Information and Computation 209 (2011) 227–245

Contents lists available at ScienceDirect

Information and Computation

journal homepage: www.elsevier.com/locate/ic

Levels of undecidability in rewriting

Jörg Endrullis [a], Herman Geuvers [b,c], Jakob Grue Simonsen [d], Hans Zantema [c,b,*]

[a] Department of Computer Science, Free University Amsterdam, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
[b] Institute for Computing and Information Sciences, Radboud University Nijmegen, P.O. Box 9010, 6500 GL Nijmegen, The Netherlands
[c] Department of Computer Science, TU Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
[d] Department of Computer Science, University of Copenhagen (DIKU) Universitetsparken 1, DK-2100 Copenhagen, Denmark

ARTICLE INFO

Article history:
Received 5 March 2010
Available online 27 October 2010

ABSTRACT

Undecidability of various properties of first-order term rewriting systems is well-known. An undecidable property can be classified by the complexity of the formula defining it. This classification gives rise to a hierarchy of distinct levels of undecidability, starting from the arithmetical hierarchy classifying properties using first order arithmetical formulas, and continuing into the analytic hierarchy, where quantification over function variables is allowed.

In this paper we give an overview of how the main properties of first order term rewriting systems are classified in these hierarchies. We consider properties related to normalization (strong normalization, weak normalization and dependency problems) and properties related to confluence (confluence, local confluence and the unique normal form property). For all of these we distinguish between the single term version and the uniform version. Where appropriate, we also distinguish between ground and open terms.

Most uniform properties are Π^0_2-complete. The particular problem of local confluence turns out to be Π^0_2-complete for ground terms, but only Σ^0_1-complete (and thereby recursively enumerable) for open terms. The most surprising result concerns dependency pair problems without minimality flag: we prove this problem to be Π^1_1-complete, hence not in the arithmetical hierarchy, but properly in the analytic hierarchy.

Some of our results are new or have appeared in our earlier publications. Others are based on folklore constructions, and are included for completeness as their precise classifications have hardly been noticed previously.

1. Introduction

In classical computability theory properties on countable objects are investigated by explicitly enumerating the elements of the object. Thus, a property can be identified with a set $P \subseteq \mathbb{N}$ representing the set of elements for which the property holds. Such a set P is called *decidable* if there exists a Turing machine which for every input $x \in \mathbb{N}$ outputs 1 if $x \in P$ and 0 if $x \notin P$.

Investigating the complexity of decidable properties is well-known, defined in terms of the time (or space) consumption of a Turing machine that decides the property. Likewise, but less well-known, the undecidable properties can be classified into a hierarchy of growing complexity: one undecidable property can be harder than another. As we have infinite hierarchies like linear-quadratic-cubic-$\cdots$ in the complexity of decidable properties, there are similar infinite hierarchies in the universe of undecidable properties.

* Corresponding author. Address: Department of Computer Science, TU Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands.
E-mail addresses: joerg@few.vu.nl (J. Endrullis), herman@cs.ru.nl (H. Geuvers), simonsen@diku.dk (J.G. Simonsen), h.zantema@tue.nl (H. Zantema).

0890-5401/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2010.09.003

The twin notions of the *arithmetical hierarchy* [20] and the *analytical hierarchy* [21,22] establish such a classification of undecidable properties by the complexity of predicate logic formulas that define them—in turn defined as the number of quantifier alternations of the prenex normal form of the formula.

The arithmetical hierarchy is based on first order formulas: quantifiers are restricted to range over integers The classes in the arithmetical hierarchy are denoted Π^0_n and Σ^0_n for $n \in \mathbb{N}$. The lowest level of the hierarchy consists of the classes Π^0_0 and Σ^0_0, both consisting of the decidable properties. Then the classes Π^0_n and Σ^0_n for $n \geq 1$ are inductively defined: a property is in Π^0_n if it can be written as $\forall x P$ for $P \in \Sigma^0_{n-1}$, and a property is in Σ^0_n if it can be written as $\exists x P$ for $P \in \Pi^0_{n-1}$. For example, if $P(x, y, z)$ is a decidable property, then $\exists x\, P(x, y, z)$ is in Σ^0_1 and $\forall y\, \exists x\, P(x, y, z)$ is in Π^0_2.

Hence, a property belongs to the class Π^0_n for $n \in \mathbb{N}$ in the arithmetical hierarchy if it can be defined by a first order formula (in prenex normal form), which has n quantifiers, with the outermost quantifier being universal. Likewise a property is in Σ^0_n if the outermost quantifier is existential. The class Σ^0_1 is the class of *recursively enumerable* (or semi-decidable) properties. The blank tape halting problem is in this class. The initialized uniform halting problem is in the class Π^0_2.

Let Y be a property (typically: Y represents a term rewriting system or Turing machine with a particular property) For a class, X, of properties, Y is called *X-hard* if any property in X can be reduced to Y. It is called *X-complete* if it is both in X and X-hard. Under suitable encodings, the blank tape halting problem for Turing machines is Σ^0_1-complete; the initialized uniform halting problem is Π^0_2-complete. For proving X-hardness of any new property it suffices to reduce the standard instances such as the above to the new property.

The *analytical hierarchy* continues the classification of properties by second order formulas, allowing for quantifiers ranging over functions. An important class is Π^1_1, consisting of the properties which can be defined by $\forall \alpha : \mathbb{N} \to \mathbb{N}.\ \varphi$ where φ is an arithmetical relation.

Extensive theory concerning these hierarchies has been developed; excellent textbooks summarizing the major results are [14,31,32].

Up to renaming of function symbols, the set of all TRSs (term rewriting systems) is countable, and properties of TRSs can be classified in the arithmetical and analytical hierarchies. The goal of this paper is to establish such a classification for the most basic properties of first order TRSs. All of the properties we consider are already known to be undecidable. For some of them, their level in the relevant hierarchy easily follows from the folklore proofs of undecidability. For others, the investigation has appeared earlier in our papers [7,33]. For some others the results in this paper are new. For the sake of completeness, all of these results are presented in this paper.

One may wonder why these results would be of interest. A tantalizing argument *against* an investigation such as the one we perform is that, for undecidable problems, no algorithm exists solving them, so further investigation is useless from a practical perspective. On the other hand, the arithmetical and analytical hierarchy provide a natural, general and objective way to classify levels of undecidability, and state that some undecidable problems are essentially harder than others. This may serve for a better understanding of the hardness of several problems. For instance, checking local confluence (on open terms) can be done by critical pair analysis, while for checking local ground confluence (that is, on closed terms) this approach does *not* apply. The critical pair analysis implies the problem to be in Σ^0_1, while we show Π^0_2-hardness of local ground confluence. From this we can conclude that it is not due to lack of human effort that no technique is known for checking local ground confluence in a way similar to critical pair analysis; on the contrary, such a technique is fundamentally *impossible*.

1.1. Results

We investigate the complexity of the following properties of first order TRSs.

- **Basic properties.** Each of the following problems is on the form: "Given terms t, u, decide whether the property holds". We prove that
  - *reduction* ($t \to^* u$),
  - *joinability* ($t \to^* \cdot \leftarrow^* u$), and
  - *conversion* ($t \leftrightarrow^* u$)

  are all Σ^0_1-complete.
- **Termination-related properties.** We prove that
  - termination or strong normalization (SN) of single terms is Σ^0_1-complete; uniform termination is Π^0_2-complete;
  - weak normalization (WN) of single terms is Σ^0_1-complete; uniform weak normalization is Π^0_2-complete;
  - finiteness of dependency pair problems (DP) is Π^1_1-complete, and
  - finiteness of dependency pair problems with minimality flag ($\mathsf{DP}^{\mathsf{min}}$) is Π^0_2-complete.

  The results hold both for general (open) terms and for ground terms.
- **Confluence-related properties.** We prove that
  - confluence (CR) is Π^0_2-complete, no matter whether this is taken for single terms or uniform, or restricting to ground terms or not;
  - uniform local ground confluence (grWCR) is Π^0_2-complete;

– the other three variants of local confluence (WCR): single term local ground confluence, single term local confluence and uniform local confluence, are all Σ_1^0-complete;
– the unique normal form property is Π_1^0-complete, both for the variants UN and $\mathsf{UN}^{\rightarrow}$, and both for single terms and the uniform version.

These results are summarized in the following tables:

$\rightarrow^*$	$\rightarrow^* \cdot \leftarrow^*$	$\leftrightarrow^*$	DP	$\mathsf{DP}^{\mathsf{min}}$
Σ_1^0	Σ_1^0	Σ_1^0	Π_1^1	Π_2^0

	SN	WN	CR	grCR	WCR	grWCR	UN	$\mathsf{UN}^{\rightarrow}$
uniform	Π_2^0	Π_2^0	Π_2^0	Π_2^0	Σ_1^0	Π_2^0	Π_1^0	Π_1^0
single term	Σ_1^0	Σ_1^0	Π_2^0	Π_2^0	Σ_1^0	Σ_1^0	Π_1^0	Π_1^0

While undecidability of many of these problems is folklore [10,11], their precise hardness has hardly been studied. An exception is [15], the oldest reference to undecidability of properties of TRSs, which also includes an observation on the Turing degree of termination, hence essentially proving termination Π_2^0-complete. As we use a slightly different translation from Turing machines to TRSs, we do not use the results or proofs of [15].

We find that the standard TRS properties SN, WN, CR, WCR and UN for both the uniform and single term versions, all reside within the classes Π_2^0, Π_1^0 and Σ_1^0 of the arithmetical hierarchy.

A somewhat unexpected result we already mentioned is that local ground confluence is a *harder* decision problem than local confluence. While local confluence is Σ_1^0-complete and therefore recursively enumerable, it turns out that local ground confluence Π_2^0-complete.

Surprisingly, it turns out that dependency pair problems are of a much higher degree of undecidability: they properly exceed the expressive power of first order predicate logic, hence are not in the arithmetical hierarchy. In particular we show that dependency pair problems are Π_1^1-complete. So although dependency pair problems were invented for proving termination, the complexity of general dependency pair problems is much higher than the complexity of termination itself. As the basic technique to prove Π_1^1-hardness is by checking whether a given relation is well-founded, dependency pair problems share a natural proof-theoretic level with one of the most basic approaches to proving termination: finding a compatible well-founded order.

A variant of dependency pair problems are dependency pair problems with minimality flag. We show that for this variant the complexity is back to that of termination: it is Π_2^0-complete.

It should be noted that while our results are proved only for (first-order) TRSs, the *hardness* results also hold for variants of rewriting that faithfully simulate the reduction relation of TRSs; this is the case for each of the standard variants of higher-order rewriting systems, see e.g. [37] for a survey. For such systems, our *completeness* results will also hold when the rewrite relation is decidable and terms are finite: for instance, every reduction in the standard variants of higher-order rewriting can be encoded as an integer and checked by suitable decidable predicates P; hence, both the single term and uniform versions of CR are Π_2^0-complete for combinatory reduction systems (CRS) and for pattern rewrite systems (PRS). We believe that our results are best presented by using the standard, simple, and universally accepted notation for TRSs, whence we leave the straightforward details of lifting our proofs to the various notations of higher-order systems to the reader.

1.2. Related work on decidability in rewriting

The first undecidability results in rewriting concerned untyped lambda calculus where normalization and termination was shown to be undecidable in 1937 [36]. For first-order term rewriting, uniform termination was shown to be Π_2^0-complete in 1978 by Huet and Lankford [15]. As their construction uses only unary symbols, the result even holds for string rewriting systems. Since then for several properties in rewriting undecidability was proved, but all typically without exact classification in the arithmetical hierarchy. Undecidability of (local and global) confluence on ground or non-ground terms is a folklore result appearing in several textbooks, see for example [35]; one of the approaches is based on undecidability of the word problem for groups [4,29] and semigroups [26]. Undecidability of termination of a single term rewrite rule was proved in [5]. It is still an open problem whether termination of a single string rewrite rule is decidable. Undecidability of simple termination was proved in [27]. There is a substantial literature on more undecidability results in rewriting, including [9–11,17,19,40]. Conversely, intense research has been devoted to finding decidable special cases of termination, normalization and confluence, see for example [6,16,28,38].

For some variants of rewriting systems, certain properties decidable in ordinary TRSs become undecidable. As an example, this holds for all three standard variants of conditional term rewriting systems (CTRSs), where it is undecidable whether a term is in normal form, and indeed, the one-step rewrite relation is undecidable [3,18]; but as usual, imposing syntactic restrictions garners decidable subcases [3]. The classification of undecidability results for CTRSs in the arithmetical hierarchy has not yet been studied.

There are some results in stream specifications. In [30] it was proved that equality of streams is a Π^0_2-complete problem. In fact this work triggered the authors to start the investigations leading to the current paper. Also productivity of stream specifications has been proved to be Π^0_2-complete. This has been observed independently by several people; one proof was given in [33].

1.3. Structure of the paper

This paper is organized as follows.

