Timed Scenarios: Consistency, Equivalence and Optimization

Neda Saeedloei[1] and Feliks Kluźniak[2]

[1] Southern Illinois University, Carbondale
neda@cs.siu.edu,
[2] Logic Blox
feliks.kluzniak@logicblox.com

Abstract. We develop a new method for determining the consistency of timed scenarios. If the scenario is consistent, we obtain a canonical representation for the entire class of equivalent scenarios. This allows us to optimise a scenario according to various criteria. In particular, we are able to minimize the largest constant in the scenario's set of constraints: this technique is directly relevant to decreasing the costs of verification for timed automata synthesized from timed scenarios.

1 Introduction

Using scenarios for specification and implementation of complex systems, including real time systems, has been an active area of research for over three decades [20, 15, 22, 8]. Synthesis of formal models of systems from scenarios has also been studied in the past [20, 13, 15, 8, 22], and recently there has been renewed interest in this area [4, 16, 19].

We have recently proposed [19] a form of timed scenarios (called Timed Event Sequences or TES) for specifying partial behaviours of real-time systems. We also developed a synthesis method for constructing a timed automaton from a set of TES. What was not addressed in that work was the question of the consistency of timed scenarios.

We set out to develop, from first principles, a method for detecting whether a scenario is consistent. As a byproduct we obtained a canonical representation for the entire class of scenarios that are equivalent to the given one. This in turn allowed us to *optimise* scenarios (according to various criteria), by replacing a given scenario with an equivalent one that has more desirable properties. The current paper summarizes the results of this study.

Specifically, the main contributions of the paper are as follows:

1. We present a generalized and simplified notion of timed scenarios and their semantics. The new notion is independent of modes and mode graphs [19].
2. We propose a method for determining the consistency of timed scenarios. The method is developed from the fundamental equations and inequations that hold, in general, between the times at which ordered events occur. As a byproduct we obtain a canonical representation (a "distance table") for the entire class of scenarios that are equivalent to the given one.

Scenario ξ	Scenario η
$L_0:\ a;$ $L_1:\ b\ \{L_0 \leq 1\};$ $L_2:\ c\ \{L_1 \leq 5\};$ $L_3:\ d\ \{L_1 \leq 4, L_2 \geq 2\};$ $\quad e\ \{L_1 \leq 11, L_2 \geq 4, L_3 \leq 4\}.$	$L_0:\ a;$ $L_1:\ b\ \{L_0 \leq 1\};$ $L_2:\ c\ \{L_1 \leq 2\};$ $L_3:\ d\ \{L_1 \leq 4, L_2 \geq 2, L_2 \leq 4\};$ $\quad e\ \{L_2 \geq 4, L_3 \leq 4\}.$

Fig. 1: Two equivalent scenarios

3. We use the distance table to *optimise* scenarios according to various criteria. In particular, given a time distance table corresponding to a scenario we show how to minimize the maximum constant in the scenario. For example, our prototype tool can convert scenario ξ of Fig. 1 to the equivalent scenario η. The maximum constant in η is smaller than that in ξ, so η is better suited for the purpose of synthesizing a timed automaton: the cost of verifying a timed automaton crucially depends on the size of the maximum constant in its time constraints.[1]
4. It turns out that our distance tables are essentially isomorphic to the Difference Bounds Matrices studied by Dill in the context of verification of timed automata [11]. We show how to apply Dill's method to the case of scenarios, both to check consistency and to minimise the maximum constant.

2 Concepts

2.1 Events

Let Σ be a finite set of symbols called *events*. Let Σ^* denote the set of all sequences (finite or infinite) formed from elements of Σ. The subset of Σ^* that contains only all the sequences of length n will be denoted by Σ^n.

The intended interpretation is that $e \in \Sigma$ is the name of a concrete event in the real world, such as "a button is pressed".

Given a sequence $\sigma = e_0 e_1 e_2 \ldots \in \Sigma^*$ we will use the term "event i of σ" (or "the i-th event", etc.) to denote the i-th element of σ, i.e., the i-th occurrence of an event in the sequence. This should not be confused with e_i, which is a symbol in Σ, and which may have many occurrences in σ.

[1] Most model-checking tools for timed automata (e.g., UPPAAL [7] and KRONOS [9]) use region-based and zone-based abtraction methods in order to make verification possible in spite of the infinite state spaces of timed automata. It is well-known that both of these abstraction methods depend on the number of clocks and on the size of the constants that appear in constraints. In fact, *the size of the region graph is exponential in the number of clocks and the (encoding of) constants* [3].

2.2 Behaviours

Definition 1. *A* behaviour[2] *over Σ is a sequence $(e_0, t_0)(e_1, t_1)(e_2, t_2)\dots$, such that $e_i \in \Sigma$, $t_i \in \mathbb{R}^{\geq 0}$, $t_0 = 0$ and $t_{i-1} \leq t_i$ for $i \in \{1, 2\dots\}$.*

We will omit the phrase "over Σ" when doing so does not lead to confusion.

A behaviour can be infinite or finite, even empty. In this paper we will discuss mostly finite behaviours.

The intended interpretation of (e_i, t_i) is that the i-th occurrence of an event is an occurrence of event e_i, and takes place t_i time units after the initial occurrence of an event (namely, e_0).

A behaviour can be thought of as a pattern for an infinite number of concrete behaviours that differ only in their starting time. We say that the (abstract) behaviour *represents* all those concrete behaviours.

Given a behaviour $\mathcal{B} = (e_0, t_0)(e_1, t_1)(e_2, t_2)\dots$ we will use $eseq(\mathcal{B})$ to denote the sequence $e_0e_1e_2\dots$ and $tseq(\mathcal{B})$ to denote $t_0t_1t_2\dots$.

We often say "event i of $\mathcal{B}$" (or "the i-th event") to denote event i of $eseq(\mathcal{B})$.

Definition 2. *Let $\mathcal{B} = (e_0, t_0)(e_1, t_1)\dots(e_{n-1}, t_{n-1})$ be a behaviour of length n. Then, for any $0 \leq i < j < n$, the symbol $t^{\mathcal{B}}_{ij}$ denotes the* distance*, in time units, of event j from event i in $\mathcal{B}$. That is, $t^{\mathcal{B}}_{ij} = t_j - t_i$.*

We often write simply t_{ij} when this does not lead to ambiguity.

Observation 1 *For any behaviour of length n, and for $0 \leq i < j < k < n$:*

$$t_{ij} + t_{jk} = t_{ik} \tag{1}$$

Proof. Obvious: $t_{ij} + t_{jk} = t_j - t_i + t_k - t_j = t_k - t_i = t_{ik}$. □

We are often interested not in a particular behaviour, but in a set of behaviours that satisfy certain time constraints, e.g., that the door will open sufficiently quickly after the button is pressed. To describe sets of behaviours that satisfy such constraints we use timed scenarios.

2.3 Timed scenarios

Definition 3. *Given a natural number n, let $\Phi(n)$ denote the set of* constraints *of the form $d \sim c$, where $\sim \in \{\leq, \geq, =\}$[3] and c is a constant in the set of rational numbers, $\mathbb{Q}$. d is the symbol $\tau_{i,j}$, for some integers $0 \leq i < j < n$.*

The intended interpretation is that $\tau_{i,j}$ is the time distance between events i and j in the behaviours described by a timed scenario. This will become clear in Definition 4 and Definition 5.

