Algorithmic Computation of Strongest Postconditions of Services as Priced Timed Automata

Aida Čaušević, Cristina Seceleanu, and Paul Pettersson

Mälardalen Real-Time Research Centre (MRTC),
Mälardalen University, Västerås, Sweden
[aida.delic,cristina.seceleanu,paul.pettersson]@mdh.se

Abstract. Service-Oriented Systems (SOS) have gained importance in different application domains thanks to their ability to enable reusable functionality provided via well-defined interfaces, and the increased opportunities to compose existing units, called services, into various configurations. Developing applications in such a setup, by reusing existing services, brings some concerns regarding the assurance of the expected Quality-of-Service (QoS), and correctness of the employed services. In this paper, we provide a formal mechanism of computing service guarantees, automatically. We assume service models annotated with pre- and postconditions, their semantics given as Priced Timed Automata (PTA), and the forward analysis method for checking the service correctness w.r.t. given requirements. Under these assumptions, we show how to compute the strongest postcondition of the corresponding automata algorithmically, with respect to the specified precondition. The approach is illustrated on a small example of a service modeled as Priced Timed Automaton (PTAn).

1 Introduction

The complexity of software systems has been continuously increasing during the last decade. One of the reasons underlying such phenomenon is a new trend that aims to integrate and connect heterogeneous applications and available resources under the requirement of improved software reusability. Service-oriented systems (SOS), which have emerged as context independent component-based systems (CBS), are becoming one of the dominant paradigms for designing, implementing, and developing large scale systems out of self-contained and loosely coupled services. Among the main benefits of the approach, the most appealing are: the reusable functionality via well-defined interfaces, the service infrastructure that enables services to be published, discovered, invoked, and, if needed destroyed on demand, as well as the fast application development by employing existing services.

In systems built up in such a setup, it becomes essential to ensure a satisfying level of the system's Quality-of-Service (QoS). Sometimes, based on their QoS,

one simply needs to decide which service to select out of a number of available services that offer similar functionality. To deliver guarantees on provided QoS, some SOS approaches [3, 14, 18, 20] support formal analysis; however, in most cases building the formal system model, out of formalized services, is far from straightforward.

Once a model is created, it becomes crucial to be able to check the fulfilment of requirements of the employed services, both in isolation, as well as in the context of the newly created system that involves service compositions. An important aspect, many times ignored, is the service's resource usage. Any analysis approach that would abstract from service resource constraints might produce analysis results that are insufficiently correct, or reliable.

For instance, let us consider a three shuttle system, previously modeled and analyzed in the Priced Timed Automata (PTA) formal framework [5]. In brief, the system provides transportation services to three different locations. Let us assume a scenario in which two out of three shuttles are supposed to stay in a convoy and reach the common final location. Assuming energy to be the most critical resource in the system (i.e., each shuttle operates on batteries with a limited capacity), it would be beneficial to be able to check if the current energy level in each shuttle is sufficient to reach the final destination, before the actual convoy is created. In addition, each shuttle has timing constraints, which should be in accordance with the deadline of the convoy.

In this paper, we focus on computing functional and extra-functional service guarantees, automatically. The service model is time- and resource-aware, being described in REMES [22], a behavioral language intended for modeling and analysis of interacting embedded components and services. The system is obtained by composing REMES models, via operators that we have defined previously [6].

We have shown how service correctness can be checked using Hoare triples, and strongest postcondition semantics, described in Section 3.1. However, the postcondition calculation, on which the verification relies, is not currently automated, hindering the applicability of the method. Here, we address this deficit by presenting algorithms for computing strongest postconditions (service guarantees) automatically, by applying minimum/maximum reachability analysis on PTA [16] translations of the REMES service models (Section 3).

We consider the service resource usage in REMES as a cost variable in PTA, and we include the computation of the minimum and maximum reachability costs of a final PTA location in our algorithms, alongside with calculating the strongest postcondition of reaching such location, over symbolic states. The approach, described in Section 3, is accompanied by an illustrative example of a simple model of a PTA service. Last but not least, we compare our approach with some relevant work in Section 4 before concluding the paper in Section 5.

2 Preliminaries and A Simple Example

2.1 REMES modeling language

To model functional and extra-functional behavior such as timing and resource usage of SOS, in this paper, we use the dense-time state-based hierarchical modeling language called REMES [22]. The language has been initially intended as a meaningful basis for modeling and analysis of embedded systems in a component-based fashion. To make it suitable for modeling SOS too, we have recently extended REMES with constructs fit for an SOS description [6]. To enable formal analysis, REMES models can be transformed into Timed Automata (TA) [1], or PTA [2], depending on the analysis type [12].

We find REMES language appropriate also for SOS, since it is a language well-suited for abstract modeling, supports hierarchical modeling, has an input/ouput distinction, a well-defined formal semantics, and tool support [13] [1].

