

The Π_2^0 -Completeness of Most of the Properties of Rewriting Systems You Care About (and Productivity)

Jakob Grue Simonsen

Department of Computer Science, University of Copenhagen
Universitetsparken 1, DK-2100 Copenhagen Ø, Denmark
simonsen@diku.dk

Abstract. Most of the standard pleasant properties of term rewriting systems are undecidable; to wit: local confluence, confluence, normalization, termination, and completeness.

Mere undecidability is insufficient to rule out a number of possibly useful properties: For instance, if the set of normalizing term rewriting systems were recursively enumerable, there would be a program yielding “yes” in finite time if applied to any normalizing term rewriting system.

The contribution of this paper is to show (the uniform version of) each member of the list of properties above (as well as the property of being a *productive* specification of a stream) complete for the class Π_2^0 . Thus, there is neither a program that can enumerate the set of rewriting systems enjoying any one of the properties, nor is there a program enumerating the set of systems that do not.

For normalization and termination we show both the ordinary version and the ground versions (where rules may contain variables, but only ground terms may be rewritten) Π_2^0 -complete. For local confluence, confluence and completeness, we show the ground versions Π_2^0 -complete.

1 (Uniform) undecidability in term rewriting

It is well-known that almost all of the usual nice-to-have properties of term rewriting systems are undecidable [11, 2, 18], in particular normalization, termination, local confluence, confluence, and completeness are undecidable, even when rewrite steps are allowed only on ground terms.

Upon inspection, many of the proofs used for undecidability employ reduction from well-known undecidable problems that are semidecidable—technically, the problems are “only” Σ_1^0 -complete—the canonical examples being (1) the Halting Problem: the set of integers m encoding pairs $m = \langle M, n \rangle$ such that Turing machine M halts on input n , and (2) the Post Correspondence Problem (PCP) [18, Ch.5].

A standard fact is that Σ_1^0 contains exactly the recursively enumerable sets; hence, if some property is proven undecidable via a reduction from, say, PCP, it could still be possible that the set of objects enjoying the property could be

recursively enumerable, hence semi-decidable. In rewriting, two simple examples of such a property are (1) for each term rewriting system (TRS) R , the set of terms that have an R -normal form, and (2) the set of TRSs R for which there exists at least one term having a normal form.

However, it would go against all common sense that, say, the set of all (finite) TRSs that are terminating should be semi-decidable: We know that TRSs may simulate Turing machines, and validating whether all terms of some specific TRS are terminating intuitively must consist of checking an infinite number of terms for each TRS, thus requiring an infinite amount of work to validate termination.

The purpose of this paper is to show that several of the usual nice-to-have properties of TRSs are Π_2^0 -complete. As neither any semi-decidable set, nor a co-semi-decidable set can be Π_2^0 -hard, we thus corroborate the above intuition.

We consider so-called *uniform* decision problems: All problems consist of deciding whether a given TRS has some property. This is in contrast to the non-uniform or *local* problems where, given a pair $\langle R, s \rangle_R$ consisting of a TRS R and a term s of R , the problem consists of deciding whether s satisfies some property under R -reduction.

We have endeavoured to show Π_2^0 -completeness of the properties when only *ground* terms may be rewritten. This is not to be confused with decision problems for *ground* TRS where, in addition, all *rules* are ground—in that case, many of the properties are decidable [9, 12, 4]. Furthermore, for the properties of local confluence and confluence, we have succeeded in proving Π_2^0 -completeness *only* in the case of ground terms.

Our results depend crucially on the finiteness of all systems and alphabets considered; allowing for systems with, say, an infinite number of rules would propel hardness into the low tiers of the analytical hierarchy instead.

2 Preliminaries

Term rewriting systems (TRSs) (Σ, R) are defined in the usual way [11, 18], as is the set of terms. In this paper, Σ and R will always be finite. We usually suppress the alphabet Σ and refer to the TRS merely by the set of rules R .

Definition 1. A constructor TRS is a TRS (Σ, R) where $\Sigma = F \cup C$ with $F \cap C = \emptyset$ (F is called the set of defined symbols, C the set of constructor symbols), and such that every rewrite rule $l \rightarrow r \in R$ satisfies $l = f(t_1, \dots, t_n)$ with $f \in F$ and t_1, \dots, t_n are terms over the signature C (that is, contain no defined symbols). When the alphabet Σ is presupposed, we usually refer to the TRS (Σ, R) merely by R . The TRS $(R)_0$ consists of the rules of R , but only ground terms may be rewritten. If s is a term, the set of positions of s and notion of subterm at a position are defined as usual. We denote the subterm at position p of s by $s|_p$, and denote the term obtained by replacing the subterm $s|_p$ in s by term t by $s[t]_p$. The depth of a rewrite step is the length of the position of the redex contracted. If $n \geq 0$, we write $s|_{\leq n} = t|_{\leq n}$ if $s|_{\leq n}$ and $t|_{\leq n}$ are identical up to and including all positions of length n .

Definition 2. A TRS R is normalizing (WN) if every term has a normal form, terminating (SN) if all maximal reduction sequences from all terms are finite, locally confluent (WCR) if, for every fork $t \leftarrow s \rightarrow t'$ there exists a join $t \rightarrow^* s'^* \leftarrow t'$, confluent (CR) if, for every fork $t^* \leftarrow s \rightarrow^* t'$ there exists a join $t \rightarrow^* s'^* \leftarrow t'$, and complete (COM) if R is both terminating and confluent.

We say that R is ground-WN (resp. -SN, -WCR, -CR, -COM) if R is WN on ground terms, that is, if $(R)_0$ is WN (resp. SN, WCR, CR, COM).

We presuppose basic knowledge of primitive recursive functions and primitive recursive predicates, see e.g. [13]. The reader without prior knowledge may simply view primitive recursive functions as those functions that are computable using for-loops, but without unconstrained recursion or iteration.

It is easy to see that there are primitive recursive functions en- and decoding each (finite) TRS as an integer, en- and decoding pairs of integers, en- and decoding finite reduction sequences in a given TRS as an integer, and checking whether a given sequence of rewrite steps are allowed by the TRS. For ease of notation, we overload the symbols $\langle \cdot \rangle$ to mean any primitive recursive encoding and $\langle x \rangle_R$ to mean a primitive recursive encoding of x in TRS R , as appropriate by context. Thus, $\langle s_0 \rightarrow s_1 \rightarrow \dots \rightarrow s_m \rangle_R$ is the integer encoding the reduction $s_0 \rightarrow s_1 \rightarrow \dots \rightarrow s_m$ in TRS R , and $\langle R \rangle$ is an integer encoding the TRS R .

