Decidability Results for Well-Structured Transition Systems with Auxiliary Storage

R. Chadha and M. Viswanathan

Dept. of Computer Science, University of Illinois at Urbana-Champaign

Abstract. We consider the problem of verifying the safety of well-structured transition systems (WSTS) with auxiliary storage. WSTSs with storage are automata that have (possibly) infinitely many control states along with an auxiliary store, but which have a well-quasi-ordering on the set of control states. The set of reachable configurations of the automaton may themselves not be well-quasi-ordered because of the presence of the extra store. We consider the coverability problem for such systems, which asks if it is possible to reach a control state (with some store value) that covers some given control state. Our main result shows that if control state reachability is decidable for automata with some store and *finitely* many control states then the coverability problem can be decided for WSTSs (with infinitely many control states) and the same store, provided the ordering on the control states has some special property. The special property we require is defined in terms of the existence of a ranking function compatible with the transition relation. We then show that there are several classes of infinite state systems that can be viewed as WSTSs with an auxiliary storage. These observations can then be used to both reestablish old decidability results, as well as discover new ones.

1 Introduction

Algorithmic verification of infinite state systems has received considerable attention from the research community in the past decade because the semantics of many systems can be naturally described using an unbounded state space. Examples of such systems include recursive software (sequential or concurrent, with or without dynamic allocation), asynchronous distributed systems, real-time systems, hybrid systems, and stochastic systems. Since the general problem of model checking such systems is known to be undecidable, a variety of solutions have been proposed. These include semi-decision procedures to verify a system [9, 34, 11, 4, 8, 30, 55] or to find bugs [49, 48, 10], as well as identifying special classes of infinite state systems (and properties) for which model checking can be shown to be decidable [2, 19, 45, 3, 44, 39, 5, 6, 42, 26, 16, 23, 51].

While specialized approaches have been used to prove positive decidability results in many cases, a few general and broad techniques have emerged. One important technique is the use of *well-quasi-orders (wqo)* [38] (or stronger notions like *better-quasi-orders* [5]). The idea here is to identify a simulation relation (or some variant, like weak simulation, or stuttering simulation) on the (infinite state) transition system which is also a wqo. Then using the observation that any increasing sequence (with respect to subset ordering) of upward closed sets (with respect to a wqo) eventually stabilizes, a variety of problems, like backward reachability and simulation by finite state processes, can be shown to be decidable [1, 27]. Transition systems with a wqo simulation relation are called *well-structured transition systems (WSTS)*. Examples of WSTSs include petri nets (and variants) [47, 1, 27], finite state systems communicating over lossy/error-introducing channels [2, 19], integral relational automata [56], timed automata [7], dynamic networks of timed automata [3], and recursive parallel program schemes [37]. In addition to the technique of using w.q.o.s, other general techniques include: *tree interpretation* [53, 46, 17, 18, 54, 16] where decidability is proved by embedding the transition system in the infinite binary (or in general n-ary) tree and reducing the model checking problem to checking an MSO formula on the embedding which can be decided using Rabin's Theorem [50]; reducing the model checking problem to the first-order theory of reals [39, 23, 25, 24], which is known to be decidable from Tarski's result [52].

In this paper, we ask when the approach of using w.q.o.s can be combined with other techniques to prove general decidability results. We consider the problem of verifying safety properties of well-structured transition systems (WSTS) with an auxiliary store. WSTSs with auxiliary storage, which we call w.q.o.*automata*, are automata with (possibly infinitely many) control states that can store and retrieve information from an auxiliary data structure. Formally a data store is a domain of possible values, along with a set of predicates and operations to transform the store. The transitions of a w.q.o. automaton are guarded by a (pre-defined) predicate to test the value of the store, and transform the store using an allowed operation, in addition to changing the control state. Such automata are called w.q.o. automata because the control states are required to be ordered by a well-quasi-ordering that is compatible with the transition relation — a state q can be simulated by all control states greater (with respect to the ordering on states) than q. The semantics of such an automaton can be defined using a transition system, where the configurations are pairs (q, d), of a control state q, and a data value d. Notice the natural ordering on configurations — $(q_1, d_1) \preceq (q_2, d_2)$ iff $q_1 \preceq q_2$ and $d_1 = d_2$ — is not in general a w.q.o., and moreover, there maybe no w.q.o. on configurations compatible with the transitions. Therefore, techniques from the theory of WSTSs cannot be used directly to solve the model checking problem of wqo automata.

Our main theorem proves the decidability of the coverability problem for certain special w.q.o. automata. Recall that in the coverability problem we are given a control state q, and asked whether there is an execution, starting from the initial configuration, that can reach some control state $q' \succeq q$. The conditions required to prove decidability are as follows. First we require that the control state reachability problem be decidable for automata with *finitely many control states* and the same data store. Second, we require a ranking function on states compatible with the w.q.o. on states such that the number of states with a bounded rank (for any bound k) is finite. Finally, we require that transitions of the wqo automata that decrease the rank of the control state, are enabled only at a fixed data store and do not change the data store. We show that if a *wqo* automata satisfies these conditions then a backward reachability algorithm terminates, and can be used to solve the coverability problem; the termination of the algorithm relies on the properties of well-quasi orders. It is important to note that we have no requirements on the algorithm solving the control state reachability problem for the finite control state case (first condition above), and so it could rely on any of the techniques that have been discovered in the past.

We then show that there are many natural classes of systems that can be viewed as w.q.o. automata, for which our decidability result applies. Our main result can then be used to rediscover old results (but with a new proof), and establish many new decidability results. First we consider asynchronous programs [32, 28, 31, 40, 41, 20, 35], which are recursive programs that make both conventional *synchronous* function calls, where a caller waits until the callee completes computation, and *asynchronous* procedure calls, which are not immediately executed but are rather stored and "dispatched" by an external scheduler at a later point. Such systems can be abstracted (using standard techniques like predicate abstraction [29]) into automata with a multi-set (to store pending asynchronous calls) and a stack (for recursive calls), but which remove elements from the multi-set only when the stack is empty. The control state reachability problem for such recursive multi-set automata has been previously shown to be decidable [51, 33], and our main theorem provides a new proof of this fact. Moreover, because our main theorem is a generalization of these results, it can also be used to establish new decidability results. In particular we can prove the decidability of the control state reachability problem for automata with both a multi-set and a higher-order stack [36, 16, 14]. Such automata can be used to model asynchronous programs, where asynchronous procedures can be more generally (safe) higher-order recursive programs [36], rather than (first-order) recursive procedures.

Asynchronous programming as an idiom is being widely used in a variety of contexts. One particular context is that of networked embedded systems [28, 31], where asynchronous procedure calls form the basis of event-driven programming languages. Such embedded systems often need to meet real-time constraints, and so the dispatcher is required to schedule pending asynchronous calls based on the time when they were invoked. We show that such programs with boolean variables [1], can be modeled by automata with a stack, and multi-set of clocks. Then using our main theorem we prove the decidability of the control state reachability problem for such systems.

Finally, we consider networks of message passing recursive systems communicating over uni-directional, lossy FIFO channels. This model is particularly appropriate for describing networks of clients and servers, processing requests. The semantics of such systems can be defined using transition systems whose

[1] A general program can always be abstracted using techniques such as predicate abstraction [29] to obtain such restricted programs.

configurations have multiple stacks, and the contents of multiple lossy channels. We show that for control state reachability purposes, one can consider an equivalent system that has only one stack, and where losses are confined to take place just before a message receive step. Finally, the decidability of the verification problem is established using our main theorem.

Paper Outline. The rest of the paper is organized as follows. First we discuss closely related work. Then in Section 2, we present basic definitions and properties of well-quasi orders and ranking functions. We formally define w.q.o. automata in Section 3. Our main decidability result is presented next (Section 4). Section 5, gives examples of w.q.o. automata, and discusses the consequences of our main decidability result. Finally we conclude (Section 6) with some observations and future work.

1.1 Related Work

There is a large body of work on infinite state verification, and we cannot hope to justice to them in such a paper; therefore this section limits itself to work that is very close in spirit to this paper. Our work continues a line of work started in [51], where we considered automata with multi-sets and stacks to model asynchronous programs. The decidability of the control state reachability problem was proved using w.q.o. theory and Parikh's theorem. In [33], Jhala and Majumdar, simplified the proof of the decidability result, removing its reliance on Parikh's theorem. However both these proofs use features that are very specific to multi-sets and stacks, and cannot be easily generalized to obtain verification algorithms for the models considered in this paper. In particular, our original motivation was to look at the problem of verifying networked embedded systems [28, 31], which are real-time asynchronous programs, and the proof techniques in [51, 33] do not generalize to such a model. We discuss the differences between our proof approach and then one in [33] in more detail, when we present the main theorem.

Another very closely related work is the paper by Emmi and Majumdar [22]. One of the main observations concerns w.q.o. pushdown automata, which are pushdown automata with infinitely many control states that have a well-quasi ordering on control states. They show the decidability of the control state subcovering problem for such automata. Unfortunately, the proof presented in the paper is incorrect [43]. The authors conjecture that the decidability result for w.q.o. pushdown automata is true. Even if the conjecture is successfully proved there are some differences with our main theorem. First, the Emmi-Majumdar result considers *downward compatibility* of the ordering with the transitions and the sub-covering problem, whereas we consider upward compatibility and the coverability problem. Next, their result specifically applies to automata with stacks, and not to other data structures like higher-order stacks that we consider here. On the flip side, their conjecture does not impose any conditions on the wqo on states itself (like ranking functions) that we require for our result. So if the Emmi-Majumdar conjecture is successfully proved then the main theorem here apply to incomparable classes of systems.

Finally there is a long line of work on proving decidability results for concurrent recursive systems, with varying degrees of synchronization; see [17, 45, 44, 42, 15, 12] for some examples. The examples of dynamic networks of recursive programs that we consider here (like asynchronous programs, and real-time asynchronous programs) are generally incomparable in expressiveness to the these models. However, the most powerful class considered in [13] is more expressive than simple asynchronous programs. However, the results in [13] only allow to verify inductive invariants expressed in a special sub-logic that they consider; in general verifying a safety property involves finding an inductive invariant that implies the safety property.

2 Preliminaries

We first recall some standard definitions, notations and facts about well-quasi-orders [38].

2.1 Well-quasi-orders

A binary relation $\preceq$ on a set Q is said to be a *pre-order* if $\preceq$ is reflexive and transitive. Please note that a pre-order need not satisfy anti-symmetry, *i.e.*, it may be the case that $q \preceq q'$ and $q' \preceq q$ for $q \neq q'$. We shall say that q *is strictly less that* q' *(written as* $q \prec q'$*)* if $q \preceq q'$ but $q' \not\preceq q$. Two elements q, q' are said to be *comparable* if either $q \preceq q'$ or $q' \preceq q$ and said to be *incomparable* otherwise. We write $q \succeq q'$ if $q' \preceq q$ and $q \succ q'$ if $q' \prec q$.

A pre-order $\preceq$ on a set Q is said to be a *well-quasi-order* if every countably infinite sequence of elements $q_1, q_2, q_3, \ldots,$ from Q contains elements $q_r \preceq q_s$ for some $0 \leq r < s$. Equivalently, a pre-order is a well-quasi-order if there is no infinite sequence of pairwise incomparable elements and there is no strictly descending infinite sequence (of the form $q_1 \succ q_2 \succ q_3 \succ \ldots$). For the rest of paper, we shall say that $(\mathsf{Q}, \preceq)$ is a $\mathsf{w.q.o.}$ if $\preceq$ is a well-quasi-order on Q.

