t) \cup \bigcup_{l \to r \Leftarrow c \in RULES(\mathcal{R}, t)} \left(\mathcal{U}(\mathcal{R}^\sharp, r) \cup \bigcup_{s \to t \in c} \mathcal{U}(\mathcal{R}^\sharp, s) \right)$$

where (by abuse of the notation) $\mathcal{R}^\sharp = \mathcal{R} - RULES(\mathcal{R}, t)$.

That is: we consider both unconditional and conditional rules and add the rules that could be *used* to evaluate the righthand sides and the conditions when trying to apply those rules to subterms of t.

Remark 6. *Since we are dealing with OSRTs $\mathcal{R} = (\Sigma, B, R)$, rewriting happens* modulo B. *This raises the issue of whether the above definition of usable rules*

is overly syntactic, that is, not stable under B-equality. The key issue is whether in the (Repl) rule in the inference system of Figure 1 the top symbol of the redex u coincides with that of the lefthand side ℓ. This is the case by requiring the axioms B to be as follows:

$$B = \bigcup_{f:[s_1]\cdots[s_n]\to[s]\in\Sigma} B_f$$

where B_f is a set of equations $u = v$ with $u, v \in \mathcal{T}_{\{f\}}(\mathcal{X}) - \mathcal{X}$, i.e., only symbol f is allowed to (and must) occur in the equations belonging to B_f. Associativity and commutativity axioms satisfy this requirement, which can even be made to work for identity axioms by performing the semantics-preserving transformation described in [5].

Now we can give the main result of this section. For an OSRT $\mathcal{R} = (\Sigma, B, R)$, we say that B *preserves* the $\mathcal{R}$-normal forms if for all $\mathcal{R}$-normal forms t, if $t =_B u$, then u is an $\mathcal{R}$-normal form. Strict *B-coherence*, which is a usual requirement for working OSRTs, implies this property (see Lemma 1). By $\mathcal{R}_C$ we denote the OSRT obtained as the union of $\mathcal{U}(\mathcal{R}, s)$ for all left-hand sides s of conditions $s \to t$ in the rules of $\mathcal{R}$:

$$\mathcal{R}_C = \bigcup_{\ell\to r\Leftarrow c\in\mathcal{R}} \bigcup_{s\to t\in c} \mathcal{U}(\mathcal{R}, s)$$

Theorem 4. *A deterministic 3-OSRT $\mathcal{R} = (\Sigma, B, R)$ is normal if B preserves the $\mathcal{R}$-normal forms and $\mathcal{R}_C$ is operationally terminating.*

Proof. (Sketch) By contradiction. If $\mathcal{R}$ is not normal, then there is an irreducible term t which is not a normal form, i.e., t is not operationally terminating. Since every subterm within an irreducible term is irreducible, we can assume that $t = f(t_1, \ldots, t_k)$ is such that $t_1, \ldots, t_k$ are *normal forms*, i.e., they are irreducible and operationally terminating. Since t is not operationally terminating (but it is not reducible), without loss of generality we can assume that there is a conditional rule $\ell \to r \Leftarrow u_1 \to v_1, \ldots, u_n \to v_n$ for some $n > 0$ such that $t =_B \sigma(\ell)$ for some substitution σ (which is *normalized* for all variables $x \in \mathcal{V}ar(\ell)$ due to our assumptions for B and because B preserves the normal forms of $\mathcal{R}$) and, since t is not a normal form, there is i, $1 \le i \le n$ such that $\sigma(u_i) \not\to^\star_\mathcal{R} \sigma(v_i)$ and an infinite proof attempt is started for some term u in this rewrite sequence, i.e., $\sigma(u_j) \to^\star_\mathcal{R} \sigma(v_j)$ for all $1 \le j < i$, $\sigma(u_i) \to^\star_\mathcal{R} u$ for some term $u = C[\sigma(\ell')]$ and rule $\ell' \to r' \Leftarrow c'$ (with a nonempty conditional part c') such that $C[\sigma(\ell')] \to C[\sigma(r')]$ is the root of an infinite proof tree of the following general form

$$\dfrac{\dfrac{T}{\vdots}\,{\scriptstyle(\text{Cong})}}{C[\sigma(\ell')] \to C[\sigma(r')]}\,{\scriptstyle(\text{Cong})}$$

where T is an infinite proof tree with $root(T) = \sigma(\ell') \to \sigma(r')$ and (Repl) is applied with rule $\ell' \to r' \Leftarrow c'$. This means that $\sigma(u_i)$ is not operationally

terminating. Since $\sigma(x)$ is normalized for all $x \in \mathcal{V}ar(\ell)$, the determinism of $\mathcal{R}$ guarantees that, for all i, $1 \leq j < i$, $\sigma(s_i) \rightarrow^*_{\mathcal{R}_C} \sigma(t_i)$ and $\sigma(s_i) \rightarrow^*_{\mathcal{R}_C} u$. Furthermore, $\ell' \rightarrow r' \Leftarrow c' \in \mathcal{R}_C$ and any redex occurring in $\sigma(x)$ for variables in $x \in \mathcal{V}ar(\ell' \rightarrow r' \Leftarrow c')$ is an $\mathcal{R}_C$-redex (i.e., it is an instance of a rule in $\mathcal{R}_C$). Therefore, $\mathcal{R}_C$ is not operationally terminating, leading to a contradiction. □

Example 11. *Consider the functional module* `WEAK-NORM` *in Section 4.5. Here,* $\mathcal{R}_C$ *is the unconditional subOSRT consisting of the rules defining* `even`*. Note that* $\mathcal{R}_C$ *has no conditional rule and is clearly terminating, hence operationally terminating. We conclude that, as claimed,* `WEAK-NORM` *is a normal OSRT.*

Example 12. *Consider the following OSRT* $\mathcal{R}$ *defining a function* `allEven` *which checks whether all components of a list of numbers (in Peano's notation) are* even *numbers:*

```
mod AllEVEN is
  protecting BOOL .
  sorts Nat LNat .
  op 0 : -> Nat .
  op s : Nat -> Nat .
  op even : Nat -> Bool .
  ops nil nats : -> LNat .
  op _:_ : Nat LNat -> LNat .
  op from : Nat -> LNat .
  op take : Nat LNat -> LNat .
  op allEven : LNat -> Bool .
  var n x : Nat .
  var xs : LNat .
  rl even(0) => true .
  rl even(s(s(n))) => even(n) .
  rl nats => from(0) .
  rl from(n) => n : from(s(n)) .
  rl take(0,xs) => nil .
  rl take(s(n),x : xs) => x : take(n,xs) .
  rl allEven(nil) => true .
  crl allEven(x : xs) => allEven(xs) if even(x) => true .
endm
```