2.1 Turing machines

We refer to standard textbooks [13, 16] for a standard treatment of Turing machines. We introduce basic notation:

Definition 3. A (deterministic, single-tape, binary alphabet) Turing machine M is a triple $(Q, \{1, 0, \square\}, \delta)$ where

- Q is a finite set of states containing at least two distinct states q_s and q_h .
- $\{1, 0, \square\}$ is the set of tape symbols where \square is interpreted as “blank”.
- δ is a partial function from $Q \times \{1, 0, \square\}$ to $Q \times \{1, 0, \square\} \times \{L, R\}$ and is called the transition relation. L represents a move to the left and R a move to the right.

We assume that for every $q \neq q_h$ and every $s \in \{1, 0, \square\}$, $\delta(q, s)$ is defined (if $\delta(q, s)$ is undefined, the machine has halted, and we may wlog. add $\delta(q, s) \rightarrow (q_h, \square, L)$ without affecting the halting behaviour of the machine). Furthermore, we assume wlog. that the machine never returns to its starting state, i.e. that if $\delta(q, s) = (q', s', D)$, then $q' \neq q_s$.

Definition 4. A configuration of a Turing machine M is a triple wqw' where q is a state of M , w and w' are the tape contents to the left and right of the tape head (we assume the head is on the first cell of w'), and we assume that w and w' contain only a finite number of non-blank symbols.

A start configuration of M is a configuration wqw' where $q = q_s$, w contains only blanks, and w' consists of a finite number of consecutive 1s (that is, the Turing machine is in the start state, there is a unary representation of a natural number to the right of the tape head, and nothing else on the tape).

2.2 The arithmetical hierarchy and Π_2^0

Recall that the *arithmetical* (or *Kleene*) hierarchy is a classification of formulae of first-order (Peano) arithmetic; in particular, a formula ϕ is a Π_2^0 -formula (or $\forall\exists$ -formula) if it is logically equivalent to a formula on the form $\forall n\exists k.P(n, k, x)$ where $P(n, k, x)$ is a primitive recursive predicate.

Definition 5. *The class Π_2^0 comprises of all subsets K of the natural numbers such that there is a Π_2^0 -formula that is valid exactly on the elements of K .*

Details on Π_2^0 and other classes of the arithmetical hierarchy may be found in [13]. Note that replacing the demand that $P(n, k, x)$ be primitive recursive by “ $P(n, k, x)$ is a decidable predicate” does not change the class Π_2^0 . As with complexity classes such as P and NP , we may define the relation \leq_m (many-one reducibility) on Π_2^0 :

Definition 6. *The set A many-one reduces to set B , written $A \leq_m B$, if there exists a Turing machine M that on input $x \in \mathbb{N}$ outputs $\phi_M(x)$ such that $\phi_M(x) \in B$ iff $x \in A$. We say that B is Π_2^0 -hard if every $A \in \Pi_2^0$ satisfies $A \leq_m B$, and that B is Π_2^0 -complete if $B \in \Pi_2^0$ and B is Π_2^0 -hard.*

Intuitively, a Π_2^0 -hard set is *at least as difficult* to decide as any other set in Π_2^0 . Ordinarily, the notion of *Turing reducibility* \leq_T is used in connection with Π_2^0 ; it is easy to see that $A \leq_m B$ implies $A \leq_T B$, whence our hardness results also holds in the setting of \leq_T .

Definition 7. *UNIFORM is the set of (Gödel numbers of) Turing machines that reach the halting state from all configurations.*

TOTALITY is the set of (Gödel numbers of) Turing machines that reach the halting state from all start configurations.

Thus, membership in UNIFORM requires that M halts on any configuration (even ones unreachable from the start state), while membership in TOTALITY requires that M halts when run on any natural number.

Proposition 1. *Both UNIFORM and TOTALITY are Π_2^0 -complete.*

Proof. Standard. See e.g. [13, Ch. 13–14] for TOTALITY, and [8] for UNIFORM. \square

By standard results for the arithmetical hierarchy, no Π_2^0 -hard set is semi-decidable, nor is it co-semi-decidable.

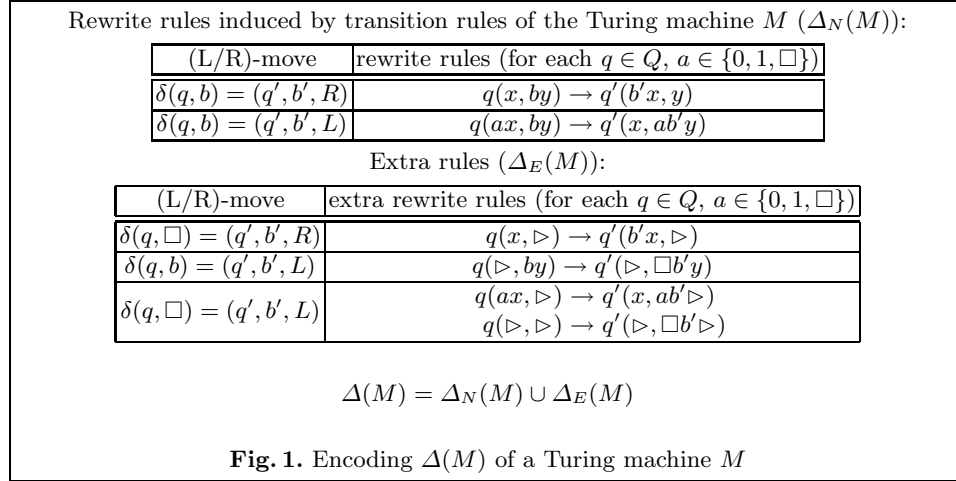
3 Encoding Turing machines

The paper uses three encodings of Turing machines M . Of these encodings, the first— $\Delta(M)$ —is the standard encoding of M as an orthogonal TRS from [18, Ch. 5], itself a clever variation of the idea in [9]. The second encoding— $\Delta_g(M)$ —adds a number function symbols and rules to $\Delta(M)$ to handle the ground-WCR

and ground-CR properties and is, by construction, non-orthogonal. The third encoding— $\Delta_S(M)$ —extends $\Delta(M)$ by letting certain function symbols take an extra argument (which is unchanged by the rewrite rules) and adding two new simple rules, obtaining an orthogonal constructor TRS.

We now give the encoding $\Delta(M)$ of [18, Ch. 5]. A Turing machine M is encoded as a TRS $\Delta(M)$ as follows: Each state $q \in Q$ of the machine M is coded as a binary function symbol q . The alphabet of $\Delta(M)$ furthermore contains unary function symbols 1 , 0 and \square corresponding to the elements of the tape alphabet of M , and a nullary function symbol \triangleright representing an infinite stretch of blanks (\triangleright may thus be thought of as an “endmarker” demarcating the finite part of the tape containing non-blank symbols). A word $w \in \{0, 1, \square\}^*$ is translated into a term $\phi(w)$ by setting $\phi(\epsilon) \triangleq \triangleright$ (where ϵ is the empty word), and $\phi(bw) \triangleq b(\phi(w))$ for $b \in \{1, 0, \square\}$ and $w \in \{1, 0, \square\}^*$. When b_1, \dots, b_n are unary function symbols and t is a term, we write $b_1 b_2 \dots b_n t$ instead of $b_1(b_2(\dots(b_n(t))))$. A configuration $w_1 q w_2$ of M is translated to the $\Delta(M)$ -term $q(\phi(\text{reverse}(w_1)), \phi(w_2))$ where $\text{reverse}(w_1)$ is w_1 reversed. For ease of notation, we suppress ϕ and reverse and write $q(w_1, w_2)$.