Let $(\mathsf{Q}, \preceq)$ be a $\mathsf{w.q.o.}$ and $\mathsf{Q}' \subseteq \mathsf{Q}$. We say that $\mathsf{M}_{\mathsf{Q}'} \subseteq \mathsf{Q}'$ is a *minor set* for Q' if for i) for all $q \in \mathsf{Q}'$ there is a $q' \in \mathsf{M}_{Q'}$ such that $q' \preceq q$, and ii) for all $q_1, q_2 \in \mathsf{M}_{Q'}$, $q_1 \neq q_2$ implies $q_1 \not\preceq q_2$. The definition of well-quasi-ordering implies that each subset of Q has at least one minor set and all minor sets are finite. The elements of the minor sets are minimal elements in the following sense.

Proposition 1. *Let* M_{Q} *be a minor set for* Q *and let* $q \in \mathsf{M}_{\mathsf{Q}}$. *For any* $q_1 \in \mathsf{Q}$, *if* $q_1 \preceq q$ *then* $q \preceq q_1$.

Proof. Let $q_1 \in \mathsf{Q}$ be such that $q_1 \preceq q$. Since M_{Q} is a minor set for Q, there is a $q' \in \mathsf{M}_{\mathsf{Q}}$ such that $q' \leq q_1$. By transitivity, we get $q' \leq q$. We conclude by observing that since $q, q' \in \mathsf{M}_{\mathsf{Q}}$, we have $q = q'$ by definition. $\square$

A set $\mathsf{U} \subseteq \mathsf{Q}$ is said to be *upward closed* if for every $q_1 \in \mathsf{U}$ and $q_2 \in \mathsf{Q}$, $q_1 \preceq q_2$ implies that $q_2 \in \mathsf{U}$. An upward closed set is completely determined by its minor set: if M_U is a minor set for U then $\mathsf{U} = \{q \in \mathsf{U} \mid \exists q_m \in \mathsf{M}_\mathsf{U} \text{ s.t. } q_m \preceq q\}$. Also any subset $\mathsf{Q}' \subseteq \mathsf{Q}$ determines an upward closed set, $\mathsf{U}_{\mathsf{Q}'} = \{q \mid \exists q' \in Q' \text{ s.t. } q' \preceq q\}$. The following important observation follows from $\mathsf{w.q.o.}$ theory.

Proposition 2. *For every infinite sequence of upward closed sets* $\mathsf{U}_1, \mathsf{U}_2 \ldots\ldots$ *such that* $\mathsf{U}_r \subseteq \mathsf{U}_{r+1}$ *there is a* j *such that* $\mathsf{U}_l = \mathsf{U}_j$ *for all* $l \geq j$.

2.2 Ranking functions

If the order $\preceq$ also satisfies anti-symmetry then it is possible to define a function rank from Q into the class of ordinals as: $\mathsf{rank}(q) = 0$ if Q does not have any elements strictly less than q and $\mathsf{rank}(q) = sup(\{\mathsf{rank}(q') \mid q' \prec q\}) + 1$ otherwise. The function rank guarantees that if q_1, q_2 are comparable then $\mathsf{rank}(q_1) < \mathsf{rank}(q_2)$ iff $q_1 \prec q_2$. We adapt the concept of the rank function for pre-orders. First instead of working with the whole class of ordinals, we shall work with the set of natural numbers[2]. Furthermore, we shall only require that $\mathsf{rank}(q_1) \leq \mathsf{rank}(q_2)$ if $q_1 \preceq q_2$.

Definition 1. *[Ranking function] Given a* $\mathsf{w.q.o.}$ $(\mathsf{Q}, \preceq)$*, a function* $\alpha : \mathsf{Q} \to \mathbb{N}$ *is said to be a ranking function if for every* $q_1, q_2 \in \mathsf{Q}$*,* $q_1 \preceq q_2$ *implies* $\alpha(q_1) \leq \alpha(q_2)$.

The ranking function α can be extended to a function on the set of upward closed sets of $(\mathsf{Q}, \preceq)$ as follows. Let M_U be a minor set for an upward closed U. Let $\alpha_{\max}(\mathsf{U}) = \max\{\alpha(q) | q \in \mathsf{M}_\mathsf{U}\}$. It can be easily shown that this extension is well-defined, *i.e.*, does not depend on the choice of M_U.

3 Well-structured transition systems with auxiliary storage

The paper considers automata with an auxiliary store and possibly infinitely many control states, such that there is a well-quasi-ordering defined on the control states. We formally, define such automata in this section. We begin by first introducing the concept of a pointed data structure that formalizes the notion of an auxiliary store.

3.1 Pointed data structures

Definition 2 (Pointed data structure). *A pointed data structure is a tuple* $\mathsf{D} = (D, \widetilde{op}, \widetilde{pred}, d_i, p_i)$ *such that* D *is a set,* $\widetilde{op}$ *is a collection of functions* $f : D \to D$*,* $\widetilde{pred}$ *is a collection of unary predicates on* D*,* d_i *is an element of* D *and* $p_i \in \widetilde{pred}$ *is a unary predicate on* D *such that* $p_i(d) \Leftrightarrow (d = d_i)$*. The elements of* D *are henceforth called data values and the data value* d_i *is said to be the initial data value.*

[2] The set of natural numbers is also the set of all finite ordinals.

For example, a pushdown store on a alphabet Γ in a pushdown automata can be formalized as follows. The set Γ^* (set of all finite strings over Γ) can be taken as the set of $\widetilde{data}$ values with the empty string ϵ as the initial value. The set of predicates $\widetilde{pred}$ can be chosen as $\{\mathsf{empty}\} \cup \{\mathsf{top}_\gamma \,|\, \gamma \in \Gamma\} \cup \{\mathsf{any}\}$, where $p_i = \{\epsilon\}$ (the initial predicate), $\mathsf{top}_\gamma = \{w\gamma \,|\, w \in \Gamma^*\}$ (the top of stack is γ) and $\mathsf{any} = \Gamma^*$ (any stack). The set of functions $\widetilde{op}$ can be defined as $\{id\} \cup \{\mathsf{push}_\gamma \,|\, \gamma \in \Gamma\} \cup \{\mathsf{pop}_\gamma \,|\, \gamma \in \Gamma\}$ where push_γ and pop_γ are defined as follows. For all $w \in \Gamma^*$, $\mathsf{push}_\gamma(w) = w\gamma$ and $\mathsf{pop}_\gamma(w) = w_1$ if $w = w_1\gamma$ and w otherwise. In a pushdown system the function pop_γ will be enabled only when the store satisfies top_γ. The function push_γ is enabled when the store satisfies any.

For the rest of paper, we shall assume that our data structure has a finite number of predicates and a finite number of functions. This is mainly a matter of convenience and we could have dealt with countable number of predicates and functions as long as the data structure is finitely presentable. We could also have dealt with a finite number of initial values, partial functions and relations in the set $\widetilde{op}$. However, for the sake of clarity, we have omitted these cases here.

3.2 w.q.o. Automata

We now define a w.q.o. automaton $\mathbf{A}$ that formalizes the notion of WSTS with auxiliary storage.

Definition 3. *[w.q.o. automaton] A w.q.o. automaton on a pointed data structure $\mathsf{D} = (D, \widetilde{op}, \widetilde{pred}, d_i, p_i)$ is a tuple $(\mathsf{Q}, \preceq, \delta, q_i)$ such that*

1. *$(\mathsf{Q}, \preceq)$ is a w.q.o.,*
2. *$q_i \in \mathsf{Q}$ and*
3. *$\delta \subseteq \mathsf{Q} \times \widetilde{pred} \times \widetilde{op} \times \mathsf{Q}$. Furthermore, the set δ is upward compatible, i.e. for every $(q, p, g, q') \in \delta$ and $q \preceq q_1$ there exists q'_1 such that $q' \preceq q'_1$ and $(q_1, p, g, q'_1) \in \delta$.*

The set Q is said to be the set of control states of the automaton, δ the transition function and q_i the initial state. If the set Q is finite then we say that it is a finite w.q.o. automaton.

Please note that given an upward closed set $\mathsf{U} \subseteq \mathsf{Q}$, a predicate $p \in \widetilde{pred}$, $g \in \widetilde{op}$ let $\delta^{-1}(p, g, \mathsf{U})$ be the set $\{q \,|\, \exists q' \in \mathsf{U} \,\text{s.t.}\, (q, p, g, q') \in \delta\}$. The upward compatibility of δ ensures that $\delta^{-1}(p, g, \mathsf{U})$ is either empty or upward closed.

An example of such an automaton in literature is multi-set pushdown automata [51, 33] and discussed in Section 5.1. For the remainder of the section, we shall fix a pointed data structure $\mathsf{D} = (D, \widetilde{op}, \widetilde{pred}, d_i, p_i)$ and a w.q.o. automaton $\mathbf{A}$ on D.

The semantics of a *wqo* automaton is defined in terms of a transition system over a set of configurations. A *configuration* is a pair (q, d), where $q \in \mathsf{Q}$ is a control state and $d \in D$ is a data value. The pair (q_i, d_i) is said to be the *initial configuration.* For $\delta_0 \subseteq \delta$, we say $(q_1, d_1) \rightarrow_{\mathbf{A},\delta_0} (q_2, d_2)$ if there is a transition $(q_1, p, g, q_2) \in \delta_0$ such that $p(d_1)$ and $d_2 = g(d_1)$. We shall omit $\mathbf{A}$ if the automaton under consideration is clear from the context. The transition relation on configurations will be $\rightarrow_{\mathbf{A},\delta}$. The n-fold composition of $\rightarrow_{\delta_0}$ will be denoted by $\rightarrow^n_{\delta_0}$. In other words, $(q, d) \rightarrow^n_{\delta_0} (q', d')$ iff there are $(q_0, d_0), (q_1, d_1), \dots (q_n, d_n)$ such that $q_0 = q, d_0 = d$, $q_n = q', d_n = d'$ and for every $0 \le r < n$ $(q_r, d_r) \rightarrow_{\delta_0} (q_{r+1}, d_{r+1})$. We shall write $(q, d) \rightarrow^*_{\delta_0} (q', d')$ if there is some $j \ge 0$ such that $(q, d) \rightarrow^j_{\delta_0} (q', d')$. Finally, we shall say that the configuration (q', d') is reachable from (q, d) using transition in δ_0 if $(q, d) \rightarrow^*_{\delta} (q', d')$.

Given a set $\mathsf{Q}' \subseteq Q$, it will be useful to define two sets, $\mathsf{Pre}^*_{\mathbf{A},\,\delta_0,d_i}(Q')$ and $\mathsf{Pre}^*_{\mathbf{A},\,\delta_0}(Q')$. The set $\mathsf{Pre}^*_{\mathbf{A},\,\delta_0,d_i}(\mathsf{Q}') = \{q \mid \exists q' \in \mathsf{Q}' \text{ s.t. } (q, d_i) \rightarrow^*_{\delta_0} (q', d_i)\}$ gives the set of all states from which some control state in Q' can be reached starting with and ending with the initial data value d_i. The set $\mathsf{Pre}^*_{\mathbf{A},\,\delta_0}(\mathsf{Q}') = \{q \mid \exists q' \in \mathsf{Q}', d \in D \text{ s.t. } (q, d_i) \rightarrow^*_{\delta_0} (q', d)\}$ gives the set of all states from which some control state in Q' can be reached starting with the initial data value and ending with some data value. We will omit the subscript $\mathbf{A}$ if it is clear from the context.

