# Degrees of Undecidability in Rewriting

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

[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. Weak and strong normalization for single terms turn out to be $\Sigma^0_1$-complete, while their uniform versions as well as dependency pair problems with minimality flag are $\Pi^0_2$-complete. We find that confluence is $\Pi^0_2$-complete both for single terms and uniform. Unexpectedly weak confluence for ground terms turns out to be harder than weak confluence for open terms. The former property is $\Pi^0_2$-complete while the latter is $\Sigma^0_1$-complete (and thereby recursively enumerable).
The most surprising result is on dependency pair problems without minimality flag: we prove this to be $\Pi^1_1$-complete, which means that this property exceeds the arithmetical hierarchy and is essentially analytic.

## 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 allowed; its classes are denoted $\Pi^0_n$ and $\Sigma^0_n$ for $n \in \mathbb{N}$. The lowest level of the hierarchy, the classes $\Pi^0_0$ and

$\Sigma_0^0$, consists of the decidable relations (for which there is a total computable function that decides it). Then the classes $\Pi_n^0$ and $\Sigma_n^0$ 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 $\Sigma_1^0$ and $\forall y\, \exists x\, P(x, y, z)$ is in $\Pi_2^0$. In other words, a relation belongs to the class $\Pi_n^0$ 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 $\Sigma_n^0$ if the formula starts with an existential quantifier. The class $\Sigma_1^0$ is the class of *recursively enumerable* (or semi-decidable) relations; the special halting problem is in this class. The general halting problem is in the class $\Pi_2^0$.

The *analytical hierarchy* continues the classification of relations by second order formulas, allowing for function quantifiers. Its classes are denoted $\Pi_n^1$ and $\Sigma_n^1$ for $n \in \mathbb{N}$. The lowest level of the analytical hierarchy, that is, the classes $\Pi_0^1$ and $\Sigma_0^1$ consist of all arithmetical relations. The classes $\Pi_n^1$ and $\Sigma_n^1$ for $n \geq 1$ are defined inductively, each time adding an universal $(\forall \alpha : \mathbb{N} \to \mathbb{N}.\, \varphi)$ or existential function quantifier $(\exists \alpha : \mathbb{N} \to \mathbb{N}.\varphi)$, respecively. For example the class $\Pi_1^1$ consists of relations which can be defined by $\forall \alpha : \mathbb{N} \to \mathbb{N}.\, \varphi$ where $\varphi$ is an arithmetical relation.

*Our Contribution* We investigate the arithymetic complexity, of various properties of first order TRSs:

- termination or strong normalization (SN),
- weak normalization (WN),
- confluence (CR) and ground confluence (grCR),
- weak confluence (WCR) and weak ground confluence (grWCR),
- finiteness of dependency pair problems (DP), and
- finiteness of dependency pair problems with minimality flag ($\mathsf{DP}^{\mathsf{min}}$).

| | SN | WN | CR | grCR | WCR | grWCR |
|---|---|---|---|---|---|---|
| uniform | $\Pi_2^0$ | $\Pi_2^0$ | $\Pi_2^0$ | $\Pi_2^0$ | $\Sigma_1^0$ | $\Pi_2^0$ |
| single term | $\Sigma_1^0$ | $\Sigma_1^0$ | $\Pi_2^0$ | $\Pi_2^0$ | $\Sigma_1^0$ | $\Sigma_1^0$ |

| | DP | $\mathsf{DP}^{\mathsf{min}}$ |
|---|---|---|
| uniform | $\Pi_1^1$ | $\Pi_2^0$ |
| single term | $\Pi_1^1$ | $\Sigma_1^0$ |

**Fig. 1.** Degrees of undecidability

While undecidability of these concepts is folklore [3] their degree of undecidability, their precise hardness, has hardly been studied, with the exception of [7] who study the Turing degree of termination. Turing degrees give a classification of undecidable properties in terms of their computational 'hardness' which is independent of the syntactic form of a predicate that describes it. Their is a connection between the Turing degree of a property and its place in teh arithmetic hierrachy, so most of the proofs of [7] can be carry over to our setting. As we use a different translation from Turing machines to TRSs, we do not use the results or proofs of [7].

In this paper we pinpoint the precise complexities of these properties in terms of the arithmetic (and analytic) hierachy, see Figure 1; we study these properties *uniformly* for all terms (as a system property) as well as for *single terms*.

We find that the standard TRS properties $\mathsf{SN}$, $\mathsf{WN}$, $\mathsf{CR}$, $\mathsf{WCR}$ reside within the classes $\Pi^0_2$ and $\Sigma^0_1$ of the arithmetical hierarchy, for the uniform and single term versions, respectively. That is, they are of a low degree of undecidability, being at most as hard as the general halting problem.

