Modular Automatic Complexity Analysis of Recursive Integer Programs

Nils Lommen(✉) and Jürgen Giesl

RWTH Aachen University, Aachen, Germany
{lommen,giesl}@cs.rwth-aachen.de

Abstract. In previous work, we developed a modular approach for automatic complexity analysis of integer programs. However, these integer program do not allow non-tail *recursive* calls or subprocedures. In this work, we consider integer programs with function calls and present a natural extension of our modular complexity analysis approach to the recursive setting based on a new form of ranking functions. Hence, our approach combines already existing powerful techniques on the "imperative" parts of the program and our novel ranking functions on the recursive parts. The strength of this combination is demonstrated by our implementation in the complexity analysis tool KoAT.

1 Introduction

There exist numerous approaches to analyze complexity of programs automatically, e.g., [1, 2, 4, 6–8, 10, 13, 17, 19, 23, 24], but most of them are essentially limited to non-recursive programs. There are also several techniques for complexity analysis of term rewrite systems (TRSs) which can handle arbitrary recursion, e.g., [3, 22]. However, TRSs have the drawback that they do not support built-in data types like integers. Thus, the goal of this paper is to analyze complexity of programs with built-in integers *and* arbitrary (possibly non-tail) recursion.

In previous work, we developed a *modular* technique for complexity analysis of programs with built-in integers which we implemented in the complexity analysis tool KoAT. It automatically infers runtime bounds for integer transition systems (ITSs) possibly consisting of multiple loops by handling some subprograms as so-called twn-loops (where there exist "complete" techniques for analyzing termination and complexity [6, 11, 12, 15, 16, 18]) and by using multiphase-linear ranking functions [4, 5, 10] for other subprograms. By inferring bounds for one subprogram after the other, in the end we obtain a bound on the runtime of the whole program. In this paper, we extend our approach to ITSs which allow *function calls*, including non-tail recursion. In contrast to the first attempt for such an extension from [6, Sect. 5], our novel approach takes the results of function calls into account which leads to a much higher precision.

Example 1. The first **while**-loop of the procedure **main** in Fig. 1 computes $x! + \cdots + 1!$ by calling the subprocedure **fac**. We introduce a novel class of ranking functions for recursive programs to show that this loop has quadratic runtime.

```
main(x, y):
while x > 0 do
  y ← y + fac(x); x ← x − 1;
x ← 1;
while x < y do
  x ← 3 · x; y ← 2 · y;

fac(a):
if a = 0 then
  return 1;
else if a > 0 then
  return a · fac(a − 1);
```

Fig. 1: Recursive Integer Program with two Procedures

Furthermore, y's value is bounded by $y + x^{x^2+1}$, where x and y refer to the initial values of the program variables. This observation is crucial for the runtime of the second loop since it is executed at most $\log_2(\text{size}(y)) + 2 = \log_2(y + x^{x^2+1}) + 2$ times, where $\text{size}(y)$ denotes the value of y before the second loop. Hence, the overall program has less than cubic runtime. Here, [6, Sect. 5] fails to infer a finite runtime bound, as it disregards the return value of **fac**. The runtime bound for the second loop can be obtained by our technique based on twn-loops, but not by linear ranking functions. Thus, our novel approach for recursive integer programs allows us to combine various techniques for automated complexity analysis and to benefit from their individual strengths.

In this work, we extend our notions of runtime and size bounds [6, 10, 15, 16, 18] to the new setting of ITSs with function calls. On the one hand, as illustrated by Ex. 1, we need size bounds to compute runtime bounds, and on the other hand we also need runtime bounds to infer size bounds. Thus, our approach alternates between the computation of runtime and size bounds.

Structure: In Sect. 2 we introduce our new notion of ITSs with function calls and define runtime and size bounds for these programs. In Sect. 3, we show how to compute modular runtime bounds for our new class of programs. Analogously, we present a technique to infer size bounds in a modular way in Sect. 4. In Sect. 5, we conclude and discuss our implementation in the tool KoAT. All proofs can be found in App. A (and they will also be published on arXiv).

2 Recursive Integer Transition Systems

In Sect. 2.1 we extend ITSs by function calls and recursion. Afterwards, in Sect. 2.2 we define runtime and size bounds which extend the corresponding notions for ITS without function calls [6, 10, 15, 16, 18] in a natural way.

2.1 Syntax and Semantics of Recursive Integer Transition Systems

For a finite set of variables $\mathcal{V}$, as usual, $\mathbb{Z}[\mathcal{V}]$ is the polynomial ring over the variables $\mathcal{V}$ with integer coefficients. *Constraints* are used in the guards of transitions.

Definition 2 (Atoms and Constraints). *The set of* atoms $\mathcal{A}(\mathcal{V})$ *consists of all inequations* $p_1 < p_2$ *for polynomials* $p_1, p_2 \in \mathbb{Z}[\mathcal{V}]$. *The set* $\mathcal{C}(\mathcal{V})$ *of* constraints *consists of all formulas built from atoms* $\mathcal{A}(\mathcal{V})$ *and* $\wedge$.

We also use "$\geq$", "$=$", "$\neq$" in atoms, and negations "$\neg$" which can be simulated by formulas (e.g., $p_1 \geq p_2$ is equivalent to $p_2 < p_1 + 1$ for integers). Disjunctions "$\vee$" are modeled by several transitions with the same start and target location.

Non-recursive ITSs are a widely studied formalism in automatic program verification. Since ITSs do not allow any non-tail recursion, Def. 3 extends them to ITSs with *function calls* (so-called ρ-ITSs). In an ITS, the value of a program variable $v \in \mathcal{PV}$ is changed according to the updates $\eta : \mathcal{PV} \to \mathbb{Z}[\mathcal{V}]$ of its transitions. More precisely, we move from a configuration $(\ell, \sigma) \in \mathcal{L} \times \Sigma$ to (ℓ', σ') by evaluating a transition, where $\sigma, \sigma' : \mathcal{V} \to \mathbb{Z}$ are *states*, Σ denotes the set of all states, and $\mathcal{L}$ are the *locations* of the program. We now introduce a set $\mathcal{F}$ of *function calls* $\ell(v|\zeta)$. Here, ℓ is the start location of the subprogram that is called, the update $\zeta : \mathcal{PV} \to \mathbb{Z}[\mathcal{V}]$ sets the program variables of the subprogram to their initial values, and $v \in \mathcal{PV}$ contains the "return value" of the subprogram upon its termination. More precisely, if $\sigma \in \Sigma$ is the state before calling the subprogram via $\ell(v|\zeta)$, then the subprogram starts in a configuration $(\ell, \tilde{\sigma})$, where $\tilde{\sigma}$ results from "applying" the update ζ to the state σ. We also introduce a subset of *return locations* $\Omega \subseteq \mathcal{L}$. As soon as the called subprogram reaches a configuration (ℓ', σ') with $\ell' \in \Omega$, the value $\sigma'(v)$ is returned as the result of the function call $\ell(v|\zeta)$. (We will define the semantics of ρ-ITSs formally in Def. 5.) Thus, transitions may now have updates which map program variables to polynomial combinations of variables *and* function calls (denoted by $\mathbb{Z}[\mathcal{V} \cup \mathcal{F}]$).

Definition 3 (ρ-ITS). *The tuple $(\mathcal{PV}, \mathcal{L}, \ell_0, \Omega, \mathcal{F}, \mathcal{T})$ is an* ITS with function calls *(ρ-ITS) where*

- *$\mathcal{PV}$ is a finite set of* program variables*, $\mathcal{V} \setminus \mathcal{PV}$ are* temporary variables*,*
- *$\mathcal{L}$ is a finite set of* locations *with an initial location ℓ_0,*
- *$\mathcal{F}$ is a finite set of* function calls *$\ell(v|\zeta)$ with $\ell \in \mathcal{L} \setminus \{\ell_0\}$, $v \in \mathcal{PV}$, and $\zeta : \mathcal{PV} \to \mathbb{Z}[\mathcal{V}]$,*
- *$\Omega \subseteq \mathcal{L} \setminus \{\ell_0\}$ is a finite set of* return locations*, and*
- *$\mathcal{T}$ is a finite set of* transitions*: A transition is a 4-tuple $(\ell, \varphi, \eta, \ell')$ with start location $\ell \in \mathcal{L} \setminus \Omega$, target location $\ell' \in \mathcal{L} \setminus \{\ell_0\}$, guard $\varphi \in \mathcal{C}(\mathcal{V})$, and update function $\eta : \mathcal{PV} \to \mathbb{Z}[\mathcal{V} \cup \mathcal{F}]$.*

We often denote the set of function calls in a polynomial p, an update η, or a transition t by $\mathrm{fun}(p)$, $\mathrm{fun}(\eta)$, or $\mathrm{fun}(t)$, respectively. Similarly, for $\rho \in \mathcal{F}$, $\mathrm{fun}^{-1}(\rho)$ is the set of all transitions of the ITS in which ρ occurs in an update. Transitions $(\ell_0, \_, \_, \_)$ are called *initial* and $\mathcal{T}_0$ denotes the set of all initial transitions.

A ρ-ITSs may contain two kinds of non-determinism: First, non-deterministic branching is realized by multiple transitions with the same start location. Second, non-deterministic sampling is modeled by temporary variables (which can be restricted in the guard of a transition). Temporary variables are updated arbitrarily in each evaluation step (and also in function calls), and are only restricted by the transition's guard. Intuitively, these variables are set by an adversary trying to "sabotage" the program in order to obtain long runtimes.

Example 4. The ρ-ITS in Fig. 2 corresponds to the program from Fig. 1. In Fig. 2, we omitted trivial guards $\varphi = \texttt{true}$ and identity updates of the form $\eta(v) = v$. The ρ-ITS has the program variables $\mathcal{PV} = \{a, x, y\}$, five locations

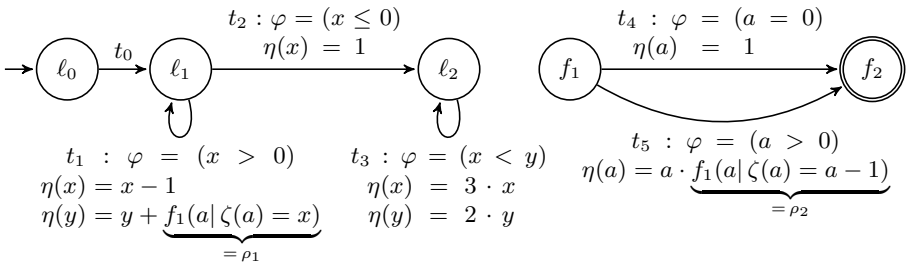

Fig. 2: An Integer Transition System with Function Calls ρ_1 and ρ_2

$\mathcal{L} = \{\ell_0, \ell_1, \ell_2, f_1, f_2\}$, and two function calls $\rho_1 = f_1(a|\,\zeta(a) = x)$ and $\rho_2 = f_1(a|\,\zeta(a) = a - 1)$. The subprogram with the locations f_1 and f_2 computes the factorial $a!$ recursively and returns this result in the return location f_2 (indicated by the doubled node). This subprogram is called iteratively in the loop t_1 with the argument x. The factorials $x!, (x-1)!, \ldots, 1$ are summed up in the variable y. Afterwards, x is set to 1 in t_2, and the second loop t_3 at location ℓ_2 is executed.

In the following, we also allow the application of states $\sigma \in \Sigma$ to arithmetic expressions e and constraints c, i.e., the number $[\![e]\!]_\sigma$ or the Boolean value $[\![c]\!]_\sigma$ results from e or c resp. by replacing each variable v by $\sigma(v)$.

From now on, we fix a ρ-ITS $(\mathcal{PV}, \mathcal{L}, \ell_0, \Omega, \mathcal{F}, \mathcal{T})$ over the variables $\mathcal{V}$. Formally, an evaluation step of a ρ-ITS is a transformation of a tree $\mathbb{T}$ whose nodes are labeled with configurations from $\mathcal{L} \times (\Sigma \cup \{\bot\})$. We distinguish two kinds of evaluation steps: If a leaf of $\mathbb{T}$ is labeled with a configuration (ℓ, σ) where a transition $t = (\ell, \varphi, \eta, \ell')$ can be applied, then a *t-evaluation step* extends $\mathbb{T}$ at the position of this leaf to a new tree $\mathbb{T}'$, denoted $\mathbb{T} \prec_t \mathbb{T}'$. If the update η does not contain any function calls, then $\mathbb{T}'$ results from $\mathbb{T}$ by adding an edge to a new node labeled with a configuration (ℓ', σ') where $\sigma'(v) = [\![\eta(v)]\!]_\sigma$ for all program variables v, i.e., by $(\ell, \sigma) \to_t (\ell', \sigma')$. However, if t contains function calls $\rho_i = \ell_i(v_i|\zeta_i)$ for $1 \leq i \leq n$, then $\mathbb{T}'$ results from $\mathbb{T}$ by adding $n + 1$ children to the former leaf labeled with (ℓ, σ), i.e., $(\ell, \sigma) \to_t (\ell', \bot)$ and $(\ell, \sigma) \to_{\rho_i} (\ell_i, \sigma_i)$ for all $1 \leq i \leq n$, where $\sigma_i(v) = [\![\zeta_i(v)]\!]_\sigma$ for all program variables v. Here, $\bot$ denotes an undefined state which will be instantiated later if the function calls reach return locations.

To this end, we use so-called *ε-evaluation steps*. If for all $1 \leq i \leq n$, there are paths from the nodes (ℓ_i, σ_i) to configurations (ℓ_i', σ_i') where $\ell_i' \in \Omega$ is a return location, and these paths only contain edges marked with transitions (and not with function calls ρ), then the undefined state $\bot$ can be replaced by a state σ' such that $\sigma'(v) = [\![\eta'(v)]\!]_\sigma$ for all program variables v. Here, $\eta'(v)$ results from $\eta(v)$ by replacing every function call $\ell_i(v_i|\zeta_i)$ by the returned value $\sigma_i'(v_i)$ for all $1 \leq i \leq n$. We denote this by $\eta'(v) = \eta(v)\,[\ell_i(v_i|\zeta_i)/\sigma_i'(v_i)]$.

Definition 5 (Evaluation of ρ-ITSs). *Let $\mathbb{T}$ be a tree whose nodes are labeled with configurations from $\mathcal{L} \times (\Sigma \cup \{\bot\})$. $\mathbb{T} \prec_t \mathbb{T}'$ is a* t-evaluation step *with transition $t = (\ell, \varphi, \eta, \ell')$ iff $\mathbb{T}$ has a leaf labeled with (ℓ, σ) where $\sigma \in \Sigma$, $\sigma \models \varphi$, and*

- *if $\mathrm{fun}(\eta) = \varnothing$, then $\mathbb{T}'$ is the extension of $\mathbb{T}$ by an edge $(\ell, \sigma) \to_t (\ell', \sigma')$ to a new node labeled with (ℓ', σ') where $\sigma'(v) = [\![\eta(v)]\!]_\sigma$ for all $v \in \mathcal{PV}$.*

- *if η contains the function calls $\rho_1 = \ell_1(v_1|\zeta_1), \ldots, \rho_n = \ell_n(v_n|\zeta_n)$, then $\mathbb{T}'$ is the extension of $\mathbb{T}$ by the edges $(\ell,\sigma) \to_t (\ell', \bot)$ and $(\ell,\sigma) \to_{\rho_i} (\ell_i, \sigma_i)$ to $n+1$ new nodes, where $\sigma_i(v) = [\![\zeta_i(v)]\!]_\sigma$ for all $v \in \mathcal{PV}$ and all $1 \leq i \leq n$.*

If there is a transition $t = (\ell, \varphi, \eta, \ell')$, $\mathbb{T}$ contains a node N labeled with (ℓ,σ) with n+1 children such that $(\ell,\sigma) \to_t (\ell', \bot)$ and $(\ell,\sigma) \to_{\rho_i} (\ell_i, \sigma_i)$ for all function calls $\rho_i = \ell_i(v_i|\zeta_i) \in \mathrm{fun}(\eta)$, and from each child labeled with (ℓ_i, σ_i) there is a path to a node labeled with $(\_, \sigma_i') \in \Omega \times \Sigma$ whose edges are all marked with transitions, then $\mathbb{T} \prec_\varepsilon \mathbb{T}'$ is an ε-evaluation step *iff $\mathbb{T}'$ results from $\mathbb{T}$ by replacing N's label $(\ell', \bot)$ by (ℓ', σ'), where $\sigma'(v) = [\![\eta(v)\,[\ell_i(v_i|\zeta_i)/\sigma_i'(v_i)]]\!]_\sigma$ for all $v \in \mathcal{PV}$.*

For an initial state $\sigma_0 \in \Sigma$, the evaluation always starts with $\mathbb{T}_{\sigma_0} = (\{(\ell_0, \sigma_0)\}, \varnothing)$ which has the only node (ℓ_0, σ_0). We write $\prec_\mathcal{T}$ for $\bigcup_{t \in \mathcal{T}} \prec_t$ and $\prec$ for $\prec_{\mathcal{T} \cup \{\varepsilon\}}$. Moreover, we denote finitely many evaluations steps $\mathbb{T} \prec \cdots \prec \mathbb{T}'$ by $\mathbb{T} \prec^* \mathbb{T}'$.

