Mani Swaminathan

Quantitative and Structural Analysis of Real-Time and Probabilistic Systems

– Dissertation –

Department für Informatik
Carl von Ossietzky Universität Oldenburg
26111 Oldenburg
Thesis Defence on: 09.11.2015
Reviewers:
1. Prof. Dr. Martin Fränzle
2. Prof. Dr. Ir. Joost-Pieter Katoen

Abstract This dissertation contributes to the quantitative and structural analysis of real-time and probabilistic systems. The quantitative analysis herein goes beyond the classical boolean notion of system correctness, and is performed on system models that incorporate *perturbations*, *prices*, and *probabilities* in their behaviour. The structural analysis investigated in this dissertation entails reduction techniques for networks of real-time and probabilistic systems that exhibit *parallelism*. These four aspects (namely, perturbations, prices, probabilities, and parallelism) are analyzed on the following system models w.r.t variants of reachability properties:

- *perturbed timed automata* with drifting clocks and with clock resynchronization,
- *multi-priced timed automata* with a bounded budget,
- *networks of timed automata* extended with shared data variables,
- *networks of probabilistic automata* extended with shared data variables, and
- *perturbed probabilistic timed automata* with drifting clocks and with clock resynchronization.

Zusammenfassung Diese Dissertation trägt zur quantitativen und strukturellen Analyse von Realzeit- und probabilistischen Systemen bei. Für die quantitative Analyse berücksichtigen wir Systemmodelle, die *Ungenauigkeiten*, *Kosten* und *Wahrscheinlichkeiten* aufweisen, und dabei über die klassische "ja/nein"-Korrektheit hinausgehen. Für die strukturelle Analyse berücksichtigen wir Reduktionsansätze für Netzwerke von Realzeit- und probabilistischen Systemen, die *Nebenläufigkeit* aufweisen. Diese quantitative und strukturelle Analyse wird in Bezug auf (Varianten von) Erreichbarkeitseigenschaften auf den folgenden Systemmodellen durchgeführt:

- *Realzeitautomaten* mit *Ungenauigkeiten* in den Uhren und mit deren Resynchronization,
- *kostenbehafteten Realzeitautomaten* mit begrenztem Budget,
- *Netzwerken von Realzeitautomaten* mit geteilten Datenvariablen,
- *Netzwerken von probabilistichen Automaten* mit geteilten Datenvariablen, und
- *probabilistichen Realzeitautomaten* mit Ungenauigkeiten in den Uhren und mit deren Resynchronization.

Contents

1

Introduction

This dissertation contributes to the analysis of several classes of *real-time and probabilistic systems* along two axes - *quantitative* and *structural*, which contribute to the paradigms of "beyond yes/no" and "design meets verification", respectively, as detailed next.

The quantitative analysis herein goes beyond the classical *boolean* notion of system correctness, which rests on obtaining a simple "yes/no" answer to the question of whether a given system model (expressed typically as an automaton) satisfies a given set of properties (expressed typically as formulae in a specification logic). Such a "yes/no" answer is however often inadequate in practice for *comparing* systems - e.g., two systems S_1 and S_2 might both satisfy a given safety property by avoiding a set B of designated bad states. However, even the slightest change in the parameters of S_1 might cause a state in B to be reached, while S_2 might tolerate some changes to its parameters, and still avoid the states in B. Thus, while both S_1 and S_2 are identical in the classical boolean sense of avoiding the states in B, we see that S_2 "behaves better" than S_1 in practice. Such notions of better behaviour are formalized by automata models that admit the *quantitative analysis* of systems. In this dissertation, such quantitative analysis "beyond yes/no" is restricted to automata models that are strict sub-classes of *probabilistic hybrid automata* [Spr01], while the property specifications are confined to variants of *reachability*, whose verification is rendered easier by the *structural analysis* that this dissertation additionally considers.

In particular, we consider four aspects (the four "p"s) of such quantitative and structural analysis, namely *perturbations*, *prices*, *probabilities*, and *parallelism*, mainly in the setting of *continuous real-time*. We primarily look at extensions to the classical *timed automaton* [AD94] model that enable quantitative evaluation of the system's timing behaviour, with the exception of *probabilistic automata* [Seg00] that are investigated in the untimed setting. Timed and probabilistic automata are additionally investigated along the structural axis, by means of *transformations* that essentially aim at containing the state-space explosion arising from parallelism in a networked setting. The four "p's" of the quantitative and structural analysis considered in this dissertation are as detailed next:

- *Perturbations*: We investigate a notion of *robustness* that permits us to state (in the setting of continuous real-time) the conditions under which the system

S_2 discussed earlier is indeed better than S_1. To this end, we investigate *robust reachability* for *closed timed automata under perturbations*, whose clock-rates may drift by small amounts. We then ask the question of whether, for any given depth of iterations of the system's transition relation, there exists a suitably small clock-drift that (still) permits the system to avoid some bad state. By making the allowable drift dependent on the iteration depth, our notion of robustness (in contrast to related works) *discounts the system's future* [dAHM03], in the sense that an error in the system's behaviour is considered less serious as the system evolves. Such a discounted robustness notion yields the somewhat surprising result that the standard reachability analysis suffices to determine whether closed timed automata are robust against perturbations manifesting as drifting clocks. We next investigate a *clock resynchronization* scheme that *bounds* the drift in the clocks. This yields the result that the standard reachability analysis again suffices to determine whether closed timed automata are robust against (small enough) perturbations in the clock-rates, bounded by regular resynchronization. The robustness notion in this case however does not discount the system's future.

- *Prices*: We are concerned here with *cost optimization*, where we ask whether it is possible to be both *on time and within budget*. To this end, we look at extensions of timed automata whose guards and invariants are annotated with *prices* that capture budgetary information w.r.t staying in a location or taking an edge. This then permits the analysis and optimization of phenomena (such as scheduling) that are beyond the scope of timed automata. Such priced extensions to timed automata lie at the frontier between timed automata (for which reachability and related properties are decidable) and linear hybrid automata (for which such properties become undecidable, cf. [HKPV98]). Though the prices here exhibit richer dynamics than the clocks of timed automata, they may be neither queried nor reset while taking edges. Such an *observer restriction* on the prices leads to decidability of the *optimum reachability problem* for certain classes of priced timed automata, in particular for a single priced variable taking on both non-negative and negative values (owing to a priced refinement of the region-graph of timed automata, cf. [BBBR07]), and for multiple priced variables taking on only non-negative values (owing to a manipulation of priced extensions to the zones of timed automata, where termination relies on a well-quasi ordering of the symbolic states under liveness and cost-divergence assumptions on cycles of the underlying transition system, cf. [LR08]).
  Our contribution in this setting is an investigation of priced timed automata with multiple priced variables taking on both non-negative and negative values. To this end, we show that such *multi-priced timed automata* are as expressive as *stop-watch automata* under a (semantic) condition that the cost-optimization is restricted to *viable* paths of the underlying transition system, which respect (upper- and lower-) *bounds* on the priced variables corresponding to *budgetary constraints* in practice. As stop-watch automata are as expressive as linear hybrid automata [HKPV98], this then yields an *undecidability result* of the *cost-optimal reachability problem* for *bounded multi-priced timed automata*.
  However, when bounded multi-priced timed automata are additionally subject to a semantic condition of *cost-charging* on (quasi-) cyclic paths of the underlying transition system (similar to the cost divergence assumption in [LR08]), the cost-optimal reachability problem may be reduced to the solution of a linear

constraint system representing the path conditions over a finite number of viable paths of bounded length, thus yielding an effective decision procedure.
The cost bounds on viable paths are central to both the undecidability result and the decision procedure (under the additional cost-charging condition) discussed above. We leave open the investigation of multi-priced timed automata without cost-bounds on viable paths.

