Scenario-Based Analysis and Synthesis of Real-Time Systems Using Uppaal

Kim G. Larsen, Shuhao Li, Brian Nielsen, and Saulius Pusinskas

Center for Embedded Software Systems (CISS)
Aalborg University
Selma Lagerlöfs Vej 300, DK-9220 Aalborg, Denmark
{kgl, li, bnielsen, saulius}@cs.aau.dk

Abstract. We propose an approach to scenario-based analysis and synthesis of real-time embedded systems. The inter-process behaviors of a system are modeled as a set of driving universal Live Sequence Charts (LSCs), and the scenario-based user requirement is specified as a separate monitored universal or existential LSC. By translating the set of LSCs into a behavior-equivalent network of timed automata (TA), we reduce the problems of model consistency checking and property verification to CTL real-time model checking problems. Similarly, we reduce the problem of centralized synthesis for open systems to a timed game solving problem. We implement a prototype LSC-to-TA translator, which can be linked to our LSC editor and the existing real-time model checker Uppaal and timed game solver Uppaal-Tiga. Preliminary experiments on a number of examples and a case study show the applicability and effectiveness of this approach.

1 Introduction

In the early stage of model-based development of real-time embedded systems, it is desirable to prototype a system in question and its operating environment using a number of use cases and scenarios. A next round refinement of the system model and the transition from model to implementation can start only if the scenario-based model is checked to be consistent (i.e., implementable), and correct with respect to some specified scenario-based user requirements. Furthermore, from a correct-by-construction perspective, as a part of the long-known dream for "automated system design", we may wish to synthesize executable object systems from scenario-based descriptions.

Live Sequence Chart (LSC) [7, 14] is a scenario-based specification and programming language, which can describe systems in an assume-guarantee style. A universal LSC chart can optionally contain a *prechart*, which specifies the scenario which, if successfully executed, forces the system to satisfy the scenario given in the actual chart body (the *main chart*). Essentially, LSC extends Message Sequence Chart (MSC) [15] by adding modalities. The *existential* and *cold* (resp. *universal* and *hot*) modalities represent the provisional (resp. mandatory) global and local requirements, respectively. An existential LSC chart (resp.

universal LSC chart) specifies restrictions over at least one satisfying (resp. all possible) system runs. A cold condition may be violated, whereas a hot one must be satisfied. The rich language facilities and the unambiguous semantics make LSC a nice visual formalism for early-stage characterization of distributed, real-time and embedded systems.

Scenario-based approaches that use LSCs enjoy the advantages of piecewise incremental construction of system models, i.e., new pieces of scenarios can be added into the models during the development process. However, scenario-based analysis such as model consistency checking and property verification (i.e., to check whether an LSC-modeled system satisfies a scenario-based requirement) are difficult due to the need to consider both the explicitly as well as implicitly specified behaviors in each scenario, and the interplays among the different scenarios. Scenario-based synthesis is difficult because all possible scenario interactions have to be considered [1]. The problems become even more complicated for real-time systems, as time-enriched LSCs may contain subtle timing errors that are difficult to diagnose.

Powerful verification techniques and tools are utilized to assist scenario-based analysis and synthesis for LSCs, e.g. in smart play-out [11, 12, 6], satisfiability checking [22, 6], consistency checking [22] and synthesis [10, 13, 23, 5, 18].

Some of these efforts address the problems mainly from a theoretical viewpoint [10, 5]. Some of them handle property verification with only existential charts [22, 6]. A number of them have no real-time support [10, 11, 22, 23, 5]. While synthesis for LSC-modeled real-time systems is supported in [13], it is *incomplete* in the sense that some systems are announced not synthesizable while they actually are. The reason is that the smart play-out mechanism [11, 12] that [13] relies on implements only a *local* consistency checking where the Play-Engine looks only one super-step ahead in the LSC state space. A recent work [18] tackles this by employing controller synthesis techniques to achieve complete consistency checking and subsequent complete synthesis. However as an extension to smart play-out, this method, like the earlier work [13], is limited to discrete time and a restricted form of timing constraints [2].

In this paper we equip a kernel subset of the LSC language with timed automaton (TA)[1] -like real-valued clock variables and clock constraints. We use a set of driving universal LSC charts (collectively called an *LSC system*) to model the inter-process interaction behaviors of the system in question, and use a monitored universal/existential chart to specify the user requirement. We

[1] It has been shown that for an untimed subset of the LSC language, consistency checking and the subsequent synthesis is at least EXPTIME-complete, and model checking a given system implementation (e.g. in I/O automata) against an LSC specification is complete for co-NP (for centralized implementation of a closed system) or PSPACE (otherwise) [4].

[2] In the original definition and the Play-Engine implementation of time-enriched LSC [14], the special `Clock` object has a property *Time* which is an integer variable, and a method *Tick* which each time increases *Time* by 1. Timing constraints take the form of only "*Time* **op** (time variable + delay expression)", where **op** is an relational operator.

translate the LSC system into a behavior-equivalent network of interacting timed automata, which can properly mimic the important LSC features such as message sending and intra/inter-chart coordinations. The problems of consistency checking and property verification can be reduced to CTL real-time model checking problems in UPPAAL [3]. Furthermore, by viewing the interaction between the controllable system processes ($\mathcal{S}ys$) of an LSC system and their "hostile" (uncontrollable) environment processes ($\mathcal{E}nv$) as playing a (safety) game with the aim of constantly avoiding hot violations, we can reduce the problem of scenario-based synthesis for open systems (i.e., $\mathcal{S}ys$) to a game solving problem. The timed game solver UPPAAL-TIGA [2] can be employed to check the solvability, and if yes, to generate a strategy for $\mathcal{S}ys$. Compared with existing work, our approach of scenario-based analysis and synthesis features:

- TA-like real-valued clock variables and clock constraints (compared with [12, 13, 6, 18]);
- Complete consistency checking and synthesis (compared with [10, 13]);
- Property verification with the properties being specified as universal charts (compared with [22, 6]); and
- Automated, tool-supported approach (compared with [10, 5]).

The benefits of building the scenario-based analysis and synthesis methods on top of the well-developed real-time model checking and timed game solving engines are twofold: (1) for system analysts and designers who are interested in early-stage validations and automated system designs using live sequence charts, now they can carry out scenario-based analysis and synthesis of *timed* systems without having to develop and implement the underlying algorithms; (2) for the users of conventional model checkers that work on state/transition-based models and temporal logical properties, now they can horizontally scale up to *scenario-based* models and *scenario-based* user requirements.

1.1 Related Work

In addition to being used as a requirement specification language [8, 17, 16], LSC can also be used as a modeling language [9–12, 4, 22, 23, 13, 6, 18]. In the latter case, problems of scenario-based analysis such as model consistency checking and property verification, and of scenario-based synthesis arise. This paper will mainly consider the latter usage of LSC charts.

Analysis of scenario-based behaviors can be carried out inside the Play-engine [14] according to the operational semantics of LSC [9–13, 6, 18], or achieved by first transforming LSCs into other formalisms such as CSP [22, 23] and timed Büchi automata (TBA) [17], and then calling for other mature techniques and tools to accomplish the tasks. In this paper we will follow the latter approach to take advantage of the TA formalism [1], the real-time model checker UPPAAL [3] and the timed game solver UPPAAL-TIGA [2]. Similar in spirit to the approaches of [11, 22], during the translation we will create one process for each instance line of the charts, and thus avoid the explicit construction of the global transition system.

A set of LSC charts are *consistent* if and only if these charts are not internally contradictory, i.e., they can be satisfied by a certain state-based object system [9, 10]. Consistency checking of LSC systems is a prerequisite step towards *synthesis* [13, 4, 22, 23, 18], i.e., to construct one such satisfying state-based object system. In this paper, we consider both the consistency checking and the synthesis problems, and will reduce them to the model checking and game solving problems, respectively.

Current work on verification of scenario-based requirement either has state/transition-based object systems as the subject, e.g., STATEMATE model implementations [8, 17] and Kripke structures [16], or has an LSC system as the subject. In the latter case, existing work mainly concerns satisfiability checking [22, 6], i.e., to check whether the requirement that is expressed as an existential chart can be satisfied by the LSC system. In this paper we will also check whether a requirement specified as a *universal* chart can always be respected by the LSC system.

1.2 Organization

Section 2 presents our timed extensions to a subset of the LSC language and defines a trace-based semantics. Section 3 describes how we translate LSCs into a network of timed automata, shows how complex the outcome of the translation is, and shows the behavior-equivalence of the translation. The scenario-based analysis and synthesis problems are formulated as model checking and game solving problems in Section 4. The prototype tool implementation and preliminary experiments are reported in Section 5. We conclude in Section 6. The detailed translation rules, the complexity analysis of the translation outcomes, and the proofs of lemmas and theorems in Sections 3.4 and 4 are provided in the appendices.

2 Live Sequence Charts

In this paper, LSC in its simplest form is a message-only untimed chart, i.e., there are only language elements of instance lines, locations, messages and precharts/main charts.

We make the *synchrony* hypothesis, i.e., system events consume no real time and time may elapse only between events. In this way message synchronizations will be instantaneous, i.e., the sending and reception of a message are assumed to happen at the same moment in time. Therefore the terms of (message sending or receiving) *event* and *message* will be used interchangeably.

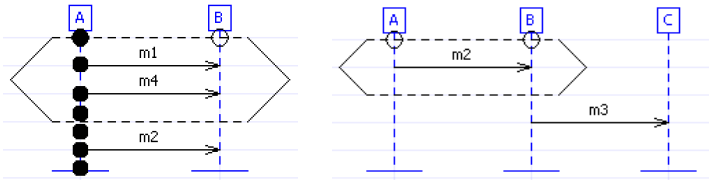
A chart has a *role*, either for system modeling (i.e., as a driving chart), or for system property specification (i.e., as a monitored chart). Driving charts can only be universal charts, whereas a monitored chart can be either a universal or an existential chart. In this paper we use a set of universal charts to model the behaviors of the system in question, and use a separate universal or existential chart to specify the system property.

A universal LSC chart has an *activation mode*, i.e., how often a chart should be activated. In this paper, we consider the invariant mode, i.e., the prechart is being constantly monitored, regardless of whether any incarnation of the chart has entered its main chart portion.

2.1 Syntax and semantics for a single universal chart

A universal LSC chart has a main chart (Mch) and optionally a prechart (Pch). If it has no prechart, then it can be simply treated as having a satisfying prechart. In this paper we assume that a universal chart has a prechart.

We start with message-only untimed charts, see Fig. 1 for the examples.



(a) chart 1 (b) chart 2

Fig. 1. Two consistent untimed charts.

Given a universal chart $\mathcal{L}$, let $I = inst(\mathcal{L})$ be the set of instance lines (i.e., processes) in $\mathcal{L}$. Along each instance line $I_i \in I$ there are a finite set of "positions" $pos(\mathcal{L}, I_i) = \{0, 1, 2, \ldots, p\_max_{\mathcal{L},I_i}\} \subset \mathbb{N}_{\geq 0}$. See Fig. 1(a), black filled circles. Specifically, along each instance line I_i there are four "standard" positions $StdPos(\mathcal{L}, I_i) = \{Pch\_top, Pch\_bot, Mch\_top, Mch\_bot\} \subset pos(\mathcal{L}, I_i)$, denoting the entry/exit points of the prechart/main chart, respectively, such that:

- $0 = Pch\_top < Pch\_bot < Mch\_top < Mch\_bot = p\_max_{\mathcal{L},I_i}$; and
- $Pch\_bot + 1 = Mch\_top$. □

A chart *location* is a position on a certain instance line of the chart. The set of all locations of chart $\mathcal{L}$ are denoted as:

$$loc(\mathcal{L}) = \{\langle I_i, p\rangle \mid I_i \in inst(\mathcal{L}),\, p \in pos(\mathcal{L}, I_i)\}.$$

The set of all *message-anchoring* locations of $\mathcal{L}$ are denoted as:

$$loc_M(\mathcal{L}) = \{\langle I_i, p\rangle \mid I_i \in inst(\mathcal{L}),\, p \in pos(\mathcal{L}, I_i) \backslash StdPos(\mathcal{L}, I_i)\}.$$

Furthermore, we define function $psn : loc(\mathcal{L}) \rightarrow \bigcup_{I_i \in inst(\mathcal{L})} pos(\mathcal{L}, I_i)$ to project a location to its **posi**tio**n** on its instance line.

Let $ML(\mathcal{L})$ be the set of **m**essage **l**abels (or "signals", or "channels" in UPPAAL) of chart $\mathcal{L}$. A *message occurrence* $mo = (\langle I_i, p\rangle, m, \langle I_{i'}, p'\rangle) \in loc_M(\mathcal{L}) \times ML(\mathcal{L}) \times loc_M(\mathcal{L})$ corresponds to instance I_i, while in its position $(p - 1)$,

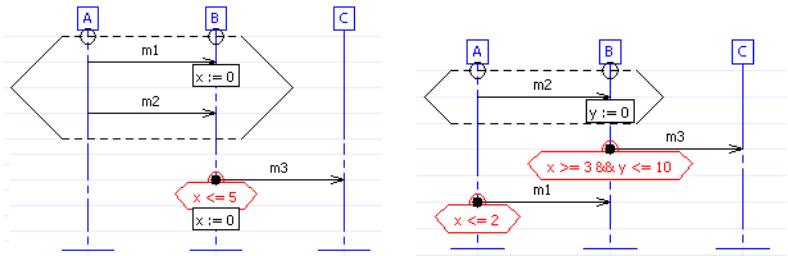
sending signal $m \in ML(\mathcal{L})$ to instance $I_{i'}$ at its position $(p'-1)$, and then arriving at positions p and p', respectively. We call $lab(mo) = m$ the message label, $head(mo) = \langle I_{i'}, p' \rangle$ and $tail(mo) = \langle I_i, p \rangle$ the message head and tail locations, and $src(mo) = I_i$ and $dest(mo) = I_{i'}$ the source and destination instances, respectively. We use $loc(mo) = \{head(mo), tail(mo)\}$ to denote the message anchoring locations. The set of all message occurrences in chart $\mathcal{L}$ are denoted as:

$$MO(\mathcal{L}) \subseteq \{(\langle I_i, p \rangle, m, \langle I_{i'}, p' \rangle) \in loc_M(\mathcal{L}) \times ML(\mathcal{L}) \times loc_M(\mathcal{L}) \mid i \neq i',$$
$$p \leq StdPos(\mathcal{L}, I_i).Pch\_bot \Leftrightarrow p' \leq StdPos(\mathcal{L}, I_{i'}).Pch\_bot\}.$$

We omit the parameter $\mathcal{L}$ in $MO(\mathcal{L})$ (and thus abbreviating it as MO) when it is clear from the context. Furthermore, we use $\Sigma = MA(\mathcal{L})$ to denote the projection of $MO(\mathcal{L})$ onto $inst(\mathcal{L}) \times ML(\mathcal{L}) \times inst(\mathcal{L})$. In this way, we get the **m**essage **a**lphabet Σ, where each letter is a *message* which denotes that a particular signal is sent from one to another objects (instance lines). For a given message occurrence, we may overload its "message label" to also denote the corresponding letter in Σ.

This paper does not consider concurrent messages, thus each location can be the end point of at most one message occurrence in the chart.

Now we continue to define our timed extensions to the above kernel subset of the LSC language. In our time-enriched LSC charts, there are further elements of (clock) variables, conditions (clock constraints), assignments (clock resets) and simregion (i.e., "simultaneous region"). Fig. 2 gives two example time-enriched LSC charts.



(a) chart 1 (b) chart 2

Fig. 2. Two consistent time-enriched charts.

Assume that in chart $\mathcal{L}$ there are a finite set X of real-valued *clock variables* that range over $\mathbb{R}_{\geq 0}$. A *clock valuation* is a function $v : X \to \mathbb{R}_{\geq 0}$ that maps each clock variable to a non-negative real number, also denoted $v \in {\mathbb{R}_{\geq 0}}^X$.

A *clock constraint* is of the form $x \bowtie n$ or $x - y \bowtie n$ where $x, y \in X$, $n \in \mathbb{Z}$, and $\bowtie \in \{<, \leq, =, \geq, >\}$. Let $B(X)$ be the set of finite conjunctions over these constraints. The set of *conditions* (or guards) in the chart are denoted

$G \subseteq B(X)$. A condition $g \in G$ has a temperature, denoted $g.temp$, which may be either hot or cold in the main chart, and only cold in the prechart.

A *clock reset* is of the form $x := 0$ where $x \in X$. An *assignment* (or update) a is the union of a finite set of clock resets. For simplicity we use a to denote the set of clocks to be reset. The set of all assignments in the chart is denoted $A \subseteq 2^X$. We can also view $a \in A$ as a transformer on the functions of clock valuations, and as such the new valuation of v after assignment a is denoted as $v' = a(v)$.

In our time-enriched LSCs, each message occurrence mo can be optionally associated with a condition g and/or an assignment a. The intuitive meaning of message synchronization $[g]\, mo\, /a$ from location $\langle I_i, p\rangle$ to $\langle I_{i'}, p'\rangle$ is that, if when mo occurs, the clock valuation v satisfies g, then this synchronization can fire; and immediately after the firing, v will be updated according to a. A message occurrence and the corresponding condition and/or assignment attached thereto can be collectively viewed as an atomic step of LSC execution, i.e., they take place at the same moment in time, hence they constitute a simregion. We denote the set of all simregions as SR. For simplicity, we do not consider stand-alone conditions or assignments, i.e., we assume that any condition and assignment is associated with a certain message.

