Journal of Automated Reasoning manuscript No.
(will be inserted by the editor)

Lower Bounds for Runtime Complexity of Term Rewriting

Florian Frohn · Jürgen Giesl · Jera Hensel · Cornelius Aschermann · Thomas Ströder

Abstract We present the first approach to deduce lower bounds for (worst-case) runtime complexity of term rewrite systems (TRSs) automatically. Inferring lower runtime bounds is useful to detect bugs and to complement existing methods that compute upper complexity bounds. Our approach is based on two techniques: The *induction technique* generates suitable families of rewrite sequences and uses induction proofs to find a relation between the length of a rewrite sequence and the size of the first term in the sequence. The *loop detection technique* searches for "decreasing loops". Decreasing loops generalize the notion of loops for TRSs, and allow us to detect families of rewrite sequences with linear, exponential, or infinite length. We implemented our approach in the tool AProVE and evaluated it by extensive experiments.

Keywords Complexity Analysis · Term Rewriting · Induction · Lower Bounds

1 Introduction

There exist numerous methods to infer *upper bounds* for the runtime complexity of TRSs [3,16,19,25,32]. We present the first automatic approach to infer *lower bounds* for their runtime complexity, based on two techniques: the *induction technique* (Sect. 3) and *loop detection* (Sect. 4). *Runtime complexity* [16] refers to the "worst" cases in terms of evaluation length and our goal is to find lower bounds for these cases (i.e., in this paper we are not interested in "best-case lower bounds", but in "worst-case lower bounds"). While upper bounds for runtime complexity help to prove the absence of bugs that worsen the performance of programs, lower bounds can be used to *find* such bugs. Moreover, in combination with methods for upper bounds, our approach can prove *tight* complexity results. In addition to *asymptotic* lower bounds, our induction technique can often compute *concrete* bounds. We implemented our contributions in the tool AProVE [2,15] and demonstrate its power by extensive experiments in Sect. 5. In App. A we briefly discuss the adaption of our techniques for *innermost* rewriting and App. B contains all proofs.

Supported by the DFG grant GI 274/6-1 and the Air Force Research Laboratory (AFRL).

LuFG Informatik 2, RWTH Aachen University, Germany

While most methods to infer upper bounds are adaptions of termination techniques, our approach is related to methods that prove non-termination of TRSs. More precisely, the *induction technique* is inspired by our technique to prove non-termination of (possibly non-looping) TRSs [8]. Both techniques generate "meta-rules" (called *rewrite lemmas* in the present paper) which represent infinitely many rewrite sequences. However, our rewrite lemmas are more general than the meta-rules in [8], as they can be parameterized by *several* variables. *Loop detection* is related to techniques that search for loops (e.g., [12,13,27,29,31,34]), where each loop gives rise to a non-terminating rewrite sequence. Our approach generalizes the notion of loops to *decreasing loops*, which give rise to families of rewrite sequences with linear, exponential, or infinite runtime complexity.

Recently, we also introduced a technique to infer lower bounds on the worst-case runtime complexity of integer programs [10]. In contrast to the current paper, in [10] we do not consider structured data like lists or trees (which can easily be represented in TRSs), but we we handle programs operating on built-in integers, which are not available in TRSs. Hence, the current paper is orthogonal to the work from [10], and a combination of both techniques is subject of future work.

We published a preliminary version of the *induction technique* (for innermost rewriting) in [9]. In the present paper, we adapted the technique to full rewriting and improved it by using a more general form of rewrite lemmas that do not have to express the *exact* length of rewrite sequences anymore, but just a *lower bound*. Consequently, we can now use more general forms of induction proofs to prove rewrite lemmas, which increases the applicability of our approach. Moreover, we included all proofs (which were missing in [9]) and extended the experimental evaluation substantially. The *loop detection* technique of Sect. 4 is completely new.

2 Preliminaries

In this section, we introduce the required notions and notations for term rewriting.

Example 1 (TRS $\mathcal{R}_{\mathsf{plus}}$ for Addition) Consider the following TRS $\mathcal{R}_{\mathsf{plus}}$ for addition.

$$\mathsf{plus}(\mathsf{zero}, y) \to y \qquad \mathsf{plus}(\mathsf{succ}(x), y) \to \mathsf{succ}(\mathsf{plus}(x, y))$$

See, e.g., [5] for the basics of rewriting, where we only consider finite TRSs. $\mathcal{T}(\Sigma, \mathcal{V})$ is the set of all *terms* over a (finite) signature Σ and a set of variables $\mathcal{V}$, and $\mathcal{T}(\Sigma) = \mathcal{T}(\Sigma, \varnothing)$ is the set of ground terms. For any term t, $\mathcal{V}(t)$ is the set of its variables, and for a non-variable term t, $\mathrm{root}(t)$ denotes its top symbol (i.e., $\mathrm{root}(f(t_1, \ldots, t_k)) = f$).

Let $\mathbb{N}^*$ be the set of all finite sequences of natural numbers, where ε is the empty sequence. For any term t, the set $\mathrm{pos}(t) \subseteq \mathbb{N}^*$ of its *positions* is defined by $\mathrm{pos}(x) = \{\varepsilon\}$ for $x \in \mathcal{V}$ and $\mathrm{pos}(f(t_1, \ldots, t_k)) = \{i.\pi \mid 1 \leq i \leq k, \pi \in \mathrm{pos}(t_i)\}$. For $\pi \in \mathrm{pos}(t)$, $t|_\pi$ denotes the subterm of t at position π, where $t|_\varepsilon = t$ and $f(t_1, \ldots, t_k)|_{i.\pi} = t_i|_\pi$. Moreover, $t[s]_\pi$ is the result of replacing t's subterm at position π by s. Thus, $t[s]_\varepsilon = s$ and $f(t_1, \ldots, t_i, \ldots, t_k)[s]_{i.\pi} = f(t_1, \ldots, t_i[s]_\pi, \ldots, t_k)$. As an example, for $t = \mathsf{plus}(\mathsf{succ}(x), y)$ we have $\mathrm{pos}(t) = \{\varepsilon, 1, 1.1, 2\}$, $t|_1 = \mathsf{succ}(x)$, and $t[x]_1 = \mathsf{plus}(x, y)$. We write $t \trianglerighteq s$ iff s is a subterm of t, i.e., iff $t|_\pi = s$ for some $\pi \in \mathrm{pos}(t)$. For two positions ξ and π we have $\xi < \pi$ ("ξ is above π") iff ξ is a proper prefix of π, i.e., iff there is a $\xi' \neq \varepsilon$ with $\xi.\xi' = \pi$. For example, we have $1 < 1.1$.

A *context* C is a term from $\mathcal{T}(\Sigma \uplus \{\Box\}, \mathcal{V})$ which contains exactly one occurrence of the constant $\Box$ (called "hole"). If $C|_\pi = \Box$, then $C[s]$ is a shorthand for $C[s]_\pi$. An example for a context is $C = \mathsf{plus}(\Box, y)$ and we have $C[\mathsf{succ}(x)] = \mathsf{plus}(\mathsf{succ}(x), y)$.

A *substitution* is a function $\sigma : \mathcal{V} \to \mathcal{T}(\Sigma, \mathcal{V})$ whose *domain* $\mathrm{dom}(\sigma) = \{x \mid x \neq \sigma(x)\}$ is finite. A substitution σ with $\mathrm{dom}(\sigma) = \{x_1, \ldots, x_n\}$ can be denoted as $[x_1/\sigma(x_1), \ldots, x_n/\sigma(x_n)]$. The *range* of a substitution σ is $\mathrm{range}(\sigma) = \{x\sigma \mid x \in \mathrm{dom}(\sigma)\}$. As an example, for $\sigma = [x/\mathsf{succ}(x)]$ we have $\mathrm{dom}(\sigma) = \{x\}$, $\mathrm{range}(\sigma) = \{\mathsf{succ}(x)\}$, $\sigma(x) = \mathsf{succ}(x)$, and $\sigma(y) = y$ for all $y \in \mathcal{V} \setminus \{x\}$. Substitutions are extended homomorphically to terms where we often write $t\sigma$ instead of $\sigma(t)$. Thus, we have $\mathsf{plus}(x, y)\sigma = \mathsf{plus}(\mathsf{succ}(x), y)$. For $\mathcal{V}' \subseteq \mathcal{V}$, let $\sigma|_{\mathcal{V}'}$ denote the restriction of σ to $\mathcal{V}'$ where $\sigma|_{\mathcal{V}'}(x) = \sigma(x)$ for $x \in \mathcal{V}'$ and $\sigma|_{\mathcal{V}'}(x) = x$ for $x \in \mathcal{V} \setminus \mathcal{V}'$.

A *TRS* $\mathcal{R}$ is a set of *rules* $\ell \to r$ where $\ell, r \in \mathcal{T}(\Sigma, \mathcal{V})$ such that $\ell \notin \mathcal{V}$ and $\mathcal{V}(r) \subseteq \mathcal{V}(\ell)$. The *rewrite relation* $\to_\mathcal{R}$ is defined as $s \to_\mathcal{R} t$ iff there is a $\pi \in \mathrm{pos}(s)$, a rule $\ell \to r \in \mathcal{R}$, and a substitution σ such that $s|_\pi = \ell\sigma$ and $t = s[r\sigma]_\pi$. Any instance $\ell\sigma$ of a left-hand side of a rule is called a *redex*. For example, we have $\mathsf{plus}(\mathsf{succ}(\mathsf{succ}(\mathsf{zero})), y) \to_{\mathcal{R}_\mathsf{plus}} \mathsf{succ}(\mathsf{plus}(\mathsf{succ}(\mathsf{zero})), y) \to_{\mathcal{R}_\mathsf{plus}} \mathsf{succ}(\mathsf{succ}(\mathsf{plus}(\mathsf{zero}, y))) \to_{\mathcal{R}_\mathsf{plus}} \mathsf{succ}(\mathsf{succ}(y))$. The transitive closure of $\to_\mathcal{R}$ is denoted by $\to_\mathcal{R}^+$ and the transitive-reflexive closure of $\to_\mathcal{R}$ is denoted by $\to_\mathcal{R}^*$ (thus, $\mathsf{plus}(\mathsf{succ}(\mathsf{succ}(\mathsf{zero})), y) \to^+_{\mathcal{R}_\mathsf{plus}} \mathsf{succ}(\mathsf{succ}(y))$ and $\mathsf{plus}(\mathsf{succ}(\mathsf{succ}(\mathsf{zero})), y) \to^*_{\mathcal{R}_\mathsf{plus}} \mathsf{succ}(\mathsf{succ}(y))$). Moreover for $n \in \mathbb{N}$, $\to^n_\mathcal{R}$ denotes the n-fold application of $\to_\mathcal{R}$ (e.g., $\mathsf{plus}(\mathsf{succ}(\mathsf{succ}(\mathsf{zero})), y) \to^3_{\mathcal{R}_\mathsf{plus}} \mathsf{succ}(\mathsf{succ}(y))$). A term t is in *normal form* w.r.t. $\to_\mathcal{R}$ iff there is no term t' with $t \to_\mathcal{R} t'$. We say that t is a normal form of s iff $s \to^*_\mathcal{R} t$ and t is in normal form. If the normal form t of s is unique, we write $s{\downarrow_\mathcal{R}} = t$. In our example, we have $\mathsf{plus}(\mathsf{succ}(\mathsf{succ}(\mathsf{zero})), y){\downarrow_\mathcal{R}} = \mathsf{succ}(\mathsf{succ}(y))$. Finally, a term is *linear* if it does not contain multiple occurrences of the same variable (thus, $\mathsf{plus}(\mathsf{succ}(x), y)$ is linear while $\mathsf{plus}(\mathsf{succ}(x), x)$ is not). A TRS is *left-linear* iff it only contains rules $\ell \to r$ where ℓ is a linear term. A TRS is *linear* iff for every rule $\ell \to r$, both ℓ and r are linear terms.

Moreover, we also consider rewriting *modulo* a set of equations $\mathcal{E}$ of the form $\ell = r$ with $\ell, r \in \mathcal{T}(\Sigma, \mathcal{V})$. We write $s \equiv_\mathcal{E} t$ as a shorthand for $\mathcal{E} \models s = t$, i.e., $s \equiv_\mathcal{E} t$ means that the equation $s = t$ is true in all models of $\mathcal{E}$. For a TRS $\mathcal{R}$ and a set of equations $\mathcal{E}$, the *rewrite relation of* $\mathcal{R}$ *modulo* $\mathcal{E}$ is defined as $\to_{\mathcal{R}/\mathcal{E}} = \equiv_\mathcal{E} \circ \to_\mathcal{R} \circ \equiv_\mathcal{E}$. Similarly, we also define *narrowing modulo equations* by performing unification instead of matching when applying rewrite rules: $s \rightsquigarrow_{\mathcal{R}/\mathcal{E}} t$ holds iff there is a term s', a $\pi \in \mathrm{pos}(s')$ with $s'|_\pi \notin \mathcal{V}$, a variable-renamed rule $\ell \to r \in \mathcal{R}$, and a substitution σ such that $s\sigma \equiv_\mathcal{E} s'\sigma$, $s'|_\pi\, \sigma = \ell\sigma$, and $s'[r]_\pi\, \sigma \equiv_\mathcal{E} t$.

Our goal is to infer lower bounds for the *runtime complexity* $\mathrm{rc}_\mathcal{R}$ of a TRS $\mathcal{R}$. To define $\mathrm{rc}_\mathcal{R}$, the *derivation height* of a term t w.r.t. a relation $\to$ is the length of the longest $\to$-sequence starting with t, i.e., $\mathrm{dh}(t, \to) = \sup\{\, m \mid \exists t' \in \mathcal{T}(\Sigma, \mathcal{V}).\, t \to^m t' \,\}$, cf. [17,25]. Here, for any $M \subseteq \mathbb{N} \cup \{\omega\}$, $\sup M$ is the least upper bound of M and $\sup \varnothing = 0$. Since we only regard finite TRSs, $\mathrm{dh}(t, \to_\mathcal{R}) = \omega$ iff t starts an infinite sequence of $\to_\mathcal{R}$-steps. Hence as in [25], dh treats terminating and non-terminating terms in a uniform way. In our example, $\mathrm{dh}(\mathsf{plus}(\mathsf{succ}^{n_1}(\mathsf{zero}), \mathsf{succ}^{n_2}(\mathsf{zero})), \to_{\mathcal{R}_\mathsf{plus}}) = n_1 + 1$ for all $n_1, n_2 \geq 0$, because of the rewrite sequence

$$\mathsf{plus}(\mathsf{succ}^{n_1}(\mathsf{zero}), \mathsf{succ}^{n_2}(\mathsf{zero})) \to^{n_1}_{\mathcal{R}_\mathsf{plus}} \mathsf{succ}^{n_1}(\mathsf{plus}(\mathsf{zero}, \mathsf{succ}^{n_2}(\mathsf{zero}))) \to_{\mathcal{R}_\mathsf{plus}} \mathsf{succ}^{n_1+n_2}(\mathsf{zero}).$$

The *defined symbols* of a TRS $\mathcal{R}$ are $\Sigma_{def}(\mathcal{R}) = \{\, \mathrm{root}(\ell) \mid \ell \to r \in \mathcal{R} \,\}$ and the *constructors* $\Sigma_{con}(\mathcal{R})$ are all other function symbols in $\mathcal{R}$. Thus, $\Sigma_{def}(\mathcal{R}_\mathsf{plus}) =$

$\{\mathsf{plus}\}$ and $\Sigma_{con}(\mathcal{R}_{\mathsf{plus}}) = \{\mathsf{zero}, \mathsf{succ}\}$. When analyzing the complexity of *programs*, one is interested in evaluations of *basic terms* $f(t_1, \ldots, t_k)$ where a defined symbol $f \in \Sigma_{def}(\mathcal{R})$ is applied to data objects $t_1, \ldots, t_k \in \mathcal{T}(\Sigma_{con}(\mathcal{R}), \mathcal{V})$. We define $\mathcal{T}_B$ to be the set of all basic terms. Thus, in our example we have $\mathcal{T}_B = \{\mathsf{plus}(\mathsf{succ}^{n_1}(t_1), \mathsf{succ}^{n_2}(t_2)) \mid t_1, t_2 \in \mathcal{V} \cup \{\mathsf{zero}\}, n_1, n_2 \geq 0\}$. The *runtime complexity function* $\mathrm{rc}_{\mathcal{R}}$ corresponds to the usual notion of "complexity" for programs. It maps any $n \in \mathbb{N}$ to the length of the longest sequence of $\to_{\mathcal{R}}$-steps starting with a basic term t with $|t| \leq n$. Here, the *size* of a term is $|x| = 1$ for $x \in \mathcal{V}$ and $|f(t_1, \ldots, t_k)| = 1 + |t_1| + \ldots + |t_k|$. Hence in our example, we have $|\,\mathsf{plus}(\mathsf{succ}^{n_1}(t_1), \mathsf{succ}^{n_2}(t_2))\,| = n_1 + n_2 + 3$ for all $t_1, t_2 \in \mathcal{V} \cup \{\mathsf{zero}\}$.

Definition 2 (Runtime Complexity $\mathrm{rc}_{\mathcal{R}}$ [16, 25]) *For a TRS $\mathcal{R}$, its* runtime complexity function $\mathrm{rc}_{\mathcal{R}} : \mathbb{N} \to \mathbb{N} \cup \{\omega\}$ *is* $\mathrm{rc}_{\mathcal{R}}(n) = \sup\{\, \mathrm{dh}(t, \to_{\mathcal{R}}) \mid t \in \mathcal{T}_B, |t| \leq n \,\}$.

In our example, for $n \geq 3$ we have $\mathrm{rc}_{\mathcal{R}_{\mathsf{plus}}}(n) = n - 2$. The reason is that the basic term of size at most n with the longest possible derivation is, e.g., $\mathsf{plus}(\mathsf{succ}^{n-3}(\mathsf{zero}), \mathsf{zero})$ with $\mathsf{plus}(\mathsf{succ}^{n-3}(\mathsf{zero}), \mathsf{zero}) \to_{\mathcal{R}_{\mathsf{plus}}}^{n-2} \mathsf{succ}^{n-3}(\mathsf{zero})$. This implies $\mathrm{rc}_{\mathcal{R}_{\mathsf{plus}}}(n) \in \mathcal{O}(n)$ and $\mathrm{rc}_{\mathcal{R}_{\mathsf{plus}}}(n) \in \Omega(n)$, i.e., $\mathrm{rc}_{\mathcal{R}_{\mathsf{plus}}}(n) \in \Theta(n)$. While there exist several techniques and tools to compute upper bounds for $\mathrm{rc}_{\mathcal{R}}$ (e.g., to infer $\mathrm{rc}_{\mathcal{R}_{\mathsf{plus}}}(n) \in \mathcal{O}(n)$), the goal of the present paper is to compute lower bounds for $\mathrm{rc}_{\mathcal{R}}$ (e.g., to infer $\mathrm{rc}_{\mathcal{R}_{\mathsf{plus}}}(n) \in \Omega(n)$). In this way, one can check whether the deduced asymptotic upper bounds are *tight* (e.g., our implementation in AProVE [2] can now prove $\mathrm{rc}_{\mathcal{R}_{\mathsf{plus}}}(n) \in \Theta(n)$). Moreover, such lower bounds have important applications for security analysis, e.g., to detect possible denial-of-service attacks.

Note that $\mathrm{rc}_{\mathcal{R}}$ refers to the "worst" cases in terms of evaluation length. Alternatively, one could also consider lower bounds for the "best" cases in terms of evaluation length. However, for many algorithms, these best cases correspond to (trivial) base cases and thus, these "best-case lower bounds" are not always useful. To illustrate this, consider the TRS $\mathcal{R}_{\mathsf{plus}}$ again. For $n \geq 3$, the basic term of size n with the shortest possible derivation is, e.g., $\mathsf{plus}(x, \mathsf{succ}^{n-3}(\mathsf{zero}))$. Since this term is in normal form, the resulting "best-case lower bound" is 0. If one only considers evaluations of basic *ground* terms, then the basic term of size n with the shortest evaluation is, e.g., $\mathsf{plus}(\mathsf{zero}, \mathsf{succ}^{n-3}(\mathsf{zero}))$ and the "best-case lower bound" is 1.

3 Induction Technique to Infer Lower Bounds

In this section, we present our first approach to generate lower bounds for $\mathrm{rc}_{\mathcal{R}}$ (by the so-called *induction technique*). To illustrate the idea, consider the following TRS $\mathcal{R}_{\mathsf{qs}}$ for *Quicksort*.[1] The auxiliary function $\mathsf{low}(x, xs)$ returns those elements from the list xs that are smaller than x (and high works analogously). To ease readability, we use infix notation for the function symbols $\leq$ and $\mathsf{++}$.

[1] This TRS corresponds to the example "`Rubio_04/quick.xml`" from the *Termination Problem Data Base* (*TPDB*) used in the annual *Termination Competition* [28].

Example 3 (TRS $\mathcal{R}_{\mathsf{qs}}$ for Quicksort)

$$
\begin{aligned}
\mathsf{qs}(\mathsf{nil}) &\to \mathsf{nil} && (1)\\
\mathsf{qs}(\mathsf{cons}(x, xs)) &\to \mathsf{qs}(\mathsf{low}(x, xs)) \mathbin{+\!\!+} \mathsf{cons}(x, \mathsf{qs}(\mathsf{high}(x, xs))) && (2)\\
\mathsf{low}(x, \mathsf{nil}) &\to \mathsf{nil}\\
\mathsf{low}(x, \mathsf{cons}(y, ys)) &\to \mathsf{ifLow}(x \leq y, x, \mathsf{cons}(y, ys))\\
\mathsf{ifLow}(\mathsf{true}, x, \mathsf{cons}(y, ys)) &\to \mathsf{low}(x, ys)\\
\mathsf{ifLow}(\mathsf{false}, x, \mathsf{cons}(y, ys)) &\to \mathsf{cons}(y, \mathsf{low}(x, ys))\\
\mathsf{high}(x, \mathsf{nil}) &\to \mathsf{nil}\\
\mathsf{high}(x, \mathsf{cons}(y, ys)) &\to \mathsf{ifHigh}(x \leq y, x, \mathsf{cons}(y, ys))\\
\mathsf{ifHigh}(\mathsf{true}, x, \mathsf{cons}(y, ys)) &\to \mathsf{cons}(y, \mathsf{high}(x, ys))\\
\mathsf{ifHigh}(\mathsf{false}, x, \mathsf{cons}(y, ys)) &\to \mathsf{high}(x, ys)\\
\mathsf{zero} \leq x &\to \mathsf{true}\\
\mathsf{succ}(x) \leq \mathsf{zero} &\to \mathsf{false}\\
\mathsf{succ}(x) \leq \mathsf{succ}(y) &\to x \leq y\\
\mathsf{nil} \mathbin{+\!\!+} ys &\to ys && (3)\\
\mathsf{cons}(x, xs) \mathbin{+\!\!+} ys &\to \mathsf{cons}(x, xs \mathbin{+\!\!+} ys)
\end{aligned}
$$

For any $n \in \mathbb{N}$, let $\gamma_{\mathbf{List}}(n)$ be the term $\overbrace{\mathsf{cons}(\mathsf{zero}, \ldots, \mathsf{cons}(\mathsf{zero}, \mathsf{nil})\ldots)}^{n \text{ times}}$, i.e., the list of length n where all elements have the value zero *(we also use the notation "$\mathsf{cons}^n(\mathsf{zero}, \mathsf{nil})$"). To find lower bounds, we show how to automatically generate* rewrite lemmas *that describe families of rewrite sequences. For example, our induction technique infers the following rewrite lemma automatically.*

$$\mathsf{qs}(\gamma_{\mathbf{List}}(n)) \quad \to^{\geq 3n^2+2n+1} \quad \gamma_{\mathbf{List}}(n) \tag{4}$$

The rewrite lemma means that for any $n \in \mathbb{N}$, there is a rewrite sequence of at least length $3n^2+2n+1$ that reduces $\mathsf{qs}(\mathsf{cons}^n(\mathsf{zero}, \mathsf{nil}))$ to $\mathsf{cons}^n(\mathsf{zero}, \mathsf{nil})$. From this rewrite lemma, our technique concludes that the runtime of $\mathcal{R}_{\mathsf{qs}}$ is at least quadratic.

Sect. 3.1 introduces the concepts of *rewrite lemmas* and *generator functions* like $\gamma_{\mathbf{List}}$. Sect. 3.2 shows how our implementation automatically speculates conjectures that may result in rewrite lemmas. In Sect. 3.3, we explain how to verify speculated conjectures automatically by induction. From these induction proofs, one can deduce information on the lengths of the rewrite sequences that are represented by a rewrite lemma, cf. Sect. 3.4. Thus, the use of induction to infer lower runtime bounds is a novel application for automated inductive theorem proving. This complements our earlier work on using induction proofs for termination analysis [11]. Finally, Sect. 3.5 shows how rewrite lemmas are used to infer lower bounds for the runtime complexity of a whole TRS.

3.1 Generator Functions and Rewrite Lemmas

Our approach is based on rewrite lemmas containing *generator functions* such as $\gamma_{\mathbf{List}}$ for types like **List**. Thus, in the first step of our approach we compute suitable types for the TRS $\mathcal{R}$ to be analyzed. Ordinary TRSs do not have any type annotations or built-in types, but they are defined over untyped signatures Σ. Def. 4 extends them with types (see, e.g., [11,19,33]), where for simplicity, here we restrict ourselves to monomorphic types.

Definition 4 (Typing) *Let Σ be an (untyped) signature. A many-sorted signature Σ' is a* typed variant *of Σ if it contains the same function symbols as Σ, with the same arities. Similarly, in a typed variant $\mathcal{V}'$ of the variables $\mathcal{V}$, every variable has a type τ. We always assume that for every type τ, $\mathcal{V}'$ contains infinitely many variables of type τ. Given Σ' and $\mathcal{V}'$, $t \in \mathcal{T}(\Sigma, \mathcal{V})$ is a* well-typed *term of type τ w.r.t. $\Sigma', \mathcal{V}'$ iff*

- *$t \in \mathcal{V}'$ is a variable of type τ or*
- *$t = f(t_1, \ldots, t_k)$ with $k \geq 0$, where each t_i is a well-typed term of type τ_i, and where $f \in \Sigma'$ has the type $\tau_1 \times \ldots \times \tau_k \to \tau$.*

A rewrite rule $\ell \to r$ is well typed iff ℓ and r are well-typed terms of the same type. A TRS is well typed iff all of its rules are well typed.[2]

For any TRS $\mathcal{R}$, a standard type inference algorithm (e.g., [23]) can compute a typed variant Σ' such that $\mathcal{R}$ is well typed. Here, we compute typed variants where the set of terms is decomposed into as many types as possible (i.e., where as few terms as possible are considered to be "well typed"). Thus, to make $\mathcal{R}_{\mathsf{qs}}$ from Ex. 3 well typed, we obtain a typed variant of its signature with the types **Nats**, **Bool**, and **List**, where the function symbols have the following types:

$$\begin{array}{rr}
\mathsf{nil} : \mathbf{List} & \mathsf{qs} : \mathbf{List} \to \mathbf{List} \\
\mathsf{cons} : \mathbf{Nats} \times \mathbf{List} \to \mathbf{List} & \mathsf{++} : \mathbf{List} \times \mathbf{List} \to \mathbf{List} \\
\mathsf{zero} : \mathbf{Nats} & \leq : \mathbf{Nats} \times \mathbf{Nats} \to \mathbf{Bool} \\
\mathsf{succ} : \mathbf{Nats} \to \mathbf{Nats} & \mathsf{low}, \mathsf{high} : \mathbf{Nats} \times \mathbf{List} \to \mathbf{List} \\
\mathsf{true}, \mathsf{false} : \mathbf{Bool} & \mathsf{ifLow}, \mathsf{ifHigh} : \mathbf{Bool} \times \mathbf{Nats} \times \mathbf{List} \to \mathbf{List}
\end{array}$$

A type τ *depends* on τ' (denoted $\tau \sqsupseteq_{dep} \tau'$) iff $\tau = \tau'$ or if there is a $c \in \Sigma'_{con}(\mathcal{R})$ of type $\tau_1 \times \ldots \times \tau_k \to \tau$ where $\tau_i \sqsupseteq_{dep} \tau'$ for some $1 \leq i \leq k$. For example, we have $\mathbf{List} \sqsupseteq_{dep} \mathbf{Nats}$. To ease the presentation, we do not allow mutually recursive types (i.e., if $\tau \sqsupseteq_{dep} \tau'$ and $\tau' \sqsupseteq_{dep} \tau$, then $\tau' = \tau$).

To represent families of terms, we now introduce generator functions γ_τ. For any $n \in \mathbb{N}$, $\gamma_\tau(n)$ is a term from $\mathcal{T}(\Sigma'_{con}(\mathcal{R}))$ where a recursive constructor of type τ is nested n times. A constructor $c : \tau_1 \times \ldots \times \tau_k \to \tau$ is called *recursive* iff $\tau_i = \tau$ for some $1 \leq i \leq k$. For **Nats** above, we have $\gamma_{\mathbf{Nats}}(0) = \mathsf{zero}$ and $\gamma_{\mathbf{Nats}}(n+1) = \mathsf{succ}(\gamma_{\mathbf{Nats}}(n))$. If a constructor has a non-recursive argument of type τ', then γ_τ instantiates this argument by $\gamma_{\tau'}(0)$. For **List**, we get $\gamma_{\mathbf{List}}(0) = \mathsf{nil}$ and $\gamma_{\mathbf{List}}(n+1) = \mathsf{cons}(\gamma_{\mathbf{Nats}}(0), \gamma_{\mathbf{List}}(n)) = \mathsf{cons}(\mathsf{zero}, \gamma_{\mathbf{List}}(n))$. If a constructor has several recursive arguments, then several generator functions are possible. For a type **Tree** with the constructors $\mathsf{leaf} : \mathbf{Tree}$ and $\mathsf{node} : \mathbf{Tree} \times \mathbf{Tree} \to \mathbf{Tree}$, we have $\gamma_{\mathbf{Tree}}(0) = \mathsf{leaf}$, and $\gamma_{\mathbf{Tree}}(n+1) = \mathsf{node}(\gamma_{\mathbf{Tree}}(n), \mathsf{leaf})$ or $\gamma_{\mathbf{Tree}}(n+1) = \mathsf{node}(\mathsf{leaf}, \gamma_{\mathbf{Tree}}(n))$. Similarly, if a type has several non-recursive or recursive constructors, then different generator functions can be obtained by considering all combinations of non-recursive and recursive constructors.

To ease readability, we only consider generator functions for *simply structured* types τ. Such types have exactly two constructors $c, d \in \Sigma'_{con}(\mathcal{R})$, where c is not recursive, d has exactly one argument of type τ, and each argument type $\tau' \neq \tau$ of

[2] W.l.o.g., here one may rename the variables in every rule. Then it is not a problem if the variable x is used with type τ_1 in one rule and with type τ_2 in another rule. Requiring that ℓ and r have the same type ensures that rewriting transforms any well-typed term of type τ into a well-typed term of the same type τ.

c or d is simply structured, too. Our approach is easily extended to more complex types by heuristically choosing one of the possible generator functions.[3]

Definition 5 (Generator Functions and Equations) *Let $\mathcal{R}$ be a TRS that is well typed w.r.t. Σ' and $\mathcal{V}'$. We extend the set of types by a fresh type $\mathbb{N}$. For every type $\tau \neq \mathbb{N}$, let γ_τ be a fresh* generator function symbol *of type $\mathbb{N} \to \tau$. The set $\mathcal{G}_\mathcal{R}$ consists of the following* generator equations *for every simply structured type τ with the constructors $c : \tau_1 \times \ldots \times \tau_k \to \tau$ and $d : \rho_1 \times \ldots \times \rho_b \to \tau$, where $\rho_j = \tau$.*

$$\begin{aligned}\gamma_\tau(0) &= c(\gamma_{\tau_1}(0), \ldots, \gamma_{\tau_k}(0))\\ \gamma_\tau(n+1) &= d(\gamma_{\rho_1}(0), \ldots, \gamma_{\rho_{j-1}}(0), \gamma_\tau(n), \gamma_{\rho_{j+1}}(0), \ldots, \gamma_{\rho_b}(0))\end{aligned}$$

We write $\mathcal{G}$ instead of $\mathcal{G}_\mathcal{R}$ if $\mathcal{R}$ is clear from the context.