- In Section 2 we give preliminaries on term rewriting, and define the basic properties to be classfied into the arithmetical and analytical hierarchies.
- In Section 3 we present Turing machines.
- In Section 4 we introduce the arithmetic and analytic hierarchy, and outline the proof obligations for proving X-completeness where X is one of Π^0_1, Σ^0_1, Π^0_2 and Π^1_1.
- In Section 5 we come to our first results: we give a basic transformation from Turing machines to TRSs such that (i) the halting of the Turing machine on the blank tape coincides with termination on a particular starting term, and (ii) uniform halting of the Turing machine coincides with uniform termination. The same construction serves for weak normalization.
- In Section 6 we give a slight extension to this basic construction by which our results on the basic properties are obtained.
- In Section 7 we give our results on some variants of the unique normal form property, again by a slight extension to the basic construction.
- In Section 8 we present our results on confluence, both the general and ground variant. Again the key is an extension of the basic construction by a few rewrite rules.
- In Section 9 we do the same for local confluence.
- In Section 10 we make the step to the analytic hierarchy: we prove our results on dependency pair problems.
- In Section 11 we are back in the arithmetic hierarchy: we show that dependency pair problems with minimality flag behave as normal termination.
- Finally, in Section 12 we give conclusions and discuss future work.

2. Term rewriting

We briefly review the most basic concepts and notation for term rewriting systems; comprehensive accounts can be found in, for example, [23,35].

A *signature* Σ is a finite set of symbols each having a fixed *arity* $\sharp(f) \in \mathbb{N}$. Let Σ be a signature and $\mathcal{X}$ a set of variable symbols such that $\Sigma \cap \mathcal{X} = \varnothing$. The *set* $Ter(\Sigma, \mathcal{X})$ *of terms over* Σ *and* $\mathcal{X}$ is the smallest set satisfying:

- $\mathcal{X} \subseteq Ter(\Sigma, \mathcal{X})$, and
- $f(t_1, \ldots, t_n) \in Ter(\Sigma, \mathcal{X})$ if $f \in \Sigma$ with arity n and $\forall i : t_i \in Ter(\Sigma, \mathcal{X})$.

We use $x, y, z, \ldots$ to range over variables. The set of positions $\mathcal{P}os(t) \subseteq \mathbb{N}^*$ of a term $t \in Ter(\Sigma, \mathcal{X})$ is inductively defined by: $\mathcal{P}os(f(t_1, \ldots, t_n)) = \{\varepsilon\} \cup \{ip \mid 1 \le i \le \sharp(f),\ p \in \mathcal{P}os(t_i)\}$, and $\mathcal{P}os(x) = \{\varepsilon\}$ for variables $x \in \mathcal{X}$. We use $\equiv$ for syntactical equivalence of terms.

A substitution σ is a map $\sigma : \mathcal{X} \to Ter(\Sigma, \mathcal{X})$ from variables to terms. For terms $t \in Ter(\Sigma, \mathcal{X})$ and substitutions σ we define $t\sigma$ as the result of replacing each $x \in \mathcal{X}$ in t by $\sigma(x)$. That is, $t\sigma$ is inductively defined by $x\sigma := \sigma(x)$ for variables $x \in \mathcal{X}$ and otherwise $f(t_1, \ldots, t_n)\sigma := f(t_1\sigma, \ldots, t_n\sigma)$. Let $\Box$ be a fresh symbol, $\Box \notin \Sigma \cup \mathcal{X}$. A *context* C is a term from $Ter(\Sigma, \mathcal{X} \cup \{\Box\})$ containing precisely one occurrence of $\Box$. Then $C[s]$ denotes the term $C\sigma$ where $\sigma(\Box) = s$ and $\sigma(x) = x$ for all $x \in \mathcal{X}$.

A *term rewriting system (TRS)* over $\Sigma, \mathcal{X}$ is a set R of finitely many pairs $\langle \ell, r\rangle \in Ter(\Sigma, \mathcal{X})$, called *rewrite rules* and usually written as $\ell \to r$, for which the *left-hand side* ℓ is not a variable ($\ell \notin \mathcal{X}$) and all variables in the *right-hand side* r occur in ℓ ($Var(r) \subseteq Var(\ell)$). Let R be a TRS. For terms $s, t \in Ter(\Sigma, \mathcal{X})$ we write $s \to_R t$ if there exists a rule $\ell \to r \in R$, a substitution σ and a context $C \in Ter(\Sigma, \mathcal{X} \cup \{\Box\})$ such that $s \equiv C[\ell\sigma]$ and $t \equiv C[r\sigma]$; $\to_R$ is the *rewrite relation* induced by R. In case R is fixed, we shortly write $\to$ for $\to_R$.

A *rewrite sequence* with respect to R is a finite or infinite sequence $u_1, u_2, \ldots$ of terms such that $u_i \to_R u_{i+1}$ for all appropriate i. If $\vec{u} = u_1, \ldots, u_n$ is a finite rewrite sequence with $t = u_1$ and $s = u_n$, we say that $\vec{u}$ *is a rewrite sequence from* t *to* s and we write $t \xrightarrow{\vec{u}}{}^*_R s$, occasionally omitting $\vec{u}$ and/or R. If $\vec{u}$ is empty we just have $s = t$.

A *normal form* t with respect to R is a term t such that no term u exists for which $t \to_R u$. We write $\to^*$ for the reflexive transitive closure of $\to$, and $\leftrightarrow^*$ for the reflexive symmetric transitive closure of $\to$.

Definition 2.1. Let R be a TRS and $t \in \mathit{Ter}(\Sigma, \mathcal{X})$ a term. Then R

- is *strongly normalizing (or terminating) on t*, denoted $\mathsf{SN}_R(t)$, if every rewrite sequence starting from t is finite;
- is *weakly normalizing on t*, denoted $\mathsf{WN}_R(t)$, if t admits a rewrite sequence $t \to^* s$ to a normal form s;
- is *confluent (or Church–Rosser) on t*, denoted $\mathsf{CR}_R(t)$, if every pair of finite coinitial reductions starting from t can be extended to a common reduct, that is, $\forall t_1, t_2.\ t_1 \leftarrow^* t \to^* t_2 \Rightarrow \exists d.\ t_1 \to^* d \leftarrow^* t_2$;
- is *locally confluent (or weakly Church–Rosser) on t*, denoted $\mathsf{WCR}_R(t)$, if every pair of coinitial rewrite steps starting from t can be joined, that is, $\forall t_1, t_2.\ t_1 \leftarrow t \to t_2 \Rightarrow \exists d.\ t_1 \to^* d \leftarrow^* t_2$;
- has the *unique normal form property on t*, denoted $\mathsf{UN}_R(t)$, if there is at most one normal form n satisfying $t \leftrightarrow^* n$,
- has the *unique normal form property on t with respect to reduction*, denoted $\mathsf{UN}^{\rightarrow}_R(t)$, if there is at most one normal form n satisfying $t \to^* n$.

The TRS R is *strongly normalizing* (SN_R), *weakly normalizing* (WN_R), *confluent* (CR_R) or *locally confluent* (WCR_R), or has the *unique normal form property (with respect to reduction)* (UN_R, respectively $\mathsf{UN}^{\rightarrow}_R$) if the respective property holds on all terms $t \in \mathit{Ter}(\Sigma, \mathcal{X})$. For each property P, we say that R has the "*ground P*"-property if P holds for all ground terms $t \in \mathit{Ter}(\Sigma, \varnothing)$.

A fruitful variant of König's Lemma is the following.

Lemma 2.2. *A finitely branching relation $\to$ is terminating on t if and only if there exists $n \in \mathbb{N}$ such that every reduction of t has length $< n$.*

Proof. The 'if'-part is trivial. For the 'only if'-part assume t has reductions of unbounded length. As t has finitely many successors, at least one of these successors has reductions of unbounded length too. Repeating the argument on this successor yields an infinite reduction, contradicting the assumption of termination. □

3. Turing machines

Definition 3.1. We now recapitulate the notions for Turing machines we need in the remainder of the paper; comprehensive accounts of Turing machines and computability can be found in [8,31,34].

A *Turing machine* M is a quadruple $\langle Q, \Gamma, q_0, \delta\rangle$ consisting of:

- finite set of states Q,
- an initial state $q_0 \in Q$,
- a finite alphabet Γ containing a designated symbol $\square$, called *blank*, and
- a partial *transition function* $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$.

A *configuration* of a Turing machine is a pair $\langle q, \mathit{tape}\rangle$ consisting of a state $q \in Q$ and the tape content $\mathit{tape} : \mathbb{Z} \to \Gamma$ such that the carrier $\{n \in \mathbb{Z} \mid \mathit{tape}(n) \neq \square\}$ is finite. The set of all configurations is denoted $\mathcal{C}\mathit{onf}_{\mathsf{M}}$. We define the relation $\to_{\mathsf{M}}$ on the set of configurations $\mathcal{C}\mathit{onf}_{\mathsf{M}}$ as follows: $\langle q, \mathit{tape}\rangle \to_{\mathsf{M}} \langle q', \mathit{tape}'\rangle$ whenever:

- $\delta(q, \mathit{tape}(0)) = \langle q', f, L\rangle$, $\mathit{tape}'(1) = f$ and $\forall n \neq 0.\ \mathit{tape}'(n+1) = \mathit{tape}(n)$, or
- $\delta(q, \mathit{tape}(0)) = \langle q', f, R\rangle$, $\mathit{tape}'(-1) = f$ and $\forall n \neq 0.\ \mathit{tape}'(n-1) = \mathit{tape}(n)$.

Without loss of generality we assume that $Q \cap \Gamma = \varnothing$. This enables us to denote configurations as $\langle w_1, q, w_2\rangle$, denoted $w_1^{-1} q w_2$ for short, with $w_1, w_2 \in \Gamma^*$ and $q \in Q$, which is shorthand for $\langle q, \mathit{tape}\rangle$ where $\mathit{tape}(n) = w_2(n+1)$ for $0 \leq n < |w_2|$, and $\mathit{tape}(-n) = w_1(n)$ for $1 \leq n \leq |w_1|$ and $\mathit{tape}(n) = \square$ for all other positions $n \in \mathbb{Z}$.

The Turing machines we consider are deterministic. As a consequence, final configurations are unique (if they exist), which justifies the following definition.

Definition 3.2. Let M be a Turing machine and $\langle q, \mathit{tape}\rangle \in \mathcal{C}\mathit{onf}_{\mathsf{M}}$. We denote by $\mathit{final}_{\mathsf{M}}(\langle q, \mathit{tape}\rangle)$ the $\to_{\mathsf{M}}$-normal form of $\langle q, \mathit{tape}\rangle$ if it exists and undefined, otherwise. Whenever $\mathit{final}_{\mathsf{M}}(\langle q, \mathit{tape}\rangle)$ exists then we say that M *halts on* $\langle q, \mathit{tape}\rangle$ *with final configuration* $\mathit{final}_{\mathsf{M}}(\langle q, \mathit{tape}\rangle)$. Furthermore we say M *halts on tape* as shorthand for M *halts on* $\langle q_0, \mathit{tape}\rangle$.

Turing machines can compute n-ary functions $f : \mathbb{N}^n \to \mathbb{N}$ or relations $S \subseteq \mathbb{N}^*$. For our purposes, we require only unary functions f_{M} and binary relations $>_{\mathsf{M}} \subseteq \mathbb{N} \times \mathbb{N}$.

Definition 3.3. Let $\mathsf{M} = \langle Q, \Gamma, q_0, \delta\rangle$ be a Turing machine with $\mathsf{S}, 0 \in \Gamma$. We define a partial function $f_{\mathsf{M}} : \mathbb{N} \rightharpoonup \mathbb{N}$ for all $n \in \mathbb{N}$ by:

$$f_{\mathsf{M}}(n) = \begin{cases} m & \text{if } \mathit{final}_{\mathsf{M}}(q_0\mathsf{S}^n0) = \cdots q\mathsf{S}^m0\cdots \\ \text{undefined} & \text{otherwise} \end{cases}$$

and for M total (i.e. M halts on all tapes) we define the binary relation $>_{\mathsf{M}} \subseteq \mathbb{N}\times\mathbb{N}$ by:

$$n >_{\mathsf{M}} m \iff \mathit{final}_{\mathsf{M}}(0\mathsf{S}^nq_0\mathsf{S}^m0) = \cdots q0\cdots .$$

Observe that the set $\{ >_{\mathsf{M}} |\ \mathsf{M}$ a Turing machine that halts on all tapes $\}$ is the set of recursive binary relations on $\mathbb{N}$.

4. Levels of undecidability

In the introduction we have briefly mentioned the arithmetical and analytical hierarchy. We now summarize the main notions and results relevant for this paper. For details we refer to standard texts on mathematical logic, e.g. [31,32]. that contain further technical results regarding these hierarchies.

Definition 4.1. Let $A \subseteq \mathbb{N}$. The *set membership problem for A*, or just the *problem A*, is the question of deciding for given $a \in \mathbb{N}$ whether $a \in A$.