Observe that since δ is upward compatible, if $(q, d) \rightarrow^n_{\delta} (q', d')$ then for every $q_1 \succeq q$ there exists an $q'_1 \succeq q'$ such that $(q_1, d) \rightarrow^n_{\delta} (q'_1, d')$. Therefore, for any upward-closed set U, the sets $\mathsf{Pre}^*_{\mathbf{A},\,\delta}(\mathsf{U})$ and $\mathsf{Pre}^*_{\mathbf{A},\,\delta,d_i}(\mathsf{U})$ are upward closed.

It will also be useful to identify two kinds of transitions in a *wqo* automaton. The first ones are fired only when the data is initial and preserve the data.

Definition 4 (Initial data preserving). *A transition $(q, p, g, q') \in \delta)$ is said to be initial data preserving if $p = p_i$ and $g = id$ where id is the identity function.*

The other kind of transitions will be *rank non-decreasing* transitions which will be defined with respect to a ranking function (see Definition 1).

Definition 5 (Rank non-decreasing). *Given a ranking function α on $(\mathsf{Q}, \preceq)$ and a set of transitions $\delta_0 \subseteq \delta$, we say that δ_0 is rank α non-decreasing if for each predicate $p \in \widetilde{pred}$, function $g \in \widetilde{op}$ and upward closed set $\mathsf{U} \subset \mathsf{Q}$ either $\delta_0^{-1}(p, g, \mathsf{U}) = \emptyset$ or $\delta_0^{-1}(p, g, \mathsf{U})$ is upward-closed and $\alpha_{\max}(\delta_0^{-1}(p, g, \mathsf{U})) \le \alpha_{\max}(\mathsf{U})$.*

The above condition can also be stated in the following way.

Proposition 3. *An upward-compatible set $\delta_0 \subseteq \delta$ is rank α non-decreasing iff for any $\hat{q}, q_1, q'_1 \in \mathsf{Q}$, $p \in \widetilde{pred}$, $g \in \widetilde{op}$ such that $\hat{q} \preceq q'_1$ and $(q_1, p, g, q'_1) \in \delta_0$ then there are q, q' such that $\alpha(q) \le \alpha(\hat{q})$, $q \preceq q_1$, $\hat{q} \preceq q'$ and $(q, p, g, q') \in \delta_0$.*

Proof. ($\Rightarrow$) Assume that δ_0 is non-decreasing. Let $\hat{q}, q_1, q'_1 \in \mathsf{Q}$, $p \in \widetilde{pred}$, $g \in \widetilde{op}$ be such that $(q_1, p, g, q'_1) \in \delta_0$. Consider the upper closed set $\mathsf{U} = \mathsf{U}_{\{\hat{q}\}}$. Let $\mathsf{U}_1 = \delta_0^{-1}(p, g, \mathsf{U})$. By definition, $\alpha_{\max}(\mathsf{U}) = \alpha(\hat{q})$, $q'_1 \in U$, $q_1 \in \mathsf{U}_1$ and $\alpha_{\max}(\mathsf{U}_1) \le \alpha(\hat{q})$.

Let $\mathsf{M}_{\mathsf{U}_1}$ be a minor set for U_1. As $q_1 \in \mathsf{U}_1$, there is a $q \in \mathsf{M}_{\mathsf{U}_1}$ such that $q \le q_1$. Now, since $\mathsf{U}_1 = \delta_0^{-1}(p, g, \mathsf{U})$, there is a $q' \in \mathsf{U}$ such that $(q, p, g, q') \in \delta_0$. Also, as $\mathsf{U} = \mathsf{U}_{\hat{q}}$, we get $\hat{q} \le q'$. Finally, please note that by definition $\alpha(q) \le \alpha_{\max}(\mathsf{U}_1)$. Hence, $\alpha(q) \le \alpha(\hat{q})$.

($\Leftarrow$) Let δ_0 be an upward compatible set of transitions, $p \in \widetilde{pred}$ be a predicate, $g \in \widetilde{op}$ be a function and U be an upward closed set such that $\delta_0^{-1}(p, g, \mathsf{U}) \neq \emptyset$. Let M_U be a minor set for U, $\mathsf{U}_1 = \delta_0^{-1}(p, g, \mathsf{U})$ and $\mathsf{M}_{\mathsf{U}_1}$ be a minor set for U_1. Pick $q_1 \in \mathsf{M}_{\mathsf{U}_1}$ and fix it. It suffices to show that $\alpha(q_1) \leq \alpha_{\max}(U)$.

As $q_1 \in \mathsf{U}_1$, there exists $q_1' \in \mathsf{U}$ such that $(q_1, p, g, q_1') \in \mathsf{U}$. Now, by definition, there is $\hat{q} \in \mathsf{M}_\mathsf{U}$ such that $\hat{q} \preceq q_1'$. By the hypothesis of the proposition, there are q, q' such that $\alpha(q) \leq \alpha(\hat{q})$, $q \preceq q_1$, $\hat{q} \preceq q'$ and $(q, p, g, q') \in \delta_0$.

Please note by definition, $q \in \mathsf{U}_1$ and $\alpha(\hat{q}) \leq \alpha_{\max}(\mathsf{M}_U)$. Also since $q_1 \in \mathsf{M}_{\mathsf{U}_1}$ and $q \preceq q_1$, we have by Proposition 1 $q_1 \preceq q_1$ also. Hence, we get $\alpha(q) \leq \alpha(q_1) \leq \alpha(\hat{q}) \leq \alpha_{\max}(\mathsf{M}_U)$ as required. $\square$

As mentioned in the introduction, we will consider special w.q.o. automata, namely those in which the only rank decreasing transition are those that are also initial data preserving. We will say that such a restricted w.q.o. automata is *compatible to a ranking function*; we define this formally next.

Definition 6 (Compatibility of ranking function). *Given a ranking function α on $(\mathsf{Q}, \preceq)$ we say that α is compatible with δ if*

1. $\alpha(q_i) = 0$*, and*
2. *there exist $\delta = \delta_a \cup \delta_b$ such that*
   - *δ_b is the set of all initial data preserving transitions.*
   - *$\delta_a = \delta \setminus \delta_b$ and δ_a is rank α non-decreasing. The pair (δ_a, δ_b) is said to be the α-compatible splitting of δ.*

The condition $\alpha(q_i) = 0$ is not a serious constraint as we can always define a ranking function $\alpha'(q) = \max(\alpha(q) - \alpha(q_i), 0)$. The second condition in the definition implies that the transition function δ can be split into two disjoint upward-compatible sets δ_a and δ_b. All transitions in δ_b are enabled only when the data value is initial and these transitions also do not modify the data. This condition on δ_b ensures that any computation $(q, d_i) \rightarrow_\delta^* (q', d')$ is of the form $(q, d_i) \rightarrow_{\delta_a}^* (q_0, d_i) \rightarrow_{\delta_b} (q_1, d_i) \rightarrow_{\delta_a}^* (q_2, d_i) \rightarrow_{\delta_b} (q_2, d_i) \ldots \rightarrow_{\delta_b} (q_n, d_i) \rightarrow_{\delta_a}^* (q', d')$ for some $q_0, q_1, \ldots q_n \in \mathsf{Q}$.

4 Decidability of the coverability problem

Given a w.q.o. automaton $\mathbf{A} = (\mathsf{Q}, \preceq, \delta, q_i)$ on a pointed data structure $\mathsf{D} = (D, \widetilde{op}, \widetilde{pred}, d_i, p_i)$, an upward closed set U, we are interested in deciding if there is some configuration (q, d) such that $q \in \mathsf{U}$ and q is reachable from the initial configuration q_i. The upward closed set U is often represented by its finite minor set. This problem is known as *coverability problem* and we shall study two versions of this problem. The first is whether there exists some state $q \in \mathsf{U}$ such that the configuration (q, d_i) is reachable from (q_i, d_i), *i.e.*, if $q_i \in \mathsf{Pre}^*_{\mathsf{A},\,\delta, d_i}(\mathsf{U})$. Secondly whether there exists some state $q \in \mathsf{U}$ and $d \in D$ such that (q, d) is reachable from (q_i, d_i), *i.e.*, if $q_i \in \mathsf{Pre}^*_{\mathsf{A},\,\delta}(\mathsf{U})$.

We start by describing how the coverability problem is tackled in WSTSs without the store. In that case, the transition relation $\delta \subseteq \mathsf{Q} \times \mathsf{Q}$ and we are interested in deciding whether the reflexive transitive closure $(\delta^{-1})^*$of the relation δ^{-1} contains q_i or not. A backward reachability analysis is performed in order to decide the coverability problem. An increasing sequence U_i is generated such that $\mathsf{U}_1 = \mathsf{U}$ and $\mathsf{U}_{j+1} = \mathsf{U}_j \cup \delta^{-1}(\mathsf{U}_j)$. The upward compatibility ensures that $\delta^{-1}(\mathsf{U}_j)$ is upward closed. Since any increasing sequence of upward closed sets in a w.q.o. must be eventually constant, we terminate once $\mathsf{U}_j = \mathsf{U}_{j+1}$.

Our approach to the coverability problem for a w.q.o. automaton will follow a similar approach. Given a w.q.o. automaton $\mathbf{A} = (\mathsf{Q}, \preceq, \delta, q_i)$ on a pointed data structure $\mathsf{D} = (D, \widetilde{op}, \widetilde{pred}, d_i, p_i)$, we shall assume the existence of a ranking function α compatible with the transition relation δ. Let (δ_a, δ_b) be the α-splitting of δ. We shall always assume that we can compute the minor set for $\delta_b^{-1}(\mathsf{U}, p_i, id)$ given a minor set for U. However, we shall need to compute $\mathsf{Pre}^*_{\mathbf{A},\, \delta_a, d_i}(\mathsf{U})$. For this we shall need the notion of rank k-approximations.

4.1 Rank k-approximations

Intuitively, the rank k-approximation $\mathbf{A}_k$ of $\mathbf{A}$ is constructed from the subset of control states of $\mathbf{A}$ whose rank is less than k. The construction is carried out in a way that captures all the computations of the original automaton that use the transitions in δ_a.