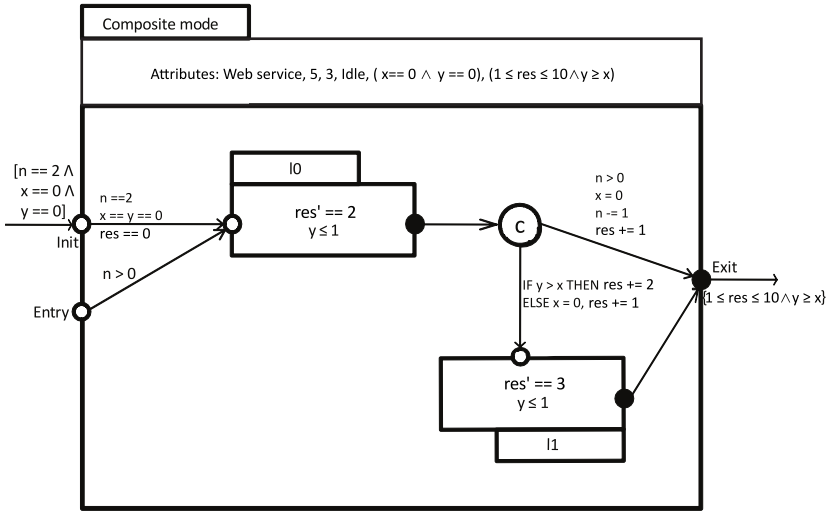

Fig. 1. An example of a REMES service

Let us assume a simple example of a composite mode that models a web service depicted in Fig. 1. The composite mode contains two submodes, i.e., *atomic* modes l0 and l1. The mode has a special Init entry point, visited when the service first executes, and where all variables are initialized.

In REMES one may model timed behavior and resource consumption. Timed behavior is modeled by global continuous variables of specialized type *clock* evolving at rate 1 (x, y in Fig. 1). Each (sub)mode can be annotated with the corresponding continuous resource usage, if any, modeled by the first derivative of the

[1] The REMES tool-chain is available at http://www.fer.hr/dices/remes-id.

real-valued variables that denote resources that evolve at positive integer rates (r1 in Fig. 1). Discrete resources are allocated through updates, e.g., r2 += 1 in Fig. 1.

The REMES service shown in Fig. 1 contains a list of attributes (i.e., service type is Web service, capacity is 5, time-to-serve is 3, status is Idle, service precondition is (x == 0 ∧ y == 0), and postcondition ($1 \leq res \leq 10 \wedge y \geq x$)) exposed at the interface of the REMES service. A service precondition is a predicate that constrains the start of service execution, and must be true at the time a REMES service is invoked. A postcondition must hold at the end of a REMES service execution and it can be the same or included into the user defined requirement, also modeled as a predicate.

To verify the service correctness, we use the forward analysis technique based on the computation of the strongest postcondition of a REMES service w.r.t. a given precondition. To prove the correctness of a REMES service in isolation, we check that the calculated strongest postcondition is no more than the given requirement. Since REMES models can be automatically transformed to PTA via a well-defined set of rules [12, 13, 19], in this paper, we propose an algorithmic technique of strongest postcondition calculation on the PTAn description of a service, in order to provide automation to our REMES verification procedure.

The service composition correctness check reduces to discharging similar boolean implications, as we have shown in our recent work [6]. Therefore, automating the strongest postcondition calculation of services is central to the applicability of our analysis method.

For a more thorough description of the REMES language, we refer the reader to [6, 22].

2.2 Priced Timed Automata

In the following, we recall the model of PTAn [2, 4], an extension of TA [1] with prices on both locations and edges.

Let us assume a finite alphabet *Act* ranging over a, b etc., a finite set of real-valued clocks χ and $\mathcal{B}(\chi)$ the set of formulas obtained as conjunctions of atomic constraints of the form $x \bowtie n$, where $x \in \chi$, $n \in \mathbb{N}$, and $\bowtie \in \{<, \leq, =, \geq, >\}$. The elements of $\mathcal{B}(\chi)$ are called *clock constraints* over χ. Additionally, $\mathcal{P}(\chi)$ represents the powerset of χ.

Definition 1 *A linearly Priced Timed Automaton (PTAn) over clocks χ and actions Act is a tuple (L, l_0, E, I, P), where L is a finite set of locations, l_0 is the initial location, $E \subseteq L \times \mathcal{B}(\chi) \times Act \times \mathcal{P}(\chi) \times L$ is the set of edges, $I : L \to \mathcal{B}(\chi)$ assigns invariants to locations, and $P : (L \cup E) \to \mathbb{N}$ assigns prices (or costs) to both locations and edges. In the case of $(l, g, a, r, l') \in E$, we write $l \overset{g,a,r}{\to} l'$.* ■

The explanation of (l, g, a, r, l') follows in text below. Fig. 2 depicts the PTA description of the REMES service introduced in Fig. 1. We omit the REMES interface from the transformation, only the internal behavior is transformed. The PTA description consists of three locations: l_0, l_1, and l_2 (with l_0 as the