We extend $\sqsupseteq_{dep}$ to $\Sigma_{def}(\mathcal{R})$ by defining $f \sqsupseteq_{dep} h$ iff $f = h$ or if there is a rule $f(\ldots) \to r$ and a symbol g in r with $g \sqsupseteq_{dep} h$. For example, we have $\mathsf{qs} \sqsupseteq_{dep} \mathsf{low}$. When speculating conjectures, we take the dependencies between defined symbols into account. If $f \sqsupseteq_{dep} g$ and $g \not\sqsupseteq_{dep} f$, then we first generate a rewrite lemma for g. This lemma can be used when generating a lemma for f afterwards.

For $f \in \Sigma'_{def}(\mathcal{R})$ of type $\tau_1 \times \ldots \times \tau_k \to \tau$ with simply structured types $\tau_1, \ldots, \tau_k$, our goal is to speculate a *conjecture* of the form $f(\gamma_{\tau_1}(s_1), \ldots, \gamma_{\tau_k}(s_k)) \to^* t$, where $s_1, \ldots, s_k$ are polynomials over some variables $n_1, \ldots, n_m$ of type $\mathbb{N}$. Moreover, t is a term built from Σ, polynomials over $n_1, \ldots, n_m$, and generator functions. After speculating such a conjecture (in Sect. 3.2), in Sect. 3.3 we prove its *validity*.

Definition 6 (Validity of Conjectures) *Let $\mathcal{R}$ be a well-typed TRS w.r.t. Σ' and $\mathcal{V}'$, and let $\mathcal{A}$ be the infinite set of all valid equalities in the theory of $\mathbb{N}$ with addition and multiplication. For any term q, let $q{\downarrow_{\mathcal{G}/\mathcal{A}}}$ be q's normal form w.r.t. $\mathcal{G}_\mathcal{R}$, where the generator equations are applied from left to right and $\mathcal{A}$-equivalent (sub)terms are considered to be equal.*[4] *Let $s \to^* t$ be a conjecture with $\mathcal{V}(s) = \{n_1, \ldots, n_m\} \neq \varnothing$, where $\overline{n} = (n_1, \ldots, n_m)$ are pairwise different variables of type $\mathbb{N}$, s is well typed, $\mathrm{root}(s) \in \Sigma_{def}(\mathcal{R})$, and s has no symbol from $\Sigma_{def}(\mathcal{R})$ below the root. Then the conjecture $s \to^* t$ is* valid *for $\mathcal{R}$ iff $s\sigma{\downarrow_{\mathcal{G}/\mathcal{A}}} \to^*_\mathcal{R} t\sigma{\downarrow_{\mathcal{G}/\mathcal{A}}}$ holds for all $\sigma : \mathcal{V}(s) \to \mathbb{N}$.*

For instance, the conjecture $\mathsf{qs}(\gamma_{\mathbf{List}}(n)) \to^* \gamma_{\mathbf{List}}(n)$ is valid for $\mathcal{R}_\mathsf{qs}$, as $\sigma(n) = b \in \mathbb{N}$ implies $\mathsf{qs}(\gamma_{\mathbf{List}}(b)){\downarrow_{\mathcal{G}/\mathcal{A}}} = \mathsf{qs}(\mathsf{cons}^b(\mathsf{zero}, \mathsf{nil})) \to^* \mathsf{cons}^b(\mathsf{zero}, \mathsf{nil}) = \gamma_{\mathbf{List}}(b){\downarrow_{\mathcal{G}/\mathcal{A}}}$.

From a valid conjecture $f(\gamma_{\tau_1}(s_1), \ldots, \gamma_{\tau_k}(s_k)) \to^* t$, we then infer a *rewrite lemma* $f(\gamma_{\tau_1}(s_1), \ldots, \gamma_{\tau_k}(s_k)) \to^{\geq rt(n_1,\ldots,n_m)} t$ in Sect. 3.4, whose *runtime* function $rt : \mathbb{N}^m \to \mathbb{N}$ describes a lower bound for the length of the corresponding evaluation.

[3] For types with several recursive or non-recursive constructors, our heuristic prefers to use those constructors for the generator equations that occur in the left-hand sides of (preferably recursive) rules of the TRS. For a constructor with several recursive argument positions like node, we examine how often the TRS contains recursive calls in the respective arguments of node. If there are more recursive calls in the first arguments of node than in the second one, then we take the generator equation $\gamma_{\mathbf{Tree}}(n+1) = \mathsf{node}(\gamma_{\mathbf{Tree}}(n), \mathsf{leaf})$ instead of $\gamma_{\mathbf{Tree}}(n+1) = \mathsf{node}(\mathsf{leaf}, \gamma_{\mathbf{Tree}}(n))$.

[4] Termination of $\to_{\mathcal{G}/\mathcal{A}}$ follows from the definition of generator equations (Def. 5), since the argument of γ_τ decreases with each application of an equation and since we excluded mutually recursive types.

Definition 7 (Rewrite Lemma) *Let $\mathcal{R}$, s, t, $\overline{n}$ be as in Def. 6 and let $rt : \mathbb{N}^m \to \mathbb{N}$ be weakly monotonic (i.e., $n_i \geq n_i'$ implies $rt(n_1, ..., n_i, ..., n_m) \geq rt(n_1, ..., n_i', ..., n_m)$). Then $s \to^{\geq rt(\overline{n})} t$ is a* rewrite lemma *for $\mathcal{R}$ iff $s\sigma{\downarrow_{\mathcal{G}/\mathcal{A}}} \to_{\mathcal{R}}^{\geq rt(\overline{n}\sigma)} t\sigma{\downarrow_{\mathcal{G}/\mathcal{A}}}$ holds for all $\sigma : \mathcal{V}(s) \to \mathbb{N}$, i.e., $s\sigma{\downarrow_{\mathcal{G}/\mathcal{A}}}$ reduces to $t\sigma{\downarrow_{\mathcal{G}/\mathcal{A}}}$ in at least $rt(n_1\sigma, \ldots, n_m\sigma)$ $\mathcal{R}$-steps.*

The reason for the restriction to weakly monotonic runtime functions rt is that in this way, one can infer a suitable bound from the induction proof of a conjecture $s \to^* t$, cf. Sect. 3.4.

3.2 Speculating Conjectures

We now show how to speculate conjectures $f(\gamma_{\tau_1}(s_1), \ldots, \gamma_{\tau_k}(s_k)) \to^* t$ (whose validity must be proved afterwards in Sect. 3.3). While lemma speculation was investigated in inductive theorem proving and verification since decades [6], we want to find lemmas of a special form in order to extract suitable lower bounds from their induction proofs.

Of course, our algorithm for the speculation of conjectures is just one possible implementation for this task. The soundness of our approach (i.e., the correctness of the theorems and lemmas in Sect. 3.3 – 3.5) is independent of the specific implementation that is used for the speculation of conjectures.

To speculate a conjecture for a function f, we first generate *sample conjectures* that describe the effect of applying f to specific arguments. To obtain them, we narrow $f(\gamma_{\tau_1}(n_1), \ldots, \gamma_{\tau_k}(n_k))$ where $n_1, \ldots, n_k \in \mathcal{V}$, using the rules of the TRS and the lemmas we have proven so far, taking also the generator equations and integer arithmetic into account. This narrowing corresponds to a case analysis over the possible derivations.

For any proven rewrite lemma $s \to^{\geq rt(\cdots)} t$, let the set $\mathcal{L}$ contain the rule $s \to t$. Then let "$s \rightsquigarrow t$" be a shorthand for "$s \rightsquigarrow_{(\mathcal{R}\cup\mathcal{L})/(\mathcal{G}\cup\mathcal{A})} t$". Thus, as explained in Sect. 2, $s \rightsquigarrow t$ holds if there is a term s', a position π with $s'|_\pi \notin \mathcal{V}$, a variable-renamed rule $\ell \to r \in \mathcal{R} \cup \mathcal{L}$, and a substitution σ that maps variables of type $\mathbb{N}$ to polynomials, such that $s\sigma \equiv_{\mathcal{G}\cup\mathcal{A}} s'\sigma$, $s'|_\pi\sigma = \ell\sigma$, and $s'[r]_\pi\sigma \equiv_{\mathcal{G}\cup\mathcal{A}} t$. For instance, we have $\mathsf{qs}(\gamma_{\mathbf{List}}(n)) \rightsquigarrow \mathsf{nil}$ using the substitution $[n/0]$ and Rule (1).

Although checking $s\sigma \equiv_{\mathcal{G}\cup\mathcal{A}} s'\sigma$ is undecidable for general equations $\mathcal{G}$, equational unification is decidable in quadratic time for the generator equations of Def. 5 [22]. Moreover, the resulting integer constraints can usually be solved easily by SMT solvers. Thus, due to the restricted form of generator equations in Def. 5, the required narrowing works reasonably efficient in practice.

Example 8 (Narrowing) In Ex. 3 we have $\mathsf{qs} \sqsupseteq_{dep} \mathsf{low}$ and $\mathsf{qs} \sqsupseteq_{dep} \mathsf{high}$. If the lemmas

$$\mathsf{low}(\gamma_{\mathbf{Nats}}(0), \gamma_{\mathbf{List}}(n)) \to^{\geq 3n+1} \gamma_{\mathbf{List}}(0) \quad (5) \qquad \mathsf{high}(\gamma_{\mathbf{Nats}}(0), \gamma_{\mathbf{List}}(n)) \to^{\geq 3n+1} \gamma_{\mathbf{List}}(n) \quad (6)$$

were already proved, then the narrowing tree in Fig. 1 can be generated to find sample conjectures for qs. The arrows are labeled with the rules and substitutions used for variables of type $\mathbb{N}$. To save space, some arrows correspond to several *narrowing steps. The goal is to get representative rewrite sequences, not to cover all reductions.*

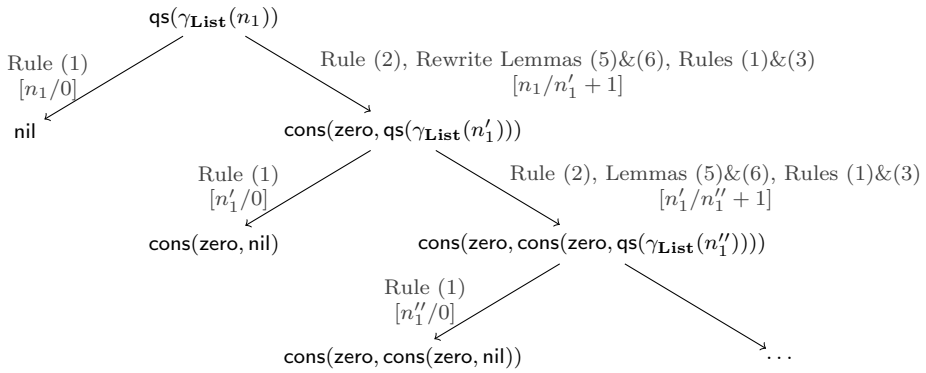

Fig. 1: Narrowing Tree

Hence, we stop constructing the tree after some steps and choose suitable narrowings heuristically.[5]

After constructing a narrowing tree for f, we collect *sample points* (t, σ, d). Here, t results from a leaf q of the tree which is in $\rightsquigarrow$-normal form by normalizing q w.r.t. the generator equations $\mathcal{G}$ applied from right to left.[6] Thus, terms from $\mathcal{T}(\Sigma, \mathcal{V})$ are rewritten to generator symbols with polynomials as arguments. Moreover, σ is the substitution for variables of type $\mathbb{N}$ on the path from the root to q, and d is the number of applications of recursive f-rules on the path (the *recursion depth*). A rule $f(\ldots) \to r$ is *recursive* iff r contains a symbol g with $g \sqsupseteq_{dep} f$.

Example 9 (Sample Points) In Ex. 8, we obtain the following set of sample points:[7]

$$S = \{\, (\gamma_{\mathbf{List}}(0), [n_1/0], 0), \quad (\gamma_{\mathbf{List}}(1), [n_1/1], 1), \quad (\gamma_{\mathbf{List}}(2), [n_1/2], 2)\,\} \tag{7}$$

The sequence from $\mathsf{qs}(\gamma_{\mathbf{List}}(n_1))$ *to* nil *does not use recursive* qs*-rules. Hence, its recursion depth is 0 and the* $\rightsquigarrow$*-normal form* nil *rewrites to* $\gamma_{\mathbf{List}}(0)$ *when applying the generator equations* $\mathcal{G}$ *from right to left. The sequence from* $\mathsf{qs}(\gamma_{\mathbf{List}}(n_1))$ *to* $\mathsf{cons}(\mathsf{zero}, \mathsf{nil})$ *(resp.* $\mathsf{cons}(\mathsf{zero}, \mathsf{cons}(\mathsf{zero}, \mathsf{nil}))$*) uses the recursive* qs*-rule (2) once (resp. twice), i.e., it has recursion depth 1 (resp. 2). Moreover, this* $\rightsquigarrow$*-normal form rewrites to* $\gamma_{\mathbf{List}}(1)$ *(resp.* $\gamma_{\mathbf{List}}(2)$*) when applying* $\mathcal{G}$ *from right to left.*

A sample point (t, σ, d) for a narrowing tree with the root $s = f(\ldots)$ represents the *sample conjecture* $s\sigma \to^* t$, which stands for a reduction with d applications of recursive f-rules. For $s = \mathsf{qs}(\gamma_{\mathbf{List}}(n_1))$, the sample points in (7) represent the sample conjectures $\mathsf{qs}(\gamma_{\mathbf{List}}(0)) \to^* \gamma_{\mathbf{List}}(0)$, $\mathsf{qs}(\gamma_{\mathbf{List}}(1)) \to^* \gamma_{\mathbf{List}}(1)$, and

[5] In our implementation, the (breadth-first) construction of a narrowing tree is aborted when reaching depth 50, or when the tree has 250 nodes or 40 leaves. Here, we prefer narrowing with previously proven rewrite lemmas $\mathcal{L}$ (i.e., narrowing with the rules $\mathcal{R}$ is only done for those subterms where no rewrite lemma is applicable). Our implementation generates (at most) one rewrite lemma $f(\ldots) \to^{\geq n(\ldots)} \ldots$ for every $f \in \Sigma_{def}(\mathcal{R})$.

[6] Rewriting with $\{r \to \ell \mid \ell = r \in \mathcal{G}_{\mathcal{R}}\}$ terminates as each rewrite step reduces the number of symbols from $\Sigma_{con}(\mathcal{R})$.

[7] We always simplify arithmetic expressions in terms and substitutions, e.g., the substitution $[n_1/0+1]$ in the second sample point is simplified to $[n_1/1]$.

$\mathsf{qs}(\gamma_{\mathbf{List}}(2)) \to^* \gamma_{\mathbf{List}}(2)$. Now the goal is to find a maximal subset of these sample conjectures whose elements are suitable for generalization. Then, this subset is used to speculate a general conjecture (whose validity must be proved afterwards).

For a narrowing tree with root s, let S_{max} be a maximal subset of all sample points such that for all $(t, \sigma, d), (t', \sigma', d') \in S_{max}$, the sample conjectures $s\sigma \to^* t$ and $s\sigma' \to^* t'$ are identical up to the occurring natural numbers and variable names. For instance, $\mathsf{qs}(\gamma_{\mathbf{List}}(0)) \to^* \gamma_{\mathbf{List}}(0)$, $\mathsf{qs}(\gamma_{\mathbf{List}}(1)) \to^* \gamma_{\mathbf{List}}(1)$, and $\mathsf{qs}(\gamma_{\mathbf{List}}(2)) \to^* \gamma_{\mathbf{List}}(2)$ are identical up to the occurring numbers. To obtain a general conjecture, we replace all numbers in these sample conjectures by polynomials. In our example, we want to speculate a conjecture of the form $\mathsf{qs}(\gamma_{\mathbf{List}}(pol^{left})) \to^* \gamma_{\mathbf{List}}(pol^{right})$. Here, pol^{left} and pol^{right} are polynomials in one variable n (the *induction variable* of the conjecture) that stands for the recursion depth. This facilitates the proof of the resulting conjecture by induction on n.

For any term q, let $\Pi^q_{\mathbb{N}} = \{\pi \in \mathrm{pos}(q) \mid q|_\pi \in \mathbb{N}\}$. Then for each $\pi \in \Pi^{s\sigma}_{\mathbb{N}}$ (resp. $\pi \in \Pi^t_{\mathbb{N}}$) with $(t, \sigma, d) \in S_{max}$, we search for a polynomial pol^{left}_π (resp. pol^{right}_π). To obtain these polynomials, for every $(t, \sigma, d) \in S_{max}$ we generate the constraints

$$\text{“}pol^{left}_\pi(d) = s\sigma|_\pi\text{”} \text{ for all } \pi \in \Pi^{s\sigma}_{\mathbb{N}} \quad \text{and} \quad \text{“}pol^{right}_\pi(d) = t|_\pi\text{”} \text{ for all } \pi \in \Pi^t_{\mathbb{N}}. \quad (8)$$

Here, pol^{left}_π and pol^{right}_π are polynomials with abstract coefficients. If one searches for polynomials of degree e, then the polynomials have the form $c_0 + c_1 \cdot n + \ldots + c_e \cdot n^e$ and the constraints in (8) are linear diophantine equations over the unknown coefficients $c_i \in \mathbb{N}$.[8] These equations are easily solved automatically. Finally, the generalized speculated conjecture is obtained from $s\sigma \to^* t$ by replacing $s\sigma|_\pi$ with pol^{left}_π for every $\pi \in \Pi^{s\sigma}_{\mathbb{N}}$ and by replacing $t|_\pi$ with pol^{right}_π for every $\pi \in \Pi^t_{\mathbb{N}}$.

Example 10 (Speculating Conjectures) *In Ex. 8, we narrowed* $s = \mathsf{qs}(\gamma_{\mathbf{List}}(n_1))$ *and* S_{max} *is the set* S *in (7), cf. Ex. 9. For each* $(t, \sigma, d) \in S_{max}$, $\Pi^{s\sigma}_{\mathbb{N}}$ *only contains the position* 1.1 *and* $\Pi^t_{\mathbb{N}} = \{1\}$. *Hence, from the sample conjecture* $\mathsf{qs}(\gamma_{\mathbf{List}}(0)) \to^* \gamma_{\mathbf{List}}(0)$, *where the recursion depth is* $d = 0$, *we obtain the constraints* $pol^{left}_{1.1}(d) = pol^{left}_{1.1}(0) = \mathsf{qs}(\gamma_{\mathbf{List}}(0))|_{1.1} = 0$ *and* $pol^{right}_1(d) = pol^{right}_1(0) = \gamma_{\mathbf{List}}(0)|_1 = 0$. *Similarly, from the two other sample conjectures we get* $pol^{left}_{1.1}(1) = pol^{right}_1(1) = 1$ *and* $pol^{left}_{1.1}(2) = pol^{right}_1(2) = 2$. *When using* $pol^{left}_{1.1} = c_0 + c_1 \cdot n + c_2 \cdot n^2$ *and* $pol^{right}_1 = d_0 + d_1 \cdot n + d_2 \cdot n^2$ *with the abstract coefficients* $c_0, \ldots, c_2, d_0, \ldots, d_2$, *the solution* $c_0 = c_2 = d_0 = d_2 = 0$, $c_1 = d_1 = 1$ *(i.e.,* $pol^{left}_{1.1} = n$ *and* $pol^{right}_1 = n$*) is easily found automatically. The resulting speculated conjecture is* $\mathsf{qs}(\gamma_{\mathbf{List}}(pol^{left}_{1.1})) \to^* \gamma_{\mathbf{List}}(pol^{right}_1)$, *i.e.,* $\mathsf{qs}(\gamma_{\mathbf{List}}(n)) \to^* \gamma_{\mathbf{List}}(n)$.

If S_{max} contains sample points with e different recursion depths,[9] then there are unique polynomials of at most degree $e - 1$ satisfying the constraints (8). The reason is that the sample points give rise to e independent constraints for the

[8] In the constraints (8), n is instantiated by an actual number d. Thus, if $pol^{left}_\pi = c_0 + c_1 \cdot n + \ldots + c_e \cdot n^e$, then $pol^{left}_\pi(d)$ is a *linear* polynomial over the unknowns $c_0, \ldots, c_e$. While solving the constraints (8) only requires linear integer arithmetic, the resulting rewrite lemmas contain polynomials pol^{left}_π and pol^{right}_π of degree e. Thus, rewriting or narrowing w.r.t. the rewrite lemmas $\mathcal{L}$ may require non-linear integer arithmetic.

[9] In our implementation, whenever 5 new depths of a narrowing tree have been computed, we check how many sample points one can generate from the current tree. If we can create at least 3 sample points with different recursion depths, we stop constructing the narrowing tree.

unknown coefficients of the polynomial, and a polynomial of degree $e-1$ has e coefficients.

Example 11 (Several Variables in Conjecture) We consider the TRS $\mathcal{R}_{\mathsf{plus}}$ from Ex. 1 to show how to speculate conjectures with several *variables. Narrowing $s = \mathsf{plus}(\gamma_{\mathbf{Nats}}(n_1), \gamma_{\mathbf{Nats}}(n_2))$ yields the sample points $(\gamma_{\mathbf{Nats}}(n_2), [n_1/0], 0)$, $(\gamma_{\mathbf{Nats}}(n_2+1), [n_1/1], 1)$, $(\gamma_{\mathbf{Nats}}(n_2+2), [n_1/2], 2)$, and $(\gamma_{\mathbf{Nats}}(n_2+3), [n_1/3], 3)$. For the last three sample points (t, σ, d), the only number in $s\sigma$ is at position 1.1 and the polynomial $pol_{1.1}^{left} = n$ satisfies the constraint $pol_{1.1}^{left}(d) = s\sigma|_{1.1}$. Moreover, the only number in t is at position 1.2 and the polynomial $pol_{1.2}^{right} = n$ satisfies $pol_{1.2}^{right}(d) = t|_{1.2}$. Thus, we speculate the conjecture $\mathsf{plus}(\gamma_{\mathbf{Nats}}(n), \gamma_{\mathbf{Nats}}(n_2)) \to^* \gamma_{\mathbf{Nats}}(n_2+n)$ with the induction variable n.*

Example 12 (Larger Coefficients of Polynomials) The following TRS illustrates how we speculate conjectures where the coefficients of the polynomials are larger than 1.

$$\mathsf{half}(\mathsf{zero}) \to \mathsf{zero} \qquad \mathsf{half}(\mathsf{succ}(\mathsf{succ}(x))) \to \mathsf{succ}(\mathsf{half}(x))$$

By narrowing $s = \mathsf{half}(\gamma_{\mathbf{Nats}}(n_1))$, we obtain the sample points $(\gamma_{\mathbf{Nats}}(0), [n_1/0], 0)$, $(\gamma_{\mathbf{Nats}}(1), [n_1/2], 1)$, $(\gamma_{\mathbf{Nats}}(2), [n_1/4], 2)$. For these sample points (t, σ, d), the only numbers in $s\sigma$ (resp. t) are at position 1.1 (resp. at position 1). The polynomial $pol_{1.1}^{left} = 2 \cdot n$ satisfies the constraint $pol_{1.1}^{left}(d) = s\sigma|_{1.1}$ and the polynomial $pol_{1}^{right} = n$ satisfies $pol_{1}^{right}(d) = t|_{1}$. Hence, we speculate the conjecture $\mathsf{half}(\gamma_{\mathbf{Nats}}(2 \cdot n)) \to^* \gamma_{\mathbf{Nats}}(n)$ with the induction variable n.

Algorithm 13 (Speculating Conjectures) *The following algorithm summarizes our method to speculate conjectures for a TRS $\mathcal{R}$.*

(i) Compute a typed variant of the TRS $\mathcal{R}$ which decomposes $\mathcal{T}(\Sigma, \mathcal{V})$ into as many types as possible.
(ii) For every $f \in \Sigma_{def}(\mathcal{R})$ (starting with the smallest symbols w.r.t. $\sqsupseteq_{dep}$):
  (ii.1) Compute a narrowing tree for $f(\gamma_{\tau_1}(n_1), \ldots, \gamma_{\tau_k}(n_k))$.
  (ii.2) Obtain a maximal set S_{max} of sample points suitable for generalization.
  (ii.3) Generalize the sample conjectures corresponding to S_{max}.
  For this, replace all occurring numbers by polynomials.

3.3 Proving Conjectures

Now we show how to prove the validity of speculated conjectures, cf. Def. 6. To prove validity of a conjecture $s \to^* t$ by induction, we use rewriting with $\rightharpoonup = \to_{(\mathcal{R} \cup \mathcal{L})/(\mathcal{G} \cup \mathcal{A})}$. In the induction step, we try to reduce $s[n/n+1]$ to $t[n/n+1]$, where one may use the rule IH: $s \to t$ as induction hypothesis. Here, the induction variable n must not be instantiated and the remaining variables in IH may only be instantiated by an increasing substitution. A substitution σ is *increasing* iff $\mathcal{A} \models x\sigma \geq x$ holds for all $x \in \mathrm{dom}(\sigma)$. For example, the substitution $\sigma = \{x \,/\, (x+y)\}$ is increasing because $\mathcal{A} \models x + y \geq x$. The restriction to increasing substitutions results in induction proofs that are particularly suitable for inferring runtimes of rewrite lemmas. More precisely, increasing substitutions are necessary to ensure the soundness of the recurrence equations that we will construct for lower bounds in Sect. 3.4.

Thus, for any rule IH: $\ell \to r$ containing only variables of type $\mathbb{N}$ and any $n \in \mathcal{V}$, let $s \mapsto_{\mathrm{IH},n} t$ iff there exist a term s', an increasing substitution σ with $n\sigma = n$, and a position π such that $s \equiv_{\mathcal{G}\cup\mathcal{A}} s'$, $s'|_\pi = \ell\sigma$, and $s'[r\sigma]_\pi \equiv_{\mathcal{G}\cup\mathcal{A}} t$. Let $\rightharpoonup_{\mathrm{IH},n} = \rightharpoonup \cup \mapsto_{\mathrm{IH},n}$. Moreover, $\rightharpoonup^*$ (resp. $\rightharpoonup^*_{\mathrm{IH},n}$) is the union of the transitive-reflexive closure of $\rightharpoonup$ (resp. $\rightharpoonup_{\mathrm{IH},n}$) and $\equiv_{\mathcal{G}\cup\mathcal{A}}$. Thm. 14 shows which rewrite sequences are needed to prove a conjecture $s \to^* t$ by induction on its induction variable n.

Theorem 14 (Proving Conjectures) *Let $\mathcal{R}$, s, t, and $\overline{n}$ be as in Def. 6 and let $n \in \mathcal{V}(s)$. If $s[n/0] \rightharpoonup^* t[n/0]$ and $s[n/n+1] \rightharpoonup^*_{\mathrm{IH},n} t[n/n+1]$, where* IH *is the rule $s \to t$, then the conjecture $s \to^* t$ is valid for $\mathcal{R}$.*

Example 15 (Proof of Conjecture for qs*)* *We continue the analysis of $\mathcal{R}_{\mathsf{qs}}$. As in Ex. 8, assume that we already know the rewrite lemmas (5) and (6). To prove the conjecture $\mathsf{qs}(\gamma_{\mathbf{List}}(n)) \to^* \gamma_{\mathbf{List}}(n)$ from Ex. 10, in the induction base we show $\mathsf{qs}(\gamma_{\mathbf{List}}(0)) \rightharpoonup \gamma_{\mathbf{List}}(0)$ and in the induction step, we obtain the sequence $\mathsf{qs}(\gamma_{\mathbf{List}}(n+1)) \rightharpoonup^* \mathsf{nil} +\!\!+ \mathsf{cons}(\mathsf{zero}, \mathsf{qs}(\gamma_{\mathbf{List}}(n))) \mapsto_{\mathrm{IH},n} \mathsf{nil} +\!\!+ \mathsf{cons}(\mathsf{zero}, \gamma_{\mathbf{List}}(n)) \rightharpoonup \gamma_{\mathbf{List}}(n+1)$.*

Example 16 (Instantiating Non-Induction Variables) *This alternative TRS for addition illustrates why one may have to instantiate non-induction variables in the induction hypothesis when proving conjectures. For that reason, our preliminary version of the induction technique in [9] failed to infer a linear lower bound in this example.*

$$\mathsf{add}(\mathsf{zero}, y) \to y \qquad\qquad \mathsf{add}(\mathsf{succ}(x), y) \to \mathsf{add}(x, \mathsf{succ}(y))$$

The technique of Sect. 3.2 speculates the conjecture $\mathsf{add}(\gamma_{\mathbf{Nats}}(n), \gamma_{\mathbf{Nats}}(n_2)) \to^* \gamma_{\mathbf{Nats}}(n_2+n)$ with the induction variable n. To prove this conjecture, in the induction base we have $\mathsf{add}(\gamma_{\mathbf{Nats}}(0), \gamma_{\mathbf{Nats}}(n_2)) \rightharpoonup \gamma_{\mathbf{Nats}}(n_2+0)$. In the induction step, we obtain $\mathsf{add}(\gamma_{\mathbf{Nats}}(n+1), \gamma_{\mathbf{Nats}}(n_2)) \rightharpoonup \mathsf{add}(\gamma_{\mathbf{Nats}}(n), \gamma_{\mathbf{Nats}}(n_2+1))$. To apply the induction hypothesis IH : $\mathsf{add}(\gamma_{\mathbf{Nats}}(n), \gamma_{\mathbf{Nats}}(n_2)) \to \gamma_{\mathbf{Nats}}(n_2+n)$*, we therefore have to instantiate the non-induction variable n_2 by n_2+1. Clearly, this is an increasing substitution since $n_2+1 \geq n_2$. Thus, the proof of the induction step continues with $\mathsf{add}(\gamma_{\mathbf{Nats}}(n), \gamma_{\mathbf{Nats}}(n_2+1)) \mapsto_{\mathrm{IH},n} \gamma_{\mathbf{Nats}}((n_2+1)+n) \equiv_{\mathcal{A}} \gamma_{\mathbf{Nats}}(n_2+(n+1))$.*

3.4 Inferring Bounds for Rewrite Lemmas

For a valid conjecture $s \to^* t$, we now show how to infer a lower bound on the length of the corresponding rewrite sequences, i.e., how to generate a *rewrite lemma* $s \to^{\geq rt(\overline{n})} t$, cf. Def. 7. More precisely, we show that one can infer a suitable bound from the induction proof of a conjecture $s \to^* t$. If $n \in \overline{n}$ is the induction variable and the induction hypothesis is applied ih times in the induction step, then we get the following recurrence equations for rt where $\widetilde{n}$ is $\overline{n}$ without the variable n:

$$rt(\overline{n}[n/0]) = ib(\widetilde{n}) \qquad \text{and} \qquad rt(\overline{n}[n/n+1]) = ih \cdot rt(\overline{n}) + is(\overline{n}) \tag{9}$$

Here, $ib(\widetilde{n})$ is a lower bound on the length of the reduction $s[n/0]\!\downarrow_{\mathcal{G}/\mathcal{A}} \to^*_{\mathcal{R}} t[n/0]\!\downarrow_{\mathcal{G}/\mathcal{A}}$, which must exist due to the induction base. The addend $is(\overline{n})$ is a lower bound on the length of $s[n/n+1]\!\downarrow_{\mathcal{G}/\mathcal{A}} \to^*_{\mathcal{R}} t[n/n+1]\!\downarrow_{\mathcal{G}/\mathcal{A}}$, but without those subsequences that are covered by the induction hypothesis IH : $s \to t$.