Unexpectedly we find that weak ground confluence is a harder decision problem than weak confluence. While weak confluence is $\Sigma^0_1$-complete and therefore recursively enumerable it turns out that weak ground confluence $\Pi^0_2$-complete.

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 $\Pi^1_1$-complete, a class within the analytical hierarchy with one universal function quantifier. So although dependency pair problems are invented for proving termination, the complexity of general dependency pair problems is much higher than the complexity of termination itself. The same holds for the property $\mathsf{SN}^\infty$ of termination in infinitary rewriting [9]. We sketch how by the same argument $\mathsf{SN}^\infty$ can be concluded to be $\Pi^1_1$-complete.

A variant of dependency pair problems are dependency pair problems with minimality flag. We will show that for this variant the complexity is back to that of termination: it is $\Pi^0_2$-complete.

## 2 Preliminaries

### Term rewriting

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

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

We use $x, y, z, \ldots$ to range over variables. We frequently drop $\mathcal{X}$ and write $\mathit{Ter}(\Sigma)$ for the set of terms over $\Sigma$ and a fixed, countably infinite set of variables $\mathcal{X}$. The set of positions $\mathcal{P}os(t) \subseteq \mathbb{N}^*$ of a term $t \in \mathit{Ter}(\Sigma, \mathcal{X})$ 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 $\sigma$ is a map $\sigma : \mathcal{X} \to \mathit{Ter}(\Sigma, \mathcal{X})$ from variables to terms. For terms $t \in \mathit{Ter}(\Sigma, \mathcal{X})$ and substitutions $\sigma$ 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 $\mathit{Ter}(\Sigma, \mathcal{X} \cup \{\Box\})$ 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 $R$ 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* $\ell$ is not a variable $\ell \notin \mathcal{X}$ and all variables in the *right-hand side* $r$ occur in $\ell$, $\mathit{Var}(r) \subseteq \mathit{Var}(\ell)$. 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 $\sigma$ 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 $R$.

**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.
- *weakly normalizing on $t$*, denoted $\mathsf{WN}_R(t)$, if $t$ admits a rewrite sequence $t \twoheadrightarrow s$ to a normal form $s$.
- *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, $\forall t_1, t_2 \in \mathit{Ter}(\Sigma).\, t_1 \twoheadleftarrow t \twoheadrightarrow t_2 \Rightarrow \exists d.\, t_1 \twoheadrightarrow d \twoheadleftarrow t_2$.
- *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, $\forall t_1, t_2 \in \mathit{Ter}(\Sigma).\, t_1 \leftarrow t \to t_2 \Rightarrow \exists d.\, t_1 \twoheadrightarrow d \twoheadleftarrow t_2$.

The TRS $R$ is *strongly normalizing* ($\mathsf{SN}_R$), *weakly normalizing* ($\mathsf{WN}_R$), *confluent* ($\mathsf{CR}_R$) or *weakly confluent* ($\mathsf{WCR}_R$) if the respective property holds on all terms $t \in \mathit{Ter}(\Sigma, \mathcal{X})$. We say that $R$ is *ground confluent* (or *ground weakly confluent*) if $R$ is confluent (or weakly confluent) on all ground terms $t \in \mathit{Ter}(\Sigma, \varnothing)$.

### Turing machines

**Definition 2.2.** A *Turing machine* $\mathsf{M}$ 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 $\Gamma$ 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 $w_1, w_2 \in \Gamma^\infty$ 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 states are unique (if they exist), which justifies the following definition.

**Definition 2.3.** Let $\mathsf{M}$ be a Turing machine and $\langle q, tape\rangle \in \mathcal{C}onf_{\mathsf{M}}$. We denote by $final_{\mathsf{M}}(\langle q, tape\rangle)$ the $\rightarrow_{\mathsf{M}}$-normal form of $\langle q, tape\rangle$ if it exists and undefined, otherwise. Whenever $final_{\mathsf{M}}(\langle q, tape\rangle)$ exists then we say that $\mathsf{M}$ *halts on* $\langle q, tape\rangle$ *with final configuration* $final_{\mathsf{M}}(\langle q, tape\rangle)$. Furthermore we say $\mathsf{M}$ *halts on tape* as shorthand for $\mathsf{M}$ *halts on* $\langle q_0, tape\rangle$.

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 $f_{\mathsf{M}}$ 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:

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

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

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

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

### The arithmetic and analytical hierachy

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 hierrarchies.