Example 6. Reconsider the ρ-ITS from Fig. 2 and let us denote states $\sigma \in \Sigma$ as tuples $(\sigma(a), \sigma(x), \sigma(y)) \in \mathbb{Z}^3$. The following tree shows an evaluation starting in $\mathbb{T}_{(0,2,0)}$. Here, a dashed arrow indicates that a state, which was reached via a function call, was used to replace $\bot$ via an ε-evaluation step. So for example, the value $\sigma_2(a) = [\![\eta'(a)]\!]_{\sigma_1} = 2$ in c_2 is obtained from $\eta(a) = a \cdot f_1(a|\,\zeta(a) = a - 1)$ by replacing the function call $f_1(a|\,\zeta(a) = a-1)$ by the returned value $\sigma_3(a) = 1$ of the function call and by the instantiation $\sigma_1(a) = 2$.

$$
\begin{array}{ccccc}
(\ell_0,(0,2,0)) & \xrightarrow{t_0} & (\ell_1,(0,2,0)) & \xrightarrow{t_1} & (\ell_1,\bot) \\
 & & \downarrow \rho_1 & & \\
 & & c_1 = (f_1,(2,2,0)) & \xrightarrow{t_5} & (f_2,(2,2,0)) = c_2 \\
 & & \downarrow \rho_2 & & \uparrow \varepsilon \\
 & & (f_1,(1,2,0)) & \xrightarrow{t_5} & (f_2,(1,2,0)) = c_3 \\
 & & \downarrow \rho_2 & & \uparrow \varepsilon \\
 & & (f_1,(0,2,0)) & \xrightarrow{t_4} & (f_2,(1,2,0)) = c_4
\end{array}
$$

In the next evaluation step, $(\ell_1, \bot)$ can be instantiated by considering c_2. Then, $\bot$ would be replaced by $(2,1,2)$. Note that while in this tree, every node has at most one child connected by a ρ-edge, in general a node can have several outgoing ρ-edges if there exist transitions whose updates contain several function calls.

The goal of complexity analysis is to derive an upper bound on the number of t-evaluation steps starting in $\mathbb{T}_{\sigma_0}$. For any tree $\mathbb{T}$ and set of transitions $\mathcal{T}$, $|\mathbb{T}|_\mathcal{T}$ is the number of edges which are marked by a transition from $\mathcal{T}$. The *runtime complexity* measures how many transitions are evaluated in the worst case.

Definition 7 (Runtime Complexity). *The* runtime complexity *is* $\mathrm{rc} : \Sigma \to \overline{\mathbb{N}}$ *with* $\overline{\mathbb{N}} = \mathbb{N} \cup \{\omega\}$ *and* $\mathrm{rc}(\sigma_0) = \sup\{|\mathbb{T}|_\mathcal{T} \mid \mathbb{T}_{\sigma_0} \prec^* \mathbb{T}\}$.

2.2 Runtime and Size Bounds for ρ-ITSs

Now we define our notion of *bounds*. We only consider bounds which are weakly monotonically increasing in all variables, since they can be composed easily (i.e., if f and g increase monotonically, then so does their composition $f \circ g$). As in [18],

bounds can also be *logarithmic*. In contrast to our earlier papers, we also consider exponential bounds with non-constant bases to represent bounds like x^{x^2+1}.

Definition 8 (Bounds). *The set of* bounds $\mathcal{B}$ *is the smallest set with* $\overline{\mathbb{N}} \subseteq \mathcal{B}$, $\mathcal{PV} \subseteq \mathcal{B}$, *and* $\{b_1 + b_2, \max(b_1, b_2), b_1 \cdot b_2, p^{b_1}, \log_k(b_1)\} \subseteq \mathcal{B}$ *for all* $b_1, b_2 \in \mathcal{B}$, *all polynomials* $p \in \mathbb{N}[\mathcal{PV}]$, *and all* $k \in \mathbb{R}_{>1}$.[1]

Note that in $\mathcal{B}$ we require bounds to only contain program variables since the values of temporary variables are "set by the adversary".

A *runtime bound* $\mathcal{RB}(t)$ over-approximates the number of t-evaluations that can occur in an arbitrary evaluation starting in a state $\sigma_0 \in \Sigma$, i.e., it is a bound on the number of t-edges in any evaluation tree resulting from $\mathbb{T}_{\sigma_0}$. In the following, let $|\sigma|$ denote the state with $|\sigma|(v) = |\sigma(v)|$ for all $v \in \mathcal{V}$.

Definition 9 (Runtime Bound). $\mathcal{RB} : \mathcal{T} \to \mathcal{B}$ *is a* runtime bound *if for all* $\sigma_0 \in \Sigma$, *all* $t \in \mathcal{T}$, *and all trees* $\mathbb{T}$ *with* $\mathbb{T}_{\sigma_0} \prec^* \mathbb{T}$, *we have* $|\mathbb{T}|_{\{t\}} \leq [\![\mathcal{RB}(t)]\!]_{|\sigma_0|}$.

Cor. 10 shows that to obtain an upper bound on the runtime complexity, one can compute runtime bounds for each transition separately and add them.

Corollary 10 (Over-Approximating rc**).** *Let* $\mathcal{RB}$ *be a runtime bound. Then for all states* $\sigma_0 \in \Sigma$, *we have* $\mathrm{rc}(\sigma_0) \leq [\![\sum_{t \in \mathcal{T}} \mathcal{RB}(t)]\!]_{|\sigma_0|}$.

Example 11. In Fig. 2, the transitions t_0 and t_2 executed at most once, i.e., $\mathcal{RB}(t_0) = \mathcal{RB}(t_2) = 1$. In Ex. 22, we will infer a runtime bound with $\mathcal{RB}(t_1) = \mathcal{RB}(t_4) = x$, $\mathcal{RB}(t_3) = \log_2(y + x^{x^2+1}) + 2$, and $\mathcal{RB}(t_5) = x^2$. This results in a less than cubic bound on the runtime complexity of the ρ-ITS.

Our approach performs a *modular* analysis, i.e., parts of the program are analyzed as standalone programs and the results are then lifted to contribute to the overall analysis. So to compute a runtime bound for a transition t, our approach considers all transitions and function calls $\tau \in \mathcal{T} \cup \mathcal{F}$ that can occur directly before t in evaluations, and it needs *size bounds* $\mathcal{SB}(\tau, v)$ to over-approximate the absolute values that the variables $v \in \mathcal{PV}$ may have *after* these "previous" transitions and function calls τ. We call $\mathcal{RV} = (\mathcal{T} \cup \mathcal{F}) \times \mathcal{PV}$ the set of *result variables*. Note that in contrast to runtime bounds (and to our earlier papers), we now also have to capture the effect of function calls $\mathcal{F}$ via size bounds.

Definition 12 (Size Bound). *A function* $\mathcal{SB} : \mathcal{RV} \to \mathcal{B}$ *is called a* size bound *if for all* $(\tau, v) \in \mathcal{RV}$, *all states* $\sigma_0 \in \Sigma$, *and all trees* $\mathbb{T}$ *with* $\mathbb{T}_{\sigma_0} \prec^* \mathbb{T}$ *containing a path* $(\ell_0, \sigma_0) \to \cdots \to_\tau (\_, \sigma)$ *with* $\sigma \neq \bot$, *we have* $|\sigma|(v) \leq [\![\mathcal{SB}(\tau, v)]\!]_{|\sigma_0|}$.

Example 13. In Fig. 2, $\mathcal{SB}(t_0, x) = x$ is a size bound, since the value of x after evaluating t_0 is bounded by the initial value of x. (Ex. 27 will show how to compute such bounds.) Similarly, we have $\mathcal{SB}(t_2, x) = 1$ and $\mathcal{SB}(t_2, y) = y + x^{x^2+1}$, see Ex. 31. The size bound $\mathcal{SB}(\rho_1, a) = x$ (see Ex. 27) expresses that the value of a after executing the function call ρ_1 is bounded by the initial value of x.

[1] More precisely, instead of $\log_k(b_1)$ we use the function $\lceil \log_k(\max\{1, b_1\}) \rceil$ to ensure that bounds are well defined, weakly monotonically increasing, and evaluate to $\mathbb{N}$.

3 Modular Computation of Runtime Bounds

Now we introduce our modular approach for the computation of runtime bounds. To be precise, we infer runtime bounds for subprograms $\mathcal{T}'$ and then lift them to runtime bounds for the full program. For any non-empty $\mathcal{T}' \subseteq \mathcal{T} \setminus \mathcal{T}_0$, let $\mathcal{L}_{\mathcal{T}'} = \{\ell \in \mathcal{L} \mid (\ell, \_, \_, \_) \in \mathcal{T}'\}$ contain all start locations of transitions from $\mathcal{T}'$.

In contrast to global bounds, a *local runtime bound* $\mathcal{RB}_{\text{loc}}^{\mathcal{T}'_>,\mathcal{T}'}\colon \mathcal{L}_{\mathcal{T}'} \to \mathcal{B}$ only takes the subprogram $\mathcal{T}'$ into account. It considers a subset $\mathcal{T}'_> \subseteq \mathcal{T}'$ and for every $t \in \mathcal{T}'_>$, $\mathcal{RB}_{\text{loc}}^{\mathcal{T}'_>,\mathcal{T}'}(\ell_{in})$ over-approximates the number of applications of t in any run of $\mathcal{T}'$ starting in ℓ_{in}. However, local runtime bounds do not consider how often such a run is started or how large the variables are before starting a run.

Definition 14 (Local Runtime Bound). *Let $\varnothing \neq \mathcal{T}'_> \subseteq \mathcal{T}' \subseteq \mathcal{T} \setminus \mathcal{T}_0$. $\mathcal{RB}_{\text{loc}}^{\mathcal{T}'_>,\mathcal{T}'}\colon \mathcal{L}_{\mathcal{T}'} \to \mathcal{B}$ is a* local runtime bound *for $\mathcal{T}'_>$ w.r.t. $\mathcal{T}'$ if for all $\sigma_0 \in \Sigma$, all $\ell_{in} \in \mathcal{L}_{\mathcal{T}'}$, and all trees $\mathbb{T}$ with $(\{(\ell_{in},\sigma_0)\},\varnothing) \prec^*_{\mathcal{T}' \cup \{\varepsilon\}} \mathbb{T}$, we have $|\mathbb{T}|_{\mathcal{T}'_>} \leq [\![\mathcal{RB}_{\text{loc}}^{\mathcal{T}'_>,\mathcal{T}'}(\ell_{in})]\!]_{|\sigma_0|}$.*

For readability, Def. 14 considers arbitrary initial states σ_0, but it could also be refined to only consider states σ_0 where (ℓ_{in}, σ_0) is reachable in the full program.

For $\mathcal{T}' \subseteq \mathcal{T}$, let $\text{fun}(\mathcal{T}')$ denote the set of all function calls $\ell(\_|\_) \in \text{fun}(t)$ for transitions $t \in \mathcal{T}'$ such that $\ell \in \mathcal{L}_{\mathcal{T}'}$. Moreover, $\text{fun}^{-1}(\mathcal{T}')$ denotes the set of all transitions $t \in \mathcal{T}'$ with function calls $\ell(\_|\_) \in \text{fun}(t)$ such that $\ell \in \mathcal{L}_{\mathcal{T}'}$.

The following "*function call ranking functions*" (ρ-RFs) yield a local runtime bound for the set of transitions $\mathcal{T}'_>$ w.r.t. $\mathcal{T}'$, provided that $\text{fun}^{-1}(\mathcal{T}') \subseteq \mathcal{T}'_>$. A ρ-RF $\langle r_{\text{tf}}, r_{\text{t}}, r_{\text{f}}\rangle$ combines three ranking functions $r_{\text{tf}}, r_{\text{t}}, r_{\text{f}} : \mathcal{L} \to \mathbb{Z}[\mathcal{PV}]$. For any $\mathcal{T}'$-evaluation tree, $r_{\text{tf}}(\ell)$ is a bound on the number of edges labeled with transitions from $\text{fun}^{-1}(\mathcal{T}')$ (i.e., **t**ransitions with **f**unction calls) in any path of the tree that starts in a configuration of the form $(\ell, \_)$. So these paths may contain both steps with transitions and steps with function calls. Similarly, $r_{\text{t}}(\ell)$ is a bound on the number of edges with **t**ransitions from $\mathcal{T}'_> \setminus \text{fun}^{-1}(\mathcal{T}')$ in any path of a $\mathcal{T}'$-evaluation tree starting with $(\ell, \_)$. Finally, $r_{\text{f}}(\ell)$ is a bound on the number of ρ-edges (i.e., steps with **f**unction calls from $\text{fun}(\mathcal{T}')$) in any path of a $\mathcal{T}'$-evaluation tree starting with $(\ell, \_)$. In Ex. 6, these function calls correspond to vertical edges, whereas steps with transitions correspond to horizontal edges.

To ensure these properties in Def. 15, (a) requires that r_{tf} is decreasing and bounded for edges labeled with $\text{fun}^{-1}(\mathcal{T}')$, (b) requires this for r_{t} and edges labeled with $\mathcal{T}'_> \setminus \text{fun}^{-1}(\mathcal{T}')$, and (c) requires this for r_{f} and edges labeled with $\text{fun}(\mathcal{T}')$. In (d), we require that r_{tf}, r_{t}, and r_{f} do not increase for any edge.

In the following definition, we extend the evaluation of arithmetic expression e to the "undefined" state $\bot$ by defining $[\![e]\!]_\bot = 0$. We also use the relations $\to_t$ and $\to_\rho$ without referring to an actual evaluation tree. Thus, we say that $(\ell, \sigma) \to_\tau (\ell', \sigma')$ holds for some $\tau \in \mathcal{T} \cup \mathcal{F}$ if there exists an evaluation $\mathbb{T}_{\sigma_0} \prec^*_{\mathcal{T} \cup \{\varepsilon\}} \mathbb{T}$ for some state $\sigma_0 \in \Sigma$ such that $\mathbb{T}$ contains an edge $(\ell, \sigma) \to_\tau (\ell', \sigma')$.

Definition 15 (Function Call Ranking Function). *Let $\varnothing \neq \mathcal{T}'_> \subseteq \mathcal{T}' \subseteq \mathcal{T} \setminus \mathcal{T}_0$ with $\text{fun}^{-1}(\mathcal{T}') \subseteq \mathcal{T}'_>$. Then $\langle r_{\text{tf}}, r_{\text{t}}, r_{\text{f}}\rangle$ with $r_{\text{tf}}, r_{\text{t}}, r_{\text{f}} : \mathcal{L} \to \mathbb{Z}[\mathcal{PV}]$ is a* function call ranking function *(ρ-RF) for $\mathcal{T}'_>$ w.r.t. $\mathcal{T}'$ if for all evaluation steps $(\ell, \sigma) \to_\tau (\ell', \sigma')$ with $\tau \in \mathcal{T}' \cup \text{fun}(\mathcal{T}')$, we have:*

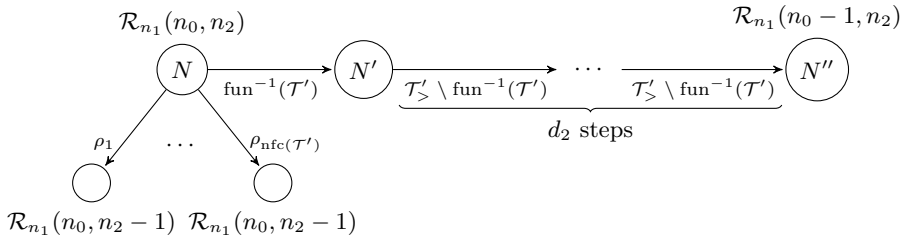

Fig. 3: Illustration of $\mathcal{R}_{n_1}(n_0, n_2)$

(a) *if* $\tau \in \mathrm{fun}^{-1}(\mathcal{T}')$, *then* $[\![r_{\mathrm{tf}}(\ell)]\!]_\sigma > [\![r_{\mathrm{tf}}(\ell')]\!]_{\sigma'}$ *and* $[\![r_{\mathrm{tf}}(\ell)]\!]_\sigma > 0$
(b) *if* $\tau \in \mathcal{T}'_> \setminus \mathrm{fun}^{-1}(\mathcal{T}')$, *then* $[\![r_{\mathrm{t}}(\ell)]\!]_\sigma > [\![r_{\mathrm{t}}(\ell')]\!]_{\sigma'}$ *and* $[\![r_{\mathrm{t}}(\ell)]\!]_\sigma > 0$
(c) *if* $\tau \in \mathrm{fun}(\mathcal{T}')$, *then* $[\![r_{\mathrm{f}}(\ell)]\!]_\sigma > [\![r_{\mathrm{f}}(\ell')]\!]_{\sigma'}$ *and* $[\![r_{\mathrm{f}}(\ell)]\!]_\sigma > 0$
(d) *if* $\tau \in \mathcal{T}' \cup \mathrm{fun}(\mathcal{T}')$, *then* $[\![r_i(\ell)]\!]_\sigma \geq [\![r_i(\ell')]\!]_{\sigma'}$ *for all* $i \in \{\mathrm{tf}, \mathrm{t}, \mathrm{f}\}$

Note that if τ's update η contains function calls, then in general it is not decidable whether $(\ell, \sigma) \to_\tau (\ell', \sigma')$ holds. Thus, to over-approximate $\to_\tau$ in our automation, we consider a modified update η' where all function calls are replaced by fresh variables. In practice, we restrict ourselves to linear polynomial ranking functions and use the SMT solver Z3 [20] to infer ρ-RFs automatically.

Example 16. For the program from Fig. 2, we first consider $\mathcal{T}' = \{t_1\}$. Then $\mathrm{fun}(\mathcal{T}') = \varnothing$ and $\mathrm{fun}^{-1}(\mathcal{T}') = \varnothing$, since the location f_1 of t_1's function call is not in $\mathcal{L}_{\mathcal{T}'}$. A ρ-RF for $\mathcal{T}'_> = \mathcal{T}'$ is $r_{\mathrm{tf}}(\ell_1) = r_{\mathrm{f}}(\ell_1) = 0$ and $r_{\mathrm{t}}(\ell_1) = x$. The ranking functions can always map all remaining locations outside the subprogram $\mathcal{T}'$ to 0. For $\widetilde{\mathcal{T}'} = \{t_4, t_5\}$ we have $\mathrm{fun}(\widetilde{\mathcal{T}'}) = \{\rho_2\}$ and $\mathrm{fun}^{-1}(\widetilde{\mathcal{T}'}) = \{t_5\}$. A ρ-RF for $\widetilde{\mathcal{T}'_>} = \{t_5\}$ is $\widetilde{r_{\mathrm{tf}}}(f_1) = 1$, $\widetilde{r_{\mathrm{tf}}}(f_2) = \widetilde{r_{\mathrm{t}}}(f_1) = \widetilde{r_{\mathrm{t}}}(f_2) = 0$, and $\widetilde{r_{\mathrm{f}}}(f_1) = \widetilde{r_{\mathrm{f}}}(f_2) = a$.