[2] Behaviours are essentially the "timed words" of Alur [2].

[3] To keep the presentation compact, we do not allow sharp inequalities: allowing them would complicate our definitions and proofs, without affecting the general principles. Notice that sharp inequalities are of mainly theoretical interest: in practice we can measure time only with some finite granularity γ, so $x < c$ is for all practical purposes equivalent to $x \leq c - \gamma$.

Definition 4. *Let n be a natural number and Σ the set of events. A* timed scenario *of length 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 (i.e., $\mathcal{E} \in \Sigma^n$);*
- *$\mathcal{C} \subset \Phi(n)$ is a finite set of constraints.*

Given a scenario $\xi = (\mathcal{E}, \mathcal{C})$, we will use *events*$(\xi)$ to denote $\mathcal{E}$ and *constraints*(ξ) to denote $\mathcal{C}$.

In this paper the term "scenario" will always refer to a timed scenario. We will omit the phrase "over Σ" when that does not lead to confusion.

We will use the term "event i of ξ" to denote event i in *events*(ξ).

External representation To make scenarios fit for human consumption, we will usually describe them in a notation that is not unlike a simple programming language. A scenario will be written as a sequence of events, separated by semicolons and terminated by a period. If the scenario contains a constraint such as $\tau_{i,j} \leq c$, then event i in the sequence will be labelled by a unique symbol L_i, and event j will be annotated with a set of constraints that contains $L_i \leq c$.

In Fig. 2, ξ_2 is a representation of $(abcf, \{\tau_{0,1} \geq 2, \tau_{1,2} \geq 2, \tau_{0,3} \leq 2\})$.

2.4 Scenarios and behaviours

Definition 5. *Let ξ be a scenario of length n over Σ.*
A behaviour $\mathcal{B} = (e_0, t_0)(e_1, t_1) \dots (e_{n-1}, t_{n-1})$ over Σ is supported *by ξ iff*

- *events$(\xi) = e_0 \dots e_{n-1}$ and*
- *every $\tau_{i,j} \sim c$ in constraints(ξ) evaluates to true after $\tau_{i,j}$ is replaced by $t^{\mathcal{B}}_{ij}$.*

For a given scenario ξ, we use *Supp*(ξ) to denote the set of behaviours that are supported by ξ. (We will often say simply: "the set of behaviours of ξ".)[4]

The set of behaviours of scenario ξ_1 of Fig. 2 is

Supp$(\xi_1) = \{(a, t_0)(b, t_1)(c, t_2)(b, t_3) \mid t_0 = 0 \wedge t_3 \geq t_2 \geq t_1 \geq t_0 \wedge t_1 - t_0 \leq 5 \wedge t_2 - t_0 \leq 4\}$.

Observation 2 *Let ξ be a scenario of length n and let ξ' be the scenario obtained by adding some constraint to constraints(ξ). Then Supp$(\xi') \subset$ Supp(ξ).*

Proof. A behaviour of ξ' must satisfy all the constraints in ξ. □

Definition 6. *The* semantics *of scenario ξ, denoted by $[\![\xi]\!]$, is the set of behaviours that are supported by ξ, i.e., $[\![\xi]\!] =$ Supp(ξ).*

Definition 7. *Two scenarios ξ and η are* equivalent *iff $[\![\xi]\!] = [\![\eta]\!]$.*

Definition 8. *Let $\xi = (\mathcal{E}, \mathcal{C})$ and $\eta = (\mathcal{E}, \mathcal{C}')$ be two scenarios (with the same sequence of events). $\mathcal{C}$ and $\mathcal{C}'$ are* equivalent *iff ξ and η are equivalent.*

[4] There is nothing new in the intuition that a scenario describes a set of behaviours: see, e.g., the paper by Somé et al. [21].

ξ_1
L_0 : a;
b $\{L_0 \leq 5\}$;
c $\{L_0 \leq 4\}$;
b.

ξ_2
L_0 : a;
L_1 : b $\{L_0 \geq 2\}$;
c $\{L_1 \geq 2\}$;
f $\{L_0 \leq 2\}$.

Fig. 2: Two scenarios

L_0 : a;
L_1 : b $\{L_0 \leq 2\}$;
L_2 : c $\{L_0 \leq 4, L_1 \geq 2\}$;
d $\{L_1 \geq 3, L_2 \leq 4\}$;
e $\{L_0 \leq 6\}$.

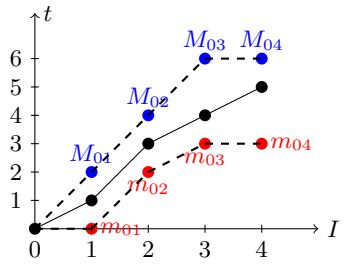

Fig. 3: A scenario and one of its behaviours

Definition 9. *A scenario ξ is* consistent *iff $[\![\xi]\!] \neq \emptyset$. A scenario is* inconsistent *iff it is not consistent.*

For instance, scenario ξ_1 of Fig. 2 is consistent, while ξ_2 is inconsistent, as the constraint that annotates event 3 (f) cannot be satisfied.

Definition 10. *For a consistent scenario ξ of length n, and for $0 \leq i < j < n$, we define*

$$m_{ij}^{\xi} = min\{t_{ij}^{\mathcal{B}} \mid \mathcal{B} \in Supp(\xi)\}$$

$$M_{ij}^{\xi} = max\{t_{ij}^{\mathcal{B}} \mid \mathcal{B} \in Supp(\xi)\}$$

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

We will often write just m_{ij} and M_{ij} when ξ is understood.

Observation 3 *For any behaviour in $Supp(\xi)$,*

$$0 \leq m_{ij} \leq t_{ij} \leq M_{ij} \leq \infty \tag{2}$$

Proof. A direct consequence of Definition 10. □

Observation 4 *Let ξ be a scenario of length n, and let $\mathcal{B}$ be a behaviour such that $eseq(\mathcal{B}) = events(\xi)$. If $m_{ij}^{\xi} \leq t_{ij}^{\mathcal{B}} \leq M_{ij}^{\xi}$ for every $0 \leq i < j < n$, then $\mathcal{B} \in Supp(\xi)$.*

Proof. $\mathcal{B}$ obviously satisfies all the constraints of ξ. □

In other words, the set of values from Definition 10 completely characterizes the semantics of a scenario.

Observation 5 *Let ξ and η be two scenarios of length n, such that $events(\xi) = events(\eta)$. Then $[\![\xi]\!] = [\![\eta]\!]$ iff $\forall_{0 \leq i < j < n}(m_{ij}^{\xi} = m_{ij}^{\eta} \wedge M_{ij}^{\xi} = M_{ij}^{\eta})$.*

Proof. A direct consequence of Observations 3 and 4. □

A behaviour of ξ can be viewed as a sequence of discrete timed events in a Cartesian plane, where the x-axis represents event numbers and the y-axis represents time. It might be useful to visualize this as a curve that is obtained by connecting all such timed events. For instance, Fig. 3 shows a consistent scenario along with one of its behaviours (shown with a solid line), namely $(a,0)(b,1)(c,3)(d,4)(e,5)$.