Although $\mathcal{R}$ *is* not *terminating (due to rule defining symbol* `from`*), it is* orthogonal *and the right-hand side of the condition in the (only) conditional rule is a* ground *term which is a* normal form *with respect to the* underlying *OSRT. Thus,* $\mathcal{R}$ *is a* III_n *OSRT. Note that* $\mathcal{R}_C$ *consists of the following rules:*

$$\begin{aligned} \texttt{even(0)} &\rightarrow \texttt{true} \\ \texttt{even(s(s(n)))} &\rightarrow \texttt{even(n)} \end{aligned}$$

and is easily proved terminating. *Since* $\mathcal{R}_C$ *has no conditional rules, it is operationally terminating. By Theorem 4,* $\mathcal{R}$ *is* normal. *Furthermore, it is not*

difficult to see that $\mathcal{R}$ *is actually operationally* 1*-terminating. We reason as follows: only terms* t *of sort* `LNat` *can be nonterminating; and the only way for* $\mathcal{R}$ *to be operationally non-*1*-terminating is that some expression* $\sigma(\ell)$*, for* $\ell \to r \Leftarrow c$ *the only conditional rule in* $\mathcal{R}$ *and an appropriate (sorted) substitution* σ*, initiates an infinite proof tree with root*

$$\sigma(\texttt{allEven(x:xs)}) \to \sigma(\texttt{allEven(xs)}) \tag{20}$$

However, since variable `x` *can only be instantiated to values of sort* `Nat`*, and expressions of sort* `Nat` *are all terminating, the evaluation of the condition* $\sigma(\texttt{even(x)}) \to^\star \texttt{true}$ *always terminates. Hence, there can be no infinite proof tree for goal (20).*

Following the discussion in Section 4.5.1 if a term t *has a reduction sequence* $t \to^\star_{\mathcal{R}} \bar{t}$ *for some* irreducible term $\bar{t}$*, then the* parallel-outermost reduction strategy *which reduces the parallel-outermost* $\mathcal{R}_u$*-redexes is able to obtain* $\bar{t}$*. For instance,* `allEven(take(s(0),nats))` *has a normal form* `true` *which can be obtained by using the parallel-outermost reduction strategy:*

$$\begin{aligned}
\texttt{allEven(take(s(0),}\underline{\texttt{nats}}\texttt{))} &\to_{po} \texttt{allEven(take(s(0),}\underline{\texttt{from(0)}}\texttt{))} && (21)\\
&\to_{po} \texttt{allEven(}\underline{\texttt{take(s(0),0:from(s(0)))}}\texttt{)} && (22)\\
&\to_{po} \underline{\texttt{allEven(0:take(0,from(s(0)))}} && (23)\\
&\to_{po} \texttt{allEven(}\underline{\texttt{take(0,from(s(0)))}}\texttt{)} && (24)\\
&\to_{po} \underline{\texttt{allEven(nil)}} && (25)\\
&\to_{po} \texttt{true} && (26)
\end{aligned}$$

Note that step (24) is an application of the conditional rule where an 'administrative' step `even(0)` $\to$ `true` *is performed for the evaluation of the condition.*

Also, term $t =$ `allEven(take(s(s(0)),nats))` *has an* $\mathcal{R}$*-normal form*

`allEven(s(0):nil)`.

Note, however, that it is not *an* $\mathcal{R}_u$*-normal form. Actually, we could further rewrite* `allEven(s(0):nil)` *into* `true` *by using* $\mathcal{R}_u$*, but this would yield the* wrong *conclusion that* `take(s(s(0)),nats)` *consists of even numbers only. In other words,* $\mathcal{R}$ *and* $\mathcal{R}_u$ *are not semantically equivalent. However, the* $\mathcal{R}$*-normal form of* t *can be obtained by using the parallel-outermost reduction strategy.*

$$\begin{aligned}
\texttt{allEven(take(s(s(0)),}\underline{\texttt{nats}}\texttt{))} &\to_{po} \texttt{allEven(take(s(s(0)),}\underline{\texttt{from(0)}}\texttt{))} && (27)\\
&\to_{po} \texttt{allEven(}\underline{\texttt{take(s(s(0)),0:from(s(0)))}}\texttt{)} && (28)\\
&\to_{po} \underline{\texttt{allEven(0:take(s(0),from(s(0)))}} && (29)\\
&\to_{po} \texttt{allEven(take(s(0),}\underline{\texttt{from(s(0))}}\texttt{))} && (30)\\
&\to_{po} \texttt{allEven(}\underline{\texttt{take(s(0),s(0):from(s(s(0))))}}\texttt{)} && (31)\\
&\to_{po} \texttt{allEven(s(0):take(0,}\underline{\texttt{from(s(s(0)))}}\texttt{))} && (32)\\
&\to_{po} \texttt{allEven(s(0):}\underline{\texttt{take(0,s(s(0)):from(s(s(s(0)))))}}\texttt{)} && (33)\\
&\to_{po} \texttt{allEven(s(0):nil)} && (34)
\end{aligned}$$

Note that in step (33) the whole term

$$\texttt{allEven(s(0):take(0,from(s(s(0)))))}$$

is the outermost $\mathcal{R}_u$-redex *(of the unconditional version of the conditional rule in $\mathcal{R}$). However, it is* not *an $\mathcal{R}$-redex because the condition* $\texttt{even(s(0))} \to^\star \texttt{true}$ *does not hold.*

Finally, we show that Theorem 4 does *not* characterize normal OSRTs:

Example 13. *Consider the following deterministic* 1*-CTRS:*

$$\begin{array}{lcl} a & \to & b \\ b & \to & a \end{array} \qquad f(x) \to x \Leftarrow c \to d, a \to c$$

Every term $f(t)$ is irreducible *and also a normal form because the unsatisfiable condition $c \to d$ prevents the looping evaluation of the condition $a \to c$. However, $\mathcal{R}_C = \{a \to b, b \to a\}$ is not (operationally) terminating.*

Remark 7. *In general, Theorem 4* cannot *be used to prove the operational* 1*-termination of an OSRT and then its normality as a* consequence *of operational* 1*-termination. For instance, the* unsorted *version of $\mathcal{R}$ in Example 12 is not operationally* 1*-terminating (for instance, we can then bound the variable x in the conditional rule to the nonterminating expression* `nats`*, thus initiating an infinite proof tree for the evaluation of the condition). However, $\mathcal{R}_C$ is still terminating in this case. Thus, $\mathcal{R}$ is still proved normal (using Theorem 4), but it is not operationally* 1*-terminating.*

5.1. Proving Ground Normality

In order to prove that a deterministic OSRT $\mathcal{R} = (\Sigma, B, R)$ is ground normal, we can proceed as follows:

1. Identify a subsignature of constructors Ω with nonempty sorts such that the rules in R decompose as a disjoint union $R_{(\Sigma-\Omega)} \cup R_\Omega$, where the R_Ω have only Ω terms in their rules and conditions, and each $\ell \to r \Leftarrow s_1 \to t_1, \ldots, s_n \to t_n$ in $R_{(\Sigma-\Omega)}$ has $l = f(t_1, \ldots, t_n)$ for some $f \in \Sigma - \Omega$. We also assume that the axioms B decompose as a disjoint union $B_{(\Sigma-\Omega)} \cup B_\Omega$ with the B_Ω involving only Ω terms, and the $B_{(\Sigma-\Omega)}$ *not* Ω-equations. This yields an OSRT inclusion $\mathcal{R}_\Omega \subseteq \mathcal{R}$, with $\mathcal{R}_\Omega = (\Omega, B_\Omega, R_\Omega)$.
2. Prove (by inductive theorem proving) that for all defined symbols $f \in \Sigma - \Omega$, say with rank $f : s_1 \cdots s_n \to s$, the following inductive property holds:
$$\forall x_1 \in \mathcal{T}_{\Omega_{s_1}}, \ldots, x_n \in \mathcal{T}_{\Omega_{s_n}}, \exists y f(x_1, \ldots, x_n) \to_\mathcal{R} y$$

Then, if $\mathcal{R}_\Omega$ is a ground normal theory, $\mathcal{R}$ is *ground normal* and, furthermore, $\mathsf{GNF}(\mathcal{R}) \subseteq \mathcal{T}_\Omega$. This is because, by (2), any ground term t having some symbols in $\Sigma - \Omega$ is always reducible (at an innermost occurence of any such symbol), so that $\mathsf{GIrr}(\mathcal{R}) \subseteq \mathcal{T}_\Omega$. But since the only rules that can apply to terms in $\mathcal{T}_\Omega$ are

those in $\mathcal{R}_\Omega$, which is a ground normal theory, we have $\mathsf{GIrr}(\mathcal{R}) = \mathsf{GIrr}(\mathcal{R}_\Omega) \cap \mathcal{T}_\Omega = \mathsf{GNF}(\mathcal{R}_\Omega) \cap \mathcal{T}_\Omega = \mathsf{GNF}(\mathcal{R}) \cap \mathcal{T}_\Omega = \mathsf{GNF}(\mathcal{R})$, as desired.

The above method can be easily generalized to a method for proving the ground μ-normality of a CSOSRT $\mathcal{R} = (\Sigma, B, R, \mu)$. The only differences are that: (i) in step (2) the inductive properties we have to prove are of the form:

$$\forall x_1 \in \mathcal{T}_{\Omega_{s_1}}, \ldots, x_n \in \mathcal{T}_{\Omega_{s_n}}, \exists y f(x_1, \ldots, x_n) \to_\mu y,$$

(ii) $\mathcal{R}_\Omega$ is now assumed to be a ground μ-normal theory; and (iii) we now get $\mathsf{GNF}^\mu(\mathcal{R}) \subseteq \mathcal{T}_\Omega$.

Example 14. *We can apply this method to prove that, disregarding its replacement map μ, the module* `FROZENNESS` *in Section 4.5.2 is ground normal. Choose as Ω the the signature with just* `0` *and* `s`*, so that $\mathcal{R}_\Omega$ has no rules and is trivially ground normal. Property (2) follows for* `+` *by structural induction on the second argument, is trivial for* `f` *because of the rule* `f(x) => h(f(x))`*, and is also trivial for* `h` *because of the rule* `h(n) => 0` *and the fact that all Ω-terms have sort* `Nat`*, which is the sort of the variable* `n`*. Therefore,* `FROZENNESS` *is ground normal and has $\mathcal{T}_\Omega$, i.e., the natural numbers in Peano notation, as its set of ground normal forms.*

6. Orderings, Quasi-Decreasingness, and Operational Termination

A binary relation R on a set A is *terminating* (or well-founded) if there is no infinite sequence $a_1\ R\ a_2\ R\ a_3 \cdots$. Given $f : A^k \to A$ and $i \in \{1, \ldots, k\}$, we say that f is *i-monotonic* on its i-th argument (or that f is *i-monotone* with respect to R) if $f(x_1, \ldots, x_{i-1}, x, \ldots, x_k)\, R\, f(x_1, \ldots, x_{i-1}, y, \ldots, x_k)$ whenever $x\, R\, y$, for all $x, y, x_1, \ldots, x_k \in A$. We say that R is *monotonic* if, for all symbols f, f is monotonic for all its arguments w.r.t. R. In [13] we have shown that operational termination of deterministic 3-CTRSs (which are special deterministic 3-OSRTs where the set of sorts S contains a single sort and the set of equations B is empty) is equivalent to *quasi-decreasingness*, i.e., the existence of a well-founded partial ordering $\succ$ on terms satisfying that: (1) the one-step rewriting relation is contained in $\succ$: $\to_\mathcal{R} \subseteq \succ$, (2) the strict subterm relation is contained in $\succ$: $\rhd \subseteq \succ$, and (3) for every rule $\ell \to r \Leftarrow s_1 \to t_1, \ldots, s_n \to t_n$, substitution σ, and index i, $0 \leq i < n$, if $\sigma(s_j) \to^\star_\mathcal{R} \sigma(t_j)$ for every $1 \leq j \leq i$, then $\sigma(l) \succ \sigma(s_{i+1})$. In the following, we generalize this result to deterministic 3-OSRTs under the assumptions on B stated in Section 2. First we address the problem of defining appropriate orderings for dealing with order-sorted terms and rewrite theories.

6.1. Orderings for Order-Sorted Terms

A strict ordering $\succ_s$ on terms of sort s is an irreflexive and transitive binary relation on $\mathcal{T}_\Sigma(\mathcal{X})_s$. In particular, a strict ordering $\succ_{\top_{[s]}}$, abbreviated to $\succ_{[s]}$, is a transitive binary relation on $\mathcal{T}_\Sigma(\mathcal{X})_{\top[s]}$, that is, on *all* terms in the connected component $[s] \in S/\equiv_\leq$.

Remark 8. *Order-sorted rewriting proceeds by transforming terms in the same connected component* $[s] \in S/\equiv_{\leq}$. *Therefore, orderings* $\succ_{[s]}$ *indexed by top sorts* $\top_{[s]}$ *rather than by sorts are more appropriate for compatibility with the order-sorted rewrite relation. Indeed, note that* $\rightarrow^{+}_{\mathcal{R}} = \bigcup_{s \in S}(\rightarrow^{+}_{\mathcal{R}_{[s]}})$ *is a well-founded* S*-ordering if* $\rightarrow_{\mathcal{R}}$ *is terminating, and that, by the kind-completeness assumption about* Σ *in* $\mathcal{R} = (\Sigma, B, R)$*, it is monotonic. On the other hand, we can always obtain an ordering* $\succ_s$ *on terms of sort* s *as follows:* $\succ_s = \succ_{[s]} \cap\, \mathcal{T}_{\Sigma}(\mathcal{X})^2_s$.