When applying IH, s and t are instantiated by an increasing substitution σ. By the induction hypothesis, each rewrite sequence $s\sigma \to^* t\sigma$ has at least length $rt(\overline{n}\sigma)$. Since σ is *increasing*, we have $\overline{n}\sigma \geq \overline{n}$ when comparing tuples pointwise. As rt is *weakly monotonic*, this implies $rt(\overline{n}\sigma) \geq rt(\overline{n})$. Thus, $rt(\overline{n})$ is a lower bound for the length of the reduction $s\sigma \to^* t\sigma$. Hence, the restriction to weakly monotonic functions rt and increasing substitutions σ allows us to underapproximate $rt(\overline{n}\sigma)$ by $rt(\overline{n})$ in (9), resulting in recurrence equations that are suitable for automation.

For each previous rewrite lemma $s' \to^{\geq rt'(\overline{n}')} t'$ that was used in the proof of the conjecture $s \to^* t$, we assume that rt' is known. Thus, rt' can be used as a lower bound on the length of the rewrite sequences represented by that previous lemma. Then one can obtain ih and is directly from the induction proof of the conjecture. To avoid treating rules and rewrite lemmas separately, in Def. 17 we regard each rule $s \to t \in \mathcal{R} \cup \{\mathrm{IH}\}$ as a rewrite lemma $s \to^{\geq 1} t$.

Definition 17 (ih, ib, is) *Let $s \to^* t$ be a conjecture with an induction proof as in Thm. 14. More precisely, let $u_1 \rightharpoonup \ldots \rightharpoonup u_{b+1}$ be the rewrite sequence $s[n/0] \rightharpoonup^* t[n/0]$ for the induction base and $v_1 \rightharpoonup_{\mathrm{IH},n} \ldots \rightharpoonup_{\mathrm{IH},n} v_{k+1}$ be the rewrite sequence $s[n/n+1] \rightharpoonup^*_{\mathrm{IH},n} t[n/n+1]$ for the induction step, where* IH*: $s \to t$ is applied ih times. For $j \in \{1,\ldots,b\}$, let $\ell_j \to^{\geq rt_j(\overline{y}_j)} r_j$ and σ_j be the rewrite lemma and substitution used to reduce u_j to u_{j+1}. Similarly for $j \in \{1,\ldots,k\}$, let $p_j \to^{\geq rt'_j(\overline{z}_j)} q_j$ and θ_j be the lemma and substitution used to reduce v_j to v_{j+1}. Then we define:*

$$ib(\widetilde{n}) \;=\; \sum\nolimits_{j\in\{1,\ldots,b\}} rt_j(\overline{y}_j\sigma_j) \quad \text{and} \quad is(\overline{n}) \;=\; \sum\nolimits_{j\in\{1,\ldots,k\},\ p_j\to q_j\neq \mathrm{IH}} rt'_j(\overline{z}_j\theta_j)$$

By solving the recurrence equations (9), we can now compute rt explicitly.

Theorem 18 (Explicit Runtime of Rewrite Lemmas) *Let $s \to^* t$ be a conjecture with an induction proof as in Thm. 14, where ih, ib, and is are as in Def. 17. Then $s \to^{\geq rt(\overline{n})} t$ is a rewrite lemma, where $rt(\overline{n}) = ih^n \cdot ib(\widetilde{n}) + \sum_{i=0}^{n-1} ih^{n-1-i} \cdot is(\overline{n}[n/i])$.*

Example 19 (Computing rt for qs*)* Reconsider the induction proof of the conjecture $\mathsf{qs}(\gamma_{\mathbf{List}}(n)) \to^* \gamma_{\mathbf{List}}(n)$ in Ex. 15. The proof of the induction base is $\mathsf{qs}(\gamma_{\mathbf{List}}(0)) \equiv_{\mathcal{G}} \mathsf{qs}(\mathsf{nil}) \to_{\mathcal{R}_{\mathsf{qs}}} \mathsf{nil} \equiv_{\mathcal{G}} \gamma_{\mathbf{List}}(0)$. Hence, $ib = rt_1 = 1$. The proof of the induction step is as follows. Here, we use $3n+1$ as the runtime function of both previously proved rewrite lemmas (5) and (6). Moreover, $\to_{\mathcal{L}/\mathcal{G}}$ stands for $\equiv_{\mathcal{G}} \circ \to_{\mathcal{L}} \circ \equiv_{\mathcal{G}}$.

$$\begin{array}{rcl|l}
\mathsf{qs}(\gamma_{\mathbf{List}}(n+1)) & \equiv_{\mathcal{G}} & \mathsf{qs}(\mathsf{cons}(\mathsf{zero},\gamma_{\mathbf{List}}(n))) \to_{\mathcal{R}_{\mathsf{qs}}} & rt'_1 = 1 \\
\mathsf{qs}(\mathsf{low}(\mathsf{zero},\gamma_{\mathbf{List}}(n))) \;\mathsf{++}\; \mathsf{cons}(\mathsf{zero},\mathsf{qs}(\mathsf{high}(\mathsf{zero},\gamma_{\mathbf{List}}(n)))) & & \to_{\mathcal{L}/\mathcal{G}} & rt'_2(n) = 3n+1 \\
\mathsf{qs}(\mathsf{nil}) \;\mathsf{++}\; \mathsf{cons}(\mathsf{zero},\mathsf{qs}(\mathsf{high}(\mathsf{zero},\gamma_{\mathbf{List}}(n)))) & & \to_{\mathcal{L}/\mathcal{G}} & rt'_3(n) = 3n+1 \\
\mathsf{qs}(\mathsf{nil}) \;\mathsf{++}\; \mathsf{cons}(\mathsf{zero},\mathsf{qs}(\gamma_{\mathbf{List}}(n))) & & \to_{\mathcal{R}_{\mathsf{qs}}} & rt'_4 = 1 \\
\mathsf{nil} \;\mathsf{++}\; \mathsf{cons}(\mathsf{zero},\mathsf{qs}(\gamma_{\mathbf{List}}(n))) & & \mapsto_{\mathrm{IH},n} & rt'_5(n) = rt(n) \\
\mathsf{nil} \;\mathsf{++}\; \mathsf{cons}(\mathsf{zero},\gamma_{\mathbf{List}}(n)) & & \to_{\mathcal{R}_{\mathsf{qs}}} & rt'_6 = 1 \\
\mathsf{cons}(\mathsf{zero},\gamma_{\mathbf{List}}(n)) & \equiv_{\mathcal{G}} & \gamma_{\mathbf{List}}(n+1) &
\end{array}$$

$$\begin{aligned}
\text{Thus, } is(n) &= \textstyle\sum_{j\in\{1,2,3,4,6\}} rt'_j(\overline{z}_j\theta_j) \qquad\qquad = rt'_1 + rt'_2(n) + rt'_3(n) + rt'_4 + rt'_6 \\
&= 1 + (3n+1) + (3n+1) + 1 + 1 = 6n+5.
\end{aligned}$$

In our example we have $ih = 1$. Now Thm. 18 implies $rt(n) = ib + \sum_{i=0}^{n-1} is(i) = 1 + \sum_{i=0}^{n-1}(6i+5) = 3n^2+2n+1$. Hence, we get the following rewrite lemma:

$$\mathsf{qs}(\gamma_{\mathbf{List}}(n)) \to^{\geq 3n^2+2n+1} \gamma_{\mathbf{List}}(n) \tag{4}$$

In general, the recurrence equations (9) do not describe the exact length of the corresponding rewrite sequence. The reason is that when proving a conjecture $s \to^* t$ by induction, one may instantiate non-induction variables in the induction hypothesis, but this instantiation is ignored in the recurrence equations (9). Hence, in general the function rt in Thm. 18 is only a lower bound for the runtime.

Example 20 (Non-Exact Bounds) *The proof of* $\mathsf{add}(\gamma_{\mathbf{Nats}}(n), \gamma_{\mathbf{Nats}}(n_2)) \to^* \gamma_{\mathbf{Nats}}(n_2 + n)$ *in Ex. 16 used one rewrite step for the induction base and one for the induction step (i.e.,* $\mathit{ib}(n_2) = 1$ *and* $\mathit{is}(n, n_2) = 1$*). As the induction hypothesis was applied once (i.e.,* $\mathit{ih} = 1$*), Thm. 18 results in* $\mathit{rt}(n, n_2) = \mathit{ib}(n_2) + \sum_{i=0}^{n-1} \mathit{is}(i, n_2) = 1 + n$*. This yields the rewrite lemma* $\mathsf{add}(\gamma_{\mathbf{Nats}}(n), \gamma_{\mathbf{Nats}}(n_2)) \to^{\geq 1+n} \gamma_{\mathbf{Nats}}(n_2 + n)$*, which results in a linear lower bound for the runtime complexity of the whole TRS.*

In this example, the bound $1 + n$ *for the runtime of the rewrite lemma is exact, because* $\mathit{ib}(n_2)$ *and* $\mathit{is}(n, n_2)$ *do not depend on* n_2*. But the following modification of the* add*-TRS illustrates why our approach might fail to compute exact bounds. Here,* $\mathsf{add\_double}(x, y)$ *corresponds to a subsequent application of* add *and* double*, i.e., it first computes the addition of* x *and* y*, and then it doubles the result.*

$$\begin{aligned} \mathsf{add\_double}(\mathsf{zero}, y) &\to \mathsf{double}(y) & \mathsf{double}(\mathsf{zero}) &\to \mathsf{zero} \\ \mathsf{add\_double}(\mathsf{succ}(x), y) &\to \mathsf{add\_double}(x, \mathsf{succ}(y)) & \mathsf{double}(\mathsf{succ}(x)) &\to \mathsf{succ}(\mathsf{succ}(\mathsf{double}(x))) \end{aligned}$$

For double*, we infer the rewrite lemma* $\mathsf{double}(\gamma_{\mathbf{Nats}}(n)) \to^{\geq 1+n} \gamma_{\mathbf{Nats}}(2n)$*. For* add_double*, the technique of Sect. 3.2 speculates the conjecture* $\mathsf{add\_double}(\gamma_{\mathbf{Nats}}(n), \gamma_{\mathbf{Nats}}(n_2)) \to^* \gamma_{\mathbf{Nats}}(2n_2 + 2n)$*, which is proved by induction. In the induction base, we have* $\mathsf{add\_double}(\gamma_{\mathbf{Nats}}(0), \gamma_{\mathbf{Nats}}(n_2)) \equiv_{\mathcal{G}} \mathsf{add\_double}(\mathsf{zero}, \gamma_{\mathbf{Nats}}(n_2)) \to_{\mathcal{R}} \mathsf{double}(\gamma_{\mathbf{Nats}}(n_2)) \to_{\mathcal{L}} \gamma_{\mathbf{Nats}}(2n_2)$ *which yields* $\mathit{ib}(n_2) = 2 + n_2$*. In the induction step, we get* $\mathit{is}(n, n_2) = 1$ *and* $\mathit{ih} = 1$ *as before. Now Thm. 18 yields* $\mathit{rt}(n, n_2) = \mathit{ib}(n_2) + \sum_{i=0}^{n-1} \mathit{is}(i, n_2) = 2 + n_2 + n$*, resulting in the rewrite lemma* $\mathsf{add\_double}(\gamma_{\mathbf{Nats}}(n), \gamma_{\mathbf{Nats}}(n_2)) \to^{\geq 2+n_2+n} \gamma_{\mathbf{Nats}}(2n_2 + 2n)$*.*

However, $2 + n_2 + n$ *is only a lower bound on the length of this rewrite sequence: The non-induction variable* n_2 *in* $\mathsf{add\_double}$*'s second argument increases in each application of* $\mathsf{add\_double}$*'s recursive rule. Therefore finally,* $\mathsf{double}(\gamma_{\mathbf{Nats}}(n_2 + n))$ *has to be evaluated and rewriting* $\mathsf{add\_double}(\gamma_{\mathbf{Nats}}(n), \gamma_{\mathbf{Nats}}(n_2))$ *takes* $2 + n_2 + 2n$ *steps. However, the increase of* n_2 *is ignored in the recurrence equations (9) and in Thm. 18.*

Often, one is mainly interested in *asymptotic* instead of explicit bounds. Based on Thm. 18, asymptotic bounds for rewrite lemmas can be obtained automatically from their induction proofs, i.e., one only needs ib, is, and ih and does not have to solve any recurrence equation. To express asymptotic bounds w.r.t. just one variable, we define the unary function $\mathit{rt}_{\mathbb{N}} : \mathbb{N} \to \mathbb{N}$ as $\mathit{rt}_{\mathbb{N}}(n) = \mathit{rt}(n, \ldots, n)$.

If the induction hypothesis was not used in the proof of a rewrite lemma (i.e., $\mathit{ih} = 0$), then (9) implies $\mathit{rt}(\overline{n}[n/0]) = \mathit{ib}(\widetilde{n})$ and $\mathit{rt}(\overline{n}[n/n + 1]) = \mathit{is}(\overline{n})$. Thus, if ib and is are polynomials of degree d_{ib} and d_{is}, we obtain $\mathit{rt}_{\mathbb{N}}(n) \in \Omega(n^{\max\{d_{\mathit{ib}}, d_{\mathit{is}}\}})$.

If $\mathit{ih} = 1$, then Thm. 18 implies $\mathit{rt}(\overline{n}) = \mathit{ib}(\widetilde{n}) + \sum_{i=0}^{n-1} \mathit{is}(\overline{n}[n/i])$. Again, let ib and is be polynomials of degree d_{ib} and d_{is}, respectively. Then $\mathit{is}(\overline{n}) = t_0 + t_1 n + t_2 n^2 + \ldots + t_{d_{\mathit{is}}} n^{d_{\mathit{is}}}$, where the t_j are polynomials of degree at most $d_{\mathit{is}} - j$ over variables from $\widetilde{n}$. Moreover, there is at least one t_j with $t_j \neq 0$ which has the *exact* degree $d_{\mathit{is}} - j$. Hence, $\mathit{rt}(\overline{n}) = \mathit{ib}(\widetilde{n}) + \sum_{i=0}^{n-1} (t_0 + t_1 i + t_2 i^2 + \ldots + t_{d_{\mathit{is}}} i^{d_{\mathit{is}}}) =$

$$\mathit{ib}(\widetilde{n}) + t_0 \cdot \sum_{i=0}^{n-1} i^0 + t_1 \cdot \sum_{i=0}^{n-1} i^1 + t_2 \cdot \sum_{i=0}^{n-1} i^2 + \ldots + t_{d_{\mathit{is}}} \cdot \sum_{i=0}^{n-1} i^{d_{\mathit{is}}}. \quad (10)$$

By Faulhaber's formula [21], for any $e \in \mathbb{N}$, $\sum_{i=0}^{n-1} i^e$ is a polynomial of degree $e+1$ over n. Thus for $e = 1$, $\sum_{i=0}^{n-1} i^1 = \frac{n \cdot (n-1)}{2}$ has degree 2. By taking also the degree d_{ib} of ib into account, rt has degree $\max\{d_{ib}, d_{is}+1\}$, i.e., $rt_{\mathbb{N}}(n) \in \Omega(n^{\max\{d_{ib}, d_{is}+1\}})$.

Finally we consider the case where the induction hypothesis was used several times, i.e., $ih > 1$. By construction we always have $is(\overline{n}) \geq 1$ (since the induction step cannot only consist of applying the induction hypothesis). Thus, Thm. 18 implies $rt(\overline{n}) \geq \sum_{i=0}^{n-1} ih^{n-1-i} = \sum_{j=0}^{n-1} ih^j = \frac{ih^n - 1}{ih - 1}$. Hence, $rt_{\mathbb{N}}(n) \in \Omega(ih^n)$, i.e., the runtime of the rewrite lemma is exponential. Cor. 21 summarizes the above observations.

Corollary 21 (Asymptotic Runtime of Rewrite Lemmas) *Let $s \to^* t$ be a valid conjecture with ih, ib, and is as in Def. 17. Moreover, let ib and is be polynomials of degree d_{ib} and d_{is}, respectively. Then there is a rewrite lemma $s \to^{\geq rt(\overline{n})} t$ such that*

- $rt_{\mathbb{N}}(n) \in \Omega(n^{\max\{d_{ib}, d_{is}\}})$, *if* $ih = 0$
- $rt_{\mathbb{N}}(n) \in \Omega(n^{\max\{d_{ib}, d_{is}+1\}})$, *if* $ih = 1$
- $rt_{\mathbb{N}}(n) \in \Omega(ih^n)$, *if* $ih > 1$

Example 22 (Exponential Runtime) *To illustrate Cor. 21, let $\mathcal{R}_{exp} = \{\mathsf{f}(\mathsf{succ}(x), \mathsf{succ}(x)) \to \mathsf{f}(\mathsf{f}(x,x), \mathsf{f}(x,x)), \mathsf{f}(\mathsf{zero}, \mathsf{zero}) \to \mathsf{zero}\}$. Our approach speculates and proves the conjecture $\mathsf{f}(\gamma_{\mathbf{Nats}}(n), \gamma_{\mathbf{Nats}}(n)) \to^* \mathsf{zero}$. The induction base is $\mathsf{f}(\gamma_{\mathbf{Nats}}(0), \gamma_{\mathbf{Nats}}(0)) \equiv_{\mathcal{G}} \mathsf{f}(\mathsf{zero}, \mathsf{zero}) \to_{\mathcal{R}_{exp}} \mathsf{zero}$, i.e., $ib = 1$. The induction step is:*

$$\begin{array}{rcl|l} \mathsf{f}(\gamma_{\mathbf{Nats}}(n+1), \gamma_{\mathbf{Nats}}(n+1)) & \equiv_{\mathcal{G}} & \mathsf{f}(\mathsf{succ}(\gamma_{\mathbf{Nats}}(n)), \mathsf{succ}(\gamma_{\mathbf{Nats}}(n))) \to_{\mathcal{R}_{exp}} & rt'_1 = 1 \\ & & \mathsf{f}(\mathsf{f}(\gamma_{\mathbf{Nats}}(n), \gamma_{\mathbf{Nats}}(n)), \mathsf{f}(\gamma_{\mathbf{Nats}}(n), \gamma_{\mathbf{Nats}}(n))) \mapsto^2_{\mathrm{IH}} & \\ & & \mathsf{f}(\mathsf{zero}, \mathsf{zero}) \to_{\mathcal{R}_{exp}} & rt'_4 = 1 \\ & & \mathsf{zero} & \end{array}$$

Thus, $ih = 2$ and $is(n)$ is the constant 2 for all $n \in \mathbb{N}$. Hence, by Cor. 21 there is a rewrite lemma $\mathsf{f}(\gamma_{\mathbf{Nats}}(n), \gamma_{\mathbf{Nats}}(n)) \to^{\geq rt(n)} \mathsf{zero}$ with $rt(n) \in \Omega(2^n)$. Indeed, Thm. 18 implies $rt(n) = 2^n + \sum_{i=0}^{n-1} 2^{n-1-i} \cdot 2 = 2^{n+1} + 2^n - 2$.

3.5 Inferring Bounds for TRSs

We now use rewrite lemmas to infer lower bounds for the *runtime complexity* $\mathrm{rc}_{\mathcal{R}}$ of a TRS $\mathcal{R}$. Sect. 3.4 showed how to compute a lower bound $rt(\overline{n})$ for the length of the rewrite sequences represented by a valid conjecture $s \to^* t$, where $\overline{n}$ are the variables in s. However, $\mathrm{rc}_{\mathcal{R}}$ is defined w.r.t. the size of the start term of a rewrite sequence (i.e., $\mathrm{rc}_{\mathcal{R}}(n) = \sup\{\, \mathrm{dh}(t, \to_{\mathcal{R}}) \mid t \in \mathcal{T}_B, |t| \leq n \,\}$, cf. Def. 2). Thus, to obtain a lower bound for $\mathrm{rc}_{\mathcal{R}}$ from $rt(\overline{n})$, for any $\sigma : \mathcal{V}(s) \to \mathbb{N}$ one has to take the relation between $rt(\overline{n}\sigma)$ and the size of the start term $s\sigma{\downarrow_{\mathcal{G}/\mathcal{A}}}$ into account. Our approach in Sect. 3.2 only speculates lemmas where $s = f(\gamma_{\tau_1}(s_1), \ldots, \gamma_{\tau_k}(s_k))$ for some $f \in \Sigma_{def}(\mathcal{R})$, polynomials $s_1, \ldots, s_k$ with $\mathcal{V}(s_j) \subseteq \mathcal{V}(s)$, and simply structured types $\tau_1, \ldots, \tau_k$. For any τ_i, let $d_{\tau_i} : \rho_1 \times \cdots \times \rho_b \to \tau$ be τ_i's recursive constructor. Then for any $n \in \mathbb{N}$, Def. 5 implies $|\gamma_{\tau_i}(n){\downarrow_{\mathcal{G}/\mathcal{A}}}| = sz_{\tau_i}(n)$ for $sz_{\tau_i} : \mathbb{N} \to \mathbb{N}$ with

$$sz_{\tau_i}(n) = |\gamma_{\tau_i}(0){\downarrow_{\mathcal{G}/\mathcal{A}}}| + n \cdot \big(1 + |\gamma_{\rho_1}(0){\downarrow_{\mathcal{G}/\mathcal{A}}}| + \cdots + |\gamma_{\rho_b}(0){\downarrow_{\mathcal{G}/\mathcal{A}}}| - |\gamma_{\tau_i}(0){\downarrow_{\mathcal{G}/\mathcal{A}}}|\big)\,.$$

The reason is that $\gamma_{\tau_i}(n){\downarrow_{\mathcal{G}/\mathcal{A}}}$ contains n occurrences of d_{τ_i} and of each $\gamma_{\rho_1}(0){\downarrow_{\mathcal{G}/\mathcal{A}}}$, $\ldots$, $\gamma_{\rho_b}(0){\downarrow_{\mathcal{G}/\mathcal{A}}}$ except $\gamma_{\tau_i}(0){\downarrow_{\mathcal{G}/\mathcal{A}}}$, and just one occurrence of $\gamma_{\tau_i}(0){\downarrow_{\mathcal{G}/\mathcal{A}}}$. For

instance, $|\gamma_{\mathbf{Nats}}(n)\downarrow_{\mathcal{G}/\mathcal{A}}|$ is $sz_{\mathbf{Nats}}(n)$ and $|\gamma_{\mathbf{List}}(n)\downarrow_{\mathcal{G}/\mathcal{A}}|$ is $sz_{\mathbf{List}}(n)$ with

$$\begin{aligned} sz_{\mathbf{Nats}}(n) &= |\gamma_{\mathbf{Nats}}(0)\downarrow_{\mathcal{G}/\mathcal{A}}| + n \cdot (1 + |\gamma_{\mathbf{Nats}}(0)\downarrow_{\mathcal{G}/\mathcal{A}}| - |\gamma_{\mathbf{Nats}}(0)\downarrow_{\mathcal{G}/\mathcal{A}}|) \;=\; |\mathsf{zero}| + n \\ &= 1 + n \\ sz_{\mathbf{List}}(n) &= |\gamma_{\mathbf{List}}(0)\downarrow_{\mathcal{G}/\mathcal{A}}| + n \cdot (1 + |\gamma_{\mathbf{Nats}}(0)\downarrow_{\mathcal{G}/\mathcal{A}}|) \;=\; |\mathsf{nil}| + n \cdot (1 + |\mathsf{zero}|) \\ &= 1 + n \cdot 2 \end{aligned}$$

Thus $|s\downarrow_{\mathcal{G}/\mathcal{A}}| = |f(\gamma_{\tau_1}(s_1), \ldots, \gamma_{\tau_k}(s_k))\downarrow_{\mathcal{G}/\mathcal{A}}|$ with $\mathcal{V}(s) = \overline{n}$ is given by $sz : \mathbb{N}^m \to \mathbb{N}$:

$$sz(\overline{n}) = 1 + sz_{\tau_1}(s_1) + \cdots + sz_{\tau_k}(s_k)$$

For instance, $\mathsf{qs}(\gamma_{\mathbf{List}}(n)) \downarrow_{\mathcal{G}/\mathcal{A}} = \mathsf{qs}(\mathsf{cons}^n(\mathsf{zero}, \mathsf{nil}))$ has the size $sz(n) = 1 + sz_{\mathbf{List}}(n) = 2n + 2$. Since $|\gamma_\tau(0) \downarrow_{\mathcal{G}/\mathcal{A}}|$ is a constant for each type τ, sz is a polynomial whose degree is the maximal degree of the polynomials $s_1, \ldots, s_k$.

Hence, the rewrite lemma (4) for qs states that there are terms of size $sz(n) = 2n+2$ with reductions of at least length $rt(n) = 3n^2 + 2n + 1$. To determine a lower bound for $\mathrm{rc}_{\mathcal{R}_{\mathsf{qs}}}$, we construct an inverse function sz^{-1} with $(sz \circ sz^{-1})(n) = n$. In our example where $sz(n) = 2n+2$, we have $sz^{-1}(n) = \frac{n-2}{2}$ if n is even. Thus, for all even n there are terms of size n with reductions of length $rt(sz^{-1}(n)) = rt(\frac{n-2}{2}) = \frac{3}{4}n^2 - 2n + 2$. Since multivariate polynomials $sz(n_1, \ldots, n_m)$ cannot be inverted, we invert the unary function $sz_{\mathbb{N}} : \mathbb{N} \to \mathbb{N}$ with $sz_{\mathbb{N}}(n) = sz(n, \ldots, n)$ instead.

Of course, inverting $sz_{\mathbb{N}}$ fails if $sz_{\mathbb{N}}$ is not injective. However, the conjectures speculated in Sect. 3.2 only contain polynomials with natural coefficients. Then, $sz_{\mathbb{N}}$ is always strictly monotonically increasing. Hence, we only proceed if there is a $sz_{\mathbb{N}}^{-1} : \mathrm{img}(sz_{\mathbb{N}}) \to \mathbb{N}$ where $(sz_{\mathbb{N}} \circ sz_{\mathbb{N}}^{-1})(n) = n$ holds for all $n \in \mathrm{img}(sz_{\mathbb{N}}) = \{n \in \mathbb{N} \mid \exists v \in \mathbb{N}.\ sz_{\mathbb{N}}(v) = n\}$. To extend $sz_{\mathbb{N}}^{-1}$ to a function on $\mathbb{N}$, for any (total) function $h : M \to \mathbb{N}$ with $M \subseteq \mathbb{N}$, we define $\lfloor h \rfloor(n) : \mathbb{N} \to \mathbb{N}$ by:

$$\lfloor h \rfloor(n) = h(\max\{n' \mid n' \in M, n' \leq n\}),\ \text{if } n \geq \min(M) \qquad \text{and} \qquad \lfloor h \rfloor(n) = 0,\ \text{otherwise}$$

Using this notation, Thm. 23 states how we can derive lower bounds for $\mathrm{rc}_{\mathcal{R}}$.

Theorem 23 (Explicit Bounds for $\mathrm{rc}_{\mathcal{R}}$) *Let $s \to^{\geq rt(n_1,\ldots,n_m)} t$ be a rewrite lemma for $\mathcal{R}$, let $sz : \mathbb{N}^m \to \mathbb{N}$ such that $sz(b_1, \ldots, b_m)$ is the size of $s[n_1/b_1, \ldots, n_m/b_m]\downarrow_{\mathcal{G}/\mathcal{A}}$ for all $b_1, \ldots, b_m \in \mathbb{N}$, and let $sz_{\mathbb{N}}$ be injective, i.e., $sz_{\mathbb{N}}^{-1}$ exists. Then for all $n \in \mathbb{N}$ with $n \geq \min(\mathrm{img}(sz_{\mathbb{N}}))$, $rt_{\mathbb{N}} \circ \lfloor sz_{\mathbb{N}}^{-1} \rfloor$ is a lower bound for $\mathrm{rc}_{\mathcal{R}}$, i.e., $(rt_{\mathbb{N}} \circ \lfloor sz_{\mathbb{N}}^{-1} \rfloor)(n) \leq \mathrm{rc}_{\mathcal{R}}(n)$.*

In the rewrite lemma (4) for qs where $sz_{\mathbb{N}}(n) = 2n + 2$, we have $\lfloor sz_{\mathbb{N}}^{-1} \rfloor(n) = \lfloor \frac{n-2}{2} \rfloor \geq \frac{n-3}{2}$ and $\mathrm{rc}_{\mathcal{R}_{\mathsf{qs}}}(n) \geq rt(\lfloor sz_{\mathbb{N}}^{-1} \rfloor(n)) \geq rt(\frac{n-3}{2}) = \frac{3}{4}n^2 - \frac{7}{2}n + \frac{19}{4}$ for all $n \geq 2$.

However, even if $sz_{\mathbb{N}}^{-1}$ exists, finding resp. approximating $sz_{\mathbb{N}}^{-1}$ automatically can be non-trivial in general. Therefore, we now show how to obtain an asymptotic lower bound for $\mathrm{rc}_{\mathcal{R}}$ directly from a rewrite lemma $f(\gamma_{\tau_1}(s_1), \ldots, \gamma_{\tau_k}(s_k)) \to^{\geq rt(\overline{n})} t$ without constructing $sz_{\mathbb{N}}^{-1}$. As mentioned, if e is the maximal degree of the polynomials $s_1, \ldots, s_k$, then sz is also a polynomial of degree e and thus, $sz_{\mathbb{N}}(n) \in \mathcal{O}(n^e)$. Moreover, from the induction proof of the rewrite lemma we obtain an asymptotic lower bound for $rt_{\mathbb{N}}$, cf. Cor. 21. Using these bounds, Lemma 24 can be used to infer an asymptotic lower bound for $\mathrm{rc}_{\mathcal{R}}$ directly.[10]

[10] In the second case of Lemma 24, we fix a small inaccuracy from [9,10] where we inadvertently wrote $(rt_{\mathbb{N}} \circ \lfloor sz_{\mathbb{N}}^{-1} \rfloor)(n) \in \Omega(b^{\sqrt[e]{n}})$.

Lemma 24 (Asymptotic Bounds for Function Composition) *Let* $rt_{\mathbb{N}}, sz_{\mathbb{N}} : \mathbb{N} \to \mathbb{N}$ *where* $sz_{\mathbb{N}} \in \mathcal{O}(n^e)$ *for some* $e \geq 1$ *and* $sz_{\mathbb{N}}$ *is strictly monotonically increasing.*

- *If* $rt_{\mathbb{N}}(n) \in \Omega(n^d)$ *with* $d \geq 0$, *then* $(rt_{\mathbb{N}} \circ \lfloor sz_{\mathbb{N}}^{-1} \rfloor)(n) \in \Omega(n^{\frac{d}{e}})$.
- *If* $rt_{\mathbb{N}}(n) \in \Omega(b^n)$ *with* $b \geq 1$, *then* $(rt_{\mathbb{N}} \circ \lfloor sz_{\mathbb{N}}^{-1} \rfloor)(n) \in b^{\Omega(\sqrt[e]{n})}$.

For the rewrite lemma $\mathsf{qs}(\gamma_{\mathbf{List}}(n)) \to^{\geq rt(n)} \gamma_{\mathbf{List}}(n)$ where $rt_{\mathbb{N}} = rt$ and $sz_{\mathbb{N}} = sz$, we only need the asymptotic bounds $sz(n) \in \mathcal{O}(n)$ and $rt(n) \in \Omega(n^2)$ to infer that Quicksort has at least quadratic complexity, i.e., $\mathrm{rc}_{\mathcal{R}_{\mathsf{qs}}}(n) \in \Omega(n^{\frac{2}{1}}) = \Omega(n^2)$.

While Thm. 23 explains how to find concrete lower bounds for $\mathrm{rc}_{\mathcal{R}}$ (if $sz_{\mathbb{N}}$ can be inverted), Cor. 25 summarizes our results on asymptotic lower bounds for $\mathrm{rc}_{\mathcal{R}}$. It combines Cor. 21 on inferring asymptotic bounds for rt with Lemma 24.

