Corrections to “A Menagerie of Timed Automata”

JEROEN J. A. KEIREN, Open University of The Netherlands and Radboud University, Institute for Computing and Information Sciences, Nijmegen, The Netherlands
PETER FONTANA and RANCE CLEAVELAND, University of Maryland, USA

This note corrects a technical error in the *ACM Computing Surveys* article mentioned in the title. The flaw involved constructions for showing that timed automata with urgent locations have the same expressiveness as timed automata that allow false location invariants. Corrected constructions are presented in this note, and the affected results are reproved.

CCS Concepts: • **Theory of computation → Timed and hybrid models**; • **Software and its engineering → Software verification**; *Formal software verification;*

Additional Key Words and Phrases: Timed automata, timed transition systems, transition systems

ACM Reference Format:
Jeroen J. A. Keiren, Peter Fontana, and Rance Cleaveland. 2017. Corrections to “A Menagerie of Timed Automata”. ACM Comput. Surv. 50, 3, Article 42 (June 2017), 8 pages.
DOI: http://dx.doi.org/10.1145/3078809

1. INTRODUCTION

This note corrects a technical shortcoming in the *ACM Computing Surveys* article “A Menagerie of Timed Automata,” published January 3, 2014 [Fontana and Cleaveland 2014] (DOI: http://dx.doi.org/10.1145/2518102). This note often refers to that article for details to avoid repeating a large part of it. It is therefore advisable to have a copy of Fontana and Cleaveland [2014] for reference.

That article developed a unified framework for so-called *timed automata*, which extend traditional finite-state machines with real-valued clock variables. The states, or *locations*, in these timed automata are equipped with *location invariants* describing a property that must hold of the clock variables in order for control to remain within the given location. Some accounts of timed automata do not allow control to change into locations whose invariants are false; others permit this behavior, in which case time is not permitted to advance until control exits from the location.

In the baseline version of timed automata considered in the original article, transitions were not allowed into states whose location invariants would be violated by such a transition. However, states whose invariants were violated were allowed to engage in action transitions. Such states were not reachable from initial states, with the following exception: Initial states themselves were allowed to have invariant violations

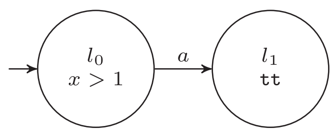
Authors’ addresses: J. J. A. Keiren, Open University of the Netherlands, Faculty of Management, Science & Technology, P. O. Box 2960, 6401 DL Heerlen, The Netherlands and Radboud Universiteit, Institute for Computing and Information Sciences, P. O. Box 9010, 6500 GL Nijmegen, The Netherlands; email: Jeroen.Keiren@ou.nl; P. Fontana and R. Cleaveland, University of Maryland, Department of Computer Science, A.V. Williams Building, 8223 Paint Branch Drive, College Park, MD, 20742.

DOI: http://dx.doi.org/10.1145/3078809

in them. Specifically, if $l_0 \in L_0$ and $\nu_0 \not\models I(l_0)$, then (l_0, ν_0) was nevertheless allowed to be an initial state.

This decision makes some of the semantic conversions contained in the article (and this note) easier, but it also may be viewed as being inconsistent with our treatment of invariants in non-initial states. In particular, one might wish for the following to be true in each reachable state (l, ν) in a timed automaton TA: $\nu \models I(l)$. This can fail to hold for initial states in TA, as illustrated in the following example.

Example 1.1. Consider the following timed automaton, in which invariant $x > 1$ in the initial location l_0 is not satisfied by the clock valuation $[x := 0]$ assigning 0 to the only clock x.



According to the baseline semantics, only states whose location invariant is satisfied may be entered. However, it does not disallow the situation in which the invariant is violated initially. Hence, the baseline semantics allows the executions $(l_0, [x := 0]) \xrightarrow{a} (l_1, [x := 0]) \xrightarrow{\delta} (l_1, [x := \delta])$ for all $\delta \in \mathbb{R}^{\geq 0}$, even though initially $I(l_0)$ is not satisfied.

In two places in the original article, an implicit assumption was made that when a location invariant is violated in a starting location, no behavior is possible in that location. As illustrated in the previous example, this assumption is at odds with assumptions made elsewhere in that article. As a result, two of the semantic transformations given in the article do not correctly handle invariant violations in initial locations. This note explains how the transformations may be modified so initial invariant violations are handled consistently. The parts of the article that this corrigendum addresses involve Section 5.1, where transformations in question are defined, and associated appendices, where proofs are given.

To facilitate the description of the timed-automaton conversions below, we recall [Fontana and Cleaveland 2014, Definitions 3.1 (clock constraints) and 3.2 (timed automaton)], which are Definitions 1.2 and 1.3 here.

Definition 1.2 (*Clock Constraint $\phi \in \Phi(CX)$ from Fontana and Cleaveland [2014]*). Given a nonempty finite set of clocks $CX = \{x_1, x_2, \ldots, x_n\}$ and $c \in \mathbb{Z}^{\geq 0}$ (a non-negative integer), a *clock constraint* ϕ may be constructed using the following grammar:

$$\phi ::= x_i < c \mid x_i \leq c \mid x_i > c \mid x_i \geq c \mid \phi \wedge \phi,$$

where $\Phi(CX)$ is the set of all possible clock constraints over CX. We also use the following abbreviations: true (tt) for $x_1 \geq 0$, false (ff) for $x_1 < 0$, and $x_i = c$ for $x_i \leq c \wedge x_i \geq c$.

Definition 1.3 (*Timed Automaton from Fontana and Cleaveland [2014]*). A *timed automaton* $TA = (L, L_0, L_u, \Sigma, CX, I, E)$ is a tuple where

—L is the finite set of *locations*.
—$L_0 \subseteq L$ is the nonempty set of *initial locations*.
—$L_u \subseteq L$ is the set of *urgent locations*.
—Σ is the finite set of *action symbols*.
—CX is the nonempty finite set of *clocks* ($CX = \{x_1, x_2, \ldots, x_n\}$).
—$I : L \longrightarrow \Phi(CX)$ gives a clock constraint for each location l. $I(l)$ is referred to as the *invariant* of l.

—$E \subseteq L \times \Sigma \times \Phi(CX) \times 2^{CX} \times L$ is the set of *edges*. In an edge $e = (l, a, \phi, \lambda, l')$ from l to l' with action a, $\phi \in \Phi(CX)$ is the *guard* of e, and $\lambda \in 2^{CX}$ represents the set of clocks to *reset* to 0 when the edge is executed.

One assumption made in that article, and in others involving timed automata, is that $\Sigma \cap \mathbb{R}^{\geq 0} = \emptyset$; in other words, Σ does not include any non-negative real numbers, which are reserved for use in the semantics of these automata.

In Fontana and Cleaveland [2014], timed automata are given a baseline semantics in the form of a translation function that maps a timed automaton $TA = (L, L_0, L_u, \Sigma, CX, I, E)$ to a timed transition system $TS(TA) = (Q, Q_0, \Delta(\Sigma), \longrightarrow)$, where the set of states Q consists of pairs of automaton locations and clock assignments (i.e., mappings of clocks to non-negative real numbers), $Q_0 \subseteq Q$ is the set of initial states, $\Delta(\Sigma) = \Sigma \cup \mathbb{R}^{\geq 0}$ is the set of transition labels (actions or time elapses), and $\longrightarrow \subseteq Q \times \Delta(\Sigma) \times Q$ is the transition relation.[1] The details of this construction may be found in Fontana and Cleaveland [2014, Definition 3.7] and forbids transitions into states (l, ν) where $\nu \not\models I(l)$ (i.e., the clock assignment violates the location invariant of location l. Satisfaction of clock valuations, $\models$, is made precise in the usual fashion).

In Fontana and Cleaveland [2014, Section 5.1], a semantic variant of timed automata is considered that weakens the restriction on transitions into transition-system states (l, ν) for which $\nu \not\models I(l)$. Specifically, the new semantics associates a transition system $TS'(TA) = (Q, Q_0, \Delta(\Sigma), \longrightarrow)$ with TA, where Q, Q_0, and $\Delta(\Sigma)$ retain the definitions above and $\longrightarrow$ is redefined as specified in the lower part of Fontana and Cleaveland [2014, page 20 in Section 5.1]. (The notation TS' is not used in the article but is introduced here to simplify the presentation.)

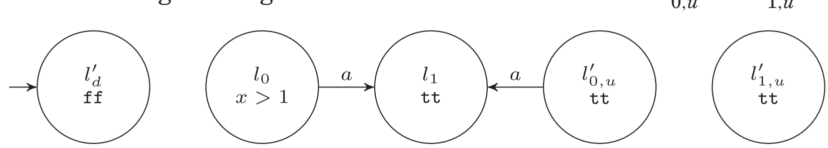
Two transformations are then given in Fontana and Cleaveland [2014, Section 5.1], INV and URG, that are intended to have the following properties. Given a timed automaton $TA = (L, L_0, \emptyset, \Sigma, CX, I, E)$ with an empty set of urgent locations, $INV(TA)$ has the property that $TS'(TA)$ and $TS(INV(TA))$ are semantically indistinguishable, in a precisely defined sense [Fontana and Cleaveland 2014, Theorem 5.5]. That is, TA interpreted in a semantics in which action transitions are allowed in states with location-invariant violations is equivalent to $INV(TA)$ interpreted in our baseline semantics. Similarly, given a baseline timed automaton TA, URG has the property that $TS(TA)$ and $TS'(URG(TA))$ are appropriately related [Fontana and Cleaveland 2014, Theorem 5.6].

The constructions INV and URG are the ones that this note redefines to eliminate the issues with violated invariants. The modified conversions are given, and Fontana and Cleaveland [2014, Theorems 5.5 and 5.6] reproved.

2. CONVERSION *INV* (TO BASELINE VERSION)

Before we continue defining the translation INV we first illustrate the problem with the translation in the original article.

Example 2.1. The timed automaton from Example 1.1 is translated into the following timed automaton using the original translation INV. Locations $l'_{0,u}$ and $l'_{1,u}$ are urgent.



[1]The original article does not introduce the notation $\Delta(\Sigma)$; we do so here for improved clarity.

The timed transition system underlying this automaton, using the baseline semantics, does not allow any transitions from the initial state, whereas the timed automaton from Example 1.1, interpreted in the unsatisfied invariants semantics, allows an a transition from the initial state. Therefore, the translation does not preserve the semantics of the original timed automaton.

The original version of *INV* incorrectly introduced dead locations l_d for initial locations whose invariants are not satisfied by the initial clock valuation. To correct the definition, let $TA = (L, L_0, \emptyset, \Sigma, CX, I, E)$ be a timed automaton with an empty set of urgent locations, and let $L_u = \{l_u \mid l \in L\}$ be a fresh set of locations with the property that $L_u \cap L = \emptyset$ and $l_u \neq l'_u$ if $l \neq l'$. Also let ν_0 be the clock valuation assigning 0 to every clock in CX. Finally, we recall Fontana and Cleaveland [2014, Definition 5.2] from the original article, which introduces of $\texttt{resetPred}(\phi, \lambda)$, where ϕ is a clock constraint and $\lambda \subseteq CX$ is a set of clocks to be reset. The constraint $\texttt{resetPred}(\phi, \lambda)$ may be viewed as the weakest precondition of ϕ with respect to the simultaneous assignment of each clock in λ to 0; it is the weakest property ϕ' such that if $\nu \models \phi'$, then $\nu[\lambda := 0] \models \phi$. We now redefine $INV(TA) = (L', L'_0, L'_u, \Sigma, CX, I', E')$ as follows:

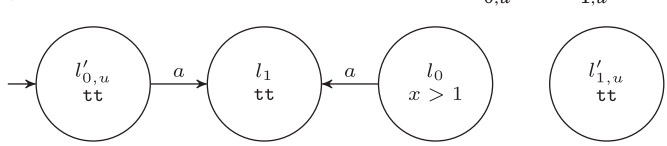
—$L' = L \cup L_u$.
—$L'_0 = \{l \in L_0 \mid \nu_0 \models I(l)\} \cup \{l_u \in L_u \mid l \in L_0 \wedge \nu_0 \not\models I(l)\}$.
—$L'_u = L_u$.
—$I'(l') = \begin{cases} I(l') & \text{if } l' \in L \\ \texttt{tt} & \text{otherwise}(i.e.\ 6\ l' \in L_u) \end{cases}$
—For each edge $(l, a, \phi, \lambda, l') \in E$, E' includes the following four edges, where $\phi_1 = \phi \wedge \texttt{resetPred}(I(l'), \lambda)$ and $\phi_2 = \phi \wedge \neg\texttt{resetPred}(I(l'), \lambda)$.

$$(l, a, \phi_1, \lambda, l'),$$
$$(l, a, \phi_2, \lambda, l'_u),$$
$$(l_u, a, \phi_1, \lambda, l'),$$
$$(l_u, a, \phi_2, \lambda, l'_u).$$

Disjunctive guard constraints may arise from negating $\texttt{resetPred}(I(l'), \lambda)$. Following the process used in Fontana and Cleaveland [2014, Section 4.1] of the original article, any disjunctive guard constraint is eliminated by converting the edge with such a constraint to a set of edges.

The key difference in the redefinition of *INV* involves L'_0. In the original construction, L'_0 was incorrectly taken to include a set of dead locations L_d to represent those initial locations whose invariants were violated by the initial clock assignment ν_0. In the new construction, initial locations $l \in L_0$ that are violated by the initial clock assignment ν_0 are replaced in L'_0 by their urgent versions l_u.

Example 2.2. The timed automaton from Example 1.1, interpreted in the unsatisfied invariants semantics, is translated into the following timed automaton in the baseline semantics using the fixed translation *INV*. Locations $l'_{0,u}$ and $l'_{1,u}$ are urgent.



It is not hard to see that the underlying timed transition system in the baseline semantics, when restricted to reachable states, is the same as that of the original timed automaton in the unsatisfied invariants semantics.

We now state and prove [Fontana and Cleaveland 2014, Theorem 5.5] from the original article, which is Theorem 2.3 here.

THEOREM 2.3. *Let $TA = (L, L_0, \emptyset, \Sigma, CX, I, E)$ be a timed automaton with an empty set of urgent locations. Then the reachable subsystems of $TS'(TA)$ and $TS(INV(TA))$ are isomorphic, that is, $TS'(TA)) \cong_r TS(INV(TA))$.*

PROOF OF THEOREM 2.3. Given transition system $T = (Q, Q_0, \Delta(\Sigma), \longrightarrow)$, define the reachable state space of T, $R(T) \subseteq Q$, to be the smallest subset of Q satisfying the following:

—$Q_0 \subseteq R(T)$;
—if $q \in R(T)$ and $q \xrightarrow{\alpha} q'$ for some $\alpha \in \Delta(\Sigma)$ then $q' \in R(T)$.

In what follows, we sometimes abuse notation and write $R(T)$ for the transition system $(R(T), Q_0, \Delta(\Sigma), \longrightarrow)$.

Let $TS'(TA) = (Q_1, Q_{0,1}, \Delta(\Sigma), \longrightarrow_1)$ and $TS(INV(TA)) = (Q_2, Q_{0,2}, \Delta(\Sigma), \longrightarrow_2)$. To prove the theorem, we must give an isormorphism f from $R(TS'(TA))$ to $R(TS(INV(TA)))$; specifically $f : R(TS'(TA)) \longrightarrow R(TS(INV(TA)))$ must have the following properties.

(1) f is one to one.
(2) $f(Q_{0,1}) = Q_{0,2}$.
(3) For every $q, q' \in R(TS'(TA))$ and $\alpha \in \Delta(\Sigma)$, $q \xrightarrow{\alpha}_1 q'$ iff $f(q) \xrightarrow{\alpha}_2 f(q')$.

We define $f : Q_1 \longrightarrow Q_2$ as follows and then show it is an isomorphism from $R(TS'(TA))$ to $R(TS(INV(TA)))$. So consider

$$f((l, \nu)) = \begin{cases} (l, \nu), & \text{if } \nu \models I(l) \\ (l_u, \nu) & \text{otherwise.} \end{cases}$$

We now prove that f has the necessary properties.

f is one to one. Suppose $f((l, \nu)) = f((l', \nu')) = (l'', \nu'')$; we must show that $l = l'$ and $\nu = \nu'$. From the definition of f it follows that if $f((l, \nu)) = (l'', \nu'')$, then $\nu = \nu''$; hence, $\nu = \nu' = \nu''$. Moreover, if $\nu \models I(l)$, then $l'' = l$, whence it must be the case in $l = l'$. Finally, if $\nu \not\models I(l)$, then $l'' = l_u$, and again it follows that $l = l'$.

$f(Q_{0,1}) = Q_{0,2}$. Suppose $(l, \nu) \in Q_{0,1}$; we must show that $f((l, \nu)) = (l', \nu') \in Q_{0,2}$. First, note that $l \in L_0$ and $\nu = \nu_0$, and thus $\nu' = \nu_0$ by the definition of f. By the definition of *INV*, it also follows that $l' = l$ if $\nu_0 \models I(l)$ and $l' = l_u$ otherwise. In either case, $l' \in L_0'$ and $(l', \nu') \in Q_{0,2}$. Now suppose $(l', \nu') \in Q_{0,2}$; we must give $(l, \nu) \in Q_{0,1}$ such that $f((l, \nu)) = (l', \nu')$. As before, $\nu' = \nu_0$ by definition of $Q_{0,2}$. Now either $l' \in L_0$, meaning $\nu_0 \models I(l')$ and thus $f((l', \nu')) = (l', \nu)$, or $l' = l_u$ for some $l \in L_0$ with $\nu_0 \not\models I(l)$; in this case, $f((l, \nu_0)) = (l', \nu')$, with $(l, \nu_0) \in Q_{0,1}$.

$(l, \nu) \xrightarrow{\alpha}_1 (l', \nu')$ iff $f((l, \nu)) \xrightarrow{\alpha}_2 f((l', \nu'))$. There are two cases to consider: $\alpha = \delta$ for some $\delta \in \mathbb{R}^{\geq 0}$ or $\alpha \in \Sigma$. So suppose $\alpha = \delta$ for some $\delta \geq 0$. Now, $(l, \nu) \xrightarrow{\delta}_1 (l', \nu')$ iff $l = l'$, $\nu' = \nu + \delta$, and for all k such that $0 \leq k \leq \delta$, $\nu + k \models I(l)$. This in turn holds iff $l = l'$, $f((l, \nu)) = (l, \nu)$, $f((l', \nu')) = (l', \nu')$, and $f((l, \nu)) \xrightarrow{\delta}_2 f((l', \nu'))$.

Now consider the case where $\alpha = a \in \Sigma$ and suppose $(l, \nu) \xrightarrow{a}_1 (l', \nu')$ is an action transition, meaning there is an edge $e = (l, a, \phi, \lambda, l') \in E$ such that $\nu \models \phi$ and $\nu' = \nu[\lambda := 0]$. We must show that $f((l, \nu)) \xrightarrow{a}_2 f((l', \nu'))$. There are four cases to consider.

—$\nu \models I(l)$ and $\nu' \models I(l')$. In this case, $f((l, \nu)) = (l, \nu)$, $f((l', \nu')) = (l', \nu')$, and $\nu \models \phi_1$ as defined in *INV*. By our conversion, we have the edge $e = (l, a, \phi \cap \texttt{resetPred}(I(l'), \lambda), \lambda, l')$ in *INV*(TA) and, $f((l', \nu[\lambda := 0])) = (l', \nu[\lambda := 0])$. Since we know $\nu[\lambda := 0] \models I(l')$ and $\nu \models \phi$, by Corollary B.5, we know $\nu \models \phi \cap \mathtt{resetPred}(I(l'), \lambda)$. Therefore, $INV(TA)$ has the transition $f((l, \nu)) \xrightarrow{a} f((l', \nu[\lambda := 0]))$.

—$\nu \models I(l)$ and $\nu' \not\models I(l')$. In this case, $f((l, \nu)) = (l, \nu)$, $f((l', \nu')) = (l'_u, \nu')$, and $\nu \models \phi_2$ as defined in INV. By our conversion, we use the edge $e_u = (l_u, a, \phi \cap \mathtt{resetPred}(I(l'), \lambda), \lambda, l')$ in $INV(TA)$. Otherwise, the proof is the same as the previous case's. It therefore follows that $f((l, \nu)) \xrightarrow{a}_2 f((l', \nu'))$, since the edge $(l, a, \phi_2, \lambda, l'_u)$ is in the edge set of $INV(TA)$.

—$\nu \not\models I(l)$ and $\nu' \models I(l')$. In this case, $f((l, \nu)) = (l_u, \nu)$, $f((l', \nu')) = (l', \nu')$, and $\nu \models \phi_1$ as defined in INV. By our conversion, $f((l', \nu[\lambda := 0])) = (l'_u, \nu[\lambda := 0])$. Since l'_u is the urgent copy of l', we know $\nu[\lambda := 0] \models I(l'_u)$. Since $\nu \models \phi$, by Corollary B.5, we know that $\nu \models \phi \cap \neg\mathtt{resetPred}(I(l'), \lambda)$. By the definition of the transition system semantics, $INV(TA)$ has the transition $f((l, \nu)) \xrightarrow{a} f(l', \nu[\lambda := 0])$.

—$\nu \not\models I(l)$ and $\nu' \not\models I(l')$. In this case, $f((l, \nu)) = (l_u, \nu)$, $f((l', \nu')) = (l'_u, \nu')$, and $\nu \models \phi_2$ as defined in INV. By our conversion, we use the edge $e_u = (l_u, a, \phi \cap \neg\mathtt{resetPred}(I(l'), \lambda), \lambda, l'_u)$ in $INV(TA)$. Otherwise, the proof is the same as the previous case's. It therefore follows that $f((l, \nu)) \xrightarrow{a}_2 f((l', \nu'))$, since the edge $(l_u, a, \phi_2, \lambda, l'_u)$ is in the edge set of $INV(TA)$.

For the converse, we assume that $f((l, \nu)) \xrightarrow{a}_2 f((l', \nu'))$ and must show that $(l, \nu) \xrightarrow{a}_1 (l', \nu')$. The argument follows the lines above and relies on a case analysis of which of the four types of edges in $INV(TA)$ supports the conclusion that $f((l, \nu)) \xrightarrow{a}_2 f((l', \nu'))$. The details are omitted (one may wish to use the definition of f^{-1} when proving the converse).

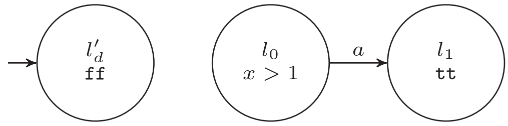
$(l, \nu) \in R(TS'(TA))$ iff $f((l, \nu)) \in R(TS(INV(TA)))$. This can be proved by induction on the definition of $R(\cdot)$ and is a consequence of the fact that $f(Q_{0,1}) = Q_{0,2}$ and that $(l, \nu) \xrightarrow{a}_1 (l', \nu')$ iff $f((l, \nu)) \xrightarrow{a}_2 f((l', \nu'))$. Alternatively, to show this, suppose we have a state (l_{inv}, ν_{inv}) in $INV(TA)$ that is not mapped to by f. We claim (l_{inv}, ν_{inv}) is not reachable from an initial state. By the definition of f, if l_{inv} is not an urgent copy and $\nu_{inv} \models I(l_{inv})$, then (l_{inv}, ν_{inv}) is covered by f. Likewise, if l_{inv} is an urgent copy location l_u and $(l, \nu_{inv}) \not\models I(l)$, then (l_{inv}, ν_{inv}) is covered by f. If l_{inv} is an urgent copy location l_u and $(l, \nu_{inv}) \models I(l)$, then by the construction of $INV(TA)$, this state is not reachable. If l_{inv} is not an urgent copy and $\nu_{inv} \not\models I(l_{inv})$, then by the semantics of $INV(TA)$, (l_{inv}, ν_{inv}) is only reachable if and only it is an initial state. Furthermore, by construction, only urgent initial locations do not satisfy their invariant. Thus, applying f to a reachable state in the original timed automaton results in a reachable state in the converted timed automaton.

f is an Isomorphism from $R(TS'(TA))$ to $R(TS(INV(TA)))$. This conclusion is a consequence of the previous facts. Since f is one to one on Q_1, it is one to one when restricted to $R(TS'(TA)) \subseteq Q_1$. Moreover, as $(l, \nu) \in R(TS'(TA))$ iff $f((l, \nu)) \in R(TS(INV(TA)))$, it follows that $f(R(TS'(TA)) = R(TS(INV(TA)))$, meaning f when restricted to $R(TS'(TA))$ is onto with respect to $R(TS(INV(TA)))$. Hence f is a bijection from $R(TS'(TA))$ to $R(TS(INV(TA)))$ that preserves start states and the transition relation and is therefore the required isomorphism. □

3. CONVERSION *URG* (FROM BASELINE VERSION)

The original definition of *URG* also erroneously introduced dead locations for locations in *TA* that were not satisfied by the initial clock assignment.

Example 3.1. The timed automaton from Example 1.1 is translated into the following timed automaton using the original translation *URG*.



Using the unsatisfied-invariants semantics, the timed transition system underlying this automaton does not allow any transitions from the initial state, whereas the timed automaton from Example 1.1, interpreted in the baseline semantics, allows an a transition from the initial state. The translation thus does not preserve the semantics.

We now detail the modified construction *URG*, which converts timed automata from our baseline formalism into automata permitting transitions into states with location invariants, and reprove the associated correctness result.

Specifically, let $TA = (L, L_0, L_u, \Sigma, CX, I, E)$ be a timed automaton. We wish to define another timed automaton, $URG(TA) = (L', L'_0, \emptyset, \Sigma, CX, I', E')$, with an empty set of urgent locations, so $TS(TA)$ and $TS'(URG(TA))$ are isomorphic in an appropriate sense. $URG(TA)$ may be given as follows:

—$L' = L$
—$L'_0 = L_0$
—$I'(l) = \begin{cases} I(l) & \text{if } l \notin L_u \\ \texttt{ff} & \text{otherwise.} \end{cases}$
—$E' = \{(l, a, \phi \wedge \texttt{resetPred}(I(l'), \lambda), \lambda, l') \mid (l, a, \phi, l, \lambda, l') \in E\}$.

Example 3.2. The timed automaton from Example 1.1 is not modified by the new translation *URG*. We have already observed that the underlying timed transition system is the same for the baseline and the unsatisfied invariants semantics.

We now restate and reprove Fontana and Cleaveland [2014, Theorem 5.6], which is Theorem 3.3 here.

THEOREM 3.3. *Let* $TA = (L, L_0, L_u, \Sigma, CX, I, E)$ *be a timed automaton. Then* $TS(TA)$ *and* $TS'(URG(TA))$ *are isomorphic, that is,* $TS'(TA) \cong TS(INV(TA))$.

PROOF OF THEOREM 3.3. It should first be noted that both transition systems $TS(TA)$ and $TS'(URG(TA))$ have the same set of states and initial states. Unlike the proof of correctness of *INV*, in this case full isomorphism of the timed transition systems of $TS(TA) = (Q, Q_0, \Delta(\Sigma), \longrightarrow_1)$ and $TS'(URG(TA)) = (Q, Q_0, \Delta(\Sigma), \longrightarrow_2)$ can be established. Consider the function f:

$$f : Q \longrightarrow Q$$
$$f((l, \nu)) = (l, \nu)$$

or the identity function. We must show that that f is an isomorphism from $TS(TA)$ to $TS'(URG(TA))$. That f is a bijection, and that $f(Q_0) = Q_0$, follow from f being the identity function. It remains to show that $(l, \nu) \xrightarrow{\alpha}_1 (l', \nu')$ iff $f((l, \nu)) \xrightarrow{\alpha}_2 f((l', \nu'))$ for all $\alpha \in \Delta(\Sigma)$. There are two cases to consider.

—$(l, \nu) \xrightarrow{\delta}_1 (l', \nu')$ some $\delta \geq 0$. This happens iff $l \notin L_u$, $l = l'$, $\nu' = \nu + \delta$, and for all k such that $0 \leq k \leq \delta$, $\nu + k \models I(l)$, which holds iff $f((l, \nu)) = (l, \nu) \xrightarrow{\delta}_2 (l', \nu') = f((l', \nu'))$.
—$(l, \nu) \xrightarrow{a}_1 (l', \nu')$ some $a \in \Sigma$. This happens iff there exists an edge $(l, a, \phi, \lambda, l') \in E$ such that $\nu \models \phi$, $\nu' = \nu[\lambda := 0]$, and $\nu' \models I(l')$, which in turn is logically equivalent to asserting the existence of an edge $(l, a, \phi, \lambda, l') \in E$ such that $\nu' = \nu[\lambda := 0]$ and $\nu \models \phi \wedge \texttt{resetPred}(\phi, \lambda)$. This holds iff there is an edge $(l, a, \phi \wedge \texttt{resetPred}(I(l'), \lambda), \lambda, l') \in E'$

iff (by Corollary B.5) $\nu \models \phi \cap \texttt{resetPred}(I(l'), \lambda)$, which in turn holds iff $f((l, \nu)) = (l, \nu) \xrightarrow{a}_2 (l', \nu') = f((l', \nu'))$. $\square$

REFERENCE

Peter Fontana and Rance Cleaveland. 2014. A menagerie of timed automata. *Comput. Surv.* 46, 3 (Jan. 2014), 40:1–40:56. DOI: http://dx.doi.org/10.1145/2518102

Received August 2015; revised July 2016; accepted April 2017