**Definition 2.5.** Let $A \subset \mathbb{N}$. 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 $\mathrm{Seq}(n)$.

Using the encoding of finite lists of natural numbers, we can encode Turing machines, terms and finite term rewriting systems. The following, known as Kleene's $T$-predicate, is a well-known decidable problem: $T(m, \langle x \rangle, u) := m$ encodes a Turing Machine $M$, $u$ encodes the computation of $M$ on $x$ whose end result is $(u)_{\mathrm{lth}(u)}$.

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 are bounded (amounting to a finite search), this is a decidable problem. Note that the fact that the TRS is finite and thus finitely branching 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 $q$: $t \twoheadrightarrow_R q := \exists \langle s_1, \ldots, s_n \rangle (t = s_1 \to_R \ldots \to_R s_n = q)$. This problem is undecidable in general and it resides in the class $\Sigma_1^0$, 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.) Due to the encoding of a finite list of numbers into numbers, a sequence of $\exists$ can always be replaced by one.

Similar to $\Sigma_1^0$, we have the class $\Pi_1^0$, 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 $\Sigma_n^0$ and $\Pi_n$ for every $n \in \mathbb{N}$.

**Definition 2.10.** $\Sigma_n^0$ is the class of problems of the form
$A(k) = \exists x_n \forall x_{n-1} \ldots P(x_1, \ldots, x_n, k)$ where $P$ is decidable. So, there is a sequence of $n$ alternating quantifiers in front of $P$. $\Pi_n^0$ is the class of problems of the form $A(k) = \forall x_n \exists x_{n-1} \ldots P(x_1, \ldots, x_n, k)$ where $P$ is decidable. $\Delta_n^0 := \Sigma_n^0 \cap \Pi_n^0$

That this definition is useful is based on the following fact, for which 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.
For every formula of the form $\exists n \exists m \varphi$ there is an equivalent formula of the form $\exists p \varphi'$, where $\varphi'$ has the same quantifier structure as $\varphi$. Similarly, for every formula of the form $\forall n \forall m \varphi$ there is an equivalent formula of the form $\forall p \varphi'$, where $\varphi'$ has the same quantifier structure as $\varphi$.

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

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].

**Lemma 2.12.** *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 arithmetic hierarchy is usually depicted as in Figure 2, where every arrow denotes a proper inclusion. Schematically one usually writes $\exists$REC for $\Sigma_1^0$, $\forall\exists$REC for $\Pi_2^0$, etc. 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). To put it more succinctly: $\forall < \mathcal{P} = \mathcal{P}$ for all classes $\mathcal{P}$ in the arithmetic hierarchy.

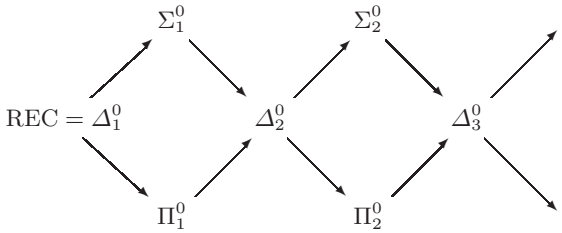


**Fig. 2.** Arithmetic Hierarchy

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

Above the arithmetic hierrachy, we find the *analytic hierarchy*, where we also allow quantification over infinite sequences of numbers. As variables ranging over infinite sequences we use $\alpha$, $\beta$, etc. An example of an analtyical formula is $\forall\alpha(\forall x(\alpha(x) \twoheadrightarrow_R \alpha(x+1)) \to \exists x(\alpha(x) = \alpha(x+1)))$, stating that the rewrite system is $\mathsf{SN}$. This is a $\Pi_1^1$-formula. In Section 3 we will see that we can express $\mathsf{SN}$ for TRSs with a formula that is much lower in the hierarchy: it is $\Pi_2^0$. The proof essentially uses the fact that TRSs are finitely branching.

The class $\Pi_1^1$ is the class of problems of the form $\forall\alpha\, \exists x\, P(n, \alpha, x)$, where $P$ decidable. Similarly $\Sigma_1^1$ is the class of problems of the form $\exists\alpha\, \forall x\, P(n, \alpha, x)$, where $P$ is decidable. For analytical problems we also have all kinds of simplification procedures (analoguous to the ones of Remark 2.11).