In the following, we usually identify the membership problem for A with the set A itself, hence we shall refer to A as a *problem*. There is an obvious relation between a problem defined as a predicate over the natural numbers and as a set: $\varphi(n)$ iff $n \in \{m \in \mathbb{N} | \varphi(m)\}$, so we will interchange these notions freely.

Definition 4.2. Let $A \subseteq \mathbb{N}$ and $B \subseteq \mathbb{N}$. Then *A can be many-one reduced to B*, notation $A \leq_m B$ if there exists a total computable function $f : \mathbb{N} \to \mathbb{N}$ such that $\forall n \in \mathbb{N}.\ n \in A \Leftrightarrow f(n) \in B$.

In the remainder of the text, we refer to many-one reductions simply as *reductions*.

Definition 4.3. Let $B \subseteq \mathbb{N}$ and $\mathcal{P} \subseteq \mathbf{2}^{\mathbb{N}}$. Then B is called *$\mathcal{P}$-hard* if every $A \in \mathcal{P}$ can be reduced to B, and B is *$\mathcal{P}$-complete* whenever additionally $B \in \mathcal{P}$.

Thus, a problem B is *$\mathcal{P}$-hard* if, for every problem $A \in \mathcal{P}$, we can reduce a question about A to a question about B: To decide "$n \in A$" we need only decide "$f(n) \in B$", where f is the total computable function that reduces A to B.

The classification results in the following sections employ the following well-known lemma, which states that whenever a problem A can be reduced via a computable function to a problem B, then B is at least as hard as A.

Lemma 4.4. *If A can be reduced to B and A is $\mathcal{P}$-hard, then B is $\mathcal{P}$-hard.*

4.1. Preliminaries on encoding

The decidability problems we deal with in this paper are concerned with term rewriting systems and Turing machines. To phrase these problems in terms of natural numbers involves a coding step: one encodes a Turing machine M as a natural number $\ulcorner\mathsf{M}\urcorner$ and subsequently transforms the question 'Does the Turing machine M halt on the blank tape?' to the problem $\{n | n = \ulcorner\mathsf{M}\urcorner$ and M halts on the blank tape$\}$, which is a subset of $\mathbb{N}$.

The above procedure is standard and the actual encodings are therefore usually not spelled out. The crucial property is that the encoding is computable; for historical reasons, such computable encodings are usually called 'effective':

Definition 4.5. We call an encoding $\ulcorner - \urcorner$ of Turing machines as elements of $\mathbb{N}$ *effective* in case

(i) one can decide if a number n is the encoding of some Turing machine M,
(ii) one can decide if the encoded Turing machine $\ulcorner\mathsf{M}\urcorner$ can do a computation step $\langle q, \mathit{tape}\rangle \to_{\mathsf{M}} \langle q', \mathit{tape}'\rangle$.

The second condition means that the set $\{(m, n_q, n_t, n'_q, n'_t) | m = \ulcorner\mathsf{M}\urcorner, n_q = \ulcorner q\urcorner, n_t = \ulcorner \mathit{tape}\urcorner, n'_q = \ulcorner q'\urcorner, n'_t = \ulcorner \mathit{tape}'\urcorner, \langle q, \mathit{tape}\rangle \to_{\mathsf{M}} \langle q', \mathit{tape}'\rangle\}$ is decidable.

Henceforth we will assume that we have an effective coding for Turing machines. The existence of such an encoding rests on the fact that we can effectively encode finite lists of natural numbers as natural numbers. This can, for example, be done using the well-known Gödel encoding: $\langle n_1, \ldots, n_k\rangle := p_1^{n_1+1} \cdot \ldots p_k^{n_k+1}$, where $p_1, \ldots, p_k$ are the first k prime numbers

[31], or by using an encoding of Turing machines as lists of bits without leading zeroes and observing that each such list corresponds to a unique natural number [25].

Definition 4.6. We call an encoding $\langle - \rangle$ of finite list of numbers as elements of $\mathbb{N}$ *effective* in case

(i) we have a computable length function $\mathrm{lth}\langle n_1, \ldots, n_k \rangle = k$,
(ii) we have a computable projection function $(-)_i$ such that $(\langle n_1, \ldots, n_k \rangle)_i = n_i$ (if $1 \leq i \leq k$),
(iii) it is decidable if a number is the encoding of a list: $\mathrm{Seq}(n) \Leftrightarrow \exists n_1, \ldots, n_k (n = \langle n_1, \ldots, n_k \rangle)$ is decidable.

Using the encoding of finite lists of natural numbers, we can effectively encode Turing machines. If fix such an encoding, the following, known as Kleene's T-predicate, is a well-known decidable problem: $T(m, x, u, y) := m$ encodes a Turing Machine M, u encodes the computation of M on x whose end result is y. Using the Gödel encoding, we can rephrase this as a predicate over $\mathbb{N}$. In the present paper, we will – as usual – suppress these encodings and just say that

$$T(\mathsf{M}, x, u, y) := u \text{ is a computation of } \mathsf{M} \text{ on input } x \text{ with output } y$$

is decidable. Similarly, we can effectively encode notions from term rewriting as natural numbers and thus we can cast problems of TRSs as problems over $\mathbb{N}$. The only requirement of this encoding is that *matching* (whether a given term matches a certain pattern) should be decidable.

Definition 4.7. We call an encoding $\ulcorner - \urcorner$ of TRSs as elements of $\mathbb{N}$ *effective* if for any TRS R, from an encoding of a term $\ulcorner s \urcorner$, we can compute all triples $\langle \ulcorner \ell \urcorner, \ulcorner \sigma \urcorner, \ulcorner C \urcorner \rangle$ such that $s \equiv C[\ell\sigma]$.

An effective encoding can easily be constructed for the class of finite first-order TRSs due to the fact that it is decidable whether a rewrite rule applies to a term in a (finite) TRS. Furthermore, we will always require that there are *finitely* many rewrite rules, whence reduction is finitely branching.

We will leave the encoding implicit and draw a couple of basic consequences regarding decidability from the existence of an effective encoding of TRSs as described in Definition 4.7. These will be the basics for all our decidability results in TRSs.

Lemma 4.8. *The following TRS properties are decidable.*

(i) *Given a finite TRS R, it is decidable if a term s is in normal form.*
(ii) *Given a finite TRS R and two terms t and s in R, it is decidable whether $s \to_R t$.*
(iii) *Given a finite TRS R, two terms t, s and a finite list of terms $\vec{u}$ in R, it is decidable whether $s \overset{\vec{u}}{\to}{}^*_R t$, that is, whether $\vec{u}$ is a reduction sequence from s to t.*

Proof. The first is obvious. For the second,

$$s \to_R t \quad \equiv \quad \exists \ell \to r \in R \, \exists \sigma \, \exists C \, (s = C[\ell\sigma] \wedge t = C[r\sigma]).$$

Each of these existential quantifiers is bounded, so they amount to a finite search; hence, this is a decidable problem. The fact that the TRS is finite is crucial here. For the third, $s \overset{\vec{u}}{\to}{}^*_R t$ just means

$$s = u_1 \wedge u_1 \to_R u_2 \wedge u_2 \to_R u_3 \wedge \cdots \wedge u_{n-1} \to_R u_n \wedge u_n = t,$$

so this follows from the second. □

4.2. The arithmetical and analytical hierarchies

Undecidable problems can be divided into hierarchies of increasing complexity, the most well-known of which is the *arithmetical hierarchy*. An example of a problem in this hierarchy is the problem whether t reduces in finitely many steps to s: $t \to^*_R s$, i.e. whether $\exists \vec{u}(t \overset{\vec{u}}{\to}{}^*_R s)$. Observe that as terms, term rewriting systems and reduction sequences may be encoded as natural numbers, we may regard the example as a "problem" in the sense of Definition 4.1.

The problem is undecidable in general and resides in the class Σ^0_1, which is the class of problems $A \subseteq \mathbb{N}$ such that $A = \{n | \exists x \in \mathbb{N} \, P(x, n)\}$ where $P(x, n)$ is a decidable problem. Similar to Σ^0_1, there is the class Π^0_1, which is the class of problems A such that $A = \{n | \forall x \in \mathbb{N} \, P(x, n)\}$ with $P(x, n)$ a decidable problem. If we continue this procedure in the obvious manner, we obtain the classes Σ^0_n and Π^0_n for every $n \in \mathbb{N}$. Before we introduce the classes Σ^0_n and Π^0_n, we recall and fix some conventions.

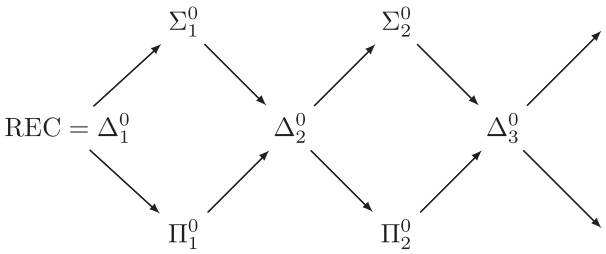

Fig. 1. Arithmetical hierarchy.

Remark 4.9

- We will suppress the domain of the existential quantifier: modulo an effective encoding as natural numbers, the domain may be any countable (or finite) set.
- As there exists an effective bijective encoding of natural numbers as finite lists of natural numbers (Definition 4.6), we may replace any quantification over $\mathbb{N}^k$ by a single quantification over $\mathbb{N}$, and any problem over $\mathbb{N}^k$ can thus be seen as a problem over $\mathbb{N}$. Therefore we identify, for example, a problem over $\mathbb{N}^2$ with a problem over $\mathbb{N}$.
- A finite sequence of existential quantifiers can always be replaced by a single existential quantifier, because $\exists x, y(P(x, y)) \iff \exists z \in \mathbb{N}(P((z)_1, (z)_2))$. Similarly for the universal quantifier.

Definition 4.10. Set $\Sigma_0^0 := \Pi_0^0 :=$ REC, the class of decidable problems.
For $n \geq 1$, we define:
The class of problems Σ_n^0 consists of the A such that $A = \{n | \exists x \in \mathbb{N}\, P(x, n)\}$ where P is in Π_{n-1}^0.
The class of problems Π_n^0 consists of the A such that $A = \{n | \forall x \in \mathbb{N}\, P(x, n)\}$ where P is in Σ_{n-1}^0.
Finally, $\Delta_n^0 := \Sigma_n^0 \cap \Pi_n^0$.

Thus, a problem $A(x)$ in Σ_n^0 can be written as $\exists y_1 \forall y_2 \ldots Qy_n(R(x, y_1, \ldots, y_n))$, where Q is $\exists$ or $\forall$, depending on the parity of n, and $R(x, y_1, \ldots, y_n)$ is decidable. Similarly, $A(x)$ in Π_n^0 can be written as $\forall y_1 \exists y_2 \ldots Qy_n(R(x, y_1, \ldots, y_n))$.

Recall (e.g. from [31]) that the set of formulas in first-order arithmetic are built from propositional connectives, quantifiers, equality, and a small set of constants for multiplication, addition and comparison of natural numbers. The usefulness of Definition4.10 is illustrated in the following lemma; we refer to [14,31,32] for a proof and further details.

Lemma 4.11. *Every formula in first order arithmetic is equivalent to a formula in prenex normal form, i.e. a formula with all quantifiers on the outside of the formula.*

Together with the fact that a sequence of quantifiers of the same type can be replaced by one, as pointed out in Remark 4.9, we may conclude that every problem that can be described by a formula in first order arithmetic is in one of the classes Σ_n^0 or Π_n^0.

The reason for the superscript "0" in Σ_n^0 and Π_n^0 is that all quantifiers in the formulae characterizing the classes range over "the lowest possible type": $\mathbb{N}$; there are no quantifiers over elements of higher type, e.g. elements of $\mathbb{N} \to \mathbb{N}$.

A natural question is whether the classes defined in Definition 4.10 are distinct. The following fundamental result in logic says that they are (see also [32], [31] or [14]).

Lemma 4.12. *REC* $= \Delta_1^0$ *and for all* $n \in \mathbb{N}$, $\Delta_n^0 \subsetneq \Sigma_n^0 \subsetneq \Delta_{n+1}^0$ *and* $\Delta_n^0 \subsetneq \Pi_n^0 \subsetneq \Delta_{n+1}^0$. *For all* $n \in \mathbb{N}$ *and all* $A \subset \mathbb{N}$, $A \in \Sigma_n^0 \Leftrightarrow \overline{A} \in \Pi_n^0$.

The arithmetical hierarchy is usually depicted as in Fig. 1, where every arrow denotes a proper inclusion. Schematically, one often writes $\exists$REC for Σ_1^0, $\forall\exists$REC for Π_2^0, etc. All classes are closed under bounded quantification: if $A(n) \Leftrightarrow \exists y < f(n)\, P(n, y)$, with f computable and P is decidable, then A is decidable (and similarly for other classes in the hierarchy). To put it more succinctly: $\forall < \mathcal{P} = \mathcal{P}$ for all classes $\mathcal{P}$ in the arithmetical hierarchy.