- *Probabilities:* Many *communication protocols* (such as the IEEE 801.11 WLAN MAC, cf. [KNPS06]) incorporate *probabilistic behaviour* for message exchanges among multiple stations sharing a common channel. The aim here is not to achieve absolute collision freedom, but rather an "efficient" means of transmitting messages. The performance measure for efficiency here is a *high probability* of successful message transmission, under some timing constraints.
  Probabilistic Timed Automata, cf. [KNSS02], which are timed automata annotated with discrete probability distributions on the edges, lend themselves as a natural formalism for modelling such protocols, whose performance is evaluated (among others) by the notion of *probabilistic timed reachability*. Our contribution here is an analysis of probabilistic timed automata under *perturbations on the clock-rates*. We attempt to show that *closed* probabilistic timed automata with perturbations on the clock-rates are robust w.r.t probabilistic timed reachability, under both a discounted notion of robustness, and under bounds on the clock-drifts resulting from regular resynchronization without discounting the system's future, yielding probabilistic counterparts to the analogous results for timed automata.
- *Parallelism:* In the untimed setting, *randomized distributed algorithms* (which solve fundamental problems in distributed computing such as consensus, mutual exclusion, and leader election by symmetry-breaking) are naturally modelled as (networks of) *probabilistic automata* [Seg00]. The communication between the processes participating in the (randomized) distributed algorithm is modelled primarily by *shared variables* that the processes may read from and write to. The design and analysis of randomized distributed algorithms is however highly non-trivial. This is mainly due to the fact that the stochastic process describing the evolution of a randomized distributed algorithm changes depending on the generally unknown scheduling policies and relative speeds of the individual process components, entailing a complex interplay between randomization and nondeterminism. The behaviour of such algorithms is typically expressed in terms of probabilistic temporal properties.
  This dissertation attempts to contribute to the easier verification of randomized distributed algorithms modelled as networks of probabilistic automata, by means of a *structural simplification* of the parallelism underlying the system's model. In particular, we attempt to simplify the reasoning of such algorithms by enriching probabilistic automata with the concept of *layering*. The main underlying idea is that the computations of randomized distributed algorithms often exhibit a sequential (i.e., layered) structure owing to the absence of dependencies between processes, which may then be exploited to *restructure* the algorithm's model such that the resulting restructuring exhibits a *reduced state-space*, while also enabling a *conceptual simplification* of the algorithm's underlying behaviour, owing to reduced parallelism. Such a layered transformation is investigated on an algorithm for *randomized mutual exclusion* by Kushilevitz and Rabin [KR92].

Luis Maria Ferrer Fioriti at Saarland University however pointed out that our layered transformation would not be sound when no restrictions are placed on the allowable resolutions of non-determinism in probabilistic automata. Moreover, there appear to technical issues concerning the necessary termination conditions and the class of properties preserved. Notwithstanding these issues, the applicability of our layered transformation to the randomized distributed algorithm of [KR92] was confirmed by Ian Larson under the supervision of Arpit Sharma at RWTH Aachen University, through an extensive experimental evaluation using the PRISM model checker for a wide range of probabilistic temporal properties [Sha15], thus demonstrating the "design meets verification" paradigm by means of reduced parallelism in the algorithm of [KR92] for randomized mutual exclusion.

The main contribution to the structural axis of this dissertation is an extensive analysis of *networks of timed automata extended with shared data-variables* by means of several structural transformations, including two variants of layering. In particular, we analyze the properties that are preserved (in parallel contexts) by the transformations of *separation*, *flattening*, and (independence-based and precedence-based) *layering*. The interplay of these transformations is demonstrated on an enhanced version of Fischer's protocol for *real-time mutual exclusion*, whose subsequent verification is rendered almost trivial, thus demonstrating the "design meets verification" paradigm by means of substantially reduced parallelism in extended timed automata networks.

The dissertation thus contributes to the (quantitative and/or structural) analysis of the following system models, and is based on (extended and revised versions of) the following (co-authored) refereed publications listed separately using a numeric citation style:

- *perturbed timed automata* with drifting clocks and with clock resynchronization, considered in Chapter 2 and published as [1, 2],
- *bounded multi-priced timed automata*, considered in Chapter 3 and published as [3],
- *networks of timed automata* extended with shared data variables, considered in Chapter 4 and published as [4, 6, 7],
- *networks of probabilistic automata* extended with shared data variables, considered in Chapter 5 and published as [5],
- *perturbed probabilistic timed automata* with drifting clocks and with clock resynchronization, considered in Chapter 6.

Chapters 5 and 6 dealing with probabilistic models contain results that have not yet been fully worked out. The dissertation concludes with Chapter 7, where a classification scheme is proposed for the models considered in this dissertation, within the unifying framework of *symbolic probabilistic systems* [KNS01].

Acknowledgements

This dissertation would not have come about were it not for the following individuals and institutions, whose help and support is hereby gratefully acknowledged:

- Prof. Dr. Martin Fränzle and Prof. Dr. Ernst-Rüdiger Olderog for their mentorship of the author over the years,

- Prof. Dr. Ir. Joost-Pieter Katoen for having introduced the author to probabilistic systems,
- Dr. Stephanie Kemper for having agreed to serve on the thesis committee,
- Ms. Andrea Göken, Mr. Jürgen Niehaus, and Ms. Ira Wempe for administrative support,
- colleagues at the research groups "Correct System Design" and "Hybrid Systems" for a pleasant atmosphere at work,
- friends and family of the author for support concerning personal issues, and
- the Deutsche Forschungsgemeinschaft for its funding through the Graduiertenkolleg "Trustsoft" (Trustworthy Software Systems) and the Sonderforschungsbereich "AVACS" (Automatic Verification and Analysis of Complex Systems).

About the author

Mani Swaminathan holds a Master's degree in Telecommunications from the Indian Institute of Technology, Delhi. From 2000 to 2005 he has held positions in industry and academia in India and Switzerland, including General Electric Global Research and the Ecole Polytechniqe Federale de Lausanne. Later on he was a PhD student in the DFG-funded Graduiertenkolleg Trustsoft at the University of Oldenburg, and a consultant in avionics and aerospace at Altran Deutschland. From November 2008 to December 2015, he was a research assistant at the Univesity of Oldenburg within the DFG-funded Sonderforschungsbereich AVACS (Automatic Verification and Analysis of Complex Systems).

2

Robustness of Closed Perturbed Timed Automata

2.1 Introduction

Real-time systems, which have strict timing requirements, have emerged as an enabling technology for several important application domains, and hence rigorous methods and techniques to ensure their correct functioning are of utmost importance. Timed Automata (TA) [AD94] have been extensively studied as a formalism for modelling real-time systems. TA extend finite automata by augmenting them with "clock" variables based on a dense-time model, which quantitatively capture the behaviour of the system with time. TA model checkers such as UPPAAL [BY04] and KRONOS [Yov97] have been successfully used in several industrial case studies, such as [LPY01].