**Lemma 2.13.** *In the analytical hierarchy we have the following ways of simplifying a sequence of quantifiers:*

$$\forall^1\forall^1 \mapsto \forall^1 \quad \forall \mapsto \forall^1 \quad \exists\forall^1 \mapsto \forall^1\exists \quad \forall\exists^1 \mapsto \exists^1\forall$$

For the first two simplifications, we of course have the analogous versions with $\exists$. For the proof we refer to the standard literature; here we just give a rough idea. 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 $\forall\gamma\psi(\gamma)$, with $\psi$ in the same class as $\varphi$. (Just take $\psi(\gamma) := \varphi((\gamma)_1, \gamma)_2)$), where $(\gamma)_1$ denotes the sequence with $(\gamma)_1(n) = (\gamma(n))_1$.) The meaning of the other simplifications should be clear and from these simplifications one derives that each analytic formula is equivalent to one of the form $Q_n\alpha_n Q_{n-1}\alpha_{n-1}\ldots Q_0 x\, P(\alpha_1,\ldots,\alpha_n,x,k)$ where $P$ is decidable and $\boldsymbol{Q}$ is a sequence of alternating quantifiers. Any analytical problem can be written in this form.

**Definition 2.14.** The analytical problems are the ones of the form $Q_n\alpha_n Q_{n-1}\alpha_{n-1}\ldots Q_0 x\, P(\alpha_1,\ldots,\alpha_n,x,k)$ with $P$ is decidable and $\boldsymbol{Q}$ is a sequence of alternating quantifiers. If $n > 0$ and $Q_n = \exists^1$, then it is in the class $\Sigma^1_1$. If $n > 0$ and $Q_n = \forall^1$, then it is in the class $\Pi^1_1$. $\Delta^1_n := \Sigma^1_n \cap \Pi^1_n$

For the analytical hierarchy we can draw a same diagram as the one in Figure 2: replace $\Sigma^0_1$ by $\Sigma^1_1$ etc. We have the same results as Lemma 2.12: each class is a proper subclass of the ones above it. The whole arithmetic hierarchy is also a proper subclass of the lowest class, $\Delta^1_1$.

**Lemma 2.15.** *We have the following well-known results:*

*(i) the* special halting problem $\{\,\mathsf{M} \mid \mathsf{M}$ *halts on the blank tape*$\,\}$ *is* $\Sigma^0_1$*-complete,*
*(ii) the* general halting problem $\{\,\mathsf{M} \mid \mathsf{M}$ *halts on all inputs*$\,\}$ *is* $\Pi^0_2$*-complete,*
*(iii) the* totality problem $\{\,\mathsf{M} \mid \mathsf{M}$ *halts on* $q_0\mathsf{S}^n$ *for every* $n \in \mathbb{N}\,\}$ *is* $\Pi^0_2$*-complete,*
*(iv) the set* $\mathsf{WF} := \{\,\mathsf{M} \mid >_\mathsf{M}$ *is well-founded*$\,\}$ *is* $\Pi^1_1$*-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 "$\mathsf{WN}$ for a single term" and thus conclude that "$\mathsf{WN}$ for a single term" is $\Sigma^0_1$. This will be done by effectively giving for every Turing machine $\mathsf{M}$, a TRS $R_\mathsf{M}$ and a term $t_\mathsf{M}$ 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.

## 3 Strong and Weak Normalization

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

**Definition 3.1.** For every Turing machine $\mathsf{M} = \langle Q, \Gamma, q_0, \delta\rangle$ we define a TRS $R_\mathsf{M}$ 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 $R_\mathsf{M}$ 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':

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

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

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

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

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

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

**Lemma 3.3.** *Let $\mathsf{M}$ be a Turing machine. Then $R_{\mathsf{M}}$ simulates $\mathsf{M}$, 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.

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

*Remark 3.5.* As discussed in Remark 2.9, terms and term rewriting systems can be encoded as natural numbers. Finite rewrite sequnences $\sigma : t_1 \to \ldots \to t_n$ 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'.

We arrive at our first results.

**Theorem 3.6.** *The properties* $\mathsf{SN}$ *and* $\mathsf{WN}$ *for single terms are* $\Sigma^0_1$*-complete.*

*Proof.* For $\Sigma_1^0$-hardness we reduce the special halting problem to a termination problem for single terms. Therefore let $\mathsf{M}$ be an arbitrary Turing machine. Then $\mathsf{SN}_{R_\mathsf{M}}(q_0(\triangleright,\triangleright))$ if and only if $\mathsf{M}$ halts on the blank tape by Corollary 3.4. Moreover observe that $R_\mathsf{M}$ is orthogonal and non-erasing, thus the $\mathsf{SN}$ and $\mathsf{WN}$ coincide [12]. Hence both properties $\mathsf{SN}$ and $\mathsf{WN}$ for single terms are $\Sigma_1^0$-hard by Lemma 2.8.