Corollary 25 (Asymptotic Bounds for $\mathrm{rc}_{\mathcal{R}}$) *Let* $s \to^* t$ *be a valid conjecture and let* $sz : \mathbb{N}^m \to \mathbb{N}$ *be the function* $sz(b_1, \ldots, b_m) = \left|s[n_1/b_1, \ldots, n_m/b_m]\!\downarrow_{\mathcal{G}/\mathcal{A}}\right|$, *where* $sz_{\mathbb{N}}(n) \in \mathcal{O}(n^e)$ *for some* $e \geq 1$, *and* $sz_{\mathbb{N}}$ *is strictly monotonically increasing. Moreover, let* $i\!h$, $i\!b$, *and* $i\!s$ *be defined as in Def. 17, where* $i\!b$ *and* $i\!s$ *have the degrees* $d_{i\!b}$ *and* $d_{i\!s}$.

(a) $\mathrm{rc}_{\mathcal{R}}(n) \in \Omega(n^{\frac{\max\{d_{i\!b}, d_{i\!s}\}}{e}})$, *if* $i\!h = 0$
(b) $\mathrm{rc}_{\mathcal{R}}(n) \in \Omega(n^{\frac{\max\{d_{i\!b}, d_{i\!s}+1\}}{e}})$, *if* $i\!h = 1$
(c) $\mathrm{rc}_{\mathcal{R}}(n) \in i\!h^{\Omega(\sqrt[e]{n})}$, *if* $i\!h > 1$

4 Detecting Decreasing Loops to Infer Lower Bounds

As mentioned, the *induction technique* of Sect. 3 is related to our approach for proving (possibly *non-looping*) non-termination from [8]. In contrast, most other non-termination techniques for TRSs try to detect *loops*. In this section, we show how to adapt such techniques in order to infer lower complexity bounds.

The induction technique of Sect. 3 has two main drawbacks: its efficiency is limited, since it heavily relies on SMT solving and equational unification. Moreover, it builds upon several heuristics, which restrict its power. For instance, narrowing is used to speculate conjectures in Sect. 3.2, which is non-deterministic. Hence, heuristics are applied to reduce the search space and to decide when to stop narrowing. Consequently, the induction technique may fail due to unfavorable heuristic decisions during the construction of narrowing trees. Moreover, the definition of the generator functions in Def. 5 is a heuristic as well, i.e., $\gamma_\tau(n)$ should be a "suitable" term of type τ whose size is linear in n. However, there are examples where other generator functions than those in Def. 5 are needed.

Example 26 (Failure due to $\gamma_{\mathbf{List}}$) *This TRS checks if a list contains* zero.

$$\mathsf{contains}(\mathsf{nil}) \to \mathsf{false} \qquad \mathsf{contains}(\mathsf{cons}(\mathsf{succ}(x), xs)) \to \mathsf{contains}(xs)$$
$$\mathsf{contains}(\mathsf{cons}(\mathsf{zero}, xs)) \to \mathsf{true}$$

If one uses the heuristic of Def. 5 for the choice of generator functions, $\gamma_{\mathbf{List}}$ *only yields lists of* zeros. *However, for such inputs the complexity of* contains *is constant, i.e., then one cannot prove the desired linear lower bound.*

The loop detection technique of this section does not require SMT solving or equational unification (i.e., it is more efficient than the induction technique). Moreover, it is not based on type inference and generator equations. Thus, it avoids the problems that are due to the heuristics in the induction technique. While the loop detection technique also applies narrowing, it only needs to find a *single* narrowing sequence satisfying a specific condition. In contrast, the induction technique requires *multiple* narrowing sequences which are suitable for generalization, resulting in a narrowing *tree*. However, the loop detection technique can only infer linear and exponential bounds. Hence, it does not subsume the induction technique.

In Sect. 4.1 we adapt the notion of loops to prove linear lower bounds. Sect. 4.2 extends this approach to exponential bounds. Finally, Sect. 4.3 discusses the relation between the induction technique of Sect. 3 and the loop detection technique.

4.1 Loop Detection for Linear Lower Bounds

A *loop* is a reduction sequence $s \to_{\mathcal{R}}^+ C[s\,\sigma]$ for some context C and some substitution σ. Each loop gives rise to a non-terminating reduction $s \to_{\mathcal{R}}^+ C[s\,\sigma] \to_{\mathcal{R}}^+ C[C\,\sigma[s\,\sigma^2]] \to_{\mathcal{R}}^+ \dots$ The idea of the technique in this section is to detect rewrite sequences which are similar to loops, but at some position π of s, a context D of constant size is removed (i.e., we want to detect so-called *decreasing loops*). Hence, we want to find infinite families of rewrite sequences of the form

$$\begin{array}{lllll} s[D^n[t]]_\pi & \to_{\mathcal{R}}^+ & C[s[D^{n-1}[t]]_\pi\,\sigma] & \trianglerighteq & s[D^{n-1}[t]]_\pi\,\sigma \\ & \to_{\mathcal{R}}^+ & C[s[D^{n-2}[t]]_\pi\,\sigma^2] & \trianglerighteq & s[D^{n-2}[t]]_\pi\,\sigma^2 \\ & \to_{\mathcal{R}}^+ \circ \trianglerighteq & \dots & \to_{\mathcal{R}}^+ \circ \trianglerighteq & s[t]_\pi\,\sigma^n. \end{array}$$

Again, $s' \trianglerighteq s$ means that s is a subterm of s', cf. Sect. 2. If there is a decreasing loop, then the runtime complexity of $\mathcal{R}$ is at least linear, i.e., $\mathrm{rc}_{\mathcal{R}}(n) \in \Omega(n)$. To find such families of rewrite sequences, we look for a rewrite step of the form $s[D[x]]_\pi \to_{\mathcal{R}} C[s[x]_\pi\,\sigma]$ for a variable x. Then the term $s[D^n[x]]_\pi$ starts a reduction of length n. This term is obtained by applying the substitution θ^n with $\theta = [x/D[x]]$ to $s[x]_\pi$.

The rule $\mathsf{contains}(\mathsf{cons}(\mathsf{succ}(x), xs)) \to \mathsf{contains}(xs)$ from Ex. 26 removes the context $D = \mathsf{cons}(\mathsf{succ}(x), \Box)$ around the variable xs in every rewrite step. The size of this context is constant. Thus, if one starts with a context D^n of size $3 \cdot n$, then one can perform n rewrite steps to remove this context, which shows $\mathrm{rc}_{\mathcal{R}}(n) \in \Omega(n)$.

Note that the variable xs occurs exactly once in the left-hand side $\ell = \mathsf{contains}(\mathsf{cons}(\mathsf{succ}(x), xs))$, at position $\pi = 1.2$ (i.e., ℓ is *linear*). Moreover, this variable also appears in the right-hand side r at a position $\xi = 1$ that is above π (i.e., $\xi < \pi$). Thus, every rewrite step removes the context that is around xs in $\ell|_\xi = \mathsf{cons}(\mathsf{succ}(x), xs)$. Let $\overline{\ell}$ be the term that results from ℓ by replacing the subterm $\ell|_\xi$ by the variable xs, i.e., $\overline{\ell} = \ell[xs]_\xi = \mathsf{contains}(xs)$. Moreover, let θ be the substitution that replaces xs by $\ell|_\xi$ again (i.e., $\theta = [xs/\mathsf{cons}(\mathsf{succ}(x), xs)]$). Suppose that $\overline{\ell}\,\sigma = r$ for some matcher σ that does not instantiate the variables in $\ell|_\xi$ (i.e., σ does not *interfere* with θ). A rewrite rule $\ell \to r$ satisfying these conditions is called a *decreasing loop*. In our example, $\overline{\ell} = \mathsf{contains}(xs)$ matches the right-hand side $r = \mathsf{contains}(xs)$ with the matcher $\sigma = \varnothing$, i.e., with the identical substitution. Thus, rewriting $\overline{\ell}\,\theta = \ell$ results in an instance of $\overline{\ell}$ again (i.e., in $r = \overline{\ell}\,\sigma$). Hence, every

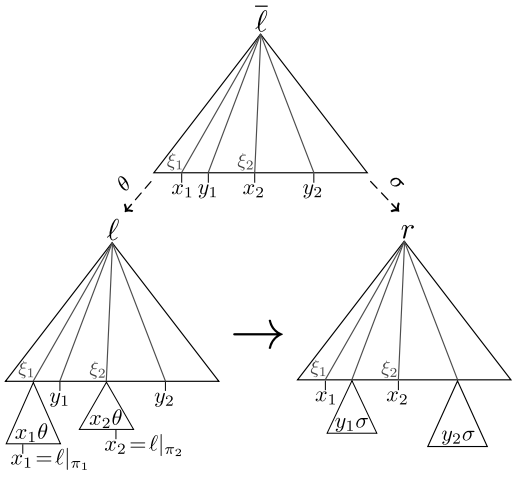

Fig. 2: Decreasing Loop

decreasing loop results in a rewrite sequence of linear length: if one starts with $\overline{\ell}\,\theta^n$, then this rewrite step can be repeated n times, removing one application of θ in each step. More precisely, we have $\overline{\ell}\,\theta^n = \ell\,\theta^{n-1} \to_{\mathcal{R}} r\,\theta^{n-1} = \overline{\ell}\,\sigma\,\theta^{n-1} = \overline{\ell}\,\theta^{n-1}\,\sigma'$ for some substitution σ', as σ does not interfere with θ. Hence, in our example, the term $\overline{\ell}\,\theta^n = \mathsf{contains}(D^n[xs])$ with $D = \mathsf{cons}(\mathsf{succ}(x), \Box)$ starts a reduction of length n.

Based on this idea, three improvements enhance the applicability of the resulting technique: First, it suffices to require that $\overline{\ell}$ matches a *subterm* r of the right-hand side (i.e., the right-hand side may have the form $C[r]$ for some context C). Second, instead of creating $\overline{\ell}$ by replacing one subterm $\ell|_\xi$ of ℓ with a variable $x \in \mathcal{V}(\ell|_\xi)$, we can replace *several* subterms $\ell|_{\xi_1}, \ldots, \ell|_{\xi_m}$ with variables $x_i \in \mathcal{V}(\ell|_{\xi_i})$. Here, $\xi_1, \ldots, \xi_m$ must be *parallel* positions, i.e., we have $\xi_i \not\leq \xi_j$ and $\xi_j \not\leq \xi_i$ whenever $i \neq j$. The structure of ℓ, $\overline{\ell}$, and r is illustrated in Fig. 2. Here, a dashed arrow labeled with a substitution like $\overline{\ell} \overset{\theta}{\dashrightarrow} \ell$ means that applying the substitution θ to $\overline{\ell}$ results in ℓ. Third, instead of checking whether a single rule $\ell \to_{\mathcal{R}} C[r]$ is a decreasing loop, we can also consider rewrite sequences $\ell \to^+_{\mathcal{R}} C[r]$. To find such rewrite sequences, we repeatedly narrow the right-hand sides of those rules whose left-hand sides are basic.[11] This leads to Def. 27. (Note that here, (a) implies that $\xi_1, \ldots, \xi_m$ are parallel positions, since the $r|_{\xi_i}$ are pairwise different variables.)

Definition 27 (Decreasing Loop) *Let $\ell \to^+_{\mathcal{R}} C[r]$ for some linear basic term ℓ and some $r \notin \mathcal{V}$. We call $\ell \to^+_{\mathcal{R}} C[r]$ a* decreasing loop *if there are pairwise different variables $x_1, \ldots, x_m$ (with $m \geq 0$) and positions $\pi_1, \ldots, \pi_m$ with $x_i = \ell|_{\pi_i}$ for all $1 \leq i \leq m$ such that:*

(a) for each x_i, there is a $\xi_i < \pi_i$ such that $r|_{\xi_i} = x_i$
(b) there is a substitution σ with $\overline{\ell}\,\sigma = r$ for $\overline{\ell} = \ell[x_1]_{\xi_1} \ldots [x_m]_{\xi_m}$

We call σ the result substitution*, $\theta = [x_i/\ell|_{\xi_i} \mid 1 \leq i \leq m]$ the* pumping substitution*, and $\xi_1, \ldots, \xi_m$ the* abstracted positions *of the decreasing loop.*

[11] For each node t in the resulting narrowing tree, we check if the corresponding rewrite sequence from the root to t is a decreasing loop. As a heuristic, our implementation stops the (breadth-first) construction of narrowing trees if we reach depth 20 or constructed 100 terms.

Example 28 (Result Substitution) Consider Ex. 16 again to illustrate the result substitution σ. The rule $\mathsf{add}(\mathsf{succ}(x), y) \to \mathsf{add}(x, \mathsf{succ}(y))$ *is a decreasing loop where* $\overline{\ell} = \mathsf{add}(x, y)$, $\xi = 1$, $\theta = [x/\mathsf{succ}(x)]$, *and* $\sigma = [y/\mathsf{succ}(y)]$. *Indeed, we have* $\overline{\ell}\,\theta^n \to_{\mathcal{R}}^n \overline{\ell}\,\sigma^n$.

Example 29 (Linearity) To see why we require linearity of ℓ, consider $\mathcal{R} = \{\mathsf{f}(\mathsf{succ}(x), x) \to \mathsf{f}(x, x)\}$. *If non-linear terms ℓ were allowed by Def. 27, then* $\mathsf{f}(\mathsf{succ}(x), x) \to \mathsf{f}(x, x)$ *would be a decreasing loop with the abstracted position $\xi = 1$. Thus, we would falsely conclude a linear lower runtime bound although* $\mathrm{rc}_{\mathcal{R}}(n)$ *is constant.*

Example 30 (Non-Variable Right-Hand Sides) The requirement $r \notin \mathcal{V}$ in Def. 27 is needed to ensure that θ instantiates variables by constructor terms. Otherwise, for the TRS $\mathcal{R} = \{\mathsf{f}(x) \to x\}$ we would falsely detect a decreasing loop although $\mathrm{rc}_{\mathcal{R}}(n)$ is constant. The reason is that for $\overline{\ell} = x$ and $\theta = [x/\mathsf{f}(x)]$, $\overline{\ell}\,\theta^n$ starts a rewrite sequence of length n, but $\overline{\ell}\,\theta^n$ is not a basic term.

Thm. 31 states that any decreasing loop gives rise to a linear lower bound.

Theorem 31 (Linear Lower Bounds by Loop Detection) *If a TRS $\mathcal{R}$ has a decreasing loop, then we have* $\mathrm{rc}_{\mathcal{R}}(n) \in \Omega(n)$.

Of course, the linear bound recognized by Thm. 31 is just a lower bound. In particular, if $\{\xi_1, \ldots, \xi_m\} = \varnothing$ (i.e., $\overline{\ell} = \ell$), then we have the loop $\ell \to_{\mathcal{R}}^+ C[r] \trianglerighteq r = \overline{\ell}\,\sigma = \ell\,\sigma$. Thus, there is even an infinite lower bound (i.e., a basic term starts an infinite reduction). Hence, loops are indeed special cases of decreasing loops.

Corollary 32 (Infinite Lower Bounds by Loop Detection) *If there is a decreasing loop for a TRS $\mathcal{R}$ with an empty set of abstracted positions, then* $\mathrm{rc}_{\mathcal{R}}(n) \in \Omega(\omega)$.

4.2 Loop Detection for Exponential Lower Bounds

We now adapt the criterion of Thm. 31 in order to detect exponential lower bounds. Thm. 31 characterizes TRSs where a context around a variable x is removed in each rewrite step and the same rewrite rule is again applicable to the right-hand side. We now consider reduction sequences $\ell \to^+ r$ such that $r = C_1[r_1]_{\iota_1} = C_2[r_2]_{\iota_2}$ for parallel positions ι_1 and ι_2 where both $\ell \to^+ C_1[r_1]_{\iota_1}$ and $\ell \to^+ C_2[r_2]_{\iota_2}$ are decreasing loops. Then each rewrite step removes some context, but at the same time it creates *two* redexes on the right-hand side where the same rewrite rule is applicable again. This give rise to an (asymptotic) exponential lower bound of 2^n.

Example 33 (Exponential Bound for Fibonacci Numbers) Consider the following TRS $\mathcal{R}$ with $\mathrm{rc}_{\mathcal{R}}(n) \in \Omega(2^n)$ *which computes the Fibonacci numbers.*

$$\begin{aligned}
\mathsf{fib}(\mathsf{zero}) &\to \mathsf{zero} & \mathsf{add}(\mathsf{zero}, y) &\to y\\
\mathsf{fib}(\mathsf{succ}(\mathsf{zero})) &\to \mathsf{succ}(\mathsf{zero}) & \mathsf{add}(\mathsf{succ}(x), y) &\to \mathsf{add}(x, \mathsf{succ}(y))\\
\mathsf{fib}(\mathsf{succ}(\mathsf{succ}(x))) &\to \mathsf{add}(\mathsf{fib}(\mathsf{succ}(x)), \mathsf{fib}(x))
\end{aligned}$$

In the last fib*-rule, there are* two *recursive calls on the right-hand side where each recursive call gives rise to a decreasing loop. More precisely,* $\mathsf{fib}(\mathsf{succ}(\mathsf{succ}(x))) \to C_1[\mathsf{fib}(\mathsf{succ}(x))]$ *is a decreasing loop with* $\overline{\ell}_1 = \mathsf{fib}(\mathsf{succ}(x))$, *pumping substitution* $\theta_1 = [x/\mathsf{succ}(x)]$, *and result substitution* $\sigma_1 = \varnothing$. *On the other hand,* $\mathsf{fib}(\mathsf{succ}(\mathsf{succ}(x))) \to$

$C_2[\mathsf{fib}(x)]$ *is a decreasing loop with* $\overline{\ell}_2 = \mathsf{fib}(x)$*, pumping substitution* $\theta_2 = [x/\mathsf{succ}(\mathsf{succ}(x))]$*, and result substitution* $\sigma_2 = \varnothing$*.*

The two decreasing loops give rise to an exponential lower bound. We have $\overline{\ell}_1\,\theta_1^n\,\theta_2^n \to_{\mathcal{R}} \mathsf{add}(\overline{\ell}_1\,\theta_1^{n-1}\,\theta_2^n,\ \overline{\ell}_2\,\theta_1^{n-1}\,\theta_2^n)$*. Note that the pumping substitutions* θ_1 *and* θ_2 commute*, i.e.,* $\theta_1\,\theta_2 = \theta_2\,\theta_1$*. Thus, for the subterm in the second argument of* add*, we have* $\overline{\ell}_2\,\theta_1^{n-1}\,\theta_2^n = \overline{\ell}_2\,\theta_2\,\theta_1^{n-1}\,\theta_2^{n-1}$*. Hence after each application of the recursive* fib*-rule to* $\overline{\ell}_i\,\theta_1^n\,\theta_2^n$ *we obtain* two *new similar terms where one application of* θ_i *has been removed, but* $2n-1$ *applications of pumping substitutions remain. Since the pumping substitutions commute, the next reduction step again yields* two *new similar terms with* $2n-2$ *remaining applications of pumping substitutions. Thus, rewriting* $\overline{\ell}_1\,\theta_1^n\,\theta_2^n$ *yields a binary "tree" of reductions which is complete up to height* n*. Hence, the resulting rewrite sequence has an exponential length, i.e.,* $\mathrm{rc}_{\mathcal{R}}(n) \in \Omega(2^n)$*.*

Example 34 (Commutation) The commutation of the pumping substitutions is indeed crucial. Otherwise, it would not be sound to infer an exponential lower bound from the existence of two parallel decreasing loops. For instance, we would then obtain false exponential bounds for typical algorithms that traverse trees (the TRS below represents the simplest possible tree traversal algorithm).

$$\mathsf{traverse}(\mathsf{leaf}) \to \mathsf{leaf} \qquad \mathsf{traverse}(\mathsf{tree}(xs, ys)) \to \mathsf{tree}(\mathsf{traverse}(xs), \mathsf{traverse}(ys))$$

Each recursive call in the right-hand side of the last rule gives rise to a decreasing loop. For $\mathsf{traverse}(\mathsf{tree}(xs, ys)) \to C_1[\mathsf{traverse}(xs)]$ *we have* $\overline{\ell}_1 = \mathsf{traverse}(xs)$ *with the pumping substitution* $\theta_1 = [xs/\mathsf{tree}(xs, ys)]$ *and the result substitution* $\sigma_1 = \varnothing$*. The decreasing loop* $\mathsf{traverse}(\mathsf{tree}(xs, ys)) \to C_2[\mathsf{traverse}(ys)]$ *has* $\overline{\ell}_2 = \mathsf{traverse}(ys)$ *with* $\theta_2 = [ys/\mathsf{tree}(xs, ys)]$ *and* $\sigma_2 = \varnothing$*. However, this does* not *imply an exponential lower bound. The reason is that* θ_1 *and* θ_2 *do not commute. Thus,* $\overline{\ell}_1\,\theta_1^n\,\theta_2^n \to_{\mathcal{R}} \mathsf{tree}(\overline{\ell}_1\,\theta_1^{n-1}\,\theta_2^n,\ \overline{\ell}_2\,\theta_1^{n-1}\,\theta_2^n)$*. But instead of* $\overline{\ell}_2\,\theta_1^{n-1}\,\theta_2^n = \overline{\ell}_2\,\theta_2\,\theta_1^{n-1}\,\theta_2^{n-1}$ *as in Ex. 33, we have* $\overline{\ell}_2\,\theta_1^{n-1}\theta_2^n = \overline{\ell}_2\,\theta_2^n$*. Thus, the resulting runtime is only linear.*

The following example shows that in addition to the commutation property of the pumping substitutions, the result substitution of one decreasing loop must not interfere with the pumping substitution of the other loop.

Example 35 (Interference of Result and Pumping Substitution) The rule $\ell \to r$ *with* $\ell = \mathsf{f}(\mathsf{succ}(x), \mathsf{succ}(y))$ *and* $r = \mathsf{c}(\mathsf{f}(x, \mathsf{succ}(\mathsf{zero})), \mathsf{f}(x, y))$ *gives rise to two decreasing loops. The first one is* $\ell \to C_1[\mathsf{f}(x, \mathsf{succ}(\mathsf{zero}))]$ *with* $\overline{\ell}_1 = \mathsf{f}(x, \mathsf{succ}(y))$*,* $r_1 = \mathsf{f}(x, \mathsf{succ}(\mathsf{zero}))$*,* $\theta_1 = [x/\mathsf{succ}(x)]$*, and* $\sigma_1 = [y/\mathsf{zero}]$*. The other is* $\ell \to C_2[\mathsf{f}(x, y)]$ *with* $\overline{\ell}_2 = \mathsf{f}(x, y)$*,* $r_2 = \mathsf{f}(x, y)$*,* $\theta_2 = [x/\mathsf{succ}(x), y/\mathsf{succ}(y)]$*, and* $\sigma_2 = \varnothing$*. However, this does* not *imply an exponential lower bound. The reason is that the domain of the pumping substitution* σ_1 *contains the variable* y *which also occurs in the domain and range of* θ_2*. Hence:*

$$\begin{array}{lll|lll}
 & \mathsf{f}(x, \mathsf{succ}(y))\,\theta_1^n\,\theta_2^n & & & \overline{\ell}_1\,\theta_1^n\,\theta_2^n & \\
= & \mathsf{f}(\mathsf{succ}(x), \mathsf{succ}(y))\,\theta_1^{n-1}\,\theta_2^n & & = & \ell\,\theta_1^{n-1}\,\theta_2^n & \\
\to_{\mathcal{R}} & \mathsf{c}(\mathsf{f}(x, \mathsf{succ}(\mathsf{zero})), \mathsf{f}(x, y))\,\theta_1^{n-1}\,\theta_2^n & & \to_{\mathcal{R}} & C_1[r_1]\,\theta_1^{n-1}\,\theta_2^n & \\
\trianglerighteq & \mathsf{f}(x, \mathsf{succ}(\mathsf{zero}))\,\theta_1^{n-1}\,\theta_2^n & & \trianglerighteq & \overline{\ell}_1\,\sigma_1\,\theta_1^{n-1}\,\theta_2^n & \\
= & \mathsf{f}(\mathsf{succ}(x), \mathsf{succ}(\mathsf{zero}))\,\theta_1^{n-2}\,\theta_2^n & & = & \ell\,\sigma_1\,\theta_1^{n-2}\,\theta_2^n & \\
\to_{\mathcal{R}} & \mathsf{c}(\mathsf{f}(x, \mathsf{succ}(\mathsf{zero})), \underline{\mathsf{f}(x, \mathsf{zero})})\,\theta_1^{n-2}\,\theta_2^n & & \to_{\mathcal{R}} & C_2[r_2]\,\sigma_1\,\theta_1^{n-2}\,\theta_2^n &
\end{array}$$

To obtain the desired rewrite sequence of exponential length, each f*-term in the resulting term should again create a binary "tree" of reductions which is complete up to height*

$n-2$ (as there are still at least $n-2$ applications of each pumping substitution). However, the underlined subterm $\mathsf{f}(x,\mathsf{zero})$ (i.e., $r_2\,\sigma_1$) is a normal form. The problem is that the result substitution $\sigma_1 = [y/\mathsf{zero}]$ was applied in the first reduction step and this prevents the subsequent use of $\theta_2 = [y/\mathsf{succ}(y)]$ in order to turn the subterm $\mathsf{f}(x,y)$ of the right-hand side into a redex again.

In general, after one rule application one obtains the terms $\overline{\ell}_1\,\sigma_1\,\theta_1^{n-1}\,\theta_2^n$ and $\overline{\ell}_2\,\sigma_2\,\theta_2\,\theta_1^{n-1}\,\theta_2^{n-1}$, which are "similar" to the start term $\overline{\ell}_1\,\theta_1^n\,\theta_2^n$ up to the result substitutions σ_1 and σ_2. Therefore, one has to require that the result substitutions do not interfere with the pumping substitutions. Then these result substitutions do not prevent the desired exponentially long rewrite sequence.

The following definition introduces the concept of *compatible* decreasing loops. Two decreasing loops are compatible if (a) they result from the same rewrite sequence, (b) they operate on parallel positions of the right-hand side, (c) the result substitution of each loop does not interfere with the pumping substitution of the other loop, and (d) their pumping substitutions commute.

Definition 36 (Compatible Decreasing Loops) *Let $\ell \to_{\mathcal{R}}^+ C[r]_\iota$ and $\ell \to_{\mathcal{R}}^+ C'[r']_{\iota'}$ be decreasing loops with pumping substitutions θ resp. θ', result substitutions σ resp. σ', and abstracted positions $\xi_1,\ldots,\xi_m$ resp. $\xi'_1,\ldots,\xi'_{m'}$. We call $\ell \to_{\mathcal{R}}^+ C[r]_\iota$ and $\ell \to_{\mathcal{R}}^+ C'[r']_{\iota'}$* compatible *iff*

(a) $C[r]_\iota = C'[r']_{\iota'}$
(b) ι *and* ι' *are parallel positions*
(c) $\mathrm{dom}(\sigma) \cap (\mathcal{V}(\ell|_{\xi'_1}) \cup \cdots \cup \mathcal{V}(\ell|_{\xi'_{m'}})) = \mathrm{dom}(\sigma') \cap (\mathcal{V}(\ell|_{\xi_1}) \cup \cdots \cup \mathcal{V}(\ell|_{\xi_m})) = \varnothing$
(d) $\theta\,\theta' = \theta'\theta$

Thm. 37 shows that several compatible decreasing loops lead to exponential runtime.

Theorem 37 (Exponential Lower Bounds by Loop Detection) *If a TRS $\mathcal{R}$ has $d \geq 2$ pairwise compatible decreasing loops, then we have $\mathrm{rc}_{\mathcal{R}}(n) \in \Omega(d^n)$.*

4.3 Relationship between Induction Technique and Loop Detection

We now presented two different procedures to infer lower bounds, i.e., the induction technique of Sect. 3 and the loop detection technique of the current section. Thm. 38 shows that for *linear* lower bounds, loop detection subsumes the induction technique (provided that the TRS is left-linear).

Theorem 38 (Loop Detection Subsumes Induction for Linear Bounds) *Let $\mathcal{R}$ be a TRS and $\mathcal{L}$ be the set of rewrite lemmas that were speculated and proved by the technique of Sect. 3. If $\mathcal{R}$ is left-linear and there is a rewrite lemma $s \to^{\geq \mathit{rt}(\overline{n})} t \in \mathcal{L}$ where $\mathit{rt}(\overline{n})$ is not a constant, then $\mathcal{R}$ has a decreasing loop.*

Thus, for linear lower bounds, loop detection is superior to the induction technique (e.g., loop detection proves the tight lower bound $\Omega(n)$ for Ex. 26, while the induction technique fails). However, as shown by the following theorem, loop detection is not a complete technique for the detection of linear lower bounds. The reason is that even for quite restricted classes of TRSs $\mathcal{R}$ it is not semi-decidable if $\mathrm{rc}_{\mathcal{R}}(n) \in \Omega(n)$ holds. The reason is that the immortality problem of Turing machines can be reduced to the problem of linear lower bounds, where (im)mortality of Turing machines is known to be undecidable [20].

Theorem 39 (Undecidability and Incompleteness of Loop Detection for Linear Bounds) *For the class of linear TRSs where $\ell, r \in \mathcal{T}_B$ for all rewrite rules $\ell \to r$, it is not semi-decidable if $\mathrm{rc}_{\mathcal{R}}(n) \in \Omega(n)$ holds. Hence for this class, loop detection is not complete for the inference of linear lower bounds.*

Thus for linear lower bounds, loop detection is incomplete but more powerful than the induction technique. Similarly, the induction technique fails for the Fibonacci example of Ex. 33, while loop detection proves the lower bound $\Omega(2^n)$. However, for exponential bounds, the induction technique is not subsumed by loop detection.

Example 40 (Loop Detection does not Subsume Induction Technique) Consider the TRS $\mathcal{R}$ with the rules $\mathsf{f}(\mathsf{zero}) \to \mathsf{zero}$ and $\mathsf{f}(\mathsf{succ}(x)) \to \mathsf{succ}(\mathsf{f}(\mathsf{f}(x)))$. Here, f rewrites its only argument to itself in exponentially many steps. The induction technique can prove the rewrite lemma $\mathsf{f}(\gamma_{\mathbf{Nats}}(n)) \to^{\geq rt(n)} \gamma_{\mathbf{Nats}}(n)$ with $rt(n) \in \Omega(2^n)$ and thus, conclude $\mathrm{rc}_{\mathcal{R}}(n) \in \Omega(2^n)$. The reason is that the induction hypothesis is applied twice in the proof of the induction step. However, Thm. 37 cannot infer an exponential lower bound by loop detection. The reason is that in this TRS, there is no function symbol with an arity greater than 1. With such symbols one cannot construct terms that have two parallel positions ι_1, ι_2. Hence, there are no two compatible decreasing loops.

Hence, for exponential bounds, there exist examples where the induction technique is successful whereas loop detection fails and vice versa. Moreover, as illustrated with the Quicksort-TRS of Ex. 3, the induction technique can also infer *non-linear polynomial bounds*, which is not possible with loop detection. Hence, the induction technique and loop detection are orthogonal. Thus, one would like to couple them to obtain even better results. Moreover, an obvious question is whether loop detection can be extended such that besides linear and exponential lower bounds, it can also infer non-linear polynomial bounds. We considered two possibilities for a coupling of loop detection and the induction technique:

(A) In Def. 27, instead of a rewrite sequence $\ell \to_{\mathcal{R}}^{+} C[r]$, one could use a sequence $\ell \to^{\geq rt(\overline{n})} C[r]$ that is obtained by using rewrite lemmas which were generated by the induction technique. If this sequence corresponds to a decreasing loop and $rt_{\mathbb{N}}(n) \in \Omega(n^d)$, then one could infer the lower bound $\mathrm{rc}_{\mathcal{R}}(n) \in \Omega(n^{d+1})$. In this way, one could infer non-linear polynomial lower bounds by loop detection.

