Deadline Verification for Web Services Using Timed Automata

Yamen El Touati
Faculty of Computing and IT
Northern Border University
Kingdom of Saudi Arabia
yamen.touati@nbu.edu.sa

Abstract-Many computation tasks are made today on remote cloud platforms using web services. Beyond the advantages provided by such services, many new challenges arise. One of the challenging problems is ensuring that web services respect critical deadlines. This is a critical issue, especially for real-time systems that use remote web services. This paper aims to propose a framework for deadline verification using Timed Automata (TA).

Keywords-web services; deadlines; timed automata; verification

I. INTRODUCTION

In the last few years, most companies migrated to remote cloud-based services. In these systems, most of the communication is ensured via web services that offer quasi-unlimited resources and computational power. However, companies face new challenges concerning security [1-4] and timing constraints [9-11, 19]. A web service is a software system designed to support interoperable machine-to-machine interaction over a network [6]. They provide flexible and efficient solutions to the sharing of information between people and businesses [6, 20]. By nature, a web service can face many random interferences that can cause delays in the required tasks [12-15, 18]. In many cases, it is not easy to check whether the specification deadlines are respected. In general, this is because some web services are composed of many correlated tasks, where delays and timing are propagated according to a dynamic schedule. In this particular context, timed automata [6-10] are suitable for modeling the evolution of a system focusing on time constraints. These are presented as an extension of finite-state automata with clocks into location. Transitions can be executed when a linear constraint on clocks, called guard, is verified. This framework presents an interesting graphical interface combined with powerful computational power. Moreover, it is more suitable to solve verification problems using a timed automata framework than performing algebraic calculus, which may be difficult to resolve if the size of the system increases. This paper proposes a new method based on timed automata reachability checking in order to resolve the problem of deadline verification.

II. BACKGROUND OF TIMED AUTOMATA AND TIME TRANSITION SYSTEMS

$\mathbb{R}_+$ denotes nonnegative reals, $X=\{x_1,x_2,\ldots,x_n\}$ denotes the finite set of real-valued variables, and the circumflex (~) is used to denote an element of the set of operators $\{<,\leq,=,\geq,>,\neq\}$. A simple inequality over X is an inequality of the form $x\sim c$, where $x\in X$ and $c\in\mathbb{R}_+$. A rectangular predicate over X is a conjunction of linear inequalities over X. $C(X)$ denotes the set of linear predicates on X.

A. Timed Automata

A timed automaton [1,3] is a tuple $A = (L, l_0, X, \Sigma, E, inv, F)$ where:

- L is a finite set of locations (nodes)
- l_0 is the initial state
- X is a finite set of non-negative real valued clocks
- Σ is a finite set of actions or events (labels)
- $E \subset L \times C(X) \times \Sigma \times 2^X \times L$ is a set of transitions. $e \in E$, and $e=(l, \delta, \alpha, R, l^i)$ is defined, where l is the start location, δ is the guard, α is the action, R is the set of clocks to reset, and l^i is the end location
- $inv \in C(X)^L$ is a function that associates an invariant to each location
- $F \subset L$ is the set of final states

B. Semantic of Timed Automata

The semantic of timed automata $A=(L, l0, X, \Sigma, E, inv)$ is given by a Time Transition System [2] $SA= (Q, q0, \rightarrow)$, where:

- $Q = L \times \mathbb{R}_+{}^X$
- $q_0 = (l_0, 0)$ is the initial state
- $\rightarrow \subset Q \times (\Sigma \cup \mathbb{R}_+) \times Q$ is defined by:
  - $(l,v) \rightarrow^a (l^i,v^i)$ (discrete transition) if $\exists$ $(l, \delta, a, R, \rho, l^i) \in E$ such that $\delta(v)=true$, and $vi=v[R \leftarrow 0][\rho]$ and $inv(l^i)(v^i)=true$
  - $(l,v)\rightarrow^t(l^i,v^i)$ (continuous transition) if $l=li$ and $vi=v+t$ and $inv(li)(vi)=true$

Corresponding author: Yamen El Touati

It is noted that $(l,v)\rightarrow^{*}(l^{i},v^{j})$ in the case of a transition sequence of leading from (l,v) to (l^{i},v^{j}).

C. Reachability in Timed automata

A *run* of a timed automaton A is a path in S_A starting from l_0. $[[A]]$ denotes the set of all runs in the timed automaton A. It is noted that $(l,v)\rightarrow^{t}(l',v')\rightarrow^{a}(l'',v'')$ is equivalent to $(l,v)\rightarrow_{a}^{t}(l'',v'')$. A state (l_i,v_i) is considered as *reachable*, if:

$$\exists(l_0,v_0)\rightarrow_{a_0}^{t_0}(l_1,v_1)\rightarrow_{a_1}^{t_1}(l_2,v_2)\rightarrow_{a_2}^{t_2}\dots\rightarrow_{a_i}^{t_i}(l_i,v_i)$$

where $(l_0,v_0)=q_0$. A run:

$$(l_0,v_0)\rightarrow_{a_0}^{t_0}(l_1,v_1)\rightarrow_{a_1}^{t_1}(l_2,v_2)\rightarrow_{a_2}^{t_2}\dots\rightarrow_{a_i}^{t_i}(l_i,v_i)\dots$$

starting from $q_0=(l_0,v_0)$ is a *timed trace*, denoted as:

$$w=(a_0=e_0,\delta_0)\rightarrow(a_1,\delta_1)\rightarrow(a_2,\delta_2)\rightarrow\cdots(a_i,\delta_i)\dots$$

where w is a sequence of pairs (a_i,δ_i), $a_i\in E$ is a transition, and $\delta_{i+1}\in\mathbb{R}_+$ is the delay between the two successive events a_i and a_{i+1} where: $\delta_0=0$ and $\forall i\geq 1$:

$$\delta_i=t_i-t_{i-1}.$$

III. Deadline verification in web services

A web service can be modeled by a set of tasks with a timed automaton, where each location corresponds to a specific configuration in the web service's lifecycle. Transitions between locations correspond to a configuration change. Thus, locations can be considered as elementary subtasks to fulfill the entire web service. Some of these tasks may take a longer than expected time to be fulfilled. Since the sequence of tasks and their exact duration is not easy to tackle in these systems, it would be interesting to investigate a way to check whether any critical deadline was broken.

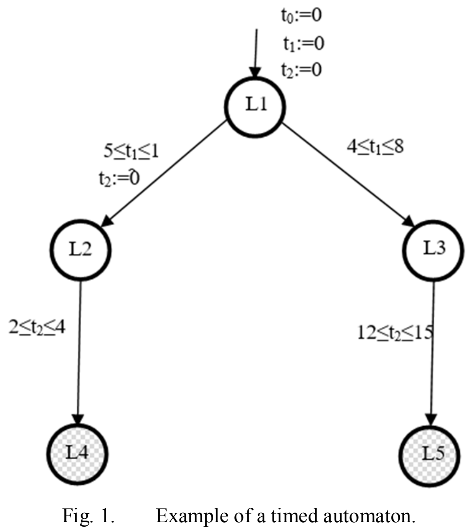


Fig. 1. Example of a timed automaton.

A. Checking Deadline in Timed Automata

Consider the example shown in Figure 1 and suppose that this timed automaton represents the behavior of a web service composed of 5 subtasks (those represented by locations L1 to L5). It can be assumed that the deadline of this web service is 16 time units (t.u). Checking whether this deadline is respected is equivalent to checking if L3 is reached always before 16 t.u. In this case, L3 is always reached before 8 t.u, which means that the previous deadline is respected by the web service. To resolve this problem, a universal clock x_0 (a clock that is never reset) can be added and check if the constraint $x_0\leq 15$ is always verified.

B. Generalization

This problem can be generalized for a timed automaton $A=(L, l_0, X, \Sigma, E, inv)$ as:

Given $A_1=(L, l_0, X\cup\{x_0\}, \Sigma\cup\{d\}, E, inv)$ and $Deadline\in\mathbb{R}_+$, *Deadline* is respected if:

$\forall l\in L$, $\forall v\in\mathbb{R}_+$ such that $(l_0,0)\rightarrow^{*}(l,v)$, $v(x0)\leq Deadline$ (1)

C. Deadline Verification Steps

To ensure that property (1) is respected, this problem is transformed into a reachability verification problem as follows.

- A new derivative automaton is constructed $A_2=(L\cup\{l_d\}, l_0, X\cup\{x_0\}, \Sigma\cup\{d\}, E\cup E_d, inv)$, such that:

  $E_d=\{(l, x_0>Deadline, \{d\}, \{\}, l_d), \quad l\in L-F\}$
- The reachability of l_d should be checked. If l_d is reachable, then the deadline is not guaranteed to be respected by the web service.

The l_d location has the behavior of a typical dead state. Transitions are added from each nonfinal location to l_d, with the guard $x_0>Deadline$, without any reset.

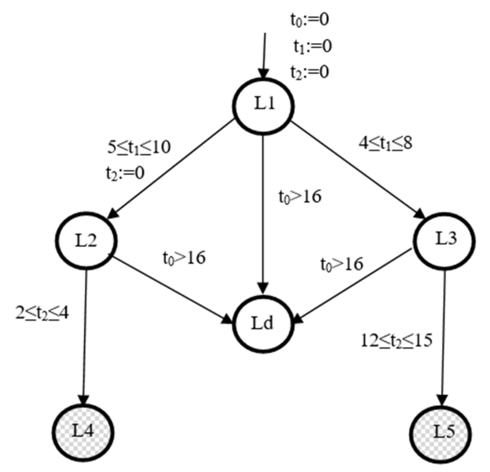


Fig. 2. Updated Timed Automaton with dead state and universal clock – l_d is not reachable

In the example shown in Figure 2, the suggested method is applied to proceed to deadline verification. Location l_d is reachable if the universal clock t_0 reaches the limit defined by the deadline, here 16. In this particular case, the location l_d is not reachable, which means that the web service satisfies the deadline constraint. However, if a more restrained deadline constraint is considered, e.g. $x\leq 6$, the transition guards leading to l_d will have the form $t_0>6$, as shown in Figure 3. In this case, the location l_d is reachable from any non-final location. In general, if the location l_d is accessible from even a single location, then the deadline constraint becomes unsatisfied.

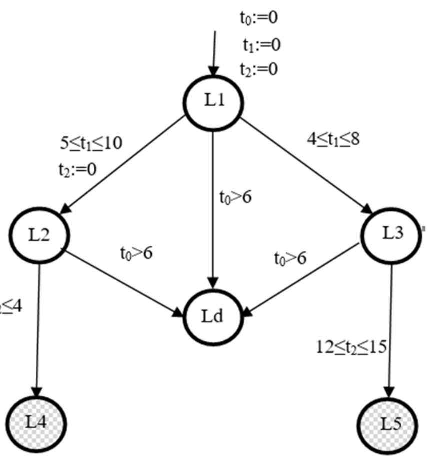
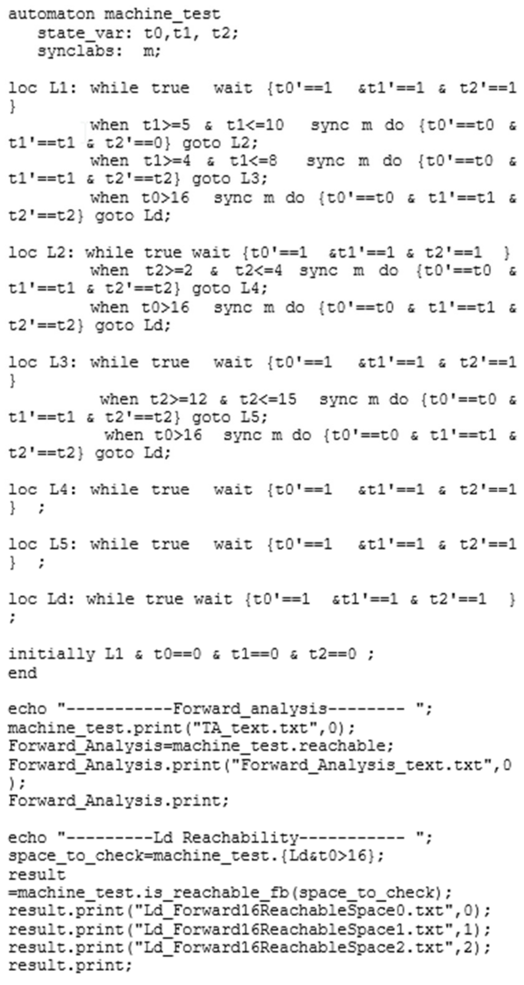
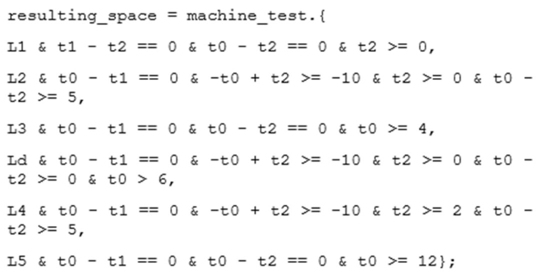


Fig. 3. Updated timed automaton with dead state and universal clock – l_d is reachable



```
automaton machine_test
   state_var: t0,t1, t2;
   synclabs:  m;

loc L1: while true  wait {t0'==1  &t1'==1 & t2'==1
}
          when t1>=5 & t1<=10  sync m do {t0'==t0 &
t1'==t1 & t2'==0} goto L2;
          when t1>=4 & t1<=8  sync m do {t0'==t0 &
t1'==t1 & t2'==t2} goto L3;
          when t0>16  sync m do {t0'==t0 & t1'==t1 &
t2'==t2} goto Ld;

loc L2: while true wait {t0'==1  &t1'==1 & t2'==1  }
          when t2>=2 & t2<=4 sync m do {t0'==t0 &
t1'==t1 & t2'==t2} goto L4;
          when t0>16  sync m do {t0'==t0 & t1'==t1 &
t2'==t2} goto Ld;

loc L3: while true  wait {t0'==1  &t1'==1 & t2'==1
}
           when t2>=12 & t2<=15  sync m do {t0'==t0 &
t1'==t1 & t2'==t2} goto L5;
            when t0>16  sync m do {t0'==t0 & t1'==t1 &
t2'==t2} goto Ld;

loc L4: while true  wait {t0'==1  &t1'==1 & t2'==1
}  ;

loc L5: while true  wait {t0'==1  &t1'==1 & t2'==1
}  ;

loc Ld: while true wait {t0'==1  &t1'==1 & t2'==1  }
;

initially L1 & t0==0 & t1==0 & t2==0 ;
end

echo "-----------Forward_analysis-------- ";
machine_test.print("TA_text.txt",0);
Forward_Analysis=machine_test.reachable;
Forward_Analysis.print("Forward_Analysis_text.txt",0
);
Forward_Analysis.print;

echo "---------Ld Reachability----------- ";
space_to_check=machine_test.{Ld&t0>16};
result
=machine_test.is_reachable_fb(space_to_check);
result.print("Ld_Forward16ReachableSpace0.txt",0);
result.print("Ld_Forward16ReachableSpace1.txt",1);
result.print("Ld_Forward16ReachableSpace2.txt",2);
result.print;
```

Fig. 4. Automata and computation of reachable space.



```
resulting_space = machine_test.{

L1 & t1 - t2 == 0 & t0 - t2 == 0 & t2 >= 0,

L2 & t0 - t1 == 0 & -t0 + t2 >= -10 & t2 >= 0 & t0 -
t2 >= 5,

L3 & t0 - t1 == 0 & t0 - t2 == 0 & t0 >= 4,

Ld & t0 - t1 == 0 & -t0 + t2 >= -10 & t2 >= 0 & t0 -
t2 >= 0 & t0 > 6,

L4 & t0 - t1 == 0 & -t0 + t2 >= -10 & t2 >= 2 & t0 -
t2 >= 5,

L5 & t0 - t1 == 0 & t0 - t2 == 0 & t0 >= 12};
```

Fig. 5. Reachable space.

For any web service modeled by a timed automaton, the problem of checking deadlines is decidable. By construction, checking the deadline respect on automaton *A* is equivalent to checking the reachability of l_d in automaton *A2*. If a transition is executed in *Ed,* then the universal clock x_0 exceeds the defined deadline, since the guard $x_0>Deadline$ becomes true. Furthermore, the reachability problem is known to be decidable for general timed automata [1], ensuring that the verification of the deadline is decidable. The proposed method to check a deadline is structural. It is sufficient to add a new l_d location, a new universal clock, and as many transitions as the cardinality of the locations set.

The proposed method was applied to the examples in Figures 2 and 3 using the Phaver tool [17]. Phaver allows the computation of reachable space of timed automata. The automata and the computation of reachable space were defined, conforming to the Phaver syntax, as shown in Figure 4. The result reachable space is shown in Figure 5.

D. How to Respect Deadline Constraints

Some web services may not meet some deadline requirements. When the number of clocks/locations increases, it is difficult to determine what are the possible updates on the behavior of the system to converge to the requirements. It is interesting to find a solution that describes the required values of the clocks with respect to the deadline constraints by following the following steps:

- Mark all locations where l_d is reachable. This set is denoted as L_d
- $\forall\, l \in L_d$, update *inv(l)* by adding the constraint $t0 \leq Deadline$
- Foreach $l \in L_d$, perform a backward analysis considering the reverse automata and the new location environment [16]
- Merge all the reachable space.

The result of these steps is a new automaton where the l_d location is never reachable. The implementability of these new constraints on the web service can be checked by the developers or designers based on the new constraints obtained on locations and guards. Moreover, it could be a starting point to enhance and upgrade the web service considering the new constraints.

IV. CONCLUSION

This paper focused on the problem of deadline verification for web services that could be modeled by timed automata. This framework offers the possibility to model the internal behavior of a web service with explicit consideration of timing constraints. The contribution of this paper is to propose a way to transform this problem into a reachability verification problem of a derived automaton. This new automaton is obtained by a structural and linear transformation. Future work consists of the generalization of this problem for a sequence of subtasks of a web service with multi-deadline constraints. Moreover, it is possible to tackle the deadline verification in the context of web service composition [4, 8]. In this case, it is necessary to deal with the synchronous composition of timed automata as modeling frameworks.

REFERENCES

[1] A. Bourouis, K. Klai, N. B. Hadj-Alouane, and Y. E. Touati, "On the Verification of Opacity in Web Services and Their Composition," *IEEE Transactions on Services Computing*, vol. 10, no. 1, pp. 66–79, Jan. 2017.

[2] A. Burouis, N. B. Hadj-Alouane, and K. Klai, "Computing Quantified Opacity for SOG-Abstracted Web Services," in *2017 IEEE International Conference on Services Computing (SCC)*, Jun. 2017, pp. 362–369, https://doi.org/10.1109/SCC.2017.53.

[3] S. Tiwari and P. Singh, "Survey of potential attacks on web services and web service compositions," in *2011 3rd International Conference on Electronics Computer Technology*, Apr. 2011, vol. 2, pp. 47–51, https://doi.org/10.1109/ICECTECH.2011.5941653.

[4] A. Bourouis, K. Klai, Y. E. Touati, and N. B. Hadj-Alouane, "Checking Opacity of Vulnerable Critical Systems On-The-Fly," *International Journal of Information Technology and Web Engineering (IJITWE)*, vol. 10, no. 1, pp. 1–30, Jan. 2015, https://doi.org/10.4018/ijitwe.2015010101.

[5] A. Bourouis, K. Klai, Y. El Touati, and N. B. Hadj-Alouane, "Opacity Preserving Abstraction for Web Services and Their Composition Using SOGs," in *2015 IEEE International Conference on Web Services*, Jun. 2015, pp. 313–320, https://doi.org/10.1109/ICWS.2015.50.

[6] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, Apr. 1994, https://doi.org/10.1016/0304-3975(94)90010-8.

[7] T. A. Henzinger, Z. Manna, and A. Pnueli, "Timed transition systems," in *Real-Time: Theory in Practice*, Berlin, Heidelberg, 1992, pp. 226–251, https://doi.org/10.1007/BFb0031995.

[8] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine, "Symbolic Model Checking for Real-Time Systems," *Information and Computation*, vol. 111, no. 2, pp. 193–244, Jun. 1994, https://doi.org/10.1006/inco.1994.1045.

[9] Y. El Touati, "Minimization of Linear Constraints in Constant Slope Hybrid Dynamic Systems," *International Journal of Advanced Computer Science and Applications*, vol. 18, no. 10, pp. 19–23, 2018.

[10] Y. E. Touati, S. Altowaijri, and M. Ayari, "Control of Industrial Systems to Avoid Failures: Application to Electrical System," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 5, 2018, https://doi.org/10.14569/IJACSA.2018.090561.

[11] Y. E. Touati, M. Ayari, and S. Altowaijri, "Extended Time Petri Net and Hybrid Petri Net : Modeling Multi- Instance Dynamic Hybrid Systems," *International Journal of Computer Science and Network Security*, vol. 18, no. 5, pp. 75–83, 2018.

[12] F. Barbon, P. Traverso, M. Pistore, and M. Trainotti, "Run-Time Monitoring of Instances and Classes of Web Service Compositions," in *2006 IEEE International Conference on Web Services (ICWS'06)*, Sep. 2006, pp. 63–71, https://doi.org/10.1109/ICWS.2006.113.

[13] R. Aggarwal, K. Verma, J. Miller, and W. Milnor, "Constraint driven Web service composition in METEOR-S," in *IEEE International Conference onServices Computing, 2004. (SCC 2004). Proceedings. 2004*, Sep. 2004, pp. 23–30, https://doi.org/10.1109/SCC.2004.1357986.

[14] E. Kirda, M. Jazayeri, C. Kerer, and M. Schranz, "Experiences in engineering flexible Web services," *IEEE MultiMedia*, vol. 8, no. 1, pp. 58–65, Jan. 2001, https://doi.org/10.1109/93.923954.

[15] M. Mao, J. Li, and M. Humphrey, "Cloud auto-scaling with deadline and budget constraints," in *2010 11th IEEE/ACM International Conference on Grid Computing*, Oct. 2010, pp. 41–48, https://doi.org/10.1109/GRID.2010.5697966.

[16] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, "What's Decidable about Hybrid Automata?," *Journal of Computer and System Sciences*, vol. 57, no. 1, pp. 94–124, Aug. 1998, https://doi.org/10.1006/jcss.1998.1581.

[17] G. Frehse, "PHAVer: Algorithmic Verification of Hybrid Systems Past HyTech," in *Hybrid Systems: Computation and Control*, Berlin, Heidelberg, 2005, pp. 258–273, https://doi.org/10.1007/978-3-540-31954-2_17.

[18] G. A. Tarnavsky and E. V. Vorozhtsov, "Cloud Computing in Science and Engineering and the 'SciShop.ru' Computer Simulation Center," *Engineering, Technology & Applied Science Research*, vol. 1, no. 6, pp. 133–138, Dec. 2011, https://doi.org/10.48084/etasr.87.

[19] B. O. Odedairo and V. Oladokun, "Relevance and Applicability of Multi-objective Resource Constrained Project Scheduling Problem: Review Article," *Engineering, Technology & Applied Science Research*, vol. 1, no. 6, pp. 144–150, Dec. 2011, https://doi.org/10.48084/etasr.53.

[20] F. H. Khoso, A. Lakhan, A. A. Arain, M. A. Soomro, S. Z. Nizamani, and K. Kanwar, "A Microservice-Based System for Industrial Internet of Things in Fog-Cloud Assisted Network," *Engineering, Technology & Applied Science Research*, vol. 11, no. 2, pp. 7029–7032, Apr. 2021, https://doi.org/10.48084/etasr.4077.