In an LSC chart $\mathcal{L}$, every location is either associated with a simregion, or it is an entry/exit point of the prechart/main chart. We define a labeling function $\lambda : loc(\mathcal{L}) \to SR \cup \{nil\}$ to map a location of the former type to its corresponding simregion, and a location of the latter type to nil.

Locations in a chart $\mathcal{L}$ are partially ordered by the following rules:

- Along each instance line I_i: location l is above $l' \Rightarrow (l \leq l') \wedge \neg(l' \leq l)$; and
- All locations in the same simregion have the same order, $\forall s \in SR, \forall l, l' \in loc(\mathcal{L})\,.\,(\lambda(l) = s) \wedge (\lambda(l') = s) \Rightarrow (l \leq l') \wedge (l' \leq l)$. □

The partial order relation $\preccurlyeq \subseteq loc(\mathcal{L}) \times loc(\mathcal{L})$ is defined as a transitive closure of $\leq$.

Definition 1 (cut of an LSC chart). *A* cut *of a chart $\mathcal{L}$ is a set $c \subseteq loc(\mathcal{L})$ of locations that span across all the instance lines in $\mathcal{L}$ which satisfies the properties of:*

- Downward-closure*. If a location l is included in cut c, so are all of its predecessor locations: $\forall l,\, l' \in loc(\mathcal{L}).\, (l \in c \wedge l' \preccurlyeq l) \Rightarrow l' \in c$; and*
- Intra-chart coordination integrity*. If a Mch_top position of a certain instance line is included in the cut, then the Mch_top positions of all other instance lines are also included in the cut: $\exists l \in loc(\mathcal{L}),\, I_i \in inst(\mathcal{L})\,.\,($ $(StdPos(\mathcal{L}, I_i).Mch\_top \leq psn(l)) \wedge (psn(l) \leq StdPos(\mathcal{L}, I_i).Mch\_top) \wedge (l \in c) \Rightarrow \forall l' \in loc(\mathcal{L}), I_{i'} \in inst(\mathcal{L})\,.\,((StdPos(\mathcal{L}, I_{i'}).Mch\_top \leq psn(l')) \wedge (psn(l') \leq StdPos(\mathcal{L}, I_{i'}).Mch\_top) \Rightarrow l' \in c))$. □

For a cut c, we use $loc(c)$ to denote its *frontier*, i.e., the set of locations that constitute the downward borderline progressed so far. The location where c "cuts" instance line $I_k \in I$ is denoted $loc(c)_{\langle k \rangle}$.

Given a cut $c \subseteq loc(\mathcal{L})$ and a simregion $s \in SR$, we say s is *enabled* at cut c (with respect to the partial order relation), denoted $c \xrightarrow{s}$, if, $\forall\, l \in c,\ l' \in loc(s)\,.\,((l \preccurlyeq l') \wedge \neg(l' \preccurlyeq l)) \wedge (\nexists\, l'' \in loc(\mathcal{L}) \backslash (c \cup loc(s))\,.\,(l \preccurlyeq l'' \wedge l'' \preccurlyeq l'))$. The enabledness of message occurrences can be defined similarly.

A cut c' is an *s-successor* of cut c, denoted $c \xrightarrow{s} c'$, if s is enabled at c (w.r.t. the partial order), and c' is achieved by adding the set of locations that s anchors into c, or formally, $(c \xrightarrow{s}) \wedge (c' = c \cup loc(s))$.

A cut c is *minimal*, denoted $\top$, if it "cuts" each instance line at its top location; and c is *maximal*, denoted $\bot$, if if it "cuts" each instance line at its bottom location. The minimal and maximal cuts of the prechart and main chart are denoted $Pch.\top$, $Pch.\bot$, $Mch.\top$ and $Mch.\bot$, respectively. The frontiers of minimal and maximal cuts do not contain simregion anchoring points. Rather the cuts $Pch.\bot$ and $Mch.\bot$ each represent a requirement for compulsory synchronization for all the instance lines in the chart. Thus the partial order relation $\preccurlyeq$ on $loc(\mathcal{L})$ is extended as follows (and finally also extended to its transitive closure):

- All locations in the frontier of the same minimal or maximal cut have the same order, $\forall c \in \{Pch.\top,\ Pch.\bot, Mch.\top, Mch.\bot\}\,,\ \forall l, l' \in loc(c)\,.\,(l \preccurlyeq l') \wedge (l' \preccurlyeq l)$. □

Definition 2 (configuration). *A* configuration *of an LSC chart is a tuple (c, v), where c is a cut and v is a clock valuation.* □

For $d \in \mathbb{R}_{\geq 0}$, notation $(v+d) : X \to \mathbb{R}_{\geq 0}$ means that the function v is shifted by d such that $\forall x \in X\,.\,(v(x+d) = v(x) + d)$.

A configuration at the minimal cut $\top$ with all clocks assigned their initial values (e.g., 0's) is called the *initial* configuration.

A configuration can be viewed as a "semantic state" of a time-enriched LSC chart. A universal chart starts from the initial configuration, advances from one to a next configuration, until a hot violation [3] occurs, or until the chart arrives at the maximal cut configuration and then starts all over again (i.e., to begin a next round execution).

There could be three kinds of advancement steps between two configurations (c, v) and (c', v') of a time-enriched LSC chart:

- *Message synchronization step.* Given a simregion s which consists of an m-labeled message occurrence mo $(m \in \Sigma)$, and optionally a condition g and/or an assignment a, there is a message synchronization step $(c, v) \xrightarrow{m} (c', v')$ if,
  - (normal advancement). $c \xrightarrow{s} c'$, $v \models g$, and $v' = a(v)$; or

[3] A *hot* violation means that some mandatory requirements are not satisfied. In this paper, it refers to the situations that in the main chart the event partial order is violated or a hot condition evaluates to false. In comparison, a *cold* violation means that some provisional requirements are not satisfied (therefore it is not a big deal). In this paper, it refers to the situations where the event partial order is violated in the prechart, or a cold condition evaluates to false.

- (cold violation). $c' = Pch.\top$, $v' = v$, and either
  - *mo* is not enabled at cut c in the prechart (w.r.t. the partial order relation); or
  - $(v \not\models g) \wedge (g.temp = \text{cold})$;

– *Silent step.* There is a silent step $(c, v) \xrightarrow{\tau} (c', v')$ if either
- $(c = Pch.\bot,\ \ c' = Mch.\top,\ \ v' = v)$; or
- $(c = Mch.\bot,\ \ c' = Pch.\top,\ \ v' = v)$; or
- c' is reached because an instance line moves to its bottom location in *Pch* or *Mch* autonomously (this happens when the instance line will not interact with other instance lines before it reaches its bottom location in *Mch* or *Pch*). Formally, there exists an instance I_k such that $v' = v$ and either
  - $loc(c')_{\langle k \rangle} = (loc(Pch.\bot))_{\langle k \rangle},\ \ psn(loc(c')_{\langle k \rangle}) = psn(loc(c)_{\langle k \rangle}) + 1$, and $loc(c')_{\langle i \rangle} = loc(c)_{\langle i \rangle}$ for all $i \neq k$; or
  - $loc(c')_{\langle k \rangle} = (loc(Mch.\bot))_{\langle k \rangle},\ \ psn(loc(c')_{\langle k \rangle}) = psn(loc(c)_{\langle k \rangle}) + 1$, and $loc(c')_{\langle i \rangle} = loc(c)_{\langle i \rangle}$ for all $i \neq k$;

– *Time delay step.* There is a time delay step $(c, v) \xrightarrow{d} (c', v')$ where $d \in \mathbb{R}_{\geq 0}$ if: $c' = c$, $v' = v + d$, and whenever there are message occurrences that are enabled at cut c (w.r.t. both the partial order relation and the guard), then after delay d there exists at least one of them that is still enabled at the same cut, i.e., $\exists s \in SR\,.\,\exists\, mo, g \in s\,.\,\forall d' \in [0, d]\,.\,(c \xrightarrow{s}) \wedge (v + d') \models g$. □

Similarly, if in the main chart, an m-labeled message violates $\preccurlyeq$, or $(v \not\models g \wedge g.temp = \text{hot})$, then the configuration (c, v) is said to be *hot-violated*, denoted $(c, v) \xrightarrow{m}\!\!\!\!\!\!\diagup$.

Definition 3 (run of an LSC chart). *A* run *of a time-enriched universal LSC chart is a sequence of configurations* $(c^0, v^0) \cdot (c^1, v^1) \cdot \ldots$ *that are connected by the advancement steps, i.e.,* $\forall i \geq 0\,.\,\exists\, u_i \in (\Sigma \cup \{\tau\} \cup \mathbb{R}_{\geq 0})\,.\,(c^i, v^i) \xrightarrow{u_i} (c^{i+1}, v^{i+1})$. □

The transition relation $\rightarrow$ as mentioned above each time consumes only a single letter $u \in (\Sigma \cup \{\tau\} \cup \mathbb{R}_{\geq 0})$. We extend it to $\rightarrow^*$ such that it consumes a (finite or infinite) word $w \in (\Sigma \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^* \cup (\Sigma \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^\omega$.

Let Π correspond to the set of all possible messages that occur in a state/ transition-based system model (i.e., a network of timed automata), or be the set of all messages in an object interaction-based system model (i.e., a set of driving universal LSC charts). In the latter case, the message alphabet for the LSC system model $\mathcal{LS} = \{\mathcal{L}_i \mid 1 \leq i \leq n\}$ is $\Pi = \bigcup_{i=1}^{n} \Sigma_i = \bigcup_{i=1}^{n} MA(\mathcal{L}_i)$.

Definition 4 (satisfaction of a prechart/main chart). *A timed trace* $\gamma \in (\Pi \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^* \cup (\Pi \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^\omega$ satisfies *an LSC prechart or main chart* $\mathcal{C}$*, denoted* $\gamma \models \mathcal{C}$*, if its projection* $\gamma|_{(\Sigma \cup \{\tau\} \cup \mathbb{R}_{\geq 0})}$ *has a prefix* μ *which is the accepted word of a run that successfully exercises* $\mathcal{C}$*, and no prefix of it ever leads to a hot violation. Formally,* $(\exists \mu \in (\Sigma \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^*, \xi \in (\Sigma \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^* \cup (\Sigma \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^\omega\,.\,\exists v' \in {\mathbb{R}_{\geq 0}}^X\,.\,(\gamma|_{(\Sigma \cup \{\tau\} \cup \mathbb{R}_{\geq 0})} = \mu \cdot \xi) \wedge (\top, v^0) \xrightarrow{\mu}{}^*$

$(\bot, v')) \wedge (\nexists \mu' \in (\Sigma \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^*, \xi \in (\Sigma \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^* \cup (\Sigma \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^\omega \,.\, \forall m \in \Sigma \,.\, ((\gamma|_{(\Sigma \cup \{\tau\} \cup \mathbb{R}_{\geq 0})} = \mu' \cdot m \cdot \xi) \wedge (\top, v^0) \xrightarrow{\mu'}{}^* \bullet \not\xrightarrow{m}))$. □

A finite trace $\gamma \in (\Pi \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^*$ satisfies chart $\mathcal{C}$ *exactly*, denoted $\gamma \Vdash \mathcal{C}$, iff $(\gamma \models \mathcal{C}) \wedge \exists \mu \in (\Pi \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^*,\ v' \in {\mathbb{R}_{\geq 0}}^X \,.\, (\gamma|_{(\Sigma \cup \{\tau\} \cup \mathbb{R}_{\geq 0})} = \mu) \wedge ((\top, v^0) \xrightarrow{\mu}{}^* (\bot, v'))$.

Now we define the satisfaction relation for a full universal chart (under the invariant activation mode):

Definition 5 (satisfaction of a universal LSC chart). *A timed trace* $\gamma \in (\Pi \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^\omega$ satisfies *(passes) a universal chart* $\mathcal{L}$*, denoted* $\gamma \models \mathcal{L}$*, iff whenever a finite sub-trace matches the prechart, then the main chart is matched immediately afterwards. Formally,* $\forall \alpha, \mu \in (\Pi \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^*, \beta \in (\Pi \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^\omega \,.\, (\alpha \cdot \mu \cdot \beta = \gamma) \wedge (\mu \Vdash Pch) \Rightarrow (\beta \models Mch)$. □

A timed language $Lang \subseteq (\Pi \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^\omega$ satisfies $\mathcal{L}$, denoted $Lang \models \mathcal{L}$, iff $\forall \gamma \in Lang \,.\, \gamma \models \mathcal{L}$. Clearly, $Lang$ characterizes the system behaviors that respect $\mathcal{L}$.

When $\mathcal{L}$ is used as a monitored chart, then for a network $\mathcal{S}$ of timed automata, we use $\mathcal{S} \models \mathcal{L}$ to denote that the timed traces (language) of $\mathcal{S}$ satisfy LSC $\mathcal{L}$.

2.2 Semantics for a set of universal charts

For an LSC system $\mathcal{LS}$ which consists of a set of driving universal charts $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n$, we denote $\bar{c} = (c_1, c_2, \ldots, c_n)$ as a *cut vector*, and v as a *valuation* of all of the clock variables in $\mathcal{LS}$. Each member cut of $\bar{c}$ is denoted as $c_i = (\bar{c})_i$, $1 \leq i \leq n$. We call $(\bar{c}, v)$ a *global configuration* of $\mathcal{LS}$.

Let $(\bar{c}, v)$ be a global configuration of an LSC system $\mathcal{LS}$. Assume that there are message occurrences $mo_1, \ldots, mo_k$ ($1 \leq k \leq n$, each in a different chart) that are simultaneously enabled at $((\bar{c})_i, v)$, $1 \leq i \leq k$, and that these message occurrences are the same message, i.e., they have exactly the same message label and the same source and destination instances, i.e., $\exists m \in \Pi,\ \mathcal{L}_j \in \mathcal{LS} \,.\, \exists I_a, I_b \in inst(\mathcal{L}_j) \,.\, \forall 1 \leq i \leq k \,.\, (lab(mo_i) = m) \wedge (src(mo_i) = I_a) \wedge (dest(mo_i) = I_b)$. In this case, these identically labeled message occurrences are said to be *enabled* at global configuration $(\bar{c}, v)$ w.r.t. their respective partial order relations.

Given a global configuration $(\bar{c}, v)$ of $\mathcal{LS}$ and a message $m \in \Pi$, there is a message synchronization step $(\bar{c}, v) \xrightarrow{m} (\bar{c}', v')$ in $\mathcal{LS}$ if:

- A maximal set of m-labeled message occurrences are enabled at $(\bar{c}, v)$, and there is no chart $\mathcal{L}_i$ whose local configuration $((\bar{c})_i, v)$ will be hot-violated by an m-labeled message. In this case, for all charts $\mathcal{L}_j$ which each has an m-labeled message occurrence enabled at $(\bar{c}, v)$, the $\xrightarrow{m}$ message synchronization steps will occur simultaneously; and

there is a silent step $(\bar{c}, v) \xrightarrow{\tau} (\bar{c}', v)$ in $\mathcal{LS}$ if:

– There is a chart $\mathcal{L}_i$ such that $((\bar{c})_i, v) \xrightarrow{\tau} ((\bar{c}')_i, v)$. In this case, for all $j \neq i$, we have $\bar{c}'_j = \bar{c}_j$; and

there is a time delay step $(\bar{c}, v) \xrightarrow{d} (\bar{c}, v + d)$ in $\mathcal{LS}$ if:

– For all $1 \leq i \leq n$, we have $((\bar{c})_i, v) \xrightarrow{d} ((\bar{c})_i, v + d)$. □

In the first case above, the *global condition* for all m-labeled message occurrences is the conjunction of all individual conditions, and the *global assignment* is the union of all individual assignments.

Similarly, we can define runs and $\rightarrow^*$ for a set of time-enriched LSC charts.

Definition 6 (satisfaction of an LSC system). *A timed trace* $\gamma \in (\Pi \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^\omega$ satisfies *(passes) an LSC system* $\mathcal{LS}$ *iff,* γ *corresponds to an infinite run of* $\mathcal{LS}$*, and it satisfies each chart* $\mathcal{L}_i$ *in* $\mathcal{LS}$ *separately.* □

3 LSC to TA translation

3.1 Motivation

Similar to [19], in this paper our scenario-based analysis and synthesis methods rely on a translation of the LSC charts to timed automata. However, unlike in [19] where each monitored chart specifies a user requirement individually, in this paper a set of driving charts are supposed to characterize the inter-object behaviors of the system *collectively*. When the system consists of a large number of driving charts, then the cut-based LSC-to-TA translation will encounter the state explosion problem: the number of possible global cuts (i.e., the number of possible system states) will increase rapidly, and explicit encoding and storing these information need a lot of space. Furthermore, the outcome of the translation as a single huge timed automaton will be difficult to visualize, to debug and to diagnose.

