Degrees of Undecidability in Term Rewriting

Jörg Endrullis[1], Herman Geuvers[2,3], and Hans Zantema[2,3]

[1] Vrije Universiteit Amsterdam, The Netherlands
joerg@few.vu.nl
[2] Radboud Universiteit Nijmegen, The Netherlands
herman@cs.ru.nl
[3] Technische Universiteit Eindhoven, The Netherlands
h.zantema@tue.nl

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 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 also quantification over function variables is allowed.

In this paper we consider properties of first order term rewriting systems and classify them in this hierarchy. Most of the standard properties are Π^0_2-complete, that is, of the same level as uniform halting of Turing machines. In this paper we show two exceptions. Weak confluence is Σ^0_1-complete, and therefore essentially easier than ground weak confluence which is Π^0_2-complete.

The most surprising result is on dependency pair problems: we prove this to be Π^1_1-complete, which means that this property exceeds the arithmetical hierarchy and is essentially analytic. A minor variant, dependency pair problems with minimality flag, turns out be Π^0_2-complete again, just like the original termination problem for which dependency pair analysis was developed.

1 Introduction

In classical computability theory a property $P \subseteq \mathbb{N}$ is called *decidable* iff there exists a Turing machine which for every input $x \in \mathbb{N}$ outputs 0 if $x \in P$ and 1 if $x \notin P$. The complexity of decidable properties is usually defined in terms of the time (or space) consumption of a Turing machine that decides the property; the respective hierarchies (linear, polynomial, exponential,...) are well-known. Likewise, but less known, the undecidable properties can be classified into a hierarchy of growing complexity. The arithmetical and the analytical hierarchy establish such a classification of undecidable properties by the complexity of predicate logic formulas that define them, which in turn is defined as the number of quantifier alternations of its prenex normal form.

The *arithmetical hierarchy* is based on first order formulas, that is, quantification is restricted to number quantifiers, function or set quantification is not

E. Grädel and R. Kahle (Eds.): CSL 2009, LNCS 5771, pp. 255–270, 2009.

allowed; its classes are denoted Π^0_n and Σ^0_n for $n \in \mathbb{N}$. The lowest level of the hierarchy, the classes Π^0_0 and Σ^0_0, consists of the decidable relations (for which there is a total computable function that decides it). Then the classes Π^0_n and Σ^0_n for $n \geq 1$ are inductively defined by allowing additional universal and existential quantifiers to define the properties. 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. In other words, a relation belongs to the class Π^0_n for $n \in \mathbb{N}$ of the arithmetical hierarchy if it can be defined by a first order formula (in prenex normal form), which has n quantifiers, starting with a universal quantifier. Likewise a relation is in Σ^0_n if the formula starts with an existential quantifier. The class Σ^0_1 consists of *recursively enumerable* (or semi-decidable) relations; the blank tape halting problem is in this class. The initialised uniform halting problem is in the class Π^0_2.

The *analytical hierarchy* continues the classification by second order formulas, allowing for function quantifiers. Its classes are denoted Π^1_n and Σ^1_n for $n \in \mathbb{N}$. The lowest level of the analytical hierarchy are the the classes Π^1_0 and Σ^1_0 which consist of all arithmetical relations. The classes Π^1_n and Σ^1_n for $n \geq 1$ are defined inductively, each time adding an universal ($\forall \alpha : \mathbb{N} \to \mathbb{N}. \varphi$) or existential function quantifier ($\exists \alpha : \mathbb{N} \to \mathbb{N}.\ \varphi$), respectively. For the current paper we employ only the class Π^1_1 of the analytical hierarchy which consists of relations that can be defined by $\forall \alpha : \mathbb{N} \to \mathbb{N}.\ \varphi$ where φ is an arithmetical relation.

Our Contribution. We investigate the complexity, of the following properties of first order TRSs:

- confluence (CR),
- weak confluence (WCR),
- finiteness of dependency pair problems [2,5] (DP), and
- finiteness of dependency pair problems with minimality flag [5] ($\mathsf{DP}^{\mathsf{min}}$).

In this paper we pinpoint the precise complexities of these properties in terms of the arithmetic and analytic hierarchy. Table 1 provides a classification of various standard properties of TRSs: CR, WCR, DP, $\mathsf{DP}^{\mathsf{min}}$, termination (SN), ground confluence (grCR), and weak ground confluence (grWCR). We note that $\mathsf{DP}^{\mathsf{min}}$ is only applicable in the uniform variant, see Section 6.