Let $\mathcal{P}$ be some class in the arithmetical hierarchy. It is easy to see that if the set A is $\mathcal{P}$-hard, and A is an element of some other class $\mathcal{C}$ in the arithmetical hierarchy, then $\mathcal{P} \subseteq \mathcal{C}$. Consequently, if A is $\mathcal{P}$-*complete*, Lemma 4.12 yields that A is 'essentially' in $\mathcal{P}$, i.e. not lower in the hierarchy.

Hence, to determine if a problem A is essentially in a certain class $\mathcal{P}$, we first show that A can be expressed with a formula of $\mathcal{P}$ (showing that A is in $\mathcal{P}$ or lower). To prove that A is not lower, we then prove that A is $\mathcal{P}$-hard.

Above the arithmetical hierarchy, there is the *analytical hierarchy*, where we also allow quantification over infinite sequences of numbers—equivalently, quantification over higher types such as $\mathbb{N} \to \mathbb{N}$. As variables ranging over infinite sequences we use α, β, etc. An example of an analytical formula is $\forall\alpha(\forall x(\alpha(x) \to^*_R \alpha(x+1)) \to \exists x(\alpha(x) = \alpha(x+1)))$,

stating that the rewrite system is SN. This is a Π^1_1-formula. In Section 5 we will see that we can express SN for finite TRSs with a formula that is much lower in the hierarchy: it is in fact Π^0_2.

The class Π^1_1 is the class of problems A such that $A = \{k | \forall\alpha\, \exists x\, P(k, \alpha, x)\}$, where P decidable. Similarly Σ^1_1 is the class of problems A such that $A = \{k | \exists\alpha\, \forall x\, P(k, \alpha, x)\}$, where P is decidable. For analytical problems we have several kinds of simplification procedures analogous to the ones of Remark 4.11.

Lemma 4.13. *In the analytical hierarchy we have the following ways of simplifying a sequence of quantifiers (where quantifiers without superscripts range over* $\mathbb{N}$*):*

$$\forall^1\forall^1 \mapsto \forall^1 \qquad \forall \mapsto \forall^1 \qquad \exists\forall^1 \mapsto \forall^1\exists \qquad \forall\exists^1 \mapsto \exists^1\forall$$

For the first two simplifications, we have analogous versions for $\exists$. For the proof we refer to the standard literature; here we just give a rough idea. For example, the meaning of the first simplification is that a formula $\forall^1\alpha\forall^1\beta\, \varphi(\alpha, \beta)$ is equivalent to a formula of the form $\forall^1\gamma\, \psi(\gamma)$, with ψ in the same class as φ. This can be observed by taking $\psi(\gamma) := \varphi((\gamma)_1, (\gamma)_2)$, where γ is a variable ranging over $\mathbb{N} \to \mathbb{N}$ and $(-)_1$ and $(-)_2$ are the projection functions lifted to the function space, i.e. $(\gamma)_1 := \lambda n \in \mathbb{N}.(\gamma(n))_1$ and $(\gamma)_2 := \lambda n \in \mathbb{N}.(\gamma(n))_2$. It is not difficult to show that ψ is in the same class as φ and $\forall^1\alpha\forall^1\beta\, \varphi(\alpha, \beta)$ is equivalent to $\forall^1\gamma\varphi((\gamma)_1, (\gamma)_2)$.

From these simplifications one derives that each analytic formula is equivalent to one of the form $Q_n\alpha_n Q_{n-1}\alpha_{n-1} \dots Q_0 x\, P(\alpha_1, \dots, \alpha_n, x, k)$ where P is decidable and $\vec{Q}$ is a sequence of alternating quantifiers.

Definition 4.14. The analytical problems are problems A such that $A = \{k | Q_n\alpha_n Q_{n-1}\alpha_{n-1} \dots Q_0 x\, P(\alpha_1, \dots, \alpha_n, x, k)\}$ with P decidable and $\vec{Q}$ a finite sequence of alternating quantifiers. If $n > 0$ and $Q_n = \exists^1$, then the problem is in the class Σ^1_n. If $n > 0$ and $Q_n = \forall^1$, then it is in the class Π^1_n. $\Delta^1_n := \Sigma^1_n \cap \Pi^1_n$.

For the analytical hierarchy we can draw a similar diagram as the one in Fig. 1: replace Σ^0_1 by Σ^1_1 etc. We have the same results as Lemma 4.12: each class is a proper subclass of the ones above it. The entire arithmetic hierarchy is also a proper subclass of the lowest class, Δ^1_1.

For later use, we note the positions of certain "standard" problems for Turing machines; we refer to [13,31].

Lemma 4.15. *We have the following well-known results:*

(i) *the blank tape halting problem* $\{ \mathsf{M} \mid \mathsf{M}$ *halts on the blank tape* $\}$ *is* Σ^0_1*-complete,*
(ii) *the initialized uniform halting problem* $\{ \mathsf{M} \mid \mathsf{M}$ *halts on all inputs* $\}$ *is* Π^0_2*-complete,*
(iii) *the totality problem* $\{ \mathsf{M} \mid \mathsf{M}$ *halts on* $q_0\mathsf{S}^n$ *for every* $n \in \mathbb{N}\}$ *is* Π^0_2*-complete,*
(iv) *the set* $\mathsf{WF} := \{ \mathsf{M} \mid >_\mathsf{M}$ *is well-founded* $\}$ *is* Π^1_1*-complete.*

These sets will be the basis for the hardness results in the following sections: we will show that $\{ \mathsf{M} \mid \mathsf{M}$ halts on the blank tape $\}$ is many-one reducible to "WN for a single term" and thus conclude that "WN for a single term" is Σ^0_1. This will be done by effectively giving for every Turing machine M, a TRS R_M and a term t_M such that

$$\mathsf{M} \text{ halts on the blank tape} \quad \text{iff} \quad \mathsf{WN}_{R_\mathsf{M}}(t_\mathsf{M}).$$

Similar constructions will be carried out for the other problems that we consider.

5. Strong and weak normalization

We use the translation of Turing machines M to TRSs R_M from [23]; we give the basic details of the translation in the following.

Definition 5.1. For every Turing machine $\mathsf{M} = \langle Q, \Gamma, q_0, \delta\rangle$ we define a TRS R_M as follows. The signature is $\Sigma = Q \cup \Gamma \cup \{\triangleright\}$ where the symbols $q \in Q$ have arity 2, the symbols $f \in \Gamma$ have arity 1 and $\triangleright$ is a constant symbol, which represents an infinite number of blank symbols. The rewrite rules of R_M are:

$$q(x, f(y)) \to q'(f'(x), y) \qquad \text{for every } \delta(q, f) = \langle q', f', R\rangle$$
$$q(g(x), f(y)) \to q'(x, g(f'(y))) \qquad \text{for every } \delta(q, f) = \langle q', f', L\rangle$$

together with four rules for 'extending the tape':

$$q(\triangleright, f(y)) \to q'(\triangleright, \Box(f'(y))) \qquad \text{for every } \delta(q, f) = \langle q', f', L\rangle$$
$$q(x, \triangleright) \to q'(f'(x), \triangleright) \qquad \text{for every } \delta(q, \Box) = \langle q', f', R\rangle$$
$$q(g(x), \triangleright) \to q'(x, g(f'(\triangleright))) \qquad \text{for every } \delta(q, \Box) = \langle q', f', L\rangle$$
$$q(\triangleright, \triangleright) \to q'(\triangleright, \Box(f'(\triangleright))) \qquad \text{for every } \delta(q, \Box) = \langle q', f', L\rangle\,.$$

We introduce a mapping from terms to configurations to make the connection between the M and the TRS R_{M} precise.

Definition 5.2. We define a mapping $\varphi : \mathit{Ter}(\Gamma \cup \{\triangleright\}, \varnothing) \to \Gamma^*$ by:

$$\varphi(\triangleright) := \varepsilon \qquad\qquad \varphi(f(t)) := f\varphi(t)$$

for every $f \in \Gamma$ and $t \in \mathit{Ter}(\Gamma \cup \{\triangleright\}, \varnothing)$. We define the set of (intended) terms:

$$\mathit{Ter}_{\mathsf{M}} := \{q(s,t) \mid q \in Q,\ s,t \in \mathit{Ter}(\Gamma \cup \{\triangleright\}, \varnothing)\}\,.$$

Then we define a map $\Phi : \mathit{Ter}_{\mathsf{M}} \to \mathcal{C}\mathit{onf}_{\mathsf{M}}$ by:

$$\Phi(q(s,t)) := \varphi(s)^{-1} q \varphi(t) \in \mathcal{C}\mathit{onf}_{\mathsf{M}}\,.$$

Here $.^{-1}$ denotes the reverse of a string. For example, if Γ contains (among others) the two symbols 0 and 1, and $s = 1(1(0(\triangleright)))$ and $t = 0(0(1(\triangleright)))$, we have $\Phi(q(s,t)) = 011q001$.

The function Φ is introduced for simulating Turing machines by rewriting, as is expressed in the following lemma of which the proof is straightforward.

Lemma 5.3. *Let* M *be a Turing machine. Then* R_{M} *simulates* M*, that is:*

(i) $\forall c \in \mathcal{C}\mathit{onf}_{\mathsf{M}}.\ \Phi^{-1}(c) \neq \varnothing$,
(ii) *for all terms* $s \in \mathit{Ter}_{\mathsf{M}}$*:* $s \to_{R_{\mathsf{M}}} t$ *implies* $t \in \mathit{Ter}_{\mathsf{M}}$ *and* $\Phi(s) \to_{\mathsf{M}} \Phi(t)$*, and*
(iii) *for all terms* $s \in \mathit{Ter}_{\mathsf{M}}$*: whenever* $\Phi(s) \to_{\mathsf{M}} c$ *then* $\exists t \in \Phi^{-1}(c).\ s \to_{R_{\mathsf{M}}} t$.

The following is an easy corollary.

Corollary 5.4. *For all* $s \in \mathit{Ter}_{\mathsf{M}}$*:* $\mathsf{SN}_{R_{\mathsf{M}}}(s) \iff$ M *halts on* $\Phi(s)$.

Proof. Induction on item (ii) of Lemma 5.3. □

Let us elaborate a bit on Turing machines and the encoding of term rewriting.

Remark 5.5. As discussed in Section 4.1, terms and term rewriting systems can be encoded as natural numbers. Finite rewrite sequences $\sigma : t_1 \to \ldots \to t_n$ can be encoded as lists of terms. Then there is a Turing machine that, given the encoding of a rewrite sequence as input, computes the length of $|\sigma| := n$ of the sequence, every term t_1,...,t_n, in particular the first $\mathit{first}(\sigma) := t_1$ and the last term $\mathit{last}(\sigma) := t_n$. There is a Turing machine that, given the TRS as input, can check whether a natural number n corresponds to a valid rewrite sequence, that is, check $t_i \to t_{i+1}$ for every $i = 1, \ldots, (n-1)$. Furthermore for a given term t and $n \in \mathbb{N}$ it can calculate the set of all reductions of length $\leq n$ admitted by t and thereby check properties like 'all reductions starting from t have length $\leq n$' or 't is a normal form'.

We arrive at our first results.

Theorem 5.6. *The properties* SN *and* WN *for single terms are* Σ^0_1*-complete.*

Proof. For Σ^0_1-hardness we reduce the blank tape halting problem to a termination problem for single terms. Therefore, let M be an arbitrary Turing machine. Then $\mathsf{SN}_{R_{\mathsf{M}}}(q_0(\triangleright,\triangleright))$ if and only if M halts on the blank tape by Corollary 5.4. Moreover observe that R_{M} is orthogonal and non-erasing, thus the SN and WN coincide [35]. Hence both properties SN and WN for single terms are Σ^0_1-hard by Lemma 4.4.

To show that SN is in Σ^0_1, let R be a TRS and $t \in \mathit{Ter}(\Sigma, \mathcal{X})$ a term. Since R is finite, the rewrite relation is finitely branching. So by Lemma 2.2 the following formula holds:

$$\mathsf{SN}_R(t) \iff \exists n \in \mathbb{N}.\ \text{all reductions starting from } t \text{ have length} \leq n.$$

Thus we have one existential number quantifier and by Remark 5.5 the predicate following the quantifier is recursive. Hence SN for single terms is Σ^0_1-complete.

To show that WN is in Σ^0_1, let R be a TRS and $t \in \mathit{Ter}(\Sigma, \mathcal{X})$ a term. The term t is WN if there exists a reduction to a normal form:

$$\mathsf{WN}_R(t) \iff \exists u, s.\ (t \overset{u}{\to}{}^*_R s \wedge s \text{ is a normal form}).$$

This is a Σ^0_1-formula, hence WN for single terms is Σ^0_1-complete. □

For showing Π^0_2-completeness of the uniform properties SN and WN we would like to use the equivalence "$\mathsf{SN}(R_\mathsf{M}) \iff$ M halts on all inputs", in combination with Lemma 4.15 (ii). However, this does not work because of the following two problems:

(1) In R_M we have terms of the form $q(w, v)$, where q is not the start state and wv is some arbitrary (finite) tape content. That M halts on all inputs, does not guarantee that M halts when started in configuration $\langle q, wv\rangle$.
(2) In R_M we have terms that do not correspond to a configuration at al, for instance terms of the form $q(q(w, v), u)$.