A strict S-ordering $\succ_S = \{\succ_{[s]}\}_{[s] \in S/\equiv_{\leq}}$ is an $(S/\equiv_{\leq})$-sorted strict ordering on $\mathcal{T}_{\Sigma}(\mathcal{X})$, i.e., given terms $u, v \in \mathcal{T}_{\Sigma}(\mathcal{X})$, $u \succ_S v$ if and only if $u, v \in \mathcal{T}_{\Sigma}(\mathcal{X})_{\top_{[s]}}$ for some $[s] \in S/\equiv_{\leq}$ and $u \succ_{[s]} v$. An S-ordering $\succ_S$ is: *well-founded* if its components $\succ_{[s]}$ are well-founded for all $[s] \in S/\equiv_{\leq}$; *stable* if for all S-sorted substitution σ, $s \in S$, and terms $u, v \in \mathcal{T}_{\Sigma}(\mathcal{X})_{\top_{[s]}}$ $u \succ_{[s]} v$, then $\sigma(u) \succ_{[s]} \sigma(v)$; *monotonic* if for all $f : s_1 \cdots s_k \rightarrow s \in \Sigma$ and terms $u_i, v_i \in \mathcal{T}_{\Sigma}(\mathcal{X})_{\top_{[s_i]}}$ for $1 \leq i \leq k$, if $u_i \succ_{[s_i]} v_i$, then $f(u_1, \ldots, u_i, \ldots, u_k) \succ_{[s]} f(u_1, \ldots, v_i, \ldots, u_k)$. An S-ordering $\succ_S$ on $\mathcal{T}_{\Sigma}(\mathcal{X})$ is compatible with a set of equations B on $\mathcal{T}_{\Sigma}(\mathcal{X})$ if for all terms u, u', v, whenever $u \succ_S v$ and $u' =_B u$, we have $u' \succ_S v$ (in short: $=_B ; \succ \subseteq \succ$, where ';' is the *composition* of relations: $a\ R ; S\ b$ iff there is c such that $a\ R\ c$ and $c\ S\ b$). The previous definitions generalize to arbitrary relations (quasi-orderings $\gtrsim$, equivalences $\approx$, etc.) on order-sorted terms.

Remark 9. *S-sorted orderings cannot compare terms in different connected components. Still, S-sorted orderings are the natural ones when comparing the left- and right-hand sides of the rules of an order-sorted (conditional) rewrite system.*

A term ordering $\succ$ is a strict order on $\mathcal{T}_{\Sigma}(\mathcal{X})$. An S-sorted ordering $\succ_S$ on $\mathcal{T}_{\Sigma}(\mathcal{X})$ defines a term ordering on $\mathcal{T}_{\Sigma}(\mathcal{X})$: $u \succ v$ iff $\exists [s] \in S/\equiv_{\leq}$ such that $u \succ_{[s]} v$. A term ordering which is *not* S-sorted is the strict subterm relation $\rhd$: $\forall u, v \in \mathcal{T}_{\Sigma}(\mathcal{X})$, $u \rhd v$ if $u = f(u_1, \ldots, u_k)$ for some $f : s_1 \cdots s_k \rightarrow s \in \Sigma$ and either $v = u_i$ or $u_i \rhd v$ for some i, $1 \leq i \leq k$. We write $u \trianglerighteq v$ if $u \rhd v$ or $u = v$.

The following result is the many-sorted version of the well-known Lankford's theorem for unsorted (and unconditional) rewriting; we use it to motivate the improvements in proofs of termination that the presence of sorts can make possible. Note that, in the many-sorted case, the poset of sorts is discrete and we can identify S with $S/\equiv_{\leq}$.

Theorem 5. *A many-sorted TRS* $\mathcal{R}$ *is terminating if and only if there is a stable and monotonic* S*-ordering* $\succ_S$ *such that* $\ell \succ_s r$ *for all rules* $\ell \rightarrow r$ *with* $\ell, r \in \mathcal{T}_{\Sigma}(\mathcal{X})_s$.

The following example illustrates an interesting feature of S-sorted orderings which is due to the presence of sorts in the signature.

Example 15. *Consider the following Maude functional module:*

```
fmod ListsAndTake is
```

```
    sorts Nat NatList .
    op 0 : -> Nat .
    op s : Nat -> Nat .
    op nil : -> NatList .
    op cons : Nat NatList -> NatList .
    op take : Nat NatIList -> NatList .
    vars M N : Nat .
    var  L : NatList .
    eq take(0, L) = nil .
    eq take(s(M), cons(N, L)) = cons(N, take(M, L)) .
endfm
```

where sort `NatList` *is intended to classify finite lists of natural numbers. The function* `take` *can be used to obtain an initial segment of a list by giving the number of items we want to extract. The S-sorted ordering $\succ_S$ where $\succ_{\texttt{Nat}} = \varnothing$ and $\succ_{\texttt{NatList}}$ is generated by the following polynomial interpretation over the naturals:*

$$\begin{array}{ccc} [\![\texttt{0}]\!] = 1 & [\![\texttt{s}]\!](x) = x+1 & [\![\texttt{nil}]\!] = 0 \\ [\![\texttt{cons}]\!](x,y) = y & [\![\texttt{take}]\!](x,y) = x+y & \end{array}$$

by $u \succ_{\texttt{NatList}} v$ if $\forall \vec{x}([\![u]\!] >_{\mathbb{N}} [\![v]\!])$ for all $s \in S$ and $u, v \in \mathcal{T}_\Sigma(\mathcal{X})_s$, where $\vec{x} = x_1, \ldots, x_n$ and $\{x_1, \ldots, x_n\}$ is the set of variables occurring in u or v (without any sort information, all ranging over the naturals). We can use it to prove termination of the previous module. We have:

1. $[\![\texttt{take(0,L)}]\!] = [\![\texttt{0}]\!] + [\![\texttt{L}]\!] = 1 + L$ *and* $[\![\texttt{nil}]\!] = 0$*;*
2. $[\![\texttt{take(s(M),cons(N, L))}]\!] = [\![\texttt{take}]\!]([\![\texttt{s(M)}]\!], [\![\texttt{cons(N,L)}]\!]) = M + L + 1$ *and* $[\![\texttt{cons(N,take(M,L))}]\!] = [\![\texttt{cons}]\!]([\![\texttt{N}]\!], [\![\texttt{take(M,L)}]\!]) = M + L$

Since (variables L, M, and N are universally quantified over the naturals):

$$\begin{array}{rcl} [\![\texttt{take(0,L)}]\!] = 1 + L & >_{\mathbb{N}} & 0 = [\![\texttt{nil}]\!] \\ [\![\texttt{take(s(M),cons(N,L))}]\!] = M + L + 1 & >_{\mathbb{N}} & M + L = [\![\texttt{cons(N,take(M,L)}]\!] \end{array}$$

we conclude that, as desired, $\texttt{take(0,L)} \succ_{\texttt{NatList}} \texttt{nil}$ *and*

$$\texttt{take(s(M),cons(N,L))} \succ_{\texttt{NatList}} \texttt{cons(N,take(M,L))}.$$

The interesting fact is that, in contrast with the unsorted case, $\succ_{\texttt{NatList}}$ is monotonic despite the fact that cons does not *take into account the information of the first argument (of sort* `Nat`*). This is possible because $\succ_{\texttt{Nat}} = \varnothing$, which makes sense because we do not need to compare terms of sort* `Nat` *in a TRS without rules of sort* `Nat`*!*

6.2. Quasi-Decreasingness and (Strong) Operational Termination of deterministic 3-OSRTs

After the previous discussion, we can provide a generalization to deterministic 3-OSRTs of the usual notion of quasi-decreasingness for deterministic 3-CTRSs.