Table 1. Degrees of undecidability

	SN	WN	CR	grCR	WCR	grWCR	DP	$\mathsf{DP}^{\mathsf{min}}$
uniform	Π^0_2	Π^0_2	$\textcircled{\Pi^0_2}$	Π^0_2	$\textcircled{\Sigma^0_1}$	Π^0_2	$\textcircled{\Pi^1_1}$	$\textcircled{\Pi^0_2}$
single term	Σ^0_1	Σ^0_1	Π^0_2	Π^0_2	$\textcircled{\Sigma^0_1}$	Σ^0_1	$\textcircled{\Pi^1_1}$	–

The contributions of this paper are encircled. The non-encircled uniform properties have been studied in [7] and [12]. For the complexity of these properties for single terms we refer to [3], an extended version of the present paper.

We deepen the study of [12] and find surprisingly that weak ground confluence is harder than weak confluence. While in [12] it has been shown that weak ground

confluence is Π^0_2-complete, we prove that weak confluence (that is, including open terms) is Σ^0_1-complete, a class strictly below Π^0_2 in the arithmetical hierarchy. This is an excellent counterexample to a common pitfall for people less familiar with complexity theory: the Π^0_2-hardness of weak confluence on all ground terms does not imply Π^0_2-hardness of weak confluence on the larger set of all terms.

As can be seen in Table 1, the standard properties of TRSs reside within the classes Π^0_2 and Σ^0_1 of the arithmetical hierarchy (both for the uniform and single term versions). That is, they are of a low degree of undecidability, being at most as hard as the initialised uniform halting problem.

Surprisingly, it turns out that dependency pair problems are of a much higher degree of undecidability: they exceed the whole arithmetical hierarchy and thereby first order predicate logic. In particular we show that dependency pair problems are Π^1_1-complete, a class within the analytical hierarchy with one universal function quantifier. So although dependency pair problems have been invented for proving termination, the complexity of general dependency pair problems is much higher than the complexity of termination itself. This even holds if we restrict to the special format of dependency pairs: dependency pairs are right-linear, all root symbols of left hand sides and right hand sides of dependency pairs are marked, and all other symbols in the dependency pairs and all symbols in the rewrite rules are unmarked. A variant of dependency pair problems again arising from termination 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^0_2-complete.

2 Preliminaries

Term Rewriting

We give a brief introduction to term rewriting, we refer to [13] for further reading. A *signature* Σ is a finite set of symbols f each having a fixed *arity* $\sharp(f) \in \mathbb{N}$. Let Σ be a signature and $\mathcal{X}$ a countably infinite set of variables 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: $\mathcal{P}os(f(t_1, \ldots, t_n)) = \{\varepsilon\} \cup \{ip \mid 1 \leq i \leq \sharp(f), p \in \mathcal{P}os(t_i)\}$, 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 Σ, $\mathcal{X}$ is a set R of finitely many pairs $\langle \ell, r\rangle \in \mathit{Ter}(\Sigma, \mathcal{X})$, 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 ($\mathit{Var}(r) \subseteq \mathit{Var}(\ell)$). Let R be a TRS. For terms $s, t \in \mathit{Ter}(\Sigma, \mathcal{X})$ we write $s \to_R t$ if there exists a rule $\ell \to r \in R$, a substitution σ and a context $C \in \mathit{Ter}(\Sigma, \mathcal{X} \cup \{\Box\})$ such that $s \equiv C[\ell\sigma]$ and $t \equiv C[r\sigma]$; $\to_R$ is the *rewrite relation* induced by R, and $\to_R^*$ denotes the reflexive, transitive closure of $\to_R$.

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

- *strongly normalizing (or terminating) on* t, denoted $\mathsf{SN}_R(t)$, if every rewrite sequence starting from t is finite.
- *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, $\forall t_1, t_2.\, t_1 \leftarrow^* t \to^* t_2 \Rightarrow \exists d.\, t_1 \to^* d \leftarrow^* t_2$.
- *weakly 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, $\forall t_1, t_2.\, t_1 \leftarrow t \to t_2 \Rightarrow \exists d.\, t_1 \to^* d \leftarrow^* t_2$.

The TRS R is *strongly normalizing* (SN_R), *confluent* (CR_R) or *weakly confluent* (WCR_R) if the respective property holds on all terms $t \in \mathit{Ter}(\Sigma, \mathcal{X})$.

Turing Machines

Definition 2.2. 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 $\Box$, 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 $\{n \in \mathbb{Z} \mid \mathit{tape}(n) \neq \Box\}$ 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 $Q \cap \Gamma = \varnothing$, that is, the set of states and the alphabet are disjoint. 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 $\mathit{tape}(n) = \Box$ 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 2.3. 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 $S \subseteq \mathbb{N}^*$. We need only unary functions f_{M} and binary $>_{\mathsf{M}} \subseteq \mathbb{N} \times \mathbb{N}$ relations.