Definition 7 (Rank k-approximation). *Given a w.q.o. automaton $\mathbf{A} = (\mathsf{Q}, \preceq, \delta, q_i)$ on a pointed data structure $\mathsf{D} = (D, \widetilde{op}, \widetilde{pred}, d_i, p_i)$, a ranking function α on $(\mathsf{Q}, \preceq)$ such that α is compatible with δ. Let (δ_a, δ_b) be the α-compatible splitting of δ. Given $k \in \mathbb{N}$, the rank k approximation is defined as the automaton $\mathbf{A}_k = (\mathsf{Q}_{\leq k}, \preceq_k, \delta_k, q_i)$ where:*

- $\mathsf{Q}_{\leq k} = \{q \in \mathsf{Q} \,|\, \alpha(q) \leq k\}$,
- *$q \preceq_k q'$ for $q, q' \in \mathsf{Q}_{\leq k}$ iff $q \preceq q'$ and*
- *$(q, p, g, q') \in \delta_k$ for $q, q' \in \mathsf{Q}_{\leq k}$ iff there exists $q'' \in \mathsf{Q}$ such that $q'' \succeq q'$ and $(q, p, g, q'') \in \delta_a$.*

For the rest of the section, we shall assume a fixed w.q.o. automaton $\mathbf{A} = (\mathsf{Q}, \preceq, \delta, q_i)$ on a fixed pointed data structure $\mathsf{D} = (D, \widetilde{op}, \widetilde{pred}, d_i, p_i)$ and a fixed ranking function α on $(\mathsf{Q}, \preceq)$ compatible with δ. Let (δ_a, δ_b) be the α-compatible splitting of the transition relation δ.

The following Lemma states that any computation in the rank k-approximation of $\mathbf{A}$ corresponds to a computation of $\mathbf{A}$ that uses the transitions in δ_a.

Lemma 1 (Soundness of approximations). *Let $\mathbf{A}_k = (\mathsf{Q}, \preceq, \delta, q_i)$ be the rank k-approximation of $\mathbf{A}$. For any $q, q_1 \in \mathsf{Q}_{\leq k}, d, d_1 \in D$, if $(q, d) \rightarrow^*_{\delta_k} (q_1, d_1)$ then there is some $q_1' \in \mathsf{Q}$ such that $q_1' \succeq q_1$ and $(q, d) \rightarrow^*_{\delta_a} (q_1', d_1)$.*

Proof. The proof is by induction on the number of steps in the computation $(q, d) \rightarrow^*_{\delta_k} (q_1, d_1)$.

Base case. The number of computation steps in 0. Then the lemma follows trivially by choosing q_1' to be q_1.

Induction Hypothesis. Assume that the lemma is true for all computations $(q,d) \rightarrow^*_{\delta_k} (q_1,d_1)$ of j steps. Consider a computation $(q,d) \rightarrow^j_{\delta_k} (q_1,d_1) \rightarrow_{\delta_k} (q_2,d_2)$ of $j+1$ steps. By induction hypothesis, there is a $q_1' \in \mathsf{Q}$ such that $q_1' \succeq q_1$ such that $(q,d) \rightarrow^*_{\delta_a} (q_1',d_1)$.

Also, by definition of rank k-approximation, since $(q_1,d_1) \rightarrow_{\delta_k} (q_2,d_2)$ there is some $q_2'' \in \mathsf{Q}$ such that $q_2'' \succeq q_2$ $(q_1,d_1) \rightarrow_{\delta_a} (q_2'',d_2)$. Now, since $q_1 \preceq q_1'$ and $(q_1,d_1) \rightarrow_{\delta_a} (q_2'',d_2)$, we get by upward-compatibility of the transition relation δ_a, there is some q_2' such that $q_2'' \preceq q_2'$ and $(q_1',d_1) \rightarrow_{\delta_a} (q_2',d_2)$.

Please note by transitivity of $\preceq$, we get $q_2 \preceq q_2'$. Furthermore, as $(q,d) \rightarrow^*_{\delta_a} (q_1',d_1)$ and $(q_1',d_1) \rightarrow_{\delta_a} (q_2',d_2)$, we get $(q,d) \rightarrow^*_{\delta_a} (q_2',d_2)$ as required. □

The following Lemma states that any computation of the automaton $\mathbf{A}$ that uses transitions in δ_a and ending in a state which is above some state $q_0 \in \mathsf{Q}_{\leq k}$ is reflected in its k-th approximation.

Lemma 2 (Faithfulness of approximation). *Let $\mathbf{A}_k = (\mathsf{Q}, \preceq, \delta, q_i)$ be a rank k-approximation of $\mathbf{A}$. If there are $q_0, q, q' \in \mathsf{Q}$ such that $q_0 \in \mathsf{Q}_{\leq k}$, $q' \succeq q_0$ and $(q,d) \rightarrow^*_{\delta_a} (q',d')$, then there is a $q_1 \in \mathsf{Q}_{\leq k}$ such that $q_1 \preceq q$ and $(q_1,d) \rightarrow^*_{\delta_k} (q_0,d')$.*

Proof. The proof is by induction on the number of steps in the computation $(q,d) \rightarrow^*_{\delta_a} (q',d_1)$.

Base case. The number of steps is 0. The lemma trivially follows by choosing q_1 to be q_0.

Induction Hypothesis. Assume that the lemma is true for all computations $(q,d) \rightarrow^*_{\delta_a} (q',d')$ of j steps. Consider a computation $(q,d) \rightarrow_{\delta_a} (\hat{q},\hat{d}) \rightarrow^j_{\delta_a} (q',d')$ of $j+1$ steps.

Now, by induction hypothesis, there is a q_2 such that $q_2 \preceq \hat{q}$, $q_2 \in \mathsf{Q}_{\leq k}$ and $(q_2,\hat{d}) \rightarrow^*_{\delta_k} (q_0,d')$.

Now, since $(q,d) \rightarrow_{\delta_a} (\hat{q},\hat{d})$ there exists $p \in \widetilde{pred}$ and $g \in \widetilde{op}$ such that $(q,p,g,\hat{q}) \in \delta_a$, $p(d)$ and $\hat{d} = g(d)$. Also since $q_2 \preceq \hat{q}$, we get by Proposition 3 that there is a $q_1 \preceq q$ and a $q_1' \succeq q_2$ such that $\alpha(q_1) \leq \alpha(q_2)$ and $(q_1,p,g,q_1') \in \delta_a$.

Since, $\alpha(q_2) \leq k$, we get $\alpha(q_1) \leq k$ and by construction $(q_1,p,g,q_2) \in \delta_k$. Thus, we get $(q_1,d) \rightarrow_{\delta_k} (q_2,\hat{(d)}) \rightarrow^*_{\delta_k} (q_0,d')$ as required. □

The above two lemmas show that if $\mathsf{U} \subseteq \mathsf{Q}$ is an upward closed set such that $\alpha_{\max}(\mathsf{U}) = k$ then minor sets for the sets $\mathsf{Pre}^*_{\mathbf{A},\delta_a,d_i}(\mathsf{U})$ and $\mathsf{Pre}^*_{\mathbf{A}_k,\delta_k}(U)$ can be constructed by just considering the rank k-approximation.

Corollary 1. *Let $\mathsf{U} \subseteq \mathsf{Q}$ be an upward-closed set and $\mathsf{M}_\mathsf{U} = \{q_1, q_2, \ldots, q_r\}$ be a minor set for U. If $\alpha_{\max}(\mathsf{U}) = k$, then let $\mathbf{A}_k = (\mathsf{Q}_{\leq k}, \preceq_k, \delta_k, q_i)$ be a rank k approximation for the automaton $\mathbf{A}$.*

Let Q_1 be a minor set for $\mathsf{Pre}^*_{\mathbf{A}_k,\,\delta_k,d_i}(\mathsf{M}_U)$ and let Q_2 be a minor set for $\mathsf{Pre}^*_{\mathbf{A}_k,\,\delta_k}(\mathsf{M}_U)$. Then Q_1 is also a minor set for $\mathsf{Pre}^*_{\mathbf{A},\,\delta_a,d_i}(\mathsf{U})$ and Q_2 is a minor set for $\mathsf{Pre}^*_{\mathbf{A},\,\delta_a}(\mathsf{U})$.

Thus, if we have algorithms to compute minor sets for $\mathsf{Pre}^*_{\mathbf{A}_k,\,\delta_k,d_i}(\mathsf{Q}')$ and $\mathsf{Pre}^*_{\mathbf{A}_k,\,\delta_k}(Q')$ for any subset $Q' \subseteq Q$, then we can compute the minor sets $\mathsf{Pre}^*_{\mathbf{A},\,\delta_a,d_i}(\mathsf{U})$ and $\mathsf{Pre}^*_{\mathbf{A},\,\delta_a}(\mathsf{U})$ for an upward closed set U whose rank is k. Towards this end, we shall define effective *wqo* automata.

4.2 Effective w.q.o.automata and coverability

We start by defining a ranking function effectively compatible with the transition relation of a *wqo* automata.

Definition 8 (Effectively compatible ranking functions). *Given a w.q.o. automaton, $\mathbf{A} = (\mathsf{Q}, \leq, \delta, q_i)$ on the pointed data structure $\mathsf{D} = (D, \widetilde{pred}, \widetilde{op}, p_i, d_i)$, a ranking function $\alpha : \mathsf{Q} \to \mathbb{N}$ compatible with δ is said to be effectively compatible with δ if the following hold.*

- *For each $k \in \mathbb{N}$, the set $Q_{\leq k} = \{q \in \mathsf{Q} \mid \alpha(q) \leq k\}$ is finite.*
- *There is an algorithm* Rank *that on input $q \in \mathsf{Q}$ computes $\alpha(q)$.*
- *There is an algorithm* Elements *that on input $k \in \mathbb{N}$ computes the set $\mathsf{Q}_{\leq k}$.*
- *There is an algorithm* Approx *that on input $k \in \mathbb{N}$ outputs the rank k-approximation.*[3]

An effectively compatible ranking function is finitely presented by the algorithms Rank*,* Elements *and* Approx*.*

If α is effectively compatible with δ, and the relation $\preceq$ is decidable then corollary 1 ensures that algorithms to check if $q_1 \in \mathsf{Pre}^*_{\mathbf{A},\delta_a,d_i}(\mathsf{U})$ or $q_1 \in \mathsf{Pre}^*_{\mathbf{A},\delta_a}(U)$ exist if state reachability can be solved for finitely many states. We are now almost ready to prove our main theorem. We need one more definition.

Definition 9 (Effective w.q.o. automaton). *A w.q.o. automaton $\mathbf{A} = (\mathsf{Q}, \leq, \delta, q_i)$ on the pointed data structure $\mathsf{D} = (D, \widetilde{pred}, \widetilde{op}, p_i, d_i)$ is said to be effective if the following hold.*

- *There is a ranking function $\alpha : \mathsf{Q} \to \mathbb{N}$ effectively compatible with δ. Let the algorithms* Rank*,* Elements*,* Approx *finitely present the ranking function.*
- *There is an algorithm* Less *that on inputs q_1 and q_2 returns true is $q_1 \preceq q_2$ and false otherwise.*
- *There is an algorithm* InvDeltab *which given a minor set for an upward closed set U returns a minor set for $\delta_b^{-1}(\mathsf{U}, p_i, id)$ where id is the identity function on D where δ_b is the set of initial data preserving transitions.*

An effective automaton is finitely presented by the algorithms Rank*,* Elements*,* Approx*,* Less *and* InvDeltab*.*

We now have the main result of the paper.

[3] Please note that since we are assuming that predicates and functions are finite sets, a finite presentation of the rank k-function always exists.

