Asymptotically Precise Ranking Functions for Deterministic Size-Change Systems

Florian Zuleger(✉)

Vienna University of Technology, Vienna, Austria
zuleger@forsyte.at

Abstract. The size-change abstraction (SCA) is a popular program abstraction for termination analysis, and has been successfully implemented for imperative, functional and logic programs. Recently, it has been shown that SCA is also an attractive domain for the automatic analysis of the computational complexity of programs. In this paper, we provide asymptotically precise ranking functions for the special case of deterministic size-change systems. As a consequence we also obtain the result that the asymptotic complexity of deterministic size-change systems is exactly polynomial and that the exact integer exponent can be computed in PSPACE.

1 Introduction

The *size-change abstraction (SCA)* is a popular program abstraction for the automated termination analysis of functional [8,9], logical [10] and imperative [1] programs as well as term rewriting systems [5]; SCA is implemented in the industrial-strength systems ACL2 [9] and Isabelle [7]. Recently SCA has also been used for computing resource bounds of imperative programs [11]. SCA is a predicate abstract domain that consists of Boolean combinations of inequality constraints of shape $x \geq y'$ or $x > y'$ in disjunctive normal form. SCA variables take values in the natural numbers and should be considered as norms on the program state. The main reason, that makes SCA an attractive domain for practical termination analysis is that size-change predicates such as $x \geq y'$ can be extracted *locally* from the program and that termination for abstracted programs can be decided in PSPACE [8]. However, the termination proofs obtained by SCA through the decision procedures in [8] do not immediately allow to understand why the program makes progress and eventually terminates. In contrast, the traditional method of proving termination by a *ranking function* provides such an understanding. A ranking function maps the program states to a well-founded domain $(W, >)$ such that every program step decreases the value of the current program state. A ranking function provides a *global argument* for termination and makes the program progress apparent. Ranking functions also allow to obtain a *bound* on the runtime of a program. If a ranking function maps to a well-founded domain $(W, >)$, the height $|W|$ of the well-founded domain provides a bound on the number of program steps. We say a ranking function is *precise*, if the transition relation of the program also has height $|W|$.

L.D. Beklemishev and D.V. Musatov (Eds.): CSR 2015, LNCS 9139, pp. 426–442, 2015.
DOI: 10.1007/978-3-319-20297-6_27

Important predecessor work has studied the construction of ranking functions for the abstract programs obtained by SCA [2–4]. Unfortunately, the cited constructions do not discuss the precision of the obtained ranking function and it is not clear how to modify these constructions to be precise. In this paper, we provide asymptotically precise ranking functions for the special case of *deterministic size-change systems* (which have been called *fan-out free* size-change systems in previous work [4]). As a consequence we obtain the additional result that the asymptotic complexity of deterministic size-change systems is exactly polynomial and that the exact integer exponent can be computed in PSPACE. We give a precise statement of our contributions at the end of Sect. 2.

1.1 Related Work

Our iterated power-set construction for lexicographic ranking functions bears strong similarities with [4], which also studies the special case of deterministic size-change systems. In contrast to our approach, the ranking function in [4] is obtained via a single monolithic construction. This makes it very hard to analyze the precision of the obtained ranking function.

The size of a *set of local* ranking functions vs the size of a *single global* ranking function is studied in [2]. Interestingly this study includes the sum of variables as local ranking function, which is a crucial building block in our construction for obtaining asymptotically precise ranking functions. However, [2] restricts itself to the special case of strict inequalities $x > y'$ and does not use the sum operator in the construction of the global ranking function.

In [3] variables are allowed to take values over the integers and generalizes size-change predicates to *monotonicity constraints* which can express any inequality between two variables in a transition. The ranking function construction in [3] is elegant, but it is unclear how to obtain precise ranking functions from this construction.

A *complete characterization* of the asymptotic complexity bounds arising from SCA is given in [6] and a method for determining the exact asymptotic bound of a given abstract program is provided. For general SCA these bounds are polynomials with rational exponents. Reference [6] does not consider the special case of deterministic size-change systems whose bounds are shown to be polynomial with integral exponents in this paper. Moreover, the construction in [6] does not allow to extract ranking functions. Further, [6] does not include a complexity result.

2 Size-Change Systems (SCSs)

We fix some finite set of *size-change variables* Var. We denote by Var' the set of primed versions of the variables in Var. A *size-change predicate* (SCP) is a formula $x \triangleright y'$ with $x, y \in Var$, where $\triangleright$ is either $>$ or $\geq$. A *size-change transition* (SCT) T is a set of SCPs. An SCT T is *deterministic*, if for every variable $x \in Var$ there is at most one variable y, such that $x \triangleright y' \in T$, where

$\triangleright$ is either $>$ or $\geq$. A *size-change system* (SCS) $\mathcal{A} = (\mathit{Locs}(\mathcal{A}), \mathit{Edges}(\mathcal{A}))$ is a directed labeled graph with a finite set of locations $\mathit{Locs}(\mathcal{A})$ and a finite set of labeled edges $\mathit{Edges}(\mathcal{A})$, where every edge is labeled by an SCT. We denote an edge of an SCS by $\ell \xrightarrow{T} \ell'$. An SCS $\mathcal{A}$ is *deterministic*, if T is deterministic for every edge $\ell \xrightarrow{T} \ell'$. In the rest of the paper, we will always assume SCSs to be deterministic. We will mention determinism, when we use it. A *path* of an SCS $\mathcal{A}$ is a sequence $\ell_1 \xrightarrow{T_1} \ell_2 \xrightarrow{T_2} \cdots$ with $\ell_i \xrightarrow{T_i} \ell_{i+1}$ for all i. An SCS $\mathcal{A}$ is *strongly connected*, if for all locations $\ell, \ell' \in \mathit{Locs}(\mathcal{A})$ there is a path from ℓ to ℓ'.

We define the semantics of size-change systems by *valuations* $\sigma : \mathit{Var} \to [0, N]$ of the size-change variables to natural numbers in the interval $[0, N]$, where N is a (symbolic) natural number. We also say σ is a valuation over $[0, N]$. We denote the set of valuations by Val_N. We write $\sigma, \tau' \models x \triangleright y'$ for two valuations σ, τ, if $\sigma(x) \triangleright \tau(y)$ holds over the natural numbers. We write $\sigma, \tau' \models T$, if $\sigma, \tau' \models x \triangleright y'$ holds for all $x \triangleright y' \in T$. A *trace* of an SCS $\mathcal{A}$ is a sequence $(\ell_1, \sigma_1) \xrightarrow{T_1} (\ell_2, \sigma_2) \xrightarrow{T_2} \cdots$ such that $\ell_1 \xrightarrow{T_1} \ell_2 \xrightarrow{T_2} \cdots$ is a path of $\mathcal{A}$ and $\sigma_i, \sigma'_{i+1} \models T_i$ for all i. The *length* of a trace is the number of edges that the trace uses, counting multiply occurring edges multiple times. An SCS $\mathcal{A}$ *terminates*, if $\mathcal{A}$ does not have a trace of infinite length for any N.

Definition 1. *Let $\mathcal{A}$ be an SCS and let $(W, >)$ be a well-founded domain. We call a function rank : $\mathit{Locs}(\mathcal{A}) \times \mathit{Val}_N \to W_N$ a* ranking function *for $\mathcal{A}$, if for every trace $(\ell_1, \sigma_1) \xrightarrow{T} (\ell_2, \sigma_2)$ of $\mathcal{A}$ we have $\mathit{rank}(\ell_1, \sigma_1) > \mathit{rank}(\ell_2, \sigma_2)$. We call the ranking function rank* asymptotically precise*, if the length of the longest trace of $\mathcal{A}$ is of order $\Omega(|W_N|)$.*

Contributions: In this paper we develop an algorithm, which either returns that a given SCS $\mathcal{A}$ does not terminate or computes a function *rank* and an integer $k \in [0, |\mathit{Var}|]$ such that $\mathit{rank} : \mathit{Locs}(\mathcal{A}) \times \mathit{Val}_N \to W_N$ is a ranking function for $\mathcal{A}$ with $|W_N| = O(N^k)$ and there is a sequence of paths $\mathit{Loop}_1, \ldots, \mathit{Loop}_k$ in $\mathcal{A}$ such that the path $((\cdots(\mathit{Loop}_k)^N \cdots \mathit{Loop}_2)^N \mathit{Loop}_1)^N$ can be completed to a trace of length $\Omega(N^k)$. The upper and lower complexity bounds show that our ranking function construction is asymptotically precise. As a corollary we get that deterministic SCSs exactly have asymptotic complexity $\Theta(N^k)$ for some $k \in [0, |\mathit{Var}|]$. Additionally, we show that the witness $\mathit{Loop}_1, \ldots, \mathit{Loop}_k$ for the lower complexity bound can be guessed in PSPACE giving rise to a PSPACE algorithm for deciding the asymptotic complexity of deterministic SCSs.

Example 1. We consider the SCS $\mathcal{A}_1$ with a single location ℓ and edges $\ell \xrightarrow{T_1} \ell, \ell \xrightarrow{T_2} \ell$ with $T_1 = \{x_1 \geq x'_2, x_2 > x'_2, x_3 \geq x'_3, x_4 \geq x'_3\}$ and $T_2 = \{x_1 \geq x'_1, x_2 > x'_1, x_3 \geq x'_4, x_4 > x'_4\}$. Our algorithm computes the ranking function $\mathit{rank}_1 = \min\{\langle x_2 + x_3, 1\rangle, \langle x_1 + x_4, 2\rangle\}$ (slightly simplified) for $\mathcal{A}_1$, where $\langle a, b\rangle$ denotes tuples ordered lexicographically. We point out that the image of rank_1 has height $O(N)$; thus rank_1 proves that $\mathcal{A}_1$ has linear complexity.