The rules of the encoding is given in Figure 1.



It is straightforward to show that for all Turing-machine configurations α, β of M , we have (1) if M moves from α to β in one move and term s represents α , then there is a term t with $s \rightarrow t$ and such that t represents β , respectively (2) if term s represents α and $s \rightarrow t$, then there is a configuration β such that M moves from α to β in one step and t represents β [18, Exercise 5.3.3].

For later use, we note the following fortuitous property of $\Delta(M)$:

Proposition 2. *Call any ground term of $\Delta(M)$ restricted if it is on the form $q(s, s')$ where s and s' contain no occurrence of any $q \in Q$. Then any restricted*

ground term represents a configuration of M , and if $\Delta(M)$ admits an infinite reduction starting from some term, there is a restricted ground term t having an infinite reduction

Proof. Straightforward. See e.g. [18, Exercise 5.3.3] or [11, Exercise 2.2.12]. \square

Corollary 1. $\Delta(M)$ is both SN and ground-SN iff M halts on all configurations.

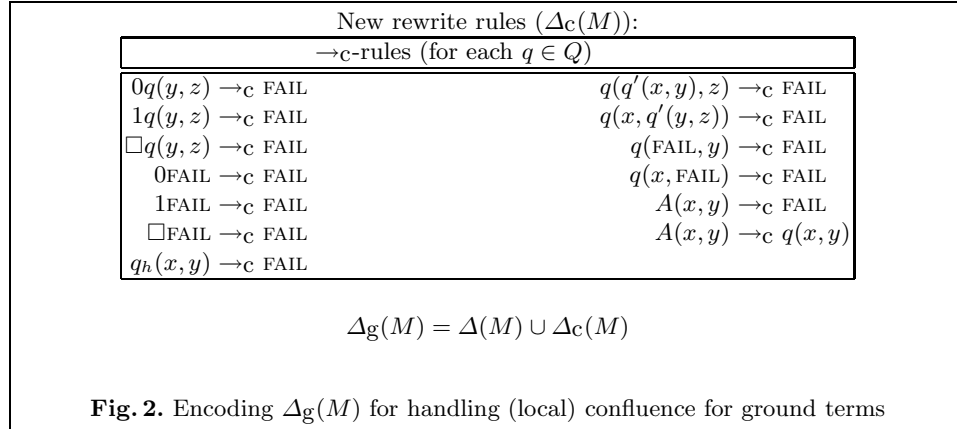
Proof. If there is a configuration wqw' on which M does not halt, then there is an infinite $\Delta(M)$ -reduction starting from the ground term $q(w, w')$, showing that $\Delta(M)$ is neither SN, nor ground-SN. If M halts on all configurations, suppose for the purpose of contradiction that $\Delta(M)$ were not SN. By Proposition 2 there is a restricted ground term t having an infinite reduction. But by the same proposition, any restricted ground term represents a configuration wqw' of M , entailing that M does not halt on wqw' , a contradiction. \square

Corollary 2. $\Delta(M)$ is both WN and ground-WN iff M halts on all configurations.

Proof. As the previous corollary. \square

3.1 Adding rules for ground-WCR and CR: the encoding $\Delta_g(M)$

For the purpose of proving ground-WCR and ground-CR Π_2^0 -hard, we shall need to extend the encoding $\Delta(M)$. We extend the alphabet of $\Delta(M)$ with two new function symbols: the nullary symbol FAIL and the binary symbol A , and we extend the rules of $\Delta(M)$ with a number of new rules shown in Figure 2. The rules are specifically designed for (1) making it possible to rewrite ground terms that do *not* represent a configuration of M to the constant FAIL, and for (2) creating a fork $\text{FAIL} \leftarrow A(w, w') \rightarrow q(w, w')$ that has no corresponding join unless M reaches the halting state from configuration wqw' .



Remark 1. Observe that if $q(w, w')$ is a term representing a configuration wqw' of M and $q \neq q_h$, then $q(w, w')$ contains no $\Delta_c(M)$ -redex (in particular, $q(w, w')$ does not contain any occurrence of FAIL). Furthermore, observe that if $q(s, t)$ is a restricted ground term containing no occurrence of FAIL or A , and if $q(s, t) \rightarrow^* q'(s', t')$, then $q'(s', t')$ contains no occurrence of FAIL or A : this is easily seen by the fact that A does not occur on any right-hand side of any rule and the fact that FAIL can only be introduced in one step by a $\Delta_c(M)$ -redex; as $q(s, t)$ is restricted ground, the only $\Delta_c(M)$ -redex possible in $q(s, t) \rightarrow^* q'(s', t')$ is in the final term, and only if $q' = q_h$.

Finally, note that if $q(s, t)$ is a restricted ground term containing no occurrence of FAIL or A , $\Delta_g(M)$ enjoys the same simulation properties as $\Delta(M)$. \square

Proposition 3. *If $s \rightarrow t$ by an application of a $\Delta_c(M)$ -redex at depth $d \geq 1$ in s , then t reduces to FAIL in at most d steps.*

Proof. Straightforward induction on d . \square

Proposition 4. *$\Delta_g(M)$ is ground-SN iff M halts on all configurations.*

Proof. Straightforward analysis appealing to the fact that $\Delta(M)$ is SN iff M halts on all configurations and the fact that $\Delta_c(M)$ can only create an $\Delta(M)$ -redex by the rules in $\{A(x, y) \rightarrow_c q(x, y) : q \in Q\}$. \square

Proposition 5. *The only ground normal form of $\Delta_g(M)$ is FAIL.*

Proof. Straightforward case analysis using the fact that any ground term that does not represent a configuration of M will either be FAIL or contain a $\Delta_c(M)$ -redex. \square

Proposition 6. *$\Delta_g(M)$ is ground-WCR iff M halts on all configurations.*

Proof. We proceed as follows.