To overcome the above problems, in this paper we propose a different method for translating LSC charts to timed automata. For each driving LSC chart $\mathcal{L}$ in the system model, we view the instance lines in $\mathcal{L}$ as a set of parallelly running processes that communicate with one another and collaborate to achieve a common goal as specified by chart $\mathcal{L}$. Since UPPAAL also operates on a network of parallelly composed processes (TAs) that communicate with each other, this motivates us to translate each instance line of $\mathcal{L}$ to a timed automaton. In this way we avoid the explicit construction of a global automaton. This idea in spirits resembles the approaches of [11, 22]. Thanks to the UPPAAL features of broadcast channels, boolean and integer variables and committed locations in timed automata, we are able to appropriately translate the LSC features such as message sending, intra/inter-chart coordinations and cold/hot violations to timed automata. Compared with the "one-TA-per-chart" approach that can be viewed as a kind of *centralized* translation [19], the "one-TA-per-instance line" approach of this section can be viewed as a kind of *distributed* translation.

3.2 Mapping LSC instance to Uppaal timed automaton

Basic structure mapping Each instance line I_i in chart $\mathcal{L}_u$ of the LSC system $\mathcal{LS}$ is mapped into a timed automaton $A_{u,i}$, where each position on I_i corresponds to a TA location in $A_{u,i}$, and each discrete advancement step (i.e., a message synchronization step or a silent step) on I_i corresponds to a TA edge in $A_{u,i}$. The sending (resp. receiving) of an m-labeled message on I_i corresponds to an $m!$ (resp. $m?$)-labeled TA edge in $A_{u,i}$.

Handling intra/inter-chart coordinations In the prechart (resp. main chart) of an LSC chart, once all the participating instance lines have progressed to their *Pch_bot* (resp. *Mch_bot*) positions, then the prechart (resp. main chart) is successfully matched. In this case, the prechart (resp. main chart) will be exited immediately, and the main chart (resp. prechart) will be entered immediately afterwards. To synchronize all the participating instance lines for such a prechart/main chart (resp. main chart/prechart) transfer, for each LSC chart $\mathcal{L}$, we create a dedicated (auxiliary) coordinator automaton $Coord_{\mathcal{L}}$. This automaton will communicate with the timed automata for the instance lines by using auxiliary binary synchronization channels such that it can bookkeep how many instance lines are done with their prechart (resp. main chart) portions. Once the coordinator automaton realizes that the prechart (resp. main chart) has been successfully matched, it will immediately launch a broadcast synchronization with the timed automata for all the relevant instance lines.

In scenario-based modeling, the same message may well appear in two or more charts. To be more specific, it is possible that an m-labeled message from instance line A to instance line B has its occurrences in LSC charts $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n$. If these messages are all enabled and one of them is chosen to be fired, then all the others must also be fired simultaneously. Clearly, this requires an inter-chart coordination. To achieve this, we use broadcast synchronization channels rather than binary synchronization channels for these messages. In the translated timed automata, if there is an $m!$-labeled edge from one to another TA locations, then we add an $m?$-labeled edge between these two TA locations to "accompany" the $m!$-labeled edge.

Handling cold and hot violations Once a cold violation occurs, we let the timed automaton that corresponds to the message sending instance line report to the coordinator automaton in charge, which in turn immediately synchronizes all the timed automata that correspond to the relevant instance lines in the chart for a reset (i.e., to go back to their initial TA locations).

In the translated network of timed automata *NTA*, we maintain a global flag boolean variable *hotviolated*, which indicates whether a hot violation has occurred in the LSC system. This variable is initialized to `false`, and it will be set to `true` whenever a hot violation occurs.

Dealing with time To mimic the behaviors of a clock constraint and clock reset in an LSC chart, we use a linked sequence of TA edges, whose atomicity is ensured

by the UPPAAL feature of committed location [4]. Upper bound constraints in the conditions can be extracted and used as TA location invariants to ensure that the constrained messages are sent out within the specified time frames; lower bound and clock difference constraints can be extracted and tested immediately after the message synchronizations.

Translating environment processes The processes (instance lines) in an LSC system can be partitioned into two sets: the environment processes ($\mathcal{E}nv$) and the system processes ($\mathcal{S}ys$). From the system's perspective, the messages sent from $\mathcal{E}nv$ to $\mathcal{S}ys$ processes are uncontrollable, whereas other message sendings are controllable. To model this edge controllability for the purpose of timed game solving, we mark the message-sending edges in the translated timed automata of the $\mathcal{E}nv$ processes as *uncontrollable* edges (in dashed lines), and other edges as *controllable* edges (in solid lines). In this way, we obtain the timed game automata [20] models.

Translating monitored chart In comparison with a driving universal chart, the translation of a monitored chart to timed automata is different in the point that a monitored chart only "listens to" the messages in the LSC system and never emits messages by itself. When translating such a chart into a network of timed automata, if at position s of instance line I_k there is a sending of an m-labeled message, then we add an $m?$-labeled TA edge from l_{s-1} to l_s, and not an $m!$-labeled one.

The translation rules, the detailed explanations and the translation examples can be found in Appendix B.

3.3 Complexity of translated timed automata

Let $\mathcal{LS}$ be a set of LSC charts $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n$, and let $NTA_{\mathcal{LS}}$ be the translated network of timed automata. Let $inst(\mathcal{L}_i)$, $ML(\mathcal{L}_i)$, $MA(\mathcal{L}_i)$ and $MO(\mathcal{L}_i)$ denote the set of instance lines, the set of message labels (i.e., "signals"), the message alphabet and the set of message occurrences of chart $\mathcal{L}_i$, respectively.

Table 1 summarizes the complexity of the outcomes of the translation in different settings, namely, a single LSC chart or an LSC system; untimed LSC chart or time-enriched LSC chart.

How we obtain these analysis results can be found in Appendix C.

3.4 Behavior equivalence of LSC and translated TAs

Theorem 1. *Let $\mathcal{LS}$ be a set of time-enriched LSC charts whose message alphabet is Π, and let $NTA_{\mathcal{LS}}$ be the translated network of timed automata which have*

[4] A *committed* location is an urgent location whose outgoing transitions have higher priority to be taken than those from non-committed ones.

Table 1. The complexity of the outcomes of LSC-to-TA translation

number of	A single chart $\mathcal{L}$	
	untimed chart	time-enriched chart
TAs	$\lvert inst(\mathcal{L})\rvert + 1$	$\lvert inst(\mathcal{L})\rvert + \lvert MA(\mathcal{L})\rvert + 1$
channels	$\lvert ML(\mathcal{L})\rvert + 2 \cdot \lvert inst(\mathcal{L})\rvert + 4$	$\lvert ML(\mathcal{L})\rvert + 2 \cdot \lvert inst(\mathcal{L})\rvert + 4 + 3 \cdot \lvert MA(\mathcal{L})\rvert$
auxiliary variables	$2 \cdot \lvert MA(\mathcal{L})\rvert + 2$	$4 \cdot \lvert MA(\mathcal{L})\rvert + 2 \cdot \lvert MO(\mathcal{L})\rvert + 2$
number of	A set of driving charts $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n$	
	untimed charts	time-enriched charts
TAs	$\sum_{i=1}^{n}(\lvert inst(\mathcal{L}_i)\rvert + 1)$	$\sum_{i=1}^{n}(\lvert inst(\mathcal{L}_i)\rvert + 1) + \lvert\bigcup_{i=1}^{n} MA(\mathcal{L}_i)\rvert$
channels	$\lvert\bigcup_{i=1}^{n} ML(\mathcal{L}_i)\rvert + \sum_{i=1}^{n}(2 \cdot \lvert inst(\mathcal{L}_i)\rvert + 4)$	$\lvert\bigcup_{i=1}^{n} ML(\mathcal{L}_i)\rvert + \sum_{i=1}^{n}(2 \cdot \lvert inst(\mathcal{L}_i)\rvert + 4) + 3 \cdot \lvert\bigcup_{i=1}^{n} MA(\mathcal{L}_i)\rvert$
auxiliary variables	$2 \cdot \lvert\bigcup_{i=1}^{n} MA(\mathcal{L}_i)\rvert + 2n$	$4 \cdot \lvert\bigcup_{i=1}^{n} MA(\mathcal{L}_i)\rvert + 2 \cdot \sum_{i=1}^{n} \lvert MO(\mathcal{L}_i)\rvert + 2n$

a set Act of normal and auxiliary channels. Then $\forall \gamma_1 \in (\Pi \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^\omega \,.\, ((\gamma_1 \models \mathcal{LS}) \Rightarrow \exists \gamma_2 \in (Act \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^\omega \,.\, (\gamma_2 \models NTA_{\mathcal{LS}}) \wedge (\gamma_2|_{(\Pi \cup \mathbb{R}_{\geq 0})} = \gamma_1|_{(\Pi \cup \mathbb{R}_{\geq 0})}))$, *and* $\forall \gamma_2 \in (Act \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^\omega \,.\, ((\gamma_2 \models NTA_{\mathcal{LS}}) \Rightarrow \exists! \gamma_1 \in (\Pi \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^\omega \,.\, (\gamma_1 \models \mathcal{LS}) \wedge (\gamma_2|_{(\Pi \cup \mathbb{R}_{\geq 0})} = \gamma_1|_{(\Pi \cup \mathbb{R}_{\geq 0})}))$.

Theorem 1 indicates that each accepted timed trace γ_1 in $\mathcal{LS}$ uniquely corresponds to a cluster of accepted timed traces in $NTA_{\mathcal{LS}}$. All these traces project to exactly the same trace on the message alphabet and time delays $(\Pi \cup \mathbb{R}_{\geq 0})$ as γ_1 does.

The lemmas for theorems in Sections 3.4 and 4, and the proofs of them can be found in Appendix D.

4 Analysis and synthesis problems

4.1 Consistency checking

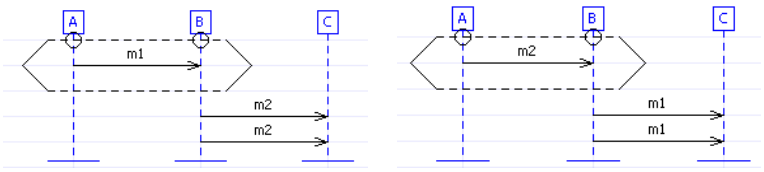
An LSC system is *inconsistent* if and only if the system model has internal contradictions [10], i.e., there does not exist an infinite message sequence such that it satisfies all driving universal LSC charts. Alternatively, an LSC system is inconsistent iff, along all possible paths there will eventually be a hot violation of the main chart of a certain LSC chart (i.e., the flag boolean variable *hotviolated*, which has been initialized to `false`, will eventually be set `true` in the translated network of timed automata).

Uppaal uses a fragment of the CTL logic as its query language. Formulas could take the forms $\mathsf{E}\Diamond\phi$, $\mathsf{E}\Box\phi$, $\mathsf{A}\Diamond\phi$, $\mathsf{A}\Box\phi$, $\phi_1 \rightsquigarrow \phi_2$, where ϕ, ϕ_1 and ϕ_2 are state formulas. In particular $\phi_1 \rightsquigarrow \phi_2$ ("ϕ_1 leads-to ϕ_2") is a shorthand for $\mathsf{A}\Box(\phi_1 \Rightarrow \mathsf{A}\Diamond\phi_2)$, which characterizes the assume-guarantee style liveness (or responsiveness).

Theorem 2. *$\mathcal{LS} = \{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n\}$ are inconsistent* $\Leftrightarrow$ $NTA_{\mathcal{LS}} \models$ $\mathtt{true} \rightsquigarrow (\mathit{hotviolated} == \mathtt{true})$.

Theorem 2 indicates that in order to check the inconsistency of a set of driving LSC charts, we can instead check whether it is true that the translated network of timed automata will eventually have its boolean flag variable *hotviolated* set to `true`.

For example, by model checking the corresponding translated network of timed automata, we find out that the two driving universal LSC charts in Fig. 1 are consistent, whereas the two charts in Fig. 3 are inconsistent.



(a) chart 1 (b) chart 2

Fig. 3. Two universal charts which are inconsistent.

4.2 Property verification

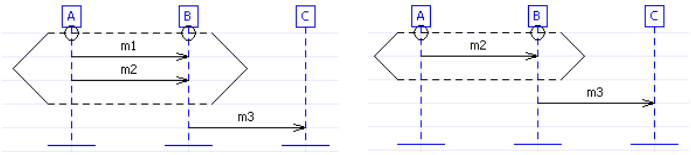
Property verification asks whether a system that is modeled as a set of driving universal LSC charts $\mathcal{LS}$ satisfies the requirement that is specified as a separate monitored universal or existential chart $\mathcal{L}'$. Here $\mathcal{L}'$ will be translated into a network of "observer" timed automata $NTA_{\mathcal{L}'}$, i.e., they only "listen to" the messages in the network of timed automata $NTA_{\mathcal{LS}}$ for $\mathcal{LS}$, and never emit messages by themselves.

Theorem 3. *Let $\mathcal{LS}$ be an LSC system, and $\mathcal{L}'$ be a monitored universal chart.* $\mathcal{LS} \models \mathcal{L}' \Leftrightarrow (NTA_{\mathcal{LS}} \,||\, NTA_{\mathcal{L}'}) \models (\mathit{Coord}_{\mathcal{L}'}.\mathit{Mch\_top} \rightsquigarrow \mathit{Coord}_{\mathcal{L}'}.\mathit{Mch\_bot})$.

In the above theorem, $\mathit{Coord}_{\mathcal{L}'}.\mathit{Mch\_top}$ and $\mathit{Coord}_{\mathcal{L}'}.\mathit{Mch\_bot}$ say that the coordinator timed automaton $\mathit{Coord}_{\mathcal{L}'}$ for chart $\mathcal{L}'$ is in its locations *Mch_top* and *Mch_bot*, respectively.

Theorem 3 indicates that in order to check whether a system $\mathcal{LS}$ satisfies the requirement in a universal chart $\mathcal{L}'$, we only need to check whether the paralelly composed translated network of timed automata satisfy the aforementioned responsiveness property.

For example, after model checking the corresponding translated network of timed automata, we find out that the (single chart) LSC system in Fig. 4(a) satisfies the requirement that is specified in Fig. 4(b).



(a) chart 1 (b) chart 2

Fig. 4. Chart 1 (model) satisfies chart 2 (property).

Theorem 4. *Let $\mathcal{LS}$ be an LSC system, and $\mathcal{L}'$ be a monitored existential chart.* $\mathcal{LS} \models \mathcal{L}' \Leftrightarrow (NTA_{\mathcal{LS}} \,||\, NTA_{\mathcal{L}'}) \models (\mathsf{E}\Diamond\ Coord_{\mathcal{L}'}.Mch\_bot)$.

Theorem 4 indicates that in order to check whether a system $\mathcal{LS}$ satisfies the requirement in an existential chart $\mathcal{L}'$, we only need to check whether $\mathcal{LS}$ have a trace that can be observed by $\mathcal{L}'$ as a satisfying run.

4.3 Centralized synthesis for open systems

A timed automaton with its edges partitioned into controllable and uncontrollable ones is called a *timed game automaton* (TGA) [20]. A network of parallelly composed timed game automata for $\mathcal{E}nv$ and $\mathcal{S}ys$ can be viewed as a timed game structure: as a player, the timed game automata for $\mathcal{S}ys$ (noted $NTA_{\mathcal{S}ys}$) master the set A_c of controllable edges; as the opponent, the timed game automata for $\mathcal{E}nv$ (noted $NTA_{\mathcal{E}nv}$) master the set A_u of uncontrollable edges. Given a winning objective, $NTA_{\mathcal{S}ys}$ will take moves in order to win the game (i.e., to bring the system into a winning state, or to prevent the system from entering a losing state), whereas $NTA_{\mathcal{E}nv}$ may spoil the game.

Let S be the state space of $NTA_{\mathcal{E}nv} \,||\, NTA_{\mathcal{S}ys}$, and $\epsilon \notin (A_c \cup A_u \cup \{\tau\})$ be an empty action which means "do nothing at this moment in time". A state-based (or memoryless) strategy for $NTA_{\mathcal{S}ys}$ is a (partial) function

$$\rho : S \to (A_c \cup \{\epsilon\}),$$

which constantly guides the timed automata in $NTA_{\mathcal{S}ys}$ to take appropriate controllable actions, or just delay (and wait for the semantic state to be changed by an uncontrollable action of $NTA_{\mathcal{E}nv}$, or by the elapse of time).

Uppaal-Tiga [2] is a timed game solver. Its inputs include a set of timed game automata, and a winning objective that is formulated as an extended ACTL (the universal fragment of CTL) formula. For example, property "control: $\mathsf{A}\Box\ \varphi$" asks whether there exists a strategy ρ for $NTA_{\mathcal{S}ys}$ such that if $NTA_{\mathcal{S}ys}$ is supervised (or "restricted", "guided") by ρ, then the system $NTA_{\mathcal{LS}}$ is guaranteed to always (i.e. invariantly) satisfy φ. If the property is satisfied, then Uppaal-Tiga will be able to synthesize a winning strategy for $NTA_{\mathcal{S}ys}$.

Synthesis for $\mathcal{S}ys$ is possible only if the entire system $\mathcal{LS}$ can be guaranteed not to be hot-violated no matter how the $\mathcal{E}nv$ processes behave. By means of

trace-wise behavior equivalence, this boils down to finding a winning strategy ρ for $NTA_{\mathcal{S}ys}$. Since the strategy (if ever exists) will oversee all $\mathcal{S}ys$ processes rather than being distributed to supervise each individual $\mathcal{S}ys$ process, it is a kind of *centralized* synthesis. It is clear that $NTA_{\mathcal{LS}}$ as supervised by ρ constitute one such desired executable (state/transition-based) object system.