Definition 2.4. Let $\mathsf{M} = \langle Q, \Gamma, q_0, \delta\rangle$ be a Turing machine with $\mathsf{S}, \mathsf{0} \in \Gamma$. We define a partial function $f_{\mathsf{M}} : \mathbb{N} \rightharpoonup \mathbb{N}$ for all $n \in \mathbb{N}$ by:

$$f_{\mathsf{M}}(n) = \begin{cases} m & \text{if } \mathit{final}_{\mathsf{M}}(q_0\mathsf{S}^n\mathsf{0}) = \ldots q\mathsf{S}^m\mathsf{0}\ldots \\ \text{undefined} & \text{otherwise} \end{cases}$$

and we define the relation $>_{\mathsf{M}} \subseteq \mathbb{N} \times \mathbb{N}$ by:

$$n >_{\mathsf{M}} m \iff \mathit{final}_{\mathsf{M}}(\mathsf{0}\mathsf{S}^n q_0 \mathsf{S}^m \mathsf{0}) = \ldots q\mathsf{0}\ldots .$$

Here, the functions f_{M} are partial since M may not terminate on certain inputs or M halts in a state which is not of the form $\ldots q\mathsf{S}^m\mathsf{0}\ldots$. Note that the set $\{ >_{\mathsf{M}} |\ \mathsf{M}$ halts on all tapes $\}$ is the set of recursive binary relations on $\mathbb{N}$.

The Arithmetic and Analytical Hierarchy

In the introduction we briefly mentioned the arithmetical and analytical hierarchy. We now summarize the main notions and results relevant for this paper. For details see a standard text on mathematical logic, e.g. [11] or [6], which contains more technical results regarding these hierarchies.

Definition 2.5. Let $A \subseteq \mathbb{N}$. The *set membership problem for* A is the problem of deciding for given $a \in \mathbb{N}$ whether $a \in A$.

Definition 2.6. 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 $\forall n \in \mathbb{N}.\ n \in A \Leftrightarrow f(n) \in B$.

Definition 2.7. 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 $B \in \mathcal{P}$.

So a problem B is $\mathcal{P}$*-hard* if every problem $A \in \mathcal{P}$ can be reduced to B: To decide "$n \in A$" we only have to 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 2.8. *If A can be reduced to B and A is* $\mathcal{P}$-hard*, then B is* $\mathcal{P}$-hard. □

Remark 2.9. Finite lists of natural numbers can be encoded as natural numbers 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. For this encoding, the length function ($\mathrm{lth}\langle n_1, \ldots, n_k \rangle = k$) and the decoding function ($\mathrm{lth}\langle n_1, \ldots, n_k \rangle_i = n_i$ if $1 \leq i \leq n$) are computable and it is decidable if a number is the code of a finite list.

Using the encoding of finite lists of natural numbers, we can encode Turing machines, terms and finite term rewriting systems. Finite rewrite sequences $\sigma : t_1 \to \ldots \to t_n$ can be encoded as lists of terms. Then of course a Turing machine can compute the length of $|\sigma| := n$ of the sequence, every term $t_1, \ldots, t_n$, in particular the first $\mathit{first}(\sigma) := t_1$ and the last term $\mathit{last}(\sigma) := t_n$. Given the TRS as input, a Turing machine 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'.

An example from term rewriting that we can encode as a problem on natural numbers is (we leave the encoding of terms as numbers implicit), $s \to_R t := \exists \ell \to r \in R.\, \exists \sigma.\, \exists C.\, (s \equiv C[\ell\sigma] \wedge t \equiv C[r\sigma])$. As all these quantifiers can be bounded by the size of s and t, respectively, this amounts to a finite search and is a decidable problem. Note that the fact that the TRS is finite is crucial here.

Undecidable problems can be divided into a hierarchy of increasing complexity, the first part of which is known as the *arithmetical hierarchy*. An example is the problem whether t reduces in finitely many steps to q: $t \to^*_R q := \exists \langle s_1, \ldots, s_n \rangle.(t = s_1 \to_R \ldots \to_R s_n = q)$. This problem is undecidable in general and it resides in Σ^0_1 which is the class of problems of the form $\exists x \in \mathbb{N}.\, P(x, n)$ where $P(x, n)$ is a decidable problem. (We usually suppress the domain behind the existential quantifier.) Similar to Σ^0_1, we have the class Π^0_1, which is the class of problems of the form $\forall x \in \mathbb{N}.\, P(x, n)$ with $P(x, n)$ a decidable problem. If we continue this procedure, we obtain the classes Σ^0_n and Π_n for every $n \in \mathbb{N}$.