Definition 12 (Quasi-decreasingness). *A deterministic 3-OSRT* (Σ, B, R) *is* quasi-decreasing *if there is a well-founded term ordering* $\succ$ *on* $\mathcal{T}_\Sigma(\mathcal{X})$ *satisfying: (1)* $\to_\mathcal{R} \subseteq \succ$*, (2)* $=_B;\succ \subseteq \succ$*, (3)* $\rhd \subseteq \succ$*, and (4) for every rule* $l \to r \Leftarrow u_1 \to v_1, \ldots, u_n \to v_n$*, substitution* σ*, and index* i*,* $0 \leq i < n$*, if* $\sigma(u_j) \to^\star_\mathcal{R} \sigma(v_j)$ *for every* $1 \leq j \leq i$*, then* $\sigma(l) \succ \sigma(u_{i+1})$*.*

Quasi-decreasingness is a sufficient condition for operational termination of deterministic 3-OSRTs.

Theorem 6. *Let* $\mathcal{R}$ *be a deterministic 3-OSRT. If* $\mathcal{R}$ *is quasi-decreasing, then it is operationally terminating.*

Proof. Let $\mathcal{R} = (\Sigma, B, R)$. Since $\mathcal{R}$ is quasi-decreasing, there is a well-founded partial ordering $\succ$ which is compatible with $=_B$, includes both $\to_\mathcal{R}$ and $\rhd$, and such that for every rule $l \to r \Leftarrow u_1 \to v_1, \ldots, u_n \to v_n$, substitution σ, and index i, $0 \leq i \leq n$, if $\sigma(u_j) \to^\star_\mathcal{R} \sigma(v_j)$ for every $1 \leq j \leq i$, then $\sigma(l) \succ \sigma(u_{i+1})$.

Assume that $\mathcal{R}$ is not is operationally terminating. Then, there is an infinite well-formed proof tree T for some judgement G (i.e., $G = \mathit{root}(T)$). We prove that there is a judgement G' at level 1 or 2 in T which is the root of an infinite well-formed proof tree T' and such that $\mathit{left}(G) = u \succ u' = \mathit{left}(G')$, where $\mathit{left}(u \to v) = u$ and $\mathit{left}(u \to^\star v) = u$.

1. In case $G = u \to v$, G must be the root of an instance of an inference rule Δ, where Δ is either (*Cong*) or (*Repl*).
   (a) If Δ is the Congruence rule, then $u = f(u_1, \ldots, u_i, \ldots, u_n)$, $v = f(u_1, \ldots, u'_i, \ldots, u_n)$ and there is an infinite well-formed proof tree T' (at level 1 in T) whose root is $G' = \mathit{root}(T') = u_i \to u'_i$. Since $\rhd \subseteq \succ$, we have $\mathit{left}(G) = s \succ u_i = \mathit{left}(G')$ as required.
   (b) If Δ is the Replacement rule, then $u =_B \sigma(l)$ for some conditional rule $\ell \to r \Leftarrow u_1 \to v_1, \ldots, u_n \to v_n$ and substitution σ and there is $i \in \{1, \ldots, n\}$ such that $T_1, \ldots, T_{i-1}$ are well-formed finite proof trees with $\mathit{root}(T_j) = \sigma(u_j) \to^\star \sigma(v_j)$ for $1 \leq j < i$ (and hence, $\sigma(u_j) \to^\star_\mathcal{R} \sigma(v_j)$ for all $1 \leq j < i$) and T_i is infinite. Let $G_i = \mathit{root}(T_i) = \sigma(u_i) \to^\star \sigma(v_i)$. Now, we have $u =_B \sigma(l) \succ \sigma(u_i)$. Thus, we let $G' = G_i$ and (using the compatibility of $\succ$ and $=_B$) conclude $\mathit{left}(G) = u \succ \sigma(u_i) = \mathit{left}(G')$.
2. Now consider the case $G = u \to^\star v$. Since T is infinite, the only possibility is to use Transitivity. Thus, we have two proof trees T_1 and T_2 such that either T_1 or T_2 are infinite. Here, $G_1 = \mathit{root}(T_1) = u \to u'$ for some term u' and $G_2 = \mathit{root}(T_2) = u' \to^\star v$.
   (a) If T_1 is infinite, then $\mathit{left}(G) = u = \mathit{left}(G_1)$. We can apply item 1 above to G_1 and T_1 to conclude that there is a judgement G' at level 1 in T_1 (i.e., at level 2 in T) such that $\mathit{left}(G) = u \succ u'' = \mathit{left}(G')$.
   (b) If T_1 is finite, then $u \to_\mathcal{R} u'$, i.e., $u \succ u'$ and T_2 is infinite. Then, we let $G' = G_2$ and conclude $\mathit{left}(G) = u \succ u' = \mathit{left}(G')$.

Thus, given the well-formed infinite proof tree T, we can define an infinite sequence $u = u_1 \succ u_2 \succ \cdots$ which contradicts the well-foundedness of $\succ$. $\square$

Quasi-decreasingness is also necessary for operational termination of an order-sorted rewrite theory $\mathcal{R} = (\Sigma, B, R)$. Recall from Section 2 that, since any order-sorted signature can be transformed into a kind-complete one without changing its deductions [19], we have assumed throughout that Σ is always *kind-complete*. This assumption is already implicit in the inference system of Figure 1, since a more involved inference system would be needed for signatures that are not kind-complete to avoid the production of ill-formed term by rewriting [19]. The proof that quasi-decreasingness is also necessary exploits the monotonicity of rewriting ensured for kind-complete signatures by by the Congruence rule in the inference system of Figure 1.

In the following results, given a set of equations B, we consider the following *B-subterm* relation $\rhd_B = (=_B; \rhd; =_B)$. Note that $\rhd \subseteq \rhd_B$ due to reflexivity of $=_B$. However, $\rhd_B$ need not be well-founded, as the following example shows.

Example 16. *Consider B given by*

$$f(a) \quad = \quad a \tag{35}$$

Note that $a =_B f(a) \rhd a$, i.e., $a \rhd_B a$. Thus, $\rhd_B$ is not well-founded.