The upper and lower curves (in dashed lines) correspond to m_{0j} and M_{0j}, for $0 < j < 5$. Notice that the constraints that annotate the first four events of the scenario seem to indicate that the maximum distance between events 0 and 3 should be 8, but the constraint associated with event 4 reduces it to 6: the curves must obviously be monotonically non-decreasing.

These two curves can be viewed as the boundaries of the set of all behaviours of the scenario, that is, the plot of every supported behaviour must fit between these two curves.

This is necessary, but not sufficient. For example, the curve of the behaviour $(a,0)(b,2)(c,4)(d,4)(e,6)$ would fit between these boundaries, but the constraint $L_1 \geq 3$ that annotates event 3 (corresponding to $\tau_{1,3} \geq 3$) would be violated.

As noted above, in order to fully characterize the set of all behaviours of a scenario ξ of length n we must determine the minimum and maximum time distances between *every* pair of events in any member of $\mathit{Supp}(\xi)$. That is, we are interested in computing m^{ξ}_{ij} and M^{ξ}_{ij} for every i and j such that $0 \leq i < j < n$.

As an example consider scenario ξ_1 of Fig. 2 once more. The time distance between events 0 and 1 (a and the first b) is constrained to be no greater than 5, event 2 (c) occurs after event 1, but its distance from event 0 is at most 4. Surely, the time distance between events 0 and 1 must be at most 4 in all the behaviours of ξ_1: the constraint $L_0 \leq 5$ on event 1 is not tight. The tightest constraint that can replace it without changing the semantics of ξ_1 is $L_0 \leq 4$. $\mathit{Supp}(\xi_1)$ does, indeed, include a behaviour for which the time distance between events 1 and 2 is exactly 4.

An immediate question is how to determine the various values of m_{ij} and M_{ij} for a given scenario. We begin by elucidating some fundamental relationships between these values.

Observation 6 *Let ξ be a consistent scenario of length n. Then the following inequations hold, for any $0 \leq i < j < k < n$:*

$$m_{ij} + m_{jk} \leq m_{ik} \quad (3) \qquad m_{ij} + M_{jk} \geq m_{ik} \quad (4) \qquad m_{ij} + M_{jk} \leq M_{ik} \quad (5)$$
$$M_{ij} + M_{jk} \geq M_{ik} \quad (6) \qquad M_{ij} + m_{jk} \geq m_{ik} \quad (7) \qquad M_{ij} + m_{jk} \leq M_{ik} \quad (8)$$

Proof. Inequations (3) and (6) are direct consequences of equation (1).

For (4), assume $m_{ij} + M_{jk} < m_{ik}$. But then m_{ij} or m_{ik} is not tight enough, or M_{jk} is too tight. That is, $\mathit{Supp}(\xi)$ cannot include a behaviour for which $t_{ij} = m_{ij}$, because then $t_{jk} > M_{jk}$ would have to hold for equation (1) and $m_{ik} \leq t_{ik}$ to be satisfied.

For (5), assume $m_{ij} + M_{jk} > M_{ik}$. But then $\mathit{Supp}(\xi)$ cannot include a behaviour for which $t_{jk} = M_{jk}$, because $t_{ij} < m_{ij}$ would have to hold for equation (1) and $t_{ik} \leq M_{ik}$ to be satisfied.

The proofs for (7) and (8) are analogous to those for (4) and (5). □

$L_0:$ a; b; c $\{L_0 \geq 2\}$.

ξ_3

$L_0:$ a; $L_1:$ b $\{L_0 \leq 2\}$; c $\{L_1 \leq 3, L_0 \leq 4\}$.

ξ_4

Fig. 4: Two scenarios

	01	12	02
Min	2	2	4
Max	3	4	7

Fig. 5: The bounds on three events

The inequations of Observation 6 can be presented in compact form:

$$m_{ij} + m_{jk} \leq m_{ik} \leq \left\{ \begin{matrix} m_{ij} + M_{jk} \\ M_{ij} + m_{jk} \end{matrix} \right\} \leq M_{ik} \leq M_{ij} + M_{jk} \tag{9}$$

It is worth emphasizing that none of the inequations of Observation 6 can be replaced by equations. For instance, consider scenario ξ_3 of Fig. 4. It is easy to see that $m_{01} = 0$ and $m_{12} = 0$, but $m_{02} = 2$. That is, within $Supp(\xi_3)$ there are behaviours for which $t_{01} = m_{01}$ and behaviours for which $t_{12} = m_{12}$, but there is no behaviour for which $t_{01} = m_{01}$ *and* $t_{12} = m_{12}$. Similarly, in scenario ξ_4 of Fig. 4, $M_{01} = 2$, $M_{12} = 3$, but $M_{03} = 4$.

Inequation (9) can be used to reason about the behaviours of a scenario. For example, consider a scenario ξ of length 3, such that $events(\xi) = aba$, where the minimum and maximum values of t_{ij}, for $0 \leq i < j < 3$, in the behaviours of ξ are summarized in Fig. 5. If we limit our attention to those of the supported behaviours whose time annotations contain only integers, we find a set of six:

$\{(a,0)(b,2)(a,4),\ (a,0)(b,2)(a,5),\ (a,0)(b,2)(a,6),$
$(a,0)(b,3)(a,5),\ (a,0)(b,3)(a,6),\ (a,0)(b,3)(a,7)\}$

Assume the minimum time distance between 0 and 2, i.e., m_{02}, is increased to 6. The inequation $m_{02} \leq M_{01} + m_{12}$ no longer holds. A way to repair it[5] is to increase m_{12} to 3. As a result, our set will change to:

$\{(a,0)(b,2)(a,6),\ (a,0)(b,3)(a,6),\ (a,0)(b,3)(a,7)\}$

If we increased m_{02} to 7 (instead of 6), m_{01} and m_{12} would change to 3 and 4, respectively. The set would then include only one behaviour: $\{(a,0)(b,3)(a,7)\}$.

We generally have to deal with scenarios that have more than three events, so a tabular representation similar to that of Fig. 5 will be even more useful.

2.5 Distance tables

Definition 11. *Let $\xi = (\mathcal{E}, \mathcal{C})$ be a scenario of length n.*
$\mathcal{C}$ is pruned *iff, for any given integers i and j such that $0 \leq i < j < n$,*
- *$\mathcal{C}$ does not contain a constraint of the form $\tau_{i,j} = c$;*
- *$\mathcal{C}$ contains at most one constraint of the form $\tau_{i,j} \geq c$ and at most one constraint of the form $\tau_{i,j} \leq c$.*

If $constraints(\xi)$ is pruned, then we also say that ξ is pruned.

Obviously, given a set of constraints $\mathcal{C}$ it is easy to convert it to a set that is equivalent, but pruned. First, replace every constraint of the form $\tau_{i,j} = c$ with two constraints, $\tau_{i,j} \geq c$ and $\tau_{i,j} \leq c$. Second, for each $0 \leq i < j < n$,

[5] The other way is to increase M_{01} to 4, but that would introduce new behaviours.