To deal with problem (2), we can use a technique [35,39]. This technique states that by introducing sorts in an unsorted TRS, termination on all terms is equivalent to termination of well-sorted terms. As this introduction of sorts is a kind of typing, it is called *type introduction*. This equivalence holds for TRSs that are non-collapsing or non-duplicating; here it applies since R_M satisfies both. Thus, the goal is to assign sorts in such a way that well-sorted terms correspond to proper configurations. We assign sort $s_0 \to s_0$ to every $f \in \Gamma$, sort s_0 to $\triangleright$ and sort $s_0 \times s_0 \to s_1$ to every $q \in Q$. The terms of sort s_0 are normal forms. The (non-variable) terms of sort s_1 are in Ter_M after replacing all variables by $\triangleright$, and by Corollary 5.4 for all terms $t \in \mathit{Ter}_\mathsf{M}$ we have $\mathsf{SN}_{R_\mathsf{M}}(t)$ if and only if M halts on $\Phi(t)$. Hence $\mathsf{SN}_{R_\mathsf{M}}$ holds if and only if M halts on all configurations $\mathcal{C}\mathit{onf}_\mathsf{M}$.

We now need to deal with problem (1); we would like that M halts on all configurations $\mathcal{C}\mathit{onf}_\mathsf{M}$ if and only if M halts on all inputs, starting from the initial state, but unfortunately this does not hold. We need a lemma about Turing machines; we use the following result from [13].

Lemma 5.7 ([13])**.** *For every Turing machine* M *that computes a function* $f : \mathbb{N} \to \mathbb{N}$ *we can effectively construct a Turing machine* $\widehat{\mathsf{M}}$ *such that*

(i) $\widehat{\mathsf{M}}$ *also computes* f.
(ii) M *halts on all configurations if and only if* f *is total.*

So, if M halts on all inputs (when started in the initial state), then $\widehat{\mathsf{M}}$ halts on all configurations. This solves problem (1) and we have the following Corollary, which follows from the fact that the initialized uniform halting problem (set (ii) in Lemma 4.15) many-one reduces to the uniform halting problem (the set in the Corollary), using Lemma 4.4. Basically, this corollary has already been stated and proved in [13].

Corollary 5.8. *The uniform halting problem*

$$\{\ \mathsf{M} \mid \mathsf{M} \textit{ halts on all configurations } \langle q, \mathit{tape}\rangle \in \mathcal{C}\mathit{onf}_\mathsf{M}\ \}$$

is Π^0_2*-complete.*

Theorem 5.9. *The properties uniform* SN *and* WN *are* Π^0_2*-complete.*

Proof. For Π^0_2-hardness: we have seen how the uniform halting problem for M many-one reduces to the uniform termination problem for R_M. Since R_M is orthogonal and non-erasing SN and WN coincide [35]. Hence SN and WN are both Π^0_2-hard by Lemma 4.4. That the uniform properties SN and WN are in Π^0_2 follows from the fact that these properties for single terms can be described by Σ^0_1-formulas and the uniform property 'adds' a universal number quantifier. □

6. Reduce, join and convert

For two terms t, u and a rewrite relation $\to$ we consider the following three basic properties:

- *reduction*: $t \to^* u$,
- *joinability*: $t \to^* \cdot \leftarrow^* u$, and
- *conversion*: $t \leftrightarrow^* u$.

We will prove that all of these properties are Σ^0_1-complete. For proving Σ^0_1-hardness, we extend the TRS R_M from Definition 5.1 by extra rules by which a fresh constant T can only reached from terms representing halting Turing machine configurations.

Definition 6.1. For an arbitrary Turing machine M we define the TRS H_M to consist of the rules

$$q(x, f(y)) \to \mathsf{T} \qquad \text{for every } f \in \Gamma,\ q \in Q \text{ with } \delta(q, f) \text{ is undefined}$$
$$q(x, \triangleright) \to \mathsf{T} \qquad \text{for every } q \in Q \text{ with } \delta(q, \square) \text{ is undefined.}$$

where T is a fresh constant.

Lemma 6.2. *Let* M *be any Turing machine and let* $\to$ *be the rewrite relation with respect to* $R_{\mathsf{M}} \cup H_{\mathsf{M}}$*, as defined in Definitions 5.1 and 6.1. Then the following properties are equivalent:*

(i) M *halts on the blank tape,*
(ii) $q_0(\triangleright, \triangleright) \to^* \mathsf{T}$,
(iii) $q_0(\triangleright, \triangleright) \to^* \cdot \leftarrow^* \mathsf{T}$,
(iv) $q_0(\triangleright, \triangleright) \leftrightarrow^* \mathsf{T}$.

Proof. (i)⇒(ii): Let $\Phi(q_0(\triangleright, \triangleright)) \to_{\mathsf{M}}^* \Phi(s)$ be the halting computation on the blank tape. Then $q_0(\triangleright, \triangleright) \to^* s$ according to Lemma 5.3, and $s \to_{H_{\mathsf{M}}} \mathsf{T}$ according to the definition of H_{M}.
(ii)⇒(iii): trivial.
(iii)⇒(iv): trivial.
(iv)⇒(i): Assume $q_0(\triangleright, \triangleright) \leftrightarrow^* \mathsf{T}$. Take such a conversion of minimal length. The last step in this conversion is of the shape $s \to_{H_{\mathsf{M}}} \mathsf{T}$ where $\Phi(s)$ is a halting configuration. Due to minimality the conversion $q_0(\triangleright, \triangleright) \leftrightarrow^* s$ does not involve T and is completely in R_{M}. From Lemma 5.3 we now conclude that the initial blank tape configuration is $\to_{\mathsf{M}}$-convertible with a halting configuration. So it remains to show that a configuration is halting if and only if it is $\to_{\mathsf{M}}$-convertible with a halting configuration. This follows by induction on the length of the conversion from the observation that if $c \to_{\mathsf{M}} d$ then c is halting if and only if d is halting, which follows from the fact that the Turing machine is deterministic. □

Theorem 6.3. *Given a TRS R with rewrite relation* $\to$*, and two terms t, u, all of the following three properties are* Σ_1^0*-complete:*

- *reduction:* $t \to^* u$,
- *joinability:* $t \to^* \cdot \leftarrow^* u$, *and*
- *conversion:* $t \leftrightarrow^* u$.

Proof. As all reductions, joins and conversions can be enumerated, all of these properties, being of the shape $\exists \cdots$, are in Σ_1^0. For the first one this is immediate from Lemma 4.8 (iii), the rest is similar.
From Lemma 6.2 we conclude Σ_1^0-hardness, and hence Σ_1^0-completeness, of all three basic properties. □

With the same argument as in Lemma 6.2, but now for an arbitrary configuration rather than only the initial configuration, we obtain the following lemma, which will be used later for results on confluence.

Lemma 6.4. *Let* M *be any Turing machine and let* $\to$ *be the rewrite relation with respect to* $R_{\mathsf{M}} \cup H_{\mathsf{M}}$*, as defined in Definitions 5.1 and 6.1. Then for all configurations* $c \in \mathcal{C}onf_{\mathsf{M}}$ *and terms* $t \in \Phi^{-1}(c)$ *we have that* M *halts on c if and only if* $t \to^* \mathsf{T}$.

7. Unique normal forms

Recall that a TRS R

- has the *unique normal form property on t*, denoted $\mathsf{UN}_R(t)$, if there is at most one normal form n satisfying $t \leftrightarrow^* n$,
- has the *unique normal form property on t with respect to reduction*, denoted $\mathsf{UN}_R^{\to}(t)$, if there is at most one normal form n satisfying $t \to^* n$.

In the literature these notions are mainly considered in their uniform variants, but as they admit the single term versions, we include those as well. In this section we prove that all of these properties are Π_1^0-complete. In order to do so we take $R_{\mathsf{M}} \cup H_{\mathsf{M}}$ and add a few extra rules introducing an extra normal form U and forcing that T and U are the only normal forms. More precisely, for a Turing machine M we define the TRS S_{M} to consist of $R_{\mathsf{M}} \cup H_{\mathsf{M}}$ from Definitions 5.1 and 6.1, extended by the rules

$$\begin{aligned} q_0(\triangleright, \triangleright) &\to \mathsf{U} \\ q(x, y) &\to q(x, y) \quad \text{for all states } q \in Q \\ f(x) &\to f(x) \quad \text{for all } f \in \Gamma \\ \triangleright &\to \triangleright \end{aligned}$$

Lemma 7.1. *The following properties are equivalent*

(i) M *does not halt on the blank tape,*
(ii) $\mathsf{UN}_{S_{\mathsf{M}}}^{\to}(q_0(\triangleright, \triangleright))$,
(iii) $\mathsf{UN}_{S_{\mathsf{M}}}(q_0(\triangleright, \triangleright))$,

(iv) $\mathsf{UN}^{\rightarrow}_{S_\mathsf{M}}(t)$ *for all terms* t,
(v) $\mathsf{UN}_{S_\mathsf{M}}(t)$ *for all terms* t.

Proof. Write $\to$ for $\to_{S_\mathsf{M}}$.
(v)⇒(iii)⇒*(ii)*: trivial.
(v)⇒(iv)⇒*(ii)*: trivial.
(ii)⇒(i): Assume M halts on the blank tape. Then according to Lemma 6.2 we have $q_0(\triangleright,\triangleright) \to^* \mathsf{T}$. Moreover, we have $q_0(\triangleright,\triangleright) \to^* \mathsf{U}$. Since T and U are normal forms, this contradicts $\mathsf{UN}^{\rightarrow}_{S_\mathsf{M}}(q_0(\triangleright,\triangleright))$.
(i)⇒(v): Assume *(v)* does not hold. Then there are two distinct normal forms that are convertible. Due to the addition of the rules $q(x,y) \to q(x,y), f(x) \to f(x)$ and $\triangleright \to \triangleright$ the only normal forms are T, U and the variables. As there are no collapsing rules, variables are not convertible to other terms, and the only possible two distinct normal forms are T and U. Take a conversion from T to U of minimal length. Then this is of the shape

$$\mathsf{T} \leftrightarrow^* q_0(\triangleright,\triangleright) \to \mathsf{U},$$

in which U is not involved in the conversion $\mathsf{T} \leftrightarrow^* q_0(\triangleright,\triangleright)$. Moreover, again due to minimality in this conversion no terms are rewritten to itself. Hence this conversion only uses rules from $R_\mathsf{M} \cup H_\mathsf{M}$. Now by Lemma 6.2 we conclude that M halts on the blank tape, contradicting (i). □

Theorem 7.2. *The properties* UN *and* $\mathsf{UN}^{\rightarrow}$ *are* Π^0_1*-complete, both uniform and for single terms.*

Proof. From Lemma 7.1 we conclude Π^0_1-hardness for all four properties.
It remains to prove that all four properties are in Π^0_1. The property $\mathsf{UN}^{\rightarrow}_R(t)$ can be expressed as $\forall\vec{u}, \vec{v} P(\vec{u},\vec{v})$ for $P(\vec{u},\vec{v})$ being the conjunction of

- $\vec{u} = (u_1,\ldots,u_n)$, where $u_1 = t$ and $u_i \to_R u_{i+1}$ for $i = 1,\ldots,n-1$,
- $\vec{v} = (v_1,\ldots,v_m)$, where $v_1 = t$ and $v_i \to_R v_{i+1}$ for $i = 1,\ldots,m-1$,
- if both u_n and v_m are normal forms, then $u_n = v_m$.

As all ingredients of this characterization are decidable by Lemma 4.8, this proves that single term $\mathsf{UN}^{\rightarrow}$ is Π^0_1. For single term UN a similar argument is given by replacing $\to_R$ in $u_i \to_R u_{i+1}$ and $v_i \to_R v_{i+1}$ by $\leftrightarrow_R$. The uniform versions are obtained by replacing the requirements $u_1 = t$ and $v_1 = t$ by $u_1 = v_1$. □

In this theorem, we consider the uniform properties on the set of all terms, not just ground terms. Also for the set of ground terms Π^0_1-completeness of UN and $\mathsf{UN}^{\rightarrow}$ holds, as in Lemma 7.1 we proved equivalence for a single ground term and all terms, and the set of all ground terms is in between.