The following theorem shows that ρ-RFs yield local runtime bounds. For a local runtime bound, we have to over-approximate how many edges labeled with $\mathcal{T}'_>$ can occur in a $\mathcal{T}'$-evaluation tree starting with a configuration of the form $(\ell, \_)$. To this end, for any transition t let $\mathrm{nfc}_{\mathcal{T}'}(t)$ denote the **n**umber of **f**unction **c**alls $\ell(\_|\_) \in \mathrm{fun}(t)$ with $\ell \in \mathcal{L}_{\mathcal{T}'}$, and let $\mathrm{nfc}(\mathcal{T}') = \max\{\mathrm{nfc}_{\mathcal{T}'}(t) \mid t \in \mathcal{T}'\}$. The ranking functions r_{tf}, r_{t}, and r_{f} influence the local runtime bound in different ways: If every path has at most n_0 edges labeled with transitions from $\mathrm{fun}^{-1}(\mathcal{T}')$, n_1 edges labeled with transitions from $\mathcal{T}'_> \setminus \mathrm{fun}^{-1}(\mathcal{T}')$, and n_2 edges labeled with function calls, then $\mathcal{R}_{n_1}(n_0, n_2)$ over-approximates the number of $\mathcal{T}'_>$-edges in any $\mathcal{T}'$-evaluation tree, where $\mathcal{R}_{n_1}(n_0, n_2)$ is defined via the following recurrence:

$$\mathcal{R}_{n_1}(n_0, n_2) = \begin{cases} n_1, & \text{if } n_0 = 0,\ n_2 = 0, \text{ or } \mathrm{nfc}(\mathcal{T}') = 0 \\ 1 + n_1 + \mathcal{R}_{n_1}(n_0 - 1, n_2) + \mathrm{nfc}(\mathcal{T}') \cdot \mathcal{R}_{n_1}(n_0, n_2 - 1), & \text{otherwise} \end{cases}$$

This can be shown by induction on $n_0 + n_2$. If $n_0 = 0, n_2 = 0$, or $\mathrm{nfc}(\mathcal{T}') = 0$, then there is no function call and thus, there can be at most n_1 edges labeled with transitions from $\mathcal{T}'_> \setminus \mathrm{fun}^{-1}(\mathcal{T}') = \mathcal{T}'_>$. The induction step is illustrated in Fig. 3. Here, the path from the root node to the first node N where a function is called uses at most $d_1 \leq n_1$ edges labeled with transitions from $\mathcal{T}'_> \setminus \mathrm{fun}^{-1}(\mathcal{T}')$. The node N has at most $\mathrm{nfc}(\mathcal{T}')$ many outgoing edges labeled with function calls

and one outgoing edge to a node N' labeled with a transition from $\mathrm{fun}^{-1}(\mathcal{T}')$. The function calls lead to at most $\mathrm{nfc}(\mathcal{T}')$ many subtrees where each contains at most $\mathcal{R}_{n_1}(n_0, n_2 - 1)$ many $\mathcal{T}'_>$-edges by the induction hypothesis. The path from the node N' to the next node N'' where a function is called uses at most d_2 edges labeled with transitions from $\mathcal{T}'_> \setminus \mathrm{fun}^{-1}(\mathcal{T}')$, where we have $d_1 + d_2 \leq n_1$. Finally, the subtree starting in node N'' has at most $\mathcal{R}_{n_1}(n_0 - 1, n_2)$ many $\mathcal{T}'_>$-edges by the induction hypothesis. Thus, the full tree has at most $|\mathbb{T}|_{\mathcal{T}'_>} \leq d_1 + 1 + \mathrm{nfc}(\mathcal{T}') \cdot \mathcal{R}_{n_1}(n_0, n_2 - 1) + d_2 + \mathcal{R}_{n_1}(n_0 - 1, n_2) \leq 1 + n_1 + \mathcal{R}_{n_1}(n_0 - 1, n_2) + \mathrm{nfc}(\mathcal{T}') \cdot \mathcal{R}_{n_1}(n_0, n_2 - 1)$ many $\mathcal{T}'_>$-edges.

As shown in App. A, $n_1 + n_2 \cdot (1 + n_1 \cdot (1 + \mathrm{nfc}(\mathcal{T}'))) \cdot (\mathrm{nfc}(\mathcal{T}') \cdot n_0)^{n_2}$ is an over-approximating closed form solution of $\mathcal{R}_{n_1}(n_0, n_2)$. Hence, instantiating this closed form with the ranking functions yields the desired local runtime bound. Here and subsequently, $[\![\cdot]\!]$ is used to transform a polynomial into a bound from $\mathcal{B}$ by taking the absolute values of the coefficients, e.g., $[\![x - y]\!] = x + y$.

Theorem 17 (Local Runtime Bounds by ρ-RFs). *Let $\varnothing \neq \mathcal{T}'_> \subseteq \mathcal{T}' \subseteq \mathcal{T} \setminus \mathcal{T}_0$ with $\mathrm{fun}^{-1}(\mathcal{T}') \subseteq \mathcal{T}'_>$ and let $\langle r_{\mathrm{tf}}, r_{\mathrm{t}}, r_{\mathrm{f}} \rangle$ be a ρ-RF. Then $\mathcal{RB}_{\mathrm{loc}}^{\mathcal{T}'_>, \mathcal{T}'}$ is local runtime bound for $\mathcal{T}'_>$ w.r.t. $\mathcal{T}'$, where for all $\ell \in \mathcal{L}_{\mathcal{T}'}$, we define $\mathcal{RB}_{\mathrm{loc}}^{\mathcal{T}'_>, \mathcal{T}'}(\ell)$ as:*

$$[\![r_{\mathrm{t}}(\ell)]\!] \;+\; [\![r_{\mathrm{f}}(\ell)]\!] \;\cdot\; (1 + [\![r_{\mathrm{t}}(\ell)]\!] \cdot (1 + \mathrm{nfc}(\mathcal{T}'))) \;\cdot\; (\mathrm{nfc}(\mathcal{T}') \cdot [\![r_{\mathrm{tf}}(\ell)]\!])^{[\![r_{\mathrm{f}}(\ell)]\!]}$$

Example 18. With the ρ-RFs of Ex. 16, Thm. 17 yields $\mathcal{RB}_{\mathrm{loc}}^{\{t_1\}, \{t_1\}}(\ell_1) = [\![r_{\mathrm{t}}(\ell_1)]\!] = x$ (as $r_{\mathrm{f}}(\ell_1) = 0$) and $\mathcal{RB}_{\mathrm{loc}}^{\{t_5\}, \{t_4, t_5\}}(f_1) = a$ (as $\mathrm{nfc}(\widetilde{\mathcal{T}'}) = 1$).

To lift local to global runtime bounds, we consider those transitions and function calls which start an evaluation of the subprogram $\mathcal{T}'$.

Definition 19 (Entry Points). *Let $\varnothing \neq \mathcal{T}' \subseteq \mathcal{T} \setminus \mathcal{T}_0$ and let $\mathcal{F}_{\mathcal{T}'} = \{\ell(\_|\_) \in \mathcal{F} \mid \ell \in \mathcal{L}_{\mathcal{T}'}\}$ be the set of function calls in the full program $\mathcal{T}$ that refer to start locations of $\mathcal{T}'$. Then $\mathcal{ET}_{\mathcal{T}'} = \{r \in \mathcal{T} \setminus \mathcal{T}' \mid \exists\, \ell \in \mathcal{L}_{\mathcal{T}'}.\, r = (\_, \_, \_, \ell)\}$ is the set of* entry transitions *and $\mathcal{EF}_{\mathcal{T}'} = \{r \in \mathcal{T} \setminus \mathcal{T}' \mid \mathrm{fun}(r) \cap \mathcal{F}_{\mathcal{T}'} \neq \varnothing\}$ is the set of* entry (function) calls *for $\mathcal{T}'$. $\mathcal{E}_{\mathcal{T}'} = \mathcal{ET}_{\mathcal{T}'} \cup \mathcal{EF}_{\mathcal{T}'}$ is the set of* entry points *for $\mathcal{T}'$.*

Example 20. For Fig. 2, we get $\mathcal{L}_{\{t_1\}} = \{\ell_1\}$, $\mathcal{F}_{\{t_1\}} = \varnothing$, and $\mathcal{E}_{\{t_1\}} = \mathcal{ET}_{\{t_1\}} = \{t_0\}$. Moreover, $\mathcal{L}_{\{t_4, t_5\}} = \{f_1\}$, $\mathcal{F}_{\{t_4, t_5\}} = \{\rho_1, \rho_2\}$, and $\mathcal{E}_{\{t_4, t_5\}} = \mathcal{EF}_{\{t_4, t_5\}} = \{t_1\}$.

To illustrate that $\mathcal{RB}_{\mathrm{loc}}^{\mathcal{T}'_>, \mathcal{T}'}(\ell_{in})$ is a bound on the number of evaluations of transitions from $\mathcal{T}'_>$ after evaluating a particular entry transition r or a function call ρ, we also write $\mathcal{RB}_{\mathrm{loc}}^{\mathcal{T}'_>}(\to_r \mathcal{T}')$ or $\mathcal{RB}_{\mathrm{loc}}^{\mathcal{T}'_>}(\to_\rho \mathcal{T}')$ instead of $\mathcal{RB}_{\mathrm{loc}}^{\mathcal{T}'_>, \mathcal{T}'}(\ell_{in})$ if $r = (\_, \_, \_, \ell_{in}) \in \mathcal{ET}_{\mathcal{T}'}$ or if $\rho = \ell_{in}(\_|\_) \in \mathrm{fun}(r) \cap \mathcal{F}_{\mathcal{T}'}$ for some $r \in \mathcal{EF}_{\mathcal{T}'}$.

Thm. 21 allows us to lift arbitrary local runtime bounds of a subprogram (e.g., local runtime bounds by ρ-RFs) to global runtime bounds for the full program. To this end, we consider $\mathcal{RB}(r)$ to over-approximate how often a local run of $\mathcal{T}'$ is started by an entry point $r \in \mathcal{E}_{\mathcal{T}'}$. In contrast to our previous work [10, 15, 18], we also have to consider *entry calls* r. Furthermore, we have to consider the size of the program variables after entering the subprogram by r or by a function call ρ in r. Hence, we replace every program variable $v \in \mathcal{PV}$ by its size bound $\mathcal{SB}(\tau, v)$ for $\tau = r$ or $\tau = \rho$, respectively. This is denoted by "$[v/\mathcal{SB}(\tau, v) \mid v \in \mathcal{PV}]$".

Theorem 21 (Lifting Local Runtime Bounds). *Let $\mathcal{RB}$ be a global runtime bound, $\mathcal{SB}$ be a size bound, and $\varnothing \neq \mathcal{T}'_> \subseteq \mathcal{T}' \subseteq \mathcal{T} \setminus \mathcal{T}_0$. Moreover, let $\mathcal{RB}_{\text{loc}}^{\mathcal{T}'_>,\mathcal{T}'}$ be a local runtime bound for $\mathcal{T}'_>$ w.r.t. $\mathcal{T}'$. Then $\mathcal{RB}'$ is also a global runtime bound, where $\mathcal{RB}'(t) = \mathcal{RB}(t)$ for all $t \in \mathcal{T} \setminus \mathcal{T}'_>$ and for $t \in \mathcal{T}'_>$, we have:*

$$\mathcal{RB}'(t) = \sum_{r \in \mathcal{ET}_{\mathcal{T}'}} \mathcal{RB}(r) \cdot \mathcal{RB}_{\text{loc}}^{\mathcal{T}'_>}(\to_r \mathcal{T}')\,[v/\mathcal{SB}(r,v) \mid v \in \mathcal{PV}]$$
$$+ \sum_{r \in \mathcal{EF}_{\mathcal{T}'}} \sum_{\rho \in \text{fun}(r) \cap \mathcal{F}_{\mathcal{T}'}} \mathcal{RB}(r) \cdot \mathcal{RB}_{\text{loc}}^{\mathcal{T}'_>}(\to_\rho \mathcal{T}')\,[v/\mathcal{SB}(\rho,v) \mid v \in \mathcal{PV}]$$

Example 22. We now compute the remaining global runtime bounds for Fig. 2, see Ex. 11. For t_1 and t_5, we had inferred the local runtime bounds $\mathcal{RB}_{\text{loc}}^{\{t_1\}}(\to_{t_0} \{t_1\}) = x$ and $\mathcal{RB}_{\text{loc}}^{\{t_5\}}(\to_{\rho_1} \{t_4, t_5\}) = a$ in Ex. 18. Thus, we obtain $\mathcal{RB}(t_1) = \mathcal{RB}(t_0) \cdot \mathcal{RB}_{\text{loc}}^{\{t_1\}}(\to_{t_0} \{t_1\})\,[x/\mathcal{SB}(t_0,x)] = x$ (with $\mathcal{RB}(t_0) = 1$ and $\mathcal{SB}(t_0,x) = x$) and $\mathcal{RB}(t_5) = \mathcal{RB}(t_1) \cdot \mathcal{RB}_{\text{loc}}^{\{t_5\}}(\to_{\rho_1} \{t_4, t_5\})\,[a/\mathcal{SB}(\rho_1,a)] = x^2$ (with $\mathcal{RB}(t_1) = x$ and $\mathcal{SB}(\rho_1,a) = x$) by Thm. 21. Similarly, $\mathcal{RB}_{\text{loc}}^{\{t_4\}}(\to_{\rho_1} \{t_4, t_5\}) = 1$ is a local runtime bound since the subprogram $\{t_4\}$ consists of a single transition without function calls. Here, we get $\mathcal{RB}(t_4) = \mathcal{RB}(t_1) \cdot \mathcal{RB}_{\text{loc}}^{\{t_4\}}(\to_{\rho_1} \{t_4, t_5\}) = x$. Finally, $\mathcal{RB}_{\text{loc}}^{\{t_3\}}(\to_{t_2} \{t_3\}) = \log_2(y) + 2$ is also a local runtime bound, which cannot be inferred by linear ranking functions but by our technique based on so-called twn-loops [11, 12, 15, 16, 18] (see App. B for the detailed construction). Lifting this local bound by Thm. 21 yields the global bound $\mathcal{RB}(t_3) = \mathcal{RB}(t_2) \cdot \mathcal{RB}_{\text{loc}}^{\{t_3\}}(\to_{t_2} \{t_3\})\,[y/\mathcal{SB}(t_2,y)] = \log_2(y + x^{x^2+1}) + 2$ (with $\mathcal{RB}(t_2) = 1$ and $\mathcal{SB}(t_2,y) = y + x^{x^2+1}$). Thus, our modular approach allows us to consider individual subprograms separately, to use different techniques to compute their local bounds, and to combine these local bounds into a global bound afterwards.

4 Modular Computation of Size Bounds

We now introduce our modular approach to compute size bounds. To this end, we extend the technique of [6] to handle ITSs with function calls. For every result variable $\langle \tau, v \rangle \in \mathcal{RV} = (\mathcal{T} \cup \mathcal{F}) \times \mathcal{PV}$, we define a *local size bound* $\mathcal{SB}_{\text{loc}}(\tau, v) \in \mathbb{N}[\mathcal{PV} \cup \mathcal{F}]$. So $\mathcal{SB}_{\text{loc}}(\tau, v)$ is a polynomial over the program variables and function calls (which are treated like variables). When instantiating every function call $\rho = \ell'_\rho(v_\rho|\_)$ in $\mathcal{SB}_{\text{loc}}(\tau, v)$ by the size $|\sigma_\rho|(v_\rho)$ of its result, then $\mathcal{SB}_{\text{loc}}(\tau, v)$ must be a bound on the size of v after a single evaluation step with τ.

Definition 23 (Local Size Bound). $\mathcal{SB}_{\text{loc}} : \mathcal{RV} \to \mathbb{N}[\mathcal{PV} \cup \mathcal{F}]$ *is a* local size bound *if for all $\langle \tau, v \rangle \in \mathcal{RV}$, all evaluations $(\ell', \sigma') \to_\tau (\ell, \sigma)$ with $\sigma \neq \bot$, and all evaluations $(\ell', \sigma') \to_\rho \circ \to^* (\ell_\rho, \sigma_\rho)$ starting with some $\rho = \ell'_\rho(v_\rho|\_) \in \mathcal{F}$ such that $\ell_\rho \in \Omega$ and $\sigma_\rho \neq \bot$, we have $|\sigma|(v) \leq [\![\, \mathcal{SB}_{\text{loc}}(\tau, v)\,[\rho \,/\, |\sigma_\rho|(v_\rho) \mid \rho \in \mathcal{F}] \,]\!]_{|\sigma'|}$.*

For every result variable $\langle t, v \rangle \in \mathcal{T} \times \mathcal{PV}$ with $t = (\_, \_, \eta, \_)$, in practice we essentially use $\mathcal{SB}_{\text{loc}}(t, v) = [\![\eta(v)]\!]$, e.g., $\mathcal{SB}_{\text{loc}}(t_1, y) = y + \rho_1$ and $\mathcal{SB}_{\text{loc}}(t_1, x) = [\![x-1]\!] = x+1$ for the program from Fig. 2. However, due to the guard $x > 0$ of t_1, here we can obtain the more precise local size bound $\mathcal{SB}_{\text{loc}}(t_1, x) = x$. Similarly,

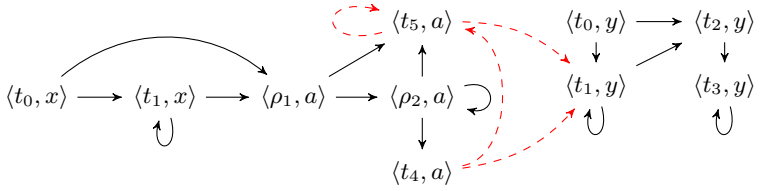

Fig. 4: Part of the RVG for Fig. 2

we essentially use $\mathcal{SB}_{\text{loc}}(\rho, v) = [\![\zeta(v)]\!]$ for every result variable $(\rho, v) \in \mathcal{F} \times \mathcal{PV}$ with $\rho = \ell(\_|\zeta)$. So for Fig. 2, we would obtain $\mathcal{SB}_{\text{loc}}(\rho_2, a) = [\![a - 1]\!] = a + 1$. However, due to the guard $a > 0$ of the transition t_5 whose update contains the function call ρ_2, we can again obtain the more precise bound $\mathcal{SB}_{\text{loc}}(\rho_2, a) = a$.