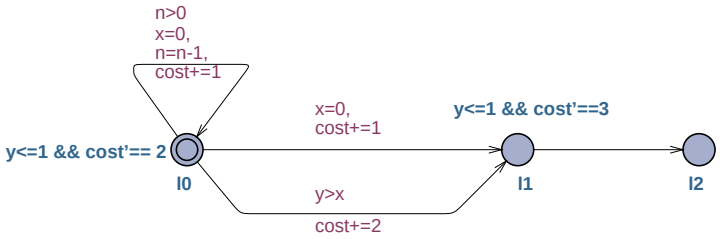

Fig. 2. The PTAn model of the REMES service of Fig. 1

initial location), and edges, which are directed lines connecting locations. The timing behavior is controlled by two clock variables, x and y. For each location, it is possible to assign an invariant that must hold in order to stay in that location (e.g., invariant $y \leq 1$), and that enforces a location change in case it ceases to hold. Further, each edge, may be decorated with guards, that is, guards, i.e., boolean expressions that must hold in order for an edge to be taken (e.g., $y > x$). For simplicity, to prevent trace explosion and infinite loops, in this example we use integer variable n. To model, simulate and verify our example we use the UPPAAL CORA tool. The tool extends definition 1 with data variables of different types, arrays of data variables, constants, and records.

The semantics of a PTA is defined in terms of a priced transition system over states of the form (l, u), where l is a location, $u \in \mathbf{R}^X$ are clock valuations, and the initial state is (l_0, u_0), where u_0 assigns all clocks in χ to 0. In this model, there are two kinds of transitions: delay transitions and discrete transitions. In delay transitions,

$$(l, u) \overset{d,p}{\rightarrow} (l, u \oplus d)$$

$u \oplus d$ is the result obtained by incrementing all clocks of the automata with delay d, and $p = P(l) * d$ is the cost of performing the delay (the cost of staying in location l_0 is described by $cost' == 2$). Discrete transitions

$$(l, u) \overset{a,p}{\rightarrow} (l', u')$$

correspond to taking an edge $l \overset{g,a,r}{\rightarrow} l'$ for which the guard g is satisfied by u. The clock valuation u' of the target state is obtained by modifying u according to updates r. The cost $p = P((l, g, a, r, l'))$ is the price associated with the edge (the cost of taking a self loop in location l_0 is annotated as $cost+ = 1$).

A timed trace σ of a PTA is a sequence of alternating delays and action transitions

$$\sigma = (l_0, u_0) \overset{a_1,p_1}{\rightarrow} (l_1, u_1) \overset{a_2,p_2}{\rightarrow} \ldots \overset{a_n,p_n}{\rightarrow} (l_n, u_n)$$

A network of PTA $A_1 \ldots A_n$ over χ and Act is defined as the parallel composition of n PTA $A_1 \parallel \ldots \parallel A_n$ over χ and Act. Semantically, a network again describes a timed transition system obtained from those components, by

requiring synchrony on delay transitions and requiring discrete transitions to synchronize on complementary actions (i.e., $a?$ is complementary to $a!$) [4].

2.3 Symbolic Optimal Reachability

The text in this subsection is an adaptation for single-cost PTA, from the one presented by Larsen and Rasmussen, for dual-priced PTA [16]. Symbolic techniques are required in the analysis of infinite state systems. They provide effective ways to describe and manipulate sets of states simultaneously. To enable cost-optimal analysis, such techniques are enriched with cost information annotated to each individual symbolic state [15].

A priced transition systems with a structure $\tau = \langle S, s_0, \Sigma, \rightarrow \rangle$, where S is a set of states, $s_0 \in S$ is the initial state, Σ is a finite set of labels, and $\rightarrow$ is a partial function from $S \times \Sigma \times S$ into the non-negative reals, $\mathbb{R}_{\geq 0}$, defines all possible systems transitions with their respective costs. An execution of τ is a sequence $\gamma = s_0 \overset{a1,p1}{\rightarrow} s_1 \overset{a2,p2}{\rightarrow} \dots \overset{an,pn}{\rightarrow} s_n$. The cost of γ with respect to some goal state $G \subseteq S$, is defined as:

$$\mathsf{COST}_G(\gamma) = \begin{cases} \infty, & \text{if } \forall \;\; i \geq 0 \;\; : \;\; s_i \notin G \\ \sum_{i=1}^{n} p_i, & \text{if } \exists \; n \geq 0 \;\; : \;\; s_n \in G \;\; \wedge \;\; \forall \; 0 \leq i < n \;\; : \;\; s_i \notin G. \end{cases}$$

For a given goal state s, the minimum cost of reaching s is the infimum of the costs of the finite traces ending in the s. Dually, the maximum cost of reaching the goal state s is the supremum of the costs of the finite traces ending in s. Similarly, the minimum/maximum cost of reaching a set of states $G \subseteq S$ is:

$$\inf \{\mathsf{COST}_G(\gamma) : \;\; \gamma \in \mathit{\Gamma}\}, \text{ and}$$
$$\sup\{\mathsf{COST}_G(\gamma) : \;\; \gamma \in \mathit{\Gamma}\}$$

where $\mathit{\Gamma}$ is the set of all executions in the priced transition system τ.