Example 2. We consider the SCS $\mathcal{A}_2$ with a single location ℓ and edges $\ell \xrightarrow{T_1} \ell, \ell \xrightarrow{T_2} \ell, \ell \xrightarrow{T_3} \ell$ with $T_1 = \{x_1 > x_1', x_2 > x_1', x_3 \geq x_3'\}$, $T_2 = \{x_1 \geq x_1', x_2 > x_2', x_3 \geq x_2'\}$ and $T_3 = \{x_1 > x_3', x_2 \geq x_2', x_3 > x_3'\}$. Our algorithm computes the ranking function $rank_2 = \min\{\langle x_1, x_2\rangle, \langle x_2, x_3\rangle, \langle x_3, x_1\rangle\}$ (slightly simplified) for $\mathcal{A}_2$. We point out that the image of $rank_2$ has height $O(N^2)$; thus $rank_2$ proves that $\mathcal{A}_2$ has quadratic complexity.

Extension to arbitrary well-founded orders: The results in this paper can easily be extended to valuations over ordinal numbers. It would only be necessary to introduce suitable machinery for dealing with arithmetic over ordinal numbers; the construction of the ranking function and the witness for the lower bound would essentially remain the same. We refrain in this paper from this extension because we want to keep the development elementary. For comparison with earlier work on SCA, where variables can take values over arbitrary well-founded orders, we sketch these extended results below: We consider valuations $\sigma : Var \to \alpha$ that map the size-change variables to ordinal numbers below α. We denote the set of valuations by Val_α. We will assume $\alpha \geq \omega$ in the following (the case $\alpha < \omega$ corresponds to the results discussed in the previous paragraph). Let $\mathcal{A}$ be some SCS. We define the *transition relation* of $\mathcal{A}$ by

$$R_{\mathcal{A}} = \{((\ell_1,\sigma_1),(\ell_2,\sigma_2)) \in (Locs(\mathcal{A}) \times Val_\alpha)^2 \mid \text{there is an SCT } T \text{ with } \ell_1 \xrightarrow{T} \ell_2 \in Edges(\mathcal{A}) \text{ and } \sigma_1, \sigma_2' \models T\}.$$

Let α be some ordinal. Let β_α be the maximal ordinal such that $\omega^{\beta_\alpha} \leq \alpha$. We set $\bar{\alpha} = \omega^{\beta_\alpha}$. We note that we always have $\bar{\alpha} \leq \alpha \leq \bar{\alpha}c$ for some natural number c. The algorithm in this paper can be adapted such that it either returns that a given SCS $\mathcal{A}$ does not terminate or computes a function *rank* and an integer $k \in [0, |Var|]$ such that $rank : Locs(\mathcal{A}) \times Val_\alpha \to W_\alpha$ is a ranking function for $\mathcal{A}$ with $|W_\alpha| \leq \alpha^k d$ for some natural number d and there is a sequence of paths $Loop_1, \ldots, Loop_k$ in $\mathcal{A}$ such that every path in $P(i_1, \ldots, i_k)$ can be completed to a trace, where $(i_1, \ldots, i_k) \in \bar{\alpha}^k$, $P(i_1, \ldots, i_k) = \{Loop_j \pi \mid \pi \in P(i_1', \ldots, i_k')$ and $i_1 = i_1', \ldots, i_{j-1} = i_{j-1}', i_j > i_j'\}$ and $P(0, \ldots, 0) = \{\epsilon\}$, with ϵ being the empty path. This establishes $\bar{\alpha}^k \leq |R_{\mathcal{A}}| \leq \alpha^k d$ and thus our construction characterizes the height or the transition relation of $\mathcal{A}$ up to a constant factor $d < \omega$. Additionally, the witness $Loop_1, \ldots, Loop_k$ for the lower bound can be guessed in PSPACE giving rise to a PSPACE algorithm for deciding the height of the transition relation of a given SCS up to a constant factor $d < \omega$.

Structure of the paper: In Sect. 3 we develop our main technical tool, an iterated application of the well-known powerset construction from automata theory. In Sect. 4 we define for-loops, which will be employed for establishing the lower complexity bounds. In Sect. 5 we develop several technical devices for the construction of ranking functions. In Sect. 6 we state our construction of ranking functions for SCSs; we apply our algorithm to Examples 1 and 2. We refer the reader to these examples for an illustration of the concepts in this paper.

3 Adding Contexts to SCSs

In the following we define a construction for adding context to SCSs. This construction mimics the powerset construction in automata theory.

Let T be an SCT. We define $suc_T : \mathbf{2}^{Var} \to \mathbf{2}^{Var}$ by $suc_T(V) = \{y \in Var \mid \text{exists } x \in Var \text{ with } x \triangleright y' \in T\}$. Let $\mathcal{A}$ be an SCS and let $\pi = \ell_1 \xrightarrow{T_1} \ell_2 \xrightarrow{T_2} \cdots$ be a finite path of $\mathcal{A}$. We define $suc_\pi : \mathbf{2}^{Var} \to \mathbf{2}^{Var}$ by $suc_\pi = \cdots \circ suc_{T_2} \circ suc_{T_1}$.

We have the following property from the powerset-like construction of suc:

Proposition 1 (Monotonicity). *Let $V_1 \subseteq V_2 \subseteq Var$. We have $suc_T(V_1) \subseteq suc_T(V_2)$ for every SCT T and $suc_\pi(V_1) \subseteq suc_\pi(V_2)$ for every path π.*

For deterministic SCTs and SCSs we have the following property:

Proposition 2 (Decrease of Cardinality). *Let $V \in \mathbf{2}^{Var}$. We have $|V| \geq |suc_T(V)|$ for every deterministic SCT T. We have $|V| \geq |suc_\pi(V)|$ for every path π of an deterministic $\mathcal{A}$.*

Definition 2 (Context). *A* context *of* length $k \in [0, |Var|]$ *is a sequence $\langle C_1, \ldots, C_k\rangle \in (\mathbf{2}^{Var})^k$ with $C_i \subseteq C_j$ for all $1 \leq i < j \leq k$. We denote the context of length $k = 0$ by ϵ. Let $\mathcal{C} = \langle C_1, \ldots, C_k\rangle$ be a context of length k. We call $\mathcal{C}$* proper*, if $C_i \subsetneq C_j$ for all $0 \leq i < j \leq k$, setting $C_0 = \emptyset$. We define the operation of retrieving the* last *component of $\mathcal{C}$ by $last(\mathcal{C}) = C_k$ for $k \geq 1$ and $last(\mathcal{C}) = \emptyset$ for $k = 0$. Given $C \in \mathbf{2}^{Var}$, we define the operation $\mathcal{C} :: C = \langle C_1, \ldots, C_k, C\rangle$ of* extending *$\mathcal{C}$ by C to a context of length $k+1$. For $k \geq 1$, we define the operation of* removing *the last component $tail(\mathcal{C}) = \langle C_1, \ldots, C_{k-1}\rangle$ from $\mathcal{C}$. For $k \geq 1$, we define the* current variables *of $\mathcal{C}$ by $curr(\mathcal{C}) = C_k \setminus C_{k-1}$, setting $C_0 = \emptyset$.*

We fix a finite set of locations *locs*. In the following we define SCSs with contexts over this set of locations *locs*. In the rest of the paper SCSs with contexts will always refer to this set of locations *locs*.

Definition 3 (SCSs with Contexts). *An SCS $\mathcal{A}$ has* contexts *of length k, if $Locs(\mathcal{A}) \subseteq locs \times (\mathbf{2}^{Var})^k$, if $\mathcal{C}$ is a context for every $(\ell, \mathcal{C}) \in Locs(\mathcal{A})$, and if for every edge $(\ell, \langle C_1, \ldots, C_k\rangle) \xrightarrow{T} (\ell', \langle C'_1, \ldots, C'_k\rangle) \in Edges(\mathcal{A})$ we have $suc_T(C_i) = C'_i$ for all $1 \leq i \leq k$.*

Lemma 1. *Let $\mathcal{A}$ be an SCS with contexts of length k. Let $(\ell, \langle C_1, \ldots, C_k\rangle)$ and $(\ell', \langle C'_1, \ldots, C'_k\rangle)$ be two locations of $Locs(\mathcal{A})$ that belong to the same SCC of $\mathcal{A}$. We have $|C_i| = |C'_i|$ for all $1 \leq i \leq k$.*

Proof. Because $(\ell, \langle C_1, \ldots, C_k\rangle)$ and $(\ell', \langle C'_1, \ldots, C'_k\rangle)$ are in the same SCC of $\mathcal{A}$, there is a path π from $(\ell, \langle C_1, \ldots, C_k\rangle)$ to $(\ell', \langle C'_1, \ldots, C'_k\rangle)$ with $suc_\pi(C_i) = C'_i$ for all $1 \leq i \leq k$. By Proposition 2 we have $|C_i| \geq |C'_i|$ for all $1 \leq i \leq k$. By a symmetrical argument we also get $|C'_i| \geq |C_i|$ for all $1 \leq i \leq k$.

Definition 4. (Adding Contexts to SCSs). *Let $\mathcal{A}$ be an SCS with contexts of length k. We define $\mathcal{A}' = History(\mathcal{A})$ to be the SCS with contexts of length $k+1$ whose set of locations $Locs(\mathcal{A}')$ and edges $Edges(\mathcal{A}')$ is the least set such that*

- $(\ell, \mathcal{C} :: Var) \in Locs(\mathcal{A}')$ *for every* $(\ell, \mathcal{C}) \in Locs(\mathcal{A})$, *and*
- *if* $(\ell, \mathcal{C} :: C) \in Locs(\mathcal{A}')$ *and* $(\ell, \mathcal{C}) \xrightarrow{T} (\ell', \mathcal{C}') \in Edges(\mathcal{A})$ *then* $(\ell, \mathcal{C} :: C) \xrightarrow{T} (\ell', \mathcal{C}' :: suc_T(C)) \in Edges(\mathcal{A}')$ *and* $(\ell', \mathcal{C}' :: suc_T(C)) \in Locs(\mathcal{A}')$.