(B) One could try to obtain rewrite lemmas from the loop detection technique (i.e., these rewrite lemmas would have to express the infinite families of rewrite sequences induced by a decreasing loop). Such a rewrite lemma for a function f could be used in the induction technique when inferring a lower bound for a function g that calls f as an auxiliary function.

Unfortunately, in general the approach in (A) is not sound.

Example 41 (Using Rewrite Lemmas for Loop Detection 1) Consider the TRS

$$\mathsf{f}(\mathsf{zero}) \to \mathsf{zero} \qquad \mathsf{f}(\mathsf{succ}(x)) \to \mathsf{f}(x) \qquad \mathsf{g}(\mathsf{succ}(x), y) \to \mathsf{g}(x, \mathsf{f}(y))$$

The induction technique infers the rewrite lemma $\mathsf{f}(\gamma_{\mathbf{Nats}}(n)) \to^{\geq n} \gamma_{\mathbf{Nats}}(0)$. Thus, we get the rewrite sequence $\mathsf{g}(\mathsf{succ}(x),\gamma_{\mathbf{Nats}}(n)) \to \mathsf{g}(x, \mathsf{f}(\gamma_{\mathbf{Nats}}(n))) \to^{\geq n} \mathsf{g}(x,\gamma_{\mathbf{Nats}}(0))$. It corresponds to a decreasing loop with $\overline{\ell} = \mathsf{g}(x, \gamma_{\mathbf{Nats}}(n))$, pumping substitution

$\theta = [x/\mathsf{succ}(x)]$, *and result substitution* $\sigma = [n/0]$. *However, this does not give rise to a quadratic lower bound, i.e., the runtime complexity of the TRS is linear.*

The reason is that the result substitution σ *instantiates* n *with* 0. *To infer correct lower bounds from rewrite sequences of the form* $\ell \to^{\geq rt(\overline{n})} C[r]$, *we therefore require that* $rt(\overline{n})$ *must not depend on variables in the domain of the result substitution* σ.

However, this requirement is still not sufficient, as the following example shows.

Example 42 (Using Rewrite Lemmas for Loop Detection 2) We replace the g-rule of Ex. 41 by $\mathsf{g}(\mathsf{cons}(x, xs)) \to \mathsf{c}(\mathsf{f}(x), \mathsf{g}(xs))$. *Now we have the sequence* $\mathsf{g}(\mathsf{cons}(\gamma_{\mathbf{Nats}}(n), xs)) \to \mathsf{c}(\mathsf{f}(\gamma_{\mathbf{Nats}}(n)), \mathsf{g}(xs)) \to^{\geq n} \mathsf{c}(\gamma_{\mathbf{Nats}}(0), \mathsf{g}(xs))$. *The subterm* $\mathsf{g}(xs)$ *in its last term results in a decreasing loop with pumping substitution* $\theta = [xs/\mathsf{cons}(\gamma_{\mathbf{Nats}}(n), xs)]$ *and result substitution* $\sigma = \varnothing$. *Again this does not yield a quadratic lower bound, i.e., the runtime complexity is linear.*

Here, the reason is that while the terms $\mathsf{g}(xs)\,\theta^n$ *indeed start rewrite sequences of quadratic length, the size of* $\mathsf{g}(xs)\,\theta^n$ *is also* quadratic *in* n, *since each application of* θ *introduces a subterm whose size is linear in* n. *Therefore, to infer correct lower bounds from rewrite sequences of the form* $\ell \to^{\geq rt(\overline{n})} C[r]$, *we also require that* $rt(\overline{n})$ *must not depend on variables in the range of the pumping substitution* θ.

We applied a prototypical implementation of (A) that complies with the restrictions imposed by Ex. 41 and 42 to the TRSs from the *"Runtime Complexity"* category of the *Termination Problem Data Base* (*TPDB* 10.3). This is the collection of examples which was used for the *Termination Competition* 2015 [28]. The improvement in (A) only yielded 8 additional non-linear lower bounds that cannot be proven by the induction technique on its own. Hence, this improvement seems to apply only to very few additional TRSs, so we did not investigate it further.

(B) is also problematic, because the families of rewrite sequences that are inferred by loop detection are usually difficult to express with generator functions. By Thm. 31, $\overline{\ell}\,\theta^n$ can be reduced to $C[r]\,\theta^{n-1} = C[\overline{\ell}\,\sigma]\,\theta^{n-1}$. Since the conditions of Thm. 31 ensure that the same reduction can be applied to $\overline{\ell}\,\sigma\,\theta^{n-1}$ again, we obtain a term[12] $C^n[\overline{\ell}\sigma^n]$ after at least n steps. Here, C and the range of σ may contain defined symbols and variables. In contrast, the generator equations in Def. 5 can only express terms $C^n[t]$ where the context C and the term t consist of constructors. Moreover, even if C and the range of σ only contain constructors, it is still difficult to represent $C^n[\overline{\ell}\,\sigma^n]$ with generator equations, as one has to express the "pumping" both in the context and in the substitution simultaneously.

Example 43 (Generating Rewrite Lemmas from Loop Detection) We extend Ex. 16 by $\mathsf{times}(\mathsf{zero}, y) \to \mathsf{zero}$ *and* $\mathsf{times}(\mathsf{succ}(x), y) \to \mathsf{add}(y, \mathsf{times}(x, y))$. *The second* times*-rule yields the decreasing loop* $\mathsf{times}(\mathsf{succ}(x), y) \to C[\mathsf{times}(x, y)]$ *with* $C = \mathsf{add}(y, \Box)$, $\overline{\ell} = \mathsf{times}(x, y)$, $\theta = [x/\mathsf{succ}(x)]$, *and* $\sigma = \varnothing$. *The infinite family of rewrite sequences represented by this decreasing loop is* $\mathsf{times}(\mathsf{succ}^n(x), y) \to^n C^n[\mathsf{times}(x, y)]$, *i.e., we might want to obtain the rewrite lemma* $\mathsf{times}(\mathsf{succ}^n(\mathsf{zero}), y) \to^{\geq n} C^n[\mathsf{zero}]$. *However, such a rewrite lemma cannot be represented in the current formalism of the induction technique, since the context* C *is not ground and contains the defined symbol* add.

[12] To simplify the explanation, here we consider the case where $\mathcal{V}(C) \cap \mathrm{dom}(\theta) = \mathcal{V}(C) \cap \mathrm{dom}(\sigma) = \varnothing$ and $\mathrm{dom}(\theta) \cap \mathcal{V}(\mathrm{range}(\sigma)) = \varnothing$. Otherwise, the resulting term is more complicated.

Hence, more general generator equations $\mathcal{G}$ would be required for (B). However, then it is not clear if equational unification would still be (efficiently) decidable. In contrast, for the generator equations in Def. 5, equational unification is decidable in quadratic time [22]. Efficient equational unification is crucial for the induction technique, as it is based on narrowing and rewriting modulo $\mathcal{G}$. Indeed, a prototypical implementation of (B) was much less efficient (and also less powerful due to timeouts) than the induction technique on its own, such that we discarded (B), too.

Thus, in our implementation we apply loop detection and the induction technique separately: For left-linear TRSs $\mathcal{R}$, we first apply loop detection. If the inferred bound for $\mathrm{rc}_{\mathcal{R}}$ is not ω, we try the induction technique afterwards to improve the found lower bound.

5 Experiments and Conclusion

We presented the first approach to infer *lower* bounds for the runtime complexity $\mathrm{rc}_{\mathcal{R}}$ of TRSs automatically, consisting of two techniques. The *induction technique* (Sect. 3) is based on speculating conjectures by narrowing and proving them by induction. From this proof, one infers a lower bound on the length of the corresponding rewrite sequences. By taking the size of the start term into account, this yields a lower bound for $\mathrm{rc}_{\mathcal{R}}$. *Loop detection* (Sect. 4) is based on decreasing loops (a generalization of loops). While a single decreasing loop results in a linear or infinite lower bound for $\mathrm{rc}_{\mathcal{R}}$, multiple compatible decreasing loops yield an exponential lower bound.

In [9], we presented two improvements for the induction technique: First, *argument filtering* removes certain arguments of function symbols. Second, the induction technique can be extended by allowing *indefinite* rewrite lemmas with unknown right-hand sides. With these two improvements, rewrite lemmas do not have to represent rewrite sequences of the original TRS precisely anymore. Our experimental evaluation includes benchmarks with and without these improvements. They show that these improvements are crucial to increase the power of the induction technique. But if the induction technique is combined with the new loop detection technique, then these improvements only have a minor impact. For that reason (and for reasons of space), we did not present them in the current paper.

Both the induction technique and loop detection can also be adapted to *innermost* rewriting. A lower bound for the runtime of full rewriting is not necessarily also a lower bound for innermost rewriting. As an example, in the TRS $\mathcal{R}$ with the rules $\mathsf{a} \to \mathsf{f}(\mathsf{b})$, $\mathsf{f}(\mathsf{b}) \to \mathsf{f}(\mathsf{b})$, $\mathsf{b} \to \mathsf{c}$, the basic term a starts an infinite reduction and thus $\mathrm{rc}_{\mathcal{R}}(n) = \omega$ for all $n \geq 1$. In contrast, the innermost runtime complexity is finite since $\mathcal{R}$ is innermost terminating. For the induction technique of Sect. 3, we presented an adaption to innermost rewriting already in [9]. Here, one has to use *non-overlapping* rewriting to prove rewrite lemmas, where it is not allowed to rewrite redexes that have a proper subterm which unifies with the left-hand side of a rule. Similarly, to adapt loop detection to the innermost strategy, one may only use decreasing loops $\ell \overset{\mathrm{no}}{\to}^{+}_{\mathcal{R}} C[r]$ where ℓ reduces to $C[r]$ by non-overlapping rewriting (see App. A for details).

Moreover, we also lifted both techniques to *relative* rewriting. A relative TRS consists of two sets of rules $\mathcal{R}$ and $\mathcal{S}$. In contrast to $\mathcal{R}$, the rules from $\mathcal{S}$ do not contribute to the derivation height of a term. For example, in this way $\mathcal{S}$ can be used to model arithmetic operations like addition, which are "built-in" in real programming languages but have to be implemented explicitly in term rewriting. To adapt the induction technique to relative rewriting, recall that we compute ih, ib, and is to infer lower bounds from the induction proofs of conjectures. In the corresponding Def. 17, each relative rule $s \to t$ from $\mathcal{S}$ now has to be regarded as a rewrite lemma $s \to^{\geq 0} t$. To extend loop detection to relative rewriting, Def. 27 has to be adapted. To find decreasing loops, one may now only consider rewrite sequences $\ell \to^{+}_{\mathcal{R} \cup \mathcal{S}} C[r]$ where at least one rule from $\mathcal{R}$ was applied.

We implemented our approach in our tool AProVE [15], using Z3 [7] to solve the arising (possibly non-linear) arithmetic constraints in the induction technique. While the induction technique can also infer concrete bounds, currently AProVE only computes asymptotic bounds. If a lower bound was inferred by the induction technique, then AProVE reports the used rewrite lemmas as witnesses. If AProVE applies loop detection, then the decreasing loops with their pumping and result substitutions are provided as witnesses.

There exist a few results on lower bounds for *derivational complexity* of TRSs [18,24,30][13] and in the *Termination Competitions* 2009 – 2011, Matchbox [29] proved lower bounds for derivational complexity.[14] Moreover, [1] presents techniques to infer exact lower and upper bounds from given recurrence equations. In contrast, our main goal is the step from the TRS (i.e., the "program") to asymptotic lower bounds without solving recurrence equations. (Recall that our loop detection technique does not use any recurrence equations at all and that in the induction technique, no recurrence equations have to be solved if one is only interested in asymptotic bounds, cf. Cor. 21).

Since there are no other tools for lower runtime complexity bounds of TRSs, we compared our results with the asymptotic *upper* bounds computed by TcT2 and TcT3 [4], the most powerful tools for inferring upper bounds of *"Runtime Complexity – Full Rewriting"* at the *Termination Competition* 2015. We tested with 865 TRSs from this category of the TPDB 10.3. We omitted 60 non-standard TRSs with extra variables on right-hand sides. We also disregarded 34 TRSs where TcT2 or TcT3 proved $\mathrm{rc}_{\mathcal{R}}(n) \in \mathcal{O}(1)$ (gray cells in the tables below), since these TRSs have no non-trivial lower bounds (i.e., no lower bounds larger than $\Omega(1)$). Each tool had a time limit of 300 s for each example. The following tables compare the lower bounds found by different combinations of our techniques with the minimum upper bounds computed by TcT2 or TcT3. In the tables, *EXP* means an exponential bound (i.e., $\mathrm{rc}_{\mathcal{R}}(n) \in \mathcal{O}(k^n)$ resp. $\mathrm{rc}_{\mathcal{R}}(n) \in \Omega(k^n)$ for some $k > 1$).

[13] The slides in [18] propose an approach for derivational complexity, where lower bounds are also deduced from induction proofs. However, no formal details are given in [18].

[14] For *derivational complexity*, one considers arbitrary rewrite sequences that may also start with non-basic terms. Here, every non-empty TRS has a trivial linear lower bound. In contrast, proving linear lower bounds for *runtime complexity* is not trivial. Thus, lower bounds for derivational complexity are in general unsound for runtime complexity. Therefore, an experimental comparison with tools for derivational complexity is not meaningful.

$\mathrm{rc}_{\mathcal{R}}(n)$	$\Omega(1)$	$\Omega(n)$	$\Omega(n^2)$	$\Omega(n^3)$	$\Omega(n^{>3})$	EXP	$\Omega(\omega)$
$\mathcal{O}(1)$	(34)	–	–	–	–	–	–
$\mathcal{O}(n)$	112	43	–	–	–	–	–
$\mathcal{O}(n^2)$	9	7	2	–	–	–	–
$\mathcal{O}(n^3)$	1	1	1	1	–	–	–
$\mathcal{O}(n^{>3})$	2	–	–	–	–	–	–
EXP	–	–	–	–	–	–	–
$\mathcal{O}(\omega)$	307	302	60	13	1	3	–

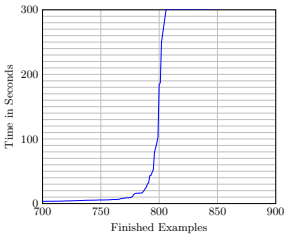

Table 1: Induction Technique

Table 1 shows the results of the induction technique. The average runtime per example was 25.6 s, but according to the chart on the right, it was usually much faster. The examples that are missing in the chart were aborted after 300 s. For these examples, we took the best lower bound that AProVE obtained within[15] these 300 s. The induction technique is especially suitable for polynomial bounds (it proves 353 linear and 78 non-linear polynomial bounds). In particular, it is powerful for TRSs that implement realistic non-linear algorithms, e.g., it shows $\mathrm{rc}_{\mathcal{R}}(n) \in \Omega(n^2)$ for many implementations of sorting algorithms from the TPDB, cf. [2].

$\mathrm{rc}_{\mathcal{R}}(n)$	$\Omega(1)$	$\Omega(n)$	$\Omega(n^2)$	$\Omega(n^3)$	$\Omega(n^{>3})$	EXP	$\Omega(\omega)$
$\mathcal{O}(1)$	(34)	–	–	–	–	–	–
$\mathcal{O}(n)$	41	114	–	–	–	–	–
$\mathcal{O}(n^2)$	5	10	3	–	–	–	–
$\mathcal{O}(n^3)$	1	1	1	1	–	–	–
$\mathcal{O}(n^{>3})$	–	2	–	–	–	–	–
EXP	–	–	–	–	–	–	–
$\mathcal{O}(\omega)$	145	445	69	13	1	13	–

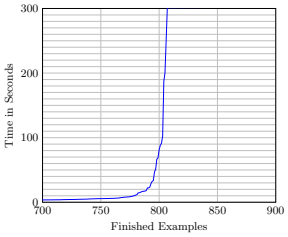

Table 2: Induction Technique & Improvements

Table 2 shows that the induction technique yields significantly better results with argument filtering and indefinite lemmas [9]. In this setting, 673 non-trivial lower bounds were proved. The average runtime was 24.5 s. Hence, even the efficiency of the induction technique benefits a little from the improvements.

$\mathrm{rc}_{\mathcal{R}}(n)$	$\Omega(1)$	$\Omega(n)$	$\Omega(n^2)$	$\Omega(n^3)$	$\Omega(n^{>3})$	EXP	$\Omega(\omega)$
$\mathcal{O}(1)$	(34)	–	–	–	–	–	–
$\mathcal{O}(n)$	15	140	–	–	–	–	–
$\mathcal{O}(n^2)$	–	18	–	–	–	–	–
$\mathcal{O}(n^3)$	–	4	–	–	–	–	–
$\mathcal{O}(n^{>3})$	–	2	–	–	–	–	–
EXP	–	–	–	–	–	–	–
$\mathcal{O}(\omega)$	13	439	–	–	–	144	90

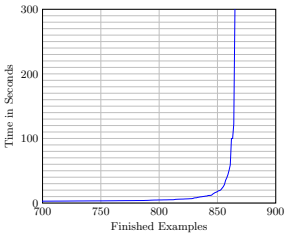

Table 3: Loop Detection

Table 3 presents the results of loop detection. The average runtime was 3.4 s, i.e., it is much more efficient than the induction technique. Loop detection infers non-trivial bounds for almost all analyzed TRSs: There are only 28 cases where

[15] Our implementation stops as soon as we have rewrite lemmas for all defined function symbols, i.e., we do not backtrack to replace a generated rewrite lemma for a function f by a "better" rewrite lemma for f. Thus if we run into a timeout, then we did not manage to obtain rewrite lemmas for all defined function symbols.

loop detection fails. Note that for some of these examples, there might not even exist a non-trivial lower bound. Furthermore, loop detection is also powerful for non-polynomial lower bounds (it finds 144 exponential and 90 infinite lower bounds).

$\mathrm{rc}_{\mathcal{R}}(n)$	$\Omega(1)$	$\Omega(n)$	$\Omega(n^2)$	$\Omega(n^3)$	$\Omega(n^{>3})$	EXP	$\Omega(\omega)$
$\mathcal{O}(1)$	(34)	–	–	–	–	–	–
$\mathcal{O}(n)$	15	140	–	–	–	–	–
$\mathcal{O}(n^2)$	–	16	2	–	–	–	–
$\mathcal{O}(n^3)$	–	2	1	1	–	–	–
$\mathcal{O}(n^{>3})$	–	2	–	–	–	–	–
EXP	–	–	–	–	–	–	–
$\mathcal{O}(\omega)$	14	382	46	10	–	144	90

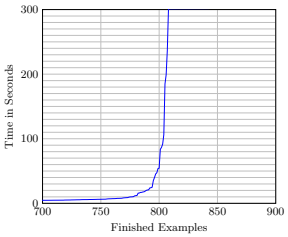

Table 4: Induction Technique & Loop Detection

In Table 4, we apply loop detection and the induction technique after each other, as described at the end of Sect. 4. Thus, we first use loop detection and if it succeeds, we apply the induction technique afterwards and try to improve the found lower bound. The average runtime was 24.6 s. Like loop detection on its own, this configuration proves non-trivial lower bounds for almost all analyzed TRSs (compared to Table 3, it fails for one more TRS due to a timeout). There are less non-linear polynomial lower bounds than in Table 1, since loop detection infers a better exponential or infinite lower bound for some TRSs where the induction technique can only prove a non-linear polynomial bound.

$\mathrm{rc}_{\mathcal{R}}(n)$	$\Omega(1)$	$\Omega(n)$	$\Omega(n^2)$	$\Omega(n^3)$	$\Omega(n^{>3})$	EXP	$\Omega(\omega)$
$\mathcal{O}(1)$	(34)	–	–	–	–	–	–
$\mathcal{O}(n)$	15	140	–	–	–	–	–
$\mathcal{O}(n^2)$	–	15	3	–	–	–	–
$\mathcal{O}(n^3)$	–	2	1	1	–	–	–
$\mathcal{O}(n^{>3})$	–	2	–	–	–	–	–
EXP	–	–	–	–	–	–	–
$\mathcal{O}(\omega)$	14	374	52	10	1	145	90

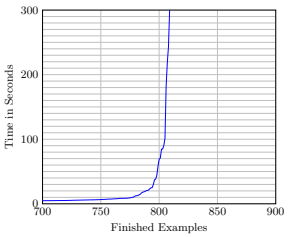

Table 5: Induction Technique & Loop Detection & Improvements

Finally, in Table 5 we combine loop detection and the induction technique with argument filtering and indefinite lemmas. Here, argument filtering was applied prior to the induction technique, but not before loop detection, since further experiments showed that loop detection does not benefit from argument filtering (see [2]). The average runtime was 24.4 s, but again the analysis usually finished much faster. The median of the runtime was 2.6 s. This is the most powerful combination of our techniques to infer lower bounds with AProVE. However, the comparison of Table 4 and 5 shows that argument filtering and indefinite lemmas have little impact on the results if we use both loop detection and the induction technique.

According to Table 5, our implementation inferred non-trivial lower bounds for 836 (97%) of the 865 TRSs. Upper bounds were only obtained for 179 (21%) TRSs, although upper bounds smaller than ω exist for at least all 647 TRSs where AProVE shows termination. Hence, although this is the first technique for lower runtime bounds, its applicability exceeds the applicability of the techniques for upper bounds which were developed for years. Of course, the task of finding worst-case upper bounds is very different from the task of inferring worst-case lower bounds. Tight bounds (where the lower and upper bounds are equal) were proven for the 234 TRSs on the diagonal of the table.

The examples in the TPDB contain many common functional programs. Thus, our experiments are indeed useful to evaluate the performance of our approach on typical realistic algorithms. For instance, AProVE infers quadratic lower bounds for numerous sorting algorithms from the TPDB (e.g., for 3 different implementations of Quicksort, 2 implementations of Minsort, 4 implementations of Maxsort, and one implementation of Selectionsort). However, there are also 3 implementations of Quicksort in the TPDB where AProVE only detects a linear lower bound. The reason is that here the filtering functions (which correspond to the functions low and high in the Quicksort TRS of Ex. 3) also delete duplicates. Therefore, for homogeneous lists, the worst case runtime is linear, not quadratic. Since our generator functions only represent homogeneous data objects (e.g., lists or trees where all elements have the same value), the induction technique cannot detect a non-linear lower bound here.

To evaluate the power of the adaption of our approach for *innermost* rewriting, we performed experiments similar to the ones in Tables 1 - 5 on 970 TRSs from the category *"Runtime Complexity – Innermost Rewriting"* of the TPDB 10.3. The results are analogous to the ones for full rewriting: Again, loop detection is much more efficient and powerful than the induction technique, but a combination of loop detection and the induction technique is preferable in order to also detect non-linear polynomial lower bounds. In this way, AProVE can infer non-trivial lower bounds for 956 (99%) of the 970 TRSs.

Detailed experimental results (which also show the performance of our technique for innermost (and possibly relative) rewriting) and a web interface for our implementation are available at [2]: `http://aprove.informatik.rwth-aachen.de/eval/lowerbounds-journal/`.

Acknowledgements We thank Fabian Emmes for important initial ideas for the induction technique and Chris Lynch, René Thiemann, and Carsten Fuhs for helpful suggestions.

References

1. E. Albert, S. Genaim, and A. N. Masud. On the inference of resource usage upper and lower bounds. *ACM Transactions on Computational Logic*, 14(3), 2012.
2. AProVE: `http://aprove.informatik.rwth-aachen.de/eval/lowerbounds-journal/`.
3. M. Avanzini and G. Moser. A combination framework for complexity. In *Proc. RTA '13*, LIPIcs 21, pages 55–70, 2013.
4. M. Avanzini, G. Moser, and M. Schaper. TcT: Tyrolean complexity tool. In *Proc. TACAS '16*, LNCS 9636, pages 407–423, 2016.
5. F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge Univ. Press, 1998.
6. Robert S. Boyer and J Moore. *A Computational Logic.* Academic Press, 1979.
7. L. de Moura and N. Bjørner. Z3: An efficient SMT solver. In *Proc. TACAS '08*, LNCS 4963, pages 337–340, 2008.
8. F. Emmes, T. Enger, and J. Giesl. Proving non-looping non-termination automatically. In *Proc. IJCAR '12*, LNAI 7364, pages 225–240, 2012.
9. F. Frohn, J. Giesl, J. Hensel, C. Aschermann, and T. Ströder. Inferring lower bounds for runtime complexity. In *Proc. RTA '15*, LIPIcs 36, pages 334–349, 2015.
10. F. Frohn, M. Naaf, J. Hensel, M. Brockschmidt, and J. Giesl. Lower runtime bounds for integer programs. In *Proc. IJCAR '16*, LNAI 9706, pages 550–567, 2016.
11. C. Fuhs, J. Giesl, M. Parting, P. Schneider-Kamp, and S. Swiderski. Proving termination by dependency pairs and inductive theorem proving. *Journal of Automated Reasoning*, 47(2):133–160, 2011.
12. A. Geser, D. Hofbauer, and J. Waldmann. Termination proofs for string rewriting systems via inverse match-bounds. *Journal of Automated Reasoning*, 34(4):365–385, 2005.

13. J. Giesl, R. Thiemann, and P. Schneider-Kamp. Proving and disproving termination of higher-order functions. In *Proc. FroCoS '05*, LNAI 3717, pages 216–231, 2005.
14. J. Giesl, M. Raffelsieper, P. Schneider-Kamp, S. Swiderski, and R. Thiemann. Automated termination proofs for Haskell by term rewriting. *ACM Transactions on Programming Languages and Systems*, 33(2), 2011.
15. J. Giesl, C. Aschermann, M. Brockschmidt, F. Emmes, F. Frohn, C. Fuhs, J. Hensel, C. Otto, M. Plücker, P. Schneider-Kamp, T. Ströder, S. Swiderski, and R. Thiemann. Analyzing program termination and complexity automatically with AProVE. *Journal of Automated Reasoning*. To appear. Preliminary version appeared in *Proc. IJCAR '14*, LNAI 8562, pages 184-191, 2014.
16. N. Hirokawa and G. Moser. Automated complexity analysis based on the dependency pair method. In *Proc. IJCAR '08*, LNAI 5195, pages 364–379, 2008.
17. D. Hofbauer and C. Lautemann. Termination proofs and the length of derivations. In *Proc. RTA '89*, LNCS 355, pages 167–177, 1989.
18. D. Hofbauer and J. Waldmann. Constructing lower bounds on the derivational complexity of rewrite systems. Slides of a talk at the *2nd Workshop on Proof Theory and Rewriting*, 2010. `http://www.imn.htwk-leipzig.de/~waldmann/talk/10/pr/main.pdf`.
19. M. Hofmann and G. Moser. Amortised resource analysis and typed polynomial interpretations. In *Proc. RTA-TLCA '14*, LNCS 8560, pages 272–286, 2014.
20. P. K. Hooper. The undecidability of the Turing machine immortality problem. *Journal of Symbolic Logic*, 31(2):219–234, 1966.
21. D. Knuth. Johann Faulhaber and sums of powers. *Math. of Comp.*, 61(203):277–294, 1993.
22. C. Lynch and B. Morawska. Basic syntactic mutation. In *Proc. CADE '18*, LNAI 2392, pages 471–485, 2002.
23. R. Milner. A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, 17(3):348–375, 1978.
24. G. Moser and A. Schnabl. The derivational complexity induced by the dependency pair method. *Logical Methods in Computer Science*, 7(3), 2011.
25. L. Noschinski, F. Emmes, and J. Giesl. Analyzing innermost runtime complexity of term rewriting by dependency pairs. *Journal of Automated Reasoning*, 51(1):27–56, 2013.
26. C. Otto, M. Brockschmidt, C. von Essen, and J. Giesl. Automated termination analysis of Java Bytecode by term rewriting. In *Proc. RTA '10*, LIPIcs 6, pages 259–276, 2010.
27. É. Payet. Loop detection in term rewriting using the eliminating unfoldings. *Theoretical Computer Science*, 403(2-3):307–327, 2008.
28. Termination Comp.: `http://termination-portal.org/wiki/Termination_Competition`.
29. J. Waldmann. Matchbox: A tool for match-bounded string rewriting. In *Proc. RTA '04*, LNCS 3091, pages 85–94, 2004.
30. J. Waldmann. Automatic termination. In *Proc. RTA '09*, LNCS 5595, pages 1–16, 2009.
31. H. Zankl, C. Sternagel, D. Hofbauer, and A. Middeldorp. Finding and certifying loops. In *Proc. SOFSEM '10*, LNCS 5901, pages 755–766, 2010.
32. H. Zankl and M. Korp. Modular complexity analysis for term rewriting. *Logical Methods in Computer Science*, 10(1), 2014.
33. H. Zantema. Termination of term rewriting: Interpretation and type elimination. *Journal of Symbolic Computation*, 17(1):23–50, 1994.
34. H. Zantema. Termination of string rewriting proved automatically. *Journal of Automated Reasoning*, 34(2):105–139, 2005.

A Detecting Innermost Decreasing Loops to Infer Lower Bounds

So far, we used loop detection to prove lower bounds for runtime complexity of *full* rewriting. However, TRSs resulting from the translation of programs typically have to be evaluated with an *innermost* strategy (e.g., [14,26]). As usual, a rewrite step is *innermost* (denoted $s \xrightarrow{i}_{\mathcal{R}} t$) if the reduced subterm of s does not have redexes as proper subterms. Hence, we now show how to adapt loop detection to innermost runtime complexity. (A corresponding adaption of the induction technique from Sect. 3 was already presented in [9].) To do so, we introduce innermost decreasing loops, which are like decreasing loops, but here only *non-overlapping* rewrite sequences are considered.

Definition 44 (Non-Overlapping Rewriting) *For a TRS $\mathcal{R}$, we say that a term s reduces to t by* non-overlapping *rewriting (denoted $s \xrightarrow{\text{no}} t$) iff there is a $\pi \in \text{pos}(s)$, a substitution σ, and a rule $\ell \to r \in \mathcal{R}$ such that $s|_\pi = \ell\sigma$, $t = s[r\sigma]_\pi$, and no proper non-variable subterm of $\ell\sigma$ unifies with any (variable-renamed) left-hand side of a rule in $\mathcal{R}$.*

Clearly, any non-overlapping rewrite step is an innermost step (i.e., $s \xrightarrow{\text{no}} t$ implies $s \xrightarrow{\text{i}} t$), but not vice versa. For innermost decreasing loops, instead of reductions $\ell \to_{\mathcal{R}}^+ C[r]$ we now consider reductions of the form $\ell \xrightarrow{\text{no}}_{\mathcal{R}}^+ C[r]$.

Definition 45 (Innermost Decreasing Loop) *Let $\ell \xrightarrow{\text{no}}_{\mathcal{R}}^+ C[r]$ for some linear basic term ℓ and some $r \notin \mathcal{V}$. We call $\ell \xrightarrow{\text{no}}_{\mathcal{R}}^+ C[r]$ an* innermost decreasing loop *if there are pairwise different variables $x_1, \ldots, x_m$ (with $m \geq 0$) and positions $\pi_1, \ldots, \pi_m$ with $x_i = \ell|_{\pi_i}$ for all $1 \leq i \leq m$ such that*

(a) for each x_i, there is a $\xi_i < \pi_i$ such that $r|_{\xi_i} = x_i$
(b) there is a substitution σ with $\overline{\ell}\sigma = r$ for $\overline{\ell} = \ell[x_1]_{\xi_1} \ldots [x_m]_{\xi_m}$

To find non-overlapping rewrite sequences, *non-overlapping narrowing* can be used instead of narrowing. Similar to non-overlapping rewriting, non-overlapping narrowing does not allow reduction steps where a proper non-variable subterm of the redex unifies with a (variable-renamed) left-hand side of a rule.