To effectively analyze priced transition systems, priced symbolic states of the form (A, π) are used, where $A \subseteq S$ is a set of states, and $\pi : A \rightarrow 2^{\mathbb{R}_{\geq 0}}$ assigns non-negative costs to all states of A. The reachability of the priced symbolic state (A, π) assumes that all $s \in A$ are reachable with all costs in $\pi(s)$. To express successors of priced symbolic states, e.g., all states that can be reached from the current state $s \in A$, we use a Post-operator $\mathit{Post}_a(A, \pi) = (\mathit{post}_a(A), \eta) = (B, \eta)$ expressed as follows:

$$B = \{s^{'} \;\; | \;\; \exists \; s \in A : s \overset{a}{\rightarrow} s^{'}\}$$
$$\eta(s) = \inf\{\pi(s^{'}) + p \mid \;\; s^{'} \in A \wedge s^{'} \overset{a,p}{\rightarrow} s\}$$

Here η provides the cheapest cost for reaching states of B via states in A, assuming that these may be reached with costs according to π.

A symbolic execution of a priced transition system τ is a sequence $\beta = (A_0, \pi_0), \dots, (A_n, \pi_n)$, where for $i < n$, $(A_{i+1}, \pi_{i+1}) = \mathit{Post}_{ai}(A_i, \pi_i)$ for some $a_i \in \Sigma$ and $A_0 = \{s_0\}$ and $\pi_0(s_0) = 0$. The relation between executions and symbolic executions is expressed as follows:

– For each execution γ of τ ending in s, there is a symbolic execution β ending in (A, π) such that $s \in A$ and $\mathsf{COST}(\gamma) \in \pi(s)$.
– Let β be a symbolic execution of τ ending in (A, π); then for each $s \in A$ and $p \in \pi(s)$, there is an execution $\gamma \in s$ such that $\mathsf{COST}(\gamma) = p$.

From the statements above, one can notice that symbolic states accurately capture the cost of reaching all states in the state space.

3 Algorithms for Calculating Strongest Postconditions of Services

To provide constructs for the correctness check of a REMES service, as described in Section 2 and introduced in [6], we assume the service described as a Hoare triple, and the forward analysis technique that relies on computations of the strongest postcondition of the REMES service w.r.t. a given precondition. Proving the correctness of a REMES service in isolation reduces to simply checking the Boolean implication between the calculated strongest postcondition and the given user requirement.

Previously [6], we have focused on less complex systems and employed the Guarded Command Language (GCL) [9] to prove service correctness by manual computation of the strongest postconditions needed in the process. In this paper, we aim for a more automated mechanism to check service correctness, focusing on developing algorithms that facilitate such computation for REMES services formally described as PTA. We can perform maximum/minimum resource-usage trace computation on the corresponding PTA, while accumulating the strongest postcondition during the analysis. The algorithms for strongest postcondition calculation, presented in this paper are inspired by the symbolic reachability algorithms for computing the minimum and the maximum reachability cost, respectively, proposed by Larsen and Rasmussen [16].

In the following, we recall the notion of strongest postcondition, as introduced by Dijkstra and Sholten [10], and the program correctness check based on it. Next, we introduce two algorithms that compute the strongest postcondition of a REMES service formally described as PTA, together with the maximum/minimum cost reachability analysis, respectively.

3.1 Strongest Postcondition

Assume $\{p\}\mathcal{S}\{q\}$ is a Hoare triple denoting the partial correctness of service $\mathcal{S}$ with respect to precondition p and postcondition q. According to Dijkstra and Sholten [10], the strongest postcondition transformer, denoted by $(sp.\mathcal{S}.p)$, is the set of final states for which there exists a computation controlled by $\mathcal{S}$, which belongs to class "initially p". Assuming that p holds, the execution of a service $\mathcal{S}$ results in $sp.\mathcal{S}.p$ true, if $\mathcal{S}$ terminates. Proving the Hoare triple, that is, proving the correctness of service S, reduces to showing that $(sp.\mathcal{S}.p \Rightarrow q)$ holds.

To illustrate the strongest postcondition calculation on a simple statement, let us assume that a service performs a simple subtraction operation $(x := x - 5)$

and that the provided precondition is $p = (x > 15)$, while the requirement is $q = (x > 10)$. Then, calculating the strongest postcondition reduces to the following:

$$sp.(x := x - 5).(x > 15) = (\exists x_0 \cdot x = x_0 - 5 \wedge (x_0 > 15))$$

where x_0 is the inital value of x. Verifying the correctness of S, with respect to p, and q above, reduces to showing that:

$$\exists x_0 \cdot x = x_0 - 5 \wedge (x_0 > 15) \Rightarrow (x > 10)$$

In the following, we show how to compute $sp.\mathcal{S}.p$ automatically, assuming $\mathcal{S}$ is the PTA semantic translation of a REMES service.

3.2 Strongest postcondition calculation and minimum cost reachability

In this section, we show the algorithm that computes the strongest postcondition, and the minimum cost of resource consumption for a given REMES service, formally described as a PTAn.

Let us assume (A, π) and (B, η) as our priced symbolic states. If $A \subseteq B$ and $\eta(s) \leq \pi(s)$, for all $s \in A$, we denote by $(B, \eta) \sqsubseteq_{inf} (A, \pi)$ the preorder expressing that (B, η) is "at most as big and cheap" as (A, π) [15].