- “ \Rightarrow ”: By contraposition. Suppose there is a configuration wqw' on which M does not halt. Then there is a restricted ground term $q(w, w')$ that represents wqw' (hence contains no occurrence of FAIL or A), and there is an infinite reduction $A(w, w') \rightarrow_c q(w, w') \rightarrow q'(s, s') \rightarrow q''(s'', s''') \rightarrow \dots$. By assumption, no M -step ever reaches the halting state, whence no reduct of $q(w, w')$ can contain an occurrence of q_h . Furthermore, by Remark 1, no reduct of $q(w, w')$ contains any occurrence of FAIL or A , whence no term in $q(w, w') \rightarrow q'(s, s') \rightarrow q''(s'', s''') \rightarrow \dots$ can reduce to FAIL. But $A(w, w') \rightarrow_c \text{FAIL}$ and $A(w, w') \rightarrow_c q(w, w')$, showing that $\Delta_g(M)$ is not ground-WCR (and not WCR).
- “ \Leftarrow ”: Suppose M halts on all configurations, hence that $\Delta_g(M)$ is ground-SN by Proposition 4. Thus, for any fork $t \leftarrow s \rightarrow t'$, both t and t' have normal forms t_f and t'_f . As s is ground, so are t and t' , and Proposition 5 yields $t_f = t'_f = \text{FAIL}$, whence $t \rightarrow^* \text{FAIL}^* \leftarrow t'$, as desired. \square

Corollary 3. $\Delta_g(M)$ is ground-CR iff M halts on all configurations.

Proof. If $\Delta_g(M)$ is ground-CR, it is in particular ground-WCR, hence by Proposition 6 M terminates on all inputs. Conversely, if M terminates on all inputs, Propositions 6 and 4 yield that $\Delta_g(M)$ is WCR and SN, respectively. Confluence of $\Delta_g(M)$ follows by Newman’s Lemma. \square

Corollary 4. $\Delta_g(M)$ is ground-COM iff M halts on all configurations.

Proof. By Proposition 4 and Corollary 3. \square

We postpone the final encoding of M , $\Delta_s(M)$, to Section 5.

4 Π_2^0 -Completeness of the standard properties

In the below, we consider ternary primitive recursive predicates $P(n, k, l)$ in Π_2^0 -formulae $\forall n \exists k. P(n, k, l)$ where l is thus the only free variable. As all the decision problems we consider take a TRS as “input”, the variable l will always be an integer encoding a TRS R . In all predicates $P(n, k, l)$, P decodes integers n and k to terms of R or finite R -reductions, and perform simple checks. For ease of notation, instead of referring explicitly to l , we will use the obvious shorthands $\langle \cdot \rangle_R$ and $\langle \cdot, \cdot \rangle_R$ in the predicates.

4.1 (Ground-)local confluence

The usual “easy” proofs of undecidability of (local) confluence employ reductions from the (local) word problem for semi-groups (which is in Σ_1^0 and hence is not Π_2^0 -complete), see e.g. [18]. More detailed analyses reveal that neither the uniform problem of whether a TRS is ground-WCR, nor the local problem of whether a term in a TRS is ground-WCR or ground-CR is semi-decidable [10]. However, even with these results, (ground-)WCR or (ground-)CR could still be co-semi-decidable. We will rule out this possibility by proving both problems Π_2^0 -complete using $\Delta_g(M)$. We do not know if our results can be extended to encompass WCR and CR: the encoding $\Delta_g(M)$ is not sufficient to do the trick as non-ground terms that do not represent configurations of M may be normal forms of $\Delta_g(M)$, rendering our proof methods unapplicable.

Proposition 7. WCR and ground-WCR are both Π_2^0 -properties of TRSs.

Proof. Using the primitive recursive en- and decoding of finite reduction sequences as integers, we may write the property of the TRS (Σ, R) being locally confluent as:

$$\forall n \exists k. \left(\begin{array}{l} n = \langle s \rightarrow s', s \rightarrow s'' \rangle_R \\ k = \langle s' \rightarrow s'_1 \rightarrow \dots \rightarrow s'_l, s'' \rightarrow s''_1 \rightarrow \dots \rightarrow s''_m \rangle_R \end{array} \Rightarrow \right)$$

\square

Theorem 1. *Deciding whether a TRS is ground-WCR is Π_2^0 -complete.*

Proof. There is clearly a recursive procedure transforming a description of a Turing machine M into the finite TRS $\Delta_g(M)$. By Proposition 6, this is a many-one reduction from UNIFORM to the set of TRSs that are ground-WCR, showing that deciding whether a given TRS is $(-)_0\text{wcr}$ is Π_2^0 -hard. Completeness now follows from Proposition 7. \square

4.2 (Ground-)confluence

In analogy to WCR, we have:

Proposition 8. *CR and ground-CR are both Π_2^0 -properties.*

Proof. As with local confluence we may write the property of TRS (Σ, R) being confluent as:

$$\forall n \exists k. \left(\begin{array}{l} n = \langle s \rightarrow s_1 \rightarrow \dots \rightarrow s_l, s \rightarrow s'_1 \rightarrow \dots \rightarrow s'_m \rangle_R \\ k = \langle s_l \rightarrow s_{l+1} \rightarrow \dots \rightarrow s_{l+r}, s'_m \rightarrow s'_{m+1} \rightarrow \dots \rightarrow s'_{m+p} \rangle_R \end{array} \Rightarrow \right)$$

\square

Theorem 2. *Deciding whether a TRS is ground-CR is Π_2^0 -complete.*

Proof. By Corollary 3 and Proposition 8, reasoning as in the proof of Theorem 1. \square

4.3 Normalization

Both WN and SN have previously been shown Π_2^0 -hard, even though the results were never explicitly stated—in fact, the encoding $\Delta(M)$ and its use in [18] to prove SN undecidable is in fact a reduction from UNIFORM to SN. We explicitly state the results below.

Proposition 9. *WN and ground-WN are both Π_2^0 -properties of TRSs.*

Proof. The predicate “term s is a normal form of (finite) TRS R ” is primitive recursive; we write $\text{nf}(s)_R$ for the predicate. Using primitive recursive en- and decoding of terms and finite reductions sequences, we may write the property of TRS (Σ, R) being normalizing as:

$$\forall n \exists k. (n = \langle s \rangle_R \Rightarrow (k = \langle s \rightarrow s_1 \rightarrow \dots \rightarrow s_m \rangle_R \wedge \text{nf}(s_m)_R))$$

\square

Theorem 3. *Deciding whether a TRS is WN (resp. ground-WN) is Π_2^0 -complete.*

Proof. By Corollary 2 and Proposition 9, reasoning as in the proof of Theorem 1. \square

4.4 Termination

Unlike the previous properties, a technical snag of proving Π_2^0 -completeness of termination is to show that the problem is *included* in Π_2^0 : Termination means that *for all* terms s , there are *no* infinite reduction sequences starting from s . Neither this phrasing, nor the phrasing using finiteness of maximal reductions, is amenable to putting in $\forall\exists$ -form. We work around this problem by appealing to König's Lemma.

Definition 8. *Let s be a term. The derivation tree of s is the (possibly infinite) rooted tree $\mathcal{G}(s)$ such that (i) level 0 in the tree contains the term s , and (ii) for $n \geq 0$, the nodes of level $n + 1$ are such that, for each node s at level n , there is a node t at level $n + 1$ and an edge from s to t iff $s \rightarrow t$.*

Nodes at level $n + 1$ are not shared, that is, a term t may occur as several different nodes if it is a reduct of several different terms at level n .