A key result for the decidability properties of TA is the region-automaton construction [AD94], which partitions the inherently infinite state space of the TA into finitely many equivalence classes or "regions". The number of such regions is, however, exponential in the number of clocks, and the region construction is therefore not suited in practice for model checking TA when the number of clocks is large. Most available tools for model checking TA (such as UPPAAL) instead use on-the-fly algorithms that dynamically search through the state space of the TA, which is partitioned into "zones" [BY04]. Associated data structures such as Difference Bound Matrices (DBMs) [BY04] are used to represent zones in TA-based verification. Reachability analysis forms the core of such verification tools [ABBL03] and is implemented by a Forward Reachability Analysis (FRA) algorithm that computes the set of successors of a zone, with termination being enforced by zone-widening using k-normalization [BY04].

However, such analyses, whether region- or zone-based, assume that the clocks of the TA are perfectly synchronous, which is not the case in practice, where the clocks could drift by small amounts. It is shown in [Pur00, WDMR08] that the usual region-based analysis is not correct w.r.t. reachability when considering perturbations in the clocks, in the sense that an unsafe state, reported as unreachable for perfect clocks, might well be reachable by iterating often enough through a cycle in the TA, even when the clocks drift by infinitesimally small amounts, and such a TA is therefore not "robustly safe". This insight leads to the definition of robust reachability, where

a reachability property is considered to be "robustly (in-)valid" iff it does not change its validity for some small relative drift between clocks.

"Robust" reachability analysis [Pur00, WDMR08] therefore computes the set of states that are reachable for *every* (i.e., even the slightest) drift, reporting the (closed) TA as not being robustly safe iff that enlarged reach-set contains an unsafe state (where the guards and invariants of a closed TA are all closed, i.e., defined only by non-strict inequalities). This is made possible due to Theorem 15 of [WDMR08] that establishes an equivalence in the reach-set computation between the clock-drift and guard-enlargement models of perturbations in closed TA , where furthermore, each cycle of the TA is a *progress cycle*, wherein every clock is reset at least once per cycle. For such TA models, with maximum clock-drift parameterized by $\varepsilon > 0$, and with the maximum allowable guard-enlargement parameterized by $\Delta > 0$, and with the corresponding reachable state-spaces being $Reach^{\varepsilon}$ ($\varepsilon > 0$ clock-drift with no guard-enlargement), $Reach_{\Delta}$ ($\Delta > 0$ guard-enlargement with no clock-drift), and $Reach^{\varepsilon}{}_{\Delta}$ ($\varepsilon > 0$ clock-drift and $\Delta > 0$ guard-enlargement), Theorem 15 of [WDMR08] establishes that $\cap_{\varepsilon>0} Reach^{\varepsilon}$ has an empty intersection with the closed target state iff there exists some $\varepsilon > 0$ and $\Delta > 0$ such that the intersections of $Reach^{\varepsilon}$ and of $Reach_{\Delta}$ with the closed target state are again empty.

Robust safety of closed TA models (under the progress cycle assumption) in [Pur00, WDMR08] is thus based on computing on the reach-set $\cap_{\Delta>0} Reach_{\Delta}$, based on searching the strongly connected components of the region-graph, thus suffering from the exponential size of the region-graph in the number of clocks. Symbolic algorithms that compute this reach-set more efficiently are presented in [DK06, JR11, KLMP14, San15]. These symbolic algorithms manipulate *zones*, which are efficient polyhedral representations of the clock-constraints of TA. The symbolic algorithms in [DK06, JR11] works alternate between forward and backward analysis, and are applicable to closed TA, whose cycles need to satisfy –in addition to the previously mentioned progress cycle condition– a *flatness* condition, wherein each location of the TA is part of at most one cycle. The symbolic algorithm in [KLMP14] is applicable to non-flat closed TA with progress cycles, and augments the standard symbolic forward analysis (in TA model checkers such as UPPAAL) by a symbolic acceleration of all (progress) cycles. The works [JR11, San15] go beyond the previously considered *existential* version of the robust safety problem under guard-enlargement, by computing a *robustness margin*. In particular, the symbolic algorithms in [JR11, San15] *compute* additionally the *maximum allowable guard-enlargment* $\Delta_{max} > 0$ such that $Reach_{\Delta_{max}}$ yields an empty intersection with the closed target. While the symbolic computation of the robustness margin in [JR11] is applicable only to closed *flat* TA with progress cycles, the most recent work [San15] does away with this flatness restriction, while still requiring closedness and progress cycles. The computation of the robustness margin in [San15] (for closed non-flat TA with progress cycles) is however only a semi-algorithm, in the sense that termination is not always guaranteed. The work in [Dim07] considers TA that do admit *strict* inequalities, but again under the progress cycle assumption. Similar to the perturbation model considered in this chapter, the symbolic algorithm in [Dim07] decides (existential) robust safety of such TA against *drifting clocks*. In contrast to our analysis, which is purely forward-based as in UPPAAL, the symbolic algorithm in [Dim07] entails a combination of forward and backward analysis, as in [Pur00, WDMR08, DK06, JR11].

The region-based or symbolic (zone-based) algorithms in each of the works [Pur00, WDMR08, DK06, Dim07, JR11, KLMP14, San15] induce a performance overhead compared to the standard forward reahability analysis (FRA) algorithm used within tools like UPPAAL. All these works (except [Dim07]) assume that the guards, invariants, and targets of the TA are closed. Furthermore, all of them assume that each cycle of the TA is a progress cycle. The unsafe states that become reachable with drifting clocks (but which are unreachable with perfect clocks) are added to such robust reach-sets only by iterating an *unbounded* number of times through the (progress) cycles of the TA, essentially implying an *inifinite life-time* of the system. Moreover, the model of clock-drift considered in these works is one of *unbounded* relative drift between the clocks, which does not take into account the *regular resynchronization* of clocks that is performed in practical real-time systems. This chapter addresses these two issues, with two main contributions:

1. We first consider the model of clock-drift introduced by Puri [Pur00]. We show that, under the assumption of closed guards and invariants in the TA, the standard zone-based FRA of TA performed by tools such as UPPAAL is indeed exact when testing for robust safety of timed systems having an *arbitrary, but finite* life-time. We test here whether the TA can robustly avoid the target arbitrarily long, in the following sense: for any given number i of iterations of the transition relation, there is $\varepsilon_i > 0$ such that $Reach_i^{\varepsilon_i}$ has an empty intersection with the target state, where $Reach_i^{\varepsilon_i}$ is the reachable state space after i iterations of the transition relation under maximum perturbation ε_i of the clocks. Note that ε_i may depend on the number i of executed iterations, with ε_i decreasing (not necessarily strictly) with i, and potentially tending to 0 as i tends to ∞. Thus, robust safety under our notion does not imply the existence of a homogeneous $\varepsilon > 0$ that is independent of the number of iterations and such that $Reach^{\varepsilon}$ has an empty intersection with the target state (which is the notion considered in [Pur00, dWDMR04, Dim07]). However, our notion of robust safety implies avoidance of the target state by some strictly positive value of the perturbation for any *arbitrary, but finite* number of iterations. This is applicable to all systems having a finite life-time.
2. Next, we introduce a more realistic model of clock-drift that takes into account the *regular resynchronization* performed in practical real-time systems (such as bit-stuffing in communication protocols), which results in a *bounded* relative clock-drift. Under the assumption of closed guards, invariants, and targets, we show that the standard zone-based FRA of TA is again exact when testing for robust safety of such timed systems with clock resynchronization. In this case, a certification of robust safety imposes no restriction on the life-time of the system — it implies avoidance of the (closed) target by all $0 < \varepsilon < 1$ (where the ε now parameterizes the maximum relative bounded clock-drift subject to periodic resynchronization) independent of the number of iterations.