For sets of equations B corresponding to commonly used theories such as any combination of *associativity*, *commutativity*, and *associativity-commutativity* axioms, $\rhd_B$ is clearly well-founded. Indeed, this follows from the fact that such axioms B are *size-preserving*, i.e., if $u =_B v$, then $|u| = |v|$, where $|t|$ is the number of nodes of t as a tree.

Proposition 3. *Let $\succ_S$ be a monotonic S-sorted relation on $T \subseteq \mathcal{T}_\Sigma(\mathcal{X})$ and B a set of equations such that $\rhd_B$ is well-founded. If $\succ_S$ is well-founded on T and $=_B; \succ_S \subseteq \succ_S; =_B$, then $\succ = (=_B; (\rhd_B \cup \succ_S)^+; =_B)$ is a well-founded term ordering on $\mathcal{T}_\Sigma(\mathcal{X})$.*

Proof. The proof is analogous to Lemma 7.2.4 in [20], which we borrow here. Note that $\succ$ is transitive: if $s \succ t$ and $t \succ u$, then $s =_B s'(\rhd_B \cup \succ_S)^+ s'' =_B t$ and $t =_B t'(\rhd_B \cup \succ_S)^+ t'' =_B u$ for some terms s', s'', t', t''. Since $=_B; \succ_S \subseteq \succ_S; =_B$ and $=_B; \rhd_B \subseteq \rhd_B; =_B$, we have $s =_B s'(\rhd_B \cup \succ_S)^+ t$ and $t(\rhd_B \cup \succ_S)^+ t'' =_B u$, i.e, $s =_B s'(\rhd_B \cup \succ_S)^+ t'' =_B u$, hence $s \succ u$. Well-foundedness follows by contradiction. If $\succ$ is not well-founded, then there is an infinite sequence

$$u_1 \ =_B; (\rhd_B \cup \succ_S)^+; =_B \ u_2 \ =_B; (\rhd_B \cup \succ_S)^+; =_B \ u_3 \ =_B; (\rhd_B \cup \succ_S)^+; =_B \ \cdots$$

for terms u_i of sorts s_i. Furthermore, by using compatibility of both $\succ_S$ and $\rhd_B$ with $=_B$, we can write instead the following infinite sequence:

$$u_1 \, (\rhd_B \cup \succ_S) \, u_2 \, (\rhd_B \cup \succ_S) \, u_3 \, (\rhd_B \cup \succ_S) \, \cdots$$

If there is only a finite number of steps of $\succ_S$, then there would be an infinite $\rhd_B$ sequence, contradicting well-foundedness of $\rhd_B$. Thus, there must be an infinite number of $\succ_S$-steps. Since $\rhd \subseteq \rhd_B$, each subsequence $C[u]\rhd u \succ_S v$, where $C[u]$ is a term of sort s', can by monotonicity be rewritten into $C[u] \succ_S C[v]\rhd_B v$ (i.e., pushing back the comparisons with $\succ_S$; note that u and v belong to the same sort s). By proceeding in this way, we obtain an infinite decreasing sequence for $\succ_{[s_1]}$ (where s_1 is the sort of u_1), thus leading to a contradiction. □

Theorem 7. *Let $\mathcal{R} = (\Sigma, B, R)$ be a deterministic 3-OSRT such that $\rhd_B$ is well-founded. If $\mathcal{R}$ is operationally terminating, then it is quasi-decreasing.*

Proof. If $\mathcal{R}$ is operationally terminating, then $\to_{\mathcal{R}}$ is a well-founded S-sorted relation which (by monotonicity) is closed under contexts. Furthermore, by our assumption of strict coherence, if $s =_B s'$ and $s \to_{\mathcal{R}} t$, then there is t' such that $s' \to_{\mathcal{R}} t'$ and $t' =_B t$, i.e., we have $=_B;\to_{\mathcal{R}} \subseteq \to_{\mathcal{R}};=_B$. Note that $\to_{\mathcal{R}} = \cup_{s\in S} \to_{\mathcal{R},s}$, where $\to_{\mathcal{R},s}$ is the rewrite relation restricted to terms of sort s. By Proposition 3, the relation $=_B;(\to_{\mathcal{R}} \cup \rhd_B)^+;=_B$ is a well-founded partial ordering on terms. We write $u \rightsquigarrow v$ if there is a conditional rule $\ell \to r \Leftarrow u_1 \to v_1, \cdots, u_n \to v_n$, a position p of u, a substitution σ and $i \in \{1, \ldots, n\}$ such that $u|_p =_B \sigma(\ell)$, $v = \sigma(u_i)$ and $\sigma(u_j) \to^\star_{\mathcal{R}} \sigma(v_j)$ for all $1 \leq j < i$. We now prove that $\succ = (=_B;(\to_{\mathcal{R}} \cup \rhd_B \cup \rightsquigarrow)^+;=_B)$ is a well-founded (strict) ordering. Transitivity is proved as in Proposition 3 thanks to the compatibility of $\to_{\mathcal{R}}$, $\rhd_B$ and $\rightsquigarrow$ with $=_B$. Irreflexivity is a consequence of well-foundedness, which we prove by contradiction: from the assumption that $\succ$ is not well-founded we are going to build an infinite well-formed proof tree which contradicts operational termination.

If $\succ$ is not well-founded, then, there is an infinite $\succ$-decreasing sequence A where, since $=_B;(\to_{\mathcal{R}} \cup \rhd_B)^+;=_B$ is well-founded, we must have an infinite number of $\rightsquigarrow$-steps which isolate finite $(\to_{\mathcal{R}} \cup \rhd_B)$-sequences. We can, then, write A as follows:

$$u_1\,(\to_{\mathcal{R}} \cup \rhd_B)^* u_1' \rightsquigarrow u_2\,(\to_{\mathcal{R}} \cup \rhd_B)^* u_2' \rightsquigarrow u_3 \cdots$$

In fact, since $(\to_{\mathcal{R}} \cup \rhd_B)^* = (\rhd_B^*;\to_{\mathcal{R}})^*;\rhd_B^*$ and the relation $\rightsquigarrow$ already takes into account B-subterms, we can eventually mix the last subterm steps of each sequence $u_i\,(\to_{\mathcal{R}} \cup \rhd)^* u_i'$ with the $\rightsquigarrow$-step on u_i'. Then, we can write each sequence from u_i to u_{i+1} as

$$\begin{array}{lcl} u_i & = & v_{i1} \\ & \rhd_B^* & v_{i1}' \\ & \to_{\mathcal{R}} & v_{i2} \\ & \vdots & \\ & (\rhd_B^*;\to_{\mathcal{R}}) & v_{in_i+1} \\ & \rightsquigarrow & u_{i+1} \end{array}$$

for some $n_i \geq 0$, where $v_{ij} = C_{ij}[v_{ij}']_{q_{ij}}$ for $i \geq 1$ and $1 \leq j \leq n_i$ and $u_i' = v_{in_i+1}$. Note that, if $n_i = 0$, then $u_i' = u_i$. Thus, by monotonicity of rewriting we can

also write:

$$\begin{array}{rcl} u_i & = & C_{i1}[v'_{i1}]_{p_{i1}} \\ & \to_{\mathcal{R}} & C_{i1}[v_{i2}]_{p_{i1}} \\ & \to_{\mathcal{R}} & \\ & \vdots & \\ & \to_{\mathcal{R}} & C_{i1}[\cdots C_{in_i}[v_{in_i+1}]_{p_{in_i}} \cdots]_{p_{i1}} \\ & \rightsquigarrow & C_{i1}[\cdots C_{in_i}[u_{i+1}]_{p_{in_i}} \cdots]_{p_{i1}} \end{array}$$

With regard to this sequence, we note the following.