Theorem 1 (Decidability of the coverability problem). *Assume that for a data structure* $\mathsf{D} = (D, \widetilde{pred}, \widetilde{op}, p_i, d_i)$ *there are two algorithms* $\mathcal{A}$ *and* $\mathcal{B}$ *such that the following hold.*

– *Given a finite wqo automaton* $\mathbf{A}_1 = (\mathsf{Q}_1, \preceq_1, \delta_1, q_i)$*, and* $q_1, q_1' \in Q_1$ *the algorithm* $\mathcal{A}$ *returns true if* $q_1 \in \mathsf{Pre}^*_{\mathbf{A}_1,\delta_1,d_i}(q_1')$ *and false otherwise.*
– *Given a finite wqo automaton* $\mathbf{A}_1 = (\mathsf{Q}_1, \preceq_1, \delta_1, q_i)$*, and* $q_1, q' \in Q_1$*, the algorithm* $\mathcal{B}$ *returns true if* $q_1 \in \mathsf{Pre}^*_{\mathbf{A}_1,\delta_1}(q_1')$ *and false otherwise.*

Let $\mathbf{A} = (Q, \widetilde{pred}, \widetilde{op}, q_i)$ *be an effective (not necessarily finite) automaton. Given a minor set* M_U *for an upward closed set* U*, there is an algorithm to decide if there is some* $q \in \mathsf{U}$ *such that the configuration* (q, d_i) *is reachable from the initial configuration* (q_i, d_i)*. Similarly there is an algorithm to decide if there is some* $q \in \mathsf{U}$ *and some* $d \in \mathsf{D}$ *such that* (q, d) *is reachable from the initial configuration* (q_i, d_i)*.*

Proof. Let α be the ranking function effectively compatible with δ. Let (δ_a, δ_b) be the α-compatible splitting of δ. In order to decide whether $q_i \in \mathsf{Pre}^*_{\mathsf{A},\,\delta,d_i}(\mathsf{U})$, we construct the increasing sequence $\mathsf{U}_0 \subseteq \mathsf{U}_1 \subseteq \mathsf{U}_2, \ldots$ such that $\mathsf{U}_0 = \mathsf{U}$ and $\mathsf{U}_{r+1} = \mathsf{U}_r \cup \mathsf{Pre}^*_{\mathbf{A},\,\delta_a,d_i}(\mathsf{U}_r \cup \delta_b^{-1}(\mathsf{U}_r, p_i, id))$ where id is the identity function.

As (δ_a, δ_b) is an α-compatible splitting of δ, any computation $(q, d_i) \rightarrow^*_\delta (q', d')$ is of the form $(q, d_i) \rightarrow^*_{\delta_a} (q_0, d_i) \rightarrow_{\delta_b} (q_1, d_i) \rightarrow^*_{\delta_a} (q_2, d_i) \rightarrow_{\delta_b} \ldots \rightarrow_{\delta_b} (q_n, d_i) \rightarrow^*_{\delta_a} (q', d')$ for some $q_0, q_1, \ldots q_n \in \mathsf{Q}$. Thus, $\mathsf{Pre}^*_{\mathbf{A},\,\delta,d_i}(\mathsf{U}) = \bigcup_{r \geq 0} \mathsf{U}_r$. Again, as every increasing sequence of increasing upward closed sets must be eventually constant, we can terminate once we get $\mathsf{U}_r = \mathsf{U}_{r+1}$. Hence, the first question can now be answered by deciding whether $q_i \in \mathsf{U}_r$. The second question can be reduced to the first problem by considering $\mathsf{U}' = \mathsf{Pre}^*_{\mathbf{A},\,\delta_a}(\mathsf{U})$. □

Please note that the above proof is essentially different from the proof of decidability of the coverability problem in multi-set pushdown automata (MPDS) given in [33]. While we perform a backward-reachability analysis, the proof in [33] constructs a sequence of over-approximations to rule out unreachable states and a sequence of under-approximations to discover the reachable states. Furthermore, the proof in [33] relies essentially on the fact that the w.q.o. is formed by products of linear orders (in there case products of $\mathbb{N}$) and is not extendible to a general w.q.o.. However, as in our case, the rank decreasing functions (decrementing of counter) are constrained and occur only when the pushdown stack is empty. We discuss the relation between the proofs in detail in Section A.

5 Applications

We will now present many natural examples of w.q.o. automata. An immediate of consequence of Theorem 1, will be decidability results for safety verification for these systems. The examples that we present here, have as data store either a pushdown stack, or some variant of it, or a higher-order pushdown stack. Therefore, when presenting these examples, we will use more standard notation for such stacks, rather than define them formally in terms of the domain, test predicates and operations. This mainly to make the examples easy to follow, and not clutter them with a lot of notation.

5.1 Multi-set pushdown system

Multi-set pushdown automata have been introduced in the literature [51, 33] as models of asynchronous programs that may make asynchronous procedure calls which are not immediately executed but stored and scheduled only after a recursive computation is completed.

Definition 10 (Multi-set pushdown system). *A Multi-set pushdown system (MPDS) is a tuple* $\mathbf{B} = (S, \Gamma, \Delta_{\mathsf{push}}, \Delta_{\mathsf{pop}}, \Delta_{\mathsf{cr}}, \Delta_{\mathsf{rt}}, s_0)$*, where* S *is a finite set of states,* Γ *is a finite set of stack and multi-set symbols,* $\Delta_{\mathsf{push}}, \Delta_{\mathsf{pop}}, \Delta_{\mathsf{cr}}, \Delta_{\mathsf{rt}} \subseteq S \times \Gamma \times S$ *together form the transition relation, and* s_0 *is the initial state.*

The semantics of an MPDS $\mathbf{B}$ is defined in terms of a transition system as follows. A configuration C of $\mathbf{B}$ is a tuple $(s, w, B) \in S \times \Gamma^* \times B[\Gamma]$ where Γ^* is the set of all finite strings over Γ and $B[\Gamma]$ is the set of all multi-sets over Γ. The initial configuration of $\mathbf{B}$ is $(s_0, \epsilon, \emptyset)$ where ϵ is the empty string and $\emptyset$ is the empty multi-set. The transition relation $\Rightarrow$ on configurations is given as a union of four relations, $\Rightarrow_{\mathsf{push}}$, $\Rightarrow_{\mathsf{pop}}$, $\Rightarrow_{\mathsf{cr}}$ and $\Rightarrow_{\mathsf{rt}}$ defined as follows: $(s, w, B) \Rightarrow_{\mathsf{push}} (s', w\gamma', B)$ iff $(s, \gamma', s') \in \Delta_{\mathsf{push}}$, $(s, w\gamma, B) \Rightarrow_{\mathsf{pop}} (s', w, B)$ iff $(s, \gamma, s') \in \Delta_{\mathsf{pop}}$, $(s, w, B) \Rightarrow_{\mathsf{cr}} (s', w, B \cup \{\gamma'\})$ iff $(s, \gamma', s') \in \Delta_{\mathsf{cr}}$ and $(s, \epsilon, B \cup \{\gamma\}) \Rightarrow_{\mathsf{rt}} (s', \epsilon, B)$ iff $(s, \gamma, s') \in \Delta_{\mathsf{rt}}$. Please note that the relation Δ_{rt} assume that the stack is empty and does not change the contents of the stack. The relation $\Rightarrow^*$ is defined as the reflexive transitive closure of $\Rightarrow$.

The ordering relation $\preceq$ on $S \times B[\Gamma]$ is defined as $(s, B) \preceq (s_1, B_1)$ iff $s = s_1$ and B is a sub-multi-set of B_1; this is known to be a well-quasi-order [21]. Using this fact, any MPDS can be seen as an instance of an *wqo* automaton with $S \times B[\Gamma]$ as the set of control states and the pushdown stack as the storage. This *wqo* automaton can be turned into an effective w.q.o. automaton by using the cardinality of the multi-set B as the ranking function. Therefore, Theorem 1 yields the following result.

Theorem 2 (Coverability of MPDS). *Given a MPDS* $\mathbf{B} = (S, \Gamma, \Delta_{\mathsf{push}}, \Delta_{\mathsf{pop}}, \Delta_{\mathsf{cr}}, \Delta_{\mathsf{rt}}, s_0)$ *and* $s \in S$*, the problem of checking whether there exist* $B \in B[\Gamma]$ *and* $w \in \Gamma^*$ *such that* $(s_0, \epsilon, \emptyset) \Rightarrow^* (s, w, B)$ *is decidable.*

Proof. An MPDS $\mathbf{B}$ can be easily seen as a w.q.o. automaton $\mathbf{A}$ as follows. As discussed in Section 3.1 the pushdown stack can be seen as a data structure as follows. The set Γ^* is the set of data values with the empty string ϵ as the initial value. The set of predicates $\widetilde{pred}$ can be chosen as $\{\mathsf{empty}\} \cup \{\mathsf{top}_\gamma \mid \gamma \in \Gamma\} \cup \{\mathsf{any}\}$, where $p_i = \{\epsilon\}$ (the initial predicate), $\mathsf{top}_\gamma = \{w\gamma \mid w \in \Gamma^*\}$ (the top of stack is γ) and $\mathsf{any} = \Gamma^*$ (any stack). The set of functions $\widetilde{op}$ can be defined as $\{id\} \cup \{\mathsf{push}_\gamma \mid \gamma \in \Gamma\} \cup \{\mathsf{pop}_\gamma \mid \gamma \in \Gamma\}$ where push_γ and pop_γ are defined as follows. For all $w \in \Gamma^*$, $\mathsf{push}_\gamma(w) = w\gamma$ and $\mathsf{pop}_\gamma(w) = w_1$ if $w = w_1\gamma$ and w otherwise.

The set of control states Q can be defined as $S \times B[\Gamma]$ which forms a $\mathsf{w.q.o.}$ under the ordering $(s, B) \preceq (s_1, B_1)$ iff $s = s_1$ and B_1 is a sub-multi-set of B_2. The transition relation δ is the union of δ_{push}, δ_{pop}, δ_{rt} and δ_{cr} where $\delta_{\mathsf{push}}, \delta_{\mathsf{pop}}, \delta_{\mathsf{rt}}$ and δ_{cr} are defined as follows. The set δ_{push} contains exactly $((s, B), \mathsf{any}, \mathsf{push}_{\gamma'}, (s', B))$ iff $(s, (s', \gamma')) \in \Delta_{\mathsf{push}}$. The set δ_{pop} contains exactly $((s, B), \top_\gamma, \mathsf{pop}_\gamma, (s', B))$ iff $((s, \gamma), s') \in \Delta_{\mathsf{pop}}$. The set δ_{cr} contains exactly $((s, B), \mathsf{any}, id, (s', B \cup \{\gamma'\}))$ iff $(s, (s', \gamma')) \in \Delta_{\mathsf{cr}}$. The set δ_{rt} contains exactly $((s, B \cup \{\gamma\}), \mathsf{empty}, id, (s', B))$ iff $((s, \gamma), s') \in \Delta_{\mathsf{rt}}$. Now, it can be easily shown that $((s_0, \emptyset), \epsilon,) \rightarrow^*_{\mathbf{A},\delta} ((s, B), w)$ iff $(s, \epsilon, \emptyset) \Rightarrow^* (s, w, B)$.

Consider the ranking function $\alpha : S \times B[T] \rightarrow \mathbb{N}$ defined as $\alpha((s, B)) = |B|$ where $|B|$ denotes the cardinality of multi-set B. Now it can be easily shown that α is effectively compatible with δ. The pair $(\delta_{\mathsf{push}} \cup \delta_{\mathsf{pop}} \cup \delta_{\mathsf{cr}}, \delta_{\mathsf{rt}})$ forms the α-compatible splitting. The *wqo* automaton $\mathbf{A}$ is now easily shown to be effective. The result then follows from Theorem 1. $\square$

The above result was proved in [51] using Parikh's theorem and in [33] using over-approximations and under-approximations; Theorem 1 yields a different and more general proof of this result.