Section 2.2 of this chapter briefly reviews TA definitions and semantic, along with our assumptions. It also presents the standard algorithm for zone-based FRA. Section 2.3 describes the robustness problem for TA in the context of the model of clock-drift introduced in [Pur00], and shows the exactness of the standard zone-based FRA algorithm w.r.t robust safety for systems having a finite life-time. Section 2.4 then introduces our model of bounded clock-drift that accounts for regular clock resynchronization, and shows the exactness of the standard zone-based FRA algo-

rithm w.r.t robust safety, but now without any restrictions on the life-time of the system. Section 2.5 concludes this chapter. The results of this chapter have been previously published as [1, 2]. This chapter is an extended and revised version of the publication [2], completed with proofs of the results stated therein, with updated references to the literature that has appeared since its publication.

2.2 Timed Automata and Forward Analysis

Given a finite set C of *clocks*, a *clock valuation* over C is a map $v : C \to \mathbb{R}_{\geq 0}$ that assigns a non-negative real value to each clock in C. If n is the number of clocks, a clock valuation is basically a point in $\mathbb{R}^n_{\geq 0}$, which we henceforth denote by $\mathbf{u}, \mathbf{v}$ etc.

Definition 2.1. *A* zone *over a set of clocks C is a system of constraints defined by the grammar $g ::= x \triangleright d \mid x - y \triangleright d \mid g \wedge g$, where $x, y \in C$, $d \in \mathbb{N}$, and $\triangleright \in \{<, \leq, >, \geq\}$. The set of zones over C is denoted $Z(C)$.*

A *closed zone* is one in which $\triangleright \in \{\leq, \geq\}$, and we denote the set of closed zones over C by $Z_c(C)$. A zone with no bounds on clock differences (i.e., with no constraint of the form $x - y \triangleright d$) is said to be *diagonal-free*, and we denote the corresponding set of zones by $Z_d(C)$. The set $Z_{cd}(C)$ denotes zones that are *both closed and diagonal-free*. The set $Z_{cdU}(C)$ denotes the set of *closed, diagonal-free zones having no lower bounds on the clocks*.

Definition 2.2. *A TA is a tuple $A = (L, C, (l_0, \mathbf{0}), T, Inv)$, with*

- *a finite set L of* locations *and a finite set C of* clocks*, with $|C| = n$.*
- *An* initial *location $l_0 \in L$ together with the initial clock-valuation $\mathbf{0}$ where all clocks are set to 0* [1]
- *a set $T \subseteq L \times Z_{cd}(C) \times 2^C \times L$ of possible* transitions *between locations. A transition t between two locations (l, l') is denoted $l \xrightarrow{t} l'$, and involves a* guard *$G(t) \in Z_{cd}(C)$ and a* reset set *$Res_t \subseteq C$.*
- *$Inv : L \to Z_{cdU}(C)$ assigns* invariants *to locations*

In the sequel, we will denote by k the *clock ceiling* of the TA A under investigation, which is the largest constant among the constraints of A (including the predicate defining the unsafe state). Note that we assume that the guards of the automaton are *closed and diagonal-free zones*. Invariants in addition have only *upper-bounds* on clocks. Diagonal constraints of the form $x - y \triangleright d$ thus are not part of the TA syntax, but are of relevance, since they occur during the course of forward reachability analysis as a result of the *time-passage* operation defined as follows:

Definition 2.3. *For a clock valuation $\mathbf{x}$, its* time-passage *is $timepass(\mathbf{x}) = \{\mathbf{x} + d \mid d > 0\}$, where $\mathbf{x} + d$ denotes the addition of a strictly positive scalar d to each component of $\mathbf{x}$. This is canonically lifted to clock-zones Z as $timepass(Z) = \bigcup_{\mathbf{x} \in Z} timepass(\mathbf{x})$.*

[1] We assume without loss of generality that all clocks are initially set to 0.

Definition 2.4. $\lfloor \mathbf{x} \rfloor_k$ *denotes the* k*-region containing* $\mathbf{x}$*, which is the equivalence class induced by the* k-region-equivalence relation $\approx_k$*. For two clock valuations* $\mathbf{x}$ *and* $\mathbf{y}$*,* $\mathbf{x} \approx_k \mathbf{y}$ *iff*

$$\forall i \leq n : \left(\begin{array}{l} (x_i > k) \wedge (y_i > k) \\ \vee\ ((int(x_i) = int(y_i)) \wedge (fr(x_i) = 0 \Leftrightarrow fr(y_i) = 0) \wedge \\ \quad \forall j \leq n :\ (fr(x_i) \leq fr(x_j) \Leftrightarrow fr(y_i) \leq fr(y_j))) \end{array} \right)$$

Here, for a clock valuation $\mathbf{x} \in \mathbb{R}^n_{\geq 0}$, x_i denotes its i-th component, i.e., the value of the i-th clock, and $int(x_i)$ and $fr(x_i)$ respectively denote the integer and fractional parts of x_i.

Definition 2.5. *[Bou04] A* k-bounded zone (k-zone) *has no constant exceeding* k *among its constraints. For a zone* Z*, its* k-normalization*, denoted* $norm_k(Z)$*, is the smallest* k*-bounded zone containing* Z*.*

If Z is a k-zone, $norm_k(Z) = Z$. k is taken to be the largest constant appearing in the constraints (including the unsafe state) of the TA.

Definition 2.6. *Reach* $\subseteq L \times (C \to \mathbb{R}_{\geq 0})$ *is the reach-set of the TA* A*, consisting of an infinite set of (concrete) states of the TA of the form* $(l, \mathbf{x})$*, where* $l \in L$ *and* $\mathbf{x} \in \mathbb{R}^n_{\geq 0}$*. It is defined inductively as follows, with* $Reach_i$ *denoting the reach-set under* $i \in \mathbb{N}$ *steps, starting from the initial state* $(l_0, \mathbf{0})$ *and alternating between time-passage and discrete-location transitions:*[2]

- $Reach_0 = \{(l_0, \mathbf{0})\}$.
  - *if* i *even* $Succ(Reach_i) = \left\{ (l, \mathbf{x}) \middle| \begin{array}{l} \exists \mathbf{u} \in Inv(l) : (l, \mathbf{u}) \in Reach_i \\ \wedge\ \mathbf{x} \in timepass(\mathbf{u}) \cap Inv(l) \end{array} \right\}$
  - *if* i *odd* $Succ(Reach_i) = \left\{ (l, \mathbf{x}) \middle| \begin{array}{l} \exists t \in T, l' \in L, \mathbf{u} \in Inv(l') \cap G(t) : \\ \quad l' \xrightarrow{t} l \wedge (l', \mathbf{u}) \in Reach_i \\ \wedge\ \mathbf{x} \in Inv(l) \cap Res_t(\mathbf{u}) \end{array} \right\}$,
    where $Res_t(\mathbf{u})(c) = \mathbf{u}(c)$ *iff* $c \notin Res_t$*, else* $Res_t(\mathbf{u})(c) = 0$.
- $\forall i \geq 0, Reach_{i+1} = Reach_i \ \cup \ Succ(Reach_i)$.
- $Reach = \bigcup_{i \in \mathbb{N}} Reach_i$.