Proposition 10. *The term s is terminating iff $\mathcal{G}(s)$ is finite.*

Proof. $\mathcal{G}(s)$ is finitely branching as each term has a finite number of reducts. König's Lemma now yields that $\mathcal{G}(s)$ is infinite iff there exists an infinite path in $\mathcal{G}(s)$. By construction of $\mathcal{G}(s)$, s is terminating iff there exists an infinite path in $\mathcal{G}(s)$, concluding the proof. \square

Observe that any *finite* tree labeled with terms over alphabet Σ can be encoded and decoded as an integer by a primitive recursive function. If G is such a finite tree, we shall denote by $\langle G \rangle_R$ its encoding. Checking that $\langle G \rangle_R$ encodes the derivation tree $\mathcal{G}(s)$ of a term s is again primitive recursive as the first n levels of $\mathcal{G}(s)$ may be generated by a primitive recursive procedure: Exhaustively try all rules in the term rewriting system R on all positions of s to obtain level 1 in the graph and proceed iteratively until n levels have been processed. Given $\langle G \rangle_R$, obtain its maximum depth n by a straightforward search, then generate $\mathcal{G}(s)$ to depth n and call the result $\mathcal{G}(s)|_n$. We then have $G = \mathcal{G}(s)$ iff all leaves in $\mathcal{G}(s)|_n$ are normal forms and $G = \mathcal{G}(G)|_n$. Let the primitive recursive predicate that $G = \mathcal{G}(s)$ be $P(\langle s \rangle_R, \langle G \rangle_R)$ —note that $\langle R \rangle$ is a suppressed third variable of P .

Proposition 11. *SN and ground-SN are both Π_2^0 -properties.*

Proof. Using $\langle G \rangle_R$ we may write the property of TRS (Σ, R) being terminating as:

$$\forall n \exists k. (n = \langle s \rangle_R \wedge k = \langle G \rangle_R \wedge P(n, k))$$

For ground-SN, all that needs to be changed is to change the encoding $\langle \cdot \rangle_R$ from arbitrary terms to ground ones. \square

Theorem 4. *Deciding whether a TRS is SN (resp. ground-SN) is Π_2^0 -complete.*

Proof. By Corollary 1 and Proposition 11, reasoning as in the proof of Theorem 1. \square

4.5 Completeness

Theorem 5. *Deciding whether a TRS is ground-COM is Π_2^0 -complete.*

Proof. If A and B are both sets in Π_2^0 , then so is $A \cap B$. Thus, by Propositions 8 and 11, ground-COM is a Π_2^0 -property. Hardness of ground-COM follows from the fact that $\Delta_C(M)$ is ground-SN and ground-CR iff M halts on all configurations by Proposition 4 and Corollary 3. \square

5 Π_2^0 -completeness of productivity (of stream specifications)

A *stream specification* is intuitively a program that lazily outputs a list. If such a program, when run for a sufficient amount of time, always outputs “the next” element of the list, it is called *productive*. A substantial amount of work aims at establishing sufficient conditions for establishing the productivity of stream specifications [5, 15, 17, 3, 7, 6]. It is well-known folklore—and almost self-evident—that the problem of establishing whether a stream specification is productive is undecidable. Some of the recent literature on stream definitions employ a complicated but clean notion of stream specification stratified into data, function and stream layer and using many-sorted rewriting [7, 6]. Here, we consider the more generic approach where a specification is simply a constructor TRS with a designated symbol for the stream.

Definition 9. *A stream specification is a pair $((F \cup C, R), S)$ consisting of a constructor TRS $(F \cup C, R)$ and a designated nullary defined symbol $S \in F$. The stream specification is said to be productive if S has an infinite constructor normal form, that is, if there exists an infinite reduction $S \rightarrow^* t_1 \rightarrow^* t_2 \rightarrow^* \dots$ such that increasingly larger prefixes of the terms t_n consist solely of constructor symbols. The stream specification is said to be orthogonal if $(F \cup C, R)$ is orthogonal.*

We note that several “streams” may be defined by the same definition; however we are only interested in *one* of these, namely the one designated by S . Most literature on productivity focuses specifically on the case where the data modelled are truly streams of bits, that is, $C = C_S \triangleq \{:, \mathbf{0}, \mathbf{1}\}$. We note that the *hardness* proof below specifically uses only C_S —in fact, uses only $\{:, \mathbf{1}\}$ —thus showing that even a very restricted constructor alphabet corresponding to a classical “lazy list of bits” implies Π_2^0 -hardness. A full account of infinite (constructor) terms is beyond the scope of this paper, and we refer to [1, 18]. For our purposes, the reader need only consider the set of infinite (constructor) terms over C_S , for example $\mathbf{1}:\mathbf{1}:\mathbf{1}:\dots$. We invariably write the binary symbol $:$ as infix, and if s is a term, we use $\mathbf{1}:\mathbf{1}:\dots:\mathbf{1}:s$ as shorthand for $\mathbf{1}:(\mathbf{1}:(\dots\mathbf{1}(:s)))$.

We note the following fact about orthogonal constructor TRSs:

Proposition 12. *If $(F \cup C, R)$ is orthogonal, then $((F \cup C, R), S)$ is productive iff for all $n \geq 0$ there is a finite rewrite sequence $S \rightarrow^* t_n$ where t is a term on the form $C[s_1, \dots, s_m]$ with $C[x_1, \dots, x_m]$ a constructor term of depth n .*

Proof. Straightforward analysis using the fact that $(F \cup C, R)$ is orthogonal, hence is confluent, and that no constructor symbol is at the root of the left-hand side of a rewrite rule $l \rightarrow r \in R$. \square

Example 1. The assumption of orthogonality cannot be omitted from Proposition 12. Consider for instance the below (non-orthogonal) constructor TRS (constructor symbols are boldface):

$$\left\{ \begin{array}{l} S \rightarrow a(\mathbf{b}) \\ a(\mathbf{b}) \rightarrow a(a(\mathbf{b})) \\ a(\mathbf{b}) \rightarrow \mathbf{c}(\mathbf{b}) \\ a(\mathbf{c}(x)) \rightarrow \mathbf{c}(\mathbf{c}(x)) \end{array} \right\}$$

For every $n \geq 1$ we have $S \rightarrow^* \underbrace{\mathbf{c}(\mathbf{c}(\dots(\mathbf{c}(\mathbf{b}))))}_n$ which is a constructor normal form. However, S has no infinite constructor normal form. \square

Proposition 13. *Productivity of orthogonal stream specifications is a Π_2^0 -property.*

Proof. Observe that there is a primitive recursive function that, when input n and an integer j encoding some term s checks whether s contains only constructor symbols up to depth n (simply use a top-down iteration on s up to depth n). Let $P(n, j, l)$ be the primitive recursive predicate checking whether the above is true for orthogonal stream specification $l = \langle (F \cup C, R), S \rangle_R$.