Next we construct a *result variable graph* (RVG) which represents the influence of result variables on each other. For any polynomial $p \in \mathbb{N}[\mathcal{PV} \cup \mathcal{F}]$, let $\text{act}(p) \subseteq \mathcal{PV} \cup \mathcal{F}$ denote the set of its *active arguments*, i.e., $x \in \text{act}(p)$ iff $x \in \mathcal{PV} \cup \mathcal{F}$ occurs in p. Furthermore, let $\text{pre}(\tau)$ denote transitions or function calls that directly precede τ, i.e., $\tau' \in \text{pre}(\tau)$ iff there exists an evaluation which contains the path $\to_{\tau'} \circ \to_{\tau}$. Then the RVG has the nodes $\mathcal{RV}$ and it has an edge from $\langle \tau', v' \rangle$ to $\langle \tau, v \rangle$ whenever $\tau' \in \text{pre}(\tau)$ and $v' \in \text{act}(\mathcal{SB}_{\text{loc}}(\tau, v))$.

Moreover, we have to consider the values of function calls. We say that t' is an *Ω-predecessor* of (t, ρ) if the transition t' ends in a return location which is reachable from a function call $\rho = \ell'_\rho(\_|\_) \in \text{fun}(t)$, i.e., if there is an evaluation $\to_\rho (\ell'_\rho, \_) \to \cdots \to_{t'} (\ell_\rho, \_)$ for some $\ell_\rho \in \Omega$. Let $\text{pre}^\Omega(t, \rho)$ denote the set of all Ω-predecessors of (t, ρ). Whenever $t = (\_, \_, \eta, \_)$, $\rho = \ell'_\rho(v'|\_) \in \text{fun}(\eta(v))$, and $t' \in \text{pre}^\Omega(t, \rho)$, then there is an Ω-edge from $\langle t', v' \rangle$ to $\langle t, v \rangle$ in the RVG.

So for Fig. 2, we have $\text{pre}^\Omega(t_1, \rho_1) = \{t_4, t_5\}$, since both t_4 and t_5 end in the return location f_2, t_1's update of y contains the function call $\rho_1 = f_1(a|\_)$, and there are evaluations $\to_{\rho_1} (f_1, \_) \to_{t_4} (f_2, \_)$ and $\to_{\rho_1} (f_1, \_) \to_{t_5} (f_2, \_)$. Thus, there are Ω-edges from both $\langle t_4, a \rangle$ and $\langle t_5, a \rangle$ to $\langle t_1, y \rangle$ in the RVG. In practice, we use efficiently computable over-approximations for $\text{pre}(\cdot)$ and $\text{pre}^\Omega(\cdot, \cdot)$.

Definition 24 (Result Variable Graph). *An* RVG *has the nodes $\mathcal{RV}$, the RV-edges $\{(\langle \tau', v' \rangle, \langle \tau, v \rangle) \mid \tau' \in \text{pre}(\tau), v' \in \text{act}(\mathcal{SB}_{\text{loc}}(\tau, v))\}$, and the Ω-edges $\{(\langle t', v' \rangle, \langle t, v \rangle) \mid t' \in \text{pre}^\Omega(t, \rho),\ t = (\_, \_, \eta, \_),\ \rho = \ell'_\rho(v'|\_) \in \text{fun}(\eta(v))\}$.*

Example 25. Fig. 4 depicts a part of the RVG for the program of Fig. 2. For instance, there is an edge $\langle t_1, x \rangle \to \langle \rho_1, a \rangle$ as the value of x after transition t_1 influences the value of a after the function call ρ_1. More precisely, $t_1 \in \text{pre}(\rho_1) = \{t_0, t_1\}$ and $x \in \text{act}(\mathcal{SB}_{\text{loc}}(\rho_1, a)) = \text{act}(x) = \{x\}$. Note that we do not have an edge from $\langle \rho_1, a \rangle$ to $\langle t_4, a \rangle$ as $\mathcal{SB}_{\text{loc}}(t_4, a) = 1$. The RVG has four Ω-edges which are red-dashed in Fig. 4. Moreover, Fig. 4 has the five non-trivial[2] SCCs $\{\langle t_1, x \rangle\}$, $\{\langle \rho_2, a \rangle\}$, $\{\langle t_1, y \rangle\}$, $\{\langle t_3, y \rangle\}$, and $\{\langle t_5, a \rangle\}$, where the latter forms a

[2] As usual, a *strongly connected component* (SCC) is a maximal subgraph with a path from each node to every other node. An SCC is *trivial* if it consists of a single node without an edge to itself.

cycle with an Ω-edge. While the full RVG has additional non-trivial SCCs, we omitted them from Fig. 4 as they have no impact on the runtime.

We already developed powerful techniques to lift local to global size bounds for ITSs without function calls in [6, 16, 18]. We now extend the technique of [6] to handle function calls.

We start with size bounds for *trivial* SCCs $\{\langle \tau, x\rangle\}$ in the RVG. Thm. 26 considers the case where $\tau \in \mathcal{F}$ or $\tau \in \mathcal{T}$ with an update η such that $\mathrm{fun}(\eta(x)) = \varnothing$. The case $\mathrm{fun}(\eta(x)) \neq \varnothing$ is handled in Thm. 28. If τ is an initial transition, i.e., $\mathrm{pre}(\tau) = \varnothing$, then $\mathcal{SB}_{\mathrm{loc}}(\tau, x)$ is already a (global) size bound. Otherwise, if $\mathrm{pre}(\tau) \neq \varnothing$, then Thm. 26 over-approximates the sizes of the variables in $\mathcal{SB}_{\mathrm{loc}}(\tau, x)$ by the size bounds corresponding to the preceding transitions.

Theorem 26 (Size Bounds for Trivial SCCs Without Function Calls). *Let $\mathcal{SB}$ be a size bound and $\{\langle \tau, x\rangle\}$ be a trivial SCC of the RVG such that $\tau \in \mathcal{F}$ or $\mathrm{fun}(\eta(x)) = \varnothing$ for the update η of $\tau \in \mathcal{T}$. Then $\mathcal{SB}'$ is also a size bound where $\mathcal{SB}'(\alpha) = \mathcal{SB}(\alpha)$ for all $\alpha \neq \langle \tau, x\rangle$, and for $\alpha = \langle \tau, x\rangle$ we have*

$$\mathcal{SB}'(\alpha) = \begin{cases} \mathcal{SB}_{\mathrm{loc}}(\alpha), & \text{if } \mathrm{pre}(\tau) = \varnothing \\ \max_{\tau' \in \mathrm{pre}(\tau)} \{\mathcal{SB}_{\mathrm{loc}}(\alpha)\,[v/\mathcal{SB}(\tau', v) \mid v \in \mathcal{PV}]\}, & \text{otherwise} \end{cases}$$

Note that due to the requirement on $\langle \tau, x\rangle$ in Thm. 26, w.l.o.g. $\mathcal{SB}_{\mathrm{loc}}(\tau, x)$ only contains variables from $\mathcal{PV}$, but not from $\mathcal{F}$.

Example 27. Reconsider Fig. 2 and 4. We have $\mathcal{SB}(t_0, x) = \mathcal{SB}_{\mathrm{loc}}(t_0, x) = x$ by Thm. 26 as $\mathrm{pre}(t_0) = \varnothing$. Moreover, we obtain $\mathcal{SB}(t_4, a) = 1$ since $\mathcal{SB}_{\mathrm{loc}}(t_4, a) = 1$. In Ex. 31 we will show that $\mathcal{SB}(t_1, x) = x$. Since $\mathcal{SB}_{\mathrm{loc}}(\rho_1, a) = x$ and $\mathrm{pre}(\rho_1) = \{t_0, t_1\}$, this implies $\mathcal{SB}(\rho_1, a) = \max\{x[x/\mathcal{SB}(t_0, x)], x[x/\mathcal{SB}(t_1, x)]\} = x$.

The following theorem handles trivial SCCs $\{\langle t, x\rangle\}$ where t contains function calls ρ. Hence, in contrast to Thm. 26, we have to instantiate these function calls by the size bounds for the transitions of $\mathrm{pre}^{\Omega}(t, \rho)$, as they reach the corresponding return locations. If a function call $\rho \in \mathrm{fun}(t)$ does not reach a return location, then we can set the size bound for $\langle t, x\rangle$ to 0, because then $\to_t$ only reaches configurations $(\_, \bot)$. For any $n \in \mathbb{N}$, let $[n]$ denote the set $\{1, \ldots, n\}$.

Theorem 28 (Size Bounds for Trivial SCCs With Function Calls). *Let $\mathcal{SB}$ be a size bound and $\{\langle t, x\rangle\}$ be a trivial SCC of the RVG such that $t \in \mathcal{T}$ and $\mathrm{fun}(\eta(x)) \neq \varnothing$. Then $\mathcal{SB}'$ is also a size bound where $\mathcal{SB}'(\alpha) = \mathcal{SB}(\alpha)$ for $\alpha \neq \langle t, x\rangle$, and for $\alpha = \langle t, x\rangle$ with $\mathrm{fun}(\eta(x)) = \{\rho_1, \ldots, \rho_n\}$ and $\rho_i = \ell_i(v_i|\_)$, we have*

$$\mathcal{SB}'(\alpha) = \begin{cases} 0, & \text{if } \mathrm{pre}^{\Omega}(t, \rho_i) = \varnothing \text{ for some } i \in [n] \\ \max_{t'_i \in \mathrm{pre}^{\Omega}(t,\rho_i) \text{ for all } i \in [n]} \{\mathcal{SB}_{\mathrm{loc}}(\alpha)\,[\rho_i/\mathcal{SB}(t'_i, v_i) \mid i \in [n]]\}, & \text{if all } \mathrm{pre}^{\Omega}(t, \rho_i) \neq \varnothing \text{ and } \mathrm{pre}(t) = \varnothing \\ \max\limits_{\substack{\tau' \in \mathrm{pre}(t) \\ t'_i \in \mathrm{pre}^{\Omega}(t,\rho_i) \text{ for all } i \in [n]}} \{\mathcal{SB}_{\mathrm{loc}}(\alpha)\,[v/\mathcal{SB}(\tau', v) \mid v \in \mathcal{PV}]\,[\rho_i/\mathcal{SB}(t'_i, v_i) \mid i \in [n]]\}, & \text{otherwise} \end{cases}$$

Example 29. Consider a variant of Fig. 2 where we replace the update $\eta(y)$ of t_1 by $\rho_1 = f_1(a \mid \zeta(a) = x)$. Thus, the self-loop at $\langle t_1, y\rangle$ is removed from the RVG in

Fig. 4. Then, we can apply Thm. 28 on the trivial SCC $\{\langle t_1, y\rangle\}$. Assume that we already computed $\mathcal{SB}(t_4, a) = 1$ and $\mathcal{SB}(t_5, a) = x^{x^2+1}$ (see Ex. 27 and 31). We have $\mathrm{pre}(t_1) = \{t_0, t_1\}$, but $\mathcal{SB}_{\mathrm{loc}}(t_1, y) = \rho_1$ does not contain variables from $\mathcal{PV}$. Hence, we get $\mathcal{SB}(t_1, y) = \max\{\rho_1[\rho_1/\mathcal{SB}(t_4, a)], \rho_1[\rho_1/\mathcal{SB}(t_5, a)]\} = x^{x^2+1}$.

Finally, we introduce our approach to handle non-trivial SCCs. Let $C \subseteq \mathcal{RV}$ be the nodes of such an SCC. Our approach can only be applied to SCCs where for all $\alpha \in C$, there exist $e_\alpha \in \mathbb{N}$ and $s_\alpha \in \mathbb{N}[\mathcal{PV}]$ such that

$$\mathcal{SB}_{\mathrm{loc}}(\alpha) \leq s_\alpha \cdot (\ e_\alpha + \textstyle\sum_{v \,\in\, \mathrm{act}(\mathcal{SB}_{\mathrm{loc}}(\alpha)) \setminus \mathrm{act}(s_\alpha)} v\ ) \tag{1}$$

where "$\leq$" is interpreted pointwise (i.e., the inequation must hold for all instantiations of the variables by natural numbers). Here, s_α captures the scaling behavior of $\mathcal{SB}_{\mathrm{loc}}(\alpha)$, e.g., it allows us to consider updates of the form $\eta(x) = 2 \cdot x$ or $\eta(x) = a \cdot x$ for a variable $a \in \mathcal{PV}$. Note that in [6], only constant factors s_α were allowed. Similarly, e_α captures the additive growth in updates like $\eta(x) = 1 + x$.

We now also define pre and pre^Ω for result variables. For $\alpha \in \mathcal{RV}$, $\mathrm{pre}(\alpha)$ ($\mathrm{pre}^\Omega(\alpha)$) is the set of all result variables α' with an RV-edge (Ω-edge) from α' to α in the RVG. Furthermore, for any result variable α in the SCC C, let $V_\alpha = \{v \in \mathcal{PV} \mid \exists \tau.\, (\tau, v) \in \mathrm{pre}(\alpha) \cap C\}$ be the set of all variables v with an RV-edge to α in C, and similarly, let $F_\alpha = \{v \in \mathcal{PV} \mid \exists t.\, (t, v) \in \mathrm{pre}^\Omega(\alpha) \cap C\}$. Finally, for any $p \in \mathbb{N}[\mathcal{PV} \cup \mathcal{F}]$, let $\mathrm{actV}(p) = \mathrm{act}(p) \cap \mathcal{PV}$ be p's *active variables* and $\mathrm{actF}(p) = \mathrm{act}(p) \cap \mathcal{F}$ be the *active function calls* of p.

To consider the additive growth, we over-approximate the sizes of variables on incoming edges from outside the SCC C. Let $\mathit{init}_\alpha(v) = \max\{\mathcal{SB}(\tau, v) \mid \exists \tau \in \mathcal{T} \cup \mathcal{F}.\, \langle \tau, v\rangle \in \mathrm{pre}(\alpha) \setminus C\}$ be a bound on the size of v when entering C via an RV-edge to α. Analogously, $\mathit{init}^\Omega_\alpha(v) = \max\{\mathcal{SB}(t, v) \mid \exists t \in \mathcal{T}.\, \langle t, v\rangle \in \mathrm{pre}^\Omega(\alpha) \setminus C\}$ is a bound on the size of v when entering C via an Ω-edge to α. The execution of α's transition or function call means that the values of the variables in V_α or F_α can be increased by adding $\mathit{init}_\alpha(v)$ for all $v \in \mathrm{actV}(\mathcal{SB}_{\mathrm{loc}}(\alpha)) \setminus V_\alpha$ (or, respectively, by adding $\mathit{init}^\Omega_\alpha(v)$ for all $\ell(v|\_) \in \mathrm{actF}(\mathcal{SB}_{\mathrm{loc}}(\alpha))$ where $v \notin F_\alpha$) plus the constant e_α. This can be repeated rb_α times, where $rb_\alpha = \mathcal{RB}(t)$ if $\alpha = \langle t, v\rangle$ and $rb_\alpha = \sum_{t \in \mathrm{fun}^{-1}(\rho)} \mathcal{RB}(t)$ if $\alpha = \langle \rho, v\rangle$, i.e., rb_α is a bound on how often α's transition or function call is evaluated during a program run. Thus, the following expression over-approximates the additive size-change resulting from α (ignoring the growth resulting from V_α, F_α, and $\mathrm{actV}(s_\alpha)$ for now):

$$\mathit{add}(\alpha) = rb_\alpha \cdot (\ e_\alpha + \sum_{\substack{v \,\in\, \mathrm{actV}(\mathcal{SB}_{\mathrm{loc}}(\alpha)) \\ v \,\notin\, \mathrm{actV}(s_\alpha) \cup V_\alpha}} \mathit{init}_\alpha(v) \quad + \sum_{\substack{\ell(v|\_) \,\in\, \mathrm{actF}(\mathcal{SB}_{\mathrm{loc}}(\alpha)) \\ v \,\notin\, F_\alpha}} \mathit{init}^\Omega_\alpha(v)\ )$$

We now take the growth resulting from the variables in V_α or F_α into account. First, since the expression $\mathit{add}(\alpha)$ only captures the *change* of the size, for an overall size bound, one has to take the initial values of the variables in V_α and F_α before entering the SCC C into account. This leads to

$$\mathit{add}(\alpha)\ +\ \textstyle\sum_{v \in V_\alpha}\ \mathit{init}_\alpha(v)\ +\ \sum_{v \in F_\alpha}\ \mathit{init}^\Omega_\alpha(v).$$

Moreover, if $|V_\alpha| + |F_\alpha| > 1$, then each execution of α's transition or function call may multiply the value of a variable by $|V_\alpha| + |F_\alpha|$. For example, consider

the update $\eta(x) = \eta(y) = x + y$ where $x, y \in V_\alpha$. Then, both x and y grow with a factor of two. A similar effect is obtained for scaling factors $s_\alpha > 1$. As for the additive growth, this multiplication must be performed rb_α times. This is captured by $scale(\alpha)$ for $\alpha = \langle \tau, \_ \rangle$:

$$scale(\alpha) = (\max_{\tau' \in \mathrm{pre}(\tau)} \{1, s_\alpha [v/\mathcal{SB}(\tau', v) \mid v \in \mathcal{PV}]\} \cdot (|V_\alpha| + |F_\alpha|))^{rb_\alpha}$$

The following theorem shows how to compute size bounds for non-trivial SCCs C by accumulating $scale(\alpha)$ and $add(\alpha)$ for all $\alpha \in C$. To simplify the presentation, in contrast to [6], we do not consider transitions individually and use a single expression instead of defining several classes of local size bounds.