Definition 2.10. The class Σ^0_n consists of all sets A that can be defined in form of $A(k) \iff \exists x_n. \forall x_{n-1}. \ldots P(x_1, \ldots, x_n, k)$ where P is a decidable relation. So, there is a sequence of n alternating quantifiers in front of P. Likewise Π^0_n is the class of sets of the form $A(k) \iff \forall x_n.\, \exists x_{n-1}.\, \ldots P(x_1, \ldots, x_n, k)$ where P is decidable. Then Δ^0_n is the intersection of Σ^0_n and Π^0_n, that is, $\Delta^0_n := \Sigma^0_n \cap \Pi^0_n$.

That this definition is useful is based on the following fact, for which we refer to [8,6,11] for a proof and further details.

Remark 2.11. 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. Furthermore a sequence of $\exists$ (or $\forall$) can always be replaced by one $\exists$ (or one $\forall$, respectively) due to the encoding of a finite list of numbers into numbers.

The reason one writes 0 as a superscript is that all quantifiers range over "the lowest type" $\mathbb{N}$; there are no quantifiers of higher types, like $\mathbb{N} \to \mathbb{N}$. So every

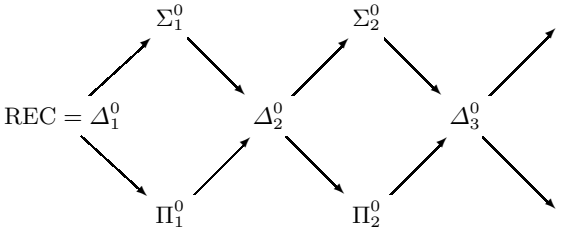

Fig. 1. Arithmetic Hierarchy

arithmetical problem is in one of the classes of Definition 2.10. A natural question is whether all these classes are distinct. A fundamental result in mathematical logic says that they are, see [11], [8] or [6].

The arithmetic hierarchy is usually depicted as in Figure 1, where every arrow denotes a proper inclusion. All classes are closed under bounded quantification: if $A(n) \Leftrightarrow \exists y < t(n)\, P(n, y)$ and P is decidable, then A is decidable (and similarly for other classes in the hierarchy).

Above the arithmetic hierarchy, we find the *analytic hierarchy*, where we also allow quantification over infinite sequences of numbers (equivalently functions $f : \mathbb{N} \to \mathbb{N}$). The classes of the analytical hierarchy are denoted Π_n^1 and Σ_n^1 with $n \in \mathbb{N}$ where the n indicates the number of function quantifiers in prenex normal form. That is, first-order quantifiers not counted. As a consequence the lowest classes Π_0^1 and Σ_0^1 subsume the whole arithmetical hierarchy. As variables ranging over infinite sequences we use α, β, etc. For the analytical hierarchy we can draw a similar diagram as the one in Figure 1: replace Σ_1^0 by Σ_1^1 etc. To keep the presentation as simple as possible we define only the class Π_1^1.

Definition 2.12. The class Π_1^1 consists of all sets A that can be defined in form of $A(k) \iff \forall\alpha.\, \varphi$ where φ is a first order formula over decidable predicates.

Note that φ does not need to be in prenex normal form. W.l.o.g. every φ can be converted into an equivalent formula φ' in prenex normal form and $\forall\alpha.\, \varphi'$ is still a Π_1^1-formula as first order quantifiers are not counted. For analytical problems we also have all kinds of simplification procedures (analogous to Remark 2.11).

An example of an analytical formula is $\forall\alpha.\, \exists x.\, \alpha(x) \not\to_R \alpha(x+1)$, stating that there exist no infinite rewrite sequences, that is, the rewrite system is SN. This is a Π_1^1-formula.

Lemma 2.13. *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_1^0*-complete,*
(ii) the totality problem $\{\mathsf{M} \mid \mathsf{M}$ *halts on* $q_0\mathsf{S}^n$ *for every* $n \in \mathbb{N}\}$ *is* Π_2^0*-complete,*
(iii) the set $\mathsf{WF} := \{\mathsf{M} \mid \mathsf{M}$ *is total and* $>_\mathsf{M}$ *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 the blank tape halting problem is many-one reducible to WCR and thus conclude that WCR is Σ_1^0-hard. This will be done by effectively giving for every Turing machine M, a TRS R_{M} such that

$$\mathsf{M} \text{ halts on the blank tape} \iff \mathsf{WCR}_{R_{\mathsf{M}}} .$$

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

To determine if a problem A is essentially in a certain class $\mathcal{P}$ (and not lower in the hierarchy), we first prove that A is $\mathcal{P}$-hard and then we show that the property A can be expressed by formula of $\mathcal{P}$.