1. Each step $C_{ij}[v'_{ij}]_{p_{ij}} \to_{\mathcal{R}} C_{ij}[v_{ij+1}]_{p_{ij}}$ has a finite proof tree T_{ij} whose root is $C_{ij}[v'_{ij}]_{p_{ij}} \to C_{ij}[v_{ij+1}]_{p_{ij}}$ and after $|p_{ij}|$ applications of the Congruence rule there is a proof of $v'_{ij} \to v_{ij+1}$:

$$\dfrac{\dfrac{\dfrac{U_{ij}}{v'_{ij} \to v_{ij+1}}}{\vdots}}{C_{ij}[v'_{ij}]_{p_{ij}} \to C_{ij}[v_{ij+1}]_{p_{ij}}}$$

   for some closed proof tree U_{ij}.
2. Since

$$w_i = C_{i1}[\cdots C_{in_i}[v_{in_i+1}]_{p_{in_i}} \cdots]_{p_{i1}} \rightsquigarrow C_{i1}[\cdots C_{in_i}[u_{i+1}]_{p_{in_i}} \cdots]_{p_{i1}},$$

   there is a rule $l_i \to r_i \Leftarrow u_i^1 \to v_i^1, \ldots, u_i^{k_i} \to v_i^{k_i}$, a position $q_i \geq p_{i1}.\cdots.p_{in_i}$, a substitution σ_i, and an index $\ell_i \in \{1, \ldots, k_i\}$ such that $w_i|_{q_i} =_B \sigma_i(l_i)$, $u_{i+1} = \sigma_i(u_i^{\ell_i})$ and $\sigma_i(u_i^j) \to_{\mathcal{R}}^{\star} \sigma_i(v_i^j)$ for all $1 \leq j < \ell_i$. Thus, there is a proof tree

$$\dfrac{T_i^1, \ldots, T_i^{\ell_i - 1}, T_i^{\ell_i}, G_i^{\ell_i+1}, \ldots, G_i^{m_i}}{G_i}$$

   with root $G_i = w_i[\sigma_i(l_i)]_{q_i} \to w_i[\sigma_i(r_i)]_{q_i}$, well-formed trees T_i^j rooted by $\sigma_i(u_i^j) \to_{\mathcal{R}}^{\star} \sigma_i(v_i^j)$ for $1 \leq j < \ell_i$, an *infinite* tree $T_i^{\ell_i}$ with $\mathit{left}(\mathit{root}(T_i^{\ell_i})) = u_{i+1} = \sigma_i(u_i^{\ell_i})$ which continues the tail of A after u_{i+1} as explained before, and open goals $G_i^{\ell_i+1}, \ldots, G_i^{m_i}$.

Thus, we obtain the infinite tree whose structure is shown in Figure 3. Therefore, assuming that $\succ$ is not well-founded contradicts operational termination of $\mathcal{R}$. Also, since well-foundedness implies irreflexivity of $\succ$, it turns out that $\succ$ is a well-founded strict ordering on terms. Now, it is not difficult to see that $\succ$ makes $\mathcal{R}$ quasi-decreasing. In particular (1) in Definition 12 holds because $\to_{\mathcal{R}} \subseteq \to_{\mathcal{R}}^{+} \subseteq (\to_{\mathcal{R}} \cup \rhd_B \cup \rightsquigarrow)^+$. And due to reflexivity of $=_B$, $(\to_{\mathcal{R}} \cup \rhd_B \cup \rightsquigarrow)^+ \subseteq \succ$ (and similarly for item (3) in the definition). With regard to item (2), if $s =_B t \succ u$, then there are terms t' and u' such that $t =_B t'(\to_{\mathcal{R}} \cup \rhd_B \cup \rightsquigarrow)^+ u' =_B u$. We prove by induction on the length n of the sequence $t'(\to_{\mathcal{R}} \cup \rhd_B \cup \rightsquigarrow)^+ u'$ that $s \succ u$.

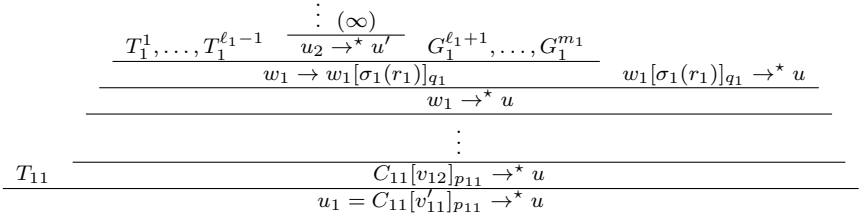


Figure 3: Proof of Theorem 7

1. Base case. If $n = 1$, then we have three possibilities:
   (a) $t' \to_{\mathcal{R}} u'$. Since $s =_B t =_B t'$, by strict coherence there are terms s', s'' such that $s =_B s'$, $s' \to_{\mathcal{R}} s''$ and $s'' =_B u' =_B u$, i.e., $s \succ u$.
   (b) $t' \rhd_B u'$, then, since $s =_B t =_B t'$, we have $s =_B t' \rhd_B u' =_B u$, i.e., $s \succ u$.
   (c) $t' \rightsquigarrow u$, then since $s =_B t =_B t'$, we have $s =_B t' \rightsquigarrow u' = u$, i.e., $s \succ u$.
2. Induction step. We have $t'(\to_{\mathcal{R}} \cup \rhd_B \cup \rightsquigarrow)t'' =_B u''(\to_{\mathcal{R}} \cup \rhd_B \cup \rightsquigarrow)^{n-1}u'$. By the induction hypothesis, $t'' \succ u$. Reasoning as in the base case, and by transitivity we conclude $s \succ u$.

Finally, item (4) follows by definition of $\rightsquigarrow$. □

Thus, quasi-decreasingness characterizes operational termination of order-sorted, sort-decreasing rewrite theories whose B-subterm relation is well-founded.