Reach is computed in tools like UPPAAL by the following zone-based forward reachability algorithm. Given a timed automaton A with the target (l, B), it decides whether $Reach \cap (l, B) \neq \emptyset$. Reachable state sets are represented by lists $\langle (l_1, Z_1), \ldots, (l_m, Z_m) \rangle$ of location-zone pairs. Let R_i denote the (symbolic) reachable state-space at the i-th ($i \geq 0$) iteration.

1. Start with the state-set $R_0 = \{(l_0, \mathbf{0})\}$, or equivalently, in DBM form, $R_0 = l_0 \times \{\bigwedge_{x \in C} x - x_0 \leq 0)\}$, where $x_0 \notin C$ is a pseudo-clock used to represent the constant 0.
2. For $i \geq 0$, compute the symbolic successors of R_i, denoted $Post(R_i)$, separately for even and odd values of i, as follows:

[2] To simplify the proofs, we use even- and odd-numbered steps to distinguish between time-passage (of possibly zero duration) and transitions between discrete locations.

- If i even, $Post(R_i) = \{(l, Z) \mid \exists (l, Z') \in R_i : Z = norm_k(timepass(Z')) \wedge Inv(l)\}$
- If i odd, $Post(R_i) = \{(l, Z) \mid \exists (l', Z') \in R_i, t \in T : l' \xrightarrow{t} l \wedge Z = Res_t(Z' \wedge G(t)) \wedge Inv(l)\}$

3. Build $R_{i+1} = R_i \cdot Post(R_i)$, where $\cdot$ denotes conditional concatenation that suppresses subsumed zones, i.e., removes (l, Z) if there is another (l, Z') with Z implying Z'.
4. Repeat steps (2) and (3) until $R_{i+1} = R_i$. Denote the last set R_i thus computed as R. Termination is guaranteed by the use of k-normalization, as there are only finitely many different k-zones such that only subsumed zones arise eventually.
5. Test whether $Z \wedge B$ is satisfiable for some $(l, Z) \in R$. If so then report "(l, B) is reachable", otherwise report "(l, B) is un-reachable".

It has been shown that this algorithm is sound and complete w.r.t. reachability [BY04] in the sense that $Reach \cap (l, B) = \emptyset$ iff $R \cap (l, B) = \emptyset$.

2.3 Robustness against drifting clocks under finite life-time

We have hitherto considered perfectly synchronous clocks. We now consider drifting clocks that could occur in practice, as introduced in [Pur00]. This phenomenon is modelled by introducing a parameter $\varepsilon > 0$ that characterizes the relative drift between the clocks. The slopes of the clocks are assumed to be within the range $\left[\frac{1}{1+\varepsilon}, 1+\varepsilon\right]$. This is equivalent to a relative drift in the range $\left[(\frac{1}{1+\varepsilon})^2, (1+\varepsilon)^2\right]$ between the clocks. We could alternatively consider the slopes to be in the range $[1-\varepsilon, 1+\varepsilon]$. The behaviour of both models w.r.t. infinitesimally small values of ε is identical, only that in our case, the slope of a clock never becomes negative no matter how large ε is. We then have a modification of the time-passage operation as follows:

Definition 2.7. *For a clock valuation* $\mathbf{x}$*, its* time-passage under perturbation of ε *is:* $timepass^{\varepsilon}(\mathbf{x}) = \left\{\mathbf{x} + d \cdot \mathbf{e} \,\middle|\, d > 0,\ \mathbf{e} \in \left[\frac{1}{1+\varepsilon}, 1+\varepsilon\right]^n\right\}$.
For a Zone Z*,* $timepass^{\varepsilon}(Z) = \bigcup_{\mathbf{x} \in Z}\ timepass^{\varepsilon}(\mathbf{x})$

While this model restricts the slopes of the clocks based on the value of parameter ε, the actual relative drift between the clocks increases without bound with increasing delay $d > 0$. The reachable state space also gets enlarged. For a given perturbation of ε, the corresponding *perturbed reach-set* $Reach^{\varepsilon}$ is defined inductively, similar to the non-perturbed case, by accounting for drifting clocks through the replacement of the deterministic $timepass()$ by an appropriate non-deterministic $timepass^{\varepsilon}()$ for steps corresponding to time-passage.

We now consider the effect of clock-drift on deciding whether some location-zone pair (l, B) is reachable. As an example (cf. Fig. 2.1), consider a timed automaton A, consisting of a single location l_0, two clocks x, y, the invariant of l_0 being $x \leq 2$, and a self-looping transition t consisting of a guard $x = 2?$, with the associated resets $x := 0, y := 0$. Let the unsafe state of A be characterized by $(l_0, B) = (l_0, y > 2)$. Assuming perfect clocks, the state-space of A is given by $Reach = (l_0, Z)$, where

$Z \equiv (x \leq 2 \wedge y = x)$, and A is clearly safe, as $Reach \cap (l_0, B) = \emptyset$. For drift characterized by a given $\varepsilon > 0$, the corresponding state space is $Reach^{\varepsilon} = (l_0, Z^{\varepsilon})$, where $Z^{\varepsilon} \equiv x \leq 2 \wedge \frac{x}{(1+\varepsilon)^2} \leq y \leq x(1+\varepsilon)^2$. Thus, $\forall \varepsilon > 0 : Reach^{\varepsilon} \cap (l_0, B) \neq \emptyset$ and A is therefore not "robustly" safe. The automaton along with the associated state-space for each case is illustrated in Fig. 2.1.

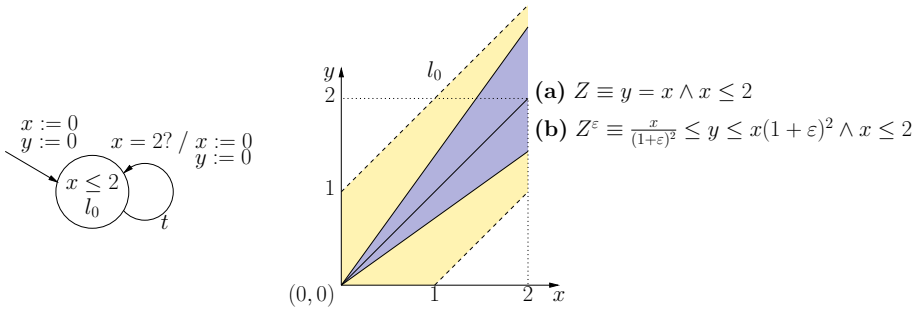

Fig. 2.1. A timed automaton A along with its state-spaces (a) without drift: (l_0, Z), (b) for a drift of ε : (l_0, Z^{ε}).