	1	2	3	4
0	(0, 1)	(0, ∞)	(0, ∞)	(0, ∞)
1		(0, 5)	(0, 4)	(0, 11)
2			(2, ∞)	(4, ∞)
3				(0, 4)

Fig. 6: A distance table for ξ of Fig. 1

	1	2	3	4
0	(0, 1)	(0, 3)	(2, 5)	(4, 9)
1		(0, 2)	(2, 4)	(4, 8)
2			(2, 4)	(4, 8)
3				(0, 4)

Fig. 7: A stable version of the same table

- if $\mathcal{C}$ contains more than one constraint of the form $\tau_{i,j} \geq c$, retain only one with the maximal constant;
- if $\mathcal{C}$ contains more than one constraint of the form $\tau_{i,j} \leq c$, retain only one with the minimal constant.

Definition 12. *Let $\xi = (\mathcal{E}, \mathcal{C})$ be a pruned scenario of length n.*
A distance table *for ξ is a triangular matrix $\mathcal{D}^\xi$, such that:*

- *$\mathcal{D}^\xi_{ij}$ is defined iff $0 \leq i < j < n$;*
- *for $0 \leq i < j < n$, $\mathcal{D}^\xi_{ij} = (l_{ij}, h_{ij})$, where*
  - *l_{ij} and h_{ij} are rational numbers;*
  - *if $\mathcal{C}$ contains a constraint $\tau_{i,j} \geq c$ then $l_{ij} = c$, otherwise $l_{ij} = 0$;*
  - *if $\mathcal{C}$ contains a constraint $\tau_{i,j} \leq c$ then $h_{ij} = c$, otherwise $h_{ij} = \infty$.*

We will sometimes refer to an l_{ij} as a *low value*, and to an h_{ij} as a *high value*. If ξ is of length n, then we will say that $\mathcal{D}^\xi$ is *of size n*.

Obviously, given $\mathcal{D}^\xi$ we can construct a set of constraints that is equivalent to *constraints*(ξ). So the distance table for ξ is just another representation for the constraints of ξ.

Fig. 6 shows a distance table corresponding to scenario ξ of Fig. 1.

Definition 13. *A distance table of size n is* valid *iff $l_{ij} \leq h_{ij}$, for all $0 \leq i < j < n$. A table that is not valid is* invalid.

Observation 7 *If $\mathcal{D}^\xi$ is invalid, then ξ is inconsistent.*

Proof. Obvious from Definition 12. □

Definition 14. *A distance table of size n is* stable *iff it is valid and, for all $0 \leq i < j < k < n$,*

$$l_{ij} + l_{jk} \leq l_{ik} \leq \left\{ \begin{matrix} l_{ij} + h_{jk} \\ h_{ij} + l_{jk} \end{matrix} \right\} \leq h_{ik} \leq h_{ij} + h_{jk} \qquad (9')$$

The distance table of Fig. 6 is not stable. Fig. 7 shows its stable version.
The distance table for a scenario with no constraints is obviously stable.

Theorem 1. *Let $\mathcal{D}^\xi$ be a stable distance table. Then ξ is consistent.*

Proof. Let $events(\xi) = e_0 e_1 \dots e_{n-1}$. It is enough to show that there exists a behaviour $\mathcal{B} = (e_0, t_0)(e_1, t_1) \dots (e_{n-1}, t_{n-1})$, such that $\mathcal{B} \in Supp(\xi)$.

That is, we must show a sequence $t_0 t_1 \dots t_{n-1}$ such that, for $0 \leq j < k < n$, $l_{jk} \leq t_k - t_j \leq h_{jk}$, i.e., t_{jk} satisfies the appropriate constraint in the table.

Let $t_0 = 0$, and for $0 < j < n$ let $t_j = l_{0j}$. Then the constraints in the first row of the table are satisfied: $t_k - t_0 = l_{0k} \leq h_{0k}$.

Let $0 < j < k < n$. The table is stable, so $l_{0j} + l_{jk} \leq l_{0k}$, hence $l_{jk} \leq l_{0k} - l_{0j} = t_k - t_j$. Moreover, $l_{0k} \leq l_{0j} + h_{jk}$, hence $l_{0k} - l_{0j} \leq h_{jk}$. □

Definition 15. *Let $\mathcal{D}$ be a stable distance table, let p and q be integers such that $0 \leq p < q < n$, and let $S = t_p t_{p+1} \dots t_q$ be a sequence of real numbers. We say that S is* compatible *with $\mathcal{D}$ iff*

- $0 \leq t_p \leq t_{p+1} \leq \dots \leq t_q$;
- $l_{ij} \leq t_{ij} \leq h_{ij}$ *for any two integers i and j such that $p \leq i < j \leq q$.*[6]

Of course, if $\mathcal{B} \in Supp(\xi)$, then $tseq(\mathcal{B})$ is compatible with $\mathcal{D}^\xi$. And vice versa, a compatible sequence S whose length is the size of $\mathcal{D}^\xi$ satisfies all the constraints of ξ, so there is a $\mathcal{B} \in Supp(\xi)$ such that $S = tseq(\mathcal{B})$.

Lemma 1 *Let $\mathcal{D}$ be a stable distance table of size n, let b and c be integers such that $0 \leq b < c < n$, and let $t_b t_{b+1} \dots t_c$ be compatible with $\mathcal{D}$. Then*

1. *if $0 \neq b$ then the sequence can be extended to the left in such a way that the extended sequence is compatible with $\mathcal{D}$;*
2. *if $c \neq n - 1$ then the sequence can be so extended to the right.*

Proof. We consider here only case 1; case 2 is similar.

Let $a = b - 1$. We must show that there exists a real number t_a, such that $0 \leq t_a \leq t_b$ and $l_{aj} \leq t_{aj} \leq h_{aj}$ for $a < j \leq c$.

For $j = b$, we must have

$$l_{ab} \leq t_{ab} \leq h_{ab} \tag{10}$$

Since $0 \leq l_{ab} \leq h_{ab}$, it is possible to find a t_{ab} that satisfies (10).

For any $j > b$, we must have $l_{aj} \leq t_{aj} \leq h_{aj}$. This is equivalent to $l_{aj} \leq t_{ab} + t_{bj} \leq h_{aj}$, and therefore to

$$l_{aj} - t_{bj} \leq t_{ab} \leq h_{aj} - t_{bj} \tag{11}$$

Obviously, $l_{aj} - t_{bj} \leq h_{aj} - t_{bj}$, because $l_{aj} \leq h_{aj}$. Moreover, $0 \leq h_{aj} - t_{bj}$. This is because, from $(9')$, $l_{ab} + h_{bj} \leq h_{aj}$, hence $l_{ab} \leq h_{aj} - h_{bj}$. But $h_{aj} - h_{bj} \leq h_{aj} - t_{bj}$ (because $t_{bj} \leq h_{bj}$), so $l_{ab} \leq h_{aj} - t_{bj}$, and of course $0 \leq l_{ab}$.

And so, for any particular $j > b$, it is possible to find a t_{ab} that satisfies (11).

We must now show that a single t_{ab} can satisfy *all* these constraints simultaneously, i.e., first, that the following inequations hold for any j such that $b < j$:

$$l_{ab} \leq h_{aj} - t_{bj} \tag{12}$$

$$l_{aj} - t_{bj} \leq h_{ab} \tag{13}$$

[6] As elsewhere, $t_{ij} = t_j - t_i$.

Second, for $b < j_0 < j_1 \leq c$, we must have

$$l_{aj_0} - t_{bj_0} \leq h_{aj_1} - t_{bj_1} \tag{14}$$

$$l_{aj_1} - t_{bj_1} \leq h_{aj_0} - t_{bj_0} \tag{15}$$

If these inequations are satisfied, then the maximum of the low bounds on t_{ab} does not exceed the minimum of the high bounds, therefore it is possible to choose a satisfactory t_{ab}, and hence t_a.