To show that $\mathsf{SN}$ is in $\Sigma_1^0$, let $R$ be a TRS and $t \in \mathit{Ter}(\Sigma,\mathcal{X})$ a term. Since $R$ is finite, $t$ is terminating if and only if there exists a bound on the length of the reductions admitted by $t$, that is, 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 3.5 the predicate behind the quantifier is recursive. Hence $\mathsf{SN}$ for single terms is $\Sigma_1^0$-complete.

To show that $\mathsf{WN}$ is in $\Sigma_1^0$, let $R$ be a TRS and $t \in \mathit{Ter}(\Sigma,\mathcal{X})$ a term. The term $t$ is $\mathsf{WN}$ if there exists a reduction to a normal form:

$$\begin{aligned}\mathsf{WN}_R(t) \iff \exists r \in \mathbb{N}.\ &(r \text{ is a reduction})\\ &\text{and } t \equiv \mathit{first}(r) \text{ and } (\mathit{last}(r) \text{ is a normal form})\end{aligned}$$

This is a $\Sigma_1^0$-formula, hence $\mathsf{WN}$ for single terms is $\Sigma_1^0$-complete. □

For showing $\Pi_2^0$-completeness of the uniform properties $\mathsf{SN}$ and $\mathsf{WN}$ we would like to use the equivalence "$\mathsf{SN}(R_\mathsf{M}) \iff \mathsf{M}$ halts on all inputs", in combinatio with Lemma 2.15 (ii). However, this does not work because of the following two problems:

(1) In $R_\mathsf{M}$ 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 $\mathsf{M}$ halts on all inputs, does not guarantee that $\mathsf{M}$ halts when started in configuration $\langle q, wv\rangle$.
(2) In $R_\mathsf{M}$ we have terms of the form $q(q(w,v),u)$ that do not correspond to a configuration at all.

To deal with problem (1), we can use *type introduction* [13, 12], since $R_\mathsf{M}$ is non-collapsing. We asssign 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 $\mathit{Ter}_\mathsf{M}$ after replacing all variables by $\triangleright$, and by Corollary 3.4 for all terms $t \in \mathit{Ter}_\mathsf{M}$ we have $\mathsf{SN}_{R_\mathsf{M}}(t)$ if and only if $\mathsf{M}$ halts on $\Phi(t)$. Hence $\mathsf{SN}_{R_\mathsf{M}}$ holds if and only if $\mathsf{M}$ halts on all configurations $\mathcal{C}\mathit{onf}_\mathsf{M}$.

We now need to deal with problem (1); we would like that $\mathsf{M}$ halts on all configurations $\mathcal{C}\mathit{onf}_\mathsf{M}$ if and only if $\mathsf{M}$ halts on all inputs, starting from the initial state, but that's just not true. We need a Lemma about Turing machines; we use the following result by [5].

**Lemma 3.7 ([5]).** *For every Turing machine $\mathsf{M}$ 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)* $\mathsf{M}$ *hals on all configurations if and only if* $f$ *is total*

So, if $\mathsf{M}$ 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 general halting problem (set (ii) in Lemma 2.15) many-one reduces to the universal halting problem (the set in the Corollary), using Lemma 2.8. Basically, this corollary has already been stated and proved in [5].

**Corollary 3.8.** *The* uniform halting problem

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

*is* $\Pi^0_2$*-complete.*

**Theorem 3.9.** *The properties uniform* $\mathsf{SN}$ *and* $\mathsf{WN}$ *are* $\Pi^0_2$*-complete.*

*Proof.* For $\Pi^0_2$-hardness: we have seen how the universal halting problem for $\mathsf{M}$ many-one reduces to the uniform termination problem for $R_{\mathsf{M}}$. Since $R_{\mathsf{M}}$ is orthogonal and non-erasing $\mathsf{SN}$ and $\mathsf{WN}$ coincide [12]. Hence $\mathsf{SN}$ and $\mathsf{WN}$ are both $\Pi^0_2$-hard by Lemma 2.8. That the uniform properties $\mathsf{SN}$ and $\mathsf{WN}$ are in $\Pi^0_2$ follows from the fact that these properties for single terms can be described by $\Sigma^0_1$-formulas and the uniform property 'adds' a universal number quantifier.

## 4 Confluence and Ground Confluence

We investigate the complexity of confluence ($\mathsf{CR}$) and ground confluence ($\mathsf{grCR}$) both uniform and for single terms.