Related work on robust reachability of closed TA [Pur00, WDMR08, DK06, JR11, KLMP14, San15] compute the set $\cap_{\varepsilon>0} Reach^{\varepsilon}$ under the progress cycle condition. For this example, $\cap_{\varepsilon>0} Reach^{\varepsilon} = Reach$. This is because, for a zone Z, $\cap_{\varepsilon>0}\ timepass^{\varepsilon}(Z) = timepass(cl(Z))$, where $cl(Z)$ is the closure of Z, obtained by relaxing each strict inequality of Z to the corresponding non-strict one. In the present case, $Z \equiv \mathbf{0}$ is closed, as is $(Z \cup timepass(Z)) \cap Inv(l_0) \equiv y = x \wedge x \leq 2$, so $\cap_{\varepsilon>0} Reach^{\varepsilon} \cap (l_0, B) = \emptyset$. Hence, if open target states were allowed, the algorithms in [Pur00, WDMR08, DK06, JR11, KLMP14, San15] would all report this automaton as being robustly safe, while even the slightest perturbation would actually make the unsafe state reachable. However, if $B \equiv y \geq 2$, we see that the automaton of Fig. 2.1 is unsafe even with perfect clocks, while for $B \equiv y \geq 3$, the automaton is now safe even for drifting clocks, for all $0 < \varepsilon < \sqrt{1.5} - 1$.

We thus observe that *closed constraints* give consistent results while testing the automaton of Fig. 2.1 for safety, both with perfect clocks and under drift. Note also that this automaton has a single *progress cycle*, which additionally *resets all clocks simultaneously in a single transition.* The remit of this chapter is to formulate conditions under which tests on TA for robust safety give identical results for both perfect and drifting clocks. We define for this purpose a *grid-point* and its associated *neighbourhood* as follows:

Definition 2.8. *Grid denotes the set-of all* grid-points *in* $\mathbb{R}^n_{\geq 0}$*, i.e.,*
$Grid = \{\mathbf{x_g} \in \mathbb{R}^n_{\geq 0} \mid \forall 1 \leq i \leq n : fract(x_{gi}) = 0\}$*. For* $\mathbf{x} \in \mathbb{R}^n_{\geq 0}$*,*
$grid(\mathbf{x}) = \{\mathbf{x_g} \in Grid \mid dist(\mathbf{x}, \mathbf{x_g}) < 1\}$*, where*
$dist(\mathbf{x}, \mathbf{x_g}) = max_{1 \leq i \leq n} |x_i - x_{gi}|$*. The subset of Grid that contains only those grid-points bounded by* k *is denoted* $k - Grid$*.*

Thus, $\forall \mathbf{x_{kg}} \in k - Grid$: $\lfloor \mathbf{x_{kg}} \rfloor_k = \mathbf{x_{kg}}$. We will henceforth denote points in $Grid$ by the suffix g ($\mathbf{x_g}$ etc.) and points in $k - Grid$ by the suffix kg ($\mathbf{x_{kg}}$ etc.).

Definition 2.9. *For* $\mathbf{u_g} \in Grid$*, we define its* neighbourhood $N_k(\mathbf{u_g}) = \bigcap_{\varepsilon>0} \lfloor timepass^{\varepsilon}(\mathbf{u_g}) \rfloor_k$*. For a zone* Z*, its* neighbourhood *is defined as:* $N_k(Z) = \bigcup_{\mathbf{u_g} \in Z \cap Grid} N_k(\mathbf{u_g})$

$N_k(\mathbf{u_g})$ is the union of all *neighbouring* k-regions of $\mathbf{u_g}$, where a k-region r is said to *neighbour* $\mathbf{u_g}$ iff a point in r is reachable by time-passage from $\mathbf{u_g}$ for *every* drift, i.e., $\forall \varepsilon > 0 : timepass^{\varepsilon}(\mathbf{u_g}) \cap r \neq \emptyset$. Thus $N_k(\mathbf{u_g})$ is the result of adding to $\mathbf{u_g}$ all k-regions of Hausdorff distance 0 in temporally non-backward directions.

It must be understood here that the neighbourhood is defined only for grid-points[3]. It then follows that for any zone Z, $N_k(Z)$ is *idempotent*, i.e., $N_k(N_k(Z)) = N_k(Z)$, and that for a zone Z that has *no closed diagonal borders*, $N_k(Z)$ contains exactly the same grid-points as $norm_k(timepass(Z))$, and thus $N_k(Z) = norm_k(timepass(Z))$ for such a zone.

Also, $\forall \mathbf{u_g} \notin k - Grid : N_k(\mathbf{u_g}) = norm_k(timepass(\mathbf{u_g}))$. The following lemmas establish some useful properties of the neighbourhood operator.

Lemma 2.1. $\forall \mathbf{x} \in \mathbb{R}^n_{\geq 0}, \forall \mathbf{u_g} \in Grid :$
$\mathbf{x} \in N_k(\mathbf{u_g}) \Leftrightarrow \forall \varepsilon > 0 \;\; \exists \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : \mathbf{y} \in timepass^{\varepsilon}(\mathbf{u_g})$

Proof. Let $\mathbf{x}$ (resp. $\mathbf{u_g}$) be arbitrary points in $\mathbb{R}^n_{\geq 0}$ (resp. $Grid$).
$\mathbf{x} \in N_k(\mathbf{u_g})$
$\Rightarrow \mathbf{x} \in \bigcap_{\varepsilon>0} \lfloor timepass^{\varepsilon}(\mathbf{u_g}) \rfloor_k$
$\Rightarrow \forall \varepsilon > 0, \;\; \mathbf{x} \in \lfloor timepass^{\varepsilon}(\mathbf{u_g}) \rfloor_k$
$\Rightarrow \forall \varepsilon > 0, \;\; \exists \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : \mathbf{y} \in timepass^{\varepsilon}(\mathbf{u_g})$.
Conversely, $\mathbf{x} \notin N_k(\mathbf{u_g})$
$\Rightarrow \mathbf{x} \notin \bigcap_{\varepsilon>0} \lfloor timepass^{\varepsilon}(\mathbf{u_g}) \rfloor_k$
$\Rightarrow \exists \varepsilon > 0 : \mathbf{x} \notin \lfloor timepass^{\varepsilon}(\mathbf{u_g}) \rfloor_k$
$\Rightarrow \exists \varepsilon > 0 \;\; \forall \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : \mathbf{y} \notin timepass^{\varepsilon}(\mathbf{u_g})$. $\square$

Lemma 2.2. *For any* $\mathbf{u_g} \in Grid$*,* $N_k(\mathbf{u_g})$ *is given by:*

$$N_k(\mathbf{u_g}) = norm_k \left\{ \mathbf{u_g} + d + \sum_{i=1}^{n} a_i \cdot \mathbf{e_i} \;\mid\; d > 0, \;\; a_i \in [0,1) \right\},$$

where $\mathbf{e_i}$ *is the* i*-th unit vector.*

Here $\mathbf{u_g} + d$ denotes the addition of d to each component of $\mathbf{u_g}$.

Proof. We show that $N_k(\mathbf{u_g}) = N'(\mathbf{u_g})$, where
$N_k(\mathbf{u_g}) = \bigcap_{\varepsilon>0} \lfloor timepass^{\varepsilon}(\mathbf{u_g}) \rfloor_k$ and
$N'(\mathbf{u_g}) = norm_k \left\{ \mathbf{u_g} + d + \sum_{i=1}^{n} a_i \cdot \mathbf{e_i} \;\mid\; d > 0, \;\; a_i \in [0,1) \right\}$,
where $\mathbf{e_i}$ is the i-th unit vector

- Let $\mathbf{x} \in N'(\mathbf{u_g})$
  $\Rightarrow 0 \leq dist(\mathbf{x}, norm_k(timepass(\mathbf{u_g})) < 1$
  $\Rightarrow \forall \varepsilon > 0 \;\; \exists \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : \mathbf{y} \in timepass^{\varepsilon}(\mathbf{u_g})$
  [From the definitions of $\lfloor \mathbf{x} \rfloor_k$ and $timepass^{\varepsilon}(\mathbf{u_g})$]
  $\Rightarrow \mathbf{x} \in N_k(\mathbf{u_g})$ [From Lemma 2.1]
  $\Rightarrow N'(\mathbf{u_g}) \subseteq N_k(\mathbf{u_g})$

[3] By considering only closed guards and invariants for the automaton, we ensure that all the zones we encounter during FRA contain at least one grid-point.