Lemma 2. *Let $\mathcal{A}$ be a strongly connected SCS with proper contexts of length k. Then History($\mathcal{A}$) has at most $2|locs||Var|!$ locations.*

Proof. Let $(\ell, \langle C_1, \ldots, C_k \rangle) \in Locs(\mathcal{A})$ be some location of $\mathcal{A}$. We set $t = |C_k|$. By Lemma 1 we have for all locations $(\ell', \langle C'_1, \ldots, C'_k \rangle) \in Locs(\mathcal{A})$ that $|C_i| = |C'_i|$ for all $1 \leq i \leq k$. It is easy to see that there are at most $\frac{|Var|}{(|Var|-t)!}$ proper contexts $\langle C'_1, \ldots, C'_k \rangle$ with $|C_i| = |C'_i|$ for all $1 \leq i \leq k$. We get $|Locs(History(\mathcal{A}))| \leq |locs| \frac{|Var|!}{(|Var|-t)!} 2^{|Var|-t} \leq 2|locs||Var|!$, because there are at most $2^{|Var|-t}$ possibilities for the last component of a context in $History(\mathcal{A})$.

Lemma 3. *If $\mathcal{A}$ is strongly connected, History($\mathcal{A}$) has a unique sink SCC.*

Proof. Let $\mathcal{A}' = History(\mathcal{A})$. We show that $\mathcal{A}'$ has a unique sink SCC by the following argument: Let $(\ell_1, \mathcal{C}_1 :: C_1), (\ell_2, \mathcal{C}_2 :: C_2) \in Locs(\mathcal{A}')$ be arbitrary locations in sink SCCs of $\mathcal{A}'$. Then $(\ell_2, \mathcal{C}_2 :: C_2)$ is reachable from $(\ell_1, \mathcal{C}_1 :: C_1)$.

By Definition 4 there is a location $(\ell, \mathcal{C}) \in Locs(\mathcal{A})$ and a path π in $\mathcal{A}$ from $(\ell, \mathcal{C})$ to $(\ell_2, \mathcal{C}_2)$ with $suc_\pi(Var) = C_2$. Because $\mathcal{A}$ is strongly connected, there is a path π' from $(\ell_1, \mathcal{C}_1)$ to $(\ell, \mathcal{C})$. Let $\pi_{1,2}$ be the concatenation of π' and π, which is a path from $(\ell_1, \mathcal{C}_1)$ to $(\ell_2, \mathcal{C}_2)$. By definition, $History(\mathcal{A})$ has a path from $(\ell_1, \mathcal{C}_1 :: C_1)$ to $(\ell_2, \mathcal{C}_2 :: suc_{\pi_{1,2}}(C_1))$. We show that $suc_{\pi_{1,2}}(C_1) = C_2$.

By definition of $\pi_{1,2}$ and by Proposition 1 we have

$$suc_{\pi_{1,2}}(C_1) = suc_\pi(suc_{\pi'}(C_1)) \subseteq suc_\pi(Var) = C_2. \qquad (*)$$

Because $(\ell_2, \mathcal{C}_2 :: suc_{\pi_{1,2}}(C_1))$ is reachable from $(\ell_1, \mathcal{C}_1 :: C_1)$ and because $(\ell_1, \mathcal{C}_1 :: C_1)$ belongs to a sink SCC, $(\ell_2, \mathcal{C}_2 :: suc_{\pi_{1,2}}(C_1))$ must belong to the same SCC as $(\ell_1, \mathcal{C}_1 :: C_1)$. By Lemma 1 we have $|C_1| = |suc_{\pi_{1,2}}(C_1)|$. With (*) we get $|C_1| \leq |C_2|$. By a symmetrical argument we get $|C_2| \leq |C_1|$. From $|C_1| = |C_2|$ and (*) we finally get $suc_{\pi_{1,2}}(C_1) = C_2$.

Lemma 3 allows us to make the following definition:

Definition 5. *Let $\mathcal{A}$ be a strongly connected SCS. We denote by Context($\mathcal{A}$) the unique sink SCC of History($\mathcal{A}$).*

Definition 6 (Loop). *Let $\mathcal{A}$ be an SCS with contexts. We call a cyclic path π of $\mathcal{A}$ a* loop *for a location $(\ell, \mathcal{C}) \in Locs(\mathcal{A})$, if (1) π starts and ends in ℓ and (2) $suc_\pi(Var) = last(\mathcal{C})$.*

We obtain from Lemma 3 that all locations of $Context(\mathcal{A})$ have loops:

Lemma 4. *Let $\mathcal{A}$ be a strongly connected SCS. Every location $(\ell, \mathcal{C}) \in Locs(Context(\mathcal{A}))$ has a loop.*

Proof. Because $(\ell, \mathcal{C})$ belongs to the unique sink SCC of $History(\mathcal{A})$ by Lemma 3 there is a path π from $(\ell, tail(\mathcal{C}))$ to $(\ell, tail(\mathcal{C}))$ in $\mathcal{A}$ such that $suc_\pi(Var) = last(\mathcal{C})$. From Proposition 1 and $last(\mathcal{C}) \subseteq Var$ we get

$$suc_\pi(last(\mathcal{C})) \subseteq last(\mathcal{C}). \qquad (*)$$

By definition, $History(\mathcal{A})$ has a path from $(\ell, \mathcal{C}) = (\ell, tail(\mathcal{C}) :: last(\mathcal{C}))$ to $(\ell, tail(\mathcal{C}) :: suc_\pi(last(\mathcal{C})))$. Because $(\ell, \mathcal{C})$ belongs to the unique sink SCC, also $(\ell, tail(\mathcal{C}) :: suc_\pi(last(\mathcal{C})))$ belongs to this SCC and we get $|last(\mathcal{C})| = |suc_\pi(last(\mathcal{C}))|$ from Lemma 1. With (*) we get $last(\mathcal{C}) = suc_\pi(last(\mathcal{C}))$.

4 For-Loops

Let $\pi = \ell_1 \xrightarrow{T_1} \ell_2 \xrightarrow{T_2} \cdots \ell_l$ be a path. We write $x \rhd y \in \pi$, if there is a chain of inequalities $x = x_1 \rhd_1 x_2 \rhd_2 \cdots x_l = y$ with $x_i \rhd_i x_{i+1} \in T_i$ for all i; we note that in a deterministic SCS there is at most one chain of such inequalities. Moreover, we set $\rhd = >$, if there is at least one i with $\rhd_i = >$, and $\rhd = \geq$, otherwise.

Definition 7 (For-loop). *Let $\mathcal{A}$ be an SCS. We call a location $\ell \in Locs(\mathcal{A})$, a proper context $\langle C_1, \ldots, C_k \rangle$ and a sequence of cyclic paths $Loop_1, \ldots, Loop_k$ that starts and ends in ℓ a* for-loop *of $\mathcal{A}$ with size k, if (1) $suc_{Loop_i}(C_j) = C_j$ for all $1 \leq j \leq i \leq k$, (2) $x \rhd y \in Loop_j$ and $x, y \in C_i$ imply $\rhd = \geq$ for all $1 \leq j < i \leq k$ and $x, y \in Var$, and (3) $suc_{Loop_i}(Var) = C_i$ for all $1 \leq i \leq k$.*

Intuitively, for-loops give rise to a trace for the path

$$((\cdots(Loop_k)^N \cdots Loop_2)^N Loop_1)^N$$

for valuations over $[0, N]$ and thus provide a lower complexity bound. The proof of the following lemma is given in the appendix.

Lemma 5. *Let $\mathcal{A}$ be an SCS. Let ℓ, $\langle C_1, \ldots, C_k \rangle$ and $Loop_1, \ldots, Loop_k$ be a for-loop of $\mathcal{A}$ with size k. Then $\mathcal{A}$ has a trace of length $\Omega(N^k)$.*

5 Ranking Functions for SCSs

Lemma 6. *Let $\mathcal{A}$ be a strongly connected SCS with contexts and let $\mathcal{A}' = Context(\mathcal{A})$. For a given location $(\ell, \mathcal{C}) \in Locs(\mathcal{A})$ we denote by $ext(\ell, \mathcal{C}) = \{(\ell, \mathcal{C}') \in Locs(\mathcal{A}') \mid tail(\mathcal{C}') = \mathcal{C}\}$ the set of all locations of $\mathcal{A}'$ that extend the context $\mathcal{C}$ by an additional component. Let $rank : Locs(\mathcal{A}') \times Val_N \to W$ be a ranking function for $\mathcal{A}'$. Let $fold(rank) : Locs(\mathcal{A}) \times Val_N \to W$ be the function $fold(rank)(\ell, \sigma) = \min_{\ell' \in ext(\ell)} rank(\ell', \sigma)$. Then $fold(rank)$ is a ranking function for $\mathcal{A}$.*

Proof. Let $(\ell_1, \sigma_1) \xrightarrow{T} (\ell_2, \sigma_2)$ be a trace of $\mathcal{A}$. Let $\ell_1' \in Locs(\mathcal{A}')$ be chosen such that ℓ_1' achieves the minimum in $\min_{\ell' \in ext(\ell_1)} rank(\ell', \sigma_1)$. By construction of $Context(\mathcal{A})$ there is a path $\ell_1' \xrightarrow{T} \ell_2'$ of $\mathcal{A}'$ such that $\ell_2' = (\ell, \mathcal{C})$ and