The following theorem shows that each innermost decreasing loop gives rise to a linear lower bound for innermost runtime complexity. Here, the *innermost* runtime complexity $\text{irc}_{\mathcal{R}}$ is defined analogous to $\text{rc}_{\mathcal{R}}$, i.e., $\text{irc}_{\mathcal{R}}(n) = \sup\{\, \text{dh}(t, \xrightarrow{\text{i}}_{\mathcal{R}}) \mid t \in \mathcal{T}_B, |t| \leq n \,\}$.

Theorem 46 (Linear Lower Bounds for irc **by Loop Detection)** *If a TRS $\mathcal{R}$ has an innermost decreasing loop, then we have $\text{irc}_{\mathcal{R}}(n) \in \Omega(n)$.*

The following example shows that we indeed have to require $\ell \xrightarrow{\text{no}}_{\mathcal{R}}^+ C[r]$ instead of just $\ell \xrightarrow{\text{i}}_{\mathcal{R}}^+ C[r]$ in the definition of innermost decreasing loops. The essential property of non-overlapping rewriting is that if a substitution δ instantiates all variables with normal forms, then $s \xrightarrow{\text{no}} t$ still implies $s\delta \xrightarrow{\text{i}} t\delta$. In contrast, $s \xrightarrow{\text{i}} t$ does not imply $s\delta \xrightarrow{\text{i}} t\delta$.

Example 47 (Non-Overlapping Rewriting) Consider the TRS $\mathcal{R}$ with the rules $\mathsf{f}(y) \to \mathsf{h}(\mathsf{g}(y))$, $\mathsf{h}(\mathsf{g}(y)) \to \mathsf{f}(\mathsf{g}(y))$, and $\mathsf{g}(\mathsf{g}(y)) \to y$. We clearly have $\mathsf{f}(y) \xrightarrow{\text{i}}_{\mathcal{R}}^+ \mathsf{f}(\mathsf{g}(y))$, but $\mathsf{f}(y) \not\xrightarrow{\text{no}}_{\mathcal{R}}^+ \mathsf{f}(\mathsf{g}(y))$. If we replaced "$\ell \xrightarrow{\text{no}}_{\mathcal{R}}^+ C[r]$" by "$\ell \xrightarrow{\text{i}}_{\mathcal{R}}^+ C[r]$" in Def. 45, then $\mathsf{f}(y) \xrightarrow{\text{i}}_{\mathcal{R}}^+ \mathsf{f}(\mathsf{g}(y))$ would be an innermost decreasing loop. However, all innermost rewrite sequences that start with basic terms have at most length 4 for this TRS, i.e., $\text{irc}_{\mathcal{R}}(n) \in \Omega(1)$. The problem is that the rewrite sequence $\mathsf{f}(y) \xrightarrow{\text{i}}_{\mathcal{R}}^+ \mathsf{f}(\mathsf{g}(y))$ does not remain an innermost sequence anymore if one instantiates y with the normal form $\mathsf{g}(y)$, i.e., we have $\mathsf{f}(\mathsf{g}(y)) \not\xrightarrow{\text{i}}_{\mathcal{R}}^+ \mathsf{f}(\mathsf{g}(\mathsf{g}(y)))$.

Based on Def. 45 and Thm. 46, it is straightforward to adapt the concept of *compatible* decreasing loops in Def. 36 and Thm. 37 to innermost rewriting.

B Proofs

Theorem 14 (Proving Conjectures) *Let $\mathcal{R}$, s, t, and $\overline{n}$ be as in Def. 6 and let $n \in \mathcal{V}(s)$. If $s[n/0] \rightharpoonup^* t[n/0]$ and $s[n/n+1] \rightharpoonup^*_{\mathrm{IH},n} t[n/n+1]$, where* IH *is the rule $s \to t$, then the conjecture $s \to^* t$ is valid for $\mathcal{R}$.*

Proof The theorem is implied by the stronger Thm. 18, which we prove instead below. □

To prove Thm. 18, we need two auxiliary lemmas. Lemma 48 shows that $\rightharpoonup$ is closed under instantiations of variables with natural numbers.

Lemma 48 (Stability of $\rightharpoonup$ and $\rightharpoonup_{\ell\to r,n}$) *Let $\mathcal{R}$ be a TRS, let ℓ, r, s, t be terms where s only contains variables of type $\mathbb{N}$, and let $\mu : \mathcal{V}(s) \to \mathbb{N}$. Then we have:*

(a) $s \rightharpoonup t$ implies $s\mu \rightharpoonup t\mu$
(b) $s \rightharpoonup_{\ell\to r,n} t$ implies that there is a substitution $\sigma : \mathcal{V}(\ell) \to \mathbb{N}$ with $n\sigma = n\mu$ and $m\sigma \geq m\mu$ for all $m \in \mathcal{V}(\ell) \setminus \{n\}$ such that $s\mu \rightharpoonup_{\ell\sigma\to r\sigma,n} t\mu$.

Proof Since rewriting is closed under substitutions, we immediately have (a). For (b), let $s \rightharpoonup_{\ell\to r,n} t$. If we also have $s \rightharpoonup t$, then the claim follows from (a). Otherwise, we have $s \mapsto_{\ell\to r,n} t$. Hence, there is a term s', an increasing substitution σ', and $\pi \in \mathrm{pos}(s')$ such that $s \equiv_{\mathcal{G}\cup\mathcal{A}} s'$, $s'|_\pi = \ell\sigma'$, and $s'[r\sigma']_\pi \equiv_{\mathcal{G}\cup\mathcal{A}} t$, where $n\sigma' = n$. Let $\sigma = \sigma'\mu$. Then $s\mu \mapsto_{\ell\sigma\to r\sigma,n} t\mu$, since $s\mu \equiv_{\mathcal{G}\cup\mathcal{A}} s'\mu$, $s'\mu|_\pi = s'|_\pi\mu = \ell\sigma'\mu = \ell\sigma$, and $s'\mu[r\sigma]_\pi = s'\mu[r\sigma'\mu]_\pi = (s'[r\sigma']_\pi)\mu \equiv_{\mathcal{G}\cup\mathcal{A}} t\mu$. Moreover, $n\sigma = n\sigma'\mu = n\mu$ and as σ' is increasing, $m\sigma' \geq m$ implies $m\sigma = m\sigma'\mu \geq m\mu$. □

Lemma 49 infers information on rewrite sequences with $\to_{\mathcal{R}}$ from the relation $\rightharpoonup$. Here, we again regard each rule $\ell \to r \in \mathcal{R}$ as a rewrite lemma $\ell \to^{\geq 1} r$.

Lemma 49 (From $\rightharpoonup$ to $\to_{\mathcal{R}}$) *Let $\mathcal{R}$ be a TRS, let ℓ, r be terms with $\mathrm{root}(\ell) \in \Sigma_{def}(\mathcal{R})$, and let s, t be ground terms. Moreover, let $\mathcal{R}$, $\ell \to r$, and s be well typed w.r.t. Σ' and $\mathcal{V}'$, where s does not have the type $\mathbb{N}$.*

(a) If $s \rightharpoonup t$ and this reduction is done using $\ell \to^{\geq \mathrm{rt}(\overline{n})} r \in \mathcal{R} \cup \mathcal{L}$ and the substitution σ, then we have $s\downarrow_{\mathcal{G}/\mathcal{A}} \to^{\geq \mathrm{rt}(\overline{n}\sigma)}_{\mathcal{R}} t\downarrow_{\mathcal{G}/\mathcal{A}}$.
(b) If $s \mapsto_{\ell\to r,n} t$ and ℓ, r are ground terms, then $s\downarrow_{\mathcal{G}/\mathcal{A}} \to_{\{\ell\downarrow_{\mathcal{G}/\mathcal{A}} \to r\downarrow_{\mathcal{G}/\mathcal{A}}\}} t\downarrow_{\mathcal{G}/\mathcal{A}}$.

Proof In (a), we have $s \rightharpoonup t$, i.e., there is a term s', a substitution σ, and $\pi \in \mathrm{pos}(s')$ such that $s \equiv_{\mathcal{G}\cup\mathcal{A}} s'$, $s'|_\pi = \ell\sigma$, and $s'[r\sigma]_\pi \equiv_{\mathcal{G}\cup\mathcal{A}} t$.

Note that $s \equiv_{\mathcal{G}\cup\mathcal{A}} s'$ implies $s\downarrow_{\mathcal{G}/\mathcal{A}} \equiv_{\mathcal{A}} s'\downarrow_{\mathcal{G}/\mathcal{A}}$ as $\to_{\mathcal{G}/\mathcal{A}}$ is terminating and confluent modulo $\mathcal{A}$. Again, $\to_{\mathcal{G}/\mathcal{A}}$ is the rewrite relation resulting from orienting $\mathcal{G}$ from left to right where rewriting is performed modulo $\mathcal{A}$ (i.e., modulo arithmetic). As s is a ground term that does not have the type $\mathbb{N}$, $s\downarrow_{\mathcal{G}/\mathcal{A}}$ does not contain subterms of type $\mathbb{N}$ and therefore, $s\downarrow_{\mathcal{G}/\mathcal{A}} \equiv_{\mathcal{A}} s'\downarrow_{\mathcal{G}/\mathcal{A}}$ implies $s\downarrow_{\mathcal{G}/\mathcal{A}} = s'\downarrow_{\mathcal{G}/\mathcal{A}}$.

Since ℓ matches $s'|_\pi$, s' has a symbol from $\Sigma_{def}(\mathcal{R})$ at position π. Hence, there are no generator symbols and no subterms of type $\mathbb{N}$ in s' on or above the position π. Therefore, $\mathcal{G}$ and $\mathcal{A}$ cannot be applied on or above π. This implies $s\downarrow_{\mathcal{G}/\mathcal{A}} = s'\downarrow_{\mathcal{G}/\mathcal{A}} = s'\downarrow_{\mathcal{G}/\mathcal{A}} [(s'|_\pi)\downarrow_{\mathcal{G}/\mathcal{A}}]_\pi = s'\downarrow_{\mathcal{G}/\mathcal{A}} [\ell\sigma\downarrow_{\mathcal{G}/\mathcal{A}}]_\pi$.

If $\ell \to r \in \mathcal{L}$, then Def. 7 implies that $\ell\sigma\downarrow_{\mathcal{G}/\mathcal{A}} \to^{\geq \mathrm{rt}(\overline{n}\sigma)}_{\mathcal{R}} r\sigma\downarrow_{\mathcal{G}/\mathcal{A}}$. Hence, $s\downarrow_{\mathcal{G}/\mathcal{A}} = s'\downarrow_{\mathcal{G}/\mathcal{A}} [\ell\sigma\downarrow_{\mathcal{G}/\mathcal{A}}]_\pi \to^{\geq \mathrm{rt}(\overline{n}\sigma)}_{\mathcal{R}} s'\downarrow_{\mathcal{G}/\mathcal{A}} [r\sigma\downarrow_{\mathcal{G}/\mathcal{A}}]_\pi = (s'[r\sigma]_\pi)\downarrow_{\mathcal{G}/\mathcal{A}} \equiv_{\mathcal{A}} t\downarrow_{\mathcal{G}/\mathcal{A}}$. As $t\downarrow_{\mathcal{G}/\mathcal{A}}$ does not contain subterms of type $\mathbb{N}$, we have $(s'[r\sigma]_\pi)\downarrow_{\mathcal{G}/\mathcal{A}} = t\downarrow_{\mathcal{G}/\mathcal{A}}$.

If $\ell \to r \in \mathcal{R}$, then ℓ does not contain generator function symbols or subterms of type $\mathbb{N}$. Hence, we have $s\downarrow_{\mathcal{G}/\mathcal{A}} = s'\downarrow_{\mathcal{G}/\mathcal{A}} [\ell\sigma\downarrow_{\mathcal{G}/\mathcal{A}}]_\pi = s'\downarrow_{\mathcal{G}/\mathcal{A}} [\ell\sigma']_\pi$ for the substitution where $x\sigma' = x\sigma\downarrow_{\mathcal{G}/\mathcal{A}}$ for all $x \in \mathcal{V}$. Then $s\downarrow_{\mathcal{G}/\mathcal{A}} = s'\downarrow_{\mathcal{G}/\mathcal{A}} [\ell\sigma']_\pi \to_{\mathcal{R}} s'\downarrow_{\mathcal{G}/\mathcal{A}} [r\sigma']_\pi = s'\downarrow_{\mathcal{G}/\mathcal{A}} [r\sigma\downarrow_{\mathcal{G}/\mathcal{A}}]_\pi = (s'[r\sigma]_\pi)\downarrow_{\mathcal{G}/\mathcal{A}} = t\downarrow_{\mathcal{G}/\mathcal{A}}$.

In case (b), $s \mapsto_{\ell\to r,n} t$ implies that there is a term s' and a $\pi \in \mathrm{pos}(s')$ such that $s \equiv_{\mathcal{G}\cup\mathcal{A}} s'$, $s'|_\pi = \ell$, and $s'[r]_\pi \equiv_{\mathcal{G}\cup\mathcal{A}} t$, since ℓ, r are ground. As in case (a), since s is a ground term that does not have the type $\mathbb{N}$, we can conclude $s\downarrow_{\mathcal{G}/\mathcal{A}} = s'\downarrow_{\mathcal{G}/\mathcal{A}}$. Moreover, s' again has a defined symbol at position π, since $\mathrm{root}(\ell) \in \Sigma_{def}(\mathcal{R})$. As in (a), this implies $s\downarrow_{\mathcal{G}/\mathcal{A}} = s'\downarrow_{\mathcal{G}/\mathcal{A}} = s'\downarrow_{\mathcal{G}/\mathcal{A}} [\ell\downarrow_{\mathcal{G}/\mathcal{A}}]_\pi$. Thus, we obtain $s\downarrow_{\mathcal{G}/\mathcal{A}} = s'\downarrow_{\mathcal{G}/\mathcal{A}} [\ell\downarrow_{\mathcal{G}/\mathcal{A}}]_\pi \to_{\{\ell\downarrow_{\mathcal{G}/\mathcal{A}} \to r\downarrow_{\mathcal{G}/\mathcal{A}}\}} s'\downarrow_{\mathcal{G}/\mathcal{A}} [r\downarrow_{\mathcal{G}/\mathcal{A}}]_\pi = s'[r]_\pi\downarrow_{\mathcal{G}/\mathcal{A}} \equiv_{\mathcal{A}} t\downarrow_{\mathcal{G}/\mathcal{A}}$. Since $\ell \to r$ is well typed and ℓ does not have type $\mathbb{N}$, $t\downarrow_{\mathcal{G}/\mathcal{A}}$ does not contain subterms of type $\mathbb{N}$. Hence, we have $s'[r]_\pi\downarrow_{\mathcal{G}/\mathcal{A}} = t\downarrow_{\mathcal{G}/\mathcal{A}}$, which proves (b). □

Now we can prove Thm. 18 (on proving conjectures by induction and on inferring complexity bounds from this proof).

Theorem 18 (Explicit Runtime of Rewrite Lemmas) *Let $s \to^* t$ be a conjecture with an induction proof as in Thm. 14, where ih, ib, and is are as in Def. 17. Then $s \to^{\geq \mathit{rt}(\overline{n})} t$ is a rewrite lemma, where $\mathit{rt}(\overline{n}) = \mathit{ih}^n \cdot \mathit{ib}(\widetilde{n}) + \sum_{i=0}^{n-1} \mathit{ih}^{n-1-i} \cdot \mathit{is}(\overline{n}[n/i])$.*

Proof Let rt be defined by the recurrence equations (9). We show that $s \to^{\geq \mathit{rt}(\overline{n})} t$ is a rewrite lemma. More precisely, for any $\mu : \mathcal{V}(s) \to \mathbb{N}$ we prove $s\mu\downarrow_{\mathcal{G}/\mathcal{A}} \to_{\mathcal{R}}^{\geq \mathit{rt}(\overline{n}\mu)} t\mu\downarrow_{\mathcal{G}/\mathcal{A}}$ by induction on $n\mu$.

In the induction base, we have $n\mu = 0$. We first regard the case $s[n/0] \equiv_{\mathcal{G}/\mathcal{A}} t[n/0]$. This implies $s\mu = s[n/0]\mu \equiv_{\mathcal{G}/\mathcal{A}} t[n/0]\mu = t\mu$. Since $\to_{\mathcal{G}/\mathcal{A}}$ is terminating and confluent modulo $\mathcal{A}$, $s\mu \equiv_{\mathcal{G}/\mathcal{A}} t\mu$ implies $s\mu\downarrow_{\mathcal{G}/\mathcal{A}} \equiv_{\mathcal{A}} t\mu\downarrow_{\mathcal{G}/\mathcal{A}}$. Since $s\mu$ and $t\mu$ are ground terms that do not have the type $\mathbb{N}$, $s\mu\downarrow_{\mathcal{G}/\mathcal{A}}$ and $t\mu\downarrow_{\mathcal{G}/\mathcal{A}}$ do not contain any subterms of type $\mathbb{N}$. Hence, $s\mu\downarrow_{\mathcal{G}/\mathcal{A}} \equiv_{\mathcal{A}} t\mu\downarrow_{\mathcal{G}/\mathcal{A}}$ implies $s\mu\downarrow_{\mathcal{G}/\mathcal{A}} = t\mu\downarrow_{\mathcal{G}/\mathcal{A}}$, which proves the desired claim, since $\mathit{ib}(\widetilde{n}\mu) = 0$ and thus $\mathit{rt}(\overline{n}\mu) = 0$.

Now we regard the case $s[n/0] = u_1 \rightharpoonup \ldots \rightharpoonup u_{b+1} = t[n/0]$ for $b \geq 1$. By Lemma 48 (a), $\rightharpoonup$ is stable and thus, $s\mu = s[n/0]\mu = u_1\mu \rightharpoonup \ldots \rightharpoonup u_{b+1}\mu = t[n/0]\mu = t\mu$. When regarding rewrite rules also as rewrite lemmas, Lemma 49 (a) implies $s\mu\downarrow_{\mathcal{G}/\mathcal{A}} = u_1\mu\downarrow_{\mathcal{G}/\mathcal{A}} \to_{\mathcal{R}}^{\geq \mathit{rt}(\overline{y}_1\sigma_1\mu)} \ldots \to_{\mathcal{R}}^{\geq \mathit{rt}(\overline{y}_b\sigma_b\mu)} u_{b+1}\mu\downarrow_{\mathcal{G}/\mathcal{A}} = t\mu\downarrow_{\mathcal{G}/\mathcal{A}}$. This means $s\mu\downarrow_{\mathcal{G}/\mathcal{A}} \to_{\mathcal{R}}^{\geq \mathit{ib}(\widetilde{n}\mu)} t\mu\downarrow_{\mathcal{G}/\mathcal{A}}$ or in other words, $s\mu\downarrow_{\mathcal{G}/\mathcal{A}} \to_{\mathcal{R}}^{\geq \mathit{rt}(\overline{n}\mu)} t\mu\downarrow_{\mathcal{G}/\mathcal{A}}$.

In the induction step, we have $n\mu > 0$. Let $\mu' : \mathcal{V}(s) \to \mathbb{N}$ where μ' is like μ for all $\mathcal{V}(s) \setminus \{n\}$ and $n\mu' = n\mu - 1$. If $s[n/n+1] \equiv_{\mathcal{G}/\mathcal{A}} t[n/n+1]$, we obtain $s\mu \equiv_{\mathcal{A}} s[n/n+1]\mu' \equiv_{\mathcal{G}/\mathcal{A}} t[n/n+1]\mu' \equiv_{\mathcal{A}} t\mu$. Thus, we again have $s\mu\downarrow_{\mathcal{G}/\mathcal{A}} \equiv_{\mathcal{A}} t\mu\downarrow_{\mathcal{G}/\mathcal{A}}$ which implies $s\mu\downarrow_{\mathcal{G}/\mathcal{A}} = t\mu\downarrow_{\mathcal{G}/\mathcal{A}}$. This proves the desired claim, since $\mathit{ih} = 0$ and $\mathit{is}(\overline{n}\mu') = 0$ and thus $\mathit{rt}(\overline{n}\mu) = \mathit{rt}(\overline{n}[n/n+1]\mu') = 0$.

Now we regard the case $s[n/n+1] = v_1 \rightharpoonup_{\mathrm{IH},n} \ldots \rightharpoonup_{\mathrm{IH},n} v_{k+1} = t[n/n+1]$ for $k \geq 1$. By Lemma 48 (b), we obtain $s[n/n+1]\mu' = v_1\mu' \rightharpoonup_{\mathrm{IH}\sigma_1,n} \ldots \rightharpoonup_{\mathrm{IH}\sigma_k,n} v_{k+1}\mu' = t[n/n+1]\mu'$ for substitutions σ_j such that $\mathrm{IH}\,\sigma_j$ is ground and such that $n\sigma_j = n\mu'$ and $m\sigma_j \geq m\mu'$ for all $m \in \mathcal{V}(s) \setminus \{n\}$.

If $v_j\mu' \rightharpoonup v_{j+1}\mu'$, then Lemma 49 (a) implies $v_j\mu'\downarrow_{\mathcal{G}/\mathcal{A}} \to_{\mathcal{R}}^{\geq \mathit{rt}'_j(\overline{z}_j\theta_j\mu')} v_{j+1}\mu'\downarrow_{\mathcal{G}/\mathcal{A}}$. Otherwise, if $v_j\mu' \mapsto_{\mathrm{IH}\sigma_j,n} v_{j+1}\mu'$, then by Lemma 49 (b) we have $v_j\mu'\downarrow_{\mathcal{G}/\mathcal{A}} \to_{\{s\sigma_j\downarrow_{\mathcal{G}/\mathcal{A}} \to t\sigma_j\downarrow_{\mathcal{G}/\mathcal{A}}\}} v_{j+1}\mu'\downarrow_{\mathcal{G}/\mathcal{A}}$. The induction hypothesis implies $s\sigma_j\downarrow_{\mathcal{G}/\mathcal{A}} \to_{\mathcal{R}}^{\geq \mathit{rt}(\overline{n}\sigma_j)} t\sigma_j\downarrow_{\mathcal{G}/\mathcal{A}}$, since $n\sigma_j = n\mu' = n\mu - 1$. As $\overline{n}\sigma_j \geq \overline{n}\mu'$ and as rt is weakly

monotonic, this implies $s\sigma_j\downarrow_{\mathcal{G}/\mathcal{A}} \to_{\mathcal{R}}^{\geq rt(\overline{n}\mu')} t\sigma_j\downarrow_{\mathcal{G}/\mathcal{A}}$. Hence, from $v_j\mu'\downarrow_{\mathcal{G}/\mathcal{A}} \to_{\{s\sigma_j\downarrow_{\mathcal{G}/\mathcal{A}}\to t\sigma_j\downarrow_{\mathcal{G}/\mathcal{A}}\}} v_{j+1}\mu'\downarrow_{\mathcal{G}/\mathcal{A}}$ we can infer $v_j\mu'\downarrow_{\mathcal{G}/\mathcal{A}} \to_{\mathcal{R}}^{\geq rt(\overline{n}\mu')} v_{j+1}\mu'\downarrow_{\mathcal{G}/\mathcal{A}}$. Since there are ih many of these steps, we get $s[n/n+1]\mu'\downarrow_{\mathcal{G}/\mathcal{A}} \to_{\mathcal{R}}^{\geq ih\cdot rt(\overline{n}\mu')+is(\overline{n}\mu')} t[n/n+1]\mu'\downarrow_{\mathcal{G}/\mathcal{A}}$ or in other words, $s[n/n+1]\mu'\downarrow_{\mathcal{G}/\mathcal{A}} \to_{\mathcal{R}}^{\geq rt(\overline{n}[n/n+1]\mu')} t[n/n+1]\mu'\downarrow_{\mathcal{G}/\mathcal{A}}$. This proves the desired claim, since $s\mu\downarrow_{\mathcal{G}/\mathcal{A}} = s[n/n+1]\mu'\downarrow_{\mathcal{G}/\mathcal{A}}$, $t\mu\downarrow_{\mathcal{G}/\mathcal{A}} = t[n/n+1]\mu'\downarrow_{\mathcal{G}/\mathcal{A}}$, and $rt(\overline{n}\mu) = rt(\overline{n}[n/n+1]\mu')$.

Finally we show by induction on n that the closed form for $rt(\overline{n})$ in Thm. 18 satisfies the recurrence equations (9). We obtain $rt(\overline{n}[n/0]) = ih^0\cdot ib(\widetilde{n}) = ib(\widetilde{n})$, as required in (9). Similarly, $rt(\overline{n}[n/n+1]) = ih^{n+1}\cdot ib(\widetilde{n}) + \sum_{i=0}^{n} ih^{n-i}\cdot is(\overline{n}[n/i]) = ih\cdot(\, ih^n\cdot ib(\widetilde{n}) + \sum_{i=0}^{n-1} ih^{n-1-i}\cdot is(\overline{n}[n/i])\,) + is(\overline{n}) = ih\cdot rt(\overline{n}) + is(\overline{n})$, as in (9). □

Corollary 21 (Asymptotic Runtime of Rewrite Lemmas) *Let $s\to^* t$ be a valid conjecture with ih, ib, and is as in Def. 17. Moreover, let ib and is be polynomials of degree d_{ib} and d_{is}, respectively. Then there is a rewrite lemma $s\to^{\geq rt(\overline{n})} t$ such that*

- $rt_{\mathbb{N}}(n)\in\Omega(n^{\max\{d_{ib},d_{is}\}})$, *if* $ih=0$
- $rt_{\mathbb{N}}(n)\in\Omega(n^{\max\{d_{ib},d_{is}+1\}})$, *if* $ih=1$
- $rt_{\mathbb{N}}(n)\in\Omega(ih^n)$, *if* $ih>1$

Proof Let rt be defined as in Thm. 18. As explained in the text, if $ih=0$ then Cor. 21 follows from (9), and if $ih>1$, we have $rt(\overline{n})\geq\frac{ih^n-1}{ih-1}$ and thus, $rt_{\mathbb{N}}(n)\in\Omega(ih^n)$.

Finally if $ih=1$, then $rt(\overline{n})$ has the form (10). Hence, we obtain

$$\begin{aligned}\text{degree}(rt) &= \max\{d_{ib},\text{degree}(t_k\cdot\textstyle\sum_{i=0}^{n-1} i^k)\mid 0\leq k\leq d_{is}\}\text{ by (10)}\\ &= \max\{d_{ib},\text{degree}(t_k)+k+1\mid 0\leq k\leq d_{is}\}\quad\text{by Faulhaber's formula}\\ &= \max\{d_{ib},d_{is}+1\},\end{aligned}$$

as $d_{is}=\max\{\text{degree}(t_kn^k)\mid 0\leq k\leq d_{is}\}=\max\{\text{degree}(t_k)+k\mid 0\leq k\leq d_{is}\}$. □

Theorem 23 (Explicit Bounds for $\mathrm{rc}_{\mathcal{R}}$) *Let $s\to^{\geq rt(n_1,\ldots,n_m)} t$ be a rewrite lemma for $\mathcal{R}$, let $sz:\mathbb{N}^m\to\mathbb{N}$ such that $sz(b_1,\ldots,b_m)$ is the size of $s[n_1/b_1,\ldots,n_m/b_m]\downarrow_{\mathcal{G}/\mathcal{A}}$ for all $b_1,\ldots,b_m\in\mathbb{N}$, and let $sz_{\mathbb{N}}$ be injective, i.e., $sz_{\mathbb{N}}^{-1}$ exists. Then for all $n\in\mathbb{N}$ with $n\geq\min(\text{img}(sz_{\mathbb{N}}))$, $rt_{\mathbb{N}}\circ\lfloor sz_{\mathbb{N}}^{-1}\rfloor$ is a lower bound for $\mathrm{rc}_{\mathcal{R}}$, i.e., $(rt_{\mathbb{N}}\circ\lfloor sz_{\mathbb{N}}^{-1}\rfloor)(n)\leq\mathrm{rc}_{\mathcal{R}}(n)$.*

Proof If $n\geq\min(\text{img}(sz_{\mathbb{N}}))$, then there is a maximal $n'\leq n$ such that $n'\in\text{img}(sz_{\mathbb{N}})$. Thus, $\lfloor sz_{\mathbb{N}}^{-1}\rfloor(n)=sz_{\mathbb{N}}^{-1}(n')$. By the rewrite lemma $s\to^{\geq rt(n_1,\ldots,n_m)} t$, $s[n_1/sz_{\mathbb{N}}^{-1}(n'),\ldots,n_m/sz_{\mathbb{N}}^{-1}(n')]\downarrow_{\mathcal{G}/\mathcal{A}}$ has an evaluation of at least length $rt(sz_{\mathbb{N}}^{-1}(n'),\ldots,sz_{\mathbb{N}}^{-1}(n'))$ $=$ $rt(\lfloor sz_{\mathbb{N}}^{-1}\rfloor(n),\ldots,\lfloor sz_{\mathbb{N}}^{-1}\rfloor(n))$ $=$ $(rt_{\mathbb{N}}\circ\lfloor sz_{\mathbb{N}}^{-1}\rfloor)(n)$. The size of the start term $s[n_1/sz_{\mathbb{N}}^{-1}(n'),\ldots,n_m/sz_{\mathbb{N}}^{-1}(n')]\downarrow_{\mathcal{G}/\mathcal{A}}$ is $sz(sz_{\mathbb{N}}^{-1}(n'),\ldots,sz_{\mathbb{N}}^{-1}(n'))=sz_{\mathbb{N}}(sz_{\mathbb{N}}^{-1}(n'))=n'$. As s has a defined symbol only at the root, $s[n_1/sz_{\mathbb{N}}^{-1}(n'),\ldots,n_m/sz_{\mathbb{N}}^{-1}(n')]\downarrow_{\mathcal{G}/\mathcal{A}}$ is a basic term. As this basic term has the size $n'\leq n$ and its evaluation has at least the length $(rt_{\mathbb{N}}\circ\lfloor sz_{\mathbb{N}}^{-1}\rfloor)(n)$, this implies $(rt_{\mathbb{N}}\circ\lfloor sz_{\mathbb{N}}^{-1}\rfloor)(n)\leq\mathrm{rc}_{\mathcal{R}}(n)$. □

Lemma 24 (Asymptotic Bounds for Function Composition) *Let $rt_{\mathbb{N}},sz_{\mathbb{N}}:\mathbb{N}\to\mathbb{N}$ where $sz_{\mathbb{N}}\in\mathcal{O}(n^e)$ for some $e\geq 1$ and $sz_{\mathbb{N}}$ is strictly monotonically increasing.*

- *If $rt_{\mathbb{N}}(n)\in\Omega(n^d)$ with $d\geq 0$, then $(rt_{\mathbb{N}}\circ\lfloor sz_{\mathbb{N}}^{-1}\rfloor)(n)\in\Omega(n^{\frac{d}{e}})$.*
- *If $rt_{\mathbb{N}}(n)\in\Omega(b^n)$ with $b\geq 1$, then $(rt_{\mathbb{N}}\circ\lfloor sz_{\mathbb{N}}^{-1}\rfloor)(n)\in b^{\Omega(\sqrt[e]{n})}$.*

Proof To prove the lemma, we first have to prove that

$$\lfloor h \rfloor(n) \in \{\lceil h \rceil(n), \lceil h \rceil(n) - 1\} \quad \text{for all } n \in \mathbb{N} \tag{11}$$

holds for any infinite subset M of $\mathbb{N}$ and any function $h : M \to \mathbb{N}$ that is strictly monotonically increasing and surjective. Here, $\lceil h \rceil(n) : \mathbb{N} \to \mathbb{N}$ is defined by $\lceil h \rceil(n) = h(\min\{n' \mid n' \in M, n' \geq n\})$. Note that infinity of h's domain M ensures that there is always an $n' \in M$ with $n' \geq n$.

To prove (11), let $n \in \mathbb{N}$. If $n \in M$, then $\lfloor h \rfloor(n) = \lceil h \rceil(n)$.

If $n \notin M$ and $n < \min(M)$, then $\lfloor h \rfloor(n) = 0$. Moreover, since h is strictly monotonically increasing and surjective, we also have $\lceil h \rceil(n) = 0$.

