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 [5,3,25] or to find bugs [23,6], as well as identifying special classes of infinite state systems (and properties) for which model checking can be shown to be decidable [22,21,4,11,7,10,24].

[*] Supported by DARPA/AFOSR MURI award F49620-02-1-0325 and NSF CCF 0429639.

[**] Supported by NSF CCF 0448178 and NSF CCF 0509321.

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)* [17] (or stronger notions like *better-quasi-orders* [4]). 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 [1,11]. Transition systems with a wqo simulation relation are called *well-structured transition systems (WSTS).* Examples of WSTSs can be found in [1,11].

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 (predefined) 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. We present herein two applications of the result, while some others may be found in [8]. First we consider asynchronous programs [15,12,14,18,19], 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 [13]) 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 [24,16], 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 [7]. Such automata can be used to model asynchronous programs, where asynchronous procedures can be more generally (safe) higher-order recursive programs, 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 [12,14], 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.

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. For lack of space reasons, we shall omit the proofs which may be found in an accompanying technical report [8].

[1] A general program can always be abstracted using techniques such as predicate abstraction [13] to obtain such restricted programs.

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 [24], where we considered automata with multi-sets and stacks to model asynchronous programs. The decidability of the control state reachability problem was proved using $\mathsf{w.q.o.}$ theory and Parikh's theorem. In [16], Jhala and Majumdar, simplified the proof, 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 [12,14], which are real-time asynchronous programs, and the proof techniques in [24,16] do not generalize to such a model. We discuss the differences between our proof approach and then one in [16] in more detail, when we present the main theorem.

Another very closely related work is the paper by Emmi and Majumdar [9]. One of the main observations concerns $\mathsf{w.q.o.}$ 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 sub-covering problem for such automata. Unfortunately, the proof presented in the paper is incorrect [20]. The authors conjecture that the decidability result for $\mathsf{w.q.o.}$ 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.

2 Preliminaries

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.

Given a $\mathsf{w.q.o.}$ $(\mathsf{Q}, \preceq)$ and $\mathsf{Q}' \subseteq \mathsf{Q}$, we say that $\mathsf{M}_{Q'} \subseteq \mathsf{Q}'$ is a *minor set* for Q' if 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.

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 $\mathsf{U} = \{q \in \mathsf{U} \,|\, \exists q_m \in \mathsf{M}_U \text{ s.t. } q_m \preceq q\}$. Also any subset $\mathsf{Q}' \subseteq \mathsf{Q}$ determines an upward closed set, $\mathsf{U}_{\mathsf{Q}'} = \{q \,|\, \exists q' \in Q' \text{ s.t. } q' \preceq q\}$. The following important observation follows from $\mathsf{w.q.o.}$ theory.

Proposition 1. *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 $l \geq j$.*

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 $\mathsf{rank}(q) = sup(\{\mathsf{rank}(q') \,|\, q' \prec q\}) + 1$ 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*

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

$f : D \to D$, $\widetilde{pred}$ 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.

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 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 functions pop_γ will be enabled only when the store satisfies top_γ and any respectively.

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 formalize 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.

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 a *wqo* automaton in literature is multi-set pushdown automata [24,16] 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 $(q_0, d_0), (q_1, d_1), \ldots (q_n, d_n)$ such that $q_0 = q, d_0 = d$, $q_n = q', d_n = d'$ and for every $0 \leq r < n$ $(q_r, d_r) \rightarrow_{\delta_0} (q_{r+1}, d_{r+1})$. We shall write $(q, d) \rightarrow^*_{\delta_0} (q', d')$ if there is some $j \geq 0$ 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 transition $\delta_0 \subseteq \delta$, 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 $\alpha_{\max}(\delta_0^{-1}(p, g, \mathsf{U})) \leq \alpha_{\max}(\mathsf{U})$.*[3]

As mentioned in the introduction, we will consider special w.q.o. automata, namely those in which the only rank decreasing transitions 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

[3] Please note that our definition of rank-decreasing functions is different from standard one.

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 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 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 $\mathsf{w.q.o.}$ 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 the coverability problem is equivalent to deciding whether the reflexive transitive closure $(\delta^{-1})^{*}$ of the relation δ^{-1} contains q_i or not. A backward reachability analysis is performed, and 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 $\mathsf{w.q.o.}$ must be eventually constant, we terminate once $\mathsf{U}_j = \mathsf{U}_{j+1}$.

Our approach to the coverability problem for a $\mathsf{w.q.o.}$ automaton will follow a similar approach. Given a $\mathsf{w.q.o.}$ 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 carefully in order to capture 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 approximation). *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* $(q, d) \rightarrow^*_{\delta_a} (q'_1, d_1)$.