- Let $\mathbf{x} \notin N'(\mathbf{u_g})$
  $\Rightarrow dist(\mathbf{x}, norm_k(timepass(\mathbf{u_g})) \geq 1$
  $\Rightarrow \exists \varepsilon > 0 \ \forall \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : \mathbf{y} \notin timepass^{\varepsilon}(\mathbf{u_g})$
  [From the definitions of $\lfloor \mathbf{x} \rfloor_k$ and $timepass^{\varepsilon}(\mathbf{u_g})$]
  $\Rightarrow \mathbf{x} \notin N_k(\mathbf{u_g})$ [From Lemma 2.1]
  $\Rightarrow N'(\mathbf{u_g}) \supseteq N_k(\mathbf{u_g})$

Thus $N'(\mathbf{u_g}) \subseteq N_k(\mathbf{u_g}) \ \wedge \ N'(\mathbf{u_g}) \supseteq N_k(\mathbf{u_g}) \quad \square$

This means that for any zone Z, $N_k(Z)$ is obtained as follows: First apply the standard unperturbed time-passage operator on Z, and then widen the *diagonal constraints which are non-strict inequalities* of the resulting conjunctive system by 1, to the *next higher strict inequalities*, i.e., $x - y \leq c$ is widened to $x - y < c + 1$, followed by standard k-normalization.

We first state the following property of any diagonal-free k-zone B [Bou04]:

Property 2.1. $\forall \mathbf{x}, \forall \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : \mathbf{x} \in B \Leftrightarrow \mathbf{y} \in B$.

This property is then used to prove the following lemma:

Lemma 2.3. *Given any (diagonal-free) k-zone Z, $\mathbf{x} \in \mathbb{R}^n_{\geq 0}$, $\mathbf{u_g} \in Grid$,* $\mathbf{x} \in N_k(\mathbf{u_g}) \cap Z \Leftrightarrow \forall \varepsilon > 0 \ \exists \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : \mathbf{y} \in timepass^{\varepsilon}(\mathbf{u_g}) \cap Z$

Proof. $\mathbf{x} \in N_k(\mathbf{u_g}) \cap Z$
$\Rightarrow \mathbf{x} \in \bigcap_{\varepsilon > 0} \lfloor timepass^{\varepsilon}(\mathbf{u_g}) \rfloor_k \cap Z$
$\Rightarrow \forall \varepsilon > 0, \ \mathbf{x} \in \lfloor timepass^{\varepsilon}(\mathbf{u_g}) \rfloor_k \cap Z$. Thus, from Property 2.1,
$\Rightarrow \forall \varepsilon > 0, \ \exists \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : \mathbf{y} \in timepass^{\varepsilon}(\mathbf{u_g}) \cap Z$
Conversely, $\mathbf{x} \notin N_k(\mathbf{u_g}) \cap Z$
$\Rightarrow \mathbf{x} \notin \bigcap_{\varepsilon > 0} \lfloor timepass^{\varepsilon}(\mathbf{u_g}) \rfloor_k \cap Z$
$\Rightarrow \exists \varepsilon > 0 : \mathbf{x} \notin \lfloor timepass^{\varepsilon}(\mathbf{u_g}) \rfloor_k \cap Z$. Thus from Property 2.1,
$\Rightarrow \exists \varepsilon > 0 \ \forall \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : \mathbf{y} \notin timepass^{\varepsilon}(\mathbf{u_g}) \cap Z \quad \square$

Lemma 2.4. *For any two* closed zones Z_1 *and* Z_2*,*

$$Z_1 \cap N_k(Z_2) = \emptyset \Leftrightarrow Z_1 \cap norm_k(timepass(Z_2)) = \emptyset$$

Proof. The proof of "$\Rightarrow$" is immediate, as $N_k(Z_2) \supseteq norm_k(timepass(Z_2))$. The proof of "$\Leftarrow$" is also obvious if $N_k(Z_2) = norm_k(timepass(Z_2))$.
When $N_k(Z_2) \supset norm_k(timepass(Z_2))$, we prove "$\Leftarrow$" as follows:
Z_1, Z_2 (and thus $timepass(Z_2)$, except for its "bottom") are closed. For $N_k(Z_2) \supset norm_k(timepass(Z_2))$, it must be the case that Z_2 is a k-zone, and so $norm_k(timepass(Z_2)) = timepass(Z_2)$ is also closed (except for its "bottom"). Thus, in order for Z_1 to have an empty intersection with $norm_k(timepass(Z_2))$, the two must be separated by a (max. norm) distance of at least 1. It also follows that the only additions to $norm_k(timepass(Z_2))$ to form $N_k(Z_2)$ are the open diagonal borders (obtained by relaxing the diagonal constraints of Z_2 by 1). These borders thus added being open, can at most *touch, but not intersect* Z_1, which entails our result. $\square$

Lemma 2.5. *For any* closed k-zone Z*, for any* $\mathbf{u_g} \in Grid$*, any* $\mathbf{v} \in \mathbb{R}^n_{\geq 0}$

$$\mathbf{v} \in Z \cap N_k(\mathbf{u_g}) \Rightarrow \exists \mathbf{v_g} \in (grid(\mathbf{v}) \cap Z \cap norm_k(timepass(\mathbf{u_g})))$$

Proof. $\exists \mathbf{v} \in Z \cap N_k(\mathbf{u_g})$
$\Rightarrow Z \cap N_k(u_g) \neq \emptyset$
$\Rightarrow Z \cap norm_k(timepass(u_g)) \neq \emptyset$ [Lemma 2.4]
$\Rightarrow \exists \mathbf{v_g} \in (grid(\mathbf{v}) \cap Z \cap norm_k(timepass(\mathbf{u_g})))$
[as closed zones always intersect in a grid-point, and from the definition of $grid(\mathbf{v})$]
□

This means that any closed guard (Z, referring to Lemma 2.5) that is enabled by a point ($\mathbf{v}$) obtained by time-passage from a grid-point ($\mathbf{u_g}$) under the smallest of drifts (and thus included into that point's ($\mathbf{u_g}$'s) neighbourhood) is also enabled by a different grid-point ($\mathbf{v_g}$) obtained by time-passage (without drift) from that grid-point ($\mathbf{u_g}$).

Lemma 2.6. *For any* closed, diagonal-free k-zone Z, *any* $\mathbf{x}, \mathbf{u} \in \mathbb{R}^n_{\geq 0}$,

$$\begin{aligned} &\mathbf{u} \in Z \ \ \wedge \ \ \forall \varepsilon > 0 : (\lfloor \mathbf{x} \rfloor_k \cap timepass^{\varepsilon}(\mathbf{u}) \cap Z) \neq \emptyset \\ \Rightarrow &\exists \mathbf{u_g} \in grid(\mathbf{u}) \cap Z \ \ \wedge \ \ \mathbf{x} \in N_k(\mathbf{u_g}) \cap Z \end{aligned}$$

The proof is immediate from Lemmas 2.3 and 2.5, and the definition of $grid(\mathbf{u})$.