$\ell_2 = (\ell, tail(\mathcal{C}))$ for some context $\mathcal{C}$. Because *rank* is a ranking function for $Context(\mathcal{A})$, we have $rank(\ell'_1, \sigma_1) > rank(\ell'_2, \sigma_2)$. Thus,

$$fold(rank)(\ell_1, \sigma_1) = \min_{\ell' \in ext(\ell_1)} rank(\ell', \sigma_1) = rank(\ell'_1, \sigma_1) > rank(\ell'_2, \sigma_2) \geq \min_{\ell' \in ext(\ell_2)} rank(\ell', \sigma_2) = fold(rank)(\ell_2, \sigma_2).$$

Definition 8 (Descending Edge, Stable SCS). *Let $\mathcal{A}$ be an SCS with contexts. We call an edge $(\ell, \mathcal{C}) \xrightarrow{T} (\ell', \mathcal{C}') \in Edges(\mathcal{A})$* descending*, if there are variables $x, y \in Var$ with $x \in curr(\mathcal{C})$, $y \in curr(\mathcal{C}')$ and $x > y' \in T$. We denote by $\mathcal{B} = DeleteDescending(\mathcal{A})$ the SCS with $Locs(\mathcal{B}) = Locs(\mathcal{A})$ and $Edges(\mathcal{B}) = \{\ell_1 \xrightarrow{T} \ell_2 \in Edges(\mathcal{A}) \mid \ell_1 \xrightarrow{T} \ell_2$ is not descending$\}$. We call $\mathcal{A}$* unstable*, if there is an edge $(\ell, \mathcal{C}) \xrightarrow{T} (\ell', \mathcal{C}') \in Edges(\mathcal{A})$ and variables $x, y \in Var$ with $x \in last(\mathcal{C})$, $y \in last(\mathcal{C}')$ and $x > y' \in T$; otherwise, we call $\mathcal{A}$* stable*.*

We note that a stable SCS $\mathcal{A}$ does not have descending edges.

Definition 9 (Quasi-ranking Function). *We call a function*

$$rank : Locs(\mathcal{A}) \times Val_N \to W$$

a quasi-ranking function *for $\mathcal{A}$, if for every trace $(\ell_1, \sigma_1) \xrightarrow{T} (\ell_2, \sigma_2)$ of $\mathcal{A}$ we have $rank(\ell_1, \sigma_1) \geq rank(\ell_2, \sigma_2)$.*

Lemma 7. *Let $\mathcal{A}$ be an SCS with contexts. Let $sum(\mathcal{A}) : Locs(\mathcal{A}) \times Val_N \to \mathbb{N}$ be the function $sum(\mathcal{A})((\ell, \mathcal{C}), \sigma) = \sum_{x \in curr(\mathcal{C})} \sigma(x)$. Then, $sum(\mathcal{A})$ is a quasi-ranking function for $\mathcal{A}$. Further, the value of $sum(\mathcal{A})$ is decreasing for descending edges of $\mathcal{A}$.*

Proof. Let $((\ell_1, \mathcal{C}_1), \sigma_1) \xrightarrow{T} ((\ell_2, \mathcal{C}_2), \sigma_2)$ be a trace of $\mathcal{A}$. By definition of SCSs with contexts, we have that for every $y \in curr(\mathcal{C}_2)$ there is a $x \in curr(\mathcal{C}_1)$ such that $x \triangleright y' \in T$. Moreover, we have $|curr(\mathcal{C}_1)| \geq |curr(\mathcal{C}_2)|$ by Proposition 2.

Then,

$$sum(\mathcal{A})(\ell_1, \sigma_1) = \sum_{x \in curr(\mathcal{C}_1)} \sigma_1(x) \geq \sum_{x \in curr(\mathcal{C}_2)} \sigma_2(x) = sum(\mathcal{A})(\ell_2, \sigma_2).$$

If $\ell_1 \xrightarrow{T} \ell_2$ is descending, we have

$$sum(\mathcal{A})(\ell_1, \sigma_1) = \sum_{x \in curr(\mathcal{C}_1)} \sigma_1(x) > \sum_{x \in curr(\mathcal{C}_2)} \sigma_2(x) = sum(\mathcal{A})(\ell_2, \sigma_2).$$

Definition 10. *Let $\mathcal{A}$ be an SCS. A function $rto : Locs(\mathcal{A}) \to [1, |Locs(\mathcal{A})|]$ is a* reverse topological ordering *for $\mathcal{A}$, if for every edge $\ell \xrightarrow{T} \ell' \in \mathcal{A}$ we have either $rto(\ell) > rto(\ell')$ or $rto(\ell) = rto(\ell')$ and ℓ and ℓ' belong to the same SCC of $\mathcal{A}$.*

We will use reverse topological orderings as quasi-ranking functions. It is well-known that reverse topological orderings can be computed in linear time.

Definition 11. *We denote by* $\mathbb{N}^*$ *the set of* finite sequences *over* $\mathbb{N}$*, where* $\mathbb{N}^*$ *includes the empty sequence* ϵ*. Given two sequences* $\langle x_1, \ldots, x_k\rangle, \langle y_1, \ldots, y_l\rangle \in \mathbb{N}^*$*, we denote their* concatenation *by* $\langle x_1, \ldots, x_k\rangle \oplus \langle y_1, \ldots, y_l\rangle = \langle x_1, \ldots, x_k, y_1, \ldots, y_l\rangle$*. Given two functions* $f, g : A \to \mathbb{N}^*$*, we denote their* concatenation *by* $f \oplus g : A \to \mathbb{N}^*$*, where* $(f \oplus g)(a) = f(a) \oplus g(a)$*. We denote by* $\mathbb{N}^{\leq k}$ *the sequences with* length at most k*. We say a function* $f : A \to \mathbb{N}^*$ *has* rank k*, if* $f(A) \subseteq \mathbb{N}^{\leq k}$*.*

We denote by $(\mathbb{N}^*, >)$ *the* lexicographic order*, where* $\langle x_1, \ldots, x_k\rangle > \langle y_1, \ldots, y_l\rangle$ *iff there is an index* $1 \leq i \leq \min\{k, l\}$ *such that* $x_i > y_i$ *and* $x_j = y_j$ *for all* $1 \leq j < i$*. We remark that* $(\mathbb{N}^*, >)$ *is not well-founded, but that every restriction* $(\mathbb{N}^{\leq k}, >)$ *to sequences with length at most* k *is well-founded.*

Let $\mathcal{A}$ *be an SCS. We call a ranking function* $rank : Locs(\mathcal{A}) \times Val_N \to W$ *for* $\mathcal{A}$ *a* lexicographic ranking function*, if* $W = \mathbb{N}^{\leq k}$ *for some* k*.*

Lemma 8. *Let* $\mathcal{A}$ *be an SCS. Let rto be a reverse topological ordering for* $\mathcal{A}$*. Let* $rank_S : Locs(S) \to \mathbb{N}^*$ *be a lexicographic ranking function with rank* k *for every non-trivial SCC* S *of* $\mathcal{A}$*. Let* $union(rto, (rank_S)_{SCC\ S}) : Locs(\mathcal{A}) \to \mathbb{N}^*$ *be the function* $union(rto, (rank_S)_{SCC\ S})(\ell, \sigma) = rto(\ell) \oplus rank_S(\ell, \sigma)$*, if* ℓ *belongs to some non-trivial* S*, and* $union(rto, (rank_S)_{SCC\ S})(\ell, \sigma) = rto(\ell)$*, otherwise. Then,* $union(rto, (rank_S)_{SCC\ S})$ *is a lexicographic ranking function for* $\mathcal{A}$ *with rank* $k + 1$*.*

Proof. Let $(\ell_1, \sigma_1) \xrightarrow{T} (\ell_2, \sigma_2)$ be a trace of $\mathcal{A}$. Assume there is no SCC S such that $\ell_1, \ell_2 \in S$. By Definition 10 we have $rto(\ell_1) > rto(\ell_2)$. Otherwise $\ell_1, \ell_2 \in S$ and $\ell_1 \xrightarrow{T} \ell_2 \in Edges(S)$ for some SCC S. By Definition 10 we have $rto(\ell_1) = rto(\ell_2)$. Moreover, $rank_S(\ell_1, \sigma_1) > rank_S(\ell_2, \sigma_2)$ because $rank_S$ is a ranking function for S. In both cases we get $union(rto, (rank_S)_{\text{SCC } S})(\ell_1, \sigma_1) > union(rto, (rank_S)_{\text{SCC } S})(\ell_2, \sigma_2)$. Clearly, the function $union(rto, (rank_S)_{\text{SCC } S})$ has rank $k + 1$.

Lemma 9. *Let* $\mathcal{A}$ *be an SCS with contexts and let* $\mathcal{B} = DeleteDescending(\mathcal{A})$*. Let rank be a lexicographic ranking function for* $\mathcal{B}$ *with rank* k*. Then* $sum(\mathcal{A}) \oplus rank$ *is a lexicographic ranking function for* $\mathcal{A}$ *with rank* $k + 1$*.*

Proof. Let $(\ell_1, \sigma_1) \xrightarrow{T} (\ell_2, \sigma_2)$ be a trace of $\mathcal{A}$. Assume $\ell_1 \xrightarrow{T} \ell_2$ is descending. Then we have $sum(\mathcal{A})(\ell_1, \sigma_1) > sum(\mathcal{A})(\ell_2, \sigma_2)$ by Lemma 7. Assume $\ell_1 \xrightarrow{T} \ell_2$ is not descending. Then we have $sum(\mathcal{A})(\ell_1, \sigma_1) \geq sum(\mathcal{A})(\ell_2, \sigma_2)$ by Lemma 7. Moreover $\ell_1 \xrightarrow{T} \ell_2$ is a transition of $\mathcal{B}$. Thus $rank(\ell_1, \sigma_1) > rank(\ell_2, \sigma_2)$. In both cases we get $(sum(\mathcal{A}) \oplus rank)(\ell_1, \sigma_1) > (sum(\mathcal{A}) \oplus rank)(\ell_2, \sigma_2)$. Clearly $sum(\mathcal{A}) \oplus rank$ has rank $k + 1$.