Theorem 30 (Size Bounds for Non-Trivial SCCs). *Let $\mathcal{SB}$ be a size bound and C be a non-trivial SCC in a RVG, where for all $\alpha \in C$, $\mathcal{SB}_{\mathrm{loc}}(\alpha)$ satisfies (1) for suitable e_α and s_α. Then $\mathcal{SB}'$ is also a size bound where $\mathcal{SB}'(\alpha) = \mathcal{SB}(\alpha)$ for all $\alpha \in \mathcal{RV} \setminus C$, and $\mathcal{SB}'(\alpha) = \mathcal{SB}'(C)$ for all $\alpha \in C$, where*

$$\mathcal{SB}'(C) = \textstyle\prod_{\alpha \in C} scale(\alpha) \cdot (\sum_{\alpha \in C}(add(\alpha) + \sum_{v \in V_\alpha} init_\alpha(v) + \sum_{v \in F_\alpha} init^\Omega_\alpha(v)))$$

Example 31. Reconsider Fig. 2 and 4. We now infer size bounds for the non-trivial SCCs $\{\langle t_1, x\rangle\}$, $\{\langle \rho_2, a\rangle\}$, $\{\langle t_5, a\rangle\}$, and $\{\langle t_1, y\rangle\}$. For $\alpha = \langle t_1, x\rangle$ with $\mathcal{SB}_{\mathrm{loc}}(t_1, x) = x$, we have $s_\alpha = 1$, $e_\alpha = 0$, $V_\alpha = \{x\}$, $F_\alpha = \varnothing$, $\mathrm{actV}(\mathcal{SB}_{\mathrm{loc}}(t_1, x)) = \{x\}$, and $\mathrm{actF}(\mathcal{SB}_{\mathrm{loc}}(t_1, x)) = \varnothing$. This implies $scale(\alpha) = 1$, $add(\alpha) = 0$, and $init_\alpha(x) = \mathcal{SB}(t_0, x) = x$. Thus, we obtain the size bound $\mathcal{SB}(t_1, x) = x$.

Similarly, for $\alpha = \langle \rho_2, a\rangle$ with $\mathcal{SB}_{\mathrm{loc}}(\rho_2, a) = a$, we obtain $s_\alpha = 1$, $e_\alpha = 0$, $V_\alpha = \{a\}$, $F_\alpha = \varnothing$, $\mathrm{actV}(\mathcal{SB}_{\mathrm{loc}}(\rho_2, a)) = \{a\}$, and $\mathrm{actF}(\mathcal{SB}_{\mathrm{loc}}(\rho_2, a)) = \varnothing$. Thus, we have $scale(\alpha) = 1$, $add(\alpha) = 0$, and $init_\alpha(a) = \mathcal{SB}(\rho_1, a) = x$ by Ex. 27. This yields the size bound $\mathcal{SB}(\rho_2, a) = x$.

For $\alpha = \langle t_5, a\rangle$ with $\mathcal{SB}_{\mathrm{loc}}(t_5, a) = a \cdot \rho_2$, we have $s_\alpha = a$, $e_\alpha = 0$, $V_\alpha = \varnothing$, $F_\alpha = \{a\}$, $\mathrm{actV}(\mathcal{SB}_{\mathrm{loc}}(t_5, a)) = \{a\}$, and $\mathrm{actF}(\mathcal{SB}_{\mathrm{loc}}(t_5, a)) = \{\rho_2\}$. As $\mathrm{pre}(t_5) = \{\rho_1, \rho_2\}$, we have $scale(\alpha) = (\max\{1, a[a/\mathcal{SB}(\rho_1, a)], a[a/\mathcal{SB}(\rho_2, a)]\})^{\mathcal{RB}(t_5)} = x^{x^2}$ (with $\mathcal{RB}(t_5) = x^2$ by Ex. 22 and $\mathcal{SB}(\rho_1, a) = \mathcal{SB}(\rho_2, a) = x$ when using the invariant $x > 0$ and thus, $\max\{1, x\} = x$), $add(\alpha) = x^2 \cdot 0 = 0$, and $init^\Omega_\alpha(a) = \mathcal{SB}(t_4, a) = 1$ by Ex. 27. Hence, we obtain the size bound $\mathcal{SB}(t_5, a) = x^{x^2}$.

Finally, for $\alpha = \langle t_1, y\rangle$ with $\mathcal{SB}_{\mathrm{loc}}(t_1, y) = y + \rho_1$, we have $s_\alpha = 1$, $e_\alpha = 0$, $V_\alpha = \{y\}$, $F_\alpha = \varnothing$, $\mathrm{actV}(\mathcal{SB}_{\mathrm{loc}}(t_1, y)) = \{y\}$, and $\mathrm{actF}(\mathcal{SB}_{\mathrm{loc}}(t_5, a)) = \{\rho_1\}$. Thus, we obtain $scale(\alpha) = 1$, $add(\alpha) = \mathcal{RB}(t_1) \cdot init^\Omega_\alpha(a) = x \cdot \max\{\mathcal{SB}(t_4, a), \mathcal{SB}(t_5, a)\} = x \cdot x^{x^2} = x^{x^2+1}$ (with $\mathcal{RB}(t_1) = x$ by Ex. 22, $\mathcal{SB}(t_4, a) = 1$, and $\mathcal{SB}(t_5, a) = x^{x^2}$), and $init_\alpha(y) = \mathcal{SB}(t_0, y) = y$. Hence, we get the size bound $\mathcal{SB}(t_1, y) = y + x^{x^2+1}$. This also implies $\mathcal{SB}(t_2, y) = y + x^{x^2+1}$.

5 Conclusion, Implementation, and Related Work

In this paper we presented a novel framework for complexity analysis of integer programs with function calls. To this end, we introduced a new class of ranking functions and extended our modular approach for inferring runtime and size bounds [6, 10] such that we can now also handle return values of function calls.

Related Work: As mentioned in the introduction, there exist many approaches to analyze complexity of programs automatically, e.g., [1, 2, 4, 6–8, 10, 13, 17, 19, 23, 24]. However, only few of them focus on programs with recursion or function calls. While we already discussed an extension to recursive ITSs in [6], here the return values of function calls were ignored.

Techniques for complexity analysis of *term rewrite systems* (e.g., based on dependency pairs [3, 22]) can handle (possibly non-tail) recursion, but standard TRSs do not support built-in types like integers. However, some works studied connections between complexity analysis for TRSs and our approach from [6] for complexity analysis of ITSs. To this end, [21] introduced *recursive natural transition systems* with potential non-tail recursion. Here, the idea is to summarize (and subsequently eliminate) subprocedures by approximating their runtime and size. Thus, this approach does not benefit from techniques such as our new class of ranking functions which allows us to handle subprograms with function calls directly. In [26], dependency pairs for TRSs were extended by the computation of size bounds from [6] and the inference of runtime bounds via classical ranking functions and the Master Theorem in order to handle *logically constrained TRSs*.

Instead of representing integer programs as ITSs, there are also techniques based on so-called *cost equation systems* which can express non-tail recursive integer programs as well, e.g., [8]. This approach analyzes program parts independently and uses linear invariants to compose the results, i.e., it differs significantly from our approach which can also infer non-linear size bounds. Moreover, [13] presents an approach for automatic complexity analysis of OCaml programs, which however has limitations w.r.t. modularity, see [21].

Implementation: We implemented our novel results and integrated them into our tool KoAT which also features powerful techniques for subprograms without function calls [6, 10, 15–18].

In the beginning, KoAT preprocesses the program, e.g., by extending the guards of transitions with invariants inferred by Apron [14]. For all SMT problems (including the generation of ranking functions), KoAT uses Z3 [20].

To our knowledge, KoAT is currently the only tool which can infer a finite runtime bound for the recursive ITS from our leading example (Fig. 2). Furthermore, it is also the most powerful tool on "classical" ITSs without function calls (see the results at the annual *Termination and Complexity Competition* (termCOMP) [9] and [10, 15–18] for evaluations). While the *Termination Problems Data Base* [25] used at termCOMP contains a large collection of ITSs without function calls, up to now there does not exist any such standard benchmark set for ITSs with function calls. To demonstrate KoAT's power on recursive ρ-ITSs, we collected 15 typical such integer programs and our evaluation showed that KoAT infers finite runtime bounds for 14 of them. KoAT's source code, a binary, a Docker image, and details on our evaluation are available at:

https://koat.verify.rwth-aachen.de/function-calls

This website also contains details on our input format for ρ-ITSs and a *web interface* to run different configurations of KoAT directly online.

References

[1] E. Albert, P. Arenas, S. Genaim, G. Puebla, and D. Zanardini. "Cost Analysis of Object-Oriented Bytecode Programs". In: *Theoretical Computer Science* 413.1 (2012), pp. 142–159. DOI: 10.1016/j.tcs.2011.07.009.

[2] E. Albert, M. Bofill, C. Borralleras, E. Martín-Martín, and A. Rubio. "Resource Analysis Driven by (Conditional) Termination Proofs". In: *Theory and Practice of Logic Programming* 19.5-6 (2019), pp. 722–739. DOI: 10.1017/S1471068419000152.

[3] M. Avanzini and G. Moser. "A Combination Framework for Complexity". In: *Information and Computation* 248 (2016), pp. 22–55. DOI: 10.1016/J.IC.2015.12.007.

[4] A. M. Ben-Amram and S. Genaim. "On Multiphase-Linear Ranking Functions". In: *Proc. CAV '17*. LNCS 10427. 2017, pp. 601–620. DOI: 10.1007/978-3-319-63390-9_32.

[5] A. M. Ben-Amram, J. J. Doménech, and S. Genaim. "Multiphase-Linear Ranking Functions and Their Relation to Recurrent Sets". In: *Proc. SAS '19*. LNCS 11822. 2019, pp. 459–480. DOI: 10.1007/978-3-030-32304-2_22.

[6] M. Brockschmidt, F. Emmes, S. Falke, C. Fuhs, and J. Giesl. "Analyzing Runtime and Size Complexity of Integer Programs". In: *ACM Transactions on Programming Languages and Systems* 38 (2016), pp. 1–50. DOI: 10.1145/2866575.

[7] Q. Carbonneaux, J. Hoffmann, and Z. Shao. "Compositional Certified Resource Bounds". In: *Proc. PLDI '15*. 2015, pp. 467–478. DOI: 10.1145/2737924.2737955.

[8] A. Flores-Montoya and R. Hähnle. "Resource Analysis of Complex Programs with Cost Equations". In: *Proc. APLAS '14*. LNCS 8858. 2014, pp. 275–295. DOI: 10.1007/978-3-319-12736-1_15.

[9] J. Giesl, A. Rubio, C. Sternagel, J. Waldmann, and A. Yamada. "The Termination and Complexity Competition". In: *Proc. TACAS '19*. LNCS 11429. 2019, pp. 156–166. DOI: 10.1007/978-3-030-17502-3_10.

[10] J. Giesl, N. Lommen, M. Hark, and F. Meyer. "Improving Automatic Complexity Analysis of Integer Programs". In: *The Logic of Software. A Tasting Menu of Formal Methods*. LNCS 13360. 2022, pp. 193–228. DOI: 10.1007/978-3-031-08166-8_10.

[11] M. Hark, F. Frohn, and J. Giesl. "Polynomial Loops: Beyond Termination". In: *Proc. LPAR '20*. EPiC 73. 2020, pp. 279–297. DOI: 10.29007/nxv1.

[12] M. Hark, F. Frohn, and J. Giesl. "Termination of Triangular Polynomial Loops". In: *Formal Methods in System Design* (2023). DOI: 10.1007/s10703-023-00440-z.

[13] J. Hoffmann, A. Das, and S.-C. Weng. "Towards Automatic Resource Bound Analysis for OCaml". In: *Proc. POPL '17*. 2017, pp. 359–373. DOI: 10.1145/3009837.3009842.

[14] B. Jeannet and A. Miné. "Apron: A Library of Numerical Abstract Domains for Static Analysis". In: *Proc. CAV '09*. LNCS 5643. 2009, pp. 661–667. DOI: 10.1007/978-3-642-02658-4_52.

[15] N. Lommen, F. Meyer, and J. Giesl. "Automatic Complexity Analysis of Integer Programs via Triangular Weakly Non-Linear Loops". In: *Proc. IJCAR '22*. LNCS 13385. 2022, pp. 734–754. DOI: 10.1007/978-3-031-10769-6_43.

[16] N. Lommen and J. Giesl. "Targeting Completeness: Using Closed Forms for Size Bounds of Integer Programs". In: *Proc. FroCoS '23*. LNCS 14279. 2023, pp. 3–22. DOI: 10.1007/978-3-031-43369-6_1.

[17] N. Lommen, É. Meyer, and J. Giesl. "Control-Flow Refinement for Complexity Analysis of Probabilistic Programs in KoAT (Short Paper)". In: *Proc. IJCAR '24*. LNCS 14739. 2024, pp. 233–243. DOI: 10.1007/978-3-031-63498-7_14.

[18] N. Lommen, É. Meyer, and J. Giesl. "Targeting Completeness: Automated Complexity Analysis of Integer Programs". In: *CoRR* abs/2412.01832 (2024). DOI: 10.48550/arXiv.2412.01832.

[19] P. López-García, L. Darmawan, M. Klemen, U. Liqat, F. Bueno, and M. V. Hermenegildo. "Interval-Based Resource Usage Verification by Translation into Horn Clauses and an Application to Energy Consumption". In: *Theory and Practice of Logic Programming* 18.2 (2018), pp. 167–223. DOI: 10.1017/S1471068418000042.

[20] L. M. de Moura and N. Bjørner. "Z3: An Efficient SMT Solver". In: *Proc. TACAS '08*. LNCS 4963. 2008, pp. 337–340. DOI: 10.1007/978-3-540-78800-3_24.

[21] M. Naaf, F. Frohn, M. Brockschmidt, C. Fuhs, and J. Giesl. "Complexity Analysis for Term Rewriting by Integer Transition Systems". In: *Proc. FroCoS '17*. LNCS 10483. 2017, pp. 132–150. DOI: 10.1007/978-3-319-66167-4_8.

[22] L. Noschinski, F. Emmes, and J. Giesl. "Analyzing Innermost Runtime Complexity of Term Rewriting by Dependency Pairs". In: *Journal of Automated Reasoning* 51.1 (2013), pp. 27–56. DOI: 10.1007/S10817-013-9277-6.

[23] L. Pham, F. A. Saad, and J. Hoffmann. "Robust Resource Bounds with Static Analysis and Bayesian Inference". In: *Proceedings of the ACM on Programming Languages* 8.PLDI (2024). DOI: 10.1145/3656380.

[24] M. Sinn, F. Zuleger, and H. Veith. "Complexity and Resource Bound Analysis of Imperative Programs Using Difference Constraints". In: *Journal of Automated Reasoning* 59.1 (2017), pp. 3–45. DOI: 10.1007/s10817-016-9402-4.

[25] *Termination Problems Data Base (TPDB)*. URL: https://github.com/TermCOMP/TPDB.

[26] S. Winkler and G. Moser. "Runtime Complexity Analysis of Logically Constrained Rewriting". In: *Proc. LOPSTR '20*. LNCS 12561. 2020, pp. 37–55. DOI: 10.1007/978-3-030-68446-4_2.

A Proofs

Theorem 17 (Local Runtime Bounds by ρ-RFs). *Let $\varnothing \neq \mathcal{T}'_> \subseteq \mathcal{T}' \subseteq \mathcal{T} \setminus \mathcal{T}_0$ with $\mathrm{fun}^{-1}(\mathcal{T}') \subseteq \mathcal{T}'_>$ and let $\langle r_{\mathrm{tf}}, r_{\mathrm{t}}, r_{\mathrm{f}} \rangle$ be a ρ-RF. Then $\mathcal{RB}_{\mathrm{loc}}^{\mathcal{T}'_>,\mathcal{T}'}$ is local runtime bound for $\mathcal{T}'_>$ w.r.t. $\mathcal{T}'$, where for all $\ell \in \mathcal{L}_{\mathcal{T}'}$, we define $\mathcal{RB}_{\mathrm{loc}}^{\mathcal{T}'_>,\mathcal{T}'}(\ell)$ as:*

$$[\![r_{\mathrm{t}}(\ell)]\!] \;+\; [\![r_{\mathrm{f}}(\ell)]\!] \;\cdot\; (1 + [\![r_{\mathrm{t}}(\ell)]\!] \cdot (1 + \mathrm{nfc}(\mathcal{T}'))) \;\cdot\; (\mathrm{nfc}(\mathcal{T}') \cdot [\![r_{\mathrm{tf}}(\ell)]\!])^{[\![r_{\mathrm{f}}(\ell)]\!]}$$

Proof. Let $(\{(\ell,\sigma)\},\varnothing) \prec^*_{\mathcal{T}' \cup \{\varepsilon\}} \mathbb{T}$ be an evaluation in the subprogram $\mathcal{T}'$ with $\ell \in \mathcal{L}_{\mathcal{T}'}$ and an arbitrary state $\sigma \in \Sigma$. We have to prove that

$$|\mathbb{T}|_{\mathcal{T}'_>} \leq [\![\mathcal{RB}_{\mathrm{loc}}^{\mathcal{T}'_>,\mathcal{T}'}(\ell)]\!]_{|\sigma|} \tag{2}$$

holds. To this end, we consider the following recurrence:

$$\mathcal{R}_{n_1}(n_0,n_2) = \begin{cases} n_1, & \text{if } n_0 = 0,\ n_2 = 0, \text{ or } \mathrm{nfc}(\mathcal{T}') = 0 \\ 1 + n_1 + \mathcal{R}_{n_1}(n_0 - 1, n_2) + \mathrm{nfc}(\mathcal{T}') \cdot \mathcal{R}_{n_1}(n_0, n_2 - 1), & \text{otherwise} \end{cases}$$

As shown in Sect. 3, by induction on n_0+n_2 one can prove that $\mathcal{R}_{n_1}(n_0,n_2)$ over-approximates the number of $\mathcal{T}'_>$-edges in any $\mathcal{T}'$-evaluation tree (i.e., $|\mathbb{T}|_{\mathcal{T}'_>} \leq \mathcal{R}_{n_1}(n_0,n_2)$), provided that every path has at most n_0 edges labeled with transitions from $\mathrm{fun}^{-1}(\mathcal{T}')$, n_1 edges labeled with transitions from $\mathcal{T}'_> \setminus \mathrm{fun}^{-1}(\mathcal{T}')$, and n_2 edges labeled with function calls. Now we show that $n_1 + n_2 \cdot (1 + n_1 \cdot (1 + \mathrm{nfc}(\mathcal{T}'))) \cdot (\mathrm{nfc}(\mathcal{T}') \cdot n_0)^{n_2}$ is an over-approximating closed form solution of $\mathcal{R}_{n_1}(n_0,n_2)$. Instantiating this closed form with the ranking functions yields the desired local runtime bound.