For proving $\Pi^0_2$-completeness of confluence one would like to use an extension of $R_{\mathsf{M}}$ with the following rules:

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

On first glance it seems that $q_0(s,t) \to^* \mathsf{T}$ if the Turing machine $\mathsf{M}$ 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 $\mathsf{M}$ we define the TRS $S_{\mathsf{M}}$ to consist of the rules of the TRS $R_{\mathsf{M}}$ extended by:

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

Then $\mathsf{T}$ 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 4.1.** *Uniform confluence (*$\mathsf{CR}$*), and uniform ground confluence (*$\mathsf{grCR}$*) are* $\Pi^0_2$*-complete.*

*Proof.* For proving $\Pi^0_2$-hardness we reduce the totality problem to confluence. Let $\mathsf{M}$ be an arbitrary Turing machine. We consider the TRS $S_\mathsf{M}$ defined above. We employ type introduction [1]: we assign sort $s_0$ to $\Gamma \cup \{\triangleright\}$ and sort $s_1$ to every symbol in $\{\mathsf{run}, \mathsf{T}\} \cup Q$; the obtained many-sorted TRS is confluend if and only if $S$ is. Note that the terms of sort $s_0$ are normal forms and for terms of $s_1$ with root symbol $\neq$ '$\mathsf{run}$' the reduction is deterministic (exhibits no branching). Therefore it suffices to consider the case

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

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 $S$ is confluent if and only if $q_0(\triangleright, \mathsf{S}^n(\triangleright)) \rightarrow^*_S \mathsf{T}$ for every $n \in \mathbb{N}$, that is, if and only if $\mathsf{M}$ halts on $q_0\mathsf{S}^n$ for every $n \in \mathbb{N}$. Moreover since the only critical terms $\mathsf{run}(t_1, t_2)$ are ground terms, we conclude that ground confluence coincides with confluence for $S$. Hence we have shown $\Pi^0_2$-hardness.

To show that both properties are in $\Sigma^0_2$ 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 universal followed by an existencial quantifier. Note that a formula for $\mathsf{grCR}$ is obtained by relacing '$t$ is a term' by '$t$ is a ground term'. Therefore both the uniform properties $\mathsf{CR}$ and $\mathsf{grCR}$ are $\Pi^0_2$-complete.

**Theorem 4.2.** *Confluence (*$\mathsf{CR}$*), and ground confluence (*$\mathsf{grCR}$*) for single terms are* $\Pi^0_2$*-complete.*

*Proof.* For $\Pi^0_2$-hardness we use the totality problem. Let $\mathsf{M}$ be an arbitrary Turing machine. We define the TRS $S$ as $R_\mathsf{M}$ extended by the following rules:

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

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

For $\Pi^0_2$-completeness note that we can formalize $\mathsf{CR}$ and $\mathsf{grCR}$ for single terms simply by dropping the universal quantification over all terms ($\forall t \in \mathbb{N}$) from the respecitive $\Pi^0_2$-formulas for the uniform properties in the proof of Theorem 4.1.

## 5 Weak Confluence and Weak Ground Confluence

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

**Theorem 5.1.** *The properties weak confluence (*CR*) both for single terms and uniform, and weak ground confluence (*grCR*) for single terms are $\Sigma_1^0$-complete.*

*Proof.* For $\Sigma_1^0$-hardness we use the special halting problem. Let $\mathsf{M}$ be an arbitrary Turing machine. We define the TRS $S$ to consist of the rules of $R_{\mathsf{M}}$ extended by the following rules:

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

The only critical pair is $\mathsf{T} \leftarrow \mathsf{run} \to q_0(\triangleright, \triangleright)$, and we have $q_0(\triangleright, \triangleright) \to_S^* \mathsf{T}$, if and only if $\mathsf{M}$ halts on the blank tape. By the Critical Pairs Lemma [12] we know that WCR holds if and only if all critical pairs are convergent (can be joined). Hence uniform WCR, and for single terms WCR and grWCR ($t := \mathsf{run}$) are $\Sigma_1^0$-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 $\Sigma_1^0$: 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 $\Sigma_1^0$ formula:

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

Surprisingly it turns out that uniform weak ground confluence is $\Sigma_2^0$-complete and thereby harder than uniform weak confluence (for the set of all open terms).

**Theorem 5.2.** *Uniform weak ground confluence (*grCR*) is $\Pi_2^0$-complete.*

*Proof.* For $\Pi_2^0$-hardness we use the uniform halting problem. Let $\mathsf{M}$ be a Turing machine. We define the TRS $S$ as extension of $R_{\mathsf{M}}$ with:

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

and rules

$$q(f(\boldsymbol{x}), g(\boldsymbol{y})) \to \mathsf{T}$$

for all combinations of symbols of $f, g$ such that the left-hand side is not matched by any of the rules in $R_{\mathsf{M}}$. Here $\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors of distinct variables such that the left-hand side of the rules are left-linear.