From (9′) we have $l_{ab} + h_{bj} \leq h_{aj}$, hence $l_{ab} \leq h_{aj} - h_{bj}$. But $t_{bj} \leq h_{bj}$, so $h_{aj} - h_{bj} \leq h_{aj} - t_{bj}$, and therefore (12) holds.

From (9′) we have $l_{aj} \leq h_{ab} + l_{bj}$, hence $l_{aj} - l_{bj} \leq h_{ab}$. But $l_{aj} - t_{bj} \leq l_{aj} - l_{bj}$, because $l_{bj} \leq t_{bj}$, so (13) holds.

From (9′), $l_{aj_0} + h_{j_0j_1} \leq h_{aj_1}$, so $l_{aj_0} \leq h_{aj_1} - h_{j_0j_1}$. But $h_{aj_1} - h_{j_0j_1} \leq h_{aj_1} - t_{j_0j_1}$, hence $l_{aj_0} \leq h_{aj_1} - t_{j_0j_1}$, hence $l_{aj_0} - t_{bj_0} \leq h_{aj_1} - t_{bj_0} - t_{j_0j_1}$. But $t_{bj_0} + t_{j_0j_1} = t_{bj_1}$, so (14) holds.

From (9′), $l_{aj_1} \leq h_{aj_0} + l_{j_\circ j_1}$, so $l_{aj_1} \leq h_{aj_0} + t_{j_\circ j_1}$, i.e., $l_{aj_1} - t_{j_\circ j_1} \leq h_{aj_0}$. Hence $l_{aj_1} - t_{bj_0} - t_{j_\circ j_1} \leq h_{aj_0} - t_{bj_0}$, i.e., $l_{aj_1} - t_{bj_1} \leq h_{aj_0} - t_{bj_0}$: (15) holds. □

Theorem 2. *Let $\mathcal{D}^\xi$ be a stable distance table of size n. Then each constraint in the table is tight, i.e., for any two integers i and j such that $0 \leq i < j < n$, there exist behaviours $\mathcal{B}_\mathcal{L}, \mathcal{B}_\mathcal{H} \in Supp(\xi)$ such that $t_{ij}^{\mathcal{B}_\mathcal{L}} = l_{ij}$ and $t_{ij}^{\mathcal{B}_\mathcal{H}} = h_{ij}$.*[7]

Proof. We consider here only the case of $\mathcal{B}_\mathcal{L}$; for $\mathcal{B}_\mathcal{H}$ the reasoning is similar. Let $t_i = l_{0i}$, and for $i < m \leq j$ let $t_m = t_i + l_{im}$. So $t_{ij} = l_{ij}$. From elementary reasoning (very similar to that in the proof of Theorem 1) we know that $t_i t_{i+1} \ldots t_j$ is compatible with $\mathcal{D}^\xi$.

Lemma 1 shows that we can repeatably extend the sequence to the left and/or right, while maintaining compatibility with $\mathcal{D}^\xi$ as an invariant. The result will be a sequence of length n, and we can use that as $tseq(\mathcal{B}_\mathcal{L})$.[8] □

Theorem 3. *Let $\mathcal{D}^\xi$ be a stable distance table of size n. Then, for any $0 \leq i < j < n$, $\mathcal{D}^\xi_{ij} = (m^\xi_{ij}, M^\xi_{ij})$.*

Proof. By Definition 12 ξ has explicit constraints that require supported behaviours to satisfy $l_{ij} \leq t_{ij} \leq h_{ij}$, for all $0 \leq i < j < n$. Moreover, from Theorem 2 we know that each constraint is tight. □

From Theorem 3 and Observation 5 we immediately see that if we could find an effective way of computing a stable distance table equivalent to the constraints of any consistent scenario ξ, then we would have an effective way of checking whether any other scenario η is equivalent to ξ: they would be equivalent if and only if the stable distance table computed from ξ were identical to that computed from η. A stable distance table could then be treated as a convenient canonical representation of all the equivalent scenarios.

An effective method of computing a stable distance table equivalent to the constraints of a given scenario does in fact exist, and is presented below (Sec. 3.1 and Sec. 3.2).

[7] If $h_{ij} = \infty$ then $t_{ij}^{\mathcal{B}_\mathcal{H}}$ can be an arbitrary number not smaller than l_{ij}.

[8] We are only proving the existence of such a behaviour. A method of actually constructing it is discussed in Sec. 3.3.

3 Algorithms

3.1 Stabilising a distance table

How can we stabilise a distance table without changing the semantics of the associated scenario? If we relax any of the existing constraints, then we are in dire danger of supporting new behaviours. So, if the table is not stable, we must find a way of restoring the validity of (9′) by *increasing* some low values and/or *decreasing* some high values. We must make sure that the modified values are not changed more than is strictly necessary, as we do not want to introduce new constraints that are not implied by the existing ones.

Inspection of inequation (9′) shows that, if it does not hold, it can be restored by applying one or more of six rules (we assume that $0 \leq i < j < k < n$):

$$l_{ij} + l_{jk} > l_{ik} \longrightarrow l_{ik} := l_{ij} + l_{jk} \tag{R1}$$

$$l_{ik} > l_{ij} + h_{jk} \longrightarrow l_{ij} := l_{ik} - h_{jk} \tag{R2}$$

$$l_{ik} > h_{ij} + l_{jk} \longrightarrow l_{jk} := l_{ik} - h_{ij} \tag{R3}$$

$$l_{ij} + h_{jk} > h_{ik} \longrightarrow h_{jk} := h_{ik} - l_{ij} \tag{R4}$$

$$h_{ij} + l_{jk} > h_{ik} \longrightarrow h_{ij} := h_{ik} - l_{jk} \tag{R5}$$

$$h_{ik} > h_{ij} + h_{jk} \longrightarrow h_{ik} := h_{ij} + h_{jk} \tag{R6}$$

For example, if it is not the case that $l_{ij} + h_{jk} \leq h_{ik}$, then the right way to fix it is to decrease the value of h_{jk}, but only enough to make $l_{ij} + h_{jk} = h_{ik}$: this is the function of rule (R4).

Of course, application of a rule may lead to another violation of (9′): if we decrease h_{jk}, then $l_{ik} \leq l_{ij} + h_{jk}$ may cease to be true. So the rules must be applied over and over again, until either the table becomes invalid, or no rule is applicable (i.e., the table satisfies (9′) for all values of i, j and k). One of these things must eventually happen, because each application of a rule strictly decreases the difference between a high value and the corresponding low value, and if this difference becomes negative, the table becomes invalid.[9]

Notice that if the various values in the table were integer to begin with, then application of any of the rules keeps them integer. Notice also that if a rule assigns a new high value, then that value is finite.

In our prototype the algorithm is implemented along the following lines:

```
for k := n − 1 downto 2:
 for i := k − 2 downto 0:
   for j := k − 1 downto i + 1:
     while there is an applicable rule R for i, j and k:
       apply R;
```

[9] The values in the table are rational numbers, but they have a least common denominator. Adding or subtracting two such values cannot produce a result that does not share that common denominator. So our algorithm cannot decrease the difference between two such values indefinitely without making it negative.

if the table is invalid, stop.