6 Main Algorithm

In the following we describe our construction of ranking functions and for-loops for SCSs. Algorithm 1 states the main algorithm $main(\mathcal{A}, i)$, which expects a

stable SCS $\mathcal{A}$ with contexts of length i as input. Algorithm 2 states the helper algorithm $mainSCC(\mathcal{A}, i)$, which expects a strongly connected stable SCS $\mathcal{A}$ with contexts of length i as input. *main* and *mainSCC* are mutually recursive. Algorithm 3 states the wrapper algorithm $ranking(\mathcal{A})$, which expects an SCS $\mathcal{A}$ with $Locs(\mathcal{A}) = locs$ and simply adds contexts of length zero to all location before calling *main*. All three algorithms return a tuple $(rank, witness, \mathcal{C}, k)$ for a given SCS $\mathcal{A}$. In Theorem 1 below we state that *rank* is a ranking function for $\mathcal{A}$ with rank $2k+1$, which proves the upper complexity bound $O(N^k)$ for $\mathcal{A}$. In Theorem 2 below we state that there is a sequence of paths $Loop_1, \ldots, Loop_k$ in $\mathcal{A}$ such that *witness*, $\mathcal{C}$ and $Loop_1, \ldots, Loop_k$ is a for-loop for $\mathcal{A}$ with size k, which proves the lower complexity bound $\Omega(N^k)$ for $\mathcal{A}$.

Example 3. We consider the SCS $\mathcal{A}_1$ from Example 1. We will identify $\mathcal{A}_1$ with the SCS obtained from $\mathcal{A}_1$ by adding contexts of zero length. Consider the call $main(\mathcal{A}_1, 0)$. $\mathcal{A}_1$ has a single SCC, namely $\mathcal{A}_1$. We consider the recursive call $mainSCC(\mathcal{A}_1, 0)$. Let $\mathcal{A}_1' := Context(\mathcal{A}_1)$. $Locs(\mathcal{A}_1')$ has two locations, namely $\ell_1 = (\ell, \{x_2, x_3\})$ and $\ell_2 = (\ell, \{x_1, x_4\})$. $Edges(\mathcal{A}_1')$ has four edges, namely $\ell_1 \xrightarrow{T_1} \ell_1$, $\ell_1 \xrightarrow{T_2} \ell_2$, $\ell_2 \xrightarrow{T_2} \ell_2$ and $\ell_2 \xrightarrow{T_1} \ell_1$. Let $\mathcal{B}_1 := DeleteDescending(\mathcal{A}_1')$. $Edges(\mathcal{B}_1)$ has the single remaining edge $\ell_2 \xrightarrow{T_1} \ell_1$. Thus, $\mathcal{B}_1$ does not have a non-trivial SCC and $main(\mathcal{B}_1, 1)$ returns the reverse topological ordering $rto_{\mathcal{B}_1} = \{\ell_1 \mapsto 1, \ell_2 \mapsto 2\}$. Then, $rank_{\mathcal{A}_1'} = sum(\mathcal{B}_1) \oplus rto_{\mathcal{B}_1} = \{(\ell_1, \sigma) \mapsto \langle \sigma(x_2) + \sigma(x_3), 1\rangle, (\ell_2, \sigma) \mapsto \langle \sigma(x_1) + \sigma(x_4), 2\rangle\}$ is a ranking function for $\mathcal{A}_1'$. Finally, $rank_{\mathcal{A}_1} = fold(rank_{\mathcal{A}_1'}) = (\ell, \sigma) \mapsto \min\{\langle \sigma(x_2) + \sigma(x_3), 1\rangle, \langle \sigma(x_1) + \sigma(x_4), 2\rangle\}$ is a ranking function for $\mathcal{A}_1$.

Example 4. We consider the SCS $\mathcal{A}_2$ from Example 2. We will identify $\mathcal{A}_2$ with the SCS obtained from $\mathcal{A}_2$ by adding contexts of zero length. $\mathcal{A}_2$ has a single SCC, namely $\mathcal{A}_2$. We consider the recursive call $mainSCC(\mathcal{A}_2, 0)$. Let $\mathcal{A}_2' := Context(\mathcal{A}_2)$. $Locs(\mathcal{A}_2')$ has three locations, namely $\ell_1 = (\ell, \{x_1\})$ and $\ell_2 = (\ell, \{x_2\})$ and $\ell_1 = (\ell, \{x_3\})$. $Edges(\mathcal{A}_2')$ has nine edges, namely $\ell_1 \xrightarrow{T_1} \ell_1$, $\ell_1 \xrightarrow{T_2} \ell_1$, $\ell_1 \xrightarrow{T_3} \ell_3$, $\ell_2 \xrightarrow{T_1} \ell_1$, $\ell_2 \xrightarrow{T_2} \ell_2$, $\ell_2 \xrightarrow{T_3} \ell_2$, $\ell_3 \xrightarrow{T_1} \ell_3$, $\ell_3 \xrightarrow{T_2} \ell_2$, and $\ell_3 \xrightarrow{T_3} \ell_3$. Let $\mathcal{B}_2 := DeleteDescending(\mathcal{A}_2')$. $Edges(\mathcal{B}_2)$ has the three remaining edges $\ell_1 \xrightarrow{T_2} \ell_1$, $\ell_2 \xrightarrow{T_3} \ell_2$ and $\ell_3 \xrightarrow{T_1} \ell_3$. Thus, $\mathcal{B}_2$ has three non-trivial SCCs consisting of a single location each. $main(\mathcal{B}_2, 1)$ returns the ranking function $rank_{\mathcal{B}_2} = (union(rto_{\mathcal{B}_2}, (rank_S)_{\text{SCC } S \text{ of } \mathcal{B}_2}) = \{(\ell_1, \sigma) \mapsto \langle 1, \sigma(x_2), 1\rangle, (\ell_2, \sigma) \mapsto \langle 1, \sigma(x_3), 1\rangle, (\ell_3, \sigma) \mapsto \langle 1, \sigma(x_1), 1\rangle\}$. Then,

$$\begin{aligned} rank_{\mathcal{A}_2'} = sum(\mathcal{B}_2) \oplus rank_{\mathcal{B}_2} = \\ \{(\ell_1, \sigma) \mapsto \langle \sigma(x_1), 1, \sigma(x_2), 1\rangle, \\ (\ell_2, \sigma) \mapsto \langle \sigma(x_2), 1, \sigma(x_3), 1\rangle, (\ell_3, \sigma) \mapsto \langle \sigma(x_3), 1, \sigma(x_1), 1\rangle\} \end{aligned}$$

is a ranking function for $\mathcal{A}_2'$. Finally,

$$rank_{\mathcal{A}_2} = fold(rank_{\mathcal{A}_2'}) = (\ell, \sigma) \mapsto \min\{\langle \sigma(x_1), 1, \sigma(x_2), 1\rangle, \langle \sigma(x_2), 1, \sigma(x_3), 1\rangle, \langle \sigma(x_3), 1, \sigma(x_1), 1\rangle\}$$

is a ranking function for $\mathcal{A}_2$.

Procedure: $main(\mathcal{A}, i)$
Input: a stable SCS $\mathcal{A}$ with contexts of length i
if *there is a loop in* $\mathcal{A}$ **then**
  raise an exception for non-termination;
foreach *non-trivial SCC* S **do**
  $(rank_S, witness_S, \mathcal{C}_S, k_S) := mainSCC(S, i)$;
if $\mathcal{A}$ *has a non-trivial SCC* **then**
  let $k := 1 + \max\{k_S \mid \text{non-trivial SCC } S \text{ of } \mathcal{A}\}$;
  let $witness := witness_S$ and $\mathcal{C} := \mathcal{C}_S$ for some S that achieves the maximum;
else
  let $k := 0$;
  choose an arbitrary location $witness \in Locs(\mathcal{A})$ and let $\mathcal{C} := \epsilon$;
compute a reverse topological ordering rto for $\mathcal{A}$;
return $(union(rto, (rank_S)_{\text{SCC } S}), witness, \mathcal{C}, k)$;

Algorithm 1. the main algorithm $main(\mathcal{A}, i)$

Procedure: $mainSCC(\mathcal{A}, i)$
Input: a strongly connected stable SCS $\mathcal{A}$ with contexts of length i
let $\mathcal{B} := DeleteDescending(Context(\mathcal{A}))$;
let $(rank_{\mathcal{B}}, witness_{\mathcal{B}}, \mathcal{C}_{\mathcal{B}}, k_{\mathcal{B}}) := main(\mathcal{B}, i+1)$;
let $(\ell, \mathcal{C}) := witness_{\mathcal{B}}$ and $\langle C_1, \ldots, C_{k_{\mathcal{B}}}\rangle := \mathcal{C}_{\mathcal{B}}$;
return $(fold(sum(\mathcal{B}) \oplus rank_{\mathcal{B}}), (\ell, tail(\mathcal{C})), \langle last(\mathcal{C}), C_1, \ldots, C_{k_{\mathcal{B}}}\rangle, k_{\mathcal{B}})$;

Algorithm 2. the helper algorithm $mainSCC(\mathcal{A}, i)$

Lemma 10. *Let $\mathcal{A}$ be a stable SCS with proper contexts of length i. Algorithm $main(\mathcal{A}, i)$ terminates.*

Proof. Let $n = |Var|$. We proceed by induction on $n-i$. Base case: $i = n$. Assume $\mathcal{A}$ has a non-trivial SCC S. We choose some location $(\ell, \mathcal{C}) \in Locs(S)$. Let π be some cyclic path for $(\ell, \mathcal{C})$ in S. By definition of an SCS with contexts, we have $suc_\pi(last(\mathcal{C})) = last(\mathcal{C})$. Because $\mathcal{C}$ is proper and $i = n$, we have $last(\mathcal{C}) = Var$. Thus π is a loop for $(\ell, \mathcal{C})$ and *main* terminates with an exception. Otherwise $\mathcal{A}$ does not have a non-trivial SCC S. Then *main* terminates because there is no recursive call.