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* $(q_1, d) \rightarrow^*_{\delta_k} (q_0, d')$.

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 $\mathsf{Pre}^*_{\mathbf{A}_k, \delta_k}(U)$ are the minor sets for $\mathsf{Pre}^*_{\mathbf{A}_k, \delta_k, d_i}(\mathsf{Q}')$ and $\mathsf{Pre}^*_{\mathbf{A}_k, \delta_k}(Q')$ respectively. 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* $\alpha : \mathsf{Q} \rightarrow \mathbb{N}$ *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.*[4]

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 Lemma 1 and Lemma 2 ensure 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.

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).

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

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 $(q, d_i) \to^*_\delta (q', d')$ is of the form $(q, d_i) \to^*_{\delta_a} (q_0, d_i) \to_{\delta_b} (q_1, d_i) \to^*_{\delta_a} (q_2, d_i) \to_{\delta_b} \ldots \to_{\delta_b} (q_n, d_i) \to^*_{\delta_a} (q', d')$ 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 [16]. While we perform a backward-reachability analysis, the proof in [16] constructs a sequence of over-approximations to rule out unreachable states and a sequence of under-approximations to discover the reachable states. The proof in [16] relies essentially on the fact that the w.q.o. is formed by products of linear orders (in that case products of $\mathbb{N}$) and cannot be extended 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 [8].

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. 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 [24,16] 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* $\mathbf{B} = (S, \varGamma, \varDelta_{\mathsf{push}}, \varDelta_{\mathsf{pop}}, \varDelta_{\mathsf{cr}}, \varDelta_{\mathsf{rt}}, s_0)$*, where* S *is a finite set of states,* $\varGamma$ *is a finite set of stack and multi-set symbols,* $\varDelta_{\mathsf{push}}, \varDelta_{\mathsf{pop}}, \varDelta_{\mathsf{cr}}, \varDelta_{\mathsf{rt}} \subseteq S \times \varGamma \times S$ *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 $(s, w, B) \in S \times \varGamma^* \times B[\varGamma]$ where $\varGamma^*$

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. 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.*

The above result was proved in [24] using Parikh's theorem and in [16] using over-approximations and under-approximations; Theorem 1 yields a different and more general proof of this result. A similar result can be obtained when the asynchronous procedures are (safe) higher-order recursive procedures [8].

5.2 Timed Multi-set Pushdown System

Asynchronous programming forms the basis of event-driven languages that are used to describe networks of embedded systems [12,14]. 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 assume that ∞ is an integer for this case). The results can easily be generalized to the case when the bounds are rationals rather than integers.

Definition 11. *[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 $B \in B[\Gamma \times \mathbb{R}^+]$ and $w \in \Gamma^*$ such that $(s_0, \epsilon, \emptyset) \Rightarrow^* (s, w, B)$. In order to apply Theorem 1, we define a well-quasi-order $\preceq$ on $B[\Gamma \times \mathbb{R}^+]$ which gives rise to regions [2] by symmetrizing the relation $\preceq$.

The relation $\preceq$ on $B[\Gamma \times \mathbb{R}^+]$ is defined as follows. Let $n_{\max}$ be the largest natural number occurring in Δ_{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 $B_1, B_2 \in B[\Gamma \times \mathbb{R}^+]$, 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 $B[\Gamma \times \mathbb{R}^+]$ [2]. The relation $\preceq$ induces an equivalence relation on $B[\Gamma \times \mathbb{R}^+]$: $B \simeq B'$ iff $B \preceq B'$ and $B' \preceq B$. An equivalence class under the relation $\simeq$ is said to be a *region*. Let $Reg(\Gamma)$ 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: $Reg(B_1) \preceq_R Reg(B_2)$ iff $B_1 \preceq B_2$. 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 $S \times \Gamma^* \times Reg[\Gamma]$ 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 [2]. 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 $S \times Reg(\Gamma)$ 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 3 (Coverability of TMPDS). *Given an TMPDS* $\mathbf{B} = (S, \Gamma, \Delta_{\mathsf{push}}, \Delta_{\mathsf{pop}}, \Delta_{\mathsf{cr}}, \Delta_{\mathsf{rt}}, s_0)$ *and* $s \in S$*, the problem of checking whether there exist* $B \in B[\Gamma \times \mathbb{R}^+]$ *and* $w \in \Gamma^*$ *such that* $(s_0, \epsilon, \emptyset) \Rightarrow^* (s, w, B)$ *is decidable.*

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 transitions only occur when the store is the same as the initial store. For the rank non-decreasing transitions, 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 the contexts of asynchronous procedure calls and networked embedded systems.