Algorithm 1 employs two data-structures, WAITING (initially containing the initial priced symbolic state (A, π_0)) and PASSED (initially empty) to hold the priced symbolic states waiting to be examined, and those that are already explored, respectively. At each iteration, the algorithm selects a priced symbolic state (A, π) from WAITING. If (A, π) is a goal state [2] not contained in a goal state previously stored in SP (strongest postcondition), it is added to the calculated postcondition SP. Otherwise, if it is not a goal state and not contained in a symbolic state previously stored in PASSED, it is added to PASSED, and all its successor states are added to WAITING. When WAITING is empty, the strongest postconditions calculated for each path reaching the goal state are returned.

We define Final (A, π) as follows:

$$\text{Final}\,(A, \pi) = \begin{cases} true, & \text{if } (A, \pi) \in \ F \\ false, & \text{otherwise.} \end{cases}$$

where F denotes the final priced symbolic state.

The algorithm terminates when WAITING is empty, that is, when no further priced symbolic state is left to be examined. The algorithm results in a set of strongest postconditions SP. Termination of the algorithm is guaranteed, provided that $\not\sqsubseteq_{inf}$ is a well quasi-ordering on symbolic states [15].

[2] Note that, in a PTAn describing a REMES service, the goal state is determined by a unique location and hence, if Final (A, π) holds, then the whole of (A, π) is a goal state, assuming that every symbolic state (A, π) satisfies the property that all states in A are in the same location.

In addition, information about the cost of service execution is carried within the calculated strongest postcondition. The cost is assumed to be initially set to ∞ and updated whenever a goal state is found, which can be reached with the lower cost than the current one.

00 SP := {}
01 PASSED := {}
02 WAITING := $\{(\{A_0\}, \pi_0)\}$
03 **while** WAITING $\neq \{\}$ **do**
04 select (A, π) from WAITING
05 **if** (Final (A, π) $\wedge$ $\forall$ $(B, \eta) \in$ SP : $(B, \eta) \not\sqsubseteq_{inf} (A, \pi)$)
06 **then** SP := SP $\cup$ (A, π) **else**
07 **if** $\forall$ $(B, \eta) \in$ PASSED : $(B, \eta) \not\sqsubseteq_{inf} (A, \pi)$ **then**
08 PASSED := PASSED $\cup$ $\{(A, \pi)\}$
09 WAITING := WAITING $\cup$ $\bigcup_{a \in \Sigma} Post_a(A, \pi)$
10 **end if**
11 **end if**
12 **end while**
13 **return** SP

Algorithm 1. Abstract algorithm for computing the service strongest postcondition and the minimum cost of reaching the goal state.

As stated, the algorithm provides a set of strongest postconditions calculated for distinctive paths that reach the goal state (location) in PTAn. Finally, to get the actual strongest postcondition, we simplify the set SP. The strongest postcondition can be simplified as follows:

$$\forall\ (A, \pi)_i \in \text{SP}\ :\ \bigcup_{j \neq\ i} (A, \pi)_j \not\sqsubseteq_{inf} (A, \pi)_i$$

The simplification assumes that each symbolic priced state that is not included into the reunion of all other symbolic priced states is subtracted. For more details regarding simplification, we refer the reader to [7, 11].

Example revisited. To illustrate our approach, we recall the simple service shown in Fig. 2. In the automaton, it is possible to delay either in location l_0 or l_1. Location l_2 is assumed to be the final location. From l_0, it is possible to take a self-loop, for maximum two times (integer n is initially set to two) and then take one of the available edges, or directly take one of the edges that lead to location l_1, and finally end up in location l_2. Staying in locations l_0 or l_1 or taking any of the available edges increases the accumulated cost, annotated via the cost variable. We are interested in calculating the minimum cost for reaching the final location (l_2) and the respective strongest postcondition.

Let us now assume that our service is annotated with precondition p, which we assume satisfied, and postcondition q, which represents the service requirement, as follows:

$$p = (x = 0 \wedge y = 0)$$
$$q = (1 \leq res \leq 10 \wedge x \leq y)$$

In the above, x and y are clock variables, n is an integer variable that bounds the number of loop iterations in location l_0, and res is the variable modeling the resource usage of the original service. In the corresponding PTAn representation, res translates into the automaton's cost variable. By verifying q, within the minimum cost reachability context, we want to check whether our service consumes at least 1 unit of resource, for the system to be considered correct.

Proving correctness of the PTAn w.r.t. this requirement relies on the strongest postcondition computation, for minimum cost, according to Algorithm 1.

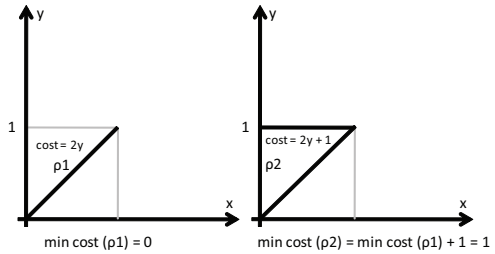

Fig. 3. Symbolic states for minimum reachability cost

In Fig. 3, we illustrate one trace of the minimum cost reachability analysis that reaches the goal location l_2. Note that in the minimum cost case, it is optimal to reach l_2 in zero time units, via location l_1. The accumulated cost is then 1. In case of the total accumulated delay 1, it is optimal to delay in l_0 with cost 2, hence the cost of reaching l_2 is $2y + 1$ and the strongest postcondition *of this trace* is $\mathsf{cost} = 2y + 1 \wedge y \leq 1 \wedge x \leq y$.