If $n \notin M$ and $n > \min(M)$, let $n' = \max\{n' \mid n' \in M, n' < n\}$ and $n'' = \min\{n'' \mid n'' \in M, n'' > n\}$. Thus, $n' < n < n''$. Strict monotonicity of h implies $h(n') < h(n'')$. Assume that $h(n'') - h(n') > 1$. Then by surjectivity of h, there is an $\hat{n} \in M$ with $h(\hat{n}) = h(n') + 1$ and thus $h(n') < h(\hat{n}) < h(n'')$. By strict monotonicity of h, we obtain $n' < \hat{n} < n''$. Since $n \notin M$ and $\hat{n} \in M$ implies $n \neq \hat{n}$, we either have $\hat{n} < n$ which contradicts $n' = \max\{n' \mid n' \in M, n' < n\}$ or $\hat{n} > n$ which contradicts $n'' = \min\{n'' \mid n'' \in M, n'' > n\}$. Hence, $\lfloor h \rfloor(n) = h(n') = h(n'') - 1 = \lceil h \rceil(n) - 1$, which proves (11).

Now we prove Lemma 24. Here, $sz_{\mathbb{N}}(n) \in \mathcal{O}(n^e)$ implies $\exists n_0, c > 0.\ \forall n \in \mathbb{N}, n > n_0.\ c \cdot n^e \geq sz_{\mathbb{N}}(n)$. By instantiating n with $sz_{\mathbb{N}}^{-1}(n)$, we obtain

$$\exists n_0, c > 0.\ \forall n \in \mathrm{img}(sz_{\mathbb{N}}), sz_{\mathbb{N}}^{-1}(n) > n_0. \quad c \cdot (sz_{\mathbb{N}}^{-1}(n))^e \geq sz_{\mathbb{N}}(sz_{\mathbb{N}}^{-1}(n)).$$

Since $sz_{\mathbb{N}}$ is strictly monotonically increasing, this also holds for $sz_{\mathbb{N}}^{-1}$. Thus, there is an n_1 such that $sz_{\mathbb{N}}^{-1}(n) > n_0$ holds for all $n > n_1$ with $n \in \mathrm{img}(sz_{\mathbb{N}})$. Hence,

$$\exists n_1, c > 0.\ \forall n \in \mathrm{img}(sz_{\mathbb{N}}), n > n_1. \quad c \cdot (sz_{\mathbb{N}}^{-1}(n))^e \geq sz_{\mathbb{N}}(sz_{\mathbb{N}}^{-1}(n)).$$

This simplifies to $\exists n_1, c > 0.\ \forall n \in \mathrm{img}(sz_{\mathbb{N}}), n > n_1. \quad c \cdot (sz_{\mathbb{N}}^{-1}(n))^e \geq n$. When dividing by c and building the e-th root on both sides, we get $\exists n_1, c > 0.\ \forall n \in \mathrm{img}(sz_{\mathbb{N}}), n > n_1. \quad sz_{\mathbb{N}}^{-1}(n) \geq \sqrt[e]{\frac{n}{c}}$. By monotonicity of $\sqrt[e]{\frac{n}{c}}$, this implies

$$\exists n_1, c > 0.\ \forall n \in \mathbb{N}, n > n_1. \quad \lceil sz_{\mathbb{N}}^{-1} \rceil(n) \geq \sqrt[e]{\tfrac{n}{c}}.$$

Note that $sz_{\mathbb{N}}$ is total and hence, $sz_{\mathbb{N}}^{-1}$ is surjective. Moreover, by strict monotonicity of $sz_{\mathbb{N}}$, $\mathrm{img}(sz_{\mathbb{N}})$ is infinite. Hence, by (11) we get $\lfloor sz_{\mathbb{N}}^{-1} \rfloor(n) + 1 \geq \lceil sz_{\mathbb{N}}^{-1} \rceil(n)$ for all $n \in \mathbb{N}$. Thus, $\exists n_1, c > 0.\ \forall n \in \mathbb{N}, n > n_1. \quad \lfloor sz_{\mathbb{N}}^{-1} \rfloor(n) + 1 \geq \sqrt[e]{\frac{n}{c}}$. Hence,

$$\exists n_1, c > 0.\ \forall n \in \mathbb{N}, n > n_1. \quad \lfloor sz_{\mathbb{N}}^{-1} \rfloor(n) \geq \sqrt[e]{\tfrac{n}{c}} - 1. \tag{12}$$

Let $rt_{\mathbb{N}}(n) \in \Omega(n^d)$ (the case $rt_{\mathbb{N}}(n) \in \Omega(b^n)$ is analogous). This implies $\exists n_0, c' > 0.\ \forall n \in \mathbb{N}, n > n_0. \quad c' \cdot n^d \leq rt_{\mathbb{N}}(n)$. By instantiating n with $\lfloor sz_{\mathbb{N}}^{-1}(n) \rfloor$, we get

$$\exists n_0, c' > 0.\ \forall n \in \mathbb{N}, \lfloor sz_{\mathbb{N}}^{-1} \rfloor(n) > n_0. \quad c' \cdot (\lfloor sz_{\mathbb{N}}^{-1} \rfloor(n))^d \leq rt_{\mathbb{N}}(\lfloor sz_{\mathbb{N}}^{-1} \rfloor(n)).$$

Since $sz_{\mathbb{N}}^{-1}$ is strictly monotonically increasing, $\lfloor sz_{\mathbb{N}}^{-1} \rfloor$ is weakly monotonically increasing by construction. As $\lfloor sz_{\mathbb{N}}^{-1} \rfloor$ is surjective, there is an n_2 such that for all $n > n_2$ we have $\lfloor sz_{\mathbb{N}}^{-1} \rfloor(n) > n_0$. Thus, we obtain

$$\exists n_2, c' > 0.\ \forall n \in \mathbb{N}, n > n_2. \quad c' \cdot (\lfloor sz_{\mathbb{N}}^{-1} \rfloor(n))^d \leq rt_{\mathbb{N}}(\lfloor sz_{\mathbb{N}}^{-1} \rfloor(n)).$$

With (12) and weak monotonicity of $c' \cdot n^d$, by choosing $n_3 = \max\{n_1, n_2\}$ we get $\exists n_3, c, c' > 0.\ \forall n \in \mathbb{N}, n > n_3. \quad c' \cdot (\sqrt[e]{\frac{n}{c}} - 1)^d \leq rt_{\mathbb{N}}(\lfloor sz_{\mathbb{N}}^{-1} \rfloor(n))$. Therefore, $\exists c > 0.\ (rt_{\mathbb{N}} \circ \lfloor sz_{\mathbb{N}}^{-1} \rfloor)(n) \in \Omega\left((\sqrt[e]{\frac{n}{c}} - 1)^d\right)$ and thus, $(rt_{\mathbb{N}} \circ \lfloor sz_{\mathbb{N}}^{-1} \rfloor)(n) \in \Omega(n^{\frac{d}{e}})$. □

Corollary 25 (Asymptotic Bounds for $\mathrm{rc}_{\mathcal{R}}$) *Let $s \to^* t$ be a valid conjecture and let $sz : \mathbb{N}^m \to \mathbb{N}$ be the function $sz(b_1, \ldots, b_m) = \left|s[n_1/b_1, \ldots, n_m/b_m]{\downarrow_{\mathcal{G}/\mathcal{A}}}\right|$, where $sz_{\mathbb{N}}(n) \in \mathcal{O}(n^e)$ for some $e \geq 1$, and $sz_{\mathbb{N}}$ is strictly monotonically increasing. Moreover, let $i\!h$, $i\!b$, and $i\!s$ be defined as in Def. 17, where $i\!b$ and $i\!s$ have the degrees $d_{i\!b}$ and $d_{i\!s}$.*

(a) $\mathrm{rc}_{\mathcal{R}}(n) \in \Omega(n^{\frac{\max\{d_{i\!b}, d_{i\!s}\}}{e}})$, *if* $i\!h = 0$
(b) $\mathrm{rc}_{\mathcal{R}}(n) \in \Omega(n^{\frac{\max\{d_{i\!b}, d_{i\!s}+1\}}{e}})$, *if* $i\!h = 1$
(c) $\mathrm{rc}_{\mathcal{R}}(n) \in i\!h^{\Omega(\sqrt[e]{n})}$, *if* $i\!h > 1$

Proof (a) In this case, Cor. 21 implies that there is a rewrite lemma $s \to^{\geq rt(\overline{n})} t$ such that $rt_{\mathbb{N}}(n) \in \Omega(n^{\max\{d_{i\!b}, d_{i\!s}\}})$. With Lemma 24, we get $(rt_{\mathbb{N}} \circ \lfloor sz_{\mathbb{N}}^{-1} \rfloor)(n) \in \Omega(n^{\frac{\max\{d_{i\!b}, d_{i\!s}\}}{e}})$. Moreover, Thm. 23 states that $(rt_{\mathbb{N}} \circ \lfloor sz_{\mathbb{N}}^{-1} \rfloor)(n) \leq \mathrm{rc}_{\mathcal{R}}(n)$ holds for all $n \in \mathbb{N}$. Thus, we obtain $\mathrm{rc}_{\mathcal{R}}(n) \in \Omega(n^{\frac{\max\{d_{i\!b}, d_{i\!s}\}}{e}})$.

(b) Now Cor. 21 implies that there is a rewrite lemma $s \to^{\geq rt(\overline{n})} t$ with $rt_{\mathbb{N}}(n) \in \Omega(n^{\max\{d_{i\!b}, d_{i\!s}+1\}})$. Thus, with Lemma 24, we result in $(rt_{\mathbb{N}} \circ \lfloor sz_{\mathbb{N}}^{-1} \rfloor)(n) \in \Omega(n^{\frac{\max\{d_{i\!b}, d_{i\!s}+1\}}{e}})$. Similar to (a), this implies $\mathrm{rc}_{\mathcal{R}}(n) \in \Omega(n^{\frac{\max\{d_{i\!b}, d_{i\!s}+1\}}{e}})$.

(c) By Cor. 21 there is a lemma $s \to^{\geq rt(\overline{n})} t$ with $rt_{\mathbb{N}}(n) \in \Omega(i\!h^n)$. By Lemma 24, we get $(rt_{\mathbb{N}} \circ \lfloor sz_{\mathbb{N}}^{-1} \rfloor)(n) \in i\!h^{\Omega(\sqrt[e]{n})}$. As in (a), we have $\mathrm{rc}_{\mathcal{R}}(n) \in i\!h^{\Omega(\sqrt[e]{n})}$. ◻

Theorem 31 (Linear Lower Bounds by Loop Detection) *If a TRS $\mathcal{R}$ has a decreasing loop, then we have $\mathrm{rc}_{\mathcal{R}}(n) \in \Omega(n)$.*

Proof For all $n \in \mathbb{N}$ and any substitution δ, we prove that $\overline{\ell}\,\theta^n\,\delta \to_{\mathcal{R}}^+ \circ \trianglerighteq \overline{\ell}\,\theta^{n-1}\,\delta'$ for a substitution δ'. Thus, these rewrite steps can be repeated n times. We have

$$\begin{array}{rclcll} \overline{\ell}\,\theta^n\,\delta & = & \ell\,\theta^{n-1}\,\delta & \to_{\mathcal{R}}^+ & C[r]\,\theta^{n-1}\,\delta & \trianglerighteq \; r\,\theta^{n-1}\,\delta \\ & = & \overline{\ell}\,\sigma\,\theta^{n-1}\,\delta & \overset{(\star)}{=} & \overline{\ell}\,\theta^{n-1}\,(\sigma\theta^{n-1})|_{\mathrm{dom}(\sigma)}\,\delta & = \; \overline{\ell}\,\theta^{n-1}\,\delta' \end{array}$$

for the substitution $\delta' = (\sigma\,\theta^{n-1})|_{\mathrm{dom}(\sigma)}\,\delta$. Here, $(\sigma\,\theta^{n-1})|_{\mathrm{dom}(\sigma)}$ denotes the composition of the substitutions σ and θ^{n-1}, but restricted to the domain of σ.

The step marked with $(\star)$ holds since σ does not instantiate variables in the domain or range of θ. To see why $\mathrm{dom}(\sigma)$ is disjoint from $\mathrm{dom}(\theta) = \{x_1, \ldots, x_m\}$, note that $x_i\,\sigma \neq x_i$ would mean $\overline{\ell}|_{\xi_i}\,\sigma \neq r|_{\xi_i}$. As $\overline{\ell}|_{\xi_i}\,\sigma = \overline{\ell}\,\sigma|_{\xi_i}$, this would imply $\overline{\ell}\,\sigma|_{\xi_i} \neq r|_{\xi_i}$, which contradicts $\overline{\ell}\,\sigma = r$. Moreover, since ℓ is linear, the sets $\mathcal{V}(\overline{\ell})$ and $(\mathcal{V}(\ell|_{\xi_1}) \cup \ldots \cup \mathcal{V}(\ell|_{\xi_m})) \setminus \{x_1, \ldots, x_m\}$ are disjoint. Clearly, the substitution σ that matches $\overline{\ell}$ to r can be chosen such that its domain only includes variables occurring in $\overline{\ell}$. Then σ also does not instantiate any variables occurring in the range of θ.

Thus, for each $n \in \mathbb{N}$, there is a rewrite sequence of length n starting with $\overline{\ell}\,\theta^n$. This term is basic, since the range of θ only contains terms of the form $\ell|_{\xi_i}$. Each $\ell|_{\xi_i}$ is a constructor term, since ξ_i cannot be the root position, due to $r \notin \mathcal{V}$. By construction, θ does not duplicate variables, as ℓ and thus $\overline{\ell}|_{\xi_1}, \ldots, \overline{\ell}|_{\xi_m}$ only contain each x_i once. Therefore, we have $|\overline{\ell}\,\theta^n| \in \mathcal{O}(n)$ and obtain $\mathrm{rc}_{\mathcal{R}}(n) \in \Omega(n)$. ◻

Corollary 32 (Infinite Lower Bounds by Loop Detection) *If there is a decreasing loop for a TRS $\mathcal{R}$ with an empty set of abstracted positions, then $\mathrm{rc}_{\mathcal{R}}(n) \in \Omega(\omega)$.*

Proof The corollary holds because of the loop $\ell \to_{\mathcal{R}}^+ C[r] \trianglerighteq r = \overline{\ell}\sigma = \ell\sigma$. ◻

The following lemma is needed for the proof of Thm. 37. It shows that for compatible loops, the ranges and domains of pumping substitutions do not interfere with each other. This ensures that the size of a start term like $\overline{\ell}_1\,\theta_1^n\,\theta_2^n$ is linear in n (i.e., applying pumping substitutions a linear number of times only leads to a term of linear size).

Lemma 50 (Ranges and Domains of Pumping Substitutions) *Let there be two compatible loops with pumping substitutions θ and θ'. For any $x \in \mathrm{dom}(\theta)$, we have*

(a) $\mathcal{V}(x\,\theta) \cap \mathrm{dom}(\theta) = \{x\}$ *and x only occurs once in $x\,\theta$*
(b) $\mathcal{V}(x\,\theta) \cap \mathrm{dom}(\theta') \subseteq \{x\}$

Proof The claim (a) follows from Def. 27, as ℓ is linear. For (b), assume there is $y \in \mathcal{V}(x\,\theta) \cap \mathrm{dom}(\theta')$ with $y \neq x$. Let $\rho_1, \ldots, \rho_d$ be all positions of $x\,\theta$ where y occurs, i.e., $(x\,\theta)|_{\rho_1} = \ldots = (x\,\theta)|_{\rho_d} = y$. Thus, $\rho_1, \ldots, \rho_d$ are parallel positions. Note that

$$x \in \mathrm{dom}(\theta'). \tag{13}$$

To prove (13), note that otherwise, we would have $(x\,\theta'\,\theta)|_{\rho_1} = (x\,\theta)|_{\rho_1} = y$. On the other hand, we obtain $(x\,\theta\,\theta')|_{\rho_1} = (x\,\theta)|_{\rho_1}\,\theta' = y\,\theta'$. Since θ and θ' commute, this implies $y = y\,\theta'$ which is a contradiction to $y \in \mathrm{dom}(\theta')$.

Since $x \in \mathrm{dom}(\theta) \cap \mathrm{dom}(\theta')$, by (a) there exist positions $\pi \neq \varepsilon$ and $\zeta \neq \varepsilon$ such that $(x\,\theta)|_\pi = x$ and $(x\,\theta')|_\zeta = x$. By (a), x only occurs once in $x\,\theta$ and only once in $x\,\theta'$. Moreover, (a) implies that applying θ (resp. θ') to any variable $y \neq x$ does not introduce occurrences of x. Hence, x only occurs once in $x\,\theta'\,\theta$. Since θ and θ' commute, the same holds for $x\,\theta\,\theta'$.

Hence, $(x\,\theta'\,\theta)|_{\zeta.\pi} = x$ and $(x\,\theta\,\theta')|_{\pi.\zeta} = x$ implies $\zeta.\pi = \pi.\zeta$. This means that

$$\text{there is an } \alpha \in \mathbb{N}^+ \text{ such that } \pi = \alpha^n \text{ and } \zeta = \alpha^m \text{ for } n, m \in \mathbb{N}. \tag{14}$$

Here, α^n stands for the position $\alpha\,.\,\alpha\,\ldots\,\alpha$ where the sequence α is repeated n times. To see why (14) holds, we prove that (14) follows from $\zeta.\pi = \pi.\zeta$ for arbitrary positions π and ζ (in this proof, we also allow $\pi = \varepsilon$ or $\zeta = \varepsilon$). The proof is done by induction on π and ζ. In the induction base, $\pi = \varepsilon$ or $\zeta = \varepsilon$ immediately imply (14). In the induction step, we have $\pi \neq \varepsilon$ and $\zeta \neq \varepsilon$. W.l.o.g., let $|\pi| \leq |\zeta|$. Then $\zeta.\pi = \pi.\zeta$ implies $\zeta = \pi.\pi'$ for some position π'. Hence, $\zeta.\pi = \pi.\zeta$ now becomes $\pi.\pi'.\pi = \pi.\pi.\pi'$ and thus, $\pi'.\pi = \pi.\pi'$. Since $\pi \neq \varepsilon$, the induction hypothesis implies $\pi = \alpha^n$ and $\pi' = \alpha^m$ for some $\alpha \in \mathbb{N}^+$ and $n, m \in \mathbb{N}$. Thus, $\zeta = \pi.\pi' = \alpha^{n+m}$, which proves (14).

We now perform a case analysis on the relationship between π and ζ.

Case 1: $\pi \leq \zeta$
In this case, we have $\zeta = \pi.\pi'$ for some position π'. We obtain $(x\,\theta'\,\theta)|_{\zeta.\rho_1} = ((x\,\theta')|_\zeta\,\theta)|_{\rho_1} = (x\,\theta)|_{\rho_1} = y$. The commutation of θ' and θ implies that we also have $(x\,\theta\,\theta')|_{\zeta.\rho_1} = y$. However, $(x\,\theta\,\theta')|_{\zeta.\rho_1} = (x\,\theta\,\theta')|_{\pi.\pi'.\rho_1} = ((x\,\theta)|_\pi\,\theta')|_{\pi'.\rho_1} = (x\,\theta')|_{\pi'.\rho_1}$. Note that $x\,\theta'$ cannot contain the variable y, since $y \in \mathrm{dom}(\theta')$ by the assumption at the beginning of the proof and $\mathcal{V}(x\,\theta') \cap \mathrm{dom}(\theta') = \{x\}$ by (13) and (a).[16] Thus, this contradicts $(x\,\theta\,\theta')|_{\zeta.\rho_1} = y$.

[16] To see why $\mathcal{V}(x\,\theta') \cap \mathrm{dom}(\theta') = \{x\}$ holds, note that we have $x \in \mathrm{dom}(\theta')$ by (13). Since (a) holds for the pumping substitution of any decreasing loop, it also holds for θ'. Hence, $x \in \mathrm{dom}(\theta')$ implies $\mathcal{V}(x\,\theta') \cap \mathrm{dom}(\theta') = \{x\}$.

Case 2: $\pi \not\leq \zeta$
By (14), we have $\zeta = \alpha^m$ and $\pi = \alpha^{m+k}$ for $m > 0$ and $k > 0$ (since $\zeta \neq \varepsilon$ and $\pi \neq \varepsilon$). As $y \in \mathrm{dom}(\theta')$, by (a) there is a unique position κ such that $y\theta'|_\kappa = y$. Recall that $\rho_1, \ldots, \rho_d$ are the only positions where y occurs in $x\theta$. Due to (a), $\rho_1.\kappa, \ldots, \rho_d.\kappa$ are the only positions where y occurs in $x\theta\theta'$ (since $(x\,\theta\,\theta')|_{\rho_i.\kappa} = ((x\,\theta)_{\rho_i}\,\theta')|_\kappa = (y\,\theta')|_\kappa = y$). Similarly, $\zeta.\rho_1, \ldots, \zeta.\rho_d$ are the only positions where y occurs in $x\,\theta'\,\theta$ (since $(x\,\theta'\,\theta)|_{\zeta.\rho_i} = ((x\,\theta')|_\zeta\,\theta)|_{\rho_i} = (x\,\theta)|_{\rho_i} = y$). As $x\,\theta\,\theta' = x\,\theta'\,\theta$, the positions $\rho_1.\kappa, \ldots, \rho_d.\kappa$ are the same as the positions $\zeta.\rho_1, \ldots, \zeta.\rho_d$. Let $\rho_1, \ldots, \rho_d$ be ordered according to the (total) lexicographic ordering $\sqsubset$ on tuples of numbers (i.e., $\rho_1 \sqsubset \rho_2 \sqsubset \ldots \sqsubset \rho_d$).[17] Then we also have $\rho_1.\kappa \sqsubset \ldots \sqsubset \rho_d.\kappa$ (as the ρ_i are parallel positions) and $\zeta.\rho_1 \sqsubset \ldots \sqsubset \zeta.\rho_d$. This implies $\rho_i.\kappa = \zeta.\rho_i$ for all $1 \leq i \leq d$, i.e., in particular $\rho_1.\kappa = \zeta.\rho_1$. As $\zeta = \alpha^m$, this means $\rho_1.\kappa = \alpha^m.\rho_1$.

Let e be the largest number such that $\rho_1 = \alpha^e.\rho'$ for some position ρ'. Thus, α is no prefix of ρ'. We perform a case analysis on the relation between e and k.

If $e \geq k$, then $y = (x\,\theta\,\theta')|_{\alpha^m.\rho_1} = (x\,\theta\,\theta')|_{\alpha^{m+e}.\rho'} \trianglelefteq (x\,\theta\,\theta')|_{\alpha^{m+k}} = (x\,\theta)|_{\alpha^{m+k}}\,\theta'$ $= x\theta'$. But this contradicts (a), as $x \in \mathrm{dom}(\theta')$ by (13). Thus, $x\theta'$ cannot contain y.

Now we consider the case $e < k$. Note that $\rho_1.\kappa = \alpha^m.\rho_1$ implies $\alpha^e.\rho'.\kappa = \alpha^m.\alpha^e.\rho'$, i.e., $\rho'.\kappa = \alpha^m.\rho'$. Since α is no prefix of ρ', ρ' must be a (proper) prefix of α, since $m > 0$. Thus, we have $\rho' < \alpha$, which implies $\alpha^m.\rho_1 = \alpha^{m+e}.\rho' < \alpha^{m+e+1} \leq \alpha^{m+k}$, as $e < k$. Hence, we have $y = (x\,\theta\,\theta')|_{\alpha^m.\rho_1} \rhd (x\,\theta\,\theta')|_{\alpha^{m+k}} = x$. This is an immediate contradiction, because the variable y cannot contain the variable x as a subterm. □

Theorem 37 (Exponential Lower Bounds by Loop Detection) *If a TRS $\mathcal{R}$ has $d \geq 2$ pairwise compatible decreasing loops, then we have $\mathrm{rc}_\mathcal{R}(n) \in \Omega(d^n)$.*

Proof For each $1 \leq j \leq d$, let θ_j be the pumping substitution and σ_j be the result substitution of the decreasing loop $\ell \to_\mathcal{R}^+ C_j[r_j]_{\iota_j}$ where $r = C_j[r_j]_{\iota_j}$. If $\xi_1, \ldots, \xi_m$ are the abstracted positions of the j-th decreasing loop and $x_i = r_j|_{\xi_i}$ for all $1 \leq i \leq m$, then let $\overline{\ell}_j = \ell[x_1]_{\xi_1} \ldots [x_m]_{\xi_m}$. Thus, we have $\overline{\ell}_j\,\theta_j = \ell$ and $\overline{\ell}_j\,\sigma_j = r_j$.

For all $1 \leq j \leq d$, all $n \in \mathbb{N}$, and any substitution δ, $\overline{\ell}_j\,\theta_1^n \ldots \theta_d^n\,\delta$ starts a reduction of asymptotic length d^n. To show this, we prove $\overline{\ell}_j\,\theta_1^n \ldots \theta_d^n\,\delta \to_\mathcal{R}^+ q$ for some q such that for all $1 \leq k \leq d$, there is a substitution δ'_k with $q|_{\iota_k} = \overline{\ell}_k\,\theta_1^n \ldots \theta_j^{n-1} \ldots \theta_d^n\,\delta'_k$. Hence, q contains d terms of the form $\overline{\ell}_k\,\theta_1^n \ldots \theta_j^{n-1} \ldots \theta_d^n\,\delta'_k$ at parallel positions.

$$\begin{array}{lcll} \overline{\ell}_j\,\theta_1^n \ldots \theta_d^n\,\delta & \overset{(\dagger)}{=} & \overline{\ell}_j\,\theta_j\,\theta_1^n \ldots \theta_j^{n-1} \ldots \theta_d^n\,\delta & = \\ \ell\,\theta_1^n \ldots \theta_j^{n-1} \ldots \theta_d^n\,\delta & \to_\mathcal{R}^+ & r\,\theta_1^n \ldots \theta_j^{n-1} \ldots \theta_d^n\,\delta & = \; q \end{array}$$

where (†) holds as θ_j commutes with all θ_i by Def. 36 (d). For any $1 \leq k \leq d$,

$$\begin{array}{rcll} q|_{\iota_k} & = & r_k\,\theta_1^n \ldots \theta_j^{n-1} \ldots \theta_d^n\,\delta & = \overline{\ell}_k\,\sigma_k\,\theta_1^n \ldots \theta_j^{n-1} \ldots \theta_d^n\,\delta \\ & \overset{(\star)}{=} & \overline{\ell}_k\,\theta_1^n \ldots \theta_j^{n-1} \ldots \theta_d^n\,(\sigma_k\,\theta_1^n \ldots \theta_j^{n-1} \ldots \theta_d^n)|_{\mathrm{dom}(\sigma_k)}\,\delta & = \overline{\ell}_k\,\theta_1^n \ldots \theta_j^{n-1} \ldots \theta_d^n\,\delta'_k \end{array}$$

for the substitution $\delta'_k = (\sigma_k\,\theta_1^n \ldots \theta_j^{n-1} \ldots \theta_d^n)|_{\mathrm{dom}(\sigma_k)}\,\delta$.

For the step marked with ($\star$), as in the proof of Thm. 31, σ_k does not instantiate variables in the domain or the range of θ_k. By Def. 36 (c) it also does not instantiate variables in the domain or the range of any other pumping substitution θ_i.

[17] $(a_1 \ldots a_n) \sqsubset (b_1 \ldots b_m)$ iff $n = 0$ and $m > 0$ or $a_1 < b_1$ or $a_1 = b_1$ and $(a_2 \ldots a_n) \sqsubset (b_2 \ldots b_m)$.

Since q contains $\overline{\ell}_k\,\theta_1^n \dots \theta_j^{n-1} \dots \theta_d^n\,\delta'_k$ at parallel positions ι_k (for $1 \le k \le d$), this results in a d-ary tree of rewrite sequences with root $\overline{\ell}_j\,\theta_1^n \dots \theta_d^n$ for a $1 \le j \le d$ which is complete up to height n. The reason is that in the beginning, there are n substitutions θ_j for each $1 \le j \le d$ and each rewrite step removes one of them.

Hence, the tree has at least $\lfloor \frac{d^{n+1}-1}{d-1} \rfloor$ nodes. By Lemma 50, $\theta_1 \dots \theta_d$ does not duplicate variables. Thus, $\left|\overline{\ell}_j\,\theta_1^n \dots \theta_d^n\right|$ is linear in n and we get $\mathrm{rc}_{\mathcal{R}}(n) \in \Omega(d^n)$. □

Theorem 38 (Loop Detection Subsumes Induction for Linear Bounds) *Let $\mathcal{R}$ be a TRS and $\mathcal{L}$ be the set of rewrite lemmas that were speculated and proved by the technique of Sect. 3. If $\mathcal{R}$ is left-linear and there is a rewrite lemma $s \to^{\geq rt(\overline{n})} t \in \mathcal{L}$ where $rt(\overline{n})$ is not a constant, then $\mathcal{R}$ has a decreasing loop.*

Proof Let $s \to^{\geq rt(\overline{n})} t \in \mathcal{L}$ be the "*first*" rewrite lemma where the induction technique infers non-constant runtime. Thus, in its induction proof (according to Sect. 3.3) one only uses other rewrite lemmas of the form $s' \to^{\geq rt'(\overline{n}')} t'$ where $rt'(\overline{n}')$ is a constant. Hence, in the induction proof for the conjecture $s \to^* t$, $is(\overline{n})$ is a constant and the induction hypothesis is applied at least once, i.e., $ih \ge 1$. Therefore the rewrite sequence $s[n/n+1] \rightharpoonup^*_{\mathrm{IH},n} t[n/n+1]$ of Thm. 14 starts with $s[n/n+1] \rightharpoonup^* C[s\sigma]$ for some context C and an increasing substitution σ with $n\sigma = n$. We even have $s[n/n+1] \rightharpoonup^+ C[s\sigma]$, since $s[n/n+1] \neq C[s\sigma]$. To see this, note that $s[n/n+1] \neq s\sigma$ since n occurs in s and $n\sigma = n$. Moreover, $s\sigma$ cannot be a proper subterm of $s[n/n+1]$, since s does not contain defined symbols below the root by Def. 6.

For any $\mu : \mathcal{V}(s) \to \mathbb{N}$, Lemma 48 (a) and 49 (a) imply $s[n/n+1]\mu\!\downarrow_{\mathcal{G}/\mathcal{A}} \to^+_{\mathcal{R}} C\mu[s\sigma\mu]\!\downarrow_{\mathcal{G}/\mathcal{A}}$. The procedure in Sect. 3.2 only speculates conjectures where s has the form $f(\gamma_{\tau_1}(s_1), \dots, \gamma_{\tau_k}(s_k))$. Here, the arguments s_i of the generator functions are polynomials over the variables $\overline{n}$ with coefficients from $\mathbb{N}$. In other words, $s = f(\gamma_{\tau_1}(pol_1(\overline{n})), \dots, \gamma_{\tau_k}(pol_k(\overline{n})))$ for polynomials $pol_1, \dots, pol_k$. By the definition of generator functions (see Def. 5) we get

$$\begin{aligned} s[n/n+1]\mu\!\downarrow_{\mathcal{G}/\mathcal{A}} = f[D_1^{pol_1(\overline{n}[n/n+1]\mu)}[t_1], \dots, D_k^{pol_k(\overline{n}[n/n+1]\mu)}[t_k]] &\to^+_{\mathcal{R}} \\ C\mu\!\downarrow_{\mathcal{G}/\mathcal{A}} [f[D_1^{pol_1(\overline{n}\sigma\mu)}[t_1], \dots, D_k^{pol_k(\overline{n}\sigma\mu)}[t_k]]] &= C\mu[s\sigma\mu]\!\downarrow_{\mathcal{G}/\mathcal{A}} \end{aligned}$$

for non-empty constructor ground contexts $D_1, \dots, D_k$ and constructor ground terms $t_1, \dots, t_k$. The reason is $\gamma_{\tau_i}(pol_i(\overline{n}\mu)) \equiv_{\mathcal{G}\cup\mathcal{A}} D_i^{pol_i(\overline{n}\mu)}[t_i]$ where $t_i = \gamma_{\tau_i}(0)$.