Assume there exists a configuration $c$ on which $\mathsf{M}$ does not halt. Then by Lemma 3.3 there exists $q(s,t) \in \Phi^{-1}(c)$ and by Corollary 3.4 $R_{\mathsf{M}}$ is not terminating on $q(s,t)$. Every reduct of $q(s,t)$ is an $R_{\mathsf{M}}$-redex and contains no further redexes. In particular, none of the extended rules is applicable to any reduct. Hence $q(s,t) \not\twoheadrightarrow \mathsf{T}$ and thus $\mathsf{T} \leftarrow \mathsf{run}(s,t) \to q(s,t)$ is not joinable.

Assume that $\mathsf{M}$ halts on all configurations. Let $D = \{\mathsf{run}\} \cup Q$. Let $V$ be the set of ground terms having a root symbol from $D$. All symbols apart from $D$ are constructor symbols. Hence for (weak) confluence it suffices to show that every reduct of a term in $V$ rewrites to $\mathsf{T}$. 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. Such a sequence can only exists if a ground of the form $q(s,t)$ admits an infinite $R_{\mathsf{M}}$-root rewrite sequence. Below the root (which is in $Q$) the rules from $R_{\mathsf{M}}$ match only symbols from $\Gamma \cup \{\triangleright\}$. Let $s'$ (and $t'$) be 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 $R_{\mathsf{M}}$-rewrite sequence, $s', t' \in \mathit{Ter}(\Sigma \cup \{\triangleright\}, \varnothing)$, and $q(s',t') \in \mathit{Ter}_{\mathsf{M}}$. Consequently $\Phi(q(s',t'))$ is a non-terminating configuration of $\mathsf{M}$ by Corollary 3.4, contradicting the assumption that $\mathsf{M}$ halts on all configurations. □

## 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 $\Pi^1_1$-complete, both uniform and for single terms. This only holds for the basic version of dependency pairs; for the version with minimality flag 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 [4] 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 $\Pi^1_1$-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 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 $\Pi^0_2$-complete.

For proving $\Pi^1_1$-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 the Turing machine $\mathsf{M}$ 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, $\mathsf{T}$ 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 $\mathsf{T}$ 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. (Recall the Definition of $>_{\mathsf{M}}$ in 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* $q_0(\mathsf{S}^n, \mathsf{S}^m) \twoheadrightarrow_{R^{\bullet}_{\mathsf{M}}} \mathsf{T}$. □

Moreover we define an auxiliary TRS $R_{\mathsf{pickn}}$ 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 $R_{\mathsf{pickn}}$ 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* $R_{\mathsf{pickn}}$ *has the following properties:*

- $\mathsf{pickn} \twoheadrightarrow \mathsf{ok}(\mathsf{S}^n(\mathsf{0}(\triangleright)))$ *for every* $n \in \mathbb{N}$*, and*
- *whenever* $\mathsf{pickn} \twoheadrightarrow \mathsf{ok}(t)$ *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 $\Pi^1_1$-completeness of dependency pair problems.

**Theorem 6.5.** *Both* $\mathsf{SN}(t, R_{\mathsf{top}}/S)$ *and* $\mathsf{SN}(R_{\mathsf{top}}/S)$ *are* $\Pi^1_1$*-complete.*

*Proof.* We prove $\Pi^1_1$-hardness even for the case where $R$ and $S$ coincide. We do this by using that the set $\mathsf{WF}$ is $\Pi^1_1$-complete, that is, checking well-foundedness of $>_{\mathsf{M}}$. Let $\mathsf{M}$ be an arbitrary Turing machine. From $\mathsf{M}$ 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) $\mathsf{SN}(t, S_{\mathsf{top}}/S) \iff >_{\mathsf{M}}$ 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:

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

Note that $q_0(\mathsf{S}^{n_i}(0(\triangleright)), \mathsf{S}^{n_{i+1}}(0(\triangleright))) \twoheadrightarrow \mathsf{T}$ (for all $i \geq 1$) because $\mathsf{M}$ 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 $\sigma$ 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 $\sigma$ have the root symbol $\mathsf{run}$. We consider the first three applications of (6) at the root in $\sigma$. After the first application the third argument of $\mathsf{run}$ is $\mathsf{pickn}$. Therefore after the second application the second argument of $\mathsf{run}$ is a reduct of $\mathsf{pickn}$ and the third is $\mathsf{pickn}$. Then before the third application the first argument is $\mathsf{T}$, and both the second and the third argument are reducts of $\mathsf{pickn}$. Thus $\mathsf{SN}(t, S_{\mathsf{top}}/S)$ cannot hold.