There are four more traces reaching l_2. The total SP becomes

$$\begin{array}{l}(\mathsf{cost} = 2y + 1 \wedge y \leq 1 \wedge x \leq y) \vee \\ (\mathsf{cost} = 2y + 2 \wedge y \leq 1 \wedge x \leq y) \vee \\ (\mathsf{cost} = 2y + 3 \wedge y \leq 1 \wedge x \leq y) \vee \\ (\mathsf{cost} = 2y + 3 \wedge y \leq 1 \wedge x < y) \vee \\ (\mathsf{cost} = 2y + 4 \wedge y \leq 1 \wedge x < y)\end{array}$$

After simplifying according to our definition, the total SP can be reduced to the following:

$$(\mathsf{cost} = 2y + 4 \wedge y \leq 1 \wedge x < y)$$

It is easy to prove that the above strongest postcondition implies the following predicate:

$$v = (1 \leq \mathsf{cost} \leq 6 \wedge x \leq y),$$

which in turn implies q, if cost is replaced by res. It then follows that the minimum-cost strongest postcondition implies q, which completes our correctness proof in this case.

3.3 Strongest postcondition calculation and maximum cost reachability

Algorithm 1 can be modified to provide the strongest postcondition calculation together with the maximum reachability cost. At the service level, this would translate into checking whether, in the worst-case of service resource-usage, the latter does not exceed a prescribed upper bound. We here briefly sketch the required modifications of Algorithm 1. As previously, we assume that all paths eventually reach the goal state. The modification concerns the lines 05 to 07 of Algorithm 1, which become as follows:

05 **if** (Final (A, π) $\wedge$ $\forall$ $(B, \eta) \in$ SP : $(B, \eta) \not\sqsupseteq_{sup} (A, \pi)$)
06 **then** SP := SP $\cup$ (A, π) **else**
07 **if** $\forall$ $(B, \eta) \in$ PASSED : $(B, \eta) \not\sqsupseteq_{sup} (A, \pi)$) **then**

Algorithm 2. Extract of abstract algorithm for computing the service strongest postcondition and the maximum cost of reaching the goal state.

The only difference from the previous algorithm is in the pruning of symbolic priced states before adding them to PASSED or SP. Any symbolic state (A, π) can be pruned if there exists already a symbolic state (B, η), such that $A \subseteq B$ and $\pi(s) \leq \eta(s)$ for all states $s \in A$. Similarly, the strongest postcondition can be simplified as follows:

$$\forall\ (A, \pi)_i \in \text{SP}\ :\ \bigcup_{j \neq\ i} (A, \pi)_j \not\sqsupseteq_{sup} (A, \pi)_i$$

Example revisited. The PTAn depicted in Fig. 2 is again used to illustrate the approach described above. According to our methodology, to verify the correctness of the service w.r.t. p and q, we need to first compute the strongest postcondition of the corresponding PTAn, under the assumption of worst-case resource usage, that is, maximum cost in PTA terms.

Fig. 4 depicts a trace of the reachability analysis, assuming the maximum cost of reaching the goal location. In this case, the trace includes two self-loops in l_0, and then a jump to l_1 via the lower of the two possible edges. The costs are $2y$, $2y + 1$, and $2y + 2$ in l_0, and then $3y + 4$ in l_1 (and in l_2). The strongest postcondition w.r.t. the maximum cost of the trace becomes $\mathsf{cost} = 3y + 4 \wedge y \leq$