Theorem 5. *An executable object system for $\mathcal{S}ys$ can be synthesized* $\Leftrightarrow NTA_{\mathcal{LS}} \models$ (control: A$\Box$ ($hotviolated == \texttt{false}$)).

Theorem 5 indicates that the problem of centralized synthesis for open systems can be reduced to a timed game solving problem in UPPAAL-TIGA.

5 Experiments

Fig. 5 shows our scenario-based analysis and synthesis framework. Among those inputs (the shaded elements), the $\mathcal{E}nv/\mathcal{S}ys$ partitioning directives specify which processes in the charts of $\mathcal{LS}$ belong to $\mathcal{E}nv$ and which belong to $\mathcal{S}ys$, respectively.

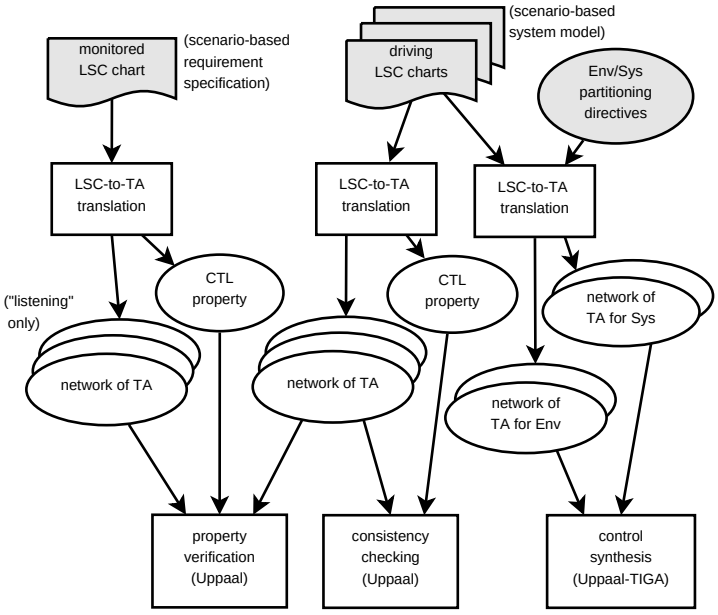


Fig. 5. Scenario-based analysis and synthesis framework.

We build a GUI-based LSC editor, with which we can construct either universal or existential charts. A prototype command line LSC-to-TA translator has been implemented, which is capable of batch translation of monitored and driving charts. The translator-generated timed automata and CTL formulas comply

with the UPPAAL timed automaton, UPPAAL-TIGA timed game automaton and their query language syntaxes, and can thus be fed into UPPAAL and UPPAAL-TIGA directly.

Preliminary translation experiments have been conducted on an Intelligent Mouse (# of charts, instance lines, message labels, message occurrences, clocks: 4, 3, 3, 12, 2), a refrigeration (4, 3, 3, 14, 1) and an ATM Machine (12, 3, 6, 21, 0) examples, and a DHCP (Dynamic Host Configuration Protocol) (34, 3, 17, 44, 0) case studies. Some comparative experiments reveal that the translation has negligible time overheads and memory consumptions than the subsequent model checking and game solving of the translated network of timed automata. This is reasonable, because the complexity of scenario-based analysis and synthesis mainly lies in the LSC models themselves. As a syntactical level manipulation, the translation only introduces some auxiliary channels and bookkeeping variables (not clock variables) to implement the LSC semantics.

6 Conclusions

We present timed extensions to a kernel subset of the LSC language and define a trace-based semantics. We show how to transform LSC charts into a network of behavior-equivalent timed automata. The LSC consistency checking, property verification and synthesis problems can be reduced to CTL real-time model checking and timed game solving problems. We implement a prototype LSC-to-TA translator. When linked with our LSC editor and the existing real-time model checker UPPAAL and timed game solver UPPAAL-TIGA, they constitute a tool chain for automated, scenario-based analysis and synthesis of real-time systems. Preliminary experiments on a number of examples show that it is a viable approach.

Scenario-based approaches enjoy the advantages of incremental construction of models, i.e., new pieces of scenarios can be added into existing ones during the development process. In order to keep the driving LSC charts and their complexity within human readable and manageable levels, a single LSC chart needs not to be very large and complex. Rather the complexity of scenario-based models mainly lies in the interplays of a large number of relatively simple charts. Our "one-TA-per-instance line" translation is in accordance with this philosophy. Instead of constructing a complex global state machine that handles all possible activities explicitly, we leave the intricate semantics of LSC chart progress and intra/inter-chart coordinations mostly up to UPPAAL.

As future work, we may consider the translation of more LSC constructs into timed automata, such as co-region, symbolic instance, control structures, etc. Other chart activation modes also need to be dealt with. The implementation of a full-fledged translator and the application of the tool chain to industrial case studies are desirable. Furthermore, scenario-based synthesis for systems with imperfect information (e.g., some uncontrollable actions are not observable) is also worth investigation.

Acknowledgements We would like to thank Itai Segall for helpful discussions and for pointing out an error in the DATE'10 conference version of this paper.

References

1. Rajeev Alur and David L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.
2. Gerd Behrmann, Agnès Cougnard, Alexandre David, Emmanuel Fleury, Kim Guldstrand Larsen, and Didier Lime. UPPAAL-TIGA: Time for playing games! In *Proc. 19th International Conference on Computer Aided Verification (CAV'07)*, pages 121–125, 2007.
3. Gerd Behrmann, Alexandre David, and Kim Guldstrand Larsen. A tutorial on uppaal. In Marco Bernardo and Flavio Corradini, editors, *SFM*, volume 3185 of *Lecture Notes in Computer Science*, pages 200–236. Springer, 2004.
4. Yves Bontemps. *Relating Inter-Agent and Intra-Agent Specifications - The Case of Live Sequence Charts*. PhD thesis, University of Namur, Namur, Belgium, 2005.
5. Yves Bontemps and Pierre-Yves Schobbens. The computational complexity of scenario-based agent verification and design. *J. Applied Logic*, 5(2):252–276, 2007.
6. Pierre Combes, David Harel, and Hillel Kugler. Modeling and verification of a telecommunication application using live sequence charts and the play-engine tool. *Software and System Modeling*, 7(2):157–175, 2008.
7. Werner Damm and David Harel. LSCs: Breathing life into message sequence charts. *Formal Methods in System Design*, 19(1):45–80, 2001. Preliminary version in *Proc. 3rd IFIP Int. Conf. on Formal Methods for Open Object-Based Distributed Systems/(FMOODS'99)*, P. Ciancarini, A. Fantechi and R. Gorrieri, eds., Kluwer Academic Publishers, 1999, pp. 293-312.
8. Werner Damm and Jochen Klose. Verification of a radio-based signaling system using the statemate verification environment. *Formal Methods in System Design*, 19(2):121–141, 2001.
9. David Harel and Hillel Kugler. Synthesizing state-based object systems from lsc specifications. In *Proc. 5th International Conference on Implementation and Application of Automata (CIAA'00)*, pages 1–33, 2000.
10. David Harel and Hillel Kugler. Synthesizing state-based object systems from lsc specifications. *Int. J. Found. Comput. Sci.*, 13(1):5–51, 2002.
11. David Harel, Hillel Kugler, Rami Marelly, and Amir Pnueli. Smart play-out of behavioral requirements. In *Proc. 4th International Conference on Formal Methods in Computer-Aided Design (FMCAD'02)*, pages 378–398, 2002.
12. David Harel, Hillel Kugler, and Amir Pnueli. Smart play-out extended: Time and forbidden elements. In *Proc. 4th International Conference on Quality Software (QSIC'04)*, pages 2–10, 2004.
13. David Harel, Hillel Kugler, and Amir Pnueli. Synthesis revisited: Generating statechart models from scenario-based requirements. In *Proc. Formal Methods in Software and Systems Modeling*, pages 309–324, 2005.
14. David Harel and Rami Marelly. *Come, Let's Play: Scenario-Based Programming Using LSC's and the Play-Engine*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.
15. ITU-T. Message sequence charts – msc-2000, itu-t recommendation z.120, 1999.
16. Jochen Klose, Tobe Toben, Bernd Westphal, and Hartmut Wittke. Check it out: On the efficient formal verification of live sequence charts. In *Proc. 18th International Conference on Computer Aided Verification (CAV'06)*, pages 219–233, 2006.

17. Jochen Klose and Hartmut Wittke. An automata based interpretation of live sequence charts. In *Proc. 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'01)*, pages 512–527, 2001.
18. Hillel Kugler, Cory Plock, and Amir Pnueli. Controller synthesis from lsc requirements. In *Proc. 12th International Conference on Fundamental Approaches to Software Engineering (FASE'09)*, pages 79–93, 2009.
19. Kim Guldstrand Larsen, Shuhao Li, Brian Nielsen, and Saulius Pusinskas. Verifying real-time systems against scenario-based requirements. In *Proc. 16th International Symposium on Formal Methods (FM'09)*, pages 676–691, 2009.
20. Oded Maler, Amir Pnueli, and Joseph Sifakis. On the synthesis of discrete controllers for timed systems (an extended abstract). In *Proc. 12th Annual Symposium on Theoretical Aspects of Computer Science (STACS'95)*, pages 229–242, 1995.
21. K. V. S. Prasad. A calculus of broadcasting systems. *Sci. Comput. Program.*, 25(2-3):285–327, 1995.
22. Jun Sun and Jin Song Dong. Model checking live sequence charts. In *Proc. 10th International Conference on Engineering of Complex Computer Systems (ICECCS'05)*, pages 529–538, 2005.
23. Jun Sun and Jin Song Dong. Synthesis of distributed processes from scenario-based specifications. In John Fitzgerald, Ian J. Hayes, and Andrzej Tarlecki, editors, *FM*, volume 3582 of *Lecture Notes in Computer Science*, pages 415–431. Springer, 2005.

Appendix A: Timed automata in Uppaal

We use the following notations: C is a set of real-valued clocks, and $B(C)$ is the set of conjunctions over simple conditions of the form $x \bowtie c$ or $x - y \bowtie c$, where $x, y \in C$, $c \in \mathbb{N}$, and $\bowtie \in \{<, \leq, =, \geq, >\}$.

Definition 7 (timed automaton, TA [3]). *A* timed automaton *is a tuple* (L, l_0, C, Act, E, Inv)*, where* L *is a set of locations,* $l_0 \in L$ *is the initial location,* C *is a set of clocks,* Act *is the alphabet of actions,* $E \subseteq L \times (Act \cup \{\tau\}) \times B(C) \times 2^C \times L$ *is a set of edges between locations, each of which has an action, a guard and a set of clocks to be reset, and* $Inv : L \to B(C)$ *assigns invariants to locations.* □

Uppaal has defined a number of extensions to the standard notations of timed automata. Specifically, an *urgent* location is such a TA location that freezes time, i.e., time is not allowed to elapse when a process is in an urgent location. A *committed* location is a special kind of urgent location whose outgoing transitions always have higher priority to be fired than those from non-committed locations.

Uppaal uses a mixture of handshake communication and broadcast communication. The CBS (Calculus of Broadcasting Systems [21])-style broadcast channels allow 1-to-many synchronization. If the emitting edge is enabled, then

it can always fire. If the emitting edge is fired, then all enabled receiving edges (might be 0 edge) will synchronize.

In UPPAAL an *urgent* channel means that if it is possible to trigger a synchronization over that channel, then it cannot delay in the source state.

Furthermore, UPPAAL also supports bounded-range integer and boolean data variables, which can be used in the guards, assignment and location invariants.

A clock valuation is a function $u : C \to \mathbb{R}_{\geq 0}$ from the set of clocks to the non-negative real numbers. Let ${\mathbb{R}_{\geq 0}}^C$ be the set of all clock valuations. Let $u_0(x) = 0$ for all $x \in C$. We will abuse the notation by considering guards and invariants as sets of clock valuations, writing $u \in Inv(l)$ to mean that valuation u satisfies $Inv(l)$.

Definition 8 (semantics of TA [3]). *Let (L, l_0, C, Act, E, Inv) be a timed automaton. The* semantics *is defined as a labeled transition system $\langle S, s_0, \to \rangle$, where $S \subseteq L \times {\mathbb{R}_{\geq 0}}^C$ is the set of states, $s_0 = (l_0, u_0)$ the initial state, and $\to \subseteq S \times (Act \cup \{\tau\} \cup \mathbb{R}_{\geq 0}) \times S$ the transition relation such that:*

- $(l, u) \xrightarrow{d} (l, u + d)$ *if* $\forall d' : 0 \leq d' \leq d \,.\, u + d' \in Inv(l)$*; and*
- $(l, u) \xrightarrow{a} (l', u')$ *if there exists* $e = (l, a, g, r, l') \in E$ *such that* $u \in g$, $u' = [r \to 0]u$, *and* $u' \in Inv(l')$,

where for $d \in \mathbb{R}_{\geq 0}$, $u + d$ maps each clock x in C to the value $u(x) + d$, and $[r \to 0]u$ denotes the clock valuation which maps each clock in r to 0 and agrees with u over $C \backslash r$. □

Definition 9 (run of TA). *A* run *of a TA (L, l_0, C, Act, E, Inv) is a sequence of states $s^0 \cdot s^1 \cdot \ldots$ that are connected by the transitions, i.e., $\forall i \geq 0 \,.\, \exists u_i \in (Act \cup \{\tau\} \cup \mathbb{R}_{\geq 0}) \,.\, s^i \xrightarrow{u_i} s^{i+1}$.* □

The transition relation $\to$ as mentioned above each time consumes only a single letter $u \in (Act \cup \{\tau\} \cup \mathbb{R}_{\geq 0})$. We extend it to $\to^*$ such that it consumes a (finite or infinite) word $w \in (Act \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^* \cup (Act \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^\omega$. A word w that corresponds to a run of the TA is called a *timed trace* of the TA.

A number of timed automata can be parallelly composed into a network of timed automata over a common set of clocks and actions, $A_i = (L_i, l_{0,i}, C, Act, E_i, Inv_i)$, $1 \leq i \leq n$. A location vector $\bar{l} = (l_1, \ldots, l_n)$ is a vector of locations of the member TA. We compose the invariant functions into a common function over location vectors $Inv(\bar{l}) = \bigwedge_i Inv_i(l_i)$. We write $\bar{l}[l'_i/l_i]$ to denote the vector where the i-th element l_i of $\bar{l}$ is replaced by l'_i.

Definition 10 (semantics of a network of TAs [3]). *Let $A_i = (L_i, l_{0,i}, C, Act, E_i, Inv_i)$ be a network of timed automata, $1 \leq i \leq n$. Let $\bar{l_0} = (l_{0,1}, \ldots, l_{0,n})$ be the initial location vector. The* semantics *is defined as a transition system $\langle S, s_0, \to \rangle$, where $S = (L_1 \times \ldots \times L_n) \times {\mathbb{R}_{\geq 0}}^C$ is the set of global states, $s_0 = (\bar{l_0}, u_0)$ the initial global state, and $\to \subseteq S \times (Act \cup \{\tau\} \cup \mathbb{R}_{\geq 0}) \times S$ the transition relation defined by:*

- $(\bar{l}, u) \xrightarrow{d} (\bar{l}, u + d)$ *if* $\forall d' : 0 \leq d' \leq d \,.\, u + d' \in Inv(\bar{l})$*;*

– $(\bar{l}, u) \xrightarrow{\tau} (\bar{l}[l'_i/l_i], u')$ *if there exists* $l_i \xrightarrow{\tau, g, r} l'_i$ *such that* $u \in g$, $u' = [r \to 0]u$ *and* $u' \in Inv(\bar{l}[l'_i/l_i])$;
– $(\bar{l}, u) \xrightarrow{a} (\bar{l}[l'_i/l_i, l'_j/l_j], u')$ *if* a *is a binary channel and there exist* $l_i \xrightarrow{c!, g_i, r_i} l'_i$ *and* $l_j \xrightarrow{c?, g_j, r_j} l'_j$ *such that* $u \in (g_i \wedge g_j)$, $u' = [r_i \cup r_j \to 0]u$ *and* $u' \in Inv(\bar{l}[l'_i/l_i, l'_j/l_j])$; *and*
– $(\bar{l}, u) \xrightarrow{a} (\bar{l}[l'_i/l_i, l'_j/l_j, l'_k/l_k, \ldots], u')$ *if* a *is a broadcast channel and there exist an* $l_i \xrightarrow{c!, g_i, r_i} l'_i$ *and a maximal set* $\{j, k, \ldots\}$: $l_j \xrightarrow{c?, g_j, r_j} l'_j$, $l_k \xrightarrow{c?, g_k, r_k} l'_k$, ..., *such that* $u \in (g_i \wedge g_j \wedge g_k \wedge \ldots)$, $u' = [r_i \cup r_j \cup r_k \cup \ldots \to 0]u$ *and* $u' \in Inv(\bar{l}[l'_i/l_i, l'_j/l_j, l'_k/l_k, \ldots])$. □

Runs and traces of a network of TAs are defined similarly as those for a single TA.