Thus,

$$\forall n \exists k. (k = \langle S \rightarrow s_1 \rightarrow \dots \rightarrow s_m \rangle_R \wedge P(n, \langle s_m \rangle, l))$$

is a Π_2^0 -formula which, by Proposition 12 is true iff $((F \cup C, R), S)$ is productive. \square

We change the encoding of Turing machines M slightly to accommodate productivity; the new encoding is shown in Figure 3. Each state q is now encoded by a *ternary* function symbol such that in $q(s, t, t')$, s and t are the contents of the left and right part of the tapes as usual, and t' represents the “current” input being processed by M . We *change* the rules of Figure 1 by letting the third argument t' of each $q(s, t, t')$ be identical in left- and right-hand sides (that is, the argument is unchanged by applying any of the rules). We add two new rules: the rule $q_h(x, y, z) \rightarrow \mathbf{1}:q_s(\triangleright, 1z, 1z)$ that “produces” $\mathbf{1}$, and the rule $S \rightarrow q_s(\triangleright, 1\triangleright, 1\triangleright)$ that initiates the computation. Thus, the translation $\Delta_S(M)$ of M produces a stream specification $((F \cup C, R), S)$ where S is a some fresh symbol, $F = \{S\} \cup \{0, 1, \square, \triangleright\} \cup Q$ where Q contains a ternary symbol for each state of M , where $C = \{\mathbf{0}, \mathbf{1}, :\}$ (where the nullary constructor symbols $\mathbf{0}$ and $\mathbf{1}$ are not to be confused with the unary defined symbols 0 and 1), and R is the set of rules obtained in Figure 3.

Remark 2. Call a ground term of $\Delta_S(M)$ restricted if it is on the form $q(w, w', t)$ with $q \neq q_h$ and where w, w', t terms over F such that no q occurs in $w, w',$ or t .

Rewrite rules induced by transition rules of the Turing machine M ($\Delta'_N(M)$):

(L/R)-move	rewrite rule (for each $q \in Q, a \in \{0, 1, \square\}$)
$\delta(q, b) = (q', b', R)$	$q(x, sy, z) \rightarrow q'(b'x, y, z)$
$\delta(q, b) = (q', b', L)$	$q(ax, by, z) \rightarrow q'(x, ab'y, z)$

Extra rules ($\Delta'_E(M)$):

(L/R)-move	extra rewrite rules (for each $q \in Q, a \in \{0, 1, \square\}$)
$\delta(q, \square) = (q', b', R)$	$q(x, \triangleright, z) \rightarrow q'(b'x, \triangleright, z)$
$\delta(q, b) = (q', b', L)$	$q(\triangleright, by, z) \rightarrow q'(\triangleright, \square b'y, z)$
$\delta(q, \square) = (q', b', L)$	$q(ax, \triangleright, z) \rightarrow q'(x, ab'\triangleright, z)$ $q(\triangleright, \triangleright, z) \rightarrow q'(\triangleright, \square b'\triangleright, z)$

New rules for transitions from the halting state ($\Delta'_p(M)$):

new rules for stream productivity
$q_h(x, y, z) \rightarrow \mathbf{1}:q_s(\triangleright, 1z, 1z)$
$S \rightarrow q_s(\triangleright, 1\triangleright, 1\triangleright)$

$$\Delta_S(M) = \Delta'_N(M) \cup \Delta'_E(M) \cup \Delta'_p(M)$$

Fig. 3. Encoding $\Delta_S(M)$ of Turing machines for stream productivity

As $\Delta_S(M)$ is constructed exactly as $\Delta(M)$, bar $\Delta'_p(M)$ and the increased arity of the state symbols q , we immediately see that if α and β are configurations of M such that M moves from α to β and $q(w, w', t)$ represents α , then there is a term s' with $s \rightarrow s'$, and if $q(w, w, t)$ represents configuration α and $s \rightarrow s'$, then s' represents a configuration β such that M moves from α to β . \square

Proposition 14. $(S, \Delta_S(M))$ is productive iff M halts on all inputs.

Proof. Reason as follows:

- “ \Rightarrow ”: The only possible reduction from S is $S \rightarrow q_s(\triangleright, 1\triangleright, 1\triangleright)$. By inspection of the rules, the only way of replacing a function symbol by a constructor is by application of the rule $q_h(x, y, z) \rightarrow \mathbf{1}:q_s(\triangleright, 1z, 1z)$. Thus, the only possibly infinite constructor normal form of S is $\mathbf{1}:\mathbf{1}:\mathbf{1}:\dots$. We show by induction on $n \geq 1$ that:

$$\begin{aligned}
S &\rightarrow q_s(\triangleright, 1\triangleright, 1\triangleright) \\
&\rightarrow^* q_h(s_1, t_1, 1\triangleright) \\
&\rightarrow \mathbf{1} : q_s(\triangleright, 11\triangleright, 11\triangleright) \\
&\rightarrow^* \underbrace{\mathbf{1} \cdots \mathbf{1}}_{n-1} : q_h(s_2, t_2, 1^n\triangleright) \\
&\rightarrow^* \underbrace{\mathbf{1} \cdots \mathbf{1}}_n : q_s(\triangleright, 1^{n+1}\triangleright, 1^{n+1}\triangleright)
\end{aligned}$$

The base case $n = 1$ follows immediately by inspection of the rules of $\Delta_S(M)$: The only way to obtain a term on the form $\mathbf{1}:s$ is by an application of the rule

$q_h(x, y, z) \rightarrow \mathbf{1}:q_s(\triangleright, 1z, 1z)$. As $q_s(\triangleright, 1\triangleright, 1\triangleright)$ is a restricted ground term, every rewrite step of $\Delta_S(M)$ that is *not* of rule $q_h(x, y, z) \rightarrow \mathbf{1}:q_s(\triangleright, 1z, 1z)$ or $S \rightarrow q_s(\triangleright, 1\triangleright, 1\triangleright)$ simulates a step of M . Hence, there is a reduction $S \rightarrow q_s(\triangleright, 1\triangleright, 1\triangleright) \rightarrow^* q_h(s_1, t_1, 1\triangleright) \rightarrow \mathbf{1} : q_s(\triangleright, 11\triangleright, 11\triangleright)$, as desired. For the case $n > 1$, the induction hypothesis yields that:

$$S \rightarrow^* \underbrace{\mathbf{1}:\cdots:\mathbf{1}}_{n-1}:q_s(\triangleright, 1^n\triangleright, 1^n\triangleright)$$

By Proposition 12, productivity of $\Delta_S(M)$ yields that $S \rightarrow^* \underbrace{\mathbf{1}:\cdots:\mathbf{1}}_n:t$ for some term t , and orthogonality (hence confluence) of $\Delta_S(M)$ yields that $\mathbf{1}:\cdots:\mathbf{1}:q_s(\triangleright, 1^n, 1^n)$ and $\mathbf{1}:\cdots:\mathbf{1}:t$ have a common reduct which, as $\Delta_S(M)$ is a constructor TRS must be on the form $\underbrace{\mathbf{1}:\cdots:\mathbf{1}}_n:t'$ for some term t' . Thus,

we have $q_s(\triangleright, 1^n, 1^n) \rightarrow \mathbf{1}:t'$, and reasoning as in the base case, we obtain $q_s(\triangleright, 1^n, 1^n) \rightarrow^* q_h(s', s'', 1^{n+1}\triangleright) \rightarrow \mathbf{1}:q_s(\triangleright, 1^{n+1}\triangleright, 1^{n+1}\triangleright)$.