5.2 Multi-set higher-order pushdown systems

Higher-order pushdown systems [18, 36, 16, 14] (HPDS) characterize safe higher-order recursive program schemes [36] and generalize the standard pushdown systems. The pushdown stack of a HPDS is sequence of stacks and is called a *higher-level store*. Stores of *level* 1 are sequences of letters (the standard pushdown stack) and stores of level $j + 1$ are sequence of stores of level j. The operations allowed on a level n store are $push_j$ and pop_j operations for each level $j \leq n$. For $j = 1$, the $push_1$ and pop_1 operations are the usual push and pop operations on the top-most level 1 store. For $2 \leq j \leq n$ the operation $push_j$ operation duplicates the top-most level j store while the operation pop_j deletes the top-most level j store.

The definition of the multi-set pushdown system (Definition 10) can be generalized as follows. Given a finite set of symbols Γ, let $O_n(\Gamma)$ be the set of level n stores on Γ and $I_n(\Gamma)$ be the set of level n operations on O_n stores.

Definition 11 (Multi-set higher-order pushdown system). *A order n multi-set pushdown system (n-MPDS) is a tuple $(S, \Gamma, \Delta_{\mathsf{st}}, \Delta_{\mathsf{cr}}, \Delta_{\mathsf{rt}}, s_0)$ where S is a finite set of stores, Γ a finite set of symbols, $\Delta \subseteq S \times I_n \times S$, $\Delta_{\mathsf{cr}}, \Delta_{\mathsf{rt}} \subseteq S \times \Gamma \times S$ define the transition relation, and s_0 is the initial state.*

The semantics of an n-MPDS $\mathbf{B}$ is defined in terms of a transition system as follows. A configuration C of $\mathbf{B}$ is a tuple $(s, St, B) \in S \times O_n(\Gamma) \times B[\Gamma]$ where $B[\Gamma]$ is the set of all multi-sets over Γ as before. The initial configuration of $\mathbf{B}$ is $(s_0, \perp, \emptyset)$ where $\perp$ is the empty store. The transition relation $\Rightarrow$ on configurations is given as a union of three relations, $\Rightarrow_{\mathsf{st}}$, $\Rightarrow_{\mathsf{cr}}$ and $\Rightarrow_{\mathsf{rt}}$ defined in the following way. We say that $(s, St, B) \Rightarrow_1 (s', g(St), B)$ iff $(s, g, s') \in \Delta_{\mathsf{st}}$, $(s, St, B) \Rightarrow_{\mathsf{cr}} (s', St, B \cup \{\gamma'\})$ iff $(s, \gamma', s')) \in \Delta_{\mathsf{cr}}$ and $(s, \perp, B \cup \{\gamma\}) \Rightarrow_{\mathsf{rt}} (s', \perp, B)$ iff $(s, \gamma, s') \in \Delta_{\mathsf{rt}}$. The relation $\Rightarrow^*$ is defined as the reflexive transitive closure of $\Rightarrow$. As in the case of a MPDS, we have the following.

Theorem 3 (Coverability of n-MPDS). *Given a n-MPDS $\mathbf{B} = (S, \Gamma, \Delta_{\mathsf{st}}, \Delta_{\mathsf{cr}}, \Delta_{\mathsf{rt}}, s_0)$ and $s \in S$, the problem of checking whether there exist $B \in B[\Gamma]$ and $St \in O_n$ such that $(s_0, \perp, \emptyset) \Rightarrow^* (s, St, B)$ is decidable.*

5.3 Timed multi-set pushdown system

Asynchronous programming forms the basis of event-driven languages that are used to describe networks of embedded systems [28, 31]. Such systems have real-time constraints, in addition to synchronous and asynchronous procedure calls. Though the asynchronous procedure calls are postponed till the end of the execution of the current recursive procedure call, they are scheduled only if they have not passed some real-time deadline. We model this by augmenting a pushdown system with real-time clocks. When a asynchronous procedure is called, a clock is added to the multi-set and set to 0. We assume that all clocks proceed at the same rate. Once the current recursive computation is completed, the next job is scheduled only if the associated clock is within some interval bounded by integers (we will assume that ∞ is an integer for this case). The results can easily be generalized to the case when the time constraints are rationals rather than integers.

Definition 12. *[Timed multi-set pushdown automata] A Timed multi-set pushdown system (TMPDS) is a tuple $\mathbf{B} = (S, \Gamma, \Delta_{\mathsf{push}}, \Delta_{\mathsf{pop}}, \Delta_{\mathsf{cr}}, \Delta_{\mathsf{rt}}, s_0)$, where S is a finite set of states, Γ is a finite set of stack and multi-set symbols, the sets $\Delta_{\mathsf{push}}, \Delta_{\mathsf{pop}}, \Delta_{\mathsf{cr}} \subseteq S \times \Gamma \times S$, and $\Delta_{\mathsf{rt}} \subseteq (S \times \Gamma \times \mathbb{N} \times \mathbb{N} \cup \{\infty\}) \times S$ are finite and together form the transition relation, and s_0 is the initial state.*

The semantics of an MPDS $\mathbf{B}$ is defined in terms of a transition system as follows. A configuration C of $\mathbf{B}$ is a tuple $(s, w, B) \in S \times \Gamma^* \times B[\Gamma \times \mathbb{R}^+]$ where Γ^* is the set of all finite strings over Γ, $\mathbb{R}^+$ is the set of positive real numbers and $B[\Gamma \times \mathbb{R}^+]$ is the set of all multi-sets over $\Gamma \times \mathbb{R}^+$. The initial configuration of $\mathbf{B}$ is $(s_0, \epsilon, \emptyset)$. The transition relation $\Rightarrow$ on configurations is given as a union of five relations, $\Rightarrow_{\mathsf{push}}$, $\Rightarrow_{\mathsf{pop}}$, $\Rightarrow_{\mathsf{cr}}$,$\Rightarrow_{\mathsf{rt}}$ and $\Rightarrow_T$.

The relation $\Rightarrow_{\mathsf{push}}$ is defined as $(s, w, B) \Rightarrow_{\mathsf{push}} (s', w\gamma', B)$ iff $(s, , \gamma', s') \in \Delta_{\mathsf{push}}$. The relation $\Rightarrow_{\mathsf{pop}}$ is defined as $(s, w\gamma, B) \Rightarrow_{\mathsf{pop}} (s', w, B)$ iff $(s, \gamma, s') \in \Delta_{\mathsf{pop}}$. The relation $\Rightarrow_{\mathsf{cr}}$ is defined as $(s, w, B) \Rightarrow_{\mathsf{pop}} (s', w, B \cup \{(\gamma', 0)\})$ iff $(s, \gamma', s') \in \Delta_{\mathsf{cr}}$. The relation $\Rightarrow_{\mathsf{rt}}$ is defined as $(s, \epsilon, B \cup \{(\gamma, t)\}) \Rightarrow_{\mathsf{rt}} (s', \gamma, B)$ iff there exist $n_1, n_2 \in \mathbb{N}$ such that $n_1 \leq t \leq n_2$ and $((s, \gamma, n_1, n_2), s') \in \Delta_{\mathsf{rt}}$. The relation $\Rightarrow_T$ is defined as $(s, w, B) \Rightarrow_T (s, w, B_t)$ for every $t \in \mathbb{R}^+$ where B_t is the multi-set obtained by replacing each $(\gamma, t') \in B$ by $(\gamma, t' + t) \in B$. Observe that elements are deleted from B only when the pushdown stack is empty. The relation $\Rightarrow^*$ is defined as the reflexive transitive closure of $\Rightarrow$.

We are once again interested in an algorithm which answers the question that given $s \in S$ whether there exists $B \in B[\mathit{\Gamma} \times \mathbb{R}^+]$ and $w \in \mathit{\Gamma}^*$ such that $(s_0, \epsilon, \emptyset) \Rightarrow^* (s, w, B)$. In order to apply Theorem 1, we define a well-quasi-order $\preceq$ on $B[\mathit{\Gamma} \times \mathbb{R}^+]$ which gives rise to regions [3] by symmetrizing the relation $\preceq$.

The relation $\preceq$ on $B[\mathit{\Gamma} \times \mathbb{R}^+]$ is defined as follows. Let $n_{\max}$ be the largest natural number occurring in $\mathit{\Delta}_{\mathsf{cr}}$. Given $t \in \mathbb{R}^+$, let $\lfloor t \rfloor$ denote the integer part of t and let $frac(t)$ denote the fractional part of t. Given $B_1, B_2 \in B[\mathit{\Gamma} \times \mathbb{R}^+]$, we say that $B_1 \preceq B_2$ iff there exists a one-to-one function $j : B_1 \to B_2$ such that the following hold.

– If $j((\gamma_1, t_1)) = (\gamma_2, t_2)$ then $\gamma_1 = \gamma_2$.
– If $j((\gamma, t_1)) = (\gamma, t_2)$ then $t_1 \geq n_{\max}$ iff $t_2 \geq n_{\max}$.
– If $j((\gamma, t_1)) = (\gamma, t_1)$, $t_1 < n_{\max}$ then $\lfloor t_1 \rfloor = \lfloor t_2 \rfloor$ and $frac(t_1) = 0$ iff $frac(t_2) = 0$.
– If $j((\gamma, t_1)) = (\gamma, t_2)$, $j((\gamma', t_1')) = (\gamma', t_2')$ and $t_1, t_1' < n_{\max}$ then $frac(t_1) \leq frac(t_1')$ iff $frac(t_2) \leq frac(t_2')$.

This relation $\preceq$ defines a well-quasi-order on $B[\mathit{\Gamma} \times \mathbb{R}^+]$ [3]. The relation $\preceq$ induces an equivalence relation on $B[\mathit{\Gamma} \times \mathbb{R}^+]$: $B \simeq B'$ iff $B \preceq B'$ and $B' \preceq B$. The set of equivalence classes under the relation $\simeq$ is said to be a *region*. Let $Reg(\mathit{\Gamma})$ be the set of all regions defined in this way and let $Reg(B)$ be the region that contains B. The well-quasi-order $\preceq$ extends to the set of regions in the standard way $Reg(B_1) \preceq_R Reg(B_2)$ iff $B_1 \preceq B_2$. Please note that the function $\alpha_R : Reg \to \mathbb{N}$ defined as $\alpha_R(R(B)) = |B|$, where $|B|$ is the cardinality of the multi-set B, is a ranking function.

The transition relation $\Rightarrow$ can be extended to a binary relation $\Rightarrow_R$ on $S \times \mathit{\Gamma}^* \times Reg[\mathit{\Gamma}]$ as follows. We say that $(s, w, Reg(B)) \Rightarrow_R (s', w', Reg(B'))$ iff $(s, w, B) \Rightarrow (s', w', B')$. The well-defineness of the relation $\Rightarrow_R$ can be shown using the standard techniques [3]. Thus, $(s_0, \epsilon, \emptyset) \Rightarrow^* (s, w, B)$ iff $(s_0, \epsilon, Reg(\emptyset)) \Rightarrow_R^*$ $(s, w, Reg[B])$ where $\Rightarrow_R^*$ is the reflexive transitive closure of $\Rightarrow_R$. Thus, using $S \times Reg(\mathit{\Gamma})$ as the set of control states, the pushdown stack as the data structure and α_R as the ranking function, we can prove the following.