Induction step: $i < n$. If $\mathcal{A}$ has a loop, *main* terminates with an exception. Otherwise $\mathcal{A}$ does not have a loop. If there is no non-trivial SCC S, *main* terminates because there is no recursive call. Assume there is a non-trivial SCC S. By definition $\mathcal{B} := DeleteDescending(Context(\mathcal{A}))$ has contexts of length $i+1$. Moreover, $\mathcal{B}$ has proper contexts, because $\mathcal{A}$ does not have a loop. Thus, we can infer from the induction assumption that the recursive call $main(\mathcal{B}, i+1)$ terminates.

Procedure: $ranking(\mathcal{A})$
Input: an SCS $\mathcal{A}$ with $Locs(\mathcal{A}) = locs$
let $\mathcal{B}$ be the SCS obtained from $\mathcal{A}$ by setting $Locs(\mathcal{B}) := \{(\ell, \epsilon) \mid \ell \in Locs(\mathcal{A})\}$ and $Edges(\mathcal{B}) = \{(\ell_1, \epsilon) \xrightarrow{T} (\ell_2, \epsilon) \mid \ell_1 \xrightarrow{T} \ell_2 \in Edges(\mathcal{A})\}$;
return $main(\mathcal{B}, 0)$;

Algorithm 3. the wrapper algorithm $ranking(\mathcal{A})$

The proof of the following lemma is given in the appendix.

Lemma 11. *If $ranking(\mathcal{A})$ terminates with an exception, then $\mathcal{A}$ does not terminate.*

Let $n = |Var|$ and let $m = |locs|$. We say a lexicographic ranking function *rank* is N, n, m*-bounded*, if for every $\langle x_1, \ldots, x_l \rangle$ in the image of *rank* we have $x_i \in [0, nN]$ for every odd index i and $x_i \in [1, 2mn!]$ for every even index i.

Theorem 1. *Assume $(rank, \_, \_, k) := main(\mathcal{A}, \_)$. Then rank is a N, n, m-bounded ranking function for $\mathcal{A}$ with rank $2k + 1$.*

Proof. We note for later use that by Lemma 2 we have

$$|Locs(\mathcal{A})| \leq 2mn! \ . \qquad (*)$$

The proof proceeds by induction on k. Base case $k = 0$: Then $\mathcal{A}$ does not have non-trivial SCCs, otherwise we would have $k \geq 1$. Thus $rank = union(rto, (rank_S)_{\text{SCC } S}) = rto$. By Lemma 8 *rank* is a ranking function for $\mathcal{A}$ with rank 1. By (*) we have that the image of *rto* is contained in the interval $[1, 2mn!]$. Thus *rank* is N, n, m-bounded.

Induction case $k \geq 1$: $\mathcal{A}$ has non-trivial SCCs, otherwise we would have $k = 0$. Let $k := \max\{k_S \mid \text{non-trivial SCC } S \text{ of } \mathcal{A}\}$ (*). Let S be a non-trivial SCC of $\mathcal{A}$. We consider the recursive call $(rank_S, \_, \_, k_S) := mainSCC(S, \_)$. Let $\mathcal{A}' := Context(S)$ and $\mathcal{B} := DeleteDescending(\mathcal{A}')$. We consider the recursive call $(rank_\mathcal{B}, \_, \_, k_\mathcal{B}) := main(\mathcal{B}, \_)$ in *mainSCC*. By (*) we have $k_\mathcal{B} = k_S < k$. Thus we can apply the induction assumption: we obtain that $rank_\mathcal{B}$ is a N, n, m-bounded ranking function for $\mathcal{B}$ with rank $2k_\mathcal{B} + 1$. Let $rank_{\mathcal{A}'} = sum(\mathcal{B}) \oplus rank_\mathcal{B}$. We note that the image of $sum(S)$ is contained in the interval $[0, nN]$ for valuations σ over $[0, N]$. By Lemma 9 $rank_{\mathcal{A}'}$ is a ranking function for $\mathcal{A}'$ with rank $2k_\mathcal{B} + 2$. Let $rank_S = fold(rank_{\mathcal{A}'})$. By Lemma 6 $rank_S$ is ranking function for S with rank $2k_\mathcal{B} + 2 \leq 2k$. Because this holds for every non-trivial SCC S of $\mathcal{A}$, we infer by Lemma 8 that $rank = union(rto, (rank_S)_{\text{SCC } S}$ is a ranking function for $\mathcal{A}$ with rank $2k + 1$. By (*) we have that the image of *rto* is contained in the interval $[1, 2mn!]$. Thus *rank* is N, n, m-bounded.

Corollary 1. *Let $\mathcal{A}$ be a stable SCS with $(rank, \_, \_, k) := ranking(\mathcal{A})$. Then $\mathcal{A}$ has complexity $O(N^k)$.*

Proof. By Theorem 1 *rank* is a N, n, m-bounded ranking function for $\mathcal{A}$ with rank $2k+1$. Thus the image of *rank* is of cardinality $O(N^k)$. Because the value of *rank* needs to decrease along every edge in a trace, the length of the longest trace of $\mathcal{A}$ is of asymptotic order $O(N^k)$.

Theorem 2. *Let $\mathcal{A}$ be a strongly connected stable SCS. Assume $(\_, witness, \mathcal{C}, k) := main(\mathcal{A}, \_)$. Then there is a sequence of cyclic paths $Loop_1, \ldots, Loop_k$ in $\mathcal{A}$ such that witness, $\mathcal{C}$ and $Loop_1, \ldots, Loop_k$ is a for-loop for $\mathcal{A}$ with size k.*

Proof. We proceed by induction on k. Base case $k = 0$: $\mathcal{A}$ does not have non-trivial SCCs, otherwise we would have $k \geq 1$. Let $witness \in Locs(\mathcal{A})$ be the location chosen by *main*. Clearly *witness* and $\mathcal{C} := \epsilon$ is a for-loop with size 0.

Induction case $k \geq 1$: $\mathcal{A}$ has non-trivial SCCs, otherwise we would have $k = 1$. For each non-trivial SCC S we define $(\_, witness_S, \mathcal{C}_S, k_S) := mainSCC(S, i)$. We consider the non-trivial SCC S that is selected by *main* for the maximum in $k := 1 + \max\{k_S \mid \text{non-trivial SCC } S \text{ of } \mathcal{A}\}$. Let $\mathcal{B} := DeleteDescending(Context(S))$. We consider the recursive call $(\_, witness_{\mathcal{B}}, \mathcal{C}_{\mathcal{B}}, k_{\mathcal{B}}) := main(\mathcal{B}, \_)$ in $mainSCC(S, \_)$. Because of $k_{\mathcal{B}} = k_S = k-1$ we obtain from the induction assumption that there is a sequence of paths $Loop_1, \ldots, Loop_{k_{\mathcal{B}}}$ in $\mathcal{B}$ such that $witness_{\mathcal{B}}$, $\mathcal{C}_{\mathcal{B}}$ and $Loop_1, \ldots, Loop_{k_{\mathcal{B}}}$ is a for-loop for $\mathcal{B}$ with size $k_{\mathcal{B}}$. Let $(\ell, \mathcal{C}) := witness_{\mathcal{B}}$ and let $\langle C_1, \ldots, C_{k_{\mathcal{B}}} \rangle := \mathcal{C}_{\mathcal{B}}$. We set $C = last(\mathcal{C})$. By Lemma 4 there is a cyclic path *Loop* for $(\ell, \mathcal{C})$ in $Context(S)$ with $suc_{Loop}(Var) = C$ (1). Because every $Loop_i$ is a cyclic path in $\mathcal{B} = DeleteDescending(Context(S))$ we have $suc_{Loop_i}(C) = C$ (2) and $x \triangleright y \in Loop_i$ and $x, y \in C$ implies $\triangleright = \geq$ for all $x, y \in Var$ (3). We have $C_i \subsetneq C_j$ for all $0 \leq i < j \leq k_{\mathcal{B}}$, setting $C_0 = \emptyset$, because $\mathcal{C}_{\mathcal{B}}$ is a proper context. Moreover, $suc_{Loop_i}(Var) = C_i$ for all $i \in [1, k_{\mathcal{B}}]$. From $suc_{Loop_i}(C) = C$ and Proposition 1 we get $C = suc_{Loop_i}(C) \subseteq suc_{Loop_i}(Var) = C_i$. No cyclic path $Loop_i$ is a loop in $\mathcal{B}$, otherwise $main(\mathcal{B}, \_)$ would have terminated with an exception. Thus, $C \neq C_i$ and $\langle C, C_1, \ldots, C_{k_{\mathcal{B}}} \rangle$ is a proper context (4). From (1) - (4) we get that $(\ell, \mathcal{C})$, $\langle C, C_1, \ldots, C_{k_{\mathcal{B}}} \rangle$ and $Loop, Loop_1, \ldots, Loop_{k_{\mathcal{B}}}$ is a for-loop for $Context(S)$ with size $k = k_{\mathcal{B}} + 1$.