It turns out that this triple loop does the job: the table becomes either invalid or stable. The cost of stabilisation is thus of the order of $O(n^3)$. This is not very surprising, since there are clear similarities with the Floyd-Warshall algorithm for computing distances in a graph.

Observation 8 *(Confluence.) Let $\mathcal{D}$ be a distance table, and let a valid $\mathcal{D}'$ be the result of applying the stabilisation procedure outlined above to $\mathcal{D}$. Then $\mathcal{D}'$ is determined by $\mathcal{D}$ uniquely, i.e., regardless of the particular order in which the rules R1–R6 are applied.*

Proof. We give an informal outline of a proof.

Consider applying the rules iteratively to a particular instance of inequation $(9')$ (i.e., for some particular choice of i, j and k). There are six rules, and six values that should satisfy six inequalities. Each unsatisfied inequality enables one rule, which modifies one value, and that value can be modified only by that rule. More than one rule can be applicable at the same time, e.g., (R1) and (R6) ($l_{ij} + l_{jk} > l_{ik}$ and $h_{ik} > h_{ij} + h_{jk}$ can hold simultaneously). However, simple inspection shows that it is impossible for an applicable rule to affect either the value set by another applicable rule or its condition of applicability.[10] So the order in which rules are applied for this instance of $(9')$ cannot affect the outcome.

Consider a particular low value in the table (the argument for a high value is similar). This value may appear in different guises in different instances of $(9')$: in some of them as l_{ij}, in others as l_{jk} or l_{ik}. Regardless of the order in which the instances are treated, the low value will grow only as much as is needed to satisfy all of them. □

Theorem 4. *(Equivalence.) Let $\mathcal{D}^\xi$ be a distance table, and let $\mathcal{D}$ be the result of applying the stabilisation procedure to $\mathcal{D}^\xi$. Then $\mathcal{D}$ is equivalent to $\mathcal{D}^\xi$.*

Proof. Let η be a scenario such that $events(\eta) = events(\xi)$ and $\mathcal{D}^\eta = \mathcal{D}$.

Observation 8 shows that the result of stabilisation is unique. Each low (high) value is increased (decreased) only by as much as is needed to make the table stable. So, for each $i < j$, $l_{ij} \le m^\xi_{ij}$ and $h_{ij} \ge M^\xi_{ij}$ (because the various m^ξ and M^ξ must satisfy (9)). From this and from Observation 4 we have $Supp(\eta) \supset Supp(\xi)$. But the process of stabilisation did not relax any of the constraints, so $Supp(\eta) \subset Supp(\xi)$. □

3.2 Checking consistency, computing a stable table

We are now ready to present our method of checking the consistency of a given scenario ξ of length n.

[10] For example, (R5) depends on h_{ik} and l_{jk}, which can be changed by (R6) and (R3). But if (R5) is applicable, i.e., if $h_{ij} + l_{jk} > h_{ik}$ holds, then (R6) is not applicable: we cannot have $h_{ik} > h_{ij} + h_{jk}$, because $l_{jk} \le h_{jk}$ (the table is valid); similarly, (R3) is not applicable: $l_{ik} > h_{ij} + l_{jk}$ would imply $l_{ik} > h_{ik}$.

1. We begin by pruning ξ. Its constraints are then arranged in a sequence $\Psi = \psi_0\psi_1 \ldots \psi_K$.
2. We then iteratively compute a finite sequence of scenarios, $\eta^0, \eta^1, \ldots \eta^k$, such that:
   (a) $k \leq K$;
   (b) $events(\eta^i) = events(\xi)$ (for $0 < i \leq k$);
   (c) $[\![\eta^0]\!] \supset [\![\eta^1]\!] \supset \ldots \supset [\![\eta^k]\!] \supset [\![\xi]\!]$;
   (d) η^0 is a scenario with no constraints and $[\![\eta^k]\!] = [\![\xi]\!]$;
   (e) each η^{i+1} is obtained by adding ψ_i to η^i.
   This is done by iteratively modifying a single table, $\mathcal{D}$. After iteration i the contents of $\mathcal{D}$ represents $\mathcal{D}^{\eta^i}$.
   Each new constraint is added by amending $\mathcal{D}$. If the constraint is not tighter than the one that is already present in the table, then adding it is an empty action. (This is essentially a stronger version of pruning, performed "on the fly". In practice there is no need for the initial pruning in step 1.)
3. After all the constraints in Ψ have been added to the table, $\mathcal{D}$ is stabilised to make explicit those constraints that are only implicitly implied by the constraints in the table (cf. Theorem 4).

The cost of this algorithm is dominated by that of stabilisation, i.e., $O(n^3)$.[11]

If step 3 ended with an invalid $\mathcal{D}^{\eta^k}$, then by Observation 7 η^k is inconsistent, i.e., $[\![\eta^k]\!] = \emptyset$. But $[\![\eta^k]\!] \supset [\![\xi]\!]$, so ξ is inconsistent. If, however, the resulting table is valid, then:

- $[\![\eta^k]\!] = [\![\xi]\!]$, since $[\![\eta^k]\!] \supset [\![\xi]\!]$, and all the constraints of ξ are accounted for in η^k;
- by Theorem 1, η^k (and therefore ξ) is consistent;
- by Theorem 3 for each $0 \leq i < j < n$, $\mathcal{D}^{\eta^k}_{ij} = (m^{\eta^k}_{ij}, M^{\eta^k}_{ij})$, and by Observation 5 this is equal to $(m^{\xi}_{ij}, M^{\xi}_{ij})$, since $[\![\eta^k]\!] = [\![\xi]\!]$.

We will use $\mathcal{D}^{\xi}_s$ to denote the stable table obtained from ξ. Fig. 7 shows $\mathcal{D}^{\xi}_s$ for scenario ξ of Fig. 1.

Theorem 5. *Let ξ and η be two consistent scenarios, such that $events(\xi) = events(\eta)$. Then $[\![\xi]\!] = [\![\eta]\!]$ iff $\mathcal{D}^{\xi}_s = \mathcal{D}^{\eta}_s$.*

Proof. A direct consequence of Theorem 3 and Observation 5.

3.3 Using a distance table to find particular behaviours

A stable distance table $\mathcal{D}$ equivalent to $\mathcal{D}^{\xi}$ is useful for constructing particular behaviours of scenario ξ. If we are interested in a behaviour that has a particular

[11] In practice it is often convenient to pinpoint the first "offending" constraint that makes a scenario inconsistent. This is easily done by making sure that the order of constraints in Ψ corresponds to the textual order of constraints in the scenario, and attempting to stabilise the table each time a new constraint is added to it. If the number of constraints is proportional to n, the cost becomes $O(n^4)$.

value t_{ij} for some $i < j$, we must, of course, consult $\mathcal{D}$ to ensure that $l_{ij} \leq t_{ij} \leq h_{ij}$. Once we have chosen t_{ij}, we tighten the constraint by assigning this value to both l_{ij} and h_{ij}, then restabilise the table. This will give us a new set of constraints that will guide us in choosing the value of t_{ij} for some other i and j. We continue to do so, until the complete behaviour is known, i.e., $l_{ij} = h_{ij}$ for $0 \leq i < j < n$.