8. Confluence and ground confluence

We investigate the complexity of confluence (CR) and ground confluence (grCR), both uniform and for single terms.
For proving Π^0_2-completeness of confluence it is natural to try to use an extension of $R_\mathsf{M} \cup H_\mathsf{M}$ from Definitions 5.1 and 6.1 augmented with the following rules:

$$\mathsf{run}(x,y) \to \mathsf{T}$$
$$\mathsf{run}(x,y) \to q_0(x,y).$$

At first glance it appears that $q_0(s,t) \to^* \mathsf{T}$ if the Turing machine M halts on all configurations (using rules from $R_\mathsf{M} \cup H_\mathsf{M}$ by Lemma 6.4), but a problem arises if s and t contain variables, as the resulting term may not represent a configuration. We solve the problem as follows.
For Turing machines M we define the TRS S_M to consist of the rules of the TRS $R_\mathsf{M} \cup H_\mathsf{M}$ from Definitions 5.1 and 6.1 extended by the rules:

$$\mathsf{run}(x,\triangleright) \to \mathsf{T} \tag{1}$$
$$\mathsf{run}(\triangleright,y) \to q_0(\triangleright,y) \tag{2}$$
$$\mathsf{run}(x,\mathsf{S}(y)) \to \mathsf{run}(\mathsf{S}(x),y) \tag{3}$$
$$\mathsf{run}(\mathsf{S}(x),y) \to \mathsf{run}(x,\mathsf{S}(y)). \tag{4}$$

It is then easy to see that T and $q_0(\triangleright,s)$ are convertible using the rules (1)–(4) if and only if s is a ground term of the form $\mathsf{S}^n(\triangleright)$.

Theorem 8.1. *Uniform confluence (*CR*), and uniform ground confluence (*grCR*) are Π^0_2-complete.*

Proof. For proving Π^0_2-hardness we reduce the totality problem to confluence. Let M be an arbitrary Turing machine. We consider the TRS S_M defined above. By [1], confluence is a *persistent* property, and we may thus reduce confluence of S_M to confluence of a corresponding many-sorted TRSs. We assign sort $s_0 \to s_0$ to all symbols from Γ, sort s_0 to $\triangleright$, sort $s_0 \times s_0 \to s_1$ to every symbol in $\{\mathsf{run}\} \cup Q$, and sort s_1 to T. Then, by [1], the obtained many-sorted TRS is confluent if and only if S_M is confluent. Observe first that the terms of sort s_0 are normal forms. For terms of s_1 with root symbol $\neq$ 'run' the reduction exhibits no branching as $R_\mathsf{M} \cup H_\mathsf{M}$ is orthogonal and all redexes occur at the root as all subterms are of sort s_0. For terms of sort s_1 with root symbol 'run' we observe that (3) and (4) are inverse to each other. Hence, whenever $s \to^*_{(3)\cup(4)} t$ then also $s \to^*_{(3)} t$ or $s \to^*_{(4)} t$.

Therefore it suffices to consider the case

$$s_2 \leftarrow_{(2)} s_1 \leftarrow^*_{(3)} \mathsf{run}(t_1, t_2) \to^*_{(4)} s_3 \to_{(1)} \mathsf{T}$$

where $t_1, t_2 \in \mathit{Ter}(\Gamma \cup \{\triangleright\}, \mathcal{X})$. From the existence of such reductions we conclude that there exists $n \in \mathbb{N}$ such that $s_1 \equiv \mathsf{run}(\triangleright, \mathsf{S}^n(\triangleright))$, $s_3 \equiv \mathsf{run}(\mathsf{S}^n(\triangleright), \triangleright)$, and $s_2 \equiv q_0(\triangleright, \mathsf{S}^n(\triangleright))$. Conversely, for every $n \in \mathbb{N}$ such reductions exist. As a consequence the TRS S is confluent if and only if $q_0(\triangleright, \mathsf{S}^n(\triangleright)) \to^*_S \mathsf{T}$ for every $n \in \mathbb{N}$, that is, by Lemma 6.4 if and only if M halts on $q_0\mathsf{S}^n$ for every $n \in \mathbb{N}$. Moreover, the only terms containing critical pairs are on the form $\mathsf{run}(t_1, t_2)$ where t_1 and t_2 are ground terms; hence, ground confluence coincides with confluence for S, and we have thus proved Π^0_2-hardness.

To show that both properties are in Π^0_2 let R be a TRS. Then R is confluent if and only if the following holds:

$$\begin{aligned}\mathsf{CR}_R \iff \forall s.\, \forall u, v.\, \forall t_1, t_2.\, (s \overset{u}{\to}{}^*_R t_1 \wedge s \overset{v}{\to}{}^*_R t_2 \\ \Rightarrow \exists r, q.\, \exists t_3.\, (t_1 \overset{r}{\to}{}^*_R t_3 \wedge t_2 \overset{q}{\to}{}^*_R t_3))\end{aligned}$$

By quantifier compression we can simplify the formula such that there is only a single universal quantifier followed by a single existential quantifier. The corresponding formula for grCR is almost the same, as the quantification of t over (encodings of) all terms is merely replaced by a quantification over all ground terms. Therefore both the uniform properties CR and grCR are Π^0_2-complete. □

Theorem 8.2. *Confluence (*CR*), and ground confluence (*grCR*) for single terms are Π^0_2-complete.*

Proof. For Π^0_2-hardness we use the totality problem. Let M be an arbitrary Turing machine. We define the TRS S as $R_\mathsf{M} \cup H_\mathsf{M}$ from Definitions 5.1 and 6.1 extended by:

$$\begin{aligned}\mathsf{run}(x) &\to \mathsf{T}\\ \mathsf{run}(x) &\to \mathsf{run}(\mathsf{S}(x))\\ \mathsf{run}(x) &\to q_0(\triangleright, x).\end{aligned}$$

The term $t := \mathsf{run}(\triangleright)$ rewrites to T and $q_0(\triangleright, \mathsf{S}^n(x))$ for every $n \in \mathbb{N}$. Furthermore we have $q_0(\triangleright, \mathsf{S}^n(\triangleright)) \to^*_S \mathsf{T}$ if and only if M halts on $q_0\mathsf{S}^n$ by Lemma 6.4. Consequently, CR and grCR for single terms are Π^0_2-hard.

For Π^0_2-completeness note that we can formalize CR and grCR for single terms simply by dropping the universal quantification over all terms ($\forall t \in \mathbb{N}$) from the respective Π^0_2-formulas for the uniform properties in the proof of Theorem 8.1. □

9. Local confluence and local ground confluence

We now investigate the complexity of local confluence (WCR) and local ground confluence (grWCR) both uniform and for single terms.

Theorem 9.1. *The properties local confluence (*CR*) both for single terms and uniform, and local ground confluence (*grCR*) for single terms are Σ^0_1-complete.*

Proof. For Σ^0_1-hardness we use the blank tape halting problem. Let M be an arbitrary Turing machine, and let the TRS S consist of $R_\mathsf{M} \cup H_\mathsf{M}$ from Definitions 5.1 and 6.1, extended by the following rules:

$$\begin{aligned}\mathsf{run} &\to \mathsf{T}\\ \mathsf{run} &\to q_0(\triangleright, \triangleright).\end{aligned}$$

The only critical pair is $\mathsf{T} \leftarrow \mathsf{run} \rightarrow q_0(\triangleright, \triangleright)$. By the Critical Pairs Lemma [35], WCR holds if and only if all critical pairs can be joined. As a consequence, for this system uniform WCR coincides with $\mathsf{WCR}(t)$ and $\mathsf{grWCR}(t)$ for the single term $t := \mathsf{run}$. Moreover, we have $q_0(\triangleright, \triangleright) \to_S^* \mathsf{T}$ if and only if M halts on the blank tape by Lemma 6.2. Hence, $\mathsf{WCR}(t)$, $\mathsf{grWCR}(t)$ and WCR hold if and only if M halts on the blank tape, and consequently all these properties are Σ_1^0-hard.

Let (R, t) encode finite TRS R and term t. There is a Turing machine that on inputs (R, t) computes all (finitely many) critical pairs, and all (finitely many) one-step reducts of t. For inclusion of all three problems in Σ_1^0, it thus suffices to show that the following problem is in Σ_1^0: Decide, on input (i) a finite TRS S, (ii) $n \in \mathbb{N}$, and (iii) terms $t_1, s_1, \ldots, t_n, s_n$, whether for every $i = 1, \ldots, n$ the terms t_i and s_i have a common reduct. This property can clearly be described by the following Σ_1^0-formula:

$$\exists \vec{u}, \vec{v}, \vec{p}.\ (s_1 \to_R^{u_1 *} p_1 \leftarrow_R^{v_1 *} t_1 \wedge \cdots \wedge s_n \to_R^{u_n *} p_n \leftarrow_R^{v_n *} t_n) \qquad \square$$

Surprisingly it turns out that uniform local *ground* confluence is Σ_2^0-complete and thereby strictly harder than uniform local confluence (on the set of all open terms).

Theorem 9.2. *Uniform local ground confluence (*grCR*) is* Π_2^0*-complete.*

Proof. For Π_2^0-hardness we use reduction from the uniform halting problem. Let M be a Turing machine. We define the TRS S as extension of R_{M} with:

$$\begin{aligned} &\mathsf{run}(x, y) \to \mathsf{T} \\ &\mathsf{run}(x, y) \to q_0(x, y) \\ &q(f(\vec{x}), g(\vec{y})) \to \mathsf{T} \qquad \text{for all combinations of symbols } f, g \text{ such that} \\ &\qquad\qquad\qquad\qquad\qquad\qquad \text{the left-hand side is not matched by a rule in } R_{\mathsf{M}} \end{aligned}$$

where $\vec{x}$ and $\vec{y}$ are vectors of distinct variables chosen such that the left-hand sides of all the above rules are left-linear.

Assume there exists a configuration c on which M does not halt. Then by Lemma 5.3 there exists $q(s, t) \in \Phi^{-1}(c)$, and by Corollary 5.4 R_{M} is not terminating on $q(s, t)$. Every reduct of $q(s, t)$ is an R_{M}-redex and contains no further redexes. In particular, none of the rules above are applicable to any reduct. Hence, $q(s, t) \not\to^* \mathsf{T}$ and thus $\mathsf{T} \leftarrow \mathsf{run}(s, t) \to q(s, t)$ is not joinable.

Conversely, assume that M halts on all configurations. Let $D = \{\mathsf{run}\} \cup Q$, and let V be the set of ground terms having a root symbol in D. Observe that all symbols *not* in D are constructor symbols; for local ground confluence, it thus suffices to show that every reduct of a term in V rewrites to T. Every term from V is a redex, and all reducts of terms in V are in $V \cup \{\mathsf{T}\}$. Thus it suffices to show that no term in V admits an infinite root rewrite sequence.

Claim. There is an infinite root rewrite sequence starting from a term in V iff if a ground term on the form $q(s, t)$ admits an infinite R_{M}-root rewrite sequence.

Proof of claim. None of the rules added above can contribute essentially to such a sequence, and can be omitted. Furthermore, the rules above can only replace a subterm by T, or a term on the form $q_0(x, y)$. As the rules of R_{M} are non-duplicating, and as no pattern of any rule contains the symbol T, only a fixed, finite number of contractions of the rules added above can occur in an infinite reduction. Hence, there exists an infinite R_{M}-root rewrite sequence. Furthermore, observe that, below the root (which is in Q), the left-hand sides of rules from R_{M} match only symbols from $\Gamma \cup \{\triangleright\}$. Let s' and t' be ground terms obtained from s and t respectively by replacing all subterms having a root symbol not in $\Gamma \cup \{\triangleright\}$ with $\triangleright$. Then $q(s', t')$ admits an infinite R_{M}-rewrite sequence, $s', t' \in \mathit{Ter}(\Sigma \cup \{\triangleright\}, \varnothing)$, and $q(s', t') \in \mathit{Ter}_{\mathsf{M}}$. (End of proof of claim).

Consequently, $\Phi(q(s', t'))$ is a non-terminating configuration of M by Corollary 5.4, contradicting the assumption that M halts on all configurations. $\square$

10. Dependency pair problems

In this section we present the remarkable result that finiteness of dependency pair problems, although invented for proving termination, is of a much higher level of complexity than termination itself: Tt is Π_1^1-complete, both for the uniform property and for the property for single terms. This only holds for the basic version of dependency pairs; for the version with minimality flag we will show it is of the same level as termination itself.

For relations $\to_1, \to_2$ we write $\to_1 \,/\, \to_2 = \to_2^* \cdot \to_1$. For TRSs R, S instead of $\mathsf{SN}(\to_{R,\epsilon} \,/\, \to_S)$ we shortly write $\mathsf{SN}(R_{\mathsf{top}}/S)$; in the literature [12] this is called *finiteness of the dependency pair problem* $\{R, S\}$. So $\mathsf{SN}(R_{\mathsf{top}}/S)$ means that every infinite $\to_{R,\epsilon} \cup \to_S$ reduction contains only finitely many $\to_{R,\epsilon}$ steps. The motivation for studying this comes from the dependency pair approach [2] for proving termination: for any TRS R we can easily define a TRS $\mathsf{DP}(R)$ such that we have

$$\mathsf{SN}(\mathsf{DP}(R)_{\mathsf{top}}/R) \iff \mathsf{SN}(R).$$

The main result of this section is Π^1_1-completeness of $\mathsf{SN}(R_{\mathsf{top}}/S)$, even of $\mathsf{SN}(S_{\mathsf{top}}/S)$, for both the uniform and the single term variant. In the subsequent section we will consider the variant $\mathsf{SN}(R_{\mathsf{top}/\mathsf{min}}\, S)$ with minimality flag which only makes sense for the uniform variant, and show that it behaves as normal termination: it is Π^0_2-complete.