Finally, we obtain the cyclic paths $Loop', Loop'_1, \ldots, Loop'_{k_{\mathcal{B}}}$ for $(\ell, tail(\mathcal{C}))$ in $\mathcal{A}$ from the cyclic paths $Loop, Loop_1, \ldots, Loop_{k_{\mathcal{B}}}$ for $(\ell, \mathcal{C})$ in $Context(S)$ by removing the last component from the context for every location. Then $(\ell, tail(\mathcal{C}))$, $\langle C, C_1, \ldots, C_{k_{\mathcal{B}}} \rangle$ and $Loop', Loop'_1, \ldots, Loop'_{k_{\mathcal{B}}}$ is a for-loop for $\mathcal{A}$ with size $k = k_{\mathcal{B}} + 1$.

From Theorem 2 and Lemma 5 we obtain the following corollary:

Corollary 2. *Let $\mathcal{A}$ be an SCS with $(\_, witness, \mathcal{C}, k) := ranking(\mathcal{A})$. Then $\mathcal{A}$ has complexity $\Omega(N^k)$.*

Let $\mathcal{A}$ be an SCS. In the following we describe a PSPACE algorithm that either returns that $\mathcal{A}$ does not terminate or that computes a number $k \in [1, n]$ such

that $\mathcal{A}$ has complexity $\Theta(N^k)$. We first describe a nondeterministic PSPACE algorithm P that decides whether $\mathcal{A}$ has a for-loop for some given size k. P nondeterministically guesses a location ℓ and a context $\langle C_1, \ldots, C_k \rangle$. P further guesses k cyclic paths $\mathit{Loop}_1, \ldots, \mathit{Loop}_k$ for location ℓ and then checks that (1) $\mathit{suc}_{\mathit{Loop}_i}(C_j) = C_j$ for all $1 \leq j \leq i \leq k$, (2) $x \triangleright y \in \mathit{Loop}_j$ and $x, y \in C_i$ implies $\triangleright = \geq$ for all $1 \leq j < i \leq k$ and all $x, y \in \mathit{Var}$, and (3) $\mathit{suc}_{\mathit{Loop}_i}(\mathit{Var}) = C_i$ for all $1 \leq i \leq k$. If all checks hold, P returns true, otherwise P returns false. ℓ and $\langle C_1, \ldots, C_k \rangle$ are of linear size. $\mathit{Loop}_1, \ldots, \mathit{Loop}_k$ are of exponential size in the worst case. However, $\mathit{Loop}_1, \ldots, \mathit{Loop}_k$ do not have to be constructed explicitly. Rather, the cyclic paths $\mathit{Loop}_1, \ldots, \mathit{Loop}_k$ can be guessed on the fly during the checks (1), (2) and (3). For illustration, we consider the construction of Loop_i and the check (1): P maintains a location ℓ' and a set S_j for each $1 \leq j \leq i$. P initializes these variables by $\ell' := \ell$ and $S_j := C_j$ for each $1 \leq j \leq i$. P repeats the following operation: P guesses some edge $\ell' \xrightarrow{T} \ell''$ of $\mathcal{A}$, computes $S_j := \mathit{suc}_T(S_j)$ for each $1 \leq j \leq i$ and sets $\ell' := \ell''$. P stops this iteration, if $\ell' = \ell$ and $S_j = C_j$ for each $1 \leq j \leq i$. Clearly, P can be implemented in polynomial space. The checks (2) and (3) can be implemented in a similar way and need to be performed simultaneously with check (1) in order to make sure the same cyclic paths Loop_i satisfy all three conditions. By Savitch's Theorem P can be turned into a deterministic PSPACE algorithm, which we will also denote by P for convenience. Similarly, we also construct a PSPACE algorithm Q that decides termination by searching for a loop that witnesses non-termination of $\mathcal{A}$. The overall PSPACE algorithm R first calls Q on $\mathcal{A}$ and checks whether $\mathcal{A}$ terminates. If $\mathcal{A}$ terminates, R iteratively calls P with increasing values for k on $\mathcal{A}$. R returns the value k such that P returns true for k and false for $k+1$. In the following we state the correctness of algorithm R:

Theorem 3. *Let $\mathcal{A}$ be an SCS. It is decidable in PSPACE, whether $\mathcal{A}$ does not terminate or has complexity $\Theta(N^k)$.*

Proof. If $\mathit{ranking}(\mathcal{A})$ returns with an exception, then there is a loop that witnesses non-termination. Thus, algorithm Q can find a loop that witnesses non-termination. Assume $\mathit{ranking}(\mathcal{A})$ terminates normally and returns $(\mathit{rank}, \mathit{witness}, \mathcal{C}, k)$. By Theorem 2 there is a sequence of paths $\mathit{Loop}_1, \ldots, \mathit{Loop}_k$ in $\mathcal{A}$ such that $\mathit{witness}$, $\mathcal{C}$ and $\mathit{Loop}_1, \ldots, \mathit{Loop}_k$ is a for-loop with size k. By Lemma 5 $\mathcal{A}$ has complexity $\Omega(N^k)$. By Corollary 1 $\mathcal{A}$ has complexity $O(N^k)$. Then, $\mathcal{A}$ cannot have a for-loop with size $k+1$ because such a for-loop would imply a trace of length $\Omega(N^{k+1})$ by Lemma 5. Thus, algorithm P can find a for-loop with size k but no for-loop of size $k+1$.

A Proof of Lemma 5

Proof. Let $l_1, \ldots, l_k$ be the length of the cyclic paths $\mathit{Loop}_1, \ldots, \mathit{Loop}_k$. We set $z = \max\{l_1, \ldots, l_k\}$ and $t = N/(2nz)$. We set $t = N/(2nz)$. Because we are interested in the asymptotic behavior w.r.t. N we can assume $N \geq 8nz$.

We define a path $\pi = ((\cdots(Loop_k)^t \cdots Loop_2)^t Loop_1)^t$. We note that π has length $\Omega(N^k)$. We define a set $I = [0,t]^k$ and consider the lexicographic order $> \subseteq I \times I$, where $(i_1,\ldots,i_k) > (j_1,\ldots,j_k)$, if there is a $1 \leq a \leq k$ such that $i_a > j_a$ and $i_b = j_b$ for all $1 \leq b < a$. We note that $>$ is a linear order on I. We define $I^1 = I \setminus \{(0,\ldots,0)\}$. We denote the predecessor of an element $e \in I^1$ w.r.t. to $>$ by $pred(e)$. We use I^1 to enumerate the cyclic paths in π, i.e., $\pi = \pi(t,\ldots,t,t)\pi(t,\ldots,t,t-1)\cdots\pi(t,\ldots,t,0)\pi(t,\ldots,t-1,t)\cdots\pi(0,\ldots,0,2)\pi(0,\ldots,0,1)$. We note that by the above definitions $\pi(i_1,\cdots,i_k) = Loop_d$, if and only if $i_d \neq 0$ and $i_u = 0$ for all $d < u \leq k$.

We define a function $bw_T : Val_N \to Val_N$ that takes an SCT T and a valuation $\sigma \in Val_N$ and returns a valuation σ' with $\sigma'(x) = \sigma(y)+1$, if $x > y' \in T$, $\sigma'(x) = \sigma(y)$, if $x \geq y' \in T$, and $\sigma'(x) = 0$, otherwise.

We will recursively define valuations $\sigma(e)$ for $e \in I$ and traces $\rho(e)$ for $e \in I^1$. We define $\sigma(0,\ldots,0)(x) = 0$ for all $x \in Var$. Let $e \in I^1$. Let $d \in [1,k]$ be chosen such that $\pi(e) = Loop_d$. Let $\ell \xrightarrow{T_{l_d}} \ell_{l_d-1} \xrightarrow{T_{l_d-1}} \cdots \ell_1 \xrightarrow{T_1} \ell$ be the path denoted by $Loop_d$. We define the trace $\rho(e)$ by $(\ell,\sigma_{l_d}) \xrightarrow{T_{l_d}} (\ell_{l_d-1},\sigma_{l_d-1}) \cdots (\ell_1,\sigma_1) \xrightarrow{T_1} (\ell,\sigma_0)$, where $\sigma_0 := \sigma(pred(e))$ and $\sigma_{i+1} := bw_{T_{i+1}}(\sigma_i)$ for all $0 < i \leq l_d$. We set $\sigma(e) := \sigma_{l_d}$ and $\sigma^i(e) := \sigma_i$ for all $0 < i < l_d$.