We plan to implement the decision procedure in this paper and utilize it for 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. Abdulla, P.A., Cerans, K., Jonsson, B., Tsay, Y.-K.: Algorithmic analysis of programs with well quasi-ordered domains. Information and Computation 160(1), 109–127 (2000)
2. Abdulla, P.A., Jonsson, B.: Verifying networks of timed processes. In: Proceedings of the International Conference on Tools and Algorithms for Construction and Analysis of Systems, pp. 298–312 (1998)
3. Abdulla, P.A., Jonsson, B., Nilsson, M., Saksena, M.: A Survey of Regular Model Checking. In: Proceedings of the International Conference on Concurrency Theory, pp. 35–48 (2004)
4. Abdulla, P.A., Nyl'en, A.: Better is better than Well: On efficient verification of infinite state systems. In: Proceedings of the IEEE Symposium on Logic in Computer Science, pp. 132–140. IEEE Computer Society Press, Los Alamitos (2000)
5. Boigelot, B.: Symbolic Methods for Exploring Infinite State Spaces. PhD thesis, Collection des Publications de la Faculté des Sciences Appliquées de l'Université de Liége (1999)
6. Bouajjani, A., Esparza, J., Schwoon, S., Strejcek, J.: Reachability analysis of multithreaded software with asynchronous communication. In: Ramanujam, R., Sen, S. (eds.) FSTTCS 2005. LNCS, vol. 3821, pp. 348–359. Springer, Heidelberg (2005)
7. Carayol, A., Wohrle, S.: 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, pp. 112–123 (2003)

8. Chadha, R., Viswanthan, M.: Decidability results for well-structured transition systems with auxiliary storage. Technical Report UIUCDCS-R-2007-2865, Univ. of Illiniois at Urbana-Champaign (2007)
9. Emmi, M., Majumdar, R.: Decision problems for the verification of real-time software. In: Hespanha, J.P., Tiwari, A. (eds.) HSCC 2006. LNCS, vol. 3927, pp. 200–211. Springer, Heidelberg (2006)
10. Esparza, J., Etessami, K.: Verifying probabilistic procedural programs. In: Lodaya, K., Mahajan, M. (eds.) FSTTCS 2004. LNCS, vol. 3328, pp. 16–31. Springer, Heidelberg (2004)
11. Finkel, A., Schnoebelen, Ph.: Well-structured transition systems everywhere! Theoretical Computer Science 256(1–2), 63–92 (2001)
12. Gay, D., Levis, P., von Behren, R., Welsh, M., Brewer, E., Culler, D.: The nesC language: A holistic approach to networked embedded systems. In: Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation, pp. 1–11. ACM Press, New York (2003)
13. Graf, S., Saidi, H.: Construction of abstract state graphs with pvs. In: Grumberg, O. (ed.) CAV 1997. LNCS, vol. 1254, pp. 72–83. Springer, Heidelberg (1997)
14. Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., Pister, K.: System architecture directions for networked sensors. In: Proceedings of the International Conference on Architectural support for Programming Languages and Operating Systems, pp. 93–104 (2000)
15. Holub, A.: Taming Java Threads. APress (2000)
16. Jhala, R., Majumdar, R.: Interprocedural analysis of asynchronous programs. In: Proceedings of the ACM Symposium on Principles of Programming Languages, pp. 339–350. ACM Press, New York (2007)
17. Kruskal, J.B.: The theory of well-quasi-ordering: A frequently discovered concept. Journal of Combinatorial Theory: Series A 13(3), 297–305 (1972)
18. Libasync. `http://pdos.csail.mit.edu/6.824-2004/async/`
19. Libevent. `http://www.monkey.org/provos/libevent/`
20. Majumdar, R.: Personal communication
21. Mayr, R.: Decidability and Complexity of Model Checking Problems for Infinite-State Systems. PhD thesis, Technical University Munich (1998)
22. Moller, F.: Infinite results. In: Proceedings of the Conference on Concurrency Theory, pp. 195–216 (1996)
23. Qadeer, S., Rehof, J.: Context-bounded model checking of concurrent software. In: Proceedings of the International Conference on Tools and Algorithms for Construction and Analysis of Systems, pp. 93–107 (2005)
24. Sen, K., Viswanathan, M.: Model checking multithreaded programs with asynchronous atomic methods. In: Ball, T., Jones, R.B. (eds.) CAV 2006. LNCS, vol. 4144, pp. 300–314. Springer, Heidelberg (2006)
25. Vardhan, A.: Learning to Verify Systems. PhD thesis, University of Illinois, Urbana-Champaign (2005)