Theorem 4 (Coverability of TMPDS). *Given an TMPDS* $\mathbf{B} = (S, \mathit{\Gamma}, \mathit{\Delta}_{\mathsf{push}}, \mathit{\Delta}_{\mathsf{pop}}, \mathit{\Delta}_{\mathsf{cr}},$
$\mathit{\Delta}_{\mathsf{rt}}, s_0)$ *and* $s \in S$, *the problem of checking whether there exist* $B \in B[\mathit{\Gamma} \times \mathbb{R}^+]$ *and* $w \in \mathit{\Gamma}^*$ *such that* $(s_0, \epsilon, \emptyset) \Rightarrow^* (s, w, B)$ *is decidable.*

5.4 Network of message passing recursive programs

We can apply our result to analyze a network of recursive programs which send requests to each other on uni-directional lossy FIFO queues. Each recursive program is modeled as a pushdown system which may generate new requests while executing a recursive computation. New requests are only handled when the recursive computation finishes. We shall assume that all recursive computations terminate. As a consequence of the fact that all computations terminate and that new requests are handled only when a recursive computation is completed, the transition relation is equivalent (for reachability concerns) to a transition system in which message loss happens only when all the pushdown stores are empty. For lack of space reasons, the model is discussed in detail in Appendix B.

6 Conclusions and future work

We considered the coverability problem of a w.q.o. automaton which are well-structured transition systems with an auxiliary store. Our main result is that if the control state reachability problem is decidable for finite w.q.o. automaton, then there is a decision procedure based on backward-reachability analysis for the coverability problem with infinitely many states if the w.q.o. automaton satisfies certain conditions. The main requirement is the existence of a ranking function compatible with the WSTS. Intuitively, the compatibility of the ranking function ensures that rank decreasing functions only occur when the store is the same as the initial store. For the rank non-decreasing functions, the backward reachability is performed using the decision procedure for finite w.q.o. automaton. We showed that the decision procedure in the paper can be utilized in several contexts such as higher-order asynchronous programs with asynchronous procedure calls, networked embedded systems and a network of message passing recursive systems.

We plan to implement the decision procedure in this paper and utilize it in model-checking the systems considered in this paper. A second line of research is to investigate whether other decision procedures that rely on w.q.o. theory such as the sub-covering problem can be extended to the framework of w.q.o. automaton.

References

1. P. A. Abdulla, K. Cerans, B. Jonsson, and Y.-K. Tsay. Algorithmic analysis of programs with well quasi-ordered domains. *Information and Computation*, 160(1):109–127, 2000.
2. P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. In *Proceedings of the IEEE Symposium on Logic in Computer Science*, pages 160–170, 1993.
3. P. A. Abdulla and B. Jonsson. Verifying networks of timed processes. In *Proceedings of the International Conference on Tools and Algorithms for Construction and Analysis of Systems*, pages 298–312, 1998.
4. P. A. Abdulla, B. Jonsson, M. Nilsson, and M. Saksena. A Survey of Regular Model Checking. In *Proceedings of the International Conference on Concurrency Theory*, pages 35–48, 2004.
5. P. A. Abdulla and A. Nyl'en. Better is better than Well: On efficient verification of infinite state systems. In *Proceedings of the IEEE Symposium on Logic in Computer Science*, pages 132–140, 2000.
6. P. A. Abdulla and A. Nyl'en. Timed petri nets and bqos. In *Proceedings of the International Conference on the Application and Theory of Petri Nets*, pages 53–70, 2001.

7. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
8. C. Bartzis and T. Bultan. Widening arithmetic automata. In *Proceedings of the International Conference on Computer Aided Verification*, pages 321–333, 2004.
9. B. Boigelot. *Symbolic Methods for Exploring Infinite State Spaces.* PhD thesis, Collection des Publications de la Faculté des Sciences Appliquées de l'Université de Liége, 1999.
10. A. Bouajjani, J. Esparza, S. Schwoon, and J. Strejcek. Reachability analysis of multithreaded software with asynchronous communication. In *Proceedings of the International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 348–359, 2005.
11. A. Bouajjani, B. Jonsson, M. Nilsson, and T. Touili. Regular model checking. In *Proceedings of the International Conference on Computer-Aided Verification*, pages 403–418, 2000.
12. A. Bouajjani, Y. Jurski, and M. Sighireanu. A generic framework for reasoning about dynamic networks of infinite-state processes. In *Proceedings of the International Conference on Tools and Algorithms for Construction and Analysis of Systems*, 2007.
13. A. Bouajjani, Y. Jurski, and M. Sighireanu. Reasoning about synamic networks of infinite-state processes with global synchronization. Technical report, LIAFA, 2007.
14. A. Bouajjani and A. Meyer. Symbolic reachability analysis of higher-order context-free processes. In *Proceedings of the International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 135–147, 2004.
15. A. Bouajjani, M. Müller-Olm, and T. Touili. Regular symbolic analysis of dynamic networks of pushdown systems. In *Proceedings of the International Conference on Concurrency Theory*, pages 473–487, 2005.
16. A. Carayol and S. Wohrle. The Caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata. In *Proceedings of the International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 112–123, 2003.
17. D. Caucal. On the regular structure of prefix rewriting. *Theoretical Computer Science*, 106:61–86, 1992.
18. D. Caucal. On infinite terms having a decidable monadic theory. In *Proceedings of the International Symposium on the Mathematical Foundations of Computer Science*, pages 165–176, 2002.
19. G. Cécé, A. Finkel, and S. Purushothaman Iyer. Unreliable channels are easier to verify than perfect channels. *Information and Computation*, 124(1):20–31, 1995.
20. R. Cunningham. Eel: Tools for debugging, visualization, and verification of event-driven software. Master's thesis, University of California, Los Angeles, 2005.
21. L. E. Dickson. Finiteness of the odd perfect and primitive abundant numbers with r distinct prime factors. *American Journal of Mathematics*, 35:413–422, 1913.
22. M. Emmi and R. Majumdar. Decision problems for the verification of real-time software. In *Proceedings of the International Workshop on Hybrid Systems: Computation and Control*, pages 200–211, 2006.
23. J. Esparza and K. Etessami. Verifying probabilistic procedural programs. In *Proceedings of the International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 16–31, 2004.
24. J. Esparza, A. Kucera, and R. Mayr. Model checking probabilistic pushdown automata. *Logical Methods in Computer Science*, 2(1), 2006.

25. K. Etessami and M. Yannakakis. Recursive markov chains, stochastic grammars, and monotone systems of nonlinear equations. In *Proceedings of the Symposium on the Theoretical Aspects of Computer Science*, pages 340–352, 2005.
26. A. Finkel, S. Purushothaman Iyer, and G. Sutre. Well-abstracted transition systems: Application to FIFO automata. *Information and Computation*, 181(1):1–31, 2003.
27. A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1–2):63–92, 2001.
28. D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesC language: A holistic approach to networked embedded systems. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 1–11, 2003.
29. S. Graf and H. Saidi. Construction of abstract state graphs with pvs. In *Proceedings of the International Conference on Computer Aided Verification*, pages 72–83, 1997.
30. P. Habermehl and T. Vojnar. Regular model checking using inference of regular languages. In *Proceedings of the Infinity Workshop*, 2004.
31. J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *Proceedings of the International Conference on Architectural support for Programming Languages and Operating Systems*, pages 93–104, 2000.
32. Allen Holub. *Taming Java Threads.* APress, 2000.
33. R. Jhala and R. Majumdar. Interprocedural analysis of asynchronous programs. In *Proceedings of the ACM Symposium on Principles of Programming Languages*, pages 339–350, 2007.
34. B. Jonsson and M. Nilsson. Transitive closures of regular relations for verifying infinite-state systems. In *Proceedings of the International Conference on Tools and Algorithms for Construction and Analysis of Systems*, pages 220–234, 2000.
35. C. Killian, J. W. Anderson, R. Braud, R. Jhala, and A. Vahdat. Mace: Language support for building distributed systems. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2007.
36. T. Knapik, D. Niwinski, and P. Urzyczyn. Higher-order pushdown trees are easy. In *Proceedings of the International Conference on Foundations of Software Science and Computation Structures*, pages 205–222, 2002.
37. O. Kouchnarenko and Ph. Schnoebelen. A model for recursive-parallel programs. In *Proceedings of the International Workshop on Verification of Infinite State Systems*, 1996.
38. J. B. Kruskal. The theory of well-quasi-ordering: A frequently discovered concept. *Journal of Combinatorial Theory: Series A*, 13(3):297–305, 1972.
39. G. Lafferriere, G. J. Pappas, and S. Sastry. O-minimal hybrid systems. *Mathematical Comtrol Signals Systems*, 13:1–21, 2000.
40. Libasync. http://pdos.csail.mit.edu/6.824-2004/async/.
41. Libevent. http://www.monkey.org/ provos/libevent/.
42. D. Lugiez and Ph. Schnoebelen. The regular viewpoint on PA-processes. *Theoretical Computer Science*, 274(1–2):89–115, 2002.
43. R. Majumdar. Personal communication.
44. R. Mayr. *Decidability and Complexity of Model Checking Problems for Infinite-State Systems.* PhD thesis, Technical University Munich, 1998.
45. F. Moller. Infinite results. In *Proceedings of the Conference on Concurrency Theory*, pages 195–216, 1996.
46. D. Muller and P. Schupp. The theory of ends, pushdown automata, and second-order logic. *Theoretical Computer Science*, 37:51–75, 1985.

47. J. L. Peterson. *Petri Net Theory and the Modelling of Systems.* Prentice Hall International, 1981.
48. S. Qadeer and J. Rehof. Context-bounded model checking of concurrent software. In *Proceedings of the International Conference on Tools and Algorithms for Construction and Analysis of Systems*, pages 93–107, 2005.
49. S. Qadeer and D. Wu. KISS: Keep it simple and sequential. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 14–24, 2004.
50. M. O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the american Mathematical Society*, 141:1–35, 1969.
51. K. Sen and M. Viswanathan. Model checking multithreaded programs with asynchronous atomic methods. In *Proceedings of the International Conference on Computer Aided Verification*, pages 300–314, 2006.
52. A. Tarski. *A Decision Method for Elementary Algebra and Geometry.* University of California Press, 1951.
53. W. Thomas. A short introduction to infinite automata. In *Proceedings of the International Conference on Developments on Language Theory*, pages 130–144, 2001.
54. W. Thomas. Constructing infinite graphs with a decidable MSO-theory. In *Proceedings of the International Symposium on the Mathematical Foundations of Computer Science*, pages 113–124, 2003.
55. A. Vardhan. *Learning to Verify Systems.* PhD thesis, University of Illinois, Urbana-Champaign, 2005.
56. K. Čerāns. Deciding properties of the integral relational automata. In *Proceedings of the International Colloquium on Automata, Languages and Programming*, pages 35–46, 1994.