Let $x \in Var$ be some variable. We define $c(x) = u$, if $x \in C_u \setminus C_{u-1}$, setting $C_0 = \emptyset$, or $c(x) = \bot$, if there is no u with $x \in C_u$. Let $(i_1,\cdots,i_k) \in I$. We claim that

$$\begin{aligned}\sigma(i_1,\cdots,i_k)(x) &\leq i_u \cdot z + (u-1)\cdot N/n + N/(8n),\\ &\text{if there is a } u = c(x) \neq \bot, \text{ and}\\ \sigma(i_1,\cdots,i_k)(x) &\leq 3N/(4n) + (n-1)\cdot N/n, \text{ if } c(x) = \bot.\end{aligned} \tag{*}$$

We proceed by induction on $e = (i_1,\cdots,i_k)$. Clearly the claim holds for $e = (0,\ldots,0)$. Now consider $e \in I^1$. Let $d \in [1,k]$ be chosen such that $\pi(e) = Loop_d$. Let $(j_1,\ldots,j_k) = pred(i_1,\cdots,i_k)$. We have $i_u = j_u$ for all $1 \leq u < d$ and $j_d + 1 = i_d$. Let $x \in Var$ be some variable. Assume there is an $y \in Var$ with $x \triangleright y \in Loop_d$. Let $u = c(x)$ and $v = c(y)$. By the definition of a for-loop we have $suc_{Loop_d}(Var) = C_d$, and thus $\bot \neq v \leq d$. By induction assumption, we have $\sigma(j_1,\ldots,j_k)(y) \leq j_v \cdot z + (v-1)\cdot N/n$. If $1 \leq u \leq d$, we have $suc_{Loop_j}(C_u) = C_u$ and $suc_{Loop_j}(C_{u-1}) = C_{u-1}$ by the definition of a for-loop. Because $\mathcal{A}$ is deterministic and $\langle C_1,\ldots,C_k\rangle$ is a proper context, we get $u = v$. For $1 \leq u < d$ we have $\triangleright = \geq$ by the definition of a for-loop. We get $\sigma(i_1,\ldots,i_k)(x) = \sigma(j_1,\ldots,j_k)(y) \leq j_v \cdot z + (v-1)\cdot N/n + N/(8n) = i_u \cdot z + (u-1)\cdot N/n + N/(8n)$. If $u = d$ we have $\sigma(i_1,\ldots,i_k)(x) \leq \sigma(j_1,\ldots,j_k)(y) + z \leq j_v \cdot z + (v-1)\cdot N/n + N/(8n) + z = (j_d+1)\cdot z + (d-1)\cdot N/n + N/(8n) = i_u \cdot z + (u-1)\cdot N/n + N/(8n)$. Assume that $1 \leq u \leq d$ does not hold. If $u \neq \bot$ we have $d < u \leq k$, and thus $\sigma(i_1,\ldots,i_k)(x) \leq \sigma(j_1,\ldots,j_k)(y) + z \leq j_v \cdot z + (v-1)\cdot N/n + z + N/(8n) \leq N/(2n) + (v-1)\cdot N/n + z + N/(8n) \leq (u-1)\cdot N/n \leq i_u \cdot z + (u-1)\cdot N/n + N/(8n)$. If $u = \bot$, we have $\sigma(i_1,\ldots,i_k)(x) \leq \sigma(j_1,\ldots,j_k)(y) + z \leq j_v \cdot z + (v-1)\cdot N/n + z + N/(8n) \leq N/(2n) + (v-1)\cdot N/n + z + N/(8n) = (3N/4n) + (n-1)\cdot N/n$. Otherwise,

there is no y with $x \triangleright y \in \mathit{Loop}_j$. We have $\sigma(i_1, \ldots, i_k)(x) \leq z \leq N/(8n)$. We have established (*).

By (*) we have $\sigma(e)(x) \leq N - N/(4n) \leq N$ for all $e \in I$ and $x \in \mathit{Var}$. Moreover, we have that $\sigma^i(e)(x) \leq N - N/(4n) + i \leq N$ for all $e \in I^1$ and $0 < i < l_d$. Thus,

$$\rho(t, \ldots, t, t)\rho(t, \ldots, t, t-1) \cdots \rho(t, \ldots, t, 0) \\ \rho(t, \ldots, t-1, t) \cdots \rho(0, \ldots, 0, 2)\rho(0, \ldots, 0, 1)$$

is a trace over $[0, N]$ of length $\Omega(N^k)$.

B Proof of Lemma 11

Proof. We assume the exception has been raised in some recursive call $\mathit{main}(\mathcal{A}, i)$. We have that there is a loop Loop for some location $(\ell, \mathcal{C})$ of $\mathcal{A}$ such that (1) $\mathit{suc}_{\mathit{Loop}}(\mathit{Var}) = \mathit{last}(\mathcal{C})$ and (2) $x \triangleright y \in \mathit{Loop}$ and $x, y \in \mathit{last}(\mathcal{C})$ imply that $\triangleright = \geq$ because $\mathcal{A}$ is stable.

We define a function $bw_T : \mathit{Val}_N \to \mathit{Val}_N$ that takes an SCT T and a valuation $\sigma \in \mathit{Val}_N$ and returns a valuation σ' with $\sigma'(x) = \sigma(y) + 1$, if $x > y' \in T$, $\sigma'(x) = \sigma(y)$, if $x \geq y' \in T$, and $\sigma'(x) = 0$, otherwise.

Let $\ell_l \xrightarrow{T_l} \ell_{l-1} \xrightarrow{T_{l-1}} \cdots \ell_1 \xrightarrow{T_1} \ell_0$ with $\ell = \ell_l = \ell_0$ be the path denoted by Loop. We define a valuation $\sigma_0(x) = 0$ for all $x \in \mathit{Var}$. We define a trace ρ_0 by $(\ell, \sigma_l) \xrightarrow{T_l} (\ell_{l-1}, \sigma_{l-1}) \cdots (\ell_1, \sigma_1) \xrightarrow{T_1} (\ell, \sigma_0)$, where $\sigma_{i+1} := bw_{T_{i+1}}(\sigma_i)$ for all $0 < i \leq l$. Moreover, we define a trace ρ by $(\ell, \sigma_{2l}) \xrightarrow{T_l} (\ell_{2l-1}, \sigma_{2l-1}) \cdots (\ell_{l+1}, \sigma_{l+1}) \xrightarrow{T_1} (\ell, \sigma_l)$, where $\sigma_{i+1} := bw_{T_{i+1}}(\sigma_i)$ for all $l < i \leq 2l$. By induction we get $\sigma_i \in [0, i]$ for all $0 \leq i \leq 2l$.

We will show $\sigma_{2l} = \sigma_l$. This is sufficient to show that $\rho^\omega = \rho\rho \cdots$ is an infinite trace of $\mathcal{A}$ with valuations over $[0, 2l]$.

We denote by $\mathit{Loop}|_i = \ell \xrightarrow{T_l} \ell_{l-1} \xrightarrow{T_{l-1}} \cdots \ell_{i+1} \xrightarrow{T_{i+1}} \ell_i$ the prefix of Loop until position i. We claim that $\sigma_{l+i}(x) = \sigma_i(x)$ for all $x \in \mathit{suc}_{\mathit{Loop}|_i}(\mathit{Var})$. The proof proceeds by induction on i. Base case $i = 0$: From (1) and (2) we get that $\sigma_l(x) = \sigma_0(x) = 0$ for all $x \in \mathit{suc}_{\mathit{Loop}}(\mathit{Var}) = \mathit{last}(\mathcal{C})$. Induction step: We consider some $x \in \mathit{suc}_{\mathit{Loop}|_i}(\mathit{Var})$. Assume x does not have a successor in T_i. Then $\sigma_{l+i}(x) = \sigma_i(x) = 0$. Assume x does have a successor in T_i, i.e., $x \triangleright y \in T$ for some $y \in \mathit{Var}$. Then we have $y \in \mathit{suc}_{\mathit{Loop}|_{i-1}}(\mathit{Var})$ and thus $\sigma_{l+(i-1)}(y) = \sigma_{i-1}(y)$ by induction assumption. By the definition of bw_T we get $\sigma_{l+i}(x) = bw_T(\sigma_{l+(i-1)})(x) = bw_T(\sigma_{i-1})(x) = \sigma_i(x)$.

References

1. Anderson, H., Khoo, S.-C.: Affine-based size-change termination. In: Ohori, A. (ed.) APLAS 2003. LNCS, vol. 2895, pp. 122–140. Springer, Heidelberg (2003)
2. Ben-Amram, A.M.: A complexity tradeoff in ranking-function termination proofs. Acta Inf. **46**(1), 57–72 (2009)

3. Ben-Amram, A.M.: Monotonicity constraints for termination in the integer domain. Logical Methods Comput. Sci. **7**(3), 1–43 (2011)
4. Ben-Amram, A.M., Lee, C.S.: Ranking functions for size-change termination ii. Logical Methods Comput. Sci. **5**(2), 1–29 (2009)
5. Codish, M., Fuhs, C., Giesl, J., Schneider-Kamp, P.: Lazy abstraction for size-change termination. In: Fermüller, C.G., Voronkov, A. (eds.) LPAR-17. LNCS, vol. 6397, pp. 217–232. Springer, Heidelberg (2010)
6. Colcombet, T., Daviaud, L., Zuleger, F.: Size-change abstraction and max-plus automata. In: Csuhaj-Varjú, E., Dietzfelbinger, M., Ésik, Z. (eds.) MFCS 2014, Part I. LNCS, vol. 8634, pp. 208–219. Springer, Heidelberg (2014)
7. Krauss, A.: Certified size-change termination. In: Pfenning, F. (ed.) CADE 2007. LNCS (LNAI), vol. 4603, pp. 460–475. Springer, Heidelberg (2007)
8. Lee, C.S., Jones, N.D., Ben-Amram, A.M.: The size-change principle for program termination. In: POPL, pp. 81–92 (2001)
9. Manolios, P., Vroon, D.: Termination analysis with calling context graphs. In: Ball, T., Jones, R.B. (eds.) CAV 2006. LNCS, vol. 4144, pp. 401–414. Springer, Heidelberg (2006)
10. Vidal, G.: Quasi-terminating logic programs for ensuring the termination of partial evaluation. In: PEPM, pp. 51–60 (2007)
11. Zuleger, F., Gulwani, S., Sinn, M., Veith, H.: Bound analysis of imperative programs with the size-change abstraction. In: Yahav, E. (ed.) Static Analysis. LNCS, vol. 6887, pp. 280–297. Springer, Heidelberg (2011)