More generally, we can use a distance table $\mathcal{D}$ to quickly verify whether it is possible for a behaviour to simultaneously satisfy several constraints that are tighter than the ones in the table. We just make those constraints tighter in $\mathcal{D}$, and see whether an attempt to restabilise produces a valid table. In this case our stabilisation method becomes a "poor man's constraint solver".

4 Optimizing Scenarios

Now that we have an effective method of establishing the equivalence of scenarios, it is tempting to convert a scenario to an equivalent one that is "better". For instance, one might want to decrease the number of constraints, or to make them detect an unsupported behaviour as early as possible.

A particularly interesting possibility is that of finding a scenario that is equivalent to the given one, but that has smaller constants in its constraints. As explained in Sec. 1, this has direct consequences for the cost of verifying timed automata that are synthesised from scenarios.

The optimisation is carried out by building the constraints of the new scenario from the distance table in such a way that the constraints with larger constants need not be added if they are implied by constraints with smaller constants.

More specifically, given a scenario ξ of length n we proceed as follows:

1. We use the method of Sec. 3.2 to produce a stable distance table $\mathcal{D}_s^\xi$.[12]
2. We copy all the information from $\mathcal{D}_s^\xi$ into a list L of items with two forms: $m_{ij} = c$ and $M_{ij} = c$.
3. The list is sorted by the constants c, in increasing order.
4. We create a scenario $\eta = (\mathit{events}(\xi), \emptyset)$ and its distance table $\mathcal{D}^\eta$.
5. We now iteratively take consecutive items from L, compare each item with the contents of $\mathcal{D}^\eta$, and modify η and $\mathcal{D}^\eta$ as follows:[13]
   (a) If the item is of the form $m_{ij} = c$ and $l_{ij} < c$ then $l_{ij} := c$;
   otherwise the item is of the form $M_{ij} = c$, and if $h_{ij} > c$ then $h_{ij} := c$.
   (b) If the table (i.e., $\mathcal{D}^\eta$) was modified in the above step, add the corresponding constraint to η and stabilise the table.

Upon termination $\mathcal{D}^\eta$ is identical to $\mathcal{D}_s^\xi$, so $[\![\eta]\!] = [\![\xi]\!]$. However, if a constraint with a higher constant is implied by constraints with lower constants, then by

[12] If the attempt to do so fails because of the inconsistency of ξ, then producing an equivalent scenario with smaller constants is trivial (and probably pointless).

[13] More formally, we create a finite sequence of scenarios, $\eta^0\eta^1\ldots$ and a corresponding sequence of tables, $\mathcal{D}^{\eta^0}\mathcal{D}^{\eta^1}\ldots$. We felt that a more pedantic presentation would be harder to follow.

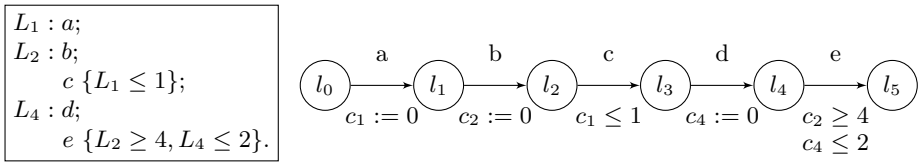

Fig. 8: Scenario ξ and its corresponding timed automaton A_ξ

the time we get to it in step 5a it will already be present in the table, and will not be explicitly added to η in step 5b.

Given scenario ξ of Fig. 1 and its stable distance table (shown in Fig. 7), the algorithm described above produces the optimised scenario η shown in Fig. 1.

5 Comparison with Difference Bounds Matrices

Difference Bounds Matrices (DBMs) have been proposed by Dill [11] as an efficient technique for representing clock zones in the context of verification of timed automata. A clock zone is a set of constraints, each of which puts a bound on the difference between the values of two clocks. A consistent DBM has a canonical form that can be obtained by computing all-pairs shortest paths.

We will now show how Dill's technique can be adapted to address the consistency of scenarios and the minimisation of constants in their constraints.

Given a scenario ξ, one can construct a timed automaton A_ξ, such that the set of timed words over which A_ξ has an accepting run is equivalent to the set of behaviours of ξ. For instance, Fig. 8 shows a scenario, ξ, and its corresponding timed automaton A_ξ. If we make sure that every transition of A_ξ (including the last transition) is annotated with a new clock reset, then, after applying Dill's technique, the DBM that corresponds to the final zone of the augmented A_ξ will contain information that is equivalent to the stable distance table for ξ.

Assume A'_ξ is the automaton obtained by annotating transitions labeled with c and e in A_ξ with two new clocks c_3 and c_5, respectively. Fig. 9 shows the stable distance table of ξ and the DBM that represents the final zone of A'_ξ (in Dill's original work the DBM would also contain information about whether the inequalities are sharp). c_0 is a clock whose value is always 0, and an entry a in row c_i and column c_j is interpreted as $c_i - c_j \leq a$. For example, $c_5 - c_2 \leq -4$, i.e., $4 \leq c_2 - c_5$: this corresponds to the minimum in row 1, column 4 of the distance table.

If $c_i - c_j \leq a$, $c_j - c_i \leq b$ and $a < b$, then the DBM is inconsistent: this corresponds to the minimum becoming larger than the maximum in the corresponding entry of the distance table.

The equivalence of a distance table and a DBM has interesting implications. On the one hand, we can replace the distance table with a DBM in the algorithm of Sec. 4 (the process would be slightly more complicated, because the constraints are encoded in a DBM somewhat less directly than in a distance table). On the

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

	c_0	c_1	c_2	c_3	c_4	c_5
c_0	0	-4	-4	-3	0	0
c_1	∞	0	1	1	∞	∞
c_2	∞	0	0	1	∞	∞
c_3	∞	0	0	0	∞	∞
c_4	2	-2	-2	-1	0	2
c_5	0	-4	-4	-3	0	0

Fig. 9: The distance table of ξ of Fig. 8 and the final DBM of A_ξ (with c_4 and c_5 added)

other hand, it is possible to take advantage of various techniques for DBMs and apply them to scenarios: for instance, the method of removing redundant constraints from a DBM (as described by Bengtsson [6]) can be used to remove redundant constraints from a distance table, and therefore—as we have shown—from a timed scenario.

The computational cost of applying Dill's original method to a scenario of length n would be $O(n^4)$: we would have to construct n instances of a DBM for n zones, and make each instance canonical at a cost of $O(n^3)$. However, it turns out that there are ways of preserving the canonicality of a DBM while moving to the next zone [6], so the overall cost would be $O(n^3)$, i.e., the same as ours.

In the final analysis, the technique of computing distance tables can be seen as an alternative and more direct approach to dealing with constraints in timed scenarios: in this context we find it simpler and more intuitive.

6 Related Work and Conclusions

For over three decades scenarios (including timed scenarios) have been proposed and used for specification, implementation and also synthesizing formal models of complex systems [22, 8, 10, 4].

For describing scenarios for real-time systems, researchers have proposed extending Message Sequence Charts (MSCs) with time constraints [5, 1].

En-Nouaary et. al [12] use timed scenarios for specifying systems, and integrate them to obtain a set of Timed Finite State Machines (TFSMs), a variant of timed automata. Their scenarios are described in a semi-formal language based on structured English or a graphical representation, and are therefore quite different from ours.