Appendix B: Rules for LSC-to-TA translation

B1. Translating message-only charts

As mentioned in Section 2.1, along each instance line I_i in chart $\mathcal{L}$, there are a set $pos(\mathcal{L}, I_i)$ of positions, among which there are a set $StdPos(\mathcal{L}, I_i) \subset pos(\mathcal{L}, I_i)$ of four "standard" positions. For example in instance line A of Fig. 1(a), there are 7 positions (black filled circles), where the four standard ones are *Pch_top* (0), *Pch_bot* (3), *Mch_top* (4) and *Mch_bot* (6).

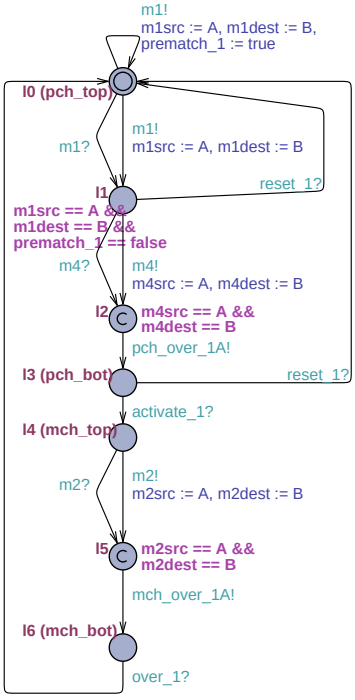
Given $I_i \in inst(\mathcal{L})$, $p \in StdPos(\mathcal{L}, I_i)$, we use $\mathcal{L}.I_i.p$ to denote the position ID of standard position p on instance line I_i of chart $\mathcal{L}$. For example in Fig. 1(a), we have $\mathcal{L}.A.Pch\_bot = 3$.

Fig. 6 shows the translated network of timed automata for the chart $\mathcal{L}_1$ of Fig. 1(a).

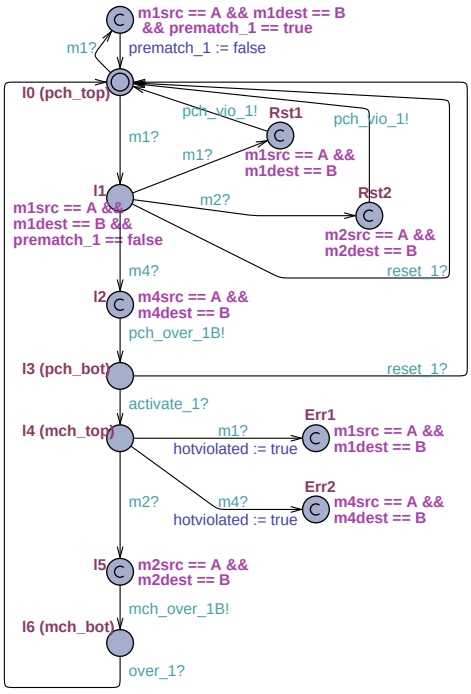
(1) Basic mapping rules

Let $\mathcal{LS}$ be an LSC system, $\mathcal{L}_u$ be a chart in $\mathcal{LS}$, and I_i be an instance line in $\mathcal{L}_u$. We map each such I_i to a timed automaton $A_{u,i}$ using the following rules:

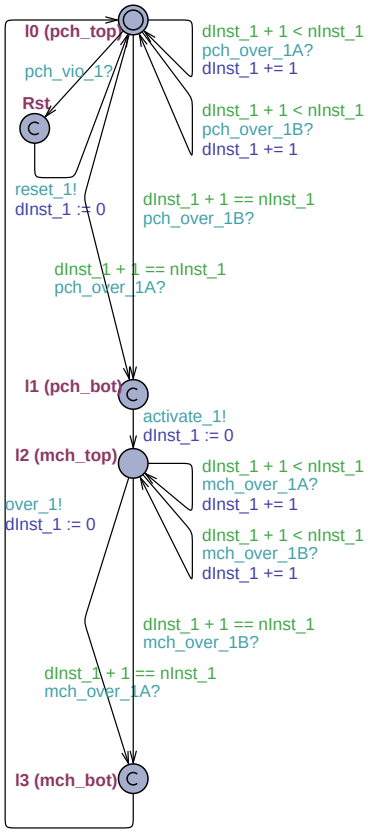
R1 Each position k on I_i of $\mathcal{L}_u$ corresponds to a TA location l_k in $A_{u,i}$, $0 \le k \le p\_max_{\mathcal{L}_u, I_i}$. See Fig. 6(a), locations l_0 - l_6.
R2 If at position k on I_i of $\mathcal{L}_u$ there is a sending of an m-labeled message to instance I_j, then there will be assigned an $m!$-labeled TA edge from location l_{k-1} to l_k in $A_{u,i}$. See Fig. 6(a), straight line edges $(l_0, l_1), (l_1, l_2), (l_4, l_5)$.
R3 If at position k on I_i of $\mathcal{L}_u$ there is a reception of an m-labeled message from instance I_j, then there will be assigned an $m?$-labeled TA edge from location l_{k-1} to l_k in $A_{u,i}$. See Fig. 6(b), straight line edges $(l_0, l_1), (l_1, l_2), (l_4, l_5)$.



(a) TA for instance A



(b) TA for instance B



(c) The coordinator TA

Fig. 6. Translated TAs for the untimed chart $\mathcal{L}_1$ of Fig. 1(a)

We abuse the notations *Pch_top*, *Pch_bot*, *Mch_top* and *Mch_bot* to also denote the TA locations that correspond to these LSC positions. For any position k other than the aforementioned four, it corresponds to a TA location l_k, meaning that upon sending/receiving the message that anchors at position k, we now arrive at l_k. Furthermore, position 0 (i.e., *Pch_top*) corresponds to the initial TA location l_0 (i.e., *Pch_top*).

When R2 is applied, the TA edge can be associated with an assignment "$m\_src := I_i, m\_dest := I_j$", where m_src and m_dest are fresh auxiliary (bounded integer) variables, meaning that an m-labeled message is sent from instance I_i to I_j in chart $\mathcal{L}_u$. In R2 and R3, the destination location l_k will have invariant "$(m\_src == I_i) \wedge (m\_dest == I_j)$" and "$(m\_src == I_j) \wedge (m\_dest == I_i)$", respectively. See Fig. 6(a), locations l_1, l_2, l_5, and Fig. 6(b), l_1, l_2, l_5.

(2) Handling intra-chart coordinations

In an LSC chart, if an instance line (process) in its prechart portion has no more interactions with the other instance lines (e.g., it has successfully sent/received the last message, or it has no interactions with other instance lines at all), then it will *immediately* progress to the bottom position *Pch_bot* of its prechart portion, to be ready for a next mandatory synchronization that involves all the instance lines in that chart.

R4 At position k on the prechart portion of I_i of $\mathcal{L}_u$, if $k = \mathcal{L}_u.I_i.Pch\_bot - 1$, then we mark l_k as a committed location in $A_{u,i}$, and we add a $pch\_over_{u,i}$!-labeled edge from l_k to l_{k+1} in $A_{u,i}$. See Fig. 6(a), location l_2.

The auxiliary channel $pch\_over_{u,i}$ (meaning "prechart portion is over") is used to notify the coordinator automaton $Coord_u$ (explained below) of the completion of instance line I_i with its prechart portion in chart $\mathcal{L}_u$.

When all the instance lines in chart $\mathcal{L}_u$ progress to their respective *Pch_bot* positions, the prechart is now successfully matched. Once this happens, all these instance lines must immediately synchronize and progress to their respective *Mch_top* positions, meaning that the main chart is now activated. To model this kind of *intra-chart* coordination at the prechart/main chart interface, for each chart $\mathcal{L}_u$, we create a dedicated (auxiliary) coordinator automaton $Coord_u$. This automaton will communicate with the automata that correspond to the instance lines of $\mathcal{L}_u$ by using auxiliary binary channels such that it can bookkeep how many instance lines are done with their prechart portions. Once the coordinator automaton realizes that the prechart has been successfully matched, it will immediately launch a broadcast synchronization with the automata that correspond to the instance lines.

Fig. 6(c) gives an example of the coordinator timed automaton for chart $\mathcal{L}_1$ of Fig. 1(a), where $pch\_over_{1,A}$ and $pch\_over_{1,B}$ are binary channels, $activate_1$ is a broadcast channel (meaning that the main chart is to be activated), $nInst_1$ is a constant that denotes the number of instance lines that participate in chart $\mathcal{L}_1$, and $dInst_1$ is an integer variable that denotes the number of instance lines that are done with their prechart (or main chart) portions of $\mathcal{L}_1$.

The coordinator TA synchronizes with the timed automata that correspond to the instance lines in the prechart/main chart according to the following rule:

R5 At position k of I_i of $\mathcal{L}_u$, if $k = \mathcal{L}_u.I_i.Pch\_bot$, then there will be assigned an $activate_u$?-labeled TA edge from l_k to l_{k+1} in $A_{u,i}$. See Fig. 6(a), l_3, and Fig. 6(b), l_3.

Similarly, intra-chart coordination upon main chart completion will correspond to the channels $mch\_over_{u,i}$ and $over_u$ (meaning that the main chart has been successfully matched).

(3) Handling inter-chart coordinations

In scenario-based modeling, the same message may well appear in two or more charts. For example given an LSC system $\mathcal{LS}$, in chart $\mathcal{L}_1$ there is an m-labeled message occurrence mo_1 from instance I_1 to I_2, and in chart $\mathcal{L}_2$ there is an m-labeled message occurrence mo_2, also from I_1 to I_2. If at a certain global configuration $(\bar{c}, v)$ these message occurrences (in the above example mo_1 and mo_2) are all enabled, then their firings should be synchronized, i.e., either all of them are chosen to be fired, or none of them is chosen. This is considered a kind of *inter-chart* coordination.

In the translated network of timed automata, this can be accomplished by using a broadcast synchronization. Recall that in a broadcast synchronization, there is only one sender. Therefore, when translating the message occurrences (in the above example mo_1 and mo_2) to edges in their respective timed automata, only one of the LSC positions that are associated with the message tails (i.e., sending locations) in $\mathcal{LS}$ can correspond to the TA location that has the sole outgoing message-emitting TA edge in the translated TAs, and all others will correspond to TA locations that have outgoing message-receiving TA edges. Since all message-sending instance lines in the relevant charts should have the equal possibility to initiate the message synchronization, we consider a universal and symmetric solution: for each m!-labeled edge from one to another TA locations, we add an m?-labeled edge between these two TA locations to "accompany" the m!-labeled edge. In other words, we let all translated TA locations that correspond to the message-sending locations (in the above example two locations in the translated TAs $A_{1,1}$ and $A_{2,1}$) have the equal chance to act also as the broadcast synchronization initiator.

R6 If at position k on I_i of $\mathcal{L}_u$ there is a sending of an m-labeled message, then there will be added an m?-labeled TA edge from l_{k-1} to l_k in $A_{u,i}$. In the translated TAs, m will be changed from a binary to a broadcast channel. See Fig. 6(a), polyline edges (l_0, l_1), (l_1, l_2), (l_4, l_5).

(4) Handling cold and hot violations

Along an instance line of an LSC chart, if an arriving message is not enabled at the current cut in the prechart, then there will be a cold violation.

In this case, all participating instance lines in this chart should be reset (i.e., brought back to their initial positions) immediately. In our translation, this is implemented by letting the timed automaton that corresponds to the message receiving instance line "report" the cold violation to the coordinator automaton in charge, which in turn immediately initiates a broadcast synchronization to ask the timed automata that correspond to all other instance lines of the chart to do a reset.

R7 Assume that at position k on the prechart portion of instance line I_i of chart $\mathcal{L}_u$, there is a reception of an m-labeled message from instance I_j. If $k \geq \mathcal{L}_u.I_i.Pch\_top + 2$, then for all m'-labeled message in $\mathcal{L}_u$ such that $m' \neq m$ (note that $m, m' \in \Pi$), there will be added first an m'?-labeled outgoing TA edge from l_{k-1}, then a fresh intermediate committed TA location with invariant $m'\_src == src(m') \wedge m'\_dest == dest(m')$, and then a pch_vio_u!-labeled TA edge that leads to l_0 in $A_{u,i}$. See Fig. 6(b), TA location `Rst1`, and TA edges (l_1, `Rst1`), (`Rst1`, l_0).

In the above rule, the auxiliary binary channel pch_vio_u (meaning "prechart violation" of chart $\mathcal{L}_u$) is used to notify the coordinator TA $Coord_u$ of the cold violation. The $reset_u$!-labeled broadcast edge will be added in $Coord_u$. See Fig. 6(c), TA edges (l_0, `Rst`), (`Rst`, l_0). In the prechart of $\mathcal{L}_u$, for all positions s on all instance lines I_t such that $\mathcal{L}_u.I_t.Pch\_top + 1 \leq s \leq \mathcal{L}_u.I_t.Pch\_bot$, we add a $reset_u$?-labeled edge from l_s to l_0 in $A_{u,t}$. See Fig. 6(a), TA edges (l_1, l_0), (l_3, l_0).

If a message violates the partial order in the main chart, then it is a hot violation. Once this happens, the corresponding TA will immediately go to a deadend error location (`Err`).

R8 If at position k on the main chart portion of instance line I_i of chart $\mathcal{L}_u$, there is a reception of an m-labeled message from instance I_j, then for all m'-labeled message in $\mathcal{L}_u$ such that $m' \neq m$, there will be added to location l_{k-1} an m'?-labeled outgoing TA edge, which arrives at a deadend error location. See Fig. 6(b), locations `Err1`, `Err2` and edges (l_4, `Err1`), (l_4, `Err2`).

(5) Prechart pre-matching

According to the semantics for invariant mode LSC chart, minimal events in the prechart are constantly being matched for. For example in Fig. 1(a), $m_1 \cdot m_1 \cdot m_4 \cdot m_2$ is a matching sequence for the second incarnation of chart $\mathcal{L}_1$ under the invariant mode.

R9 If at position 1 on the prechart portion of instance line I_i of chart $\mathcal{L}_u$, there is a sending of an m-labeled message to instance line I_j at its position 1, then there will be added to location l_0 of $A_{u,i}$ an m!-labeled self loop edge with assignment "$m\_src := I_i, m\_dest := I_j, prematch_u :=$ `true`". If location l_0 in $A_{u,i}$ has an invariant, then it will be enhanced with a further constraint "$prematch_u ==$ `false`". See Fig. 6(a), location l_0.

Similarly, if there is a reception of an m-labeled message from I_j, then we add to l_0 an $m?$-labeled edge, followed by an intermediate committed location which has invariant "$(m\_src == I_j) \wedge (m\_dest == I_i) \wedge (prematch_u == \texttt{true})$", and then an internal transition edge with assignment "$prematch_u := \texttt{false}$" leading back to l_0. See Fig. 6(b).

The flag boolean variable $prematch_u$ is initialized to `false`. Once it is set `true`, it means that chart $\mathcal{L}_u$ is currently undergoing a process of prechart pre-matching.

For simplicity, the semantics of prechart pre-matching has not been considered in Section 2.1. A remedy to this is to add one more bullet to the "silent step" case, stating that an m-consuming advancement step will just remain at the top cut $\top$.

B2. Dealing with time

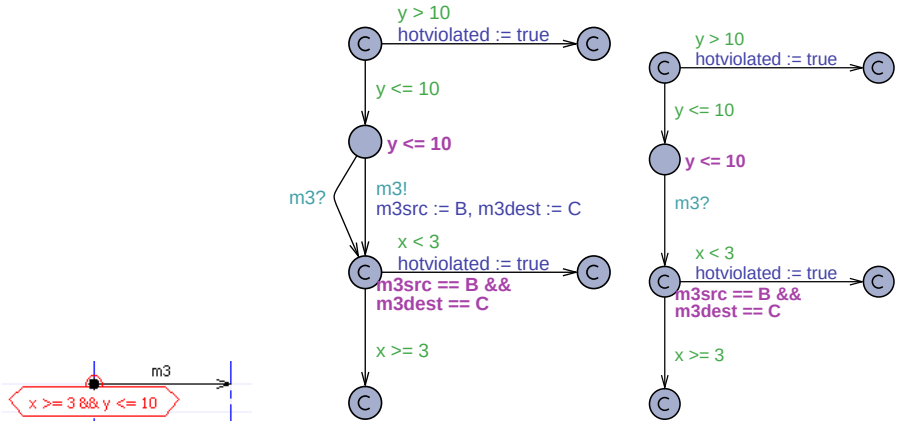
For time-enriched LSCs, there are further constructs (i.e., clock constraints and clock resets) to be considered during the translation. To mimic the behaviors of each clock constraint and clock reset in an LSC chart, we use a linked sequence of edges in the corresponding time automaton. The atomicity of executing this sequence is ensured by the UPPAAL feature of committed location.

(1) Translation of guards (clock constraints)

If an instance line has an m-labeled message sending that is guarded by a clock constraint, then a natural idea is to put this constraint on the $m!$-labeled edge of the translated TA. While this is feasible in the "one-TA-per-chart" translation method, it does not work in the "one-TA-per-instance line" method of this section. The reason is that we need to use broadcast channel m to handle the inter-chart coordination (see Section 6); however, due to the restriction of UPPAAL, broadcast channels cannot carry clock constraints [3]. To overcome this problem, in the translated TA, the upper bound constraint (if any) such as $x \leq 5$ will be tested prior to the message sending, and the lower bound and/or clock difference constraints (if any) such as $x \geq 3$ and $x - y \leq 2$ will be tested immediately after the message sending.