Thus, for all n , we have $q_s(\triangleright, 1^n\triangleright, 1^n\triangleright) \rightarrow^* \mathbf{1}:q_h(s'_n, s''_n, 1^{n+1}\triangleright)$ by a reduction that uses no rules from $\Delta'_p(M)$. As each step in the reduction simulates a step of M , we conclude that M halts on all inputs.

- “ \Leftarrow ”: If M halts on all inputs, we have in particular for all $n \geq 1$ that $q_s(\triangleright, 1^n\triangleright, 1^n\triangleright) \rightarrow^* q_h(s_n, t_n, 1^n\triangleright)$ for terms s_n, t_n , and hence that:

$$\begin{aligned} \underbrace{\mathbf{1}:\cdots:\mathbf{1}}_{n-1}:q_s(\triangleright, 1^n\triangleright, 1^n\triangleright) &\rightarrow^* \underbrace{\mathbf{1}:\cdots:\mathbf{1}}_{n-1}:q_h(s_n, t_n, 1^n\triangleright) \\ &\rightarrow \underbrace{\mathbf{1}:\cdots:\mathbf{1}}_n:q_s(\triangleright, 1^{n+1}\triangleright, 1^{n+1}\triangleright) \end{aligned}$$

for all $n \geq 1$. Hence, S reduces to the infinite normal form $\mathbf{1}:\mathbf{1}:\cdots$, as desired. \square

Proposition 15. *Deciding whether an orthogonal stream specification is productive is Π_2^0 -hard.*

Proof. By reduction from TOTALITY. Obviously, there is an effective procedure transform a description of any Turing machine M into the stream specification $(S, \Delta_S(M))$. The result now follows by Proposition 14. \square

Theorem 6. *Deciding whether an orthogonal stream specification is productive is Π_2^0 -complete.*

Proof. By Propositions 13 and 15, reasoning as in the proof of Theorem 1. \square

In the proof of membership of Π_2^0 (Proposition 13, we have crucially employed the assumption of orthogonality. We do not know if productivity of arbitrary stream specifications is in Π_2^0 . Roşu has proved that deciding equality of streams is Π_2^0 -complete (in an equational setting, but his result carries over to orthogonal stream specifications) [14]; it is also unknown whether that result holds for non-orthogonal stream specifications.

References

1. A. Arnold and M. Nivat. The metric space of infinite trees. algebraic and topological properties. *Fundamenta Informaticae*, 3(4):445–476, 1980.
2. F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, Cambridge, United Kingdom, 1998.
3. W. Buchholz. A term calculus for (co-)recursive definitions on streamlike data structures. *Annals of Pure and Applied Logic*, 136(1–2):75–90, 2005.
4. M. Dauchet, T. Heuillard, P. Lescanne, and S. Tison. Decidability of the confluence of finite ground term rewrite systems and of other related term rewrite systems. *Information and Computation*, 88(2):187–201, 1990.
5. E.W. Dijkstra. On the productivity of recursive definitions. EWD749, 1980.
6. J. Endrullis, C. Grabmayer, and D. Hendriks. Data-oblivious stream productivity. In *Proceeding of the 15th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR '08)*, volume 5330 of *Lecture Notes in Computer Science*, pages 79–96. Springer-Verlag, 2008.
7. J. Endrullis, C. Grabmayer, D. Hendriks, A. Ishihara, and J.W. Klop. Productivity of stream definitions. In *Proceedings of the Conference on Fundamentals of Computation Theory (FCT '07)*, volume 4639 of *Lecture Notes in Computer Science*, pages 274–287. Springer-Verlag, 2007.
8. G.T. Herman. Strong computability and variants of the uniform halting problem. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 17(1):115–131, 1971.
9. G. Huet and D.S. Lankford. On the uniform halting problem for term rewriting systems. Rapport Laboria 283, IRIA, 1978.
10. F. Klay. Undecidable properties of syntactic theories. In *Proceedings of RTA '91*, volume 488 of *Lecture Notes in Computer Science*, pages 136–149. Springer-Verlag, 1991.
11. J.W. Klop. Term rewriting systems. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, pages 1–116. Oxford University Press, 1992.
12. M. Oyamaguchi. The Church-Rosser property for ground term rewriting systems is decidable. *Theoretical Computer Science*, 49(1):43–79, 1987.
13. H. Rogers Jr. *Theory of Recursive Functions and Effective Computability*. The MIT Press, paperback edition, 1987.
14. G. Roşu. Equality of streams is a Π_2^0 -complete problem. In *Proceedings of the 11th ACM SIGPLAN International Conference on Functional Programming (ICFP '06)*, pages 184–191. The ACM Press, 2006.
15. B.A. Sijtsma. On the productivity of recursive list definitions. *ACM Transactions on Programming Languages and Systems*, 11(4):633–649, 1989.
16. M. Sipser. *Introduction to the Theory of Computation*. Thomson Course Technology, 2nd edition, 2006.
17. A. Telford and D. Turner. Ensuring streams flow. In *Proceedings of the International Conference on Algebraic Methodology and Software Technology (AMAST '97)*, volume 1349 of *Lecture Notes in Computer Science*, pages 509–523. Springer-Verlag, 1997.
18. Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.

A Full proofs omitted from the main text

Proof (Corollary 2). If there is a configuration wqw' on which M does not halt, then there is an infinite $\Delta(M)$ -reduction starting from the ground term $q(w, w')$. By construction, every reduct of $q(w, w')$ represents a configuration of M and as M is deterministic, $q(w, w')$ and every reduct contains at most one redex. Hence, all finite reductions starting from $q(w, w')$ are prefixes of the infinite reduction, showing that $q(w, w')$ has no normal form, whence $\Delta(M)$ is neither WN, nor ground-WN.