3 Translating Turing Machines

We use the translation of Turing machines M to TRSs R_{M} from [10].

Definition 3.1. 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:

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

together with four rules for 'extending the tape':

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

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

Definition 3.2. We define a mapping $\varphi : \mathit{Ter}(\Gamma \cup \{\triangleright\}, \varnothing) \rightharpoonup \Gamma^*$ 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:

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

We define a map $\Phi : \mathit{Ter}_{\mathsf{M}} \to \mathcal{C}\mathit{onf}_{\mathsf{M}}$ by $\Phi(q(s,t)) := \varphi(s)^{-1} q \varphi(t) \in \mathcal{C}\mathit{onf}_{\mathsf{M}}$.

Lemma 3.3. *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 3.4. *For all* $s \in \mathit{Ter}_{\mathsf{M}}$*:* $\mathsf{SN}_{R_{\mathsf{M}}}(s) \iff \mathsf{M}$ *halts on* $\Phi(s)$.

Proof. Induction on item (ii) of Lemma 3.3. □

4 Weak Confluence

We show that WCR_R (uniform) and $\mathsf{WCR}_R(t)$ (for single terms) are Σ^0_1-complete. This result is surprising since (see Table 1) usually the uniform property is harder than for single terms: for the uniform property one has to reason about all terms which normally amounts to an additional universal quantifier. Moreover this reveals a remarkable discrepancy in comparison with grWCR_R which has been shown Π^0_2-complete in [12].

Theorem 4.1. *Weak confluence is Σ^0_1-complete, both uniform WCR_R as well as for single terms $\mathsf{WCR}_R(t)$.*

Proof. For Σ^0_1-hardness we use the blank tape halting problem. Let M be a Turing machine. We define the TRS S to consist of the rules of R_M extended by the following rules:

$$\begin{aligned} \mathsf{run} &\to \mathsf{T} \qquad \mathsf{run} \to q_0(\triangleright, \triangleright) \\ q(x, f(y)) &\to \mathsf{T} \quad \text{for every } f \in \Gamma \text{ such that } \delta(q, f) \text{ is undefined}\,. \end{aligned}$$

The only critical pair is $\mathsf{T} \leftarrow \mathsf{run} \to q_0(\triangleright, \triangleright)$. We have $q_0(\triangleright, \triangleright) \to^*_S \mathsf{T}$, if and only if M halts on the blank tape. By the Critical Pairs Lemma [13] WCR_R holds if and only if all critical pairs are convergent (can be joined). Hence WCR_R and $\mathsf{WCR}_R(t)$ (where $t := \mathsf{run}$) are Σ^0_1-hard.

A Turing machine can compute on the input of a TRS R all (finitely many) critical pairs, and on the input of a TRS R and a term t all (finitely many) one step reducts of t. Therefore it suffices to show that the following problem is in Σ^0_1: decide on the input of a TRS S, $n \in \mathbb{N}$ and 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 be described by the following Σ^0_1 formula:

$$\begin{aligned} \exists r \in \mathbb{N}.\, ((&r \text{ is list } r_1, \ldots, r_{2\cdot n} \text{ of length } 2 \cdot n) \\ &\text{and for } i = 1, \ldots, n \text{ we have} \\ &\quad (r_{2\cdot i-1},\, r_{2\cdot i} \text{ are reductions}) \text{ and } (\mathit{first}(r_{2\cdot i-1}) \equiv t_i) \\ &\quad \text{and } (\mathit{first}(r_{2\cdot i}) \equiv s_i) \text{ and } (\mathit{last}(r_{2\cdot i-1}) \equiv \mathit{last}(r_{2\cdot i}))\,. \end{aligned}$$

□

We remark that the proof also shows Σ^0_1-completeness of weak ground confluence for single terms ($\mathsf{grWCR}_R(t)$).

It may be unexpected that WCR_R is easier than grWCR_R, so let us add some explanation. In principle we have to check for an infinite number of possibilities, $t \to p$ and $t \to q$, whether p and q have a common reduct. The essence of the Critical Pairs Lemma (CPL) is that for WCR_R, it suffices to check a finite set of "overlapping patterns". For grWCR_R, this is no enough, because, even if some of the overlapping patterns are not convergent (and thus WCR_R is false), grWCR_R may still hold: for ground terms, all overlapping patterns may still be convergent.