For proving Π^1_1-hardness of $\mathsf{SN}(S_{\mathsf{top}}/S)$ we now adapt the translation of Turing machines to TRSs given in Definition 5.1. The crucial difference is that every step of the Turing machine 'produces' one output pebble '$\bullet$'. Thereby we achieve that the resulting TRS $R^{\bullet}_{\mathsf{M}}$ admits only finitely many steps at root level even if M does not terminate.

Definition 10.1. For every Turing machine $\mathsf{M} = \langle Q, \Gamma, q_0, \delta\rangle$ we define the TRS $R^{\bullet}_{\mathsf{M}}$ as follows. The signature Σ is $\Sigma = Q \cup \Gamma \cup \{\triangleright, \bullet, \mathsf{T}\}$ where $\bullet$ is a unary symbol, T is a constant symbol, and the rewrite rules of $R^{\bullet}_{\mathsf{M}}$ are:

$$\ell \to \bullet(r) \qquad \text{for every } \ell \to r \in R_{\mathsf{M}}$$

and rules for rewriting to T after successful termination:

$$q(x, 0(y)) \to \mathsf{T} \qquad \text{whenever } \delta(q, \mathsf{S}) \text{ is undefined}$$
$$\bullet(\mathsf{T}) \to \mathsf{T}\,.$$

Then we obtain the following lemma. (Recall the Definition of $>_{\mathsf{M}}$ in 3.3.)

Lemma 10.2. *For every Turing machine* $\mathsf{M} = \langle Q, \Gamma, q_0, \delta\rangle$ *and* $n, m \in \mathbb{N}$ *we have* $n >_{\mathsf{M}} m$ *if and only if* $q_0(\mathsf{S}^n, \mathsf{S}^m) \to^*_{R^{\bullet}_{\mathsf{M}}} \mathsf{T}$.

Moreover we define an auxiliary TRS R_{pickn} for generating a random natural number $n \in \mathbb{N}$ in the shape of a term $\mathsf{S}^n(0(\triangleright))$:

Definition 10.3. We define the TRS R_{pickn} to consist of the following three rules:

$$\mathsf{pickn} \to \mathsf{c}(\mathsf{pickn}) \qquad \mathsf{pickn} \to \mathsf{ok}(0(\triangleright)) \qquad \mathsf{c}(\mathsf{ok}(x)) \to \mathsf{ok}(\mathsf{S}(x))\,.$$

The following lemma is straightforward.

Lemma 10.4. *The TRS* R_{pickn} *has the following properties:*

- $\mathsf{pickn} \to^* \mathsf{ok}(\mathsf{S}^n(0(\triangleright)))$ *for every* $n \in \mathbb{N}$, *and*
- *whenever* $\mathsf{pickn} \to^* \mathsf{ok}(t)$ *for some term* t *then* $t \equiv \mathsf{S}^n(0(\triangleright))$ *for some* $n \in \mathbb{N}$.

We are now ready to prove Π^1_1-completeness of dependency pair problems.

Theorem 10.5. *Both* $\mathsf{SN}(t, R_{\mathsf{top}}/S)$ *and* $\mathsf{SN}(R_{\mathsf{top}}/S)$ *are* Π^1_1*-complete.*

Proof. We prove Π^1_1-hardness even for the case where R and S coincide. We do this by using that the set checking well-foundedness of $>_{\mathsf{M}}$ Π^1_1-complete. Let M be an arbitrary Turing machine. From M we construct a TRS S together with a term t such that:

$$\mathsf{SN}(S_{\mathsf{top}}/S) \iff \mathsf{SN}(t, S_{\mathsf{top}}/S) \iff >_{\mathsf{M}} \text{ is well-founded.}$$

Let S consist of the rules of $R^{\bullet}_{\mathsf{M}} \uplus R_{\mathsf{pickn}}$ together with:

$$\mathsf{run}(\mathsf{T}, \mathsf{ok}(x), \mathsf{ok}(y)) \to \mathsf{run}(q_0(x, y), \mathsf{ok}(y), \mathsf{pickn}), \tag{5}$$

and define $t := \mathsf{run}(\mathsf{T}, \mathsf{pickn}, \mathsf{pickn})$.

As the implication from the first to the second item is trivial, we only have to prove (1) $\mathsf{SN}(t, S_{\mathsf{top}}/S) \iff >_{\mathsf{M}}$ is well-founded and (2) $>_{\mathsf{M}}$ is well-founded $\iff \mathsf{SN}(S_{\mathsf{top}}/S)$.

(1) Suppose $\mathsf{SN}(t, S_{\mathsf{top}}/S)$ and assume there is an infinite descending $>_{\mathsf{M}}$-sequence: $n_1 >_{\mathsf{M}} n_2 >_{\mathsf{M}} \ldots$. Then we have:

$$\begin{aligned}
\mathsf{run}(\mathsf{T}, \mathsf{pickn}, \mathsf{pickn}) &\to^* \mathsf{run}(\mathsf{T}, \mathsf{ok}(\mathsf{S}^{n_1}(0(\triangleright))), \mathsf{ok}(\mathsf{S}^{n_2}(0(\triangleright)))) \qquad (*)\\
&\to_{S,\epsilon} \mathsf{run}(q_0(\mathsf{S}^{n_1}(0(\triangleright)), \mathsf{S}^{n_2}(0(\triangleright))), \mathsf{ok}(\mathsf{S}^{n_2}(0(\triangleright))), \mathsf{pickn})\\
&\to^* \mathsf{run}(\mathsf{T}, \mathsf{ok}(\mathsf{S}^{n_2}(0(\triangleright))), \mathsf{ok}(\mathsf{S}^{n_3}(0(\triangleright))))\\
&\to_{S,\epsilon} \cdots
\end{aligned}$$

Note that $q_0(\mathsf{S}^{n_i}(0(\triangleright)), \mathsf{S}^{n_{i+1}}(0(\triangleright))) \to^* \mathsf{T}$ (for all $i \geq 1$) because M computes the binary predicate $>_\mathsf{M}$. So we have an infinite reduction starting from t, contradicting $\mathsf{SN}(t, S_{\mathsf{top}}/S)$. So there is no infinite descending $>_\mathsf{M}$-sequence.

(2) Suppose that $>_\mathsf{M}$ is well-founded and assume that σ is a rewrite sequence containing infinitely many root steps. Note that (5) is the only candidate for a rule which can be applied infinitely often at the root. Hence all terms in σ have the root symbol run. We consider the first three applications of (5) at the root in σ After the first application the third argument of run is pickn. Therefore after the second application the second argument of run is a reduct of pickn and the third is pickn. Then before the third application we obtained a term t whose first argument is T, and the second and the third argument are reducts of pickn. Observe from t on the rewrite sequence σ must be of the form as depicted above ($*$) (c.f. Lemma 10.4) for some $n_1, n_2, \ldots \in \mathbb{N}$. Then for all $i \geq 1$: $n_i >_\mathsf{M} n_{i+1}$ since $q_0(\mathsf{S}^{n_i}(0(\triangleright)), \mathsf{S}^{n_{i+1}}(0(\triangleright))) \to^* \mathsf{T}$. This contradicts well-foundedness of $>_\mathsf{M}$.

It remains to prove that both $\mathsf{SN}(R_{\mathsf{top}}/S)$ and $\mathsf{SN}(t, R_{\mathsf{top}}/S)$ are in Π^1_1. Let R and S be TRSs. Then $\mathsf{SN}(R_{\mathsf{top}}/S)$ holds if and only if all $\to_{R,\epsilon} \cup \to_S$ reductions contain only a finite number of $\to_{R,\epsilon}$ steps. An infinite reduction can be encoded as a function $\alpha : \mathbb{N} \to \mathbb{N}$ where $\alpha(n)$ is the n-th term of the sequence. We can express the property as follows:

$$\begin{aligned}\mathsf{SN}(R_{\mathsf{top}}/S) \iff \forall \alpha : \mathbb{N} \to \mathbb{N}.\ & \\ ((\forall n \in \mathbb{N}.\ \alpha(n) \text{ rewrites to } \alpha(n+1) \text{ via } \to_{R,\epsilon} \cup \to_S) \Rightarrow & \\ \exists m_0 \in \mathbb{N}.\ \forall m \geq m_0.\ \neg(\alpha(m) \text{ rewrites to } \alpha(m+1) \text{ via } \to_{R,\epsilon})),&\end{aligned}$$

containing one universal function quantifier in front of an arithmetic formula. Here the predicate 'n rewrites to m' tacitly includes a check that both n and m indeed encode terms. For the property $\mathsf{SN}(t, R_{\mathsf{top}}/S)$ we simply add the condition $t = \alpha(1)$ to restrict the quantification to such rewrite sequences α that start with t. Hence $\mathsf{SN}(R_{\mathsf{top}}/S)$ and $\mathsf{SN}(t, R_{\mathsf{top}}/S)$ are Π^1_1-complete. □

10.1. Infinitary rewriting

We now sketch how the proof of Theorem 10.5 also implies Π^1_1-completeness of the property SN^∞ in infinitary rewriting. For its definition and basic observations see [24]. Since in Theorem 10.5 we proved Π^1_1-hardness even for the case where R and S coincide, we conclude that $\mathsf{SN}(S_{\mathsf{top}}/S)$ is Π^1_1-complete. This property $\mathsf{SN}(S_{\mathsf{top}}/S)$ states that every infinite S-reduction contains only finitely many root steps. This is the same as the property SN^ω when restricting to finite terms; for the definition of SN^ω see [41] (basically, it states that in any infinite reduction the position of the contracted redex moves to infinity). However, when extending to infinite terms it still holds that for the TRS S in the proof of Theorem 10.5 the only infinite S-reduction containing infinitely many root steps is of the shape given in that proof, only consisting of finite terms. So SN^ω for all terms (finite and infinite) is Π^1_1-complete. It is well-known that for left-linear TRSs the properties SN^ω and SN^∞ coincide, see e.g. [41]. Since the TRS S used in the proof of Theorem 10.5 is left-linear we conclude that the property SN^∞ for left-linear TRSs is Π^1_1-complete.

11. Dependency pair problems with minimality flag

A variant in the dependency pair approach is the dependency pair problem with minimality flag. Here all terms in the infinite $\to_{R,\epsilon} \cup \to_S$-reductions are assumed to be S-terminating. This can be defined as follows. For relations $\to_1, \to_2$ we write

$$\to_1 /_{\mathsf{min}} \to_2 = (\to_2^* \cdot \to_1) \cap \to_{\mathsf{SN}(\to_2)},$$

where the relation $\to_{\mathsf{SN}(\to_2)}$ is defined to consist of all pairs (x, y) for which x is $\to_2$-terminating. For TRSs R, S instead of $\mathsf{SN}(\to_{R,\epsilon} /_{\mathsf{min}} \to_S)$ we shortly write $\mathsf{SN}(R_{\mathsf{top}}/_{\mathsf{min}} S)$. In [12] this is called finiteness of the dependency pair problem (R, Q, S) with *minimality flag*; in our setting the middle TRS Q is empty. Again the motivation for this definition is in proving termination: From [2] we know

$$\mathsf{SN}(\mathsf{DP}(R)_{\mathsf{top}}/_{\mathsf{min}} R) \iff \mathsf{SN}(R).$$

For $\mathsf{SN}(R_{\mathsf{top}}/_{\mathsf{min}} S)$ it is not clear how to define a single term variant, in particular for terms that are not S-terminating. In this section we prove that $\mathsf{SN}(R_{\mathsf{top}}/_{\mathsf{min}} S)$ is Π^0_2-complete. For doing so first we give some lemmas.

Lemma 11.1. *Let R, S be TRSs. Then $\mathsf{SN}(R_{\mathsf{top}}/_{\mathsf{min}} S)$ holds if and only if*

$$(\to_{R,\epsilon} \cup \to_S) \cap \to_{\mathsf{SN}(\to_S)}$$

is terminating.

Proof. By definition $\mathsf{SN}(R_{\mathsf{top}}/_{\mathsf{min}}\, S)$ is equivalent to termination of $(\to_S^* \cdot \to_{R,\epsilon}) \cap \to_{\mathsf{SN}(\to_S)}$. Since

$$(\to_S^* \cdot \to_{R,\epsilon}) \cap \to_{\mathsf{SN}(\to_S)} \;\subseteq\; ((\to_{R,\epsilon} \cup \to_S) \cap \to_{\mathsf{SN}(\to_S)})^+,$$

the 'if'-part of the lemma follows.