Let us abbreviate $c = \mathrm{nfc}(\mathcal{T}')$ and

$$f(n_0,n_1,n_2) = n_1 + n_2 \cdot (1 + n_1 \cdot (1 + c)) \cdot (c \cdot n_0)^{n_2}.$$

We show that for all $n_0, n_1, n_2 \in \mathbb{N}$ we have $\mathcal{R}_{n_1}(n_0,n_2) \leq f(n_0,n_1,n_2)$ by induction on n_2.

If $n_2 = 0$, then we have $\mathcal{R}_{n_1}(n_0,n_2) = f(n_0,n_1,0) = n_1$. Otherwise, if $n_2 > 0$ and $c \cdot n_0 = 0$, then we also have $\mathcal{R}_{n_1}(n_0,n_2) = f(n_0,n_1,n_2) = n_1$. Finally, if $n_2 > 0$, $c > 0$, and $n_0 > 0$, then we have

$$\begin{aligned} \mathcal{R}_{n_1}(n_0,n_2) &= 1 + n_1 + \mathcal{R}_{n_1}(n_0 - 1, n_2) + c \cdot \mathcal{R}_{n_1}(n_0, n_2 - 1) \\ &\leq 1 + n_1 + c \cdot f(n_0,n_1,n_2 - 1) + \mathcal{R}_{n_1}(n_0 - 1, n_2) \\ &\qquad\qquad \text{(by the induction hypothesis)} \\ &\leq \sum_{i=0}^{n_0-1} (1 + n_1 + c \cdot f(n_0 - i, n_1, n_2 - 1)) + \mathcal{R}_{n_1}(0,n_2) \end{aligned} \tag{3}$$

The last step (3) is clear if $n_0 = 1$. Otherwise if $n_0 > 1$, the reason for (3) is that we have

$$1 + n_1 + c \cdot f(n_0,n_1,n_2 - 1) + \mathcal{R}_{n_1}(n_0 - 1, n_2)$$

$$
\begin{aligned}
&= 1 + n_1 + c \cdot f(n_0, n_1, n_2 - 1) + \\
&\quad 1 + n_1 + \mathcal{R}_{n_1}(n_0 - 2, n_2) + c \cdot \mathcal{R}_{n_1}(n_0 - 1, n_2 - 1) \\
&\qquad\qquad \text{(evaluate } \mathcal{R}_{n_1}(n_0 - 1, n_2)\text{)} \\
&\leq 1 + n_1 + c \cdot f(n_0, n_1, n_2 - 1) + \\
&\quad 1 + n_1 + c \cdot f(n_0 - 1, n_1, n_2 - 1) + \mathcal{R}_{n_1}(n_0 - 2, n_2) \\
&\qquad\qquad \text{(by the induction hypothesis)} \\
&\leq \sum_{i=0}^{n_0 - 1} (1 + n_1 + c \cdot f(n_0 - i, n_1, n_2 - 1)) + \mathcal{R}_{n_1}(0, n_2) \\
&\qquad\qquad \text{(by performing these steps repeatedly)}
\end{aligned}
$$

So overall, we obtain

$$
\begin{aligned}
\mathcal{R}_{n_1}(n_0, n_2) &\leq \sum_{i=0}^{n_0 - 1} (1 + n_1 + c \cdot f(n_0 - i, n_1, n_2 - 1)) + \mathcal{R}_{n_1}(0, n_2) \qquad \text{(by (3))} \\
&= n_1 + \sum_{i=0}^{n_0 - 1} (1 + n_1 + c \cdot f(n_0 - i, n_1, n_2 - 1)) \\
&\leq n_1 + n_0 \cdot (1 + n_1 + c \cdot f(n_0, n_1, n_2 - 1)) \\
&\qquad\qquad \text{(as } f(n_0 - i, n_1, n_2 - 1) \leq f(n_0, n_1, n_2 - 1)\text{)} \\
&= n_1 + n_0 \cdot (1 + n_1 \cdot (1 + c)) + (n_2 - 1) \cdot (1 + n_1 \cdot (1 + c)) \cdot (c \cdot n_0)^{n_2} \\
&\leq f(n_0, n_1, n_2). \qquad (4)
\end{aligned}
$$

Here, (4) holds as $n_0 \cdot (1 + n_1 \cdot (1 + c)) \leq (1 + n_1 \cdot (1 + c)) \cdot (c \cdot n_0)^{n_2}$ for $n_2 > 0$ and $c > 0$. □

Theorem 21 (Lifting Local Runtime Bounds). *Let $\mathcal{RB}$ be a global runtime bound, $\mathcal{SB}$ be a size bound, and $\varnothing \neq \mathcal{T}'_> \subseteq \mathcal{T}' \subseteq \mathcal{T} \setminus \mathcal{T}_0$. Moreover, let $\mathcal{RB}_{\text{loc}}^{\mathcal{T}'_>, \mathcal{T}'}$ be a local runtime bound for $\mathcal{T}'_>$ w.r.t. $\mathcal{T}'$. Then $\mathcal{RB}'$ is also a global runtime bound, where $\mathcal{RB}'(t) = \mathcal{RB}(t)$ for all $t \in \mathcal{T} \setminus \mathcal{T}'_>$ and for $t \in \mathcal{T}'_>$, we have:*

$$
\begin{aligned}
\mathcal{RB}'(t) = {} & \textstyle\sum_{r \in \mathcal{ET}_{\mathcal{T}'}} \mathcal{RB}(r) \cdot \mathcal{RB}_{\text{loc}}^{\mathcal{T}'_>}(\to_r \mathcal{T}')\,[v/\mathcal{SB}(r, v) \mid v \in \mathcal{PV}] \\
+ & \textstyle\sum_{r \in \mathcal{EF}_{\mathcal{T}'}} \sum_{\rho \in \text{fun}(r) \cap \mathcal{F}_{\mathcal{T}'}} \mathcal{RB}(r) \cdot \mathcal{RB}_{\text{loc}}^{\mathcal{T}'_>}(\to_\rho \mathcal{T}')\,[v/\mathcal{SB}(\rho, v) \mid v \in \mathcal{PV}]
\end{aligned}
$$

Proof. We show that for all $t \in \mathcal{T}$, all $\sigma_0 \in \Sigma$, and all trees $\mathbb{T}$ with $\mathbb{T}_{\sigma_0} \prec^* \mathbb{T}$, we have

$$[\![\mathcal{RB}'(t)]\!]_{|\sigma_0|} \geq |\mathbb{T}|_{\{t\}}.$$

The case $t \in \mathcal{T} \setminus \mathcal{T}'_>$ is trivial, since $\mathcal{RB}'(t) = \mathcal{RB}(t)$ and $\mathcal{RB}$ is a runtime bound. For $t \in \mathcal{T}'_>$, we have to show that

$$[\![\mathcal{RB}'(t)]\!]_{|\sigma_0|} = \left[\!\!\left[\sum_{r \in \mathcal{ET}_{\mathcal{T}'}} \mathcal{RB}(r) \cdot \mathcal{RB}_{\text{loc}}^{\mathcal{T}'_>}(\to_r \mathcal{T}')[v/\mathcal{SB}(r, v) \mid v \in \mathcal{PV}] \right]\!\!\right]$$

$$+ \sum_{r \in \mathcal{EF}_{\mathcal{T}'}} \sum_{\rho \in \mathrm{fun}(r) \cap \mathcal{F}_{\mathcal{T}'}} \mathcal{RB}(r) \cdot \mathcal{RB}^{\mathcal{T}'_>}_{\mathrm{loc}}(\to_\rho \mathcal{T}')[v/\mathcal{SB}(\rho, v) \mid v \in \mathcal{PV}] \Bigg]\!\Bigg]_{|\sigma_0|}$$
$$\geq |\mathbb{T}|_{\{t\}}.$$

Let $\mathbb{T}_1, \ldots, \mathbb{T}_m$ be the maximal subtrees of $\mathbb{T}$ where all edges are labeled with transitions from $\mathcal{T}'$ or function calls from $\mathrm{fun}(\mathcal{T}')$. So the subtrees $\mathbb{T}_1, \ldots, \mathbb{T}_m$ are constructed by only using $\prec_{\mathcal{T}' \cup \{\varepsilon\}}$-steps, i.e., if $\mathbb{T}_i$'s root node is labeled with $(\widetilde{\ell}_i, \tilde{\sigma}_i)$, then we have $(\{(\widetilde{\ell}_i, \tilde{\sigma}_i)\}, \varnothing) \prec^*_{\mathcal{T}' \cup \{\varepsilon\}} \mathbb{T}_i$. Let $\tau_i \in \mathcal{T} \cup \mathcal{F}$ be the transition or the function call that the edge to $(\widetilde{\ell}_i, \tilde{\sigma}_i)$ is labeled with in $\mathbb{T}$, i.e., τ_i starts the evaluation of the subprogram $\mathcal{T}'$. Let k_i be the number of edges labeled with t in $\mathbb{T}_i$ and let $\mathbb{T}$ have k edges labeled with t, i.e., $|\mathbb{T}_i|_{\{t\}} = k_i$ and $|\mathbb{T}|_{\{t\}} = k$. Then we have $\sum_{i=1}^{m} k_i = k$.

As $\mathcal{SB}$ is a size bound, we have $[\![\mathcal{SB}(\tau_i, v)]\!]_{|\sigma_0|} \geq |\tilde{\sigma}_i(v)|$ for all $v \in \mathcal{PV}$. Hence, by the definition of local runtime bounds and as bounds are weakly monotonically increasing functions, we can conclude that

$$[\![\mathcal{RB}^{\mathcal{T}'_>}_{\mathrm{loc}}(\to_{\tau_i} \mathcal{T}')\,[v/\mathcal{SB}(\tau_i, v) \mid v \in \mathcal{PV}]]\!]_{|\sigma_0|} \;\geq\; [\![\mathcal{RB}^{\mathcal{T}'_>}_{\mathrm{loc}}(\to_{\tau_i} \mathcal{T}')]\!]_{|\tilde{\sigma}_i|} \;\geq\; k_i. \quad (5)$$

Finally, we analyze how many maximal $\mathcal{T}'$-subtrees can be reached via some $\tau \in \mathcal{T} \cup \mathcal{F}$ in the full tree $\mathbb{T}$. Every entry transition $\tau_i = r \in \mathcal{ET}_{\mathcal{T}'}$ can occur at most $[\![\mathcal{RB}(r)]\!]_{|\sigma_0|}$ times in the tree $\mathbb{T}$, as $\mathcal{RB}$ is a global runtime bound. Similarly, every function call $\tau_i = \rho \in \mathrm{fun}(r) \cap \mathcal{F}_{\mathcal{T}'}$ for an $r \in \mathcal{EF}_{\mathcal{T}'}$ can occur at most $[\![\mathcal{RB}(r)]\!]_{|\sigma_0|}$ times in the tree $\mathbb{T}$. Thus, we have

$$\begin{aligned}
[\![\mathcal{RB}'(t)]\!]_{|\sigma_0|} = \Bigg[\!\Bigg[& \sum_{r \in \mathcal{ET}_{\mathcal{T}'}} \mathcal{RB}(r) \cdot \mathcal{RB}^{\mathcal{T}'_>}_{\mathrm{loc}}(\to_r \mathcal{T}')[v/\mathcal{SB}(r, v) \mid v \in \mathcal{PV}] \\
& + \sum_{r \in \mathcal{EF}_{\mathcal{T}'}} \sum_{\rho \in \mathrm{fun}(r) \cap \mathcal{F}_{\mathcal{T}'}} \mathcal{RB}(r) \cdot \mathcal{RB}^{\mathcal{T}'_>}_{\mathrm{loc}}(\to_\rho \mathcal{T}')[v/\mathcal{SB}(\rho, v) \mid v \in \mathcal{PV}] \Bigg]\!\Bigg]_{|\sigma_0|} \\
& \geq \sum_{i=1}^{m} [\![\mathcal{RB}^{\mathcal{T}'_>}_{\mathrm{loc}}(\to_{\tau_i} \mathcal{T}')\,[v/\mathcal{SB}(\tau_i, v) \mid v \in \mathcal{PV}]]\!]_{|\sigma_0|} \\
& \geq \sum_{i=1}^{m} k_i \qquad \text{(by (5))} \\
& = k \\
& = |\mathbb{T}|_{\{t\}}.
\end{aligned}$$

□

Theorem 26 (Size Bounds for Trivial SCCs Without Function Calls). *Let $\mathcal{SB}$ be a size bound and $\{\langle \tau, x \rangle\}$ be a trivial SCC of the RVG such that $\tau \in \mathcal{F}$ or $\mathrm{fun}(\eta(x)) = \varnothing$ for the update η of $\tau \in \mathcal{T}$. Then $\mathcal{SB}'$ is also a size bound where $\mathcal{SB}'(\alpha) = \mathcal{SB}(\alpha)$ for all $\alpha \neq \langle \tau, x \rangle$, and for $\alpha = \langle \tau, x \rangle$ we have*

$$\mathcal{SB}'(\alpha) = \begin{cases} \mathcal{SB}_{\mathrm{loc}}(\alpha), & \text{if } \mathrm{pre}(\tau) = \varnothing \\ \max_{\tau' \in \mathrm{pre}(\tau)} \{\mathcal{SB}_{\mathrm{loc}}(\alpha)\,[v/\mathcal{SB}(\tau', v) \mid v \in \mathcal{PV}]\}, & \text{otherwise} \end{cases}$$

Proof. Let $\{\langle \tau, x\rangle\}$ be a trivial SCC such that $\tau \in \mathcal{F}$ or $\mathrm{fun}(\eta(x)) = \varnothing$ for the update η of $\tau \in \mathcal{T}$. Moreover, let $\sigma_0 \in \Sigma$ and $\mathbb{T}_{\sigma_0} \prec^* \mathbb{T}$ such that $\mathbb{T}$ contains a path $(\ell_0, \sigma_0) \to \cdots \to_\tau (\_, \sigma)$ with $\sigma \neq \bot$. We have to prove that

$$|\sigma|(x) \;\leq\; [\![\mathcal{SB}'(\tau, x)]\!]_{|\sigma_0|}.$$

We first consider the case $\mathrm{pre}(\tau) = \varnothing$ (i.e., τ is an initial transition from $\mathcal{T}_0$). Note that by our definition of ρ-ITSs, ℓ_0 can neither be the target location of a transition nor evaluated after a function call. Thus, the path of the tree $\mathbb{T}$ has the form $(\ell_0, \sigma_0) \to_\tau (\_, \sigma)$. Hence, we have $[\![\mathcal{SB}'(\tau, x)]\!]_{|\sigma_0|} = [\![\mathcal{SB}_{\mathrm{loc}}(\tau, x)]\!]_{|\sigma_0|} \geq |\sigma|(x)$. Note that w.l.o.g., $\mathcal{SB}_{\mathrm{loc}}(\tau, x) \in \mathbb{Z}[\mathcal{PV}]$ by the requirement $\mathrm{fun}(\eta(x)) = \varnothing$ for the update η of the transition τ.