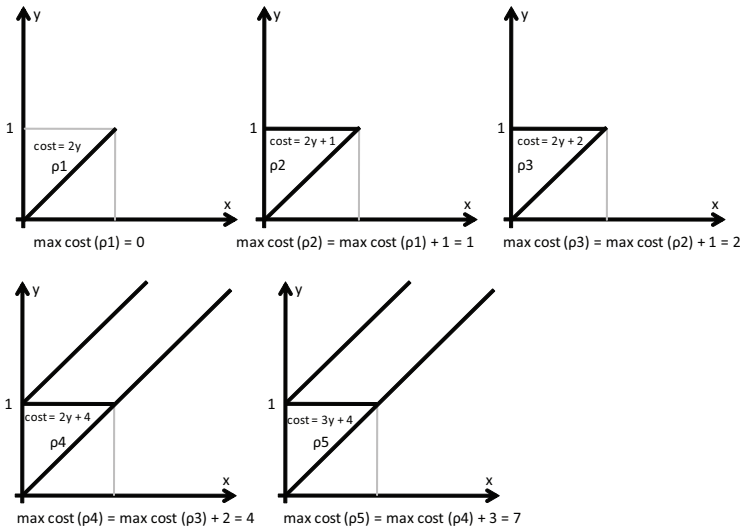

Fig. 4. Symbolic states for maximum reachability cost

$1 \wedge x < y$. The total SP of the whole PTAn w.r.t. maximum cost becomes

$$\begin{array}{l}(\mathsf{cost} = 3y + 1 \wedge y \leq 1 \wedge x \leq y) \vee \\ (\mathsf{cost} = 3y + 2 \wedge y \leq 1 \wedge x \leq y) \vee \\ (\mathsf{cost} = 3y + 3 \wedge y \leq 1 \wedge x \leq y) \vee \\ (\mathsf{cost} = 3y + 3 \wedge y \leq 1 \wedge x < y) \vee \\ (\mathsf{cost} = 3y + 4 \wedge y \leq 1 \wedge x < y)\end{array}$$

The calculated SP can now be simplified according to our definition and reduced to the following:

$$(\mathsf{cost} = 3y + 4 \wedge y \leq 1 \wedge x < y)$$

By applying simple rules of logic, we can verify that the above SP with maximum cost implies the following predicate:

$$w = (1 \leq \mathsf{cost} \leq 7 \wedge x \leq y)$$

Next, after replacing cost by res in w, it follows straightforwardly that $w[cost \leftarrow res] \Rightarrow q$, which entails that the strongest postcondition for maximum cost implies the requirement q. This actually proves the correctness of our original service, including its feasibility w.r.t. worst-case resource usage.

4 Discussion and Related Work

Beek et al. [23] give an exhaustive survey of several popular approaches [3, 14, 18, 20] that provide means for service modeling, service composition, and service

correctness check. While all the described approaches offer a rich environment for service modeling and composition, neither of them has included direct support for service correctness check. To overcome this limitation, recently, in some of these approaches [8, 17, 21] formal methods have been employed with the intention to provide guarantees for web-service compositions.

Diaz et al. describe how BPEL and WS-CDL services can be automatically translated to timed automata and verified by UPPAAL model checker [8]. However, the described approach is limited to checking service timing properties. Narayanan et al. show how semantics of OWL-S, described using first-order logic, can be translated to Petri-nets and then analyzed as such [17]. The analysis includes reachability and liveness properties, and checking if the given service or service compositions are deadlock free. Weber et al. introduce a formalism to check control-flow correctness [24]. They first verify whether the given process is sound, meaning that the control-flow of interest guarantees proper completion and that there are no deadlocks. Further, they consider process models in which the individual activities are annotated with logical preconditions and postconditions. In the last step, the authors aim to determine whether the interaction of control flow and logical states of the process is correct.

Compared to these approaches, REMES services can be both mechanically reasoned about [6], and also, automatically translated to PTA [2] where one can apply algorithmic computation of the strongest postcondition of PTAn, as presented in this paper. Moreover, REMES services formally described as PTA can be analyzed with UPPAAL , or UPPAAL CORA tools[3], for functional but also extra-functional behaviors, by which we mean resource-wise behaviors.

5 Conclusions

In this paper, we have presented an approach that facilitates the automated correctness check for services, formally described as PTA, by providing forward analysis algorithms that compute the most precise postcondition (strongest postcondition) that is guaranteed to hold upon termination of the service execution, which corresponds to reaching a final PTAn location. The approach serves as the alternative algorithmic verification method for services modeled as REMES modes, to the deductive method that uses Hoare triples and the strongest postcondition semantics to prove service correctness [6].

In our previous work, we show that proving the correctness of a REMES service reduces to showing that the calculated strongest postcondition of that particular service is at least as strong as the user-defined requirement. The algorithms that we propose here extend the existing maximum, minimum cost reachability algorithms [16], with strongest postcondition calculation. In our case, the cost variable models the service's accumulated resource-usage. Consequently, the computed strongest postcondition of a service modeled as a PTAn

[3] For more information about the UPPAAL and UPPAAL CORA tool, visit the web page www.uppaal.org.

could contain both functional, but also timing and resource-usage information, observable at the end of the service execution.

The approach is illustrated on a small example, on which we also show resource usage/cost calculation using symbolic states. However, the complexity of our algorithms, and their applicability on larger examples have not been investigated yet.

As future work, we plan to address the above issues, by first implementing the strongest postcondition algorithms in the Uppaal Cora tool. We also intend to extend the Remes tool-chain with a postcondition calculator that would run Uppaal Cora as a back-end.

References

[1] Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science 126(2), 183–235 (1994), citeseer.nj.nec.com/alur94theory.html

[2] Alur, R.: Optimal paths in weighted timed automata. In: In HSCC01: Hybrid Systems: Computation and Control. pp. 49–62. Springer (2001)

[3] Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S.: BPEL4WS, Business Process Execution Language for Web Services Version 1.1. IBM (2003)

[4] Behrmann, G., Fehnker, A., Hune, T., Larsen, K.G., Pettersson, P., Romijn, J., Vaandrager, F.: Minimum-Cost Reachability for Priced Timed Automata. In: Benedetto, M.D.D., Sangiovanni-Vincentelli, A. (eds.) Proceedings of the 4th International Workshop on Hybris Systems: Computation and Control. pp. 147–161. No. 2034 in Lecture Notes in Computer Sciences, Springer–Verlag (2001)

[5] Causevic, A., Seceleanu, C., Pettersson, P.: Formal reasoning of resource-aware services. Technical Report ISSN 1404-3041 ISRN MDH-MRTC-245/2010-1-SE, Mälardalen University (June 2010)

[6] Causevic, A., Seceleanu, C., Pettersson, P.: Modeling and reasoning about service behaviors and their compositions. In: Proceedings of 4th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation (ISOLA 2010), Formal Methods in Model-Driven Development for Service-Oriented and Cloud Computing track. Springer LNCS (October 2010)

[7] David, A., Hkansson, J., Larsen, K., Pettersson, P.: Model checking timed automata with priorities using dbm subtraction. In: Asarin, E., Bouyer, P. (eds.) Formal Modeling and Analysis of Timed Systems, Lecture Notes in Computer Science, vol. 4202, pp. 128–142. Springer Berlin / Heidelberg (2006)

[8] Daz, G., Pardo, J.J., Cambronero, M.E., Valero, V., Cuartero, F.: Automatic translation of ws-cdl choreographies to timed automata. In: Bravetti, M., Kloul, L., Zavattaro, G. (eds.) EPEW/WS-FM. Lecture Notes in Computer Science, vol. 3670, pp. 230–242. Springer (2005)

[9] Dijkstra, E.W.: Guarded commands, nondeterminacy and formal derivation of programs. Commun. ACM 18(8), 453–457 (1975)

[10] Dijkstra, E.W., Scholten, C.S.: Predicate calculus and program semantics. Springer-Verlag New York, Inc., New York, NY, USA (1990)

[11] Hsiung, P.A., Lin, S.W.: Model checking timed systems with priorities. In: Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications. pp. 539–544. RTCSA '05, IEEE Computer Society, Washington, DC, USA (2005)

[12] Ivanov, D.: Integrating formal analysis methods in PROGRESS IDE. Master of science thesis, Malardalen Research and Technology Centre, Vasteras, Sweden (June 2011)
[13] Ivanov, D., Orlic, M., Seceleanu, C., Vulgarakis, A.: Remes tool-chain - a set of integrated tools for behavioral modeling and analysis of embedded systems. In: Proceedings of the 25th IEEE/ACM International Conference on Automated Software Engineering (ASE 2010) (September 2010)
[14] Kavantzas, N., Burdett, D., Ritzinger, G., Fletcher, T., Lafon, Y., Barreto, C.: Web services choreography description language version 1.0. World Wide Web Consortium, Candidate Recommendation CR-ws-cdl-10-20051109 (November 2005)
[15] Larsen, K.G., Behrmann, G., Brinksma, E., Fehnker, A., Hune, T., Pettersson, P., Romijn, J.: As cheap as possible: Efficient cost-optimal reachability for priced timed automata. In: Proceedings of the 13th International Conference on Computer Aided Verification. pp. 493–505. CAV '01, Springer-Verlag, London, UK (2001), http://portal.acm.org/citation.cfm?id=647770.734117
[16] Larsen, K.G., Rasmussen, J.I.: Optimal reachability for multi-priced timed automata. Theor. Comput. Sci. 390, 197–213 (January 2008), http://portal.acm.org/citation.cfm?id=1330765.1330861
[17] Narayanan, S., McIlraith, S.A.: Simulation, verification and automated composition of web services. In: WWW '02: Proceedings of the 11th international conference on World Wide Web. pp. 77–88. ACM, New York, NY, USA (2002)
[18] Object Management Group (OMG): Business Process Modeling Notation (BPMN) version 1.1. (January 2008), http://www.omg.org/spec/BPMN/1.1/
[19] Orlić, M.: Resource usage prediction in component-based software systems. Phd thesis, Faculty of electrical engineering and computing, University of Zagreb (November 2010)
[20] Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., Fensel, D.: Web service modeling ontology. Applied Ontology 1(1), 77–106 (2005)
[21] Salaün, G., Bordeaux, L., Schaerf, M.: Describing and reasoning on web services using process algebra. In: ICWS '04: Proceedings of the IEEE International Conference on Web Services. p. 43. IEEE Computer Society, Washington, DC, USA (2004)
[22] Seceleanu, C., Vulgarakis, A., Pettersson, P.: Remes: A resource model for embedded systems. In: In Proc. of the 14th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2009). IEEE Computer Society (June 2009)
[23] Ter Beek, M.H., Bucchiarone, A., Gnesi, S.: Formal methods for service composition. Annals of Mathematics, Computing & Teleinformatics 1(5), 1 – 10 (2007), http://journals.teilar.gr/amct/, in: Annals of Mathematics, Computing & Teleinformatics, vol. 1 (5) pp. 1 - 10. Technological Education Institute of Larissa (TEIL), Greece, 2007.
[24] Weber, I., Hoffmann, J., Mendling, J.: Beyond soundness: on the verification of semantic business process models. Distrib. Parallel Databases 27, 271–343 (June 2010), http://dx.doi.org/10.1007/s10619-010-7060-9