For the 'only if'-part assume that $(\to_{R,\epsilon} \cup \to_S) \cap \to_{\mathsf{SN}(\to_S)}$ admits an infinite reduction. If this reduction contains finitely many $\to_{R,\epsilon}$-steps, then this reduction ends in an infinite $\to_S$-reduction, contradicting the assumption that all terms in this reduction are S-terminating. Thus, this reduction contains infinitely many $\to_{R,\epsilon}$-steps, hence can be written as an infinite $(\to_S^* \cdot \to_{R,\epsilon}) \cap \to_{\mathsf{SN}(\to_S)}$ reduction. □

Lemma 11.2. *Let R, S be TRSs. Then $\mathsf{SN}(R_{\mathsf{top}}/_{\mathsf{min}}\, S)$ holds if and only if for every term t and every $m \in \mathbb{N}$ there exists $n \in \mathbb{N}$ such that*

for every n-step $(\to_{R,\epsilon} \cup \to_S)$-reduction $t = t_0 \to t_1 \to \cdots \to t_n$ there exists $i \in [0, n]$ and an m-step $\to_S$-reduction of t_i.

Proof. Due to Lemma 11.1 $\mathsf{SN}(R_{\mathsf{top}}/_{\mathsf{min}}\, S)$ is equivalent to finiteness of all $(\to_{R,\epsilon} \cup \to_S)$-reductions only consisting of $\to_S$-terminating terms. Since $(\to_{R,\epsilon} \cup \to_S)$ is finitely branching, by Lemma 2.2 this is equivalent to:

For every term t there exists $n \in \mathbb{N}$ such that no n-step $(\to_{R,\epsilon} \cup \to_S)$-reduction $t = t_0 \to t_1 \to \cdots \to t_n$ exists for which t_i is $\to_S$-terminating for every $i \in [0, n]$.

Since $\to_S$ is finitely branching, by Lemma 2.2 $\to_S$-termination of t_i for every $i \in [0, n]$ is equivalent to the existence of $m \in \mathbb{N}$ such that no t_i admits an m-step $\to_S$-reduction. After removing double negations, this proves equivalence with the claim in the lemma. □

Theorem 11.3. *The property $\mathsf{SN}(R_{\mathsf{top}}/_{\mathsf{min}}\, S)$ for given TRSs R, S is Π^0_2-complete.*

Proof. $\mathsf{SN}(R)$ is Π^0_2-complete and $\mathsf{SN}(R)$ is equivalent to $\mathsf{SN}(\mathsf{DP}(R)_{\mathsf{top}}/_{\mathsf{min}}\, R)$, so $\mathsf{SN}(R_{\mathsf{top}}/_{\mathsf{min}}\, S)$ is Π^0_2-hard. That $\mathsf{SN}(R_{\mathsf{top}}/_{\mathsf{min}}\, S)$ is in Π^0_2 follows from Lemma 11.2. □

12. Conclusions and future work

We have analyzed the proof theoretic complexity of the basic properties of term rewriting systems and ascertained their positions in the arithmetical and analytical hierarchies.

The position of Π^0_2 of the properties WN and SN is as expected, whereas the position Π^1_1 of dependency pair problems is remarkably high.

For confluence problems, we have shown that (ground) confluence is Π^0_2-complete both uniform and for single terms. Surprisingly, there is a difference between the arithmetical complexity of local confluence (on open terms) and local confluence on ground terms. While the former is Σ^0_1, the latter turns out to be Π^0_2-complete.

There is a wide range of possible future research concerned with gauging the proof theoretic complexity of properties of properties of (subclasses of) rewriting systems. For example, one may attempt to ascertain, for every property for which undecidability is known, its precise position in the hierarchies. For certain properties this is quite easy, e.e. termination of a single term rewrite rule: Undecidability was proved in [5] by using a reduction from the uniform halting problem for Turing machines. As a consequence, this problem is Π^0_2-hard. As it is clearly also in Π^0_2 as an instance of termination of a finite TRS, we immediately obtain Π^0_2-completeness.

For other properties, earlier undecidability results cannot easily be lifted to find the exact position in the hierarchies. For instance, in [10] it was proved that restricting to the subclass of TRSs that are WN, the property SN is undecidable, so called *relative undecidability*. The technique used there for proving this is based on reduction from Post's Correspondence Problem (PCP), by which it is not possible to prove hardness for classes higher in the arithmetical hierarchy than Π^0_1 or Σ^0_1. Instead by adding a fresh symbol $\bot$ and rules $f(x_n, \ldots, x_n) \to \bot$ for every symbol $f \in \Sigma$, we easily force our construction from Theorem 5.9 to be WN without changing validity of SN. In this way it is easily proved that uniform SN is Π^0_2-complete for TRSs satisfying WN. For all other relative undecidability results obtained in [10,11] one can wonder what is the precise level in the hierarchies. For cases where this is higher than Π^0_1 or Σ^0_1, the constructions given there will not be helpful, as they are all based on PCP.

Other possible future work includes a further study of the place in the arithmetical and analytical hierarchies of properties of variations of rewriting: for example graph rewriting, conditional rewriting, probabilistic rewriting, and infinitary rewriting. For some of these variants the proof theoretic complexity of the various fundamental properties may be expected to be the same as in the "pure" setting we have considered in this paper. For infinitary rewriting, we expect that the extension to infinite terms and reductions entails that most properties will be classified in the low levels of the analytical hierarchy—for

example, we have already sketched a proof of how SN^∞ behaves like dependency pair problems yielding Π^1_1. For WN^∞ we are working on a proof of Π^1_2-completeness (surprisingly higher than SN^∞); for other properties, we do not know.

Finally, we believe it interesting to study restricted classes of rewriting systems where the properties we consider are undecidable, yet at strictly lower levels in the hierarchies than proved in this paper. Note that care must be taken with the notion of "restriction": For example, local confluence in general has turned out to be lower in the arithmetical hierarchy than the "restricted" case of local confluence on ground terms.

References

[1] T. Aoto, Y. Toyama, Persistency of confluence, Journal of Universal Computer Science 3] (1997) 1134–1147.
[2] T. Arts, J. Giesl, Termination of term rewriting using dependency pairs, Theoretical Computer Science 236 (2000) 133–178.
[3] J. Bergstra, J. Klop, Conditional rewrite rules: confluence and termination, Journal of Computer and Systems Sciences 32 (3) (1986) 323–362.
[4] W. Boone, The word problem, Proceedings of the National Academy of Sciences 44 (10) (1958) 1061–1064.
[5] M. Dauchet, Termination of rewriting is undecidable in the one-rule case, in: MFCS, number 324 in LNCS, Springer, 1988, pp. 262–270.
[6] M. Dauchet, T. Heuillard, P. Lescanne, S. Tison, Decidability of the confluence of finite ground term rewrite systems and of other related term rewrite systems, Information and Computation 88 (2) (1990) 187–201.
[7] J. Endrullis, H. Geuvers, H. Zantema, Degrees of undecidability in term rewriting, in: E. Grädel, R. Kahle (Eds.), Proceedings of Computer Science Logic (CSL09), Lecture Notes in Computer Science, vol. 5771, Springer, 2009, pp. 255–270.
[8] M. Fernandez, Models of Computation: An Introduction to Computability Theory, Undergraduate Topics in Computer Science, Springer, London, 2009.
[9] A. Geser, Omega-termination is undecidable for totally terminating term rewriting systems, Journal of Symbolic Computation 23 (4) (1997) 399–411.
[10] A. Geser, A. Middeldorp, E. Ohlebusch, H. Zantema, Relative undecidability in term rewriting part i: the termination hierarchy, Information and Computation 178 (1) (2002) 101–131.
[11] A. Geser, A. Middeldorp, E. Ohlebusch, H. Zantema, Relative undecidability in term rewriting part ii: the confluence hierachy, Information and Computation 178 (1) (2002) 132–148.
[12] J. Giesl, R. Thiemann, P. Schneider-Kamp, The dependency pair framework: combining techniques for automated termination proofs, in: F. Baader, A. Voronkov (Eds.), Proceedings of LPAR'04, Lecture Notes in Artificial Intelligence, vol. 3452, Springer, 2005, pp. 301–331.
[13] G.T. Herman, Strong computability and variants of the uniform halting problem, Zeitschrift für Mathematische Logik und Grundlagen der Mathematik 17 (1) (1971) 115–131.
[14] P.G. Hinman, Recursion-theoretic Hierarchies, Springer, 1978.
[15] G. Huet, D. Lankford, On the uniform halting problem for term rewriting systems, Technical Report 283, IRIA, France, 1978.
[16] F. Jacquemard, Decidable approximations of term rewriting systems, in: Rewriting Techniques and Applications, 7th International Conference, RTA-96, New Brunswick, NJ, USA, July 27–30, 1996, Proceedings,Lecture Notes in Computer Science, vol. 1103, Springer-Verlag, 1996.
[17] F. Jacquemard, Reachability and confluence are undecidable for flat term rewriting systems, Information Processing Letters 87 (5) (2003) 265–270.
[18] S. Kaplan, Conditional rewrite rules, Theoretical Computer Science 33 (2) (1984) 175–193.
[19] F. Klay, Undecidable properties of syntactic theories, in: Proceedings of RTA '91, Lecture Notes in Computer Science, vol. 488, Springer-Verlag, 1991, pp. 136–149.
[20] S.C. Kleene, Recursive predicates and quantifiers, Transactions of the American Mathematical Society 53 (1934) 41–73.
[21] S.C. Kleene, Arithmetical predicates and function quantifiers, Transactions of the American Mathematical Society 79 (1955) 312–340.
[22] S.C. Kleene, Hierarchies of number theoretic predicates, Bulletin of the American Mathematical Society 61 (1955) 193–213.
[23] J.W. Klop, Term rewriting systems, in: S. Abramsky, D.M. Gabbay, S.E. Maibaum (Eds.), Handbook of Logic in Computer Science, vol. 2, Oxford University Press Inc., 1992, pp. 1–116.
[24] J.W. Klop, R.C. de Vrijer, Infinitary normalization, We Will Show Them! Essays in Honour of Dov Gabbay, vol. 2, College Publications, 2005, pp. 169–192.
[25] K.-I. Ko, D.-Z. Du, Theory of Computational Complexity, Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley and Sons Inc., New York, 2000.
[26] Y. Matiyasevich, Simple examples of undecidable associative calculi, Soviet Mathematics Doklady 8 (1967) 555–557.
[27] A. Middeldorp, B. Gramlich, Simple termination is difficult, Applicable Algebra in Engineering, Communication and Computing 6 (2) (1995) 115–128.
[28] T. Nagaya, Y. Toyama, Decidability for left-linear growing term rewriting systems, Information and Computation 178 (2) (2002) 499–514.
[29] P. Novikov, On the algorithmic unsolvability of the word problem in group theory, Proceedings of the Steklov Institute of Mathematics 44 (1955) 1–143.
[30] G. Roşu, Equality of streams is a Π^0_2-complete problem, in: Proceedings of the 11th ACM SIGPLAN International Conference on Functional Programming (ICFP'06), ACM (2006).
[31] H. Rogers, Theory of Recursive Functions and Effective Computability, Mc-Graw Hill, 1967.
[32] J.R. Shoenfield, Mathematical Logic, Association for Symbolic Logic, A.K. Peters, 1967.
[33] J.G. Simonsen, The Π^0_2-completeness of most of the properties of rewriting you care about (and productivity), in: R. Treinen (Ed.), Proceedings of the 20th Conference on Rewriting Techniques and Applications (RTA), Lecture Notes in Computer Science, vol. 5595, Springer, 2009, pp. 335–349.
[34] M. Sipser, Introduction to the Theory of Computation, second ed., Course Technology, 2006.
[35] Terese, Term Rewriting Systems, Cambridge Tracts in Theoretical Computer Science, vol. 55, Cambridge University Press, 2003.
[36] A. Turing, Computability and λ-definability, Journal of Symbolic Logic 2 (1937) 153–163.
[37] F. van Raamsdonk, Higher-order rewriting, in: Proceedings of the 10th International Conference on Rewriting Techniques and Applications (RTA '99), Lecture Notes in Computer Science, vol. 1631, Springer-Verlag, 1999, pp. 220–239.
[38] R.M. Verma, A. Hayrapetyan, A new decidability technique for ground term rewriting systems with applications, ACM Transactions on Computational Logic 6 (1) (2005) 102–123.
[39] H. Zantema, Termination of term rewriting: interpretation and type elimination, Journal of Symbolic Computation 17 (1) (1994) 23–50.
[40] H. Zantema, Total termination of term rewriting is undecidable, Journal of Symbolic Computation 20 (1) (1995) 43–60.
[41] H. Zantema, Normalization of infinite terms, in: A. Voronkov (Ed.), Proceedings of the 19th Conference on Rewriting Techniques and Applications (RTA), Lecture Notes in Computer Science, vol. 5117, Springer, 2008, pp. 441–455.