Otherwise, if $\mathrm{pre}(\tau) \neq \varnothing$, then the path in $\mathbb{T}$ has the form $(\ell_0, \sigma_0) \to \cdots \to_{\tilde{\tau}} (\_, \tilde{\sigma}) \to_\tau (\_, \sigma)$ for some $\tilde{\tau} \in \mathrm{pre}(\tau)$. By the definition of size bounds, we have $[\![\mathcal{SB}(\tilde{\tau}, v)]\!]_{|\sigma_0|} \geq |\tilde{\sigma}|(v)$ for all $v \in \mathcal{PV}$. Thus, we obtain

$$\begin{aligned}
[\![\mathcal{SB}'(\tau, x)]\!]_{|\sigma_0|} &= [\![\max_{\tau' \in \mathrm{pre}(\tau)} \{\mathcal{SB}_{\mathrm{loc}}(\alpha)\,[v/\mathcal{SB}(\tau', v) \mid v \in \mathcal{PV}]\}]\!]_{|\sigma_0|} \\
&\geq [\![\mathcal{SB}_{\mathrm{loc}}(\alpha)\,[v/\mathcal{SB}(\tilde{\tau}, v) \mid v \in \mathcal{PV}]]\!]_{|\sigma_0|} \\
&\geq [\![\mathcal{SB}_{\mathrm{loc}}(\alpha)]\!]_{|\tilde{\sigma}|} \\
&\geq |\sigma|(x)
\end{aligned}$$

□

Theorem 28 (Size Bounds for Trivial SCCs With Function Calls). *Let $\mathcal{SB}$ be a size bound and $\{\langle t, x\rangle\}$ be a trivial SCC of the RVG such that $t \in \mathcal{T}$ and $\mathrm{fun}(\eta(x)) \neq \varnothing$. Then $\mathcal{SB}'$ is also a size bound where $\mathcal{SB}'(\alpha) = \mathcal{SB}(\alpha)$ for $\alpha \neq \langle t, x\rangle$, and for $\alpha = \langle t, x\rangle$ with $\mathrm{fun}(\eta(x)) = \{\rho_1, \ldots, \rho_n\}$ and $\rho_i = \ell_i(v_i|\_)$, we have*

$$\mathcal{SB}'(\alpha) = \begin{cases} 0, & \text{if } \mathrm{pre}^\Omega(t, \rho_i) = \varnothing \text{ for some } i \in [n] \\ \max_{t'_i \in \mathrm{pre}^\Omega(t,\rho_i) \text{ for all } i \in [n]} \{\mathcal{SB}_{\mathrm{loc}}(\alpha)\,[\rho_i/\mathcal{SB}(t'_i, v_i) \mid i \in [n]]\}, & \text{if all } \mathrm{pre}^\Omega(t, \rho_i) \neq \varnothing \text{ and } \mathrm{pre}(t) = \varnothing \\ \max\limits_{\substack{\tau' \in \mathrm{pre}(t) \\ t'_i \in \mathrm{pre}^\Omega(t,\rho_i) \text{ for all } i \in [n]}} \{\mathcal{SB}_{\mathrm{loc}}(\alpha)\,[v/\mathcal{SB}(\tau', v) \,|\, v \in \mathcal{PV}]\,[\rho_i/\mathcal{SB}(t'_i, v_i) \,|\, i \in [n]]\}, & \text{otherwise} \end{cases}$$

Proof. Let $\{\langle t, x\rangle\}$ be a trivial SCC of the RVG such that $t \in \mathcal{T}$ and $\mathrm{fun}(\eta(x)) = \{\rho_1, \ldots, \rho_n\} \neq \varnothing$ for the update η of t. Moreover, let $\sigma_0 \in \Sigma$ and $\mathbb{T}_{\sigma_0} \prec^* \mathbb{T}$ such that $\mathbb{T}$ contains a path $(\ell_0, \sigma_0) \to \cdots \to_\tau (\_, \sigma)$ with $\sigma \neq \bot$. We have to prove that

$$|\sigma|(x) \;\leq\; [\![\mathcal{SB}'(\tau, x)]\!]_{|\sigma_0|}.$$

If $\mathrm{pre}^\Omega(t, \rho_i) = \varnothing$ for some $i \in [n]$, then there are only evaluations $(\ell_0, \sigma_0) \to \cdots \to_\tau (\_, \sigma)$ where $\sigma = \bot$. Hence, let $\mathrm{pre}^\Omega(t, \rho_i) \neq \varnothing$ for all $i \in [n]$.

We first consider the case $\mathrm{pre}(\tau) = \varnothing$ (i.e., τ is an initial transition from $\mathcal{T}_0$). Again, by our definition of ρ-ITSs, ℓ_0 can neither be the target location of a transition nor evaluated after a function call. Thus, the path of the tree $\mathbb{T}$ has

the form $(\ell_0, \sigma_0) \to_t (\_, \sigma)$. At the same time, $\mathbb{T}$ also contains paths $(\ell_0, \sigma_0) \to_{\rho_i} (\ell_i, \_) \to \cdots \to_{t'_i} (\ell'_i, \sigma'_i)$ for all $i \in [n]$ where $\ell'_i \in \Omega$ since $\sigma \neq \bot$. Hence, $t'_i \in \mathrm{pre}^{\Omega}(t, \rho_i)$. By the definition of size bounds, we have $[\![\mathcal{SB}(t'_i, v_i)]\!]_{|\sigma_0|} \geq |\sigma'_i|(v_i)$. Hence, we obtain

$$\begin{aligned}[\![\mathcal{SB}'(t,x)]\!]_{|\sigma_0|} &\geq [\![\mathcal{SB}_{\mathrm{loc}}(t,x)\,[\rho_i/\mathcal{SB}(t'_i,v_i) \mid i \in [n]]]\!]_{|\sigma_0|}\\ &\geq [\![\mathcal{SB}_{\mathrm{loc}}(t,x)\,[\rho_i/|\sigma'_i|(v_i) \mid i \in [n]]]\!]_{|\sigma_0|}\\ &\geq |\sigma|(x).\end{aligned}$$

Otherwise, if $\mathrm{pre}(t) \neq \varnothing$, then the path in $\mathbb{T}$ has the form $(\ell_0, \sigma_0) \to \cdots \to_{\tilde{\tau}} (\tilde{\ell}, \tilde{\sigma}) \to_t (\_, \sigma)$ for some $\tilde{\tau} \in \mathrm{pre}(t)$. At the same time, $\mathbb{T}$ also contains paths $(\ell_0, \sigma_0) \to \cdots \to_{\tilde{\tau}} (\tilde{\ell}, \tilde{\sigma}) \to_{\rho_i} (\ell_i, \_) \to \cdots \to_{t'_i} (\ell'_i, \sigma'_i)$ for all $i \in [n]$ where $\ell'_i \in \Omega$ since $\sigma \neq \bot$. By the definition of size bounds, we have $[\![\mathcal{SB}(\tilde{\tau}, v)]\!]_{|\sigma_0|} \geq |\tilde{\sigma}|(v)$ for all $v \in \mathcal{PV}$ and $[\![\mathcal{SB}(t'_i, v_i)]\!]_{|\sigma_0|} \geq |\sigma'_i|(v_i)$ for all $i \in [n]$. Thus, for $\alpha = \langle t, x\rangle$ we obtain

$$\begin{aligned}[\![\mathcal{SB}'(\alpha)]\!]_{|\sigma_0|} &\geq [\![\mathcal{SB}_{\mathrm{loc}}(\alpha)\,[v/\mathcal{SB}(\tilde{\tau},v) \mid v \in \mathcal{PV}]\,[\rho_i/\mathcal{SB}(t'_i,v) \mid i \in [n]]]\!]_{|\sigma_0|}\\ &= [\![\mathcal{SB}_{\mathrm{loc}}(\alpha)\,[v/\mathcal{SB}(\tilde{\tau},v) \mid v \in \mathcal{PV}]\,[\rho_i/[\![\mathcal{SB}(t'_i,v)]\!]_{|\sigma_0|} \mid i \in [n]]]\!]_{|\sigma_0|}\\ &\geq [\![\mathcal{SB}_{\mathrm{loc}}(\alpha)\,[\rho_i/|\sigma'_i| \mid i \in [n]]]\!]_{|\tilde{\sigma}|}\\ &\geq |\sigma|(x).\end{aligned}$$

□

Theorem 30 (Size Bounds for Non-Trivial SCCs). *Let $\mathcal{SB}$ be a size bound and C be a non-trivial SCC in a RVG, where for all $\alpha \in C$, $\mathcal{SB}_{\mathrm{loc}}(\alpha)$ satisfies (1) for suitable e_α and s_α. Then $\mathcal{SB}'$ is also a size bound where $\mathcal{SB}'(\alpha) = \mathcal{SB}(\alpha)$ for all $\alpha \in \mathcal{RV} \setminus C$, and $\mathcal{SB}'(\alpha) = \mathcal{SB}'(C)$ for all $\alpha \in C$, where*

$$\mathcal{SB}'(C) = \textstyle\prod_{\alpha \in C} \mathit{scale}(\alpha) \cdot (\sum_{\alpha \in C}(\mathit{add}(\alpha) + \sum_{v \in V_\alpha} \mathit{init}_\alpha(v) + \sum_{v \in F_\alpha} \mathit{init}^{\Omega}_\alpha(v)))$$

Proof. Let C be a non-trivial SCC of the RVG, let $\sigma_0 \in \Sigma$, and $\mathbb{T}_{\sigma_0} \prec^* \mathbb{T}$ such that $\mathbb{T}$ contains a path $(\ell_0, \sigma_0) \to \cdots \to_\tau (\_, \sigma)$ with $\sigma \neq \bot$. We have to prove that

$$|\sigma|(v) \leq [\![\mathcal{SB}'(\tau, x)]\!]_{|\sigma_0|}.$$

If $\langle \tau, x\rangle \notin C$, then $\mathcal{SB}'(\tau, x) = \mathcal{SB}(\tau, x)$ is a size bound by definition.

So let us consider $\alpha = \langle \tau, x\rangle \in C$. Note that τ cannot be an initial transition as there are no transitions or function calls leading back to the initial location ℓ_0 (i.e., then C would not be a non-trivial SCC). Hence, there exists a predecessor $\tilde{\tau}$ of τ in the path, i.e., the path has the form

$$(\ell_0, \sigma_0) \to \cdots \to_{\tilde{\tau}} (\tilde{\ell}, \tilde{\sigma}) \to_\tau (\ell, \sigma).$$

Note that we must have $|V_\alpha| + |F_\alpha| > 0$ for all $\alpha \in C$ as C is a non-trivial SCC of the RVG. Thus, as $[\![\max_{\tau' \in \mathrm{pre}(\tau)} \{1,\, s_\alpha\,[v/\mathcal{SB}(\tau', v) \mid v \in \mathcal{PV}]\}]\!]_{|\sigma_0|} \geq 1$ for all $\sigma_0 \in \Sigma$ and $\alpha \in C$, we also have

$$[\![\mathit{scale}(\alpha)]\!]_{|\sigma_0|} \geq 1 \text{ for all } \sigma_0 \in \Sigma \text{ and } \alpha \in C. \tag{6}$$

We prove our claim by induction on the number $|\mathbb{T}|_{\mathcal{I}}$ where $\mathcal{I} = \{\tau \mid \langle \tau, v\rangle \in C\}$.

Induction Base: Here, we have $|\mathbb{T}|_{\mathcal{I}} = 1$ and thus $\langle \tilde{\tau}, v\rangle \notin C$ for all $v \in \mathcal{PV}$. Note that we have

$$\begin{aligned}[\![init_\alpha(v)]\!]_{|\sigma_0|} &= [\![\max\{\mathcal{SB}(\tau, v) \mid \exists \tau \in \mathcal{T} \cup \mathcal{F}.\, \langle \tau, v\rangle \in \mathrm{pre}(\alpha) \setminus C\}]\!]_{|\sigma_0|} \\ &\geq [\![\mathcal{SB}(\tilde{\tau}, v)]\!]_{|\sigma_0|} \\ &\geq |\sigma(v)| \end{aligned} \tag{7}$$

for all $v \in \mathrm{actV}(\mathcal{SB}_{\mathrm{loc}}(\alpha))$ as $\langle \tilde{\tau}, v\rangle \notin C$. We extend $\mathrm{fun}(\cdot)$ to function calls by defining $\mathrm{fun}(\rho) = \varnothing$ for all $\rho \in \mathcal{F}$. Then for all function calls $\rho_i \in \mathrm{fun}(\tau)$, there is a path $(\ell_0, \sigma_0) \to \cdots \to (\tilde{\ell}, \tilde{\sigma}) \to_{\rho_i} \cdots \to_{t_i} (\widetilde{\ell_i}, \widetilde{\sigma_i})$ in $\mathbb{T}$ such that $\tilde{\ell_i} \in \Omega$. We have

$$\begin{aligned}[\![init^{\Omega}_\alpha(v_i)]\!]_{|\sigma_0|} &= [\![\max\{\mathcal{SB}(t, v_i) \mid \exists t \in \mathcal{T}.\, \langle t, v_i\rangle \in \mathrm{pre}^{\Omega}(\alpha) \setminus C\}]\!]_{|\sigma_0|} \\ &\geq [\![\mathcal{SB}(t_i, v_i)]\!]_{|\sigma_0|} \\ &\geq |\tilde{\sigma}_i(v_i)| \end{aligned} \tag{8}$$

for all $\rho_i = \ell_i(v_i|\_) \in \mathrm{actF}(\mathcal{SB}_{\mathrm{loc}}(\alpha))$ as $\langle t_i, v_i\rangle \notin C$ since $|\mathbb{T}|_{\mathcal{I}} = 1$. Thus, we have

$$[\![\mathcal{SB}'(\tau, x)]\!]_{|\sigma_0|} \geq \left[\!\!\left[scale(\alpha) \cdot \left(add(\alpha) + \sum_{v \in V_\alpha} init_\alpha(v) + \sum_{v \in F_\alpha} init^{\Omega}_\alpha(v) \right) \right]\!\!\right]_{|\sigma_0|}$$

(by (6))

$$\geq [\![s_\alpha\,[v/\mathcal{SB}(\tilde{\tau}, v) \mid v \in \mathcal{PV}]]\!]_{|\sigma_0|} \cdot \left(e_\alpha + \sum_{\substack{v \in \mathrm{actV}(\mathcal{SB}_{\mathrm{loc}}(\alpha)) \\ v \notin \mathrm{actV}(s_\alpha)}} [\![init_\alpha(v)]\!]_{|\sigma_0|} + \sum_{\ell_i(v_i|\cdot) \in \mathrm{actF}(\mathcal{SB}_{\mathrm{loc}}(\alpha))} [\![init^{\Omega}_\alpha(v_i)]\!]_{|\sigma_0|} \right)$$

$$\geq [\![s_\alpha]\!]_{|\tilde{\sigma}|} \cdot \left(e_\alpha + \sum_{\substack{v \in \mathrm{actV}(\mathcal{SB}_{\mathrm{loc}}(\alpha)) \\ v \notin \mathrm{actV}(s_\alpha)}} |\tilde{\sigma}(v)| + \sum_{\rho_i = \ell_i(v_i|\cdot) \in \mathrm{actF}(\mathcal{SB}_{\mathrm{loc}}(\alpha))} |\tilde{\sigma}_i(v_i)| \right)$$

(by (7), (8), and as $[\![\mathcal{SB}(\tilde{\tau}, v)]\!]_{|\sigma_0|} \geq |\tilde{\sigma}|(v)$ for all $v \in \mathcal{PV}$)

$$\geq [\![\mathcal{SB}_{\mathrm{loc}}(\tau, x)\,[\rho_i/|\tilde{\sigma}_i|(v_i) \mid \rho_i \in \mathrm{fun}(\tau)]]\!]_{|\tilde{\sigma}|} \qquad \text{(by (1))}$$

$$\geq |\sigma|(x).$$

Induction Step: We have $|\mathbb{T}|_{\{t\}} \leq [\![\mathcal{RB}(t)]\!]_{|\sigma_0|}$ for all $t \in \mathcal{T}$ and $|\mathbb{T}|_{\{\rho\}} \leq [\![\sum_{t \in \mathrm{fun}^{-1}(\rho)} \mathcal{RB}(t)]\!]_{|\sigma_0|}$ for all $\rho \in \mathcal{F}$. Now for any $\overline{\alpha} = \langle \overline{\tau}, \_\rangle$, we define the

following expressions:

$$add(\overline{\alpha}, \mathbb{T}) = |\mathbb{T}|_{\{\overline{\tau}\}} \cdot \left(e_{\overline{\alpha}} + \sum_{\substack{v \in \mathrm{actV}(\mathcal{SB}_{\mathrm{loc}}(\overline{\alpha})) \\ v \notin \mathrm{actV}(s_{\overline{\alpha}}) \cup V_{\overline{\alpha}}}} init_{\overline{\alpha}}(v) + \sum_{\substack{\ell(v|\cdot) \in \mathrm{actF}(\mathcal{SB}_{\mathrm{loc}}(\overline{\alpha})) \\ v \notin F_{\overline{\alpha}}}} init^{\Omega}_{\overline{\alpha}}(v) \right)$$

$$scale(\overline{\alpha}, \mathbb{T}) = (\max_{\tau' \in \mathrm{pre}(\overline{\tau})} \{1,\, s_{\overline{\alpha}}\,[v/\mathcal{SB}(\tau', v) \mid v \in \mathcal{PV}]\} \cdot (|V_{\overline{\alpha}}| + |F_{\overline{\alpha}}|))^{|\mathbb{T}|_{\{\overline{\tau}\}}}$$

So compared to $add(\alpha)$ and $scale(\alpha)$, we have replaced the over-approximation of the runtime bound rb_α by concrete values. Let $\mathbb{T}'$ be the tree which results from $\mathbb{T}$ by removing (ℓ, σ). Then we define

$$\Psi = \prod_{\overline{\alpha} \in C_\tau} scale(\overline{\alpha}, \mathbb{T}') \cdot \prod_{\overline{\alpha} \in C \setminus C_\tau} scale(\overline{\alpha}, \mathbb{T}) \cdot \left(\sum_{\overline{\alpha} \in C_\tau} add(\overline{\alpha}, \mathbb{T}') + \sum_{\overline{\alpha} \in C \setminus C_\tau} add(\overline{\alpha}, \mathbb{T}) + \sum_{\overline{\alpha} \in C} \left(\sum_{v \in V_{\overline{\alpha}}} init_{\overline{\alpha}}(v) + \sum_{v \in F_{\overline{\alpha}}} init^{\Omega}_{\overline{\alpha}}(v) \right) \right)$$

where $C_\tau = \{\langle \tau, v\rangle \in C \mid v \in \mathcal{PV}\}$, i.e., C_τ contains all result variables of C that have α's transition or function call τ. For all $v \in V_\alpha \cup F_\alpha$, we have $[\![\Psi]\!]_{|\sigma_0|} \geq |\tilde{\sigma}(v)|$ by the induction hypothesis. Furthermore, for $v \in \mathrm{actV}(\mathcal{SB}_{\mathrm{loc}}(\alpha)) \setminus V_\alpha$, we have $\langle \tilde{\tau}, v\rangle \notin C$ and $\langle \tilde{\tau}, v\rangle \in \mathrm{pre}(\alpha)$, and thus

$$\begin{aligned} [\![init_\alpha(v)]\!]_{|\sigma_0|} &= [\![\max\{\mathcal{SB}(\tau, v) \mid \exists \tau \in \mathcal{T} \cup \mathcal{F}.\, \langle \tau, v\rangle \in \mathrm{pre}(\alpha) \setminus C\}]\!]_{|\sigma_0|} \\ &\geq [\![\mathcal{SB}(\tilde{\tau}, v)]\!]_{|\sigma_0|} \\ &\geq |\tilde{\sigma}(v)|. \end{aligned} \tag{9}$$