It remains to prove that both $\mathsf{SN}(R_{\mathsf{top}}/S)$ and $\mathsf{SN}(t, R_{\mathsf{top}}/S)$ are in $\Pi^1_1$. 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:

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

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 estabishes no problem for a Turing machine). For the property $\mathsf{SN}(t, R_{\mathsf{top}}/S)$ we simply add the condition $t = f(1)$ to restrict the quantification to such rewrite sequences $f$ that start with $t$. Hence $\mathsf{SN}(R_{\mathsf{top}}/S)$ and $\mathsf{SN}(t, R_{\mathsf{top}}/S)$ are $\Pi^1_1$-complete. □

We now sketch how this proof also implies $\Pi^1_1$-completeness of the property $\mathsf{SN}^\infty$ in infinitary rewriting, for its definition and basic observations see [9]. Since in Theorem 6.5 we proved $\Pi^1_1$-hardness even for the case where $R$ and $S$ coincide, we conclude that $\mathsf{SN}(S_{\mathsf{top}}/S)$ is $\Pi^1_1$-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 $\mathsf{SN}^\omega$ when restricting to finite terms; for the definition of $\mathsf{SN}^\omega$ 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 $\mathsf{SN}^\omega$ for all terms (finite and infinite) is $\Pi^1_1$-complete. It is well-known that for left-linear TRSs the properties $\mathsf{SN}^\omega$ and $\mathsf{SN}^\infty$ 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 $\mathsf{SN}^\infty$ for left-linear TRSs is $\Pi^1_1$-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 $x$ 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 [4] 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 $\Pi^0_2$-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 $\Pi^0_2$-complete.*

*Proof.* $\mathsf{SN}(R)$ is $\Pi^0_2$-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 $\Pi^0_2$-hard. That $\mathsf{SN}(R_{\mathsf{top}}/_{\mathsf{min}}\, S)$ is in $\Pi^0_2$ 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 analyzed the proof theoretic complexity, in term of the arithmetic and analytical hierarchy, of termination properties in term rewriting. The position of $\mathsf{WN}$ and $\mathsf{SN}$ were to be expected, but the position of dependency pair problems is remarkably high. We have shown that (ground) confluence is $\Pi^0_2$-complete both uniform and for single terms. The situation becomes more interesting when we look at weak confluence and weak confluence on ground terms. While the former is $\Sigma^0_1$, the latter turns out to be $\Pi^0_2$-complete. In future work, we will also further study the place in the analytic hierarchy of properties of infinitary rewriting like $\mathsf{WN}^\infty$.

## References


1. Takahito Aoto and Yoshihito Toyama. Persistency of confluence. *J. Universal Computer Science*, 3:1134–1147, 1997.
2. T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000.
3. A. Geser, A. Middeldorp, E. Ohlebusch, and H. Zantema. Relative undecidability in term rewriting part i: The termination hierarchy. *Information and Computation*, 178(1):101–131, 2002.
4. J. Giesl, R. Thiemann, and P. Schneider-Kamp. The dependency pair framework: Combining techniques for automated termination proofs. In Franz Baader and Andrei Voronkov, editors, *Proceedings of LPAR'04*, volume 3452 of *Lecture Notes in Artificial Intelligence*, pages 301–331. Springer, 2005.

5. Gabor T. Herman. Strong computability and variants of the uniform halting problem. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 17(1):115–131, 1971.
6. P.G. Hinman. *Recursion-Theoretic Hierarchies*. Springer, 1978.
7. G. Huet and D. Lankford. On the uniform halting problem for term rewriting systems. Technical Report 283, IRIA, France, Mars 1978.
8. Hartley Rogers Jr. *Theory of recursive functions and effective computability*. Mc Graw Hill, 1967.
9. J. W. Klop and R. C. de Vrijer. Infinitary normalization. In *We Will Show Them! Essays in Honour of Dov Gabbay*, volume 2, pages 169–192. College Publications, 2005.
10. J.W. Klop. Term rewriting systems. In S. Abramsky, Dov M. Gabbay, and S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, pages 1–116. Oxford University Press, Inc., 1992.
11. J.R. Shoenfield. *Mathematical Logic*. Association for Symbolic Logic, by A.K. Peters, 1967.
12. Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.
13. H. Zantema. Termination of term rewriting: interpretation and type elimination. *J. Symb. Comput.*, 17(1):23–50, 1994.
14. H. Zantema. Normalization of infinite terms. In A. Voronkov, editor, *Proceedings of the 19th Conference on Rewriting Techniques and Applications (RTA)*, volume 5117 of *Lecture Notes in Computer Science*, pages 441–455. Springer, 2008.