Corollary 5. *A deterministic 3-OSRT $\mathcal{R} = (\Sigma, B, R)$ such that $\rhd_B$ is well-founded is operationally terminating if and only if it is quasi-decreasing.*

7. Conclusions and Future Work

The results presented in this paper can be viewed from two complementary perspectives: one more theoretical, and another more practical. At the theoretical level, we have investigated parts of the *terra incognita* of conditional term rewriting by asking and providing precise answers to innocent-sounding questions such as: what is a normal form? How can normal forms be effectively computed? How should the notion of weakly normalizing system be understood in the conditional case? How can good executability properties be ensured for a theory? There is, however, a more practical aspect to all these results. It consists in taking to heart the idea that rewrite theories are an excellent framework for declarative programming and formal specification and verification. From this second perspective, the questions asked and answered include: what is the most general notion possible of a conditional rule-based program for which normal forms can be computed? What is the appropriate operational semantics for term normalization? How can it be made more efficient? What are the most general possible requirements under which conditional functional programs can define

total computable functions and be given an initial algebra semantics which *fully agrees* with their operational semantics?

Future work should further investigate proof methods and supporting tools for all the properties discussed here. For example, although the characterization of the operational termination of an OSRT in terms of quasi-decreasingness offers in principle a complete proof method, we are currently investigating a far-reaching generalization to the conditional case of the dependency pair method that seems considerably more effective for mechanizing actual proofs. In general, the development of *intrinsic methods* for proving both strong and weak operational termination of OSRTs seems both quite attractive and sorely needed.

With regard to proving OSRTs normal, Example 13 shows that the notion of operational termination of OSRTs is too strong to capture normality of some OSRTs. The development of techniques for specifically proving operational 1-termination is an important topic for further research. However, there are normal OSRTs that are *not* operationally 1-terminating (see Examples 3 and 4, and also Remark 7). Actually, in order to guarantee that an OSRT $\mathcal{R}$ is normal, operational 1-termination is required for irreducible terms *only* (see Definition 7). This suggests that research on proving this *localized* property (in the sense of [15]) can lead to more accurate methods than the ones discussed above, which try to use *global* termination properties of OSRTs to prove them normal. These are interesting subjects for future work.

Acknowledgements. We thank the anonymous referees for their constructive criticism and helpful comments. This work has been partially supported by NSF grant CNS 13-19109. Salvador Lucas' research was developed during a sabbatical year at UIUC and was also supported by the EU (FEDER), Spanish MINECO projects TIN2010-21062-C02-02 and TIN 2013-45732-C4-1-P, and GV grant BEST/2014/026 and project PROMETEO/2011/052.

References

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

[2] J. A. Bergstra and J. W. Klop. Conditional rewrite rules: Confluence and termination. *J. Comput. Syst. Sci.*, 32(3):323–362, 1986.

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

[4] F. Durán, S. Lucas, C. Marché, J. Meseguer, and X. Urbain. Proving operational termination of membership equational programs. *Higher-Order and Symbolic Computation*, 21(1-2):59–88, 2008.

[5] F. Durán, S. Lucas, and J. Meseguer. Termination modulo combinations of equational theories. In *Frontiers of Combining Systems, 7th International*

Symposium, FroCoS 2009, Trento, Italy, September 16-18, 2009. Proceedings, volume 5749 of *Lecture Notes in Computer Science*, pages 246–262. Springer, 2009.

[6] K. Futatsugi and R. Diaconescu. *CafeOBJ Report*. World Scientific, AMAST Series, 1998.

[7] J. Goguen and J. Meseguer. Order-sorted algebra I: Equational deduction for multiple inheritance, overloading, exceptions and partial operations. *Theoretical Computer Science*, 105:217–273, 1992.

[8] J. Goguen, T. Winkler, J. Meseguer, K. Futatsugi, and J.-P. Jouannaud. Introducing OBJ. In *Software Engineering with OBJ: Algebraic Specification in Action*, pages 3–167. Kluwer, 2000.

[9] J. Hendrix and J. Meseguer. On the completeness of context-sensitive order-sorted specifications. In *RTA*, volume 4533 of *Lecture Notes in Computer Science*, pages 229–245, 2007.

[10] J.-P. Jouannaud and HélèneKirchner. Completion of a set of rules modulo a set of equations. *SIAM Journal of Computing*, 15:1155–1194, November 1986.

[11] S. Kaplan. Conditional rewrite rules. *Theor. Comput. Sci.*, 33:175–193, 1984.

[12] S. Lucas. Context-sensitive computations in functional and functional logic programs. *J. Functl. and Log. Progr.*, 1(4):446–453, 1998.

[13] S. Lucas, C. Marché, and J. Meseguer. Operational termination of conditional term rewriting systems. *Information Processing Letters*, 95(4):446–453, 2005.

[14] S. Lucas and J. Meseguer. Strong and weak operational termination of order-sorted rewrite theories. In S. Escobar, editor, *Rewriting Logic and Its Applications - 10th International Workshop, WRLA 2014, Held as a Satellite Event of ETAPS, Grenoble, France, April 5-6, 2014, Revised Selected Papers*, volume 8663 of *Lecture Notes in Computer Science*, Springer, 2014, pages 178–194.

[15] S. Lucas and J. Meseguer. Localized operational termination in general logics. In R. D. Nicola and R. Hennicker, editors, *Software, Services, and Systems - Essays Dedicated to Martin Wirsing on the Occasion of His Retirement from the Chair of Programming and Software Engineering*, volume 8950 of *Lecture Notes in Computer Science*, Springer, 2015, pages 91–114.

[16] J. Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 96(1):73–155, 1992.

[17] J. Meseguer. Membership algebra as a logical framework for equational specification. In F. Parisi-Presicce, editor, *Proc. WADT'97*, pages 18–61. Springer LNCS 1376, 1998.

[18] J. Meseguer. Strict coherence of conditional rewriting modulo axioms. Technical Report http://hdl.handle.net/2142/50288, C.S. Department, University of Illinois at Urbana-Champaign, August 2014.

[19] J. Meseguer. Order-sorted rewriting and congruence closure. Technical report, CS Dept. University of Illinois at Urbana-Champaign, 2015. Available at: `http://hdl.handle.net/2142/78008`.

[20] E. Ohlebusch. *Advanced Topics in Term Rewriting*. Springer Verlag, 2002.

[21] C. Rocha, J. Meseguer, and C. A. Muñoz. Rewriting modulo SMT and open system analysis. In [**?**], pages 247–262, 2014.

[22] TeReSe. *Term Rewriting Systems*. Cambridge University Press, 2003.

[23] A. van Deursen, J. Heering, and P. Klint. *Language Prototyping: An Algebraic Specification Approach*. World Scientific, 1996.

[24] T. Yamada, J. Avenhaus, C. Loría-Sáenz, and A. Middeldorp. Logicality of conditional rewrite systems. *Theor. Comput. Sci.*, 236(1-2):209–232, 2000.