R10 If at position k on the main chart portion of instance line I_i of chart $\mathcal{L}_u$ there is a sending of an m-labeled message which is guarded by a clock constraint (see Fig. 7(a)), then in $A_{u,\,i}$ there will be first an intermediate committed location for upper bound constraint test. If true, then the next will be a normal location l_k with the upper bound constraint as the location invariant, which will in turn be immediately followed by a message sending edge. Finally, there will be another intermediate committed location for lower bound or clock difference constraint test. See Fig. 7(b).

For the receiving position of the guarded message, the translation is similar, see Fig. 7(c).



(a) a guarded message (b) TA for the sending instance line (c) TA for the receiving instance line

Fig. 7. Translating a guarded message to TA fragments

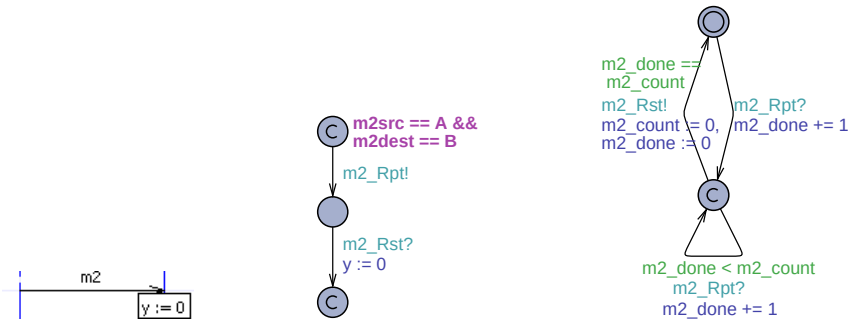
(2) Translation of assignments (clock resets)

In a time-enriched LSC chart, an assignment (i.e., clock resets) should take place immediately after the synchronization of the message occurrence that it is attached to. But in the translated TA, it cannot be put on the very edge that corresponds to the message sending/receiving, because clock resets should not occur before the lower bound or clock difference constraint test which is supposed to happen immediately *after* the message synchronization. Neither can we append the TA edge that carries the assignment to the destination locations of the lower bound or clock difference constraint test, because if several identically-labeled message occurrences are simultaneously enabled in their respective charts where those charts have different guards and/or assignments for those message occurrences (see Fig. 2, the m_3-labeled message occurrences), then there could be racing conditions (e.g., the assignment $x := 0$ that is attached to m_3 in Fig. 2(a) should happen before the lower bound test $x \geq 3$ in Fig. 2(b); however, we are unable to guarantee this).

To model the clock resets properly, for each message $m \in \Pi$ in an LSC system, we use a dedicated process (TA) A_m to coordinate the clock resets of the corresponding message occurrences that are engaged in the same broadcast synchronization on m. When the broadcast synchronization happens, we use an integer variable m_count to bookkeep how many instance lines have participated in this broadcast synchronization. Whenever one of these instance lines is done with its lower bound constraint test (if any), it will immediately notify A_m of its completion using a binary channel m_Rpt ("reporting" to A_m), and after that it will wait for a synchronization on the broadcast channel m_Rst ("resetting clocks" command from A_m), along with which it can carry out its clock resets. In A_m, an integer variable m_done is increased by 1 each time when A_m is

notified by an instance line (via $m\_Rpt?$). Once m_done rises up to m_count, A_m will immediately initiate the broadcast synchronization (via $m\_Rst!$).

R11 If at position k on instance line I_i of chart $\mathcal{L}_u$ there is a reception of an m-labeled message which has clock resets (see Fig. 8(a), instance line on the right), then there will be first an $m\_Rpt!$-labeled outgoing edge from location l_k in $A_{u,i}$, then a normal location, and then an $m\_Rst?$-labeled outgoing TA edge that carries the clock resets. See Fig. 8(b).



(a) a message with clock reset

(b) TA for the receiver

(c) the dedicated TA A_{m_2}

Fig. 8. Translating a message with clock reset to TAs

The dedicated TA A_m just waits for all the relevant instance lines to be done with their lower bound constraint tests, and then synchronizes them for clock resets. See Fig. 8(c).

(3) Just-in-Time message upper bound constraint test

When time-enriched LSC charts have upper-bound clock constraints, there are conditional tests before message sending/receiving in the translated timed automata. Given an m-labeled message occurrence in a chart, a potential problem is that in the translated timed automata for the sending and receiving instance lines, the TA locations that correspond to the sending and the receiving positions of this message may not be ready for this message synchronization at the same time (see Fig. 2(b), the m_1-labeled message occurrence). In the symbolic exploration of the state space of the translated network of timed automata, problems will arise if the upper bound of some message sending/receiving is tested when actually it *should not*. For example in Fig. 2, assume that a message sequence $m_1 \cdot m_2$ has been observed, and both charts have just entered their main charts, respectively. Note that a next m_1 is not enabled at the current cut (w.r.t. $\preccurlyeq$). But its guard will incorrectly add further constraint "$x \leq 2$" to "$(x \leq 5) \wedge (x \geq 3 \wedge y \leq 10)$". Consequently, according to our translation

method mentioned earlier in this section, all possible paths will end up with hot violations.

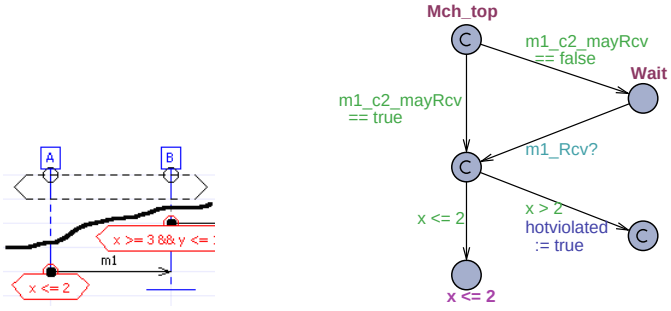
To avoid this kind of premature tests of upper bound constraints for message occurrences, we associate each message occurrence mo in each chart $\mathcal{L}_u$ with two flag boolean variables mo_u_maySnd and mo_u_mayRcv, denoting whether this message may be sent or received in chart $\mathcal{L}_u$, respectively. The upper bound constraint of mo can be tested only if both flag variables evaluate to `true`.

R12 If at position k on instance line I_i of chart $\mathcal{L}_u$ there is a sending of message occurrence mo which has a clock constraint (see Fig. 7(a)), then there will be a preceding edge carrying the predicate "mo_u_mayRcv == `true`". Once this message synchronization is fired, mo_u_maySnd will be cleared.

For the receiving instance line of message occurrence mo, the corresponding predicate will be "mo_u_maySnd == `true`".

If mo is a minimal event in the prechart (resp. main chart), then mo_u_maySnd and mo_u_mayRcv will have initial values `true` (resp. will be set to `true` by the $activate_u$ synchronization). If mo is not a minimal event, then the flag variables will be set to `true` by its predecessor event.

Given a message mo, if the predecessor positions of the head/tail positions of mo are also the head/tail (or tail/head) positions of another message occurrence, or mo is a minimal event, then mo_u_maySnd and mo_u_mayRcv will be both `true` prior to the constraint tests. Otherwise, their truth values may differ, e.g. in Fig. 9(a), message occurrence m_1 for the current cut (the solid free line). In this case, the translated TA $A_{2,A}$ will go to location `Wait` to "sleep", and will then be woken up by a dedicated message mo_1_Rcv that is sent by the TA $A_{2,B}$. See Fig. 9(b).



(a) an LSC fragment

(b) a fragment of the TA for instance line A

Fig. 9. An LSC fragment in Fig. 2(b) and the corresponding TA fragment for its instance line A

Appendix C: Complexity of the outcomes of translation

For a time-enriched LSC system, we analyze the complexity of the translated network of timed automata as follows:

Let the set $\mathcal{LS}$ of time-enriched charts $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n$ have messages $m_1, m_2, \ldots, m_k$, message occurrences $mo_1, mo_2, \ldots, mo_s$, and instance lines $I_{i,1}, I_{i,2}, \ldots, I_{i,in_i}$, where $1 \leq i \leq n$, $in_i = \#(inst(\mathcal{L}_i)) = |inst(\mathcal{L}_i)|$.

According to Section 6 ("Handling intra-chart coordinations") and Section 6 ("Translation of assignments"), the translated network of timed automata will be $NTA_{\mathcal{LS}} = \{A_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq \#(inst(\mathcal{L}_i))\} \cup \{Coord_i \mid 1 \leq i \leq n\} \cup \{A_{m_i} \mid 1 \leq i \leq k\}$. Therefore, the number of timed automata is $\sum_{i=1}^{n}(|inst(\mathcal{L}_i)|) + n + |\bigcup_{i=1}^{n} MA(\mathcal{L}_i)|$. See Table 1, lower part right column.

According to rule R2, each message label corresponds to a channel in $NTA_{\mathcal{LS}}$. According to R4, R5 and R7, there will be a set of auxiliary channels $Aux_1 = \{pch\_over_{u,i}, mch\_over_{u,i} \mid 1 \leq u \leq n, 1 \leq i \leq \#(inst(\mathcal{L}_u))\} \cup \{activate_u, over_u, pch\_vio_u, reset_u \mid 1 \leq u \leq n\}$ that will be used in $NTA_{\mathcal{LS}}$. According to Section 6 ("Translation of assignments"), in the worst case, there will be a set of auxiliary channels $Aux_2 = \{m_i\_Rpt, m_i\_Rst, m_i\_Rcv \mid 1 \leq i \leq k\}$ for translating clock resets. Therefore, in the worst case, the number of channels in $NTA_{\mathcal{LS}}$ will be $|\bigcup_{i=1}^{n} ML(\mathcal{L}_i)| + \sum_{i=1}^{n}(2 \cdot |inst(\mathcal{L}_i)| + 4) + 3 \cdot |\bigcup_{i=1}^{n} MA(\mathcal{L}_i)|$.

According to Section 6 ("Basic mapping rules"), there will be a set of auxiliary variables $\{m_i\_src, m_i\_dest \mid 1 \leq i \leq k\}$. According to rules R4, R8 and R9, there will be a set of auxiliary variables $\{prematch_u, dInst_u \mid 1 \leq u \leq n\}$. According to Section 6 ("Translation of assignments"), there will be auxiliary variables $\{m_i\_count, m_i\_done \mid 1 \leq i \leq k\}$. Furthermore, according to R12, there will be auxiliary variables $\{mo_i\_maySnd, mo_i\_mayRcv \mid 1 \leq i \leq s\}$. Therefore, the total number of auxiliary variables in $NTA_{\mathcal{LS}}$ will be $(2 \cdot |\bigcup_{i=1}^{n} MA(\mathcal{L}_i)|) + 2n + 1 + (2 \cdot |\bigcup_{i=1}^{n} MA(\mathcal{L}_i)|) + (2 \cdot \sum_{i=1}^{n} |MO(\mathcal{L}_i)|)$. □

The complexities for the other three settings can be analyzed similarly.

Appendix D: Proof of Lemmas and Theorems

Let $\mathcal{L}$ be an untimed LSC chart whose instance lines $I_1, I_2, \ldots, I_n$ correspond to timed automata $A_1, A_2, \ldots, A_n$, respectively, then the translated network of TAs will be $NTA_{\mathcal{L}} = \{A_i \mid 1 \leq i \leq n\} \cup \{Coord\}$. According to rules R4, R5 and R7, there will be a set of auxiliary channels $Aux = \{pch\_over_i, mch\_over_i \mid 1 \leq i \leq n\} \cup \{activate, over, pch\_vio, reset\}$ that will be used in $NTA_{\mathcal{L}}$. Let the message alphabet of $\mathcal{L}$ be Σ, then the alphabet of observable actions in $NTA_{\mathcal{L}}$ will be $Act = (\Sigma \cup Aux)$.

Lemma 1. *Let $\mathcal{L}$ be an untimed LSC chart whose message alphabet is Σ, and let $NTA_{\mathcal{L}}$ be the translated network of timed automata which have a set Act of observable actions. Then* $\forall \gamma_1 \in (\Sigma \cup \{\tau\})^{\omega}. ((\gamma_1 \models \mathcal{L}) \Rightarrow \exists \gamma_2 \in (Act \cup$

$\{\tau\})^{\omega}.(\gamma_2 \models NTA_{\mathcal{L}}) \wedge (\gamma_2|_{\Sigma} = \gamma_1|_{\Sigma}))$, *and* $\forall \gamma_2 \in (Act \cup \{\tau\})^{\omega}.((\gamma_2 \models NTA_{\mathcal{L}}) \Rightarrow \exists! \gamma_1 \in (\Sigma \cup \{\tau\})^{\omega}.(\gamma_1 \models \mathcal{L}) \wedge (\gamma_2|_{\Sigma} = \gamma_1|_{\Sigma}))$.

Proof. We can prove the above two implications by proving that each cut of chart $\mathcal{L}$ uniquely corresponds to a location vector in the network of timed automata $NTA_{\mathcal{L}}$, and each advancement step in $\mathcal{L}$ uniquely corresponds to either a message synchronization transition (ranging on $\Sigma \cup Aux$) or a sequence of concatenated message synchronization and internal action transitions in $NTA_{\mathcal{L}}$, such that they consume exactly the same letter from Σ if they are both projected to Σ. Note that we restrict the LSC advancement steps to represent only legal (i.e. admissible) behaviors.

Let the instance lines in chart $\mathcal{L}$ be $I_1, I_2, \ldots, I_n$. They will be translated into timed automata $A_1, A_2, \ldots, A_n$, respectively. Together with the auxiliary timed automaton *Coord* they constitute $NTA_{\mathcal{L}}$.

The initial cut c^0 of chart $\mathcal{L}$ corresponds to the LSC initial position vector $(0_1, 0_2, \ldots, 0_n)$, where i_j means that instance $I_j \in inst(\mathcal{L})$ is currently in its position $i \in pos(\mathcal{L}, I_j)$. In the translated network of timed automata $NTA_{\mathcal{L}}$, automaton *Coord* is initially in its location l^0_{coord}. By rule R1, each 0_i in position vector $(0_1, 0_2, \ldots, 0_n)$ corresponds to a TA location l^0_i (denoting location 0 in timed automaton A_i). Therefore, cut c_0 uniquely corresponds to the $NTA_{\mathcal{L}}$ initial location vector $\bar{l}^0 = (l^0_1, l^0_2, \ldots, l^0_n, l^0_{coord})$.

We show how the advancement steps from the LSC initial position vector correspond to the transitions in the network of timed automata. At LSC position vector $(0_1, 0_2, \ldots, 0_n)$, there are two kinds of possible advancement steps:

- If there is an m-labeled message occurrence mo from position 1_i of instance I_i to position 1_j of instance I_j (i.e., mo is a minimal event), then:
  - On one hand, by rules R2 and R3, there will be an $m!$-labeled TA edge from location l^0_i to l^1_i in A_i, and an $m?$-labeled TA edge from location l^0_j to l^1_j in A_j. According to the LSC semantics, there is a message synchronization advancement step on m in $\mathcal{L}$ from $(0_1, \ldots, 0_i, \ldots, 0_j, \ldots, 0_n)$ to $(0_1, \ldots, 1_i, \ldots, 1_j, \ldots, 0_n)$. Accordingly, in $NTA_{\mathcal{L}}$ there exists exactly a corresponding binary synchronization on channel m between A_i and A_j, and the location vector of $NTA_{\mathcal{L}}$ will change from $(l^0_1, \ldots, l^0_i, \ldots, l^0_j, \ldots, l^0_n, l^0_{coord})$ to $(l^0_1, \ldots, l^1_i, \ldots, l^1_j, \ldots, l^0_n, l^0_{coord})$.
  - On the other hand, according to the semantics of the invariant mode universal chart, the message as a minimal event can be constantly matched for with $\mathcal{L}$ staying in the initial cut. By rule R9, in $NTA_{\mathcal{L}}$ there will be first a binary synchronization on channel m, i.e., $(l^0_1, \ldots, l^0_i, \ldots, l^0_j, \ldots, l^0_n, l^0_{coord}) \xrightarrow{m} (l^0_1, \ldots, l^0_i, \ldots, l^{PM}_j, \ldots, l^0_n, l^0_{coord})$, and then an immediately following internal action transition that leads back to the initial location vector, i.e., $(l^0_1, \ldots, l^0_i, \ldots, l^{PM}_j, \ldots, l^0_n, l^0_{coord}) \xrightarrow{\tau} (l^0_1, \ldots, l^0_i, \ldots, l^0_j, \ldots, l^0_n, l^0_{coord})$. Here l^{PM}_j is an auxiliary TA location that is specially used for prechart pre-matching. In this sub-case of pre-matching, the m-synchronization advancement step in $\mathcal{L}$ uniquely corresponds to a sequence of the tightly concatenated $\xrightarrow{m}$ and $\xrightarrow{\tau}$ transitions.

Since a dedicated flag boolean variable *prematch* has been used to strengthen the TA transition guards, assignments and the location invariants, it follows that at $NTA_{\mathcal{L}}$ location vector $(l_1^0, \ldots, l_i^0, \ldots, l_j^0, \ldots, l_n^0, l_{coord}^0)$, there are only the two above-mentioned possible interleaved executions between the two $m!$-labeled outgoing edges from l_i^0 in I_i and the two $m?$-labeled outgoing edges from l_j^0 in I_j.