Somé et al [21] propose a method for synthesizing timed automata from a set of scenarios. Our timed scenarios are different from theirs: we do not include "conditions" in our scenarios. These "conditions" are not related to time and assert some facts about the status/mode of the decribed system.

The question of consistency and optimization of scenarios is not considered in any of the references cited above.

Harel et. al [14] study the problem of synthesizing state-based object systems from Live Sequence Charts (LSCs). They perform a consistency check of a set of LSCs to make sure that they are not contradictory with each other, in particular to check that the ordering of the events is correct. This consistency

check is different from ours: we consider individual timed scenarios and check their consistency in terms of time.

In our previous work [19] we proposed a form of timed scenarios and developed a method for synthesizing a timed automaton from a set of scenarios: we did not consider checking the consistency of scenarios or optimising them.

In the current paper we propose a notion of timed scenarios and their semantics that is both simpler and more general. We define the semantics of a scenario in terms of the set of behaviours that are supported by the scenario, and propose a method for checking the consistency of scenarios.

Checking the consistency of a set of constraints has already been studied in various contexts [11, 17, 18]. Our method is different: the consistency check is a simple byproduct of the construction of a "stable distance table". The table can be used as a canonical representation of the constraints of a class of equivalent scenarios, and is thus a good starting point for converting a scenario to an equivalent "optimised" form. Our optimisation minimizes the largest constant that appears in the constraints of a scenario, thus decreasing the maximum constant in the timed automaton synthesized from a set of scenarios.

We also show that the technique developed by Dill for representing clock zones in timed automata [11] can be applied to the domain of timed scenarios, both to check consistency and to minimise constants in constraints.

References

1. Akshay, S., Mukund, M., Kumar, K.N.: Checking Coverage for Infinite Collections of Timed Scenarios. In: CONCUR 2007 - Concurrency Theory, 18th International Conference, CONCUR 2007, Lisbon, Portugal, September 3-8, 2007, Proceedings. pp. 181–196 (2007), https://doi.org/10.1007/978-3-540-74407-8_13
2. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Comput. Sci. 126(2), 183–235 (Apr 1994)
3. Alur, R., Madhusudan, P.: 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 2004, Bertinoro, Italy, September 13-18, 2004, Revised Lectures. pp. 1–24 (2004)
4. Alur, R., Martin, M., Raghothaman, M., Stergiou, C., Tripakis, S., Udupa, A.: Synthesizing Finite-State Protocols from Scenarios and Requirements. In: Yahav, E. (ed.) Hardware and Software: Verification and Testing. pp. 75–91. Springer International Publishing, Cham (2014)
5. Ben-Abdallah, H., Leue, S.: Timing Constraints in Message Sequence Chart Specifications. In: Togashi, A., Mizuno, T., Shiratori, N., Higashino, T. (eds.) FORTE. IFIP Conference Proceedings, vol. 107, pp. 91–106. Chapman & Hall (1997)
6. Bengtsson, J.: Clocks, DBMs and states in timed systems. Ph.D. thesis, Uppsala University (2002)
7. Bengtsson, J., Larsen, K.G., Larsson, F., Pettersson, P., 0001, W.Y.: UPPAAL — a Tool Suite for Automatic Verification of Real-Time Systems. In: Hybrid Systems. Lecture Notes in Computer Science, vol. 1066, pp. 232–243. Springer (1995)
8. Bollig, B., Katoen, J., Kern, C., Leucker, M.: Replaying Play In and Play Out: Synthesis of Design Models from Scenarios by Learning. In: Tools and Algorithms for

the Construction and Analysis of Systems, 13th International Conference, TACAS 2007, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2007 Braga, Portugal, March 24 - April 1, 2007, Proceedings. pp. 435–450 (2007), https://doi.org/10.1007/978-3-540-71209-1_33
9. Bozga, M., Daws, C., Maler, O., Olivero, A., Tripakis, S., Yovine, S.: Kronos: A Model-Checking Tool for Real-Time Systems. In: Computer Aided Verification, 10th International Conference, CAV '98, Vancouver, BC, Canada, June 28 - July 2, 1998, Proceedings. pp. 546–550 (1998)
10. Damas, C., Lambeau, B., Roucoux, F., van Lamsweerde, A.: Analyzing Critical Process Models Through Behavior Model Synthesis. In: Proceedings of the 31st International Conference on Software Engineering. pp. 441–451. IEEE Computer Society (2009)
11. Dill, D.L.: Timing assumptions and verification of finite-state concurrent systems. In: Proceedings of the International Workshop on Automatic Verification Methods for Finite State Systems. pp. 197–212. Springer-Verlag New York, Inc., New York, NY, USA (1990)
12. En-Nouaary, A., Dssouli, R., Khendek, F.: From timed scenarios to SDL: specification, implementation and testing of real-time systems. In: SDL Forum. p. 67 (1999)
13. Giese, H.: Towards Scenario–Based Synthesis for Parametric Timed Automata. In: Proceedings of the 2nd International Workshop on Scenarios and State Machines: Models, Algorithms, and Tools (SCESM), Portland, USA (2003)
14. Harel, D., Kugler, H.: Synthesizing State-Based Object Systems from LSC Specifications. In: Yu, S., Păun, A. (eds.) Implementation and Application of Automata. pp. 1–33. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
15. Harel, D., Kugler, H., Pnueli, A.: Synthesis Revisited: Generating Statechart Models from Scenario-Based Requirements, pp. 309–324. Springer Berlin Heidelberg, Berlin, Heidelberg (2005), https://doi.org/10.1007/978-3-540-31847-7_18
16. Heitmeyer, C.L., Pickett, M., Leonard, E.I., Archer, M.M., Ray, I., Aha, D.W., Trafton, J.G.: Building high assurance human-centric decision systems. Autom. Softw. Eng. 22(2), 159–197 (2015)
17. Mahfoudh, M., Niebert, P., Asarin, E., Maler, O.: A satisfiability checker for difference logic. In: Fifth International Symposium on the Theory and Applications of Satisfiability Testing (2002)
18. Nieuwenhuis, R., Oliveras, A.: DPLL(T) with exhaustive theory propagation and its application to difference logic. In: Etessami, K., Rajamani, S.K. (eds.) Computer Aided Verification. pp. 321–334. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
19. Saeedloei, N., Kluźniak, F.: From Scenarios to Timed Automata. In: Formal Methods: Foundations and Applications - 20th Brazilian Symposium, SBMF 2017, Recife, Brazil, November 29 - December 1, 2017, Proceedings. pp. 33–51 (2017)
20. Somé, S., Dssouli, R., Vaucher, J.: From Scenarios to Timed Automata: Building Specifications from Users Requirements. In: Proceedings of the Second Asia Pacific Software Engineering Conference. pp. 48–57. APSEC '95, IEEE Computer Society, Washington, DC, USA (1995)
21. Somé, S., Dssouli, R., Vaucher, J.: From Scenarios to Timed Automata: Building Specifications from Users Requirements. In: Proceedings of the Second Asia Pacific Software Engineering Conference. pp. 48–57. IEEE Computer Society (1995)
22. Uchitel, S., Kramer, J., Magee, J.: Synthesis of behavioral models from scenarios. IEEE Transactions on Software Engineering 29(2), 99–115 (Feb 2003)