If pol_i is a constant, we have $D_i^{pol_i(\overline{n}[n/n+1]\mu)}[t_i] = D_i^{pol_i(\overline{n}\sigma\mu)}[t_i] = q_i$ for some constructor ground term q_i. Hence, $f[D_1^{pol_1(\overline{n}[n/n+1]\mu)}[t_1], \dots, D_k^{pol_k(\overline{n}[n/n+1]\mu)}[t_k]]$ is of the form $C'[D_{i_1}^{pol_{i_1}(\overline{n}[n/n+1]\mu)}[t_{i_1}], \dots, D_{i_m}^{pol_{i_m}(\overline{n}[n/n+1]\mu)}[t_{i_m}]]$ for a ground context C' with root f which has no defined symbols below the root. Here, $1 \le i_1 < \dots < i_m \le k$, all pol_{i_j} are non-constant, and all pol_i with $i \in \{1, \dots, k\} \setminus \{i_1, \dots, i_m\}$ are constant. Similarly, $C\mu\!\downarrow_{\mathcal{G}/\mathcal{A}} [f[D_1^{pol_1(\overline{n}\sigma\mu)}[t_1], \dots, D_k^{pol_k(\overline{n}\sigma\mu)}[t_k]]] = C\mu\!\downarrow_{\mathcal{G}/\mathcal{A}} [C'[D_{i_1}^{pol_{i_1}(\overline{n}\sigma\mu)}[t_{i_1}], \dots, D_{i_m}^{pol_{i_m}(\overline{n}\sigma\mu)}[t_{i_m}]]]$. Note that the length of the reduction

$$\begin{aligned} & C'[D_{i_1}^{pol_{i_1}(\overline{n}[n/n+1]\mu)}[t_{i_1}], \dots, D_{i_m}^{pol_{i_m}(\overline{n}[n/n+1]\mu)}[t_{i_m}]] \\ \to^+_{\mathcal{R}} \quad & C\mu\!\downarrow_{\mathcal{G}/\mathcal{A}} [C'[D_{i_1}^{pol_{i_1}(\overline{n}\sigma\mu)}[t_{i_1}], \dots, D_{i_m}^{pol_{i_m}(\overline{n}\sigma\mu)}[t_{i_m}]]] \end{aligned} \tag{15}$$

is at most $\mathit{is}(\overline{n}\mu)$. Thus, its length is *constant* and does not depend on n. Therefore, the subterms $t_{i_1}, \ldots, t_{i_m}$ are not crucial for the reduction, i.e., we also have

$$\begin{array}{ll} & C'[D_{i_1}^{pol_{i_1}(\overline{n}[n/n+1]\mu)}[x_1], \ldots, D_{i_m}^{pol_{i_m}(\overline{n}[n/n+1]\mu)}[x_m]] \\ \to_{\mathcal{R}}^{+} & C\mu{\downarrow_{\mathcal{G}/\mathcal{A}}}\, [C'[D_{i_1}^{pol_{i_1}(\overline{n}\sigma\mu)}[x_1], \ldots, D_{i_m}^{pol_{i_m}(\overline{n}\sigma\mu)}[x_m]]] \end{array}$$

for fresh pairwise different variables $x_1, \ldots, x_m$. The reason is that each t_{i_j} is embedded in the context $D_{i_j}^{pol_{i_j}(\overline{n}[n/n+1]\mu)}$. Since pol_{i_j} is not a constant, the number $pol_{i_j}(\overline{n}[n/n+1]\mu)$ depends on the instantiation μ. Therefore, the context $D_{i_j}^{pol_{i_j}(\overline{n}[n/n+1]\mu)}$ cannot be decomposed in the same constant number of rewrite steps for every μ. Hence, when replacing the subterm t_{i_j} by a fresh variable, the rewrite sequence (15) is still possible. Note that while several of the subterms t_{i_j} might be equal, we can still replace them by pairwise different fresh variables without affecting the rewrite sequence, since $\mathcal{R}$ is left-linear.

By choosing an arbitrary fixed instantiation $\mu : \mathcal{V}(s) \to \mathbb{N}$, we obtain

$$C'[D_{i_1}^{c_1+d_1}[x_1], \ldots, D_{i_m}^{c_m+d_m}[x_m]] \quad \to_{\mathcal{R}}^{+} \quad C\mu{\downarrow_{\mathcal{G}/\mathcal{A}}}\, [C'[D_{i_1}^{c_1}[x_1], \ldots, D_{i_m}^{c_m}[x_m]]]$$

for constants $c_1, ..., c_m, d_1, ..., d_m \in \mathbb{N}$. As C' is ground, $C'[D_{i_1}^{c_1+d_1}[x_1], ..., D_{i_m}^{c_m+d_m}[x_m]]$ is linear. Thus, this reduction sequence is a decreasing loop with the pumping substitution $\theta = [x_1/D_{i_1}^{d_1}[x_1], \ldots, x_m/D_{i_m}^{d_m}[x_m]]$ and result substitution $\sigma = \varnothing$. □

To prove Thm. 39, we need several auxiliary lemmas. In the following, we restrict ourselves to linear TRSs $\mathcal{R}$ containing only rules $\ell \to r$ where both ℓ and r are basic. We first show that every rewrite sequence with basic terms gives rise to a corresponding narrowing sequence starting with a basic term $f(x_1, \ldots, x_k)$. For our restricted class of TRSs, a basic term s *narrows* to t ("$s \rightsquigarrow_{\mathcal{R}} t$") iff there is a variable-renamed rule $\ell \to r \in \mathcal{R}$ with $\sigma = \mathrm{mgu}(s, \ell)$ and $t = r\sigma$.

Lemma 51 (From Rewrite Sequences to Narrowing Sequences) *Let $\mathcal{R}$ be a linear TRS where the terms in all rules are basic. Let $m \in \mathbb{N}$ and let $s \in \mathcal{T}_B$ with $\mathrm{root}(s) = f$ such that $s \to_{\mathcal{R}}^{m} t$. Then we have $f(x_1, \ldots, x_k) \rightsquigarrow_{\mathcal{R}}^{m} t'$ for pairwise different variables $x_1, \ldots, x_k$, where t' matches t.*

Proof The proof is done by induction on m. For $m = 0$, the claim is trivial. In the induction step, we have $s \to_{\mathcal{R}}^{m} \ell\delta \to_{\mathcal{R}} r\delta = t$ for some rule $\ell \to r \in \mathcal{R}$ and some substitution δ. The induction hypothesis implies $s \rightsquigarrow_{\mathcal{R}}^{m} u$ where u matches $\ell\delta$. W.l.o.g., ℓ is variable-disjoint from u. Thus, u and ℓ are unifiable. Let $\theta = \mathrm{mgu}(u, \ell)$. Hence, there exists a substitution μ such that $\theta\mu$ is like δ on the variables of ℓ. Then we have $u \rightsquigarrow_{\mathcal{R}} r\delta$ and $r\theta\mu = r\delta = t$, i.e., $r\theta$ matches t. □

Moreover, we need the following lemma.

Lemma 52 (Size of Unified Terms) *Let $s, t \in \mathcal{T}_B$ be linear terms such that $\mathcal{V}(s) \cap \mathcal{V}(t) = \varnothing$ and let $\mathrm{mgu}(s, t) = \theta$. Then for all $x \in \mathcal{V} \setminus \mathcal{V}(t)$ we have $|x\theta| \leq |t|$.*

Proof If $x \notin \mathrm{dom}(\theta)$, then we trivially have $|x\theta| = |x| = 1 \leq |t|$. Otherwise, $x \in \mathrm{dom}(\theta)$ and $x \notin \mathcal{V}(t)$ imply $x \in \mathcal{V}(s)$. Since s is linear, there is a unique position π such that $s|_\pi = x$. Moreover, $x \in \mathrm{dom}(\theta)$ implies $\pi \in \mathrm{pos}(t)$. Since t is linear and $\mathcal{V}(s) \cap \mathcal{V}(t) = \varnothing$, we have $x\theta = t|_\pi$ and thus $|x\theta| = |\,t|_\pi\,| \leq |t|$. □

Now we can show that a linear lower bound is equivalent to non-termination of narrowing for a term of the form $f(x_1, \ldots, x_k)$.

Lemma 53 (Linear Lower Bound Means Non-Termination of Narrowing) *Let $\mathcal{R}$ be a linear TRS where the terms in all rules are basic. Then there is an infinite narrowing sequence that starts with a basic term $f(x_1, \ldots, x_k)$ for pairwise different variables $x_1, \ldots, x_k$ iff $\mathrm{rc}_{\mathcal{R}}(n) \in \Omega(n)$.*

Proof For the "only if" direction, we have an infinite sequence $f(x_1, \ldots, x_k) = t_0 \overset{\sigma_1}{\rightsquigarrow}_{\mathcal{R}} t_1 \overset{\sigma_2}{\rightsquigarrow}_{\mathcal{R}} \ldots$ for pairwise different variables x_i, where σ_i is the mgu used in the i-th narrowing step. Since the terms in all rules of $\mathcal{R}$ are basic, all σ_i are constructor substitutions (i.e., $\mathrm{range}(\sigma_i)$ does not contain defined symbols). As t_0 and $\mathcal{R}$ are linear, $t_0\sigma_1 \ldots \sigma_i$ is linear and basic, and we have $|\mathcal{V}(t_0\sigma_1 \ldots \sigma_i)| \leq k$ for all $i \in \mathbb{N}$. By induction on i, we now prove that there is a $c \in \mathbb{N}$ such that $|t_0\sigma_1 \ldots \sigma_i| \leq |t_0| + c \cdot i$ for all $i \in \mathbb{N}$. Then the infinite family of rewrite sequences $t_0\sigma_1 \ldots \sigma_i \to^i_{\mathcal{R}} t_i$ is a witness for $\mathrm{rc}_{\mathcal{R}}(n) \in \Omega(n)$. Let $c = \max\{|\ell| \mid \ell \to r \in \mathcal{R}\} \cdot k$.

The case $i = 0$ is trivial. In the induction step, by the induction hypothesis we know $|t_0\sigma_1 \ldots \sigma_i| \leq |t_0| + c \cdot i$. Note that σ_{i+1} is the mgu of t_i and ℓ, both of which are linear. Hence by Lemma 52 we have $|x\sigma_{i+1}| \leq |\ell|$ for all $x \in \mathcal{V}(t_0\sigma_1 \ldots \sigma_i)$. As $t_0\sigma_1 \ldots \sigma_i$ is linear and $|\mathcal{V}(t_0\sigma_1 \ldots \sigma_i)| \leq k$, this implies $|t_0\sigma_1 \ldots \sigma_{i+1}| \leq |t_0\sigma_1 \ldots \sigma_i| + |\ell| \cdot k \leq |t_0\sigma_1 \ldots \sigma_i| + c \leq |t_0| + c \cdot (i+1)$.

For the "if" direction, $\mathrm{rc}_{\mathcal{R}}(n) \in \Omega(n)$ implies that the TRS does not have constant runtime complexity. Hence, for each $m \in \mathbb{N}$ there is a rewrite sequence of length m staring with a basic term $f(\ldots)$. Since $\Sigma_{def}(\mathcal{R})$ is finite, there exists an $f \in \Sigma_{def}(\mathcal{R})$ such that there are rewrite sequences of lengths $m_1 < m_2 < m_3 < \ldots$ that start with basic terms with root symbol f. By Lemma 51 this means that the term $f(x_1, \ldots, x_k)$ starts narrowing sequences of lengths $m_1 < m_2 < m_3 < \ldots$, i.e., the narrowing tree with the root $f(x_1, \ldots, x_k)$ has infinitely many nodes. Since $\rightsquigarrow_{\mathcal{R}}$ is finitely branching, by König's Lemma the tree has an infinite path, i.e., there is an infinite narrowing sequence starting with $f(x_1, \ldots, x_k)$. □

We now show that non-termination of narrowing is equivalent to non-termination of rewriting on possibly *infinite* basic terms. To prove this equivalence, we need the following auxiliary lemma.

Lemma 54 (Unification with Infinite Terms) *Let s, t be variable-disjoint linear finite terms. If there is a substitution σ such that $s\sigma = t\sigma$ and $\mathrm{range}(\sigma)$ contains infinite terms, then s and t unify and the range of $\mathrm{mgu}(s, t)$ consists of linear finite terms.*

Proof We use structural induction on s. If $s \in \mathcal{V}$, then $\mathrm{mgu}(s, t) = [s/t]$ and if $t \in \mathcal{V}$, then $\mathrm{mgu}(s, t) = [t/s]$. Now let $s = f(s_1, \ldots, s_k)$ and since $s\sigma = t\sigma$, we have $t = f(t_1, \ldots, t_k)$. By the induction hypothesis, the ranges of the substitutions $\sigma_1 = \mathrm{mgu}(s_1, t_1), \ldots, \sigma_k = \mathrm{mgu}(s_k, t_k)$ consist of linear finite terms. Let $\mathcal{V}(\sigma_i) = \mathrm{dom}(\sigma_i) \cup \mathcal{V}(\mathrm{range}(\sigma_i))$ for all $1 \leq i \leq n$. Since s and t are variable-disjoint and linear, the sets $\mathcal{V}(\sigma_i)$ are pairwise disjoint and we have $(\mathcal{V}(s_i) \cup \mathcal{V}(t_i)) \cap \mathcal{V}(\sigma_j) = \varnothing$ for all $i \neq j$. Hence, we get $\mathrm{mgu}(s, t) = \sigma_1 \ldots \sigma_n$. As, by the induction hypothesis, $\mathrm{range}(\sigma_i)$ consists of linear terms and the sets $\mathcal{V}(\sigma_i)$ are pairwise disjoint, the range of $\mathrm{mgu}(s, t)$ consists of linear terms, too. Similarly, as $\mathrm{range}(\sigma_i)$ only contains finite terms, this also holds for $\mathrm{range}(\sigma_1 \ldots \sigma_n)$. □

Lemma 55 (Narrowing and Rewriting with Infinite Terms) *Let $\mathcal{R}$ be a linear TRS where the terms in all rules are basic. Then $\rightsquigarrow_{\mathcal{R}}$ terminates on basic terms of the form $f(x_1, \ldots, x_k)$ iff $\to_{\mathcal{R}}$ terminates on possibly infinite basic terms.*

Proof For the "if" direction, assume there is an infinite sequence $f(x_1, \ldots, x_k) = t_1 \overset{\sigma_1}{\rightsquigarrow}_{\mathcal{R}} t_2 \overset{\sigma_2}{\rightsquigarrow}_{\mathcal{R}} \ldots$ Then $t_1\sigma_1^\omega \to_{\mathcal{R}} t_2\sigma_2^\omega \to_{\mathcal{R}} \ldots$ is an infinite $\to_{\mathcal{R}}$-sequence where $\sigma_i^\omega = \sigma_i\sigma_{i+1}\ldots$ Since the terms in the rules of $\mathcal{R}$ are basic, all terms $t_i\sigma_i^\omega$ are basic, too.

For the "only if" direction, assume that there is an infinite rewrite sequence $t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} \ldots$ on possibly infinite basic terms t_i. We now show that for every finite prefix $t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} \cdots \to_{\mathcal{R}} t_m$ of this sequence, there is a rewrite sequence $t'_1 \to_{\mathcal{R}} t'_2 \to_{\mathcal{R}} \cdots \to_{\mathcal{R}} t'_m$ with *finite* and *linear* basic terms t'_i. This suffices for the current lemma, because it implies that the TRS does not have constant runtime complexity. As in the proof of Lemma 53 one can then show that there is an infinite narrowing sequence starting with a term $f(x_1, \ldots, x_k)$.

It remains to prove that for every finite rewrite sequence $t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} \cdots \to_{\mathcal{R}} t_m$ with possibly infinite basic terms t_i, there is a rewrite sequence $t'_1 \to_{\mathcal{R}} t'_2 \to_{\mathcal{R}} \cdots \to_{\mathcal{R}} t'_m$ with finite linear basic terms t'_i, where there exists a substitution σ such that $t'_i\sigma = t_i$ for all $1 \leq i \leq m$. We prove this claim by induction on m.

The case $m = 1$ is trivial. In the induction step, we consider the rewrite sequence $t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} \cdots \to_{\mathcal{R}} t_m \to_{\mathcal{R}} t_{m+1}$ of possibly infinite basic terms. By the induction hypothesis we have $t'_1 \to_{\mathcal{R}} t'_2 \to_{\mathcal{R}} \cdots \to_{\mathcal{R}} t'_m$ for finite linear basic terms t'_i where $t'_i\sigma = t_i$ for all i. Let $\ell \to r$ be the rule applied in the rewrite step from t_m to t_{m+1}, i.e., $t_m = \ell\delta$ and $t_{m+1} = r\delta$. As $t'_m\sigma = t_m = \ell\delta$ and as w.l.o.g., ℓ is variable-disjoint from the t'_i, this means that t'_m and ℓ are unifiable. Let $\theta = \mathrm{mgu}(t'_m, \ell)$. By Lemma 54, the range of θ consists of linear finite terms, as t'_m and ℓ are linear and finite. Let μ be a substitution such that $\theta\mu$ is like σ on $t'_1, \ldots, t'_m$ and like δ on ℓ. We define $t''_i = t'_i\theta$ for all $1 \leq i \leq m$ and $t''_{m+1} = r\theta$. Then we have $t''_1 \to_{\mathcal{R}} \ldots \to_{\mathcal{R}} t''_m = t'_m\theta = \ell\theta \to_{\mathcal{R}} r\theta = t''_{m+1}$. Moreover, we have $t''_i\mu = t'_i\theta\mu = t'_i\sigma = t_i$ for all $1 \leq i \leq m$ and $t''_{m+1}\mu = r\theta\mu = r\delta = t_{m+1}$.

It remains to show that all t''_i are linear. Let $\mathcal{V}' = \mathcal{V}(t'_1) \cup \ldots \cup \mathcal{V}(t'_m)$. Since ℓ is variable-disjoint from the t'_i and $\theta = \mathrm{mgu}(t'_m, \ell)$, $\mathrm{range}(\theta|_{\mathcal{V}'})$ does not contain variables from $\mathcal{V}'$. Since each t'_i is linear and $\mathrm{range}(\theta)$ consists of linear terms, this implies that $t''_1, \ldots, t''_m$ are linear, too. Similarly, t''_{m+1} is linear, as $\theta|_{\mathcal{V}(r)}$ does not contain variables from $\mathcal{V}(r)$, r is linear, and $\mathrm{range}(\theta)$ consists of linear terms. □

Lemma 53 and 55 imply that for linear TRSs where the left- and right-hand sides of all rules are basic terms, a linear lower bound is equivalent to non-termination of rewriting on possibly infinite basic terms. We now reduce the immortality problem for Turing machines to this latter problem in order to show that it is undecidable.

Let $\mathcal{M} = (Q, \Gamma, \delta)$ be a Turing machine where Q is the set of states, Γ is the tape alphabet, and $\_ \in \Gamma$ is the blank symbol. A configuration of the Turing machine has the form (q, w, a, w') with $q \in Q$, $w, w' \in \Gamma^\omega$, and $a \in \Gamma$. It means that q is the current state, the symbol at the current position of the tape is a, the symbols right of the current position are described by the infinite word w', and the symbols left of it are described by the infinite word w. To ease the formulation, if $w = b\,.\,\overline{w}$ then this means that b is the symbol directly left of the current position, i.e., w is the word obtained when reading the symbols on the tape from right to left. The transition function $\delta : (Q \times \Gamma) \to (Q \times \Gamma \times \{L, R\})$ induces a transition relation $\to_{\mathcal{M}}$ on configurations where $(q_1, w_1, a_1, w'_1) \to_{\mathcal{M}} (q_2, w_2, a_2, w'_2)$ iff either

- $w_1 = a_2 \,.\, w_2, w_2' = b \,.\, w_1'$, and $\delta(q_1, a_1) = (q_2, b, L)$ or
- $w_2 = b \,.\, w_1, w_1' = a_2 \,.\, w_2'$, and $\delta(q_1, a_1) = (q_2, b, R)$

We say that $\mathcal{M}$ is *mortal* iff there is no infinite sequence $(q_1, w_1, a_1, w_1') \to_{\mathcal{M}} (q_2, w_2, a_2, w_2') \to_{\mathcal{M}} \ldots$ of configurations. The difference to the halting problem is that for the mortality problem, one may start with a tape containing infinitely many non-blank symbols. Moreover, one can begin with any state $q_1 \in Q$. As shown in [20], the (im)mortality problem for Turing machines is undecidable.

To reduce immortality of Turing machines to the termination of rewriting with infinite terms, we use the following encoding. For any Turing machine $\mathcal{M} = (Q, \Gamma, \delta)$, we define the TRS $\mathcal{R}_\mathcal{M}$. Here, f has arity 4, all symbols from Γ become function symbols of arity 1, and $Q \cup \{\underline{a} \mid a \in \Gamma\}$ are constants.

$$\begin{aligned}\mathcal{R}_\mathcal{M} = \{&\mathsf{f}(q_1, a_2(xs), \underline{a_1}, ys) \to \mathsf{f}(q_2, xs, \underline{a_2}, b(ys)) \mid a_2 \in \Gamma, \delta(q_1, a_1) = (q_2, b, L)\} \;\cup\\ \{&\mathsf{f}(q_1, xs, \underline{a_1}, a_2(ys)) \to \mathsf{f}(q_2, b(xs), \underline{a_2}, ys) \mid a_2 \in \Gamma, \delta(q_1, a_1) = (q_2, b, R)\}.\end{aligned}$$

$\mathcal{R}_\mathcal{M}$ is a linear TRS where all terms are basic. Lemma 56 shows that mortality of $\mathcal{M}$ is equivalent to termination of $\mathcal{R}_\mathcal{M}$ on possibly infinite basic terms.

Lemma 56 (Mortality of Turing machines and Rewriting) *A Turing machine $\mathcal{M}$ is immortal iff there is a possibly infinite basic term that starts an infinite rewrite sequence with $\mathcal{R}_\mathcal{M}$.*

Proof We define the following functions word and word^{-1} to convert infinite words over Γ to possibly infinite basic terms and vice versa.

$$\mathsf{word}(a \,.\, w) = a(\mathsf{word}(w)) \qquad \mathsf{word}^{-1}(t) = \begin{cases} a \,.\, \mathsf{word}^{-1}(t') & \text{if } t = a(t') \\ \_^\omega & \text{otherwise} \end{cases}$$

For each configuration (q, w, a, w') let $\mathsf{term}(q, w, a, w') = \mathsf{f}(q, \mathsf{word}(w), \underline{a}, \mathsf{word}(w'))$.

For the "only if" direction, it suffices to show that $(q_1, w_1, a_1, w_1') \to_\mathcal{M} (q_2, w_2, a_2, w_2')$ implies $\mathsf{term}(q_1, w_1, a_1, w_1') \to_{\mathcal{R}_\mathcal{M}} \mathsf{term}(q_2, w_2, a_2, w_2')$. We regard the case where $\delta(q_1, a_1) = (q_2, b, L)$ (the case $\delta(q_1, a_1) = (q_2, b, R)$ works analogously). Then $w_1 = a_2 \,.\, w_2$ and $w_2' = b \,.\, w_1'$. Thus, $\mathsf{term}(q_1, w_1, a_1, w_1') = \mathsf{f}(q_1, a_2(\mathsf{word}(w_2)), \underline{a_1}, \mathsf{word}(w_1')) \to_\mathcal{M} \mathsf{f}(q_2, \mathsf{word}(w_2), \underline{a_2}, b(\mathsf{word}(w_1'))) = \mathsf{term}(q_2, w_2, a_2, w_2')$, as desired.

For the "if" direction, it suffices to show that if t_1 is a possibly infinite basic term with $t_1 \to_{\mathcal{R}_\mathcal{M}} t_2$, then $\mathit{conf}_1 \to_\mathcal{M} \mathit{conf}_2$ with $\mathit{conf}_i = (t_i|_1, \mathsf{word}^{-1}(t_i|_2), \mathsf{char}(t_i|_3), \mathsf{word}^{-1}(t_i|_4))$ for both $i \in \{1, 2\}$, where $\mathsf{char}(\underline{a}) = a$. Clearly, we have $t_i|_1 \in Q$ and $t_i|_3 \in \{\underline{a} \mid a \in \Gamma\}$. We regard the case where the rule of $\mathcal{R}_\mathcal{M}$ used for the rewrite step corresponds to a shift to the left (the right shift works analogously). Then we have $t_1 = \mathsf{f}(q_1, a_2(s), \underline{a_1}, s')$ and $t_2 = \mathsf{f}(q_2, s, \underline{a_2}, b(s'))$ for $a_1, a_2, b \in \Gamma$, some possibly infinite constructor terms s, s', and $q_1, q_2 \in Q$. Moreover, by construction we have $\delta(q_1, a_1) = (q_2, b, L)$. Hence, $\mathit{conf}_1 = (q_1, a_2 \,.\, \mathsf{word}^{-1}(s), a_1, \mathsf{word}^{-1}(s')) \to_\mathcal{M} (q_2, \mathsf{word}^{-1}(s), a_2, b \,.\, \mathsf{word}^{-1}(s')) = \mathit{conf}_2$. □

Theorem 39 (Undecidability and Incompleteness of Loop Detection for Linear Bounds) *For the class of linear TRSs where $\ell, r \in \mathcal{T}_B$ for all rewrite rules $\ell \to r$, it is not semi-decidable if $\mathrm{rc}_\mathcal{R}(n) \in \Omega(n)$ holds. Hence for this class, loop detection is not complete for the inference of linear lower bounds.*

Proof For any Turing machine $\mathcal{M}$ we have the following:

$$\begin{array}{ll} & \mathcal{M} \text{ is immortal} \\ \text{iff} & \to_{\mathcal{R}_\mathcal{M}} \text{ does not terminate on possibly infinite basic terms} \quad \text{(by Lemma 56)} \\ \text{iff} & \rightsquigarrow_{\mathcal{R}_\mathcal{M}} \text{ does not terminate on basic terms } f(x_1, \ldots, x_k) \quad \text{(by Lemma 55)} \\ \text{iff} & \mathrm{rc}_{\mathcal{R}_\mathcal{M}}(n) \in \Omega(n) \quad \text{(by Lemma 53)} \end{array}$$

Note that for linear TRSs $\mathcal{R}$ where the terms in all rules are basic, it is obviously semi-decidable if $\rightsquigarrow_{\mathcal{R}_\mathcal{M}}$ terminates on basic terms of the form $f(x_1, \ldots, x_k)$. Thus, a semi-decision procedure for $\mathrm{rc}_{\mathcal{R}_\mathcal{M}}(n) \in \Omega(n)$ would result in a decision procedure for mortality of Turing machines. However, mortality of Turing machines is known to be undecidable [20]. Thus, $\mathrm{rc}_\mathcal{R}(n) \in \Omega(n)$ cannot be semi-decidable for the class of linear TRSs $\mathcal{R}$ where $\ell, r \in \mathcal{T}_B$ holds for all rewrite rules $\ell \to r$.

Note that the existence of decreasing loops is semi-decidable, since one can recursively enumerate all possible rewrite sequences $\ell \to_\mathcal{R}^+ C[r]$ and since it is decidable whether an actual rewrite sequence is a decreasing loop. This implies that loop detection by decreasing loops cannot be complete for linear lower bounds of linear TRSs where the terms in all rules are basic. □

Theorem 46 (Linear Lower Bounds for irc **by Loop Detection)** *If a TRS $\mathcal{R}$ has an innermost decreasing loop, then we have* $\mathrm{irc}_\mathcal{R}(n) \in \Omega(n)$.

Proof Let $\theta = [x_i/\ell|_{\xi_i} \mid 1 \leq i \leq m]$ be the pumping substitution of the innermost decreasing loop. We show that for all $n \in \mathbb{N}$ and any substitution δ where $\overline{\ell}\,\theta^n\,\delta$ is innermost terminating, we have $\overline{\ell}\,\theta^n\,\delta \;\xrightarrow{\mathrm{i}}_\mathcal{R}^+ \circ \trianglerighteq\; \overline{\ell}\,\theta^{n-1}\,\delta'$ for a substitution δ'. Thus, these rewrite steps can be repeated n times. In the following, let $\overline{\delta}$ be a substitution such that for all $x \in \mathcal{V}(\overline{\ell}\,\theta^n)$, we have $x\,\delta \xrightarrow{\mathrm{i}}_\mathcal{R}^* x\,\overline{\delta}$ and $x\,\overline{\delta}$ is in normal form. Then we obtain

$$\begin{array}{lllll} \overline{\ell}\,\theta^n\,\delta & \xrightarrow{\mathrm{i}}_\mathcal{R}^* & \overline{\ell}\,\theta^n\,\overline{\delta} & = \ell\,\theta^{n-1}\,\overline{\delta} \quad \xrightarrow{\mathrm{i}}_\mathcal{R}^+ & C[r]\,\theta^{n-1}\,\overline{\delta} \\ \trianglerighteq & & r\,\theta^{n-1}\,\overline{\delta} & = \overline{\ell}\,\sigma\,\theta^{n-1}\,\overline{\delta} \overset{(\star)}{=} & \overline{\ell}\,\theta^{n-1}\,(\sigma\,\theta^{n-1})|_{\mathrm{dom}(\sigma)}\,\overline{\delta} \;=\; \overline{\ell}\,\theta^{n-1}\,\delta' \end{array}$$

for the substitution $\delta' = (\sigma\,\theta^{n-1})|_{\mathrm{dom}(\sigma)}\,\overline{\delta}$. The rewrite sequence $\ell\,\theta^{n-1}\,\overline{\delta} \;\xrightarrow{\mathrm{i}}_\mathcal{R}^+ C[r]\,\theta^{n-1}\,\overline{\delta}$ is indeed an innermost reduction. To see this, recall that $\overline{\delta}$ only instantiates variables by normal forms. Moreover, θ has no defined symbols in its range, since ℓ is basic. For this reason, $\theta^{n-1}\,\overline{\delta}$ also instantiates all variables by normal forms, i.e., no rewrite step is possible for the terms in the range of $\theta^{n-1}\,\overline{\delta}$. Moreover, the subterms of the redexes in the reduction $\ell \xrightarrow{\mathrm{no}}_\mathcal{R}^+ C[r]$ do not unify with left-hand sides of rules. Hence, these subterms remain in normal form if one instantiates them with $\theta^{n-1}\,\overline{\delta}$. This implies $\ell\,\theta^{n-1}\,\overline{\delta} \;\xrightarrow{\mathrm{i}}_\mathcal{R}^+ C[r]\,\theta^{n-1}\,\overline{\delta}$. The step marked with $(\star)$ holds because σ does not instantiate variables that occur in the domain or the range of θ, as in the proof of Thm. 31.

Note that for any $n \in \mathbb{N}$, the term $\overline{\ell}\,\theta^n$ is basic, since the range of θ only contains constructor terms $\ell|_{\xi_i}$ (since ξ_i cannot be the root position due to the requirement $r \notin \mathcal{V}$). Hence, if $\overline{\ell}\,\theta^n$ is not innermost terminating for some $n \in \mathbb{N}$, then we obtain $\mathrm{irc}_\mathcal{R}(n) \in \Omega(\omega)$, and therefore also $\mathrm{irc}_\mathcal{R}(n) \in \Omega(n)$. Otherwise, by the observations above, for each $n \in \mathbb{N}$ there is an innermost rewrite sequence of length n starting with $\overline{\ell}\,\theta^n$. As in the proof of Thm. 31, θ does not duplicate variables and thus, $|\overline{\ell}\,\theta^n| \in \mathcal{O}(n)$. Hence, we obtain $\mathrm{irc}_\mathcal{R}(n) \in \Omega(n)$. □