A Relationship between two proofs

We describe briefly the relation between the proof of our main theorem 1 and the proof of the decidability of the coverability problem in multi-set pushdown automata (MPDS) given in [33]. Multi-set pushdown systems are automata with a finite set of control states, a pushdown stack and a multi-set over a finite set of symbols Γ. Elements are pushed/popped from the stack and added/deleted from the multi-set depending on the control states. The pairs (s, B) where s is a control state and B is the multi-set form a w.q.o. and hence can be seen a w.q.o. automaton. Furthermore, since Γ is finite, a multi-set B can be seen as an element of $\mathbb{N}_1 \times \mathbb{N}_2 \ldots \mathbb{N}_l$. Viewed in this manner, the maximum coordinate of B forms a ranking function. The first thing to note is that as in our w.q.o. automaton, rank-decreasing transitions in [33] only take place when the stack is empty (and this is crucial in that proof also).

The proof in [33] itself proceeds by considering over-approximations and under-approximations for each rank k. Unlike our case, however, the approximations involve both the rank non-decreasing and rank-decreasing functions. For over-approximation, the multi-set is not changed if the i-th coordinate crosses k. The under-approximations are used to enumerate the reachable states and the over-approximations are used to enumerate the unreachable states. A closer examination of the proof, however, reveals that the proof essentially relies on the following (and hence cannot be generalized for other w.q.o.'s).

- The w.q.o.in [33] is a product a finite set of states and well-orders (that is linearly ordered well-quasi-orders). This ensures that for any upward closed sets U_1 and U_2, the set $\mathsf{U}_1 \setminus \mathsf{U}_2$ can be written as a union of sets of the form $s \times A_1 \times \ldots \times A_r$ where either A_i is a singleton or an upward closed set of $\mathbb{N}$.
- A transition from (s, B) to (s', B') only effects one coordinate of B.
- For any transition from (s, B) to (s', B'), the difference in rank is bounded.

B Network of message passing recursive systems

We now consider network of message passing recursive system communicating over lossy FIFO nets. Each recursive program is modeled as a pushdown system which may send requests to other systems while a recursive computation is being executed. These requests are sent over uni-directional FIFO systems. Furthermore, we will assume that a system processes a new request only after the current request is completed.

Definition 13. *A network of n-message passing recursive systems (n-NMPRP) is a tuple $\mathbf{B} = ((S_j)_{0 \le j \le n}, \varGamma, (\varDelta_j)_{0 \le j \le n}, (\hat{s}_j)_{0 \le j \le n})$ where*

- *S_j is a finite set of states,*
- *$\varGamma$ is a finite set of stack symbols and messages,*
- *$\varDelta_j$ is a tuple $(\varDelta_{j,\mathsf{push}}, \varDelta_{j,\mathsf{pop}}, (\varDelta_{j,l,\mathsf{snd}})_{0 \le l \le n, l \ne j}, (\varDelta_{j,l,\mathsf{rcv}})_{0 \le l \le n, l \ne j})$ where $\varDelta_{j,\mathsf{push}}, \varDelta_{j,\mathsf{pop}}, \varDelta_{j,l,\mathsf{snd}}, \varDelta_{j,l,\mathsf{rcv}} \subseteq S_j \times \varGamma \times S_j$ together forms the transition relation of the j-th process, and*
- *$\hat{s}_j$ is the initial state of j-th process.*

The semantics of a n-NMPRP is defined in terms of a transition system. A configuration is a tuple $((s_j, w_j)_{0 \le j \le n}, (c_{j,l})_{0 \le j,l \le n, j \ne l})$ where $s_j \in S_j$ describes the state of system j, $w_j \in \varGamma^*$ describes the contents of the pushdown stack of system j and $c_{j,l} \in \varGamma^*$ describes the contents of uni-directional lossy FIFO channel from j to l. The initial configuration, C_i is $((\hat{s_j}, \epsilon)_{0 \le j \le n}, (\epsilon)_{0 \le j,l \le n, j \ne l})$. The transition relation $\Rightarrow$ is defined in terms of $\Rightarrow_j$ which describes the evolution of the process j and $\Rightarrow_{\mathsf{loss}}$ which describes message loss.

The transition relation $\Rightarrow_j$ itself consists of several relations as follows. Given $(s_j, \gamma', s'_j) \in \varDelta_{j,\mathsf{push}}$, the relation $\Rightarrow_{j,\mathsf{push}}$ changes (s_j, w_j) to $(s'_j, w_j\gamma')$ while keeping other components of the configuration unchanged. Given $(s_j, \gamma, s'_j) \in \varDelta_{j,pop}$, the relation $\Rightarrow_{j,\mathsf{pop}}$ changes $(s_j, w_j\gamma)$ to (s'_j, w_j) while keeping other components of the configuration unchanged. Given $(s_j, \gamma, s'_j) \in \varDelta_{j,l,\mathsf{snd}}$, the relation $\Rightarrow_{j,l,\mathsf{snd}}$ changes (s_j, w_j) to (s'_j, w_j) and $(c_{j,l})$, the contents of the FIFO queue from j to l, to $(c_{j,l}\gamma)$ while keeping other components of the configuration unchanged. Given $(s_j, \gamma, s'_j) \in \varDelta_{j,l,\mathsf{rcv}}$, the relation $\Rightarrow_{j,l,\mathsf{rcv}}$ changes (s_j, ϵ) to (s'_j, ϵ) and $(\gamma c_{l,j})$, the contents of the FIFO queue from l to j to $(c_{l,j})$ while keeping other components of the configuration unchanged. It will be useful to separate non-receiving and receiving transitions of a process j and towards this end, we define $\Rightarrow_{j,\mathsf{nrcv}}$ as the relation $\Rightarrow_{j,\mathsf{push}} \cup \Rightarrow_{j,\mathsf{pop}} \bigcup_{1\leq l\leq n, j\neq l} \Rightarrow_{j,l,\mathsf{snd}}$ and $\Rightarrow_{j,\mathsf{rcv}}$ as the relation $\bigcup_{1\leq l\leq n, j\neq l} \Rightarrow_{j,l,\mathsf{rcv}}$.

The transition relation $\Rightarrow_{\mathsf{loss}}$ is defined in terms of relation $\Rightarrow_{j,l,\mathsf{loss}}$ for $0 \leq j, l \leq n, j \neq l$. The relation $\Rightarrow_{j,l,\mathsf{loss}}$ changes the j, l-FIFO queue $c_{j,l}\gamma c'_{j,l}$ to $c_{j,l}c'_{j,l}$ while keeping the other components of the configuration unchanged.

We are interested in recursive systems in which all recursive computations terminate.

Definition 14. *A network of n message passing recursive terminating programs (n-NMPTP) is a n-NMPRP $\mathbf{B} = ((S_j)_{0\leq j\leq n}, \Gamma, (\Delta_j)_{0\leq j\leq n}, (\hat{s}_j)_{0\leq j\leq n})$ such that each of the relations $\Rightarrow_{m,\mathsf{nrcv}}, 1 \leq m \leq n$, generated by $\mathbf{B}$ satisfies for the following condition.*

For each configuration $C = ((s_j, w_j)_{0\leq j\leq n}, (c_{j,l})_{0\leq j,l\leq n, j\neq l})$ there exists a configuration $C' = ((s'_j, w'_j)_{0\leq j\leq n}, (c'_{j,l})_{0\leq j,l\leq n, j\neq l})$ such that $C \Rightarrow^*_{m,\mathsf{nrcv}} C'$ and w'_j is ϵ.

Let $\Rightarrow_{\mathsf{nrcv}}$ be the relation

$$\bigcup_{1\leq l\leq n} \Rightarrow_{l,\mathsf{rcv}}$$

and let $\Rightarrow_{\mathsf{rcv}}$ be the relation

$$\bigcup_{1\leq l\leq n} \Rightarrow_{l,\mathsf{rcv}} .$$

All recursive computations terminate in a n-NMPTP, and new requests are only addressed when a recursive computation is completed. Using these facts, we can ensure that every computation can be transformed into a computation in which all message losses happen just before the reception of messages. We have the following proposition.

Proposition 4. *Given a n-NMPTP $\mathbf{B}$, let C_ϵ be the set of all configurations in which all stack contents are empty. Then, given $C_i \Rightarrow^* C'$ where C_i is the initial configuration. there exist $C_0, C_1, C_2, C_3, \ldots, C_m, C_{m+1}$ such that $C_{m+1} = C'$, $C_r \in C_\epsilon$ for all $r < m$ and $C_0 \Rightarrow^*_{\mathsf{nrcv}} C_1 \Rightarrow^*_{\mathsf{loss}} C_1 \Rightarrow^*_{\mathsf{rcv}} C_2 \Rightarrow^*_{\mathsf{nrcv}} \cdots \Rightarrow^*_{\mathsf{nrcv}} C_m \Rightarrow^*_{\mathsf{loss}} C_{m+1}$.*

Please note that as no messages are during a recursive computation, the relation $\Rightarrow^*_{\mathsf{nrcv}}$ can itself be broken down into n-steps in which only one stack is active.

Proposition 5. *If $C \Rightarrow^*_{\mathsf{nrcv}} C'$ such $C, C' \in C_\epsilon$ that and then there exists $C_0, C_2, C_3, \ldots C_n$ such that $C_r \in C_\epsilon$ for all $0 \leq r \leq n$, $C_0 \in C$, $C_m = C'$ and $C_0 \Rightarrow^*_{1,\mathsf{nrcv}} C_1 \Rightarrow^*_{2,\mathsf{nrcv}} C_2 \Rightarrow^*_{2,\mathsf{nrcv}} \Rightarrow^*_{2,\mathsf{nrcv}} \cdots \Rightarrow^*_{n,\mathsf{nrcv}} C_n$.*

Hence, in order to answer the question whether some states $s_j \in S_j$ is reached with empty stack contents, any NMTP can be viewed as a pushdown system with multiple (lossy) message queues where message loss and message receives only occur at the end of a recursive computation. Clearly the condition on empty stack contents in the desired configuration is not a serious restriction since we can always add new states to the automaton which empties the stacks when the states s_j are reached.

Therefore, the NMPTP can be viewed equivalently (for reachability concerns) as a *wqo* automaton with a stack as the data storage and the set of control states Q as the set of tuples $((s_j)_{1\leq j\leq n}, (c_{j,l})_{1\leq l,j\leq n, l\neq j})$ where $s_j \in S_j$ and $c_{j,l} \in \Gamma^*$ which models the contents of the FIFO queue from j to l. Given $q = ((s_j)_{1\leq j\leq n}, (c_{j,l})_{1\leq l,j\leq n,l\neq j})$ and $q' = ((s'_j)_{1\leq j\leq n}, (c'_{j,l})_{1\leq l,j\leq n,l\neq j})$, we say that $q \preceq q'$ if $s_j = s'_j$ and $c_{j,l}$ is a substring of c'_{jl}. The relation $\preceq$ defines a well-quasi-order on Q. Using the total number of messages, *i.e.*, the sum of lengths of $c_{i,j}$ as a ranking function, we obtain the following.

Theorem 5. *Given a n-NMPTP* $\mathbf{B} = ((S_j)_{0\leq j\leq n}, \Gamma, (\Delta_j)_{0\leq j\leq n}, (\hat{s}_j)_{0\leq j\leq n})$, $s_1 \in S_1, s_2 \in S_2, \ldots s_n \in S_j$ *there is an algorithm to decide if there exists a configuration* C *such that* $C_0 \rightarrow^* C$ *where* C_0 *is the initial configuration and* C *is configuration such that the* j*-th state of the process* j *in* C *is* s_j.