As a simplified instance of this situation, consider a confluent term rewriting system with a unary symbol F that defines the recursive function $f : \mathbb{N} \to \mathbb{N}$.

Now, if we add the rules $\mathsf{run}(x) \to 0$ and $\mathsf{run}(x) \to F(x)$, then the rewrite system is not WCR anymore. However, it is grWCR if and only if f is the zero-function, which is a Π^0_2 statement. Note that, this argument does not go through as it stands, because there are some technical subtleties, but it gives the basic intuition why grWCR is "harder" than WCR.

5 Confluence

We investigate the complexity of confluence (CR_R). For proving Π^0_2-completeness of confluence one would like to use an extension of R_M with the following rules:

$$\begin{aligned} \mathsf{run}(x,y) &\to \mathsf{T} \\ \mathsf{run}(x,y) &\to q_0(x,y) \\ q(x,f(y)) &\to \mathsf{T} \qquad \text{for every } f \in \Gamma \text{ with } \delta(q,f) \text{ undefined} \end{aligned}$$

On first glance it seems that $q_0(s,t) \to^* \mathsf{T}$ if the Turing machine M halts on all configurations. However, a problem arises if s and t contain variables; e.g. if s or t are variables themselves. 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_M extended by:

$$\mathsf{run}(x,\triangleright) \to \mathsf{T} \tag{1}$$
$$\mathsf{run}(\triangleright,y) \to q_0(\triangleright,y) \tag{2}$$
$$q(x,f(y)) \to \mathsf{T} \qquad \text{for every } f \in \Gamma \text{ with } \delta(q,f) \text{ undefined} \tag{3}$$
$$\mathsf{run}(x,\mathsf{S}(y)) \to \mathsf{run}(\mathsf{S}(x),y) \tag{4}$$
$$\mathsf{run}(\mathsf{S}(x),y) \to \mathsf{run}(x,\mathsf{S}(y))\,. \tag{5}$$

Then T and $q_0(\triangleright,s)$ are convertible using the rules (1)–(5) if and only if s is a ground term of the form $\mathsf{S}^n(\triangleright)$.

Theorem 5.1. *Uniform confluence* CR_R *is* 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. We employ type introduction [1]: we assign sort γ_0 to $\Gamma \cup \{\triangleright\}$ and sort γ_1 to every symbol in $\{\mathsf{run},\ \mathsf{T}\} \cup Q$; the obtained many-sorted TRS is confluent if and only if S_M is. Note that the terms of sort γ_0 are normal forms and for terms of γ_1 with root symbol $\neq$ 'run' the reduction is deterministic (exhibits no branching). Therefore it suffices to consider the case

$$s_2 \leftarrow_{(2)} s_1 \leftarrow^*_{(4)} \mathsf{run}(t_1,t_2) \to^*_{(5)} s_3 \to_{(1)} \mathsf{T}$$

where $t_1, t_2 \in \mathit{Ter}(\Gamma \cup \{\triangleright\}, \mathcal{X})$. From the existence of such rewrite sequences 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 $s_2 \equiv q_0(\triangleright, \mathsf{S}^n(\triangleright))$. On the other hand for every $n \in \mathbb{N}$ such rewrite sequences exist. As a consequence the TRS S_M is confluent if and only if $q_0(\triangleright, \mathsf{S}^n(\triangleright)) \to^*_S \mathsf{T}$

for every $n \in \mathbb{N}$, and this holds if and only if M halts on $q_0\mathsf{S}^n$ for every $n \in \mathbb{N}$ by Corollary 3.4. This proves Π^0_2-hardness.

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

$$\begin{aligned}\mathsf{CR}_R \iff \forall t \in \mathbb{N}.\, \forall r_1, r_2 \in \mathbb{N}.\, \exists r_1', r_2' \in \mathbb{N}.\\ (((t \text{ is a term}) \text{ and } (r_1,\, r_2 \text{ are reductions}) \text{ and } t \equiv \mathit{first}(r_1) \equiv \mathit{first}(r_2))\\ \Rightarrow ((r_1' \text{ and } r_2' \text{ are reductions})\\ \text{and } (\mathit{last}(r_1) \equiv \mathit{first}(r_1')) \text{ and } (\mathit{last}(r_2) \equiv \mathit{first}(r_2'))\,.\\ \text{and } (\mathit{last}(r_1') \equiv \mathit{last}(r_2'))))\end{aligned}$$

By quantifier compression we can simplify the formula such that there is only one universal followed by an existential quantifier. □

6 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: it is Π^1_1-complete, both uniform and for single terms. This only holds for the basic version of dependency pairs; for the version with minimality flag (as arising from TRS termination problems) we show it is of the same level as termination itself. We emphasize that the variant without minimality flag is commonly used: they arise for example from Haskell termination problems [4], and transformations on dependency pair problems that do not preserve the minimality flag.