Again, there is a path $(\ell_0, \sigma_0) \to \cdots \to (\tilde{\ell}, \tilde{\sigma}) \to_{\rho_i} \cdots \to_{t_i} (\tilde{\ell_i}, \tilde{\sigma_i})$ in $\mathbb{T}$ for each function call $\rho_i \in \mathrm{fun}(\tau)$ such that $\tilde{\ell_i} \in \Omega$. Similarly, for all $v_i \in \mathcal{PV} \setminus F_\alpha$ with $\ell_i(v_i|\_) \in \mathrm{actF}(\mathcal{SB}_{\mathrm{loc}}(\alpha))$, we have $\langle t_i, v_i\rangle \notin C$ and $\langle t_i, v_i\rangle \in \mathrm{pre}^{\Omega}(\alpha)$. Thus,

$$\begin{aligned} [\![init^{\Omega}_\alpha(v_i)]\!]_{|\sigma_0|} &= [\![\max\{\mathcal{SB}(t, v_i) \mid \exists t \in \mathcal{T}.\, \langle t, v_i\rangle \in \mathrm{pre}^{\Omega}(\alpha) \setminus C\}]\!]_{|\sigma_0|} \\ &\geq [\![\mathcal{SB}(t_i, v_i)]\!]_{|\sigma_0|} \\ &\geq |\tilde{\sigma}_i(v_i)|. \end{aligned} \tag{10}$$

Finally, we define the following expression for all $\overline{\alpha} \in C$:

$$\Phi_{\overline{\alpha}} = e_{\overline{\alpha}} + \sum_{\substack{v \in \mathrm{actV}(\mathcal{SB}_{\mathrm{loc}}(\overline{\alpha})) \\ v \notin \mathrm{actV}(s_{\overline{\alpha}}) \cup V_{\overline{\alpha}}}} init_{\overline{\alpha}}(v) + \sum_{\substack{\ell(v|\cdot) \in \mathrm{actF}(\mathcal{SB}_{\mathrm{loc}}(\overline{\alpha})) \\ v \notin F_{\overline{\alpha}}}} init^{\Omega}_{\overline{\alpha}}(v).$$

To simplify the presentation, for $\overline{\alpha} = \langle \overline{\tau}, \_\rangle$, let

$$\widehat{s_{\overline{\alpha}}} = \max_{\tau' \in \mathrm{pre}(\overline{\tau})} \{1,\, s_{\overline{\alpha}}\,[v/\mathcal{SB}(\tau', v) \mid v \in \mathcal{PV}]\}\,.$$

In particular, the following holds:

$$[\![\widehat{s}_\alpha]\!]_{|\sigma_0|} \geq [\![s_\alpha\,[v/\mathcal{SB}(\tilde{\tau},v) \mid v \in \mathcal{PV}]]\!]_{|\sigma_0|} \geq [\![s_\alpha]\!]_{|\tilde{\sigma}|}. \tag{11}$$

Thus, we now have

$$\begin{aligned}
[\![\mathcal{SB}'(\tau,x)]\!]_{|\sigma_0|} &= \left[\!\!\left[\prod_{\overline{\alpha}\in C} scale(\overline{\alpha}) \cdot \left(\sum_{\overline{\alpha}\in C} \left(add(\overline{\alpha}) + \overbrace{\sum_{v\in V_{\overline{\alpha}}} init_{\overline{\alpha}}(v) + \sum_{v\in F_{\overline{\alpha}}} init^{\Omega}_{\overline{\alpha}}(v)}^{init_{\overline{\alpha}}} \right) \right) \right]\!\!\right]_{|\sigma_0|} \\
&\geq \left[\!\!\left[\prod_{\overline{\alpha}\in C_{\tilde{\tau}}} \widehat{s}_{\overline{\alpha}} \cdot (|V_\alpha| + |F_\alpha|) \cdot \prod_{\overline{\alpha}\in C_{\tilde{\tau}}} scale(\overline{\alpha},\mathbb{T}') \cdot \prod_{\overline{\alpha}\in C\setminus C_{\tilde{\tau}}} scale(\overline{\alpha},\mathbb{T}) \right.\right. \\
&\qquad \left.\left. \cdot \left(\sum_{\overline{\alpha}\in C_{\tilde{\tau}}} add(\overline{\alpha},\mathbb{T}') + \sum_{\overline{\alpha}\in C_{\tilde{\tau}}} \Phi_{\overline{\alpha}} + \sum_{\overline{\alpha}\in C\setminus C_{\tilde{\tau}}} add(\overline{\alpha},\mathbb{T}) + \sum_{\overline{\alpha}\in C} init_{\overline{\alpha}} \right) \right]\!\!\right]_{|\sigma_0|} \\
&\qquad\qquad \text{(extract last evaluation step)} \\
&= \left[\!\!\left[\prod_{\overline{\alpha}\in C_{\tilde{\tau}}} \widehat{s}_{\overline{\alpha}} \cdot (|V_{\overline{\alpha}}| + |F_{\overline{\alpha}}|) \cdot \Psi + \prod_{\overline{\alpha}\in C_{\tilde{\tau}}} \widehat{s}_{\overline{\alpha}} \cdot (|V_{\overline{\alpha}}| + |F_{\overline{\alpha}}|) \right.\right. \\
&\qquad \left.\left. \cdot \prod_{\overline{\alpha}\in C_{\tilde{\tau}}} scale(\overline{\alpha},\mathbb{T}') \cdot \prod_{\overline{\alpha}\in C\setminus C_{\tilde{\tau}}} scale(\overline{\alpha},\mathbb{T}) \cdot \left(\sum_{\overline{\alpha}\in C_{\tilde{\tau}}} \Phi_{\overline{\alpha}} \right) \right]\!\!\right]_{|\sigma_0|} \\
&\qquad\qquad \text{(by definition of } \Psi) \\
&\geq \left[\!\!\left[\prod_{\overline{\alpha}\in C_{\tilde{\tau}}} \widehat{s}_{\overline{\alpha}} \cdot (|V_{\overline{\alpha}}| + |F_{\overline{\alpha}}|) \cdot \left(\Psi + \sum_{\overline{\alpha}\in C_{\tilde{\tau}}} \Phi_{\overline{\alpha}} \right) \right]\!\!\right]_{|\sigma_0|} \qquad \text{(by (6))} \\
&\geq [\![\widehat{s}_\alpha]\!]_{|\sigma_0|} \cdot (|V_\alpha| + |F_\alpha|) \cdot \left([\![\Phi_\alpha]\!]_{|\sigma_0|} + [\![\Psi]\!]_{|\sigma_0|} \right) \qquad \text{(by (6))} \\
&\geq [\![s_\alpha]\!]_{|\tilde{\sigma}|} \cdot (|V_\alpha| + |F_\alpha|) \cdot \left([\![\Phi_\alpha]\!]_{|\sigma_0|} + [\![\Psi]\!]_{|\sigma_0|} \right) \qquad \text{(by (11))} \\
&\geq [\![s_\alpha]\!]_{|\tilde{\sigma}|} \cdot \left([\![\Phi_\alpha]\!]_{|\sigma_0|} + (|V_\alpha| + |F_\alpha|) \cdot [\![\Psi]\!]_{|\sigma_0|} \right) \\
&\qquad\qquad \text{(as } |V_\alpha| + |F_\alpha| \geq 1) \\
&\geq [\![s_\alpha]\!]_{|\tilde{\sigma}|} \cdot \left([\![\Phi_\alpha]\!]_{|\sigma_0|} + \sum_{v\in V_\alpha} [\![\Psi]\!]_{|\sigma_0|} + \sum_{\substack{\ell(v|\cdot)\,\in\,\mathrm{actF}(\mathcal{SB}_{\mathrm{loc}}(\alpha)) \\ v\,\in\,F_\alpha}} [\![\Psi]\!]_{|\sigma_0|} \right) \\
&\geq [\![s_\alpha]\!]_{|\tilde{\sigma}|} \cdot \left([\![\Phi_\alpha]\!]_{|\sigma_0|} + \sum_{v\in V_\alpha} |\tilde{\sigma}(v)| + \sum_{\substack{\rho_i=\ell_i(v_i|\cdot)\,\in\,\mathrm{actF}(\mathcal{SB}_{\mathrm{loc}}(\alpha)) \\ v_i\,\in\,F_\alpha}} |\tilde{\sigma}_i(v_i)| \right) \\
&\qquad\qquad \text{(by the induction hypothesis)} \\
&= [\![s_\alpha]\!]_{|\tilde{\sigma}|} \cdot \left(e_\alpha + \sum_{v\in V_\alpha} |\tilde{\sigma}(v)| + \sum_{\substack{v\,\in\,\mathrm{actV}(\mathcal{SB}_{\mathrm{loc}}(\alpha)) \\ v\,\notin\,\mathrm{actV}(s_\alpha)\cup V_\alpha}} [\![init_\alpha(v)]\!]_{|\sigma_0|} \right.
\end{aligned}$$

$$+ \sum_{\substack{\rho_i = \ell_i(v_i\mid\cdot) \in \mathrm{actF}(\mathcal{SB}_{\mathrm{loc}}(\alpha)) \\ v_i \in F_\alpha}} |\tilde{\sigma}_i(v_i))| + \sum_{\substack{\rho_i = \ell_i(v_i\mid\cdot) \in \mathrm{actF}(\mathcal{SB}_{\mathrm{loc}}(\alpha)) \\ v_i \notin F_\alpha}} [\![init^{\Omega}_{\alpha}(v_i)]\!]_{|\sigma_0|} \Bigg)$$

(by definition of Φ_α)

$$\geq [\![s_\alpha]\!]_{|\tilde{\sigma}|} \cdot \left(e_\alpha + \sum_{\substack{v \in \mathrm{actV}(\mathcal{SB}_{\mathrm{loc}}(\alpha)) \\ v \notin \mathrm{actV}(s_\alpha)}} |\tilde{\sigma}(v)| + \sum_{\rho_i = \ell_i(v_i\mid\cdot) \in \mathrm{actF}(\mathcal{SB}_{\mathrm{loc}}(\alpha))} |\tilde{\sigma}_i(v_i)| \right)$$

(by (9) and (10))

$$\geq [\![\mathcal{SB}_{\mathrm{loc}}(\tau, x)\,[\rho_i / |\tilde{\sigma}_i|(v_i) \mid \rho_i \in \mathrm{fun}(\tau)]]\!]_{|\tilde{\sigma}|} \qquad \text{(by (1))}$$

$$\geq |\sigma|(x).$$

□

B Local Runtime Bounds via TWN-Loops

In this appendix we briefly recapitulate how to infer runtime bounds for so-called *triangular weakly non-linear loops* (*twn-loops*) based on our previous work [11, 12, 15, 16, 18]. This approach can be used to infer the local runtime bound $\mathcal{RB}^{\{t_3\}}_{\mathrm{loc}}(\to_{t_2} \{t_3\}) = \log_2(y) + 2$ for transition t_3 in our leading example of Fig. 2. This local runtime bound is needed in Ex. 22 to compute $\mathcal{RB}(t_3)$.

An example for a terminating twn-loop is:

$$\textbf{while } (x_2 - x_1 > 0 \wedge x_1 > 0) \textbf{ do } (x_1, x_2) \leftarrow (3 \cdot x_1,\, 2 \cdot x_2) \tag{12}$$

Note that this loop corresponds to transition t_3 ($x \cong x_1$ and $y \cong x_2$) with the additional invariant $x_1 > 0$. In practice, we use the tool Apron [14] to automatically infer such invariants. Formally, a twn-loop (over the variables $\vec{x} = (x_1, \ldots, x_d)$) is a tuple (φ, η) with the guard φ and the update $\eta : \mathcal{V} \to \mathbb{Z}[\mathcal{V}]$ for $\mathcal{V} = \{x_1, \ldots, x_d\}$ such that for all $1 \leq i \leq d$ we have $\eta(x_i) = a_i \cdot x_i + p_i$ for some $a_i \in \mathbb{Z}$ and $p_i \in \mathbb{Z}[x_1, \ldots, x_{i-1}]$. Thus, a twn-update is triangular, i.e., the update of a variable does not depend on variables with higher indices. Furthermore, the update is weakly non-linear, i.e., a variable does not occur non-linearly in its own update.

Our algorithm for the computation of runtime bounds for twn-loops starts with computing closed forms for the loop update, which describe the values of the variables after n iterations of the loop. These closed forms can be represented as so-called *poly-exponential expressions*. The set of all poly-exponential expressions is defined as $\mathbb{PE} = \{\sum_{j=1}^{k} p_j \cdot n^{a_j} \cdot b_j^n \mid k, a_j \in \mathbb{N},\ p_j \in \mathbb{Q}[\mathcal{V}],\ b_j \in \mathbb{Z}\}$.

Example 32. The closed forms for the loop (12) are $\mathtt{cl}_{x_1} = x_1 \cdot 3^n$ and $\mathtt{cl}_{x_2} = x_2 \cdot 2^n$.

The following Thm. 33 presents a construction based on closed forms which yields polynomial runtime bounds for terminating transitions $t = (\ell, \varphi, \eta, \ell)$

which correspond to twn-loops. We insert the closed forms of the update η into every atom $\alpha = p > 0$ of the guard φ. This results in a poly-exponential expression $pe_\alpha = \sum_{j=1}^{k_\alpha} p_{\alpha,j} \cdot n^{a_{\alpha,j}} \cdot b_{\alpha,j}^n \in \mathbb{PE}$ such that the summands are ordered w.r.t. the growth rate of $n^{a_{\alpha,j}} \cdot b_{\alpha,j}^n$. Now, the polynomials $p_{\alpha,j}$ in pe_α determine the asymptotic complexity of the resulting local runtime bound.

Theorem 33 (Polynomial Runtime Bounds for TWN-Loops). *Let $t = (\ell, \varphi, \eta, \ell)$ be a terminating transition and for every atom α in φ, let $pe_\alpha = \sum_{j=1}^{k_\alpha} p_{\alpha,j} \cdot n^{a_{\alpha,j}} \cdot b_{\alpha,j}^n \in \mathbb{PE}$ be a poly-exponential expression with $p_{\alpha,j} \neq 0$ for all $1 \leq j \leq k_\alpha$ and $(b_{\alpha,k_\alpha}, a_{\alpha,k_\alpha}) >_{\text{lex}} \ldots >_{\text{lex}} (b_{\alpha,1}, a_{\alpha,1})$ such that pe_α results from inserting the closed forms of η into α. Then*

$$\mathcal{RB}_{\text{loc}}^{\{t\},\{t\}}(\ell) = 2 \cdot \max_{\alpha \text{ occurs in } \varphi} \{\llbracket p_{\alpha,1} \rrbracket + \cdots + \llbracket p_{\alpha,k_\alpha - 1} \rrbracket\} + c$$

is a local runtime bound where $c \in \mathbb{N}$ is some computable constant.

Example 34. The loop (12) is terminating as the value of x_1 eventually outgrows the value of x_2. Inserting the closed forms of Ex. 32 into the atoms yields $pe_{x_2 - x_1 > 0} = -x_1 \cdot 3^n + x_2 \cdot 2^n$ and $pe_{x_1 > 0} = x_1 \cdot 3^n$. So, we have $p_{x_2 - x_1 > 0,\,1} = x_2$, $p_{x_2 - x_1 > 0,\,2} = -x_1$, and $p_{x_1 > 0,\,1} = x_1$. Hence, for the transition t and the location ℓ corresponding to (12), we obtain the polynomial local runtime bound $\mathcal{RB}_{\text{loc}}^{\{t\},\{t\}}(\ell) = 2 \cdot \llbracket p_{x_2 - x_1 > 0,\,1} \rrbracket + c = 2 \cdot x_2 + c$ where $c = 1$ (see [18] for the detailed construction of c).

While Thm. 33 always yields polynomial runtime bounds, we recently improved this to logarithmic runtime bounds if the exponential expressions are strictly decreasing, i.e., $b_{\alpha,k_\alpha} > \cdots > b_{\alpha,1}$. Intuitively, the reason is that then the summand $p_{\alpha,j} \cdot n^{a_{\alpha,j}} \cdot b_{\alpha,j}^n$ grows exponentially faster than all summands $p_{\alpha,i} \cdot n^{a_{\alpha,i}} \cdot b_{\alpha,i}^n$ for $i < j$.

Theorem 35 (Logarithmic Runtime Bounds for TWN-Loops). *Let $t = (\ell, \varphi, \eta, \ell)$ be a terminating transition and for every atom α in φ, let $pe_\alpha = \sum_{j=1}^{k_\alpha} p_{\alpha,j} \cdot n^{a_{\alpha,j}} \cdot b_{\alpha,j}^n \in \mathbb{PE}$ be a poly-exponential expression with $p_{\alpha,j} \neq 0$ for all $1 \leq j \leq k_\alpha$ and $b_{\alpha,k_\alpha} > \ldots > b_{\alpha,1}$ such that pe_α results from inserting the closed forms of η into α. Then*

$$\mathcal{RB}_{\text{loc}}^{\{t\},\{t\}}(\ell) = c' \cdot \log_2\Big(\max_{\alpha \text{ occurs in } \varphi} \{\llbracket p_{\alpha,1} \rrbracket + \cdots + \llbracket p_{\alpha,k_\alpha - 1} \rrbracket\}\Big) + c$$

is a local runtime bound where $c, c' \in \mathbb{N}$ are some computable constants.

Example 36. Reconsider Ex. 32 and 34: As the exponential terms in $pe_{x_2 - x_1 > 0} = -x_1 \cdot 3^n + x_2 \cdot 2^n$ and $pe_{x_1 > 0} = x_1 \cdot 3^n$ are strictly decreasing (i.e., $3 > 2$ for $pe_{x_2 - x_1 > 0}$), we can apply Thm. 35. So for the transition t and the location ℓ corresponding to (12), we obtain the *logarithmic* local runtime bound $\mathcal{RB}_{\text{loc}}^{\{t\},\{t\}}(\ell) = c' \cdot \log_2(\llbracket p_{x_2 - x_1 > 0,\,1} \rrbracket) + c = c' \cdot \log_2(x_2) + c = \log_2(x_2) + 2$ where $c = 2$ and $c' = 1$ (see [18] for the detailed construction of c and c').