Definition 2.10. *Let R_i^* be the reach-set at the i-th iteration, computed by modifying the time-passage steps of the standard FRA algorithm as follows: the $norm_k(timepass())$ operator is replaced by its neighbourhood $N_k()$. R_i^* is termed the corresponding* robust reach-set.

Let R^* be the robust reach-set that is ultimately computed by the FRA algorithm by using $N_k()$ instead of $norm_k(timepass())$, while computing the time-passage successors of zones [4], and R be the reach-set that is computed by the standard zone-based FRA (cf. Definition 2.6).
From Lemma 2.4, we get $R^* = \{(l, Z \cup (N_k(Z) \wedge Inv(l))) \mid (l, Z) \in R\}$, thereby resulting in the following corollary:

Corollary 2.1. *For any closed zone B and any $l \in L$,*
$R^* \cap (l, B) = \emptyset \Leftrightarrow R \cap (l, B) = \emptyset$.

We now establish useful properties of the sets R_i^* through the following lemmas.

Lemma 2.7. *Given any $i \in \mathbb{N}$, any $l \in L$, and any $\mathbf{x} \in \mathbb{R}^n_{\geq 0}$,*

$$(l, \mathbf{x}) \models R_i^* \Rightarrow \forall \varepsilon > 0 \ \ \exists \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : (l, \mathbf{y}) \in Reach_i^{\varepsilon}$$

Proof. We prove Lemma 2.7 by induction over i.

1. Base case $i = 0$: The proof is immediate as
   $\forall \varepsilon > 0 : R_0^* = Reach_0^{\varepsilon} = \langle (l_0, \mathbf{0}) \rangle$
2. Induction Hypothesis: Assume that Lemma 2.7 holds for some $i > 0$

[4] Termination is guaranteed for such an algorithm by the use of k-normalization in the computation of the neighbourhood $N_k()$ of zones encountered during the FRA.

3. Induction Step: We prove that Lemma 2.7 holds for $i+1$. Now,
$(l,\mathbf{x}) \models R^*_{i+1} \Leftrightarrow (l,\mathbf{x}) \models R^*_i \ \vee \ (l,\mathbf{x}) \models Post(R^*_i)$

- $(l,\mathbf{x}) \models R^*_i$ The proof follows from the Induction Hypothesis, as $Reach^\varepsilon_i \subseteq Reach^\varepsilon_{i+1}$
- $(l,\mathbf{x}) \models Post(R^*_i)$
  - i even: $\exists \mathbf{u_g} \in Inv(l) : (l,\mathbf{u_g}) \models R^*_i \wedge \mathbf{x} \in N_k(\mathbf{u_g}) \cap Inv(l)$ [5]
    a) : For $\mathbf{u_g} \in k-Grid$, we have:
    $\exists \mathbf{u_g} \in Inv(l) \ \forall \varepsilon > 0 \ \exists \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : (l,\mathbf{u_g}) \in Reach^\varepsilon_i$
    $\wedge \ \ \mathbf{y} \in timepass^\varepsilon(\mathbf{u_g}) \cap Inv(l)$
    [Induction Hypothesis, as $\lfloor \mathbf{u_g} \rfloor_k = \mathbf{u_g}$ for $\mathbf{u_g} \in k-Grid$, and using Lemma 2.3]
    $\Rightarrow \forall \varepsilon > 0 \ \exists \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : \mathbf{y} \in Succ(Reach^\varepsilon_i)$
    $\Rightarrow \forall \varepsilon > 0 \ \exists \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : \mathbf{y} \in Reach^\varepsilon_{i+1}$
    b) For $\mathbf{u_g} \notin k-Grid$, $N_k(\mathbf{u_g}) = norm_k(timepass(\mathbf{u_g}))$ and thus
    $\mathbf{x} \in N_k(\mathbf{u_g}) \Rightarrow \forall \varepsilon > 0 \ \forall \mathbf{w} \in \lfloor \mathbf{u_g} \rfloor_k \ \ \exists \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k :$
    $\mathbf{y} \in timepass^\varepsilon(\mathbf{w})$.....(1) Thus,
    $\exists \mathbf{u_g} \in Inv(l) : (l,\mathbf{u_g}) \models R^*_i \wedge \mathbf{x} \in N_k(\mathbf{u_g}) \cap Inv(l)$
    $\Rightarrow \exists \mathbf{w} \in Inv(l) \ \forall \varepsilon > 0 \ \exists \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : (l,\mathbf{w}) \in Reach^\varepsilon_i$
    $\wedge \mathbf{y} \in timepass^\varepsilon(\mathbf{w}) \cap Inv(l)$
    [from Induction Hypothesis, Property 2.1, and (1) previously]
    $\Rightarrow \forall \varepsilon > 0 \ \exists \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : \mathbf{y} \in Succ(Reach^\varepsilon_i)$
    $\Rightarrow \forall \varepsilon > 0 \ \exists \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : \mathbf{y} \in Reach^\varepsilon_{i+1}$
  - i odd: $\exists t \in T \ \exists l' \in L : l' \xrightarrow{t} l, \ \exists \mathbf{u} \in Inv(l') \cap G(t) : (l',\mathbf{u}) \models R^*_i$
    $\wedge \ \ \mathbf{x} \in Inv(l) \cap Res_t(\mathbf{u})$
    $\Rightarrow \exists t \in T \ \exists l' \in L : l' \xrightarrow{t} l, \ \exists \mathbf{u} \in Inv(l') \cap G(t)$
    $\wedge \forall \varepsilon > 0 \ \exists \mathbf{w} \in \lfloor \mathbf{u} \rfloor_k : (l',\mathbf{w}) \in Reach^\varepsilon_i \ \ \wedge \ \ \mathbf{x} \in Inv(l) \cap Res_t(\mathbf{u})$
    [Induction Hypothesis]
    $\Rightarrow \forall \varepsilon > 0 \ \exists \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k \ \exists t \in T \ \exists l' \in L : l' \xrightarrow{t} l, \exists \mathbf{w} \in Inv(l') \cap G(t)$
    $: (l',\mathbf{w}) \in Reach^\varepsilon_i \wedge \mathbf{y} \in Inv(l) \cap Res_t(\mathbf{w})$
    [Property 2.1]
    $\Rightarrow \forall \varepsilon > 0 \ \exists \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : \mathbf{y} \in Succ(Reach^\varepsilon_i)$
    $\Rightarrow \forall \varepsilon > 0 \ \exists \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : \mathbf{y} \in Reach^\varepsilon_{i+1}$ $\square$

Here, by $(l,\mathbf{x}) \models R^*_i$, we mean that there exists a zone $Z \in R^*_i$ such that $\mathbf{x} \in Z$. This lemma shows that the set R^*_i collects the regions that can be "touched" in the sense of some (but not necessarily all) points within being reachable for *every* perturbation.

Lemma 2.8. *For any $l \in L$, any diagonal-free k-zone B, any $i \in \mathbb{N}$,*
$R^*_i \cap (l,B) \neq \emptyset \Rightarrow \forall \varepsilon > 0 : \ \ Reach^\varepsilon_i \cap (l,B) \neq \emptyset$. [6]

Proof. $R^*_i \cap (l,B) \neq \emptyset$
$\Rightarrow \exists \mathbf{x} \in B : (l,\mathbf{x}) \models R^*_i$
$\Rightarrