Synthesizing Timed Automata with Minimal Numbers of Clocks from Optimised Timed Scenarios

Neda Saeedloei[1] and Feliks Kluźniak

Towson University
nsaeedloei@towson.edu

Abstract. We address the problem of synthesizing a timed automaton from a set of optimised timed scenarios, and present a simple, efficient algorithm that solves the problem. Under a simplifying assumption about the set of scenarios we show that our synthesized automaton has the minimal number of clocks in the entire class of language-equivalent automata.

1 Introduction

Using scenarios for specifying complex systems (including real time systems and distributed systems [1, 2]), and synthesizing formal models of systems from scenarios have been active areas of research for several decades [3–10].

In our earlier work [11] we developed, from first principles, a formal, yet simple notation for timed scenarios. Intuitively, a scenario is a sequence of events along with a set of constraints between the times of these events, which can be used to specify the partial behaviours of a system or a component of a system (see Sec. 2.2 for more details).

We want to use such scenarios to automatically synthesize formal models in the form of timed automata. Verification of a timed automaton can be computationally expensive, and the cost depends on the number of clock regions of the automaton. The number of clock regions is exponential in the number of clocks [12][1]. We are therefore interested in the problem of synthesizing a timed automaton with a minimal number of clocks from a set of scenarios.

Previously, we studied the problem in the more limited setting of a single timed scenario [13], and proposed an algorithm for "optimising" scenarios [13]. Given a scenario, our optimisation algorithm [13] replaces the time constraints of the scenario with an equivalent set that would require the smallest number of clocks in the entire class of equivalent scenarios, when the scenarios are viewed as timed automata. *Optimality was achieved under the assumption that a timed scenario cannot be split into two [13].*

[1] For a timed automaton with $|K|$ clocks, the number of clock regions is at most $R = |K|!4^{|K|}\Pi_{x\in K}(\mu_x + 1)$, where μ_x is the maximum constant with which clock x is compared [12].

More recently, we developed the notions of intersection, union and subsumption for scenarios [14]. We introduced appropriate operations with well-defined semantics for computing the intersection and union of two consistent scenarios, as well as for determining whether a scenario is subsumed by another one.

The contributions of the current paper are as follows:

- We use the aforementioned developments to show that even when a scenario is split into two, the number of clocks does not decrease, thereby strengthening our previous result. The consequence of this is that, if γ is the union—the reverse of splitting—of two scenarios ξ and η, then the number of clocks in the automaton corresponding to γ is not larger than that for automata corresponding to ξ and η.
- We present an efficient algorithm that, given a set of scenarios, constructs a timed automaton such that
  - the number of locations of the automaton is reasonably small (though not necessarily minimal);
  - the number of clocks of the automaton does not exceed that needed by at least one of the constituent scenarios.

  It turns out that if we make a simplifying assumption about the initial set of scenarios, our synthesized automaton has the minimal number of clocks in the entire class of language-equivalent timed automata.
  We then briefly discuss the consequences of relaxing our simplifying assumption, and show that such relaxation would require an additional preprocessing stage of significant computational complexity.

2 Preliminaries

2.1 Timed automata

A *timed automaton* [15] is a tuple $\mathcal{A} = \langle \Sigma, Q, q_0, Q_f, C, T \rangle$, where Σ is a finite alphabet, Q is the (*finite*) set of locations, $q_0 \in Q$ is the initial location, $Q_f \subseteq Q$ is the set of final locations, C is a finite set of *clock* variables (clocks for short), and $T \subseteq Q \times Q \times \Sigma \times 2^C \times 2^{\Phi(C)}$ is the set of transitions. In each transition $(q, q', e, \lambda, \phi)$, λ is the set of clocks to be reset with the transition and $\phi \subset \Phi(C)$ is a set of clock constraints over C of the form $c \sim a$ (where $\sim \in \{\leq, <, \geq, >, =\}$, $c \in C$ and a is a constant in the set of rational numbers, $\mathbb{Q}$).

A *clock valuation* ν for C is a mapping from C to $\mathbb{R}^{\geq 0}$. Clock valuation ν *satisfies* $\phi \subset \Phi(C)$ iff every clock constraint in ϕ evaluates to true after each clock c is replaced with $\nu(c)$. For $\tau \in \mathbb{R}, \nu + \tau$ denotes the clock valuation which maps every clock c to the value $\nu(c) + \tau$. For $Y \subseteq C$, $[Y \mapsto \tau]\nu$ is the valuation which assigns τ to each $c \in Y$ and agrees with ν over the rest of the clocks.

A *timed word* over an alphabet Σ is a pair (σ, τ) where $\sigma = \sigma_1\sigma_2...$ is a finite [16, 17] or infinite [15] word over Σ and $\tau = \tau_1\tau_2...$ is a finite or infinite sequence of (time) values such that (i) $\tau_i \in \mathbb{R}^{\geq 0}$, (ii) $\tau_i \leq \tau_{i+1}$ for all $i \geq 1$, and (iii) if the word is infinite, then for every $t \in \mathbb{R}^{\geq 0}$ there is some $i \geq 1$ such that $\tau_i > t$.

A run ρ of $\mathcal{A}$ over a timed word (σ, τ) is a sequence of the form $\langle q_0, \nu_0 \rangle \xrightarrow[\tau_1]{\sigma_1} \langle q_1, \nu_1 \rangle \xrightarrow[\tau_2]{\sigma_2} \langle q_2, \nu_2 \rangle \xrightarrow[\tau_3]{\sigma_3} \ldots$, where for all $i \geq 0$, $q_i \in Q$ and ν_i is a clock valuation

such that (i) $\nu_0(c) = 0$ for all clocks $c \in C$ and (ii) for every $i > 1$ there is a transition in T of the form $(q_{i-1}, q_i, \sigma_i, \lambda_i, \phi_i)$, such that $(\nu_{i-1}+\tau_i-\tau_{i-1})$ satisfies ϕ_i, and ν_i equals $[\lambda_i \mapsto 0](\nu_{i-1} + \tau_i - \tau_{i-1})$. The set $inf(\rho)$ consists of $q \in Q$ such that $q = q_i$ for infinitely many $i \geq 0$ in the run ρ.

A run over a finite timed word is *accepting* if it ends in a final location [17]. A run ρ over an infinite timed word is *accepting* iff $inf(\rho) \cap Q_f \neq \emptyset$ [15]. The *language* of $\mathcal{A}$, $L(\mathcal{A})$, is the set $\{(\sigma, \tau) \mid \mathcal{A}$ has an accepting run over $(\sigma, \tau)\}$.

2.2 Timed scenarios

This subsection briefly recounts our earlier work [11, 18][2].

Let Σ be a finite set of symbols called *events*. A *behaviour*[3] over Σ is a sequence $(e_0, t_0)(e_1, t_1)(e_2, t_2)\dots$, such that $e_i \in \Sigma$, $t_i \in \mathbb{R}^{\geq 0}$ and $t_{i-1} \leq t_i$ for $i \in \{1, 2\dots\}$. For a finite behaviour $\mathcal{B} = (e_0, t_0)(e_1, t_1)\dots(e_{n-1}, t_{n-1})$ of length n, and for any $0 \leq i < j < n$, the *distance*, in time units, of event j from event i in $\mathcal{B}$ is denoted by $t^{\mathcal{B}}_{ij}$. That is, $t^{\mathcal{B}}_{ij} = t_j - t_i$.

A *timed scenario* (*scenario* for short) of length $n \in \mathbb{N}$ over Σ is a pair $(\mathcal{E}, \mathcal{C})$, where $\mathcal{E} = e_o e_1 ... e_{n-1}$ is a sequence of events, and $\mathcal{C} \subset \Phi(n)$ is a finite set of constraints. Each constraint in $\Phi(n)$ is of the form $b \sim a$, where b is the symbol $\tau_{i,j}$ (for some integers $0 \leq i < j < n$), $\sim \in \{\leq, \geq\}$ [4] and a is a constant in the set of rational numbers, $\mathbb{Q}$. The interpretation is that $\tau_{i,j}$ is the time distance between the i-th and the j-th events in the behaviours described by a scenario. The constraints $\tau_{i,j} \geq 0$ and $\tau_{i,j} \leq \infty$ are called *default constraints*.

A behaviour $\mathcal{B} = (e_0, t_0)(e_1, t_1)\dots(e_{n-1}, t_{n-1})$ over Σ is *allowed* by scenario $\xi = (\mathcal{E}, \mathcal{C})$ iff $\mathcal{E} = e_0 \dots e_{n-1}$ and every $\tau_{i,j} \sim a$ in $\mathcal{C}$ evaluates to true after $\tau_{i,j}$ is replaced by $t^{\mathcal{B}}_{ij}$. If $\mathcal{B}$ is allowed by ξ, then we say $\mathcal{B}$ satisfies all constraints of ξ.

The *semantics* of scenario ξ, denoted by $[\![\xi]\!]$, is the set of behaviours that are allowed by ξ. A scenario ξ is *consistent* iff $[\![\xi]\!] \neq \emptyset$. It is *inconsistent* iff $[\![\xi]\!] = \emptyset$.

Fig. 1 shows the "external representations"[5] of two scenarios. The one on the left corresponds to scenario $\gamma = (abc, \{\tau_{0,1} \leq 4, \tau_{0,2} \geq 6, \tau_{0,2} \leq 6, \tau_{1,2} \geq 3\})$. $[\![\gamma]\!] = \{(a, t_0)(b, t_1)(c, t_2) \mid t_0 \leq t_1 \leq t_2 \wedge t_1 - t_0 \leq 4 \wedge t_2 - t_0 = 6 \wedge t_2 - t_1 \geq 3\}$.

Two scenarios $\xi = (\mathcal{E}, \mathcal{C}_1)$ and $\eta = (\mathcal{E}, \mathcal{C}_2)$ are *equivalent* iff $[\![\xi]\!] = [\![\eta]\!]$. For example, γ and η of Fig. 1 are equivalent.

For a consistent scenario ξ of length n, and for $0 \leq i < j < n$, $m^{\xi}_{ij} = min\{t^{\mathcal{B}}_{ij} \mid \mathcal{B} \in [\![\xi]\!]\}$ and $M^{\xi}_{ij} = max\{t^{\mathcal{B}}_{ij} \mid \mathcal{B} \in [\![\xi]\!]\}$[6]. For any behaviour $\mathcal{B}$ in $[\![\xi]\!]$, $0 \leq m^{\xi}_{ij} \leq t^{\mathcal{B}}_{ij} \leq M^{\xi}_{ij} \leq \infty$. Moreover, the following inequations hold [11]:

$$m^{\xi}_{ij} + m^{\xi}_{jk} \leq m^{\xi}_{ik} \leq \left\{ \begin{matrix} m^{\xi}_{ij} + M^{\xi}_{jk} \\ M^{\xi}_{ij} + m^{\xi}_{jk} \end{matrix} \right\} \leq M^{\xi}_{ik} \leq M^{\xi}_{ij} + M^{\xi}_{jk} \qquad (1)$$

[2] The comparison of our timed scenarios and other notions of scenarios can be found elsewhere [10, 11].

[3] The notion of "behaviour" is equivalent to that of Alur's "timed word" [15]. We found the term "behaviour" more suitable and intuitive in the context of timed scenarios.

[4] To keep the presentation compact, sharp inequalities are not allowed [11]. Equality is expressed in terms of $\leq$ and $\geq$.

[5] Equality will be expressed directly using $=$ in the external representation.

[6] The absence of an upper bound for some i and j will be denoted by $M^{\xi}_{ij} = \infty$.

$0 : a$;
$1 : b$ $\{\tau_{0,1} \leq 4\}$;
$2 : c$ $\{\tau_{0,2} = 6, \tau_{1,2} \geq 3\}$.

γ

$0 : a$;
$1 : b$ $\{\tau_{0,1} \leq 3\}$;
$2 : c$ $\{\tau_{0,2} = 6\}$.

η

Fig. 1. Two equivalent scenarios

	1	2
0	(0, 4)	(6, 6)
1		(3, ∞)

	1	2
0	(0, 3)	(6, 6)
1		(3, 6)

Fig. 2. initial table of γ and stable table of γ and η (γ and η are equivalent)

Let $\xi = (\mathcal{E}, \mathcal{C})$ be a scenario of length n, such that, for any $0 \leq i < j < n$, $\mathcal{C}$ contains at most one constraint of the form $\tau_{i,j} \geq c$ and at most one of the form $\tau_{i,j} \leq c$. A *distance table* for ξ is a representation of $\mathcal{C}$ in the form of a triangular matrix $\mathcal{D}^\xi$: $\mathcal{D}^\xi[i,j] = (l^\xi_{ij}, h^\xi_{ij})$. If $\tau_{i,j} \geq c \in \mathcal{C}$ then $l^\xi_{ij} = c$, otherwise $l^\xi_{ij} = 0$; if $\tau_{i,j} \leq c \in \mathcal{C}$ then $h^\xi_{ij} = c$, otherwise $h^\xi_{ij} = \infty$. The distance table corresponding to γ of Fig. 1 is shown on the left of Fig. 2.

A distance table of size n for ξ is *valid* iff $l^\xi_{ij} \leq h^\xi_{ij}$, for all $0 \leq i < j < n$. A table that is not valid is *invalid*. If $\mathcal{D}^\xi$ is invalid, then ξ is obviously inconsistent.

A valid distance table of size n for ξ is *stable* iff, for all $0 \leq i < j < k < n$, the inequations in (1) hold when m^ξ_{ij}, m^ξ_{jk}, m^ξ_{ik} are replaced by l^ξ_{ij}, l^ξ_{jk}, l^ξ_{ik}, and M^ξ_{ij}, M^ξ_{jk}, M^ξ_{ik} are replaced by h^ξ_{ij}, h^ξ_{jk}, h^ξ_{ik}. If $\mathcal{D}^\xi$ is stable then ξ is consistent. A consistent scenario ξ can be stabilized by an algorithm whose cost is $O(n^3)$ (n is the length of ξ) [11]. The algorithm increases the values of minima and decreases the values of maxima until the inequations in (1) are satisfied.

A stable distance table[7] has two properties. First, the table includes all the constraints that are implied [18] by the initial set of constraints. Second, all the constraints represented by the table are as *tight* as possible. In other words, if ξ is a scenario of length n and $\mathcal{D}^\xi_s$ is its stable table, then for every $0 \leq i < j < n$, $l^\xi_{ij} = m^\xi_{ij}$ and $h^\xi_{ij} = M^\xi_{ij}$, that is, $\mathcal{D}^\xi_s[i,j] = (m^\xi_{ij}, M^\xi_{ij})$. $\mathcal{D}^\xi_s[i,j]$ specifies the set of all the possible values of t_{ij} that can appear in the behaviours allowed by ξ.

The table on the right of Fig. 2 is the stable distance table of γ in Fig. 1. $\mathcal{D}^\gamma_s$ represents the set of constraints $\{\tau_{0,1} \geq 0, \tau_{0,1} \leq 3, \tau_{0,2} \geq 6, \tau_{0,2} \leq 6, \tau_{1,2} \geq 3, \tau_{1,2} \leq 6\}$. Observe that $\mathcal{D}^\gamma_s[0,1] = (0,3)$: $m^\gamma_{01} = 0$, which corresponds to a default constraint, and $M^\gamma_{01} = 3$, which means the original constraint $\tau_{0,1} \leq 4$ was not tight.

Two scenarios ξ and η are equivalent iff $\mathcal{D}^\xi_s = \mathcal{D}^\eta_s$. For example, the stable table on the right-hand side of Fig. 2 could be obtained from the constraints of either γ or η, which shows that they are equivalent: $[\![\gamma]\!] = [\![\eta]\!]$.

Timed scenarios and timed automata If $\xi = (\mathcal{E}, \mathcal{C})$ is a scenario of length n, and $\mathcal{C}$ contains a constraint $\tau_{i,j} \sim a$ for some $0 \leq i < j < n$, and some $a \in \mathbb{Q}$, then the index i is an *anchor*. If $0 < j < n$ is the largest number such that $\tau_{i,j} \sim b$ is a constraint in $\mathcal{C}$, then $[i,j)$ is the *range* of anchor i. If i_1 and i_2 are two anchors with ranges $[i_1, j_1)$ and $[i_2, j_2)$ in ξ, then the two ranges *overlap* iff

[7] A detailed comparison with Dill's Difference Bounds Matrices (DBMs) [19] can be found in our earlier work [18].

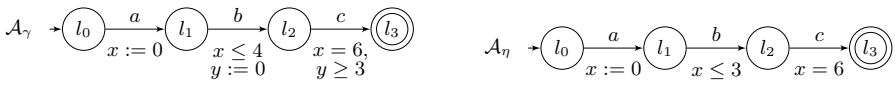

Fig. 3. Equivalent timed automata corresponding to the scenarios of Fig. 1

$i_1 < i_2 < j_1$ or $i_2 < i_1 < j_2$. For example, in scenario γ of Fig. 1, the range of anchor 0 is $[0, 2)$ and the range of anchor 1 is $[1, 2)$: these are overlapping. $Anch_\xi$ is used to denote the set of anchors of ξ. If X is a set of clock variables, then a relation $alloc_\xi \subset Anch_\xi \times X$ is a clock allocation for ξ. $alloc_\xi$ is *complete* iff for every anchor $i \in Anch_\xi$ there is a clock $x \in X$ such that $(i, x) \in alloc_\xi$. $alloc_\xi$ is *incorrect* iff there exist two different anchors i and j in $Anch_\xi$ whose ranges overlap, such that $(i, x) \in alloc_\xi$ and $(j, x) \in alloc_\xi$ for some $x \in X$. $alloc_\xi$ is *correct* iff it is not incorrect. A correct and complete clock allocation is *optimal* if there is no other correct and complete allocation that uses fewer clocks. $\{(0, x), (1, y)\}$ is an optimal clock allocation for scenario γ of Fig. 1.

A scenario ξ can be trivially converted to a simple timed automaton $\mathcal{A}_\xi$, such that the language of $\mathcal{A}_\xi$ is equivalent to the set of behaviours allowed by ξ: $L(\mathcal{A}_\xi) = [\![\xi]\!]$. The conversion preserves the clock allocation. For example, Fig. 3 shows the automata obtained from scenarios γ and η of Fig. 1.

A scenario ξ can be transformed to an equivalent scenario η by *optimising* its set of constraints [13], so that (i) η has a minimal set of constraints[8], and (ii) given an optimal clock allocation for η, $\mathcal{A}_\eta$ has the *minimal* number of clocks in the entire class of automata that are language-equivalent to $\mathcal{A}_\xi$[9]. We call η the *optimised* form of ξ. For example, η of Fig. 1 is the optimised form of γ in that figure. Notice that in Fig. 3 $\mathcal{A}_\eta$ has only one clock, while $\mathcal{A}_\gamma$ requires two clocks.

For an optimised scenario $\xi = (\mathcal{E}, \mathcal{C})$ the members of $\mathcal{C}$ will be referred to as *explicit constraints*. We know that in the stable distance table, $\mathcal{D}_s^\xi$, there are also *implicit* constraints: default constraints and constraints that are implied by $\mathcal{C}$. For example, the set of explicit constraints of η of Fig. 1 is $\{\tau_{0,1} \leq 3, \tau_{0,2} \geq 6, \tau_{0,2} \leq 6\}$, while $\{\tau_{0,1} \geq 0, \tau_{1,2} \geq 3, \tau_{1,2} \leq 6\}$ is the set of implicit constraints.

2.3 Operations on timed scenarios

(This subsection briefly recounts our recent work [14].)

If ξ is a scenario of length n with stable distance table $\mathcal{D}_s^\xi$, then, for any $0 \leq i < j < n$, the interval I_{ij}^ξ is $\{a \in \mathbb{Q} \mid m_{ij}^\xi \leq a \leq M_{ij}^\xi\}$, where $\mathcal{D}_s^\xi[i, j] = (m_{ij}^\xi, M_{ij}^\xi)$. Intuitively, I_{ij}^ξ corresponds to a pair of constraints: for every behaviour $\mathcal{B} \in [\![\xi]\!]$, $t_{ij}^\mathcal{B}$ must be at least m_{ij}^ξ and at most M_{ij}^ξ, i.e., $t_{ij}^\mathcal{B} \in I_{ij}^\xi$.

If ξ and η are two consistent scenarios with the same sequence of events, ξ is *subsumed* by η, denoted by $\xi \subseteq \eta$, when $[\![\xi]\!] \subseteq [\![\eta]\!]$. $\xi \subseteq \eta$ iff $\forall_{0 \leq i < j < n} I_{ij}^\xi \subseteq I_{ij}^\eta$.

[8] That is, a constraint cannot be removed without changing the semantics.

[9] Optimality was achieved under the assumption that a timed scenario cannot be split into two [13].

$0 : a$;
$1 : b$ $\{\tau_{0,1} \leq 5\}$;
$2 : c$ $\{\tau_{1,2} \geq 2\}$.

	1	2
0	(0, 5)	(2, ∞)
1		(2, ∞)

$0 : a$;
$1 : b$;
$2 : c$ $\{\tau_{0,2} =< 3\}$.

	1	2
0	(0, 3)	(0, 3)
1		(0, 3)

Fig. 4. ξ and its stable distance table, η and its stable distance table

$\xi \Cup \eta$

$0 : a$;
$1 : b$ $\{\tau_{0,1} \leq 5\}$;
$2 : c$.

	1	2
0	(0, 5)	(0, ∞)
1		(0, ∞)

ζ

$0 : a$;
$1 : b$;
$2 : c$ $\{\tau_{1,2} = 1, \tau_{0,2} = 4\}$.

	1	2
0	(3,3)	(4,4)
1		(1,1)

Fig. 5. The combination of ξ and η of Fig. 4, and scenario ζ with its stabilized table

If ξ and η are two consistent scenarios of length n with the same sequence of events, $\mathcal{E}$, such that $\forall_{0 \leq i < j < n}\ I_{ij}^{\xi} \cap I_{ij}^{\eta} \neq \emptyset$, then the *combination* of ξ and η, denoted by $\xi \Cup \eta$, is *defined*. In that case, $\xi \Cup \eta$ is a scenario whose sequence of events is $\mathcal{E}$ and whose constraints are given by $\mathcal{D}^{\xi \Cup \eta}$, where $\mathcal{D}^{\xi \Cup \eta}[i,j] = (\min(m_{ij}^{\xi}, m_{ij}^{\eta}), \max(M_{ij}^{\xi}, M_{ij}^{\eta}))$. The original table $\mathcal{D}^{\xi \Cup \eta}$ is stable: $\mathcal{D}^{\xi \Cup \eta} = \mathcal{D}_s^{\xi \Cup \eta}$.

If $\xi \Cup \eta$ is defined, then $[\![\xi]\!] \cup [\![\eta]\!] \subseteq [\![\xi \Cup \eta]\!]$. But, in general, $[\![\xi \Cup \eta]\!] \not\subseteq [\![\xi]\!] \cup [\![\eta]\!]$. This is because table $\mathcal{D}_s^{\xi \Cup \eta}$ allows all the behaviours in $[\![\xi]\!] \cup [\![\eta]\!]$, but there is a possibility that it may also allow some extra behaviours, namely those that satisfy all the constraints of the combination, but do not satisfy some of the constraints in ξ and some of the constraints in η. That is, $[\![\xi \Cup \eta]\!] = [\![\xi]\!] \cup [\![\eta]\!] \cup \mathcal{Z}(\xi, \eta)$, where $[\![\xi]\!] \cap \mathcal{Z}(\xi, \eta) = \emptyset$ and $[\![\eta]\!] \cap \mathcal{Z}(\xi, \eta) = \emptyset$. We call members of $\mathcal{Z}(\xi, \eta)$ *zigzagging* behaviours. As an example consider scenarios ξ and η of Fig. 4. Fig. 5 shows $\xi \Cup \eta$ along with its stable table. Scenario ζ of Fig. 5 represents a set of behaviours in which the time distance between events a and b is exactly 3, and between events a and c is exactly 4 units of time. There is no behaviour in the semantics of ζ that is allowed by either ξ or η of Fig. 4, yet $[\![\zeta]\!] \subset [\![\xi \Cup \eta]\!]$. That is, all behaviours in $[\![\zeta]\!]$ belong to $\mathcal{Z}(\xi, \eta)$. *This indicates that the union of the sets of behaviours allowed by ξ and η cannot be represented by a single scenario.*

There is a sufficient condition for the non-existence of zigzagging behaviours [14]. This condition is based on the form of explicit constraints of ξ and η and provides a basis for a procedure to determine whether $\mathcal{Z}(\xi, \eta)$ is empty [14]. Recently we have shown that the sufficient condition is also necessary [20].

If $\mathcal{Z}(\xi, \eta) = \emptyset$, the combination of ξ and η becomes their *union*, denoted by $\xi \cup \eta$. Fig. 6 shows two scenarios where the union of the sets of behaviours allowed by each *can be* represented by a single scenario, namely their union (see the scenario on the right).

In the rest of the paper when we say "the union of ξ and η exists", it means that $\xi \Cup \eta$ is defined and $\mathcal{Z}(\xi, \eta) = \emptyset$, so scenario $\xi \cup \eta$ captures the union of the sets of behaviours allowed by ξ, η or both: $[\![\xi \cup \eta]\!] = [\![\xi]\!] \cup [\![\eta]\!]$.

3 Synthesizing automata from sets of optimised scenarios

In this section we will use our previous results on optimising scenarios [13] to synthesize a timed automaton from a set Ξ of scenarios, in such a way that the automaton will have the smallest number of clocks in the entire class of equivalent automata. Optimality is obtained under the following simplifying assumption:

Assumption 1 *Each scenario in Ξ has a sequence of events that is different from that of any other scenario in Ξ.*

The assumption is not very limiting in practice. It is discussed further in Sec. 4.

We assume that each of the scenarios in Ξ describes a set of *complete* behaviours of a (sub)system, and that all such behaviours begin at the same initial state. Ξ_o is the set whose members are the optimised forms of scenarios in Ξ.

Obviously, the number of clocks in our constructed automaton should not exceed the largest number of clocks required by the optimised form of any one of the constituent scenarios, i.e., any member of Ξ_o. This is because one can always construct a trivial automaton by allowing the automata corresponding to individual optimised scenarios to share only their initial locations. So our secondary goal is to make the number of locations reasonably small, though not necessarily minimal. Next we define the problem formally.

> Let $\mathcal{TA}$ be the class of all timed automata and Ξ be a finite set of finite scenarios, such that if $\xi_1 = (\mathcal{E}_1, \mathcal{C}_1)$ and $\xi_2 = (\mathcal{E}_2, \mathcal{C}_2)$ are two different scenarios in Ξ, then $\mathcal{E}_1 \neq \mathcal{E}_2$. Let $\mathcal{TA}(\Xi) = \{\mathcal{A} \mid \mathcal{A} \in \mathcal{TA}$ and $L(\mathcal{A}) = \bigcup_{\xi \in \Xi} [\![\xi]\!]\}$. The goal is to synthesize an automaton $\mathcal{A} \in \mathcal{TA}(\Xi)$ in such a way that if $\mathcal{A}' \in \mathcal{TA}$ is language-equivalent to $\mathcal{A}$, then $\mathcal{A}'$ has no fewer clocks than $\mathcal{A}$.

We accomplish our objective by an algorithm that is both simple and efficient (see Sec. 3.2). The general idea is that if the sequences of events in two scenarios in Ξ have a common prefix, then *under certain circumstances* it is possible to combine the prefixes of the scenarios. We optimise each of the scenarios in Ξ and use them to build a tree, $\mathcal{T}_o^{\Xi}$, which can then be converted to the desired timed automaton.

As discussed in Sec. 2.3, sometimes two scenarios, ξ and η, can be combined into a single scenario, γ, that is, $\gamma = \xi \cup \eta$. Conversely, it might be possible that a scenario γ can be "split" into ξ and η. But then, if $\gamma \in \Xi_o$, it can be replaced by ξ and η in Ξ_o. In that case, for achieving optimality, we must first show that the number of clocks in $\mathcal{A}_\gamma$ will not be larger than those in $\mathcal{A}_\xi$ or $\mathcal{A}_\eta$. That is, "splitting" a scenario into two will not decrease the number of clocks.

Definition 1. *Let ξ be a scenario, and let $alloc_\xi$ be an optimal clock allocation for ξ. The* cost *of ξ is the number of clocks that is used in $alloc_\xi$.*

Definition 2. *Let Ξ be a set of scenarios. The* cost *of Ξ, denoted by $cost(\Xi)$, is the maximum of the costs of the members of Ξ.*

Observation 1 *Let $\xi = (\mathcal{E}, \mathcal{C}_1)$ and $\eta = (\mathcal{E}, \mathcal{C}_2)$ be two optimised scenarios. Let $\gamma = \xi \cup \eta$, and $\mathcal{C}$ be the set of explicit constraints of γ. Then $\mathcal{C} \subseteq \mathcal{C}_1 \cup \mathcal{C}_2$.*

Proof. Let $i < j$. At every interval $I_{ij}^{\xi \cup \eta}$ we have one of the following cases:

1. Both $\mathcal{D}_s^{\xi}[i,j] = (a_1, b_1)$ and $\mathcal{D}_s^{\eta}[i,j] = (a_2, b_2)$ correspond to explicit constraints. Then we must have one of the following cases:
   (a) If $a_1 > 0$, $a_2 > 0$, $b_1 < \infty$ and $b_2 < \infty$, then $\mathcal{C}$ will include two explicit constraints of the form $\tau_{i,j} \geq min(a_1, a_2)$ and $\tau_{i,j} \leq max(b_1, b_2)$.
   (b) If $a_1 = 0$ or $a_2 = 0$, and $b_1 < \infty$ and $b_2 < \infty$, then $\mathcal{C}$ will include one explicit constraint of the form $\tau_{i,j} \leq max(b_1, b_2)$.
   (c) If $a_1 > 0$ and $a_2 > 0$, and $b_1 = \infty$ or $b_2 = \infty$, then $\mathcal{C}$ will include one explicit constraint of the form $\tau_{i,j} \geq min(a_1, a_2)$.
   (d) If $a_1 = 0$ or $a_2 = 0$, and $b_1 = \infty$ or $b_2 = \infty$, then $\mathcal{C}$ will not include any explicit constraints between i and j.
2. $\mathcal{D}_s^{\xi}[i,j] = (a_1, b_1)$ corresponds to explicit constraint(s), but $\mathcal{D}_s^{\eta}[i,j] = (a_2, b_2)$ corresponds to default or implied constraints. Then $\mathcal{C}$ will have an explicit constraint of the form $\tau_{i,j} \geq a_1$ if $a_1 \leq a_2$, and an explicit constraint of the form $\tau_{i,j} \leq b_1$ if $b_2 \leq b_1$.
3. Both $\mathcal{D}_s^{\xi}[i,j]$ and $\mathcal{D}_s^{\eta}[i,j]$ correspond to default or implied constraints. Then $\mathcal{C}$ will not have any explicit constraints between i and j.

In all the cases above, an explicit constraint α between i and j will be in $\mathcal{C}$ only if at least one of $\mathcal{C}_1$ and $\mathcal{C}_2$ include α. □

An important consequence of Observation 1 is that an anchor in $\gamma = \xi \cup \eta$ must be an anchor in ξ or in η.

Theorem 1. *Let $\xi = (\mathcal{E}, \mathcal{C}_1)$ and $\eta = (\mathcal{E}, \mathcal{C}_2)$ be two optimised scenarios. If there exists an optimised scenario γ such that $\gamma = \xi \cup \eta$, then $cost(\{\gamma\}) \leq cost(\{\xi, \eta\})$.*

Proof. Recall that the number of clocks in an optimal clock allocation for a scenario is determined by the number of overlapping anchors of the scenario, which, in turn, is determined by the *explicit* constraints of the scenario.

Let $\mathcal{C}$ be the set of explicit constraints of γ. By Observation 1, $\mathcal{C}$ does not include anchors that are neither in ξ nor in η. We must show that *new* overlapping ranges are not introduced in γ: existing overlapping ranges of ξ or η that are present also in γ would not require additional clocks. Therefore there are two cases to consider:
(1) i_1 and i_2 are both anchors in ξ or in η or in both. But then their ranges do not overlap, so, by Observation 1, their ranges will not overlap in γ either.
(2) i_1 is an anchor in ξ but not in η and i_2 is an anchor in η but not in ξ. We must show that γ does not have overlapping ranges of i_1 and i_2.

Let α be an explicit constraint of the form $\tau_{i_1,j_1} \sim a$ in $\mathcal{C}_1$ and β be an explicit constraint of the form $\tau_{i_2,j_2} \sim b$ in $\mathcal{C}_2$, such that the integer ranges $[i_1, j_1)$ and $[i_2, j_2)$ overlap. Moreover, assume i_1 is not an anchor in η and i_2 is not an anchor in ξ. We will show that both α and β cannot exist in $\mathcal{C}$.

Proof by contradiction. Assume both α and β are in $\mathcal{C}$.

Without loss of generality assume α is of the form $\tau_{i_1,j_1} \leq a$ and β is of the form $\tau_{i_2,j_2} \leq b$, where $a, b \in \mathbb{Q}$[10].
(2a) $\alpha = \tau_{i_1,j_1} \leq a$ is both in $\mathcal{C}_1$ and in $\mathcal{C}$. By definition of union, $M^{\eta}_{i_1 j_1}$ cannot be larger than a, i.e., $M^{\eta}_{i_1 j_1} = c \leq a$. Since i_1 is not an anchor in η, $M^{\eta}_{i_1 j_1} = c$ must have been implied by some explicit constraints in η. This constraint could be implied in three cases [13]:

1. There are constraints $\tau_{i_1,j} \leq u$ and $\tau_{j,j_1} \leq v$ in $\mathcal{C}_2$[11] such that $i_1 < j < j_1$ and $c = u + v$. But i_1 is not an anchor in η, so this would be a contradiction.
2. There are constraints $\tau_{i_1,j} \leq u$ and $\tau_{j_1,j} \geq v$ in $\mathcal{C}_2$ such that $i_1 < j_1 < j$ and $u = c + v$. But i_1 is not an anchor in η, so this would be a contradiction.
3. There are constraints $\tau_{j,j_1} \leq u$ and $\tau_{j,i_1} \geq v$ in $\mathcal{C}_2$ such that $j < i_1 < j_1$ and $u = c + v$. So, for $M^{\eta}_{i_1 j_1} = c$ to be implied, there must exist $\alpha_1 = \tau_{j,j_1} \leq u$ and $\alpha_2 = \tau_{j,i_1} \geq v$.

(2b) $\beta = \tau_{i_2,j_2} \leq b$ is both in $\mathcal{C}_2$ and in $\mathcal{C}$. By definition of union, $M^{\xi}_{i_2 j_2}$ cannot be larger than b, i.e., $M^{\xi}_{i_2 j_2} = d \leq b$. Since i_2 is not an anchor in ξ, $M^{\xi}_{i_2 j_2} = d$ must have been implied by some other constraints in ξ. By an argument similar to the one in (2a) it can be shown that there must be some constraints $\beta_1 = \tau_{j',j_2} \leq u'$ and $\beta_2 = \tau_{j',i_2} \geq v'$ in $\mathcal{C}_2$ such that $j' < i_2 < j_2$ and $u' = d + v'$.

Now we consider two cases:

Case 1: $j \neq j'$, $\alpha_1 = \tau_{j,j_1} \leq u \in \mathcal{C}_1$ and $\beta_1 = \tau_{j',j_2} \leq u' \in \mathcal{C}_2$, where $j < i_1 < j_1$, $j' < i_2 < j_2$, and $i_1 \neq i_2$.

Case 2: $j = j'$, $\alpha_2 = \tau_{j,i_1} \geq v \in \mathcal{C}_1$ and $\beta_2 = \tau_{j',i_2} \geq v' \in \mathcal{C}_2$, where $j < i_1 < j_1$, $j' < i_2 < j_2$, and $i_1 \neq i_2$.

Both cases imply $\mathcal{Z}(\xi, \eta) \neq \emptyset$ [20], which is a contradiction: $\xi \cup \eta$ exists. □

Since combining two scenarios into one does not increase the number of clocks, splitting a scenario into two will not decrease the number of clocks.

3.1 Auxiliary definitions

Before presenting the algorithm, we will define a few concepts.

Definition 3. *Let $\xi = (e_0 e_1 \dots e_{n-1}, \mathcal{C})$ be a scenario. The* path for ξ*, denoted by p_ξ, is defined as follows:*

- *$\{n_0, n_1, \dots, n_n\}$ is the set of nodes of p_ξ. n_0 is the initial node.*
- *For each $0 \leq j < n$ there is a transition r_j from n_j to n_{j+1}, and r_j is labeled with e_j.*
- *For every i and j such that $0 \leq i < j < n$, transition r_j in p_ξ is annotated with the set of all the constraints of the form $\tau_{i,j} \sim a$ in $\mathcal{C}$.*

[10] If α is of the form $\tau_{i_1,j_1} \geq a$, and/or β is of the form $\tau_{i_2,j_2} \geq b$, the steps of the proof will be essentially the same.

[11] It is impossible for both $\tau_{i_1,j} \leq u$ and $M^{\eta}_{i_1 j_1} = c$ to be implied, without anchor i_1.

ξ

$0 : a$;
$1 : b$;
$2 : c\ \{\tau_{1,2} \leq 6\}$;
$3 : d\ \{\tau_{0,3} \geq 3\}$.

	1	2	3
0	$(0, \infty)$	$(0, \infty)$	$(3, \infty)$
1		$(0, 6)$	$(0, \infty)$
2			$(0, \infty)$

η

$0 : a$;
$1 : b\ \{\tau_{0,1} \leq 4\}$;
$2 : c\ \{\tau_{0,2} \leq 6\}$;
$3 : d$.

	1	2	3
0	$(0, 4)$	$(0, 6)$	$(0, \infty)$
1		$(0, 6)$	$(0, \infty)$
2			$(0, \infty)$

$\xi \cup \eta$

$0 : a$;
$1 : b$;
$2 : c\ \{\tau_{1,2} \leq 6\}$;
$3 : d$.

	1	2	3
0	$(0, \infty)$	$(0, \infty)$	$(0, \infty)$
1		$(0, 6)$	$(0, \infty)$
2			$(0, \infty)$

Fig. 6. Two scenarios and their union

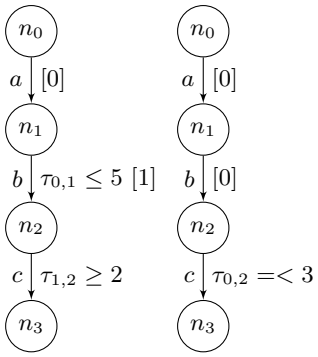

Fig. 7. p_ξ and p_η for ξ and η of Fig. 4

A path is just another representation of the corresponding scenario, one that is more convenient for our current purposes. We will often denote transition r_j in a path p by r_j^p, and refer to it as the *j-th transition* or *transition j*.

Definition 4. *Let p_ξ be the path for scenario ξ. If i is an anchor in ξ whose range is $[i, k)$, then for every $i \leq j < k$ transition r_j of p_ξ* belongs to the range of *i. We use $range(r_j)$ to denote the set of all anchors whose ranges include r_j.*

Fig. 7 shows the paths for scenarios of Fig. 4. In p_ξ, transition 0, i.e., the transition on event a is annotated with [0], which signifies that the transition belongs to the range of anchor 0. Observe that transition 1 does not belong to the range of anchor 0: the range of anchor 0 ends on transition 1. Transition 2 does not belong to the range of any anchor: the range of anchor 1 ends on transition 2.

In the remainder of this section we assume that $range(r)$ is known for every transition r of a path.

Definition 5. *Let ξ be a consistent scenario. Let $\alpha = \tau_{i,j} \sim a$, for some $i < j$, $a \in \mathbb{Q}$ and $\sim \in \{\leq, \geq\}$ be a constraint. We say α is* too restrictive for *i and j in ξ, if either $\sim$ is $\leq$ and $M_{ij}^\xi > a$, or $\sim$ is $\geq$ and $m_{ij}^\xi < a$.*

Intuitively, a constraint that is not too restrictive can be added to a scenario without changing its semantics. In our synthesis algorithm (see Sec. 3.2) we will take care to ensure that such additions will not increase the number of clocks.

Definition 6. *Let ξ and η be two scenarios and p_ξ and p_η be their paths. The j-th transition of p_ξ is* compatible *with the j-th transition of p_η, if every constraint on $\tau_{i,j}$ on the j-th transition of p_ξ is not too restrictive for i and j in η.*
The j-th transitions of p_ξ and p_η are compatible with each other*, if*

- *they agree on their events, and*
- *the j-th transition of p_ξ is compatible with the j-th transition of p_η, and vice versa, and*

– *transitions* $0, \ldots j-1$ *of* p_ξ *and* p_η *are compatible with each other.*

As an example consider the paths p_ξ and p_η of Fig. 7 (the corresponding scenarios with their stable tables are shown in Fig. 4). Transitions $r_0^{p_\xi}$ and $r_0^{p_\eta}$ are compatible with each other (because none of them have any constraints), and so are $r_1^{p_\xi}$ and $r_1^{p_\eta}$. Transition 1 of p_ξ is compatible with that of p_η, because the constraint $\tau_{0,1} \leq 5$ is not more restrictive than the existing upper bound on the time distance between events 0 and 1 in η, i.e., $M_{01}^\eta = 3 < 5$. Transition 1 of p_η is compatible with transition 1 of p_ξ, because the former does not have any constraint of the form $\tau_{i,1} \sim a$, for $i < 1$. Transitions $r_2^{p_\xi}$ and $r_2^{p_\eta}$ are not compatible with each other: constraint $\tau_{1,2} \geq 2$ of ξ is more restrictive than the existing lower bound on the time distance between events 1 and 2 in η, i.e., $m_{12}^\eta = 0 < 2$, so transition 2 of p_ξ is not compatible with that of p_η.

3.2 The synthesis algorithm

We are now ready to present the details of the algorithm.

– Let Ξ be a set of scenarios that satisfies Assumption 1, Ξ_o be the set of optimised forms of the scenarios in Ξ, and P_{Ξ_o} be the set of paths for scenarios in Ξ_o. Let $cost(\Xi_o) = m$.
– Initialize $\mathcal{T}_o^\Xi$ by merging the initial nodes of the paths in P_{Ξ_o}. Let n_0 be the resulting node, which is the root of the tree.
– Let $Groups = \{(n_0, 0, P_{\Xi_o})\}$.
– While $Groups \neq \emptyset$
  1. Let (n, i, G) be a member of $Groups$.
  2. Let $Partitions$ be the set of groups of paths obtained by partitioning G in such a way that
     (a) The i-th transitions of all the paths that belong to the same group g are compatible with each other, and
     (b) $|\bigcup_{p \in g} range(r_i^p)| \leq m$.
  3. For each group $g \in Partitions$:
     (a) Merge the i-th transitions of all the paths in g to obtain a new transition r'. The source of r' is n, the target is a new node n'.
     (b) The set of constraints on r' is the union of the sets of constraints on the i-th transitions of all paths in g.
     (c) Set $range(r')$ to $\bigcup_{p \in g} range(r_i^p)$.
     (d) Remove from g those paths for which transition i is the last one.
     (e) If $g \neq \emptyset$ then $Groups := Groups \cup \{(n', i+1, g)\}$.
  4. $Groups := Groups \setminus \{(n, i, G)\}$.

The algorithm obviously terminates: each iteration removes one group in step 4, but the total number of groups added in step 3e does not exceed $N \times |P_{\Xi_o}|$, where N is the length of the longest path. The invariant is

> For every $(n, k, g) \in Groups$ the following holds:
> for every $p, q \in g$ and every $0 \leq j < k$, transitions r_j^p and r_j^q are compatible with each other; moreover, $|\bigcup_{p \in g} range(r_j^p)| \leq m$.

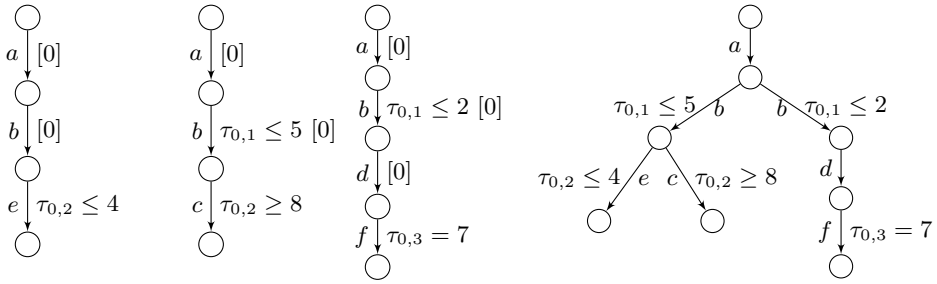

Fig. 8. Three scenarios and their synthesized tree

Notice that step 2 is under-specified. The aim, of course, is to keep the groups as large as possible, but there may be many ways to partition a group so that condition 2b can be satisfied. The choice made in one step can affect the sizes of the subtrees created in the subsequent steps, so this algorithm does not ensure that the number of locations in the resulting automaton will be minimal.

As an example consider $\xi_1 = (abe, \{\tau_{0,2} \leq 4\})$, $\xi_2 = (abc, \{\tau_{0,1} \leq 5, \tau_{0,2} \geq 8, \tau_{1,2} \geq 3\})$, and $\xi_3 = (abdf, \{\tau_{0,3} = 7, \tau_{1,3} \geq 5\})$. It is easy to see that ξ_1 requires one clock, while ξ_2 and ξ_3 require two clocks each. The optimised equivalent scenarios are $\xi_1^o = \xi_1$, $\xi_2^o = (abc, \{\tau_{0,1} \leq 5, \tau_{0,2} \geq 8\})$, and $\xi_3^o = (abdf, \{\tau_{0,1} \leq 2, \tau_{0,3} = 7\})$. Fig. 8 shows the paths for the optimised forms. Observe that after optimisation the required number of clocks remained 1 in ξ_1, but it decreased from 2 to 1 in both ξ_2 and ξ_3. So $cost(\{\xi_1^o, \xi_2^o, \xi_3^o\}) = 1$. One of the synthesized trees that can be obtained by our algorithm is shown on the right of the figure.

3.3 Obtaining the final automaton

The tree that is produced by the algorithm can be trivially converted to a timed automaton. The root is the initial location and all the leaves are the final locations. The anchors must be replaced by clocks in such a way that two anchors with overlapping ranges are not assigned the same clock. Once clocks are allocated, the constraints must be rewritten in terms of the clocks. Our existing clock allocation algorithms [21, 22] can be used for this purpose after some trivial modifications.

In the tree of Fig. 8 there is only one anchor, so the resulting automaton will require only one clock: a clock, e.g., c_0 would be reset on the transition labeled with event a (i.e., $c_0 := 0$ would be added to the transition), and every occurrence of $\tau_{0,j}$, for $1 \leq j \leq 3$, would be replaced with c_0.

It is clear from the construction that the language accepted by the resulting automaton is the union of the behaviours allowed by the scenarios in Ξ. It is also clear that the number of clocks will not exceed $cost(\Xi_o)$.

Theorem 2. *Let Ξ be a finite set of finite scenarios that satisfies Assumption 1, and let $\mathcal{A}_\Xi$ be the timed automaton synthesized by the algorithm of Sec. 3.2. Then*

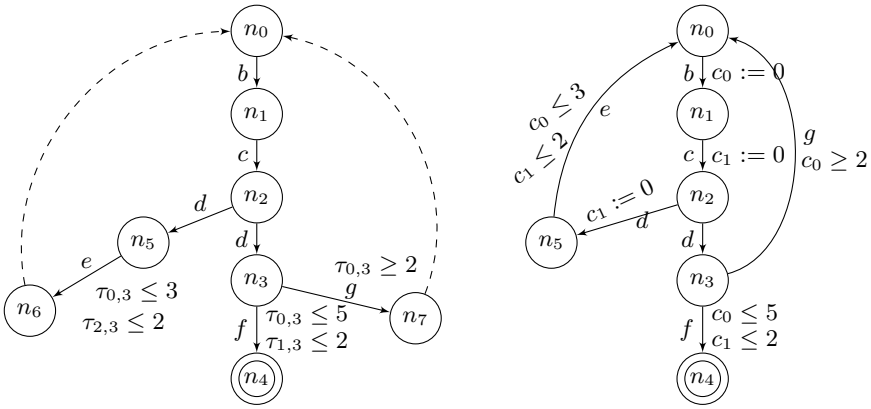

Fig. 9. A tree and its corresponding timed automaton

$\mathcal{A}_\Xi$ has the smallest number of clocks in the entire class of language-equivalent automata.

Proof. The algorithm begins with optimising each scenario in Ξ and replacing it with its optimised form, hence obtaining $\Xi_o = \{\xi_1, \ldots \xi_n\}$. Let $cost(\Xi_o) = m$.

By Theorem 1, for each $\xi_i \in \Xi_o$, there are no η and γ such that $\xi_i = \eta \cup \gamma$ and $\mathcal{A}_{\xi_i}$ requires more clocks than the automaton synthesized from η and γ. That is, $cost(\Xi_o)$ cannot be lower than m.

Consider an automaton $\mathcal{A}$ obtained by unifying the initial locations of all $\mathcal{A}_{\xi_i}$. Clearly, $L(\mathcal{A}) = \bigcup_{\xi_i \in \Xi_o} L(\mathcal{A}_{\xi_i}) = \bigcup_{\xi_i \in \Xi_o} [\![\xi_i]\!]$. Since each $\mathcal{A}_{\xi_i}$ has the optimal number of clocks in the entire class of language-equivalent automata, the result can be extended to $\mathcal{A}$.

The synthesis algorithm essentially merges the common prefixes of the paths in $\mathcal{A}$ while making sure that the the overall cost will not rise above m. □

3.4 Handling a limited form of cycles

Ξ might include scenarios that specify behaviours that all begin and end in the same state of the system being specified. Such scenarios are used to specify repetition of some "partial" behaviours in the system [22]. When such scenarios are used for synthesizing a timed automaton, the constructed automaton must include cycles in order to allow an arbitrary number of repetitions of the behaviours specified by these scenarios. But the underlying graph of the automaton generated by our algorithm takes the form of a tree, and as such cannot support specifying repeated behaviours.

However, with some minimal effort, our trees can be extended so that certain kinds of cycles can be handled. In particular, those that specify repeated behaviours that begin at the initial state of the system can be handled easily.

While constructing $\mathcal{T}_o^\Xi$, we can mark a leaf, say n, as a "looping leaf". Then the root will be the "looping ancestor" of n and the path between the root and n

will be a "looping path". The looping path in $\mathcal{T}_o^{\Xi}$ represents a sequence of events that starts at the root and can be repeated in the final automaton. That is, the looping leaf and the looping ancestor will be "unified", i.e., correspond to the same location in the final automaton, and hence there will be a loop.

Fig. 9 shows a tree obtained by our algorithm from the optimised set of scenarios $\{\xi_1, \xi_2, \xi_3\}$, where $\xi_1 = (bcde, \{\tau_{0,3} \leq 3, \tau_{2,3} \leq 2\})$, $\xi_2 = (bcdf, \{\tau_{0,3} \leq 5, \tau_{1,3} \leq 2\})$, and $\xi_3 = (bcdg, \{\tau_{0,3} \geq 2\})$. The tree includes two looping paths: one that begins at n_0 and ends at n_6, and one that begins at n_0 and ends at n_7. The correspondence between n_0 and n_6 and between n_0 and n_7 is shown with dashed lines. In the final automaton n_0, n_6 and n_7 will all correspond to the same location: see the automaton on the right of the figure.

Using this method we obtain automata whose initial locations are the only locations with more than one incoming transition.

It is worth mentioning that during the construction of the tree in Fig. 9 the three transitions with event d could not have been merged, as that would violate condition 2b of the algorithm: $cost(\{\xi_1, \xi_2, \xi_3\}) = 2$, but the merge would result in a transition that would belong to the ranges of all three anchors, i.e., 0, 1 and 2, and would thus increase the number of clocks in the synthesized automaton to three. So the three paths had to be partitioned into two groups: p_{ξ_1} could not have merged with p_{ξ_2}, but p_{ξ_3} could form a group either with p_{ξ_1} or with p_{ξ_2}. The automaton shown in the figure is the result of placing p_{ξ_3} and p_{ξ_2} in the same group. The other alternative (not shown here) would result in an automaton with the same number of clocks and locations.

4 Revisiting the assumption

We will now discuss some consequences of relaxing Assumption 1 by allowing scenarios with the same sequences of events.

Let Ξ_o include two scenarios $\xi = (\mathcal{E}, \mathcal{C}^{\xi})$ and $\eta = (\mathcal{E}, \mathcal{C}^{\eta})$ that can be combined into one scenario. That is, $\mathcal{Z}(\xi, \eta) = \emptyset$ (Sec. 2.3). We investigate the question of whether it would be advantageous to replace them with $\xi \cup \eta$ in Ξ_o.

We discuss several cases using some illustrative examples.

One scenario is subsumed by the other one. Assume $\xi \subseteq \eta$, then $\xi \cup \eta = \eta$ [14]. Obviously in this case ξ can simply be removed from Ξ_o.

As an example consider the two scenarios of Fig. 10. The automaton synthesized from the two by our synthesis algorithm of Sec. 3.2 is shown on the right of Fig. 10. Observe that this automaton requires two clocks, one each for anchors 0 and 1. A simple comparison of the stable distance tables of ξ and η shows that $\xi \subseteq \eta$, therefore $\xi \cup \eta = \eta$. That is, $\mathcal{A}_\eta$, the automaton corresponding to η, would capture the semantics of both ξ and η. So $\mathcal{A}_\eta$ (shown in the middle of Fig. 10) would be equivalent to the automaton obtained by the synthesis algorithm. But $\mathcal{A}_\eta$ would require only one clock, moreover, it would have only four locations.

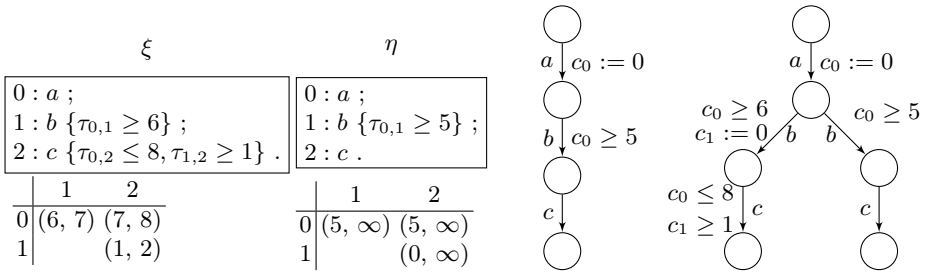

Fig. 10. $\xi \subseteq \eta$, $\mathcal{A}_{\xi\cup\eta}$ and the automaton synthesized from ξ and η

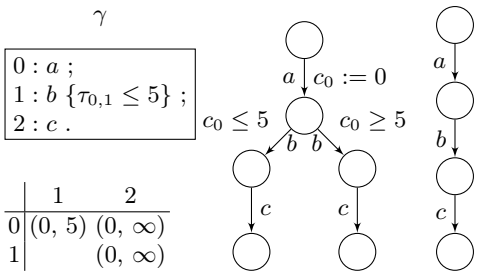

Fig. 11. η and γ with complementary constraints, the automaton synthesized from η and γ, and $\mathcal{A}_{\eta\cup\gamma}$

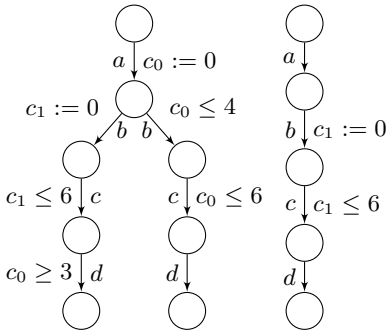

Fig. 12. The automaton synthesized from ξ and η of Fig. 6, and $\mathcal{A}_{\xi\cup\eta}$

Two scenarios with a pair of complementary constraints. Let ξ and η contain a pair of explicit constraints that are complementary, i.e., for some i, j and a we have $\tau_{i,j} \leq a \in \mathcal{C}^\xi$ and $\tau_{i,j} \geq a \in \mathcal{C}^\eta$. Then the only constraints on $\tau_{i,j}$ in $\xi \cup \eta$ will be implicit default constraints (i.e., $\tau_{i,j} \geq 0$ and $\tau_{i,j} \leq \infty$). If anchor i is not used in any other constraint, then it will no longer be needed, since the default constraints are not explicitly mentioned in scenarios. If anchor i is used in some other constraints, the range of i might become shorter. So in either case this may cause a reduction of $cost(\Xi)$.

For example, the scenario η of Fig. 10 and γ of Fig. 11 have a pair of complementary constraints ($\tau_{0,1} \geq 5$ and $\tau_{0,1} \leq 5$), moreover, $\mathcal{Z}(\eta, \gamma) = \emptyset$[12]. So $\eta \cup \gamma$ exists. The automaton synthesized from the two by the synthesis algorithm of Sec. 3.2, shown in the middle of Fig. 11, requires one clock. The automaton corresponding to $\eta \cup \gamma$, i.e., $\mathcal{A}_{\eta\cup\gamma}$, shown on the right of Fig. 11, would be equivalent to this. But it does not require any clock, and has fewer locations than the synthesized automaton.

[12] If the explicit constraints of two scenarios differ only on constraints between one pair of events, their combination cannot include zigzagging behaviours [14].

ξ

$0 : a$;
$1 : b$;
$2 : c\ \{\tau_{1,2} \leq 8\}$;
$3 : d\ \{\tau_{0,3} \leq 10\}$.

	1	2	3
0	(0, 10)	(0, 10)	(0, 10)
1		(0, 8)	(0, 10)
2			(0, 10)

η

$0 : a$;
$1 : b$;
$2 : c\ \{\tau_{1,2} \geq 8\}$;
$3 : d\ \{\tau_{0,3} \leq 10\}$.

	1	2	3
0	(0, 2)	(8, 10)	(8, 10)
1		(8, 10)	(8, 10)
2			(0, 2)

γ

$0 : a$;
$1 : b$;
$2 : c\ \{\tau_{1,2} \geq 2\}$;
$3 : d$.

	1	2	3
0	(0, ∞)	(2, ∞)	(2, ∞)
1		(2, ∞)	(2, ∞)
2			(0, ∞)

Fig. 13. Three scenarios with the same sequence of events

Two arbitrary scenarios that can be combined. ξ and η might be just some arbitrary scenarios with the same sequence of events, that can be combined into one. Fig. 6 shows one such example. Neither of the scenarios ξ and η is a subset of the other one (see the stable distance tables), the two scenarios do not contain a pair of complementary constraints, yet their union exists. The automaton corresponding to their union, shown on the right of Fig. 12, requires only one clock, while the automaton synthesized from $\{\xi, \eta\}$, shown on the left of the figure, requires two clocks. Also note that $\mathcal{A}_{\xi \cup \eta}$ has fewer locations than the synthesized one.

These examples suggest that if the initial set of optimised scenarios includes some scenarios with identical sequences of events, we should try to combine them whenever possible, before applying the synthesis algorithm. This would be advantageous, as it always results in a smaller number of locations, and sometimes in a smaller number of clocks in the final synthesized automaton. But doing so is not very easy. The difficulty becomes apparent when one tries to answer the following question.

If the initial set of scenarios has more than two scenarios with the same sequence of events, some of which can be combined, would the order in which the union is applied affect the final outcome?

Let us consider the following example. Let Ξ_o include four scenarios ξ, η, γ and θ such that both $\xi \cup \eta$ and $\gamma \cup \theta$ exist. Moreover, $\gamma \subseteq \xi \cup \eta$, that is, $(\xi \cup \eta) \cup \gamma = \xi \cup \eta$, but $\gamma \cup \theta \not\subseteq \xi \cup \eta$. So there are two possibilities: replacing the four scenarios with $\xi \cup \eta$ and θ in Ξ_o, or replacing them with $\xi \cup \eta$ and $\gamma \cup \theta$. While the first alternative may seem intuitively better, it might not be the optimal choice: as discussed above, combining γ and θ may make some of the explicit constraints in γ or θ disappear. This could result in a decrease in the overall cost, if $cost(\theta) > cost(\xi \cup \eta)$.

As a concrete example consider the three scenarios of Fig. 13, which all have the same sequences of events. Assume $\Xi_o = \{\xi, \eta, \gamma\}$ (the scenarios are already in their optimised form) and the task is to synthesize an automaton from Ξ_o.

Observe that ξ and η have a pair of complementary constraints ($\tau_{1,2} \leq 8$ and $\tau_{1,2} \geq 8$), and that $\eta \subseteq \gamma$. There are two options for combining the scenarios of

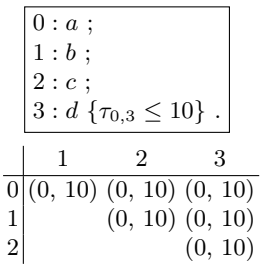

Fig. 14. $\xi \cup \eta$

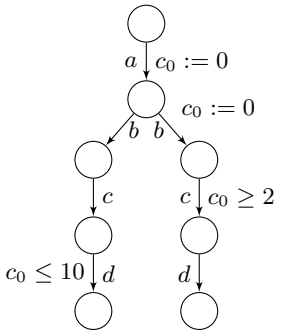

Fig. 15. $\Xi_o = \{\xi \cup \eta, \gamma\}, \mathcal{A}_{\Xi_o}$

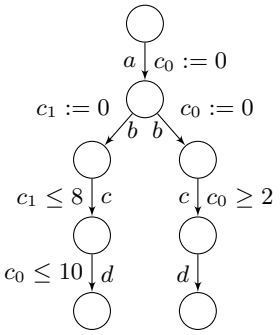

Fig. 16. $\Xi_o = \{\xi, \eta \cup \gamma\}, \mathcal{A}_{\Xi_o}$

Ξ_o before the synthesis. The first option is to combine ξ and η ($\xi \cup \eta$ is shown in Fig. 14), in which case Ξ_o will be updated to $\{\xi \cup \eta, \gamma\}$. Fig. 15 shows the resulting synthesized automaton.

The second option is to combine η and γ, in which case Ξ_o will be updated to $\{\xi, \gamma\}$, because $\eta \cup \gamma = \gamma$. Fig. 16 shows the automaton synthesized from Ξ_o in this case.

Observe that the first automaton requires only one clock, whereas the second one has two clocks. This example shows that if Assumption 1 is not satisfied, then obtaining the optimal result depends on the order in which scenarios are combined.

Choosing the right combination is a separate optimisation problem that may require significant computational resources to solve. Detailed discussion of this problem is outside the scope of the current paper.

5 Conclusions

We study the problem of synthesizing a timed automaton from a number of optimised timed scenarios. We present a simple and efficient algorithm[13] that, given a finite set of finite scenarios, such that each of them requires at most m clocks, constructs an automaton whose number of clocks is at most m. Moreover, the automaton has a reasonably small (though not necessarily minimal) number of locations. We show that, given a simplifying assumption about the initial set of scenarios, our synthesized automaton has the minimal number of clocks in the entire class of language-equivalent timed automata.

We then discuss the opportunities and difficulties that would arise out of relaxing our simplifying assumption, and argue that without the assumption achieving strict optimality might have a significant computational complexity.

[13] An implementation in C++ can be made available upon request.

References

1. A. Salah, R. Mizouni, R. Dssouli, B. Parreaux, Formal composition of distributed scenarios, in: D. de Frutos-Escrig, M. Núñez (Eds.), Formal Techniques for Networked and Distributed Systems – FORTE 2004, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 213–228.
2. J. Greenyer, Scenario-based modeling and programming of distributed systems, in: M. Köhler-Bussmeier, E. Kindler, H. Rölke (Eds.), Proceedings of the International Workshop on Petri Nets and Software Engineering 2021 co-located with the 42nd International Conference on Application and Theory of Petri Nets and Concurrency (PETRI NETS 2021), Paris, France, June 25th, 2021 (due to COVID-19: virtual conference), Vol. 2907 of CEUR Workshop Proceedings, CEUR-WS.org, 2021, pp. 241–252.
3. S. Somé, R. Dssouli, J. Vaucher, From Scenarios to Timed Automata: Building Specifications from Users Requirements, in: Proceedings of the Second Asia Pacific Software Engineering Conference, APSEC '95, IEEE Computer Society, pp. 48–57.
4. P. Chandrasekaran, M. Mukund, Matching scenarios with timing constraints, in: E. Asarin, P. Bouyer (Eds.), Formal Modeling and Analysis of Timed Systems, Springer, Berlin, Heidelberg, 2006, pp. 98–112.
5. D. Harel, H. Kugler, A. Pnueli, Synthesis Revisited: Generating Statechart Models from Scenario-Based Requirements, Springer Berlin Heidelberg, 2005, pp. 309–324.
6. S. Uchitel, J. Kramer, J. Magee, Synthesis of Behavioral Models from Scenarios, IEEE Trans. Softw. Eng. 29 (2003) 99–115.
7. S. Akshay, M. Mukund, K. N. Kumar, Checking Coverage for Infinite Collections of Timed Scenarios, in: CONCUR 2007 - Concurrency Theory, 18th International Conference, CONCUR 2007, Proceedings, pp. 181–196.
8. B. Bollig, J. Katoen, C. Kern, M. Leucker, Replaying Play In and Play Out: Synthesis of Design Models from Scenarios by Learning, in: Proceedings of the 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS, 2007, pp. 435–450.
9. R. Alur, M. Martin, M. Raghothaman, C. Stergiou, S. Tripakis, A. Udupa, Synthesizing Finite-State Protocols from Scenarios and Requirements, in: E. Yahav (Ed.), Hardware and Software: Verification and Testing, Springer International Publishing, Cham, 2014, pp. 75–91.
10. N. Saeedloei, F. Kluźniak, From Scenarios to Timed Automata, in: Formal Methods: Foundations and Applications - 20th Brazilian Symposium, SBMF 2017, Proceedings, pp. 33–51.
11. N. Saeedloei, F. Kluźniak, Timed Scenarios: Consistency, Equivalence and Optimization, in: Formal Methods: Foundations and Applications - 21st Brazilian Symposium, SBMF 2018, Proceedings, pp. 215–233.
12. R. Alur, P. Madhusudan, Decision Problems for Timed Automata: A Survey, in: Formal Methods for the Design of Real-Time Systems, International School on Formal Methods for the Design of Computer, Communication and Software Systems, SFM-RT, Revised Lectures, 2004, pp. 1–24.
13. N. Saeedloei, F. Kluźniak, Minimization of the number of clocks for timed scenarios, in: S. Campos, M. Minea (Eds.), Formal Methods: Foundations and Applications - 24th Brazilian Symposium, SBMF 2021 Proceedings, Vol. 13130 of Lecture Notes in Computer Science, Springer, 2021, pp. 122–139.
14. N. Saeedloei, F. Kluźniak, Operations on timed scenarios, in: M. Huisman, A. Ravara (Eds.), Formal Techniques for Distributed Objects, Components, and

Systems - 43rd IFIP WG 6.1 International Conference, FORTE 2023, Held as Part of the 18th International Federated Conference on Distributed Computing Techniques, DisCoTec 2023, Lisbon, Portugal, June 19-23, 2023, Proceedings, Vol. 13910 of Lecture Notes in Computer Science, Springer, 2023, pp. 97–114. doi:10.1007/978-3-031-35355-0_7.
URL https://doi.org/10.1007/978-3-031-35355-0_7

15. R. Alur, D. L. Dill, A theory of timed automata, Theor. Comput. Sci. 126 (2) (1994) 183–235.
16. P. A. Abdulla, J. Deneux, J. Ouaknine, J. Worrell, Decidability and complexity results for timed automata via channel machines, in: L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, M. Yung (Eds.), Automata, Languages and Programming, Springer, Berlin, Heidelberg, 2005, pp. 1089–1101.
17. C. Baier, N. Bertrand, P. Bouyer, T. Brihaye, When are timed automata determinizable?, in: S. Albers, A. Marchetti-Spaccamela, Y. Matias, S. Nikoletseas, W. Thomas (Eds.), Automata, Languages and Programming, Springer, Berlin, Heidelberg, 2009, pp. 43–54.
18. N. Saeedloei, F. Kluźniak, Optimization of timed scenarios, in: G. Carvalho, V. Stolz (Eds.), Formal Methods: Foundations and Applications - 23rd Brazilian Symposium, SBMF 2020, Ouro Preto, Brazil, November 25-27, 2020, Proceedings, Vol. 12475 of Lecture Notes in Computer Science, Springer, 2020, pp. 119–136.
19. D. L. Dill, Timing assumptions and verification of finite-state concurrent systems, in: Proceedings of the International Workshop on Automatic Verification Methods for Finite State Systems, Springer-Verlag New York, Inc., 1990, pp. 197–212.
20. N. Saeedloei, F. Kluźniak, Observations about Timed Scenarios, https://tigerweb.towson.edu/nsaeedloei/Observations.pdf.
21. N. Saeedloei, F. Kluźniak, Clock allocation in timed automata and graph colouring, in: Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (part of CPS Week), HSCC 2018, pp. 71–80. doi:10.1145/3178126.3178138.
22. N. Saeedloei, F. Kluźniak, Synthesizing clock-efficient timed automata, in: B. Dongol, E. Troubitsyna (Eds.), Integrated Formal Methods - 16th International Conference, IFM 2020, Lugano, Switzerland, November 16-20, 2020, Proceedings, Vol. 12546 of Lecture Notes in Computer Science, Springer, 2020, pp. 276–294.