- If instance I_i has no interactions with other instance lines in the prechart, then there is an immediate silent advancement step from $(0_1, \ldots, 0_i, \ldots, 0_n)$ to $(0_1, \ldots, 1_i, \ldots, 0_n)$. By rule R4, l_i^0 will be a committed location in $NTA_{\mathcal{L}}$, and there will be a $pch\_over_i!$-labeled edge from l_i^0 to l_i^1. Furthermore, in automaton *Coord* there will be a coupling $pch\_over_i?$-labeled edge either
  - from l_{coord}^0 to l_{coord}^1, corresponding to the case where I_i is the very last instance to complete the prechart; or
  - from l_{coord}^0 to l_{coord}^0, corresponding to the case where I_i is not yet the last instance to complete the prechart.

  In the two cases, the location vector of $NTA_{\mathcal{L}}$ will be changed from $(l_1^0, \ldots, l_i^0, \ldots, l_n^0, l_{coord}^0)$ to $(l_1^0, \ldots, l_i^1, \ldots, l_n^0, l_{coord}^1)$, and from $(l_1^0, \ldots, l_i^0, \ldots, l_n^0, l_{coord}^0)$ to $(l_1^0, \ldots, l_i^1, \ldots, l_n^0, l_{coord}^0)$, respectively. However, in both cases, there will be exactly one binary synchronization transition on pch_over_i in $NTA_{\mathcal{L}}$.

The above two kinds of possible advancement steps indicate that there is an initial correspondence between the position vector of $\mathcal{L}$ and the location vector of $NTA_{\mathcal{L}}$ [5]. Since an untimed chart is a message-only chart, a cut vector is itself an LSC configuration, and a location vector is itself a semantic state of the translated network of timed automata [6]. Therefore, there is an initial "cut-to-location vector", and "advancement step-to-(sequence of) transition" correspondence between $\mathcal{L}$ and $NTA_{\mathcal{L}}$.

The above correspondence can be generalized by using induction. Assume that at a cut c that corresponds to a position vector $(p1_1, \ldots, pi_i, \ldots, pj_j, \ldots, pn_n)$ in the prechart of $\mathcal{L}$, there is an m-labeled message occurrence sent from position $(pi+1)_i$ of instance I_i to position $(pj+1)_j$ of instance I_j. If for cut c, there uniquely exists a corresponding location vector $\bar{l}$ in $NTA_{\mathcal{L}}$, then similar to the case of the initial cut, we can prove that the message synchronization advancement step on m in $\mathcal{L}$ uniquely corresponds to a binary synchronization transition in $NTA_{\mathcal{L}}$; and after this message synchronization advancement step, the new cut c' uniquely corresponds to the destination location vector $\bar{l}'$ in $NTA_{\mathcal{L}}$. Proof by induction ensures that any normal (i.e., other than the prechart pre-matching ones) message synchronization advancement step in the prechart of $\mathcal{L}$ uniquely corresponds to a message synchronization transition in $NTA_{\mathcal{L}}$.

[5] More precisely the sub-location vector of $NTA_{\mathcal{L}}$ that is projected to $A_1||A_2||\ldots||A_n$. Note that the edges in *Coord* correspond only to auxiliary messages rather than the observable messages in Σ or the internal (τ) action.

[6] Note that in the LSC chart, the message sender/receiver and other relevant information are not defined as a part of the chart configuration. Accordingly, the auxiliary and bookkeeping variable information are excluded from the semantic states of the translated timed automata.

In case that $(p1_1, \ldots, pi_i, \ldots, pj_j, \ldots, pn_n)$ is a position vector in the main chart of $\mathcal{L}$, the unique correspondence relation can be proved similarly.

Now we prove the unique correspondence for the case that involves the intra-chart coordination (e.g., the prechart to main chart transition). Assume that in the prechart of $\mathcal{L}$, a cut c corresponds to position vector $(p1_1, \ldots, pi_i, \ldots, pn_n)$, where $pi + 1 = \mathcal{L}.I_i.Pch\_bot$. If $(p1_1, \ldots, pi_i, \ldots, pn_n)$ uniquely corresponds to a location vector $(l_1^{p1}, \ldots, l_i^{pi}, \ldots, l_n^{pn}, l_{coord}^0)$, then by rule R4, the internal advancement step $(p1_1, \ldots, pi_i, \ldots, pn_n) \xrightarrow{\tau} (p1_1, \ldots, (pi+1)_i, \ldots, pn_n)$ in $\mathcal{L}$ corresponds to either

- transition $(l_1^{p1}, \ldots, l_i^{pi}, \ldots, l_n^{pn}, l_{coord}^0) \xrightarrow{pch\_over_i} (l_1^{p1}, \ldots, l_i^{pi+1}, \ldots, l_n^{pn}, l_{coord}^1)$ in $NTA_{\mathcal{L}}$, in which case I_i is the very last instance to complete the prechart; or
- transition $(l_1^{p1}, \ldots, l_i^{pi}, \ldots, l_n^{pn}, l_{coord}^0) \xrightarrow{pch\_over_i} (l_1^{p1}, \ldots, l_i^{pi+1}, \ldots, l_n^{pn}, l_{coord}^0)$ in $NTA_{\mathcal{L}}$, in which case I_i is not yet the last instance to complete the prechart.

The above-mentioned first case will be followed by an intra-chart coordination, i.e., there will be an immediately following silent advancement step in $\mathcal{L}$, i.e., all instance lines will move from their *Pch_bot* positions to their *Mch_top* positions at the same time. By rule R5, the binary synchronization transition will be immediately followed by a broadcast synchronization transition $(l_1^{p1}, \ldots, l_i^{pi+1}, \ldots, l_n^{pn}, l_{coord}^1) \xrightarrow{activate} (l_1^{p1+1}, \ldots, l_i^{pi+2}, \ldots, l_n^{pn+1}, l_{coord}^2)$, where $p1 + 1 = \mathcal{L}.I_1.Mch\_top, \ldots, pi + 2 = \mathcal{L}.I_i.Mch\_top, \ldots, pn + 1 = \mathcal{L}.I_n.Mch\_top$. Therefore in this case, there is a correspondence between the behaviors of $\mathcal{L}$ and $NTA_{\mathcal{L}}$.

In case that $(p1_1, \ldots, pi_i, \ldots, pn_n)$ is a position vector in the main chart of $\mathcal{L}$, the unique correspondence for the case that concerns main chart completion can be proved similarly.

Now we prove the unique correspondence for the case that involves cold violations. Since an untimed chart has no conditions, a cold violation is caused only by the violation of the partial order in the prechart. In this case, all the instance lines in the prechart of $\mathcal{L}$ will be brought from where they are back to their initial positions. Recall that $psn : loc(\mathcal{L}) \to \bigcup_{I_i \in inst(\mathcal{L})} pos(\mathcal{L}, I_i)$ projects a location to its position on its instance line. Formally, let us assume that $\mathcal{L}$ is in the cut c which corresponds to the position vector $(p1_1, \ldots, pi_i, \ldots, pj_j, \ldots, pn_n)$ such that $pk_k < \mathcal{L}.I_k.Pch\_bot, 1 \leq k \leq n$. For any message label $m \in \Sigma$, if $\nexists mo \in MO(\mathcal{L}).(lab(mo) = m) \wedge (\exists I_i, I_j \in inst(\mathcal{L}).((src(mo) = I_i) \wedge (dest(mo) = I_j) \wedge (psn(tail(mo)) = pi + 1) \wedge (psn(head(mo)) = pj + 1)))$, then at cut c, the partial order will be cold-violated by any m-labeled message from I_i to I_j. For such an m-labeled message occurrence mo, by rule R7, the cold violation step $(p1_1, \ldots, pi_i, \ldots, pj_j, \ldots, pn_n) \xrightarrow{m} (0_1, \ldots, 0_i, \ldots, 0_j, \ldots, 0_n)$ in $\mathcal{L}$ uniquely corresponds to a sequence of three concatenated synchronizations in $NTA_{\mathcal{L}}$:
$(l_1^{p1}, \ldots, l_i^{pi}, \ldots, l_j^{pj}, \ldots, l_n^{pn}, l_{coord}^0) \xrightarrow{m}$
$(l_1^{p1}, \ldots, l_i^{pi+1}, \ldots, l_j^{Rst}, \ldots, l_n^{pn}, l_{coord}^0) \xrightarrow{pch\_vio}$

$(l_1^{p1}, \ldots, l_i^{pi+1}, \ldots, l_j^0, \ldots, l_n^{pn}, l_{coord}^{Rst}) \xrightarrow{reset}$
$(l_1^0, \ldots, l_i^0, \ldots, l_j^0, \ldots, l_n^0, l_{coord}^0)$.

Note that according to rule R8, a hot violation in the main chart of $\mathcal{L}$ will end up with a semantic state that has a deadend location in a certain TA of $NTA_{\mathcal{L}}$. This transition will *not* be considered as a part of an accepted trace of $NTA_{\mathcal{L}}$.

In conclusion, each possible advancement step in $\mathcal{L}$ uniquely corresponds to a sequence of concatenated message synchronization and internal action transitions in $NTA_{\mathcal{L}}$. They consume exactly the same message label in Σ. Therefore, each accepted trace in $\mathcal{L}$ uniquely corresponds to an accepted trace in $NTA_{\mathcal{L}}$ modulo the message alphabet Σ. □

Let $\mathcal{LS}$ be a set of untimed LSC charts $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n$. Each chart $\mathcal{L}_i$ contains the instance lines $I_{i,1}, I_{i,2}, \ldots, I_{i,in_i}$, where $1 \leq i \leq n$, and $in_i = \#(inst(\mathcal{L}_i))$ denotes the number of instance lines in $\mathcal{L}_i$. The entire translated network of TAs will be $NTA_{\mathcal{LS}} = \{A_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq \#(inst(\mathcal{L}_i))\} \cup \{Coord_i \mid 1 \leq i \leq n\}$. The message alphabet of $\mathcal{LS}$ will be the union of all the message alphabets for the individual charts, i.e., $\Pi = \bigcup_{i=1}^{n} \Sigma_i$. The alphabet of observable actions will be $Act = (\Pi \cup Aux)$.

Lemma 2. *Let $\mathcal{LS}$ be a set of untimed LSC charts whose message alphabet is Π, and let $NTA_{\mathcal{LS}}$ be the translated network of timed automata which have a set $Act = \Pi \cup Aux$ of normal and auxiliary channels. Then $\forall \gamma_1 \in (\Pi \cup \{\tau\})^\omega.((\gamma_1 \models \mathcal{LS}) \Rightarrow \exists \gamma_2 \in (Act \cup \{\tau\})^\omega.(\gamma_2 \models NTA_{\mathcal{LS}}) \wedge (\gamma_2|_\Pi = \gamma_1|_\Pi))$, and $\forall \gamma_2 \in (Act \cup \{\tau\})^\omega.((\gamma_2 \models NTA_{\mathcal{LS}}) \Rightarrow \exists! \gamma_1 \in (\Pi \cup \{\tau\})^\omega.(\gamma_1 \models \mathcal{LS}) \wedge (\gamma_2|_\Pi = \gamma_1|_\Pi))$.*

Proof. In this case, in order to prove the above two implications, we need to prove that each *cut vector* of $\mathcal{LS}$ uniquely corresponds to a location vector in $NTA_{\mathcal{LS}}$, and each advancement step in $\mathcal{LS}$ uniquely corresponds to an equivalence class of sequences of concatenated (broadcast) synchronization and internal action transitions in $NTA_{\mathcal{LS}}$. Although elements in the equivalence class have different intermediate location vectors, they have the same initial and final location vectors. They consume exactly the same message in Π. Note that an advancement step in $\mathcal{LS}$ always represents a legal behavior.

By Lemma 1, for each untimed chart $\mathcal{L}_i$ in $\mathcal{LS}$, each cut in $\mathcal{L}_i$ uniquely corresponds to a location vector in the corresponding network of timed automata $NTA_{\mathcal{L}_i}$, and each advancement step in $\mathcal{L}_i$ uniquely corresponds to either a single message synchronization transition, or a sequence of concatenated message synchronization and internal action transitions in $NTA_{\mathcal{L}_i}$.

The only semantic difference between the advancement steps of a single untimed chart and of a set of untimed charts is that in the latter case there exist *inter-chart* coordinations, i.e., across-chart broadcast synchronization on message occurrences of the same message is possible. This implies that:

(1) At a cut vector of $\mathcal{LS}$, if in more than one chart there are enabled message occurrences of the same message, then either all of them are chosen to be fired simultaneously, or none of them is chosen to be fired;

(2) Due to the nature of broadcast synchronization in the translated network of TAs, while a message at a cut vector of $\mathcal{LS}$ could correspond to a legal message synchronization advancement step in a certain chart, meanwhile it could also lead another chart to be reset by cold-violating the prechart of that chart (case 2.1), or lead another chart to a deadlocked situation by hot-violating the main chart of that chart (case 2.2).

In case (1), given a set $\mathcal{LS}$ of untimed LSC charts $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n$, we let $in_i = \#(inst(\mathcal{L}_i))$, $1 \leq i \leq n$. We assume that the current cut vector $\bar{c}$ of $\mathcal{LS}$ uniquely corresponds to the position vector $(p_{1,1}, p_{1,2}, \ldots, p_{1,in_1}, p_{2,1}, p_{2,2}, \ldots, p_{2,in_2}, \ldots, p_{n,1}, p_{n,2}, \ldots, p_{n,in_n})$, where $p_{i,j} \in pos(\mathcal{L}_i, I_j)$ denotes the current position on instance I_j of chart $\mathcal{L}_i$. Without loss of generality, we assume that two m-labeled message occurrences mo_1 and mo_2 are enabled at cut vector $\bar{c}$ in two charts $\mathcal{L}_i$ and $\mathcal{L}_j$, respectively. Specifically, let $(p_{i,a}+1)$ and $(p_{i,b}+1)$ be the sending and receiving positions of mo_1 in $\mathcal{L}_i$, where $1 \leq a, b \leq in_i$, and let $(p_{j,c}+1)$ and $(p_{j,d}+1)$ be the sending and receiving positions of mo_2 in $\mathcal{L}_j$, where $1 \leq c, d \leq v_j$. According to the trace-based semantics for a set of charts, these two message synchronization advancement steps in $\mathcal{L}_i$ and $\mathcal{L}_j$ will occur simultaneously. By rules R2 and R3, there will be an $m!$-labeled edge from location $l_{i,a}^{p_{i,a}}$ to $l_{i,a}^{p_{i,a}+1}$ in $A_{i,a}$, and an $m?$-labeled edge from location $l_{i,b}^{p_{i,b}}$ to $l_{i,b}^{p_{i,b}+1}$ in $A_{i,b}$, and similarly for chart $\mathcal{L}_j$. By rule R6, there will be added an extra $m?$-labeled edge from location $l_{i,a}^{p_{i,a}}$ to $l_{i,a}^{p_{i,a}+1}$ in $A_{i,a}$, and similarly in chart $\mathcal{L}_j$. Consequently, there will be a broadcast synchronization on m among $A_{i,a}$, $A_{i,b}$, $A_{j,c}$, $A_{j,d}$, initiated either by $A_{i,a}$, or by $A_{j,c}$. In either case, after this broadcast synchronization on m in $NTA_{\mathcal{LS}}$, the locations of $A_{i,a}$, $A_{i,b}$, $A_{j,c}$ and $A_{j,d}$ will progress to $l_{i,a}^{p_{i,a}+1}$, $l_{i,b}^{p_{i,b}+1}$, $l_{j,c}^{p_{j,c}+1}$ and $l_{j,d}^{p_{j,d}+1}$, respectively. Therefore, the message synchronization advancement step on m in $\mathcal{LS}$ corresponds to two possible interleaved executions among $A_{i,a}$, $A_{i,b}$, $A_{j,c}$ and $A_{j,d}$. Since both interleavings consume the same message label m, they correspond to the same portion of the accepted trace in $NTA_{\mathcal{LS}}$. These two interleaved executions constitute an equivalence class with respect to the message synchronization advancement step on m.

In case (2.1), assume that the current cut vector $\bar{c}$ of $\mathcal{LS}$ corresponds to position vector $(p_{1,1}, p_{1,2}, \ldots, p_{1,in_1}, p_{2,1}, p_{2,2}, \ldots, p_{2,in_2}, \ldots, p_{n,1}, p_{n,2}, \ldots, p_{n,in_n})$. Without loss of generality, we assume that an m-labeled message occurrence mo is currently enabled in $\mathcal{L}_i$, but not in $\mathcal{L}_j$, and that $\bar{c}$ "cuts" $\mathcal{L}_j$ in the prechart of $\mathcal{L}_j$. According to the semantics for a set of LSC charts, when message m is encountered, there will be a normal advancement step in $\mathcal{L}_i$, and a cold violation advancement step in $\mathcal{L}_j$. By Lemma 1, such a cold violation advancement step uniquely corresponds to a sequence of synchronizations in the relevant timed automata. Therefore, the system-wide synchronization on m will also uniquely correspond to a system-wide sequence of synchronizations in $NTA_{\mathcal{LS}}$.