If M halts on all configurations, suppose for the purpose of contradiction that $\Delta(M)$ were not WN (hence not ground-WN). By Proposition 2 there is a restricted ground term t having an infinite reduction. But by the same proposition, any restricted ground term represents a configuration wqw' of M , entailing that M does not halt on wqw' , a contradiction. \square

Proof (Proposition 3). By induction on d :

- $d = 1$. If $s = A(s, s')$, then $t = A(s'', s''')$ and $t \rightarrow_c \text{FAIL}$. If $s = q(s, s')$, and the redex contracted is in, say s , we have $s \rightarrow_c s''$, $t = q(s'', s')$, and we have either $s'' = q(w, w')$ or $s'' = \text{FAIL}$. In either case, $t \rightarrow_c \text{FAIL}$. The case where the redex contracted is in s' is symmetric.
- $d > 1$. Let the position of the redex contracted in $s \rightarrow_c t$ be $p = p' \cdot i$ where $d = |p| = |p'| + 1$. Performing exactly the same case analysis on the subterm $s|_{p'}$ as in the case $d = 1$ yields that $t|_{p'} \rightarrow_c \text{FAIL}$, hence that $t \rightarrow_c t[\text{FAIL}]_{p'}$, and as $s|_{\leq d-1} = t|_{\leq d-1}$, the induction hypothesis yields that $t[\text{FAIL}]_{p'} \rightarrow_c^{d-1} \text{FAIL}$, as desired. \square

Proof (Proposition 4). If $\Delta_g(M)$ is terminating, then for every term t on the form $1^n \triangleright$, we have $S : q_s(\triangleright, t) \rightarrow q(s', s'') \rightarrow^* t'$ for some $q \neq q_s$ and terms s', s'' and normal form t' . As $q_s(\triangleright, t)$ contains no occurrences of FAIL and t is a representation of a configuration, no step of S can be a $\Delta_c(M)$ -step and every rewrite step simulates a step of M . Furthermore, we must have $t' = q_h(t'', t''')$ for terms t'', t''' , and hence M halts on all inputs.

Conversely, suppose that M halts on all configurations, assume for the purpose of contradiction that there is a ground term s that has an infinite reduction. The system $\Delta_c(M) \setminus \{A(x, y) \rightarrow_c q(x, y) : q \in Q\}$ is clearly SN and no right-hand side has a symbol in common with the redex pattern of any redex in $\Delta(M)$. Furthermore, if a redex of any rule $A(x, y) \rightarrow_c q(x, y)$ is contracted at depth $d \geq 1$ in a term, its right-hand side cannot contribute to any $\Delta(M)$ -redex at depth $d' < d$. Thus, there must be a ground term s' having an infinite reduction starting from it such that s' is either a restricted ground term $q(w, w')$ containing no occurrence of FAIL or A , or $s' = A(w, w')$ where w' and w' do not contain any occurrence of FAIL or A . But in both cases, the infinite reduction starting from s' simulates an infinite run of M on configuration wqw' , contradicting the assumption. \square

Proof (Proposition 5). Let s be a ground normal form of $\Delta_{\mathbf{g}}(M)$. If $s = \text{FAIL}$, we are done. If s contains an occurrence of FAIL at depth ≥ 1 , inspection of the rules of $\Delta_{\mathbf{c}}(M)$ yields that s cannot be a normal form (any occurrence of FAIL at depth $d+1$ entails presence of a redex at depth d). If s contains an occurrence of A , it contains a redex, whence A cannot occur in s .

Thus, s contains neither FAIL nor A . If s contains an occurrence of some q at depth ≥ 1 , inspection of the rules of $\Delta_{\mathbf{c}}(M)$ yields that a redex is present in s , a contradiction. Thus, s must be a restricted ground term, hence on the form $q(s', s'')$. If $q = q_h$, there is a $\rightarrow_{\mathbf{c}}$ -redex at the root, a contradiction. If $q \neq q_h$, the assumption that $\delta(q, b)$ is defined for all $q \neq q_h$ and $b \in \{0, 1, \square\}$ yields that a \leq_m -redex is present at the root of s , also a contradiction. Hence, the only ground normal form of $\Delta_{\mathbf{c}}(M)$ is FAIL . \square

Proof (Proposition 12). Proceed as follows:

- “ \Rightarrow ”: Suppose $((F \cup C, R), S)$ is productive and S thus has an infinite constructor normal form t . Then there is an infinite reduction $S \rightarrow^{\infty} t$, and as left-hand sides of rewrite rules $l \rightarrow r \in R$ are finite and no element of C occurs at the root of l , we may write $S \rightarrow^{\infty} t$ as

$$S \rightarrow^* t_0 \rightarrow^* t_1 \rightarrow^* t_2 \rightarrow^* \dots$$

where, for each $n \geq 0$, $t_n = C[s_1, \dots, s_m]$ with $C[x_1, \dots, x_m]$ a constructor term of depth n , as desired.

- “ \Leftarrow ”: We claim that for any $n \geq 0$ we have $t_n|_{\leq n} = t_{n+1}|_{\leq n}$ and that there exists a term t'_n such that

- $t_0 = t'_0$
- $t_n \rightarrow^* t'_n$
- $t'_n \rightarrow^* t'_{n+1}$ and $t_n|_{\leq n} = t'_n|_{\leq n} = t'_{n+1}|_{\leq n}$

To prove the claim, define $t'_0 \triangleq t_0$ and proceed by induction. Assume that the claim has been proved for $n \geq 0$. By the induction hypothesis, we obtain $t_n \rightarrow^* t'_n$, and by assumption $t_n^* \leftarrow S \rightarrow^* t_{n+1}$, whence $S \rightarrow^* t'_n$. As R is orthogonal, it is confluent, whence there exists a term t'_{n+1} such that $t'_n \rightarrow^* t'_{n+1} \leftarrow t_{n+1}$. As no rule $l \rightarrow r \in R$ has a constructor symbol at the root of l . Hence if $t_{n+1} = C[s_1, \dots, s_m]$ with $C[x_1, \dots, x_m]$ a constructor term of depth $n+1$, no redex contracted in $t_{n+1} \rightarrow^* t'_{n+1}$ occurs at a depth $\leq n+1$; thus, $t_{n+1}|_{\leq n+1} = t'_{n+1}|_{\leq n+1}$, concluding the proof of the claim.

By the claim there is an infinite reduction $S \rightarrow^* t_0 = t'_0 \rightarrow^* t'_1 \rightarrow^* t'_2 \rightarrow \dots$ where for all $n \geq 0$, $t'_n = C[s'_1, \dots, s'_m]$ for some m where $C[x_1, \dots, x_m]$ is a constructor term of depth n and $t'_n|_{\leq n} = t'_{n+1}|_{\leq n}$. Thus, the sequence of terms $(t'_n)_n$ converge in the usual tree metric [18] to some infinite constructor normal form t and S reduces to t by an infinite reduction¹. \square

¹ It is easy to see that the reduction constructed in the proof is *strongly convergent* [18] and thus also satisfies the basic requirement for being a well-behaved reduction in infinitary rewriting.