For relations $\to_1, \to_2$ we write $\to_1 / \to_2$ for $\to_2^* \cdot \to_1$. 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 [5] 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, that is, R-steps are allowed only at the top, S-steps everywhere, contains only finitely many $\to_{R,\epsilon}$ steps.

The motivation for studying this comes from the dependency pair approach [2] for proving termination. There a simple syntactic construction DP is given such that for any TRS R we have

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

In this way termination of a TRS can be proved by proving $\mathsf{SN}(R_{\mathsf{top}}/S)$ for suitable TRSs R, S. This is the basis of nearly all termination proofs for TRSs as they are generated by state-of-the-art termination provers.

The main result of this section is Π^1_1-completeness of dependency pair problems $\mathsf{SN}(R_{\mathsf{top}}/S)$, even of $\mathsf{SN}(S_{\mathsf{top}}/S)$, for both the uniform and the single term variant, and also of $\mathsf{SN}(R_{\mathsf{top}}/S)$ restricting to the format in which roots in R are marked. In the next 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 like normal termination: it is Π^0_2-complete.

For proving Π^1_1-hardness of $\mathsf{SN}(S_{\mathsf{top}}/S)$ we now adopt Definition 3.1, the translation of Turing machines to TRSs. The crucial difference is that every step of the Turing machine 'produces' one output pebble '$\bullet$'. Thereby we achieve that the TRS $R^{\bullet}_{\mathsf{M}}$ is top-terminating even if M does not terminate.

Definition 6.1. 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 $\Sigma = Q \cup \Gamma \cup \{\triangleright, \bullet, \mathsf{T}\}$ 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:

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

Then we obtain the following lemma characterizing $>_{\mathsf{M}}$ as defined in Definition 2.4.

Lemma 6.2. *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 $\mathsf{S}^n(\mathsf{0}(\triangleright))$:

Definition 6.3. We define the TRS R_{pickn} to consist of the following three rules:

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

Lemma 6.4. *The TRS* R_{pickn} *has the following properties:*

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

Now we are ready to prove Π^1_1-completeness of dependency pair problems.

Theorem 6.5. *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 WF is Π^1_1-complete, that is, checking well-foundedness of $>_{\mathsf{M}}$. 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:

$$\mathsf{run}(\mathsf{T}, \mathsf{ok}(x), \mathsf{ok}(y)) \to \mathsf{run}(q_0(x, y), \mathsf{ok}(y), \mathsf{pickn})\,, \tag{6}$$

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) $\mathsf{SN}(t, S_{\mathsf{top}}/S) \Longrightarrow >_{\mathsf{M}}$ is well-founded and (2) $>_{\mathsf{M}}$ is well-founded $\Longrightarrow \mathsf{SN}(S_{\mathsf{top}}/S)$.

(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:

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

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 (6) 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 (6) 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 6.4) 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:

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

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 (which establishes no problem for a Turing machine). 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. □

By the same argument as in this proof we obtain

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

for R consisting of the single rule $\mathsf{run}(\mathsf{T}, \mathsf{ok}(x), \mathsf{ok}(y)) \to \mathsf{run}(q_0(x,y), \mathsf{ok}(y), \mathsf{pickn})$ and S consisting of the rules of $R_{\mathsf{M}}^{\bullet} \uplus R_{\mathsf{pickn}}$, again for $t = \mathsf{run}(\mathsf{T}, \mathsf{pickn}, \mathsf{pickn})$. By considering run to be marked and all other symbols to be unmarked, we conclude Π_1^1-hardness and also Π_1^1-completeness for dependency pair problems $\mathsf{SN}(R_{\mathsf{top}}/S)$ satisfying the standard requirements:

- the root symbols of both ℓ and r are marked for every rule $\ell \to r$ in R;
- all other symbols in R and all symbols in S are unmarked.

We now sketch how this proof also implies Π_1^1-completeness of the property SN^∞ in infinitary rewriting, for its definition and basic observations see [9]. Since in Theorem 6.5 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 [14] (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 6.5 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. [14]. Since the TRS S used in the proof of Theorem 6.5 is left-linear we conclude that the property SN^∞ for left-linear TRSs is Π_1^1-complete.

7 Dependency Pair Problems with Minimality Flag

A variant in the dependency pair approach is the dependency pair problem with minimality flag. Here in the infinite $\to_{R,\epsilon} \cup \to_S$ reductions all terms are assumed to be S-terminating. This can be defined as follows. On the level of 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 and y are $\to_2$-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 [5] 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_2^0-complete. For doing so first we give some lemmas.

Lemma 7.1. *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 $(\to_{R,\epsilon} \cup \to_S) \cap \to_{\mathsf{SN}(\to_S)}$ 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. So 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 7.2. *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 $i \in [0, n]$ such that t_i admits an m-step $\to_S$-reduction.*

Proof. Due to Lemma 7.1 $\mathsf{SN}(R_{\mathsf{top}}/_{\mathsf{min}}\, S)$ 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, 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, $\to_S$-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 7.3. *The property $\mathsf{SN}(R_{\mathsf{top}}/_{\mathsf{min}}\, S)$ for given TRSs R, S is Π_2^0-complete.*

Proof. $\mathsf{SN}(R)$ is Π_2^0-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_2^0-hard. That $\mathsf{SN}(R_{\mathsf{top}}/_{\mathsf{min}}\, S)$ is in Π_2^0 follows from Lemma 7.2; note that the body of the claim in Lemma 7.2 is recursive. □

8 Conclusion and Future Work

In this paper we have analysed the proof theoretic complexity, in terms of the arithmetic and analytical hierarchy, of standard properties in term rewriting. Extending the work of [12], we observed that not all properties are Π_2^0-complete. In particular, weak confluence turns out to be Σ_1^0-complete, which is a lower class, while dependency pair problems are Π_1^1-complete, being a much higher class. In future work, we will also further study the place in the analytic hierarchy of properties of infinitary rewriting like WN^∞.

References

1. Aoto, T., Toyama, Y.: Persistency of confluence. J. Universal Computer Science 3, 1134–1147 (1997)
2. Arts, T., Giesl, J.: Termination of term rewriting using dependency pairs. Theoretical Computer Science 236, 133–178 (2000)
3. Endrullis, J., Geuvers, H., Zantema, H.: Degrees of Undecidabililty of TRS Properties (2009), `http://arxiv.org/abs/0902.4723`
4. Giesl, J., Swiderski, S., Schneider-Kamp, P., Thiemann, R.: Automated Termination Analysis for Haskell: From Term Rewriting to Programming Languages (invited lecture). In: Pfenning, F. (ed.) RTA 2006. LNCS, vol. 4098, pp. 297–312. Springer, Heidelberg (2006)
5. Giesl, J., Thiemann, R., Schneider-Kamp, P.: The dependency pair framework: Combining techniques for automated termination proofs. In: Baader, F., Voronkov, A. (eds.) LPAR 2004. LNCS (LNAI), vol. 3452, pp. 301–331. Springer, Heidelberg (2005)
6. Hinman, P.G.: Recursion-Theoretic Hierarchies. Springer, Heidelberg (1978)
7. Huet, G., Lankford, D.: On the uniform halting problem for term rewriting systems. Technical Report 283, IRIA, France, Mars (1978)
8. Rogers Jr., H.: Theory of recursive functions and effective computability. McGraw-Hill, New York (1967)
9. Klop, J.W., de Vrijer, R.C.: Infinitary normalization. In: We Will Show Them! Essays in Honour of Dov Gabbay, vol. 2, pp. 169–192. College Publications (2005)
10. Klop, J.W.: Term rewriting systems. In: Abramsky, S., Gabbay, M.D., Maibaum, S.E. (eds.) Handbook of Logic in Computer Science, vol. 2, pp. 1–116. Oxford University Press, Inc., Oxford (1992)
11. Shoenfield, J.R.: Mathematical Logic. Association for Symbolic Logic, by A.K. Peters (1967)
12. Simonsen, J.G.: The Π^0_2-Completeness of Most of the Properties of Rewriting Systems You Care About (and Productivity). In: Treinen, R. (ed.) RTA 2009. LNCS, vol. 5595, pp. 335–349. Springer, Heidelberg (2009)
13. Terese: Term Rewriting Systems. Cambridge Tracts in Theoretical Computer Science, vol. 55. Cambridge University Press, Cambridge (2003)
14. Zantema, H.: Normalization of infinite terms. In: Voronkov, A. (ed.) RTA 2008. LNCS, vol. 5117, pp. 441–455. Springer, Heidelberg (2008)