In case (2.2), assume that the current cut vector $\bar{c}$ of $\mathcal{LS}$ corresponds to position vector $(p_{1,1}, p_{1,2}, \ldots, p_{1,in_1}, p_{2,1}, p_{2,2}, \ldots, p_{2,in_2}, \ldots, p_{n,1}, p_{n,2}, \ldots, p_{n,in_n})$. Without loss of generality, we assume that an m-labeled message occurrence mo is currently enabled in $\mathcal{L}_i$, but not in $\mathcal{L}_j$, and that $\bar{c}$ "cuts" $\mathcal{L}_j$ in the main chart

of $\mathcal{L}_j$. According to the semantics for a set of LSC charts, when message m is encountered, there will be a normal message synchronization advancement step in $\mathcal{L}_i$, and a hot violation in $\mathcal{L}_j$. Specifically, let $p_{i,a}$ and $p_{i,b}$ be the sending and receiving positions of mo in $\mathcal{L}_i$, where $1 \leq a, b \leq in_i$. We let the sub-position vector in $\mathcal{L}_j$ be $c_j = (p_{j,1}, p_{j,2}, \ldots, p_{j,in_j})$. Obviously, mo is not enabled at sub-cut c_j. Since $\mathcal{L}_j$ is hot-violated by mo, there must exist a position, say $p_{j,x}$, $1 \leq x \leq in_j$, such that there is an $m?$-labeled edge from $p_{j,x}$ to a sink error location `Err` in $A_{j,x}$. Furthermore, there could possibly exist another position, say $p_{j,y}$, $1 \leq y \leq in_j$, such that there is an $m!$-labeled edge from position $p_{j,y}$ to $(p_{j,y} + 1)$ in $A_{j,y}$. This means that there could be one or two possible initiating TAs of the broadcast synchronization. Whichever case could it be, the same label (m) will be consumed, and the same next semantic state of $NTA_{\mathcal{LS}}$ will be reached. This semantic state will have a deadend location `Err`, which indicates that the system will be deadlocked. Therefore, the TA transition step leading to this semantic state will not be considered as a part of the accepted trace. In this case, m will not be allowed to occur at cut vector $\bar{c}$. This demonstrates how the different charts constrain the behaviors of each others. In summary, in case (2.2), a to-be-hot violating message in $\mathcal{LS}$ uniquely corresponds to a to-be-deadlocked TA transition in $NTA_{\mathcal{LS}}$.

Based on the above discussions, we conclude that there exists a unique correspondence between the observable traces of a set of untimed LSC charts and their corresponding network of timed automata. □

Let $\mathcal{L}$ be a time-enriched chart whose instance lines $I_1, I_2, \ldots, I_n$ correspond to timed automata $A_1, A_2, \ldots, A_n$, respectively. Let the message alphabet of $\mathcal{L}$ be $\{m_1, m_2, \ldots, m_k\}$. According to Section 6 ("Translation of assignments"), there will be an auxiliary timed automaton A_{m_i} for each m_i, $1 \leq i \leq k$. Consequently, the translated network of TAs will be $NTA_{\mathcal{L}} = \{A_i \mid 1 \leq i \leq n\} \cup \{Coord\} \cup \{A_{m_i} \mid 1 \leq i \leq k\}$.

According to rules R4, R5 and R7, there will be auxiliary channels $Aux = \{pch\_over_i,\ mch\_over_i \mid 1 \leq i \leq n\} \cup \{activate, over, pch\_vio, reset\}$ used in $NTA_{\mathcal{L}}$. According to rule R11, there will be auxiliary channels $Aux' = \{m_i\_Rpt, m_i\_Rst, m_i\_Rcv \mid 1 \leq i \leq k\}$ used in $NTA_{\mathcal{L}}$. Let the message alphabet of $\mathcal{L}$ be Σ, then the alphabet of observable actions in $NTA_{\mathcal{L}}$ will be $Act = \Sigma \cup Aux \cup Aux'$.

Lemma 3. *Let $\mathcal{L}$ be a time-enriched LSC chart whose message alphabet is Σ, and let $NTA_{\mathcal{L}}$ be the translated network of timed automata which have a set $Act = \Sigma \cup Aux \cup Aux'$ of normal and auxiliary channels. Then $\forall \gamma_1 \in (\Sigma \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^{\omega}. ((\gamma_1 \models \mathcal{L}) \Rightarrow \exists \gamma_2 \in (Act \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^{\omega}.(\gamma_2 \models NTA_{\mathcal{L}}) \wedge (\gamma_2|_{(\Sigma \cup \mathbb{R}_{\geq 0})} = \gamma_1|_{(\Sigma \cup \mathbb{R}_{\geq 0})}))$, and $\forall \gamma_2 \in (Act \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^{\omega}. ((\gamma_2 \models NTA_{\mathcal{L}}) \Rightarrow \exists! \gamma_1 \in (\Sigma \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^{\omega}.(\gamma_1 \models \mathcal{L}) \wedge (\gamma_2|_{(\Sigma \cup \mathbb{R}_{\geq 0})} = \gamma_1|_{(\Sigma \cup \mathbb{R}_{\geq 0})}))$.*

Proof. In order to prove the above two implications, we need to show that each configuration of chart $\mathcal{L}$ uniquely corresponds to a certain semantic state of $NTA_{\mathcal{L}}$, and each advancement step in $\mathcal{L}$ uniquely corresponds to a sequence of concatenated message synchronization transitions, and/or internal action transitions, and/or time delay transitions in $NTA_{\mathcal{L}}$ such that they either consume

exactly the same letter from Σ, or undergo exactly the same period of time delay.

By Lemma 1, each cut of an untimed chart $\mathcal{L}$ uniquely corresponds to a semantic state in $NTA_{\mathcal{L}}$, and each advancement step in $\mathcal{L}$ uniquely corresponds to either a message synchronization transition, or a sequence of concatenated message synchronization and internal action transitions in $NTA_{\mathcal{L}}$. For a time-enriched LSC chart, we keep this skeleton correspondence, i.e., we map position pi_i of instance line I_i to location l_i^{pi} of the timed automaton A_i. Note that along an instance line of the time-enriched chart, two adjacent LSC positions typically do not correspond to two adjacent locations in the corresponding translated TA. Between location l_i^{pi} and l_i^{pi+1}, where $0 \leq pi \leq (p\_max_{\mathcal{L},I_i} - 1)$, according to rules R10, R11 and R12, we will add some intermediate auxiliary TA locations, and add some TA edges to connect them.

Now we prove that a message synchronization advancement step on m in $\mathcal{L}$ uniquely corresponds to a sequence of transitions in $NTA_{\mathcal{L}}$ that consumes m exactly. Assume that at a configuration c which corresponds to position vector $(p1_1, \ldots, pi_i, \ldots, pj_j, \ldots, pn_n)$ in the prechart of $\mathcal{L}$ and clock valuation v, there is an m-labeled message occurrence mo with condition (clock constraints) g and assignment (clock resets) a sent from position $(pi+1)_i$ of instance I_i to position $(pj+1)_j$ of instance I_j. Assume that position pi_i corresponds to location l_i^{pi} in A_i, and position $(pi+1)_i$ corresponds to location l_i^{pi+1} in A_i, then there will be 5 intermediate locations between l_i^{pi} and l_i^{pi+1} in A_i, which we denote as $l_i^{pi,1}, l_i^{pi,2}, l_i^{pi,3}, l_i^{pi,4}$ and $l_i^{pi,5}$. Here

- between l_i^{pi} and $l_i^{pi,1}$, there is a TA edge with the guard "$m\_mayRcv ==$ `true`";
- between $l_i^{pi,1}$ and $l_i^{pi,2}$, there is a TA edge which tests the upper bound constraints;
- between $l_i^{pi,2}$ and $l_i^{pi,3}$, there is an $m!$-labeled TA edge;
- between $l_i^{pi,3}$ and $l_i^{pi,4}$, there is a TA edge which tests the lower bound and/or clock difference constraints;
- between $l_i^{pi,4}$ and $l_i^{pi,5}$, there is an $m\_Rpt!$-labeled TA edge;
- between $l_i^{pi,5}$ and l_i^{pi+1}, there is an $m\_Rst!$-labeled TA edge.

Similarly, there will be 5 intermediate locations and edges that connect them in A_j. Specifically, if there are positions on other instance line that are waiting for the completion of this message synchronization according to the partial order relation, then there will be one more intermediate location $l_i^{pi,6}$, and an $m\_Rcv!$-labeled edge connecting $l_i^{pi,6}$ to l_i^{pi+1}. According to rule R12, the position sub-vector (pi_i, pj_j) corresponds to the TA location sub-vector (l_i^{pi}, l_j^{pj}), where both locations are committed locations. After these two transitions from (l_i^{pi}, l_j^{pj}), the new location sub-vector $(l_i^{pi,1}, l_j^{pj,1})$ will be reached, which are also committed locations. Since a legal advancement step in $\mathcal{L}$ will not violates the upper bound of the clock constraints, the upper bound constraint will evaluate to true and thus the next location sub-vector will be $(l_i^{pi,2}, l_j^{pj,2})$. From $(l_i^{pi,2}, l_j^{pj,2})$ there will

be the message synchronization on m leading to $(l_i^{pi,3}, l_j^{pj,3})$, which are again committed locations. After comparing the lower bound of clock constraints, the location sub-vector $(l_i^{pi,4}, l_j^{pj,4})$ will be reached. Now instance lines I_i and I_j will immediately report to the automaton A_m, telling it that the instances are done with testing the guarding flag boolean variables, testing the upper bound, message synchronization, and testing the lower bound or clock difference. Once both instance lines have notified A_m of their completions, A_m will immediately initiate an m_Rst-labeled broadcast synchronization which brings A_i from $l_i^{pi,5}$ to l_i^{pi+1}, and brings A_j from $l_j^{pj,5}$ to l_j^{pj+1}. Specifically, if there is an $l_i^{pi,6}$ in A_i, then the $m\_Rst?$-labeled edge will be from $l_i^{pi,5}$ to $l_i^{pi,6}$ in A_i, and there will be an $m\_Rcv!$-labeled edge from $l_i^{pi,6}$ to l_i^{pi+1}. This latter edge can occur autonomously without synchronizing with a message-receiving TA, because m_Rcv is declared as a broadcast channel. In summary, the message synchronization step on m in $\mathcal{L}$ will uniquely correspond to such a sequence of transitions in $NTA_{\mathcal{L}}$.

For a silent advancement step in $\mathcal{L}$, it is the same as in the untimed case. In other words, the corresponding proof for Lemma 1 also applies here.

For a time delay advancement step in $\mathcal{L}$, since the upper bounds and lower bounds of clock constraints are properly translated to tests that are prior to and after the message synchronization in $NTA_{\mathcal{L}}$, a time delay of a period of $d \in \mathbb{R}_{\geq 0}$ is allowed in $NTA_{\mathcal{L}}$ if and only if the same period d of time delay is allowed in $\mathcal{L}$.

In all the three possible cases of an advancement step in $\mathcal{L}$, there will be a uniquely corresponding sequence of transitions in $NTA_{\mathcal{L}}$ such that this sequence consumes exactly the same message, or amount of time delay as that step in $\mathcal{L}$. □

Let $\mathcal{LS}$ be a set of time-enriched LSC charts $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n$. Each chart $\mathcal{L}_i$ contains the instance lines $I_{i,1}, I_{i,2}, \ldots, I_{i,in_i}$, where $in_i = \#(inst(\mathcal{L}_i))$. Let the message alphabet Π of $\mathcal{LS}$ be $\Pi = \bigcup_{i=1}^{n} \Sigma_i = \{m_1, m_2, \ldots, m_k\}$. Then the translated network of TAs will be $NTA_{\mathcal{LS}} = \{A_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq \#(inst(\mathcal{L}_i))\} \cup \{Coord_i \mid 1 \leq i \leq n\} \cup \{A_{m_i} \mid 1 \leq i \leq k\}$. Similarly to Lemma 3, we let $Act = \Pi \cup Aux \cup Aux'$.

Theorem 1. Let $\mathcal{LS}$ be a set of time-enriched LSC charts whose message alphabet is Π, and let $NTA_{\mathcal{LS}}$ be the translated network of timed automata which have a set Act of normal and auxiliary channels. Then $\forall \gamma_1 \in (\Pi \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^{\omega}. ((\gamma_1 \models \mathcal{LS}) \Rightarrow \exists \gamma_2 \in (Act \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^{\omega}.(\gamma_2 \models NTA_{\mathcal{LS}}) \wedge (\gamma_2|_{(\Pi \cup \mathbb{R}_{\geq 0})} = \gamma_1|_{(\Pi \cup \mathbb{R}_{\geq 0})}))$, and $\forall \gamma_2 \in (Act \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^{\omega}. ((\gamma_2 \models NTA_{\mathcal{LS}}) \Rightarrow \exists! \gamma_1 \in (\Pi \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^{\omega}.(\gamma_1 \models \mathcal{LS}) \wedge (\gamma_2|_{(\Pi \cup \mathbb{R}_{\geq 0})} = \gamma_1|_{(\Pi \cup \mathbb{R}_{\geq 0})}))$.

Proof. We need to prove that each cut vector of $\mathcal{LS}$ uniquely corresponds to a location vector in $NTA_{\mathcal{LS}}$, and each message synchronization advancement step in $\mathcal{LS}$ uniquely corresponds to a sequence of concatenated message synchronization transitions, and internal action transitions in $NTA_{\mathcal{LS}}$. These transitions are connected by committed locations in $NTA_{\mathcal{LS}}$. Because any committed location

appears as a junction location only when it will be immediately followed (only) by a condition test, these concatenated transitions can be viewed as an atomic step. Although for the sake of inter-chart coordination, the outgoing transitions from locations of different TAs may be executed in an interleaved manner, the order of the consumed words in $(\Pi \cup \{\tau\})^*$ remains the same. In other words, an accepted timed trace $\gamma \in (\Pi \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^\omega$ may correspond to an equivalence class of timed traces in $(Act \cup \{\tau\} \cup \mathbb{R}_{\geq 0})^*$. They consume exactly the same timed trace in $(\Pi \cup \mathbb{R}_{\geq 0})^*$. Proof details concerning the translations of inter-chart message coordinations and message occurrences that are associated with conditions and/or assignments are similar to that for Lemmas 2 and 3, respectively. □

Let $\mathcal{LS}$ be an LSC system which consists of a set of (untimed or timed) driving universal charts $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n$. We translate $\mathcal{LS}$ to a network of timed automata $NTA_{\mathcal{LS}}$. Let $\mathcal{L}'$ be a separate monitored universal chart (the "property chart"), which will be translated to another network of timed automata $NTA_{\mathcal{L}'}$. As explained earlier, the TA locations $Coord_{\mathcal{L}'}.Mch\_top$ and $Coord_{\mathcal{L}'}.Mch\_bot$ denote that the main chart of $\mathcal{L}'$ has just been activated and has just been successfully matched, respectively. We have:

Theorem 3. $\mathcal{LS} \models \mathcal{L}' \Leftrightarrow (NTA_{\mathcal{LS}} \,||\, NTA_{\mathcal{L}'}) \models Coord_{\mathcal{L}'}.Mch\_top \rightsquigarrow Coord_{\mathcal{L}'}.Mch\_bot$.

Proof. By Theorem 1, each accepted trace in $\mathcal{LS}$ uniquely corresponds to a cluster of accepted traces in $NTA_{\mathcal{LS}}$ which consume exactly the same string from $(\Pi \cup \mathbb{R}_{\geq 0})^\omega$. And similarly for $\mathcal{L}'$ and $NTA_{\mathcal{L}'}$.

The TA location $Coord_{\mathcal{L}'}.Mch\_top$ represents the situation where the property chart $\mathcal{L}'$ is activated, and $Coord_{\mathcal{L}'}.Mch\_bot$ the situation where $\mathcal{L}'$ is satisfied (i.e., successfully matched).

Since $\mathcal{L}'$ is a property chart, its corresponding network of timed automata $NTA_{\mathcal{L}'}$ will never interfere with (or "drive") the network of timed automata $NTA_{\mathcal{LS}}$. This means that after parallelly composing the TAs in $NTA_{\mathcal{L}'}$ with the TAs in $NTA_{\mathcal{LS}}$, the behaviors in $NTA_{\mathcal{LS}}$ will not be further constrained. Since both $Coord_{\mathcal{L}'}.Mch\_top$ and $Coord_{\mathcal{L}'}.Mch\_bot$ are locations in the product automaton of $(NTA_{\mathcal{LS}} \,||\, NTA_{\mathcal{L}'})$, the right hand side formula of this theorem captures exactly the assume-guarantee style responsiveness property of the LSC requirement, which is exactly what we require of $\mathcal{LS} \models \mathcal{L}'$. □

An LSC system $\mathcal{LS}$ satisfies a monitored existential chart $\mathcal{L}'$ iff one of the traces in $\mathcal{LS}$ is included in the traces of $\mathcal{L}'$.

Theorem 4. $\mathcal{LS} \models \mathcal{L}' \Leftrightarrow (NTA_{\mathcal{LS}} \,||\, NTA_{\mathcal{L}'}) \models \mathsf{E}\Diamond\ Coord_{\mathcal{L}'}.Mch\_bot$.

Proof. This theorem can be proved similarly to Theorem 3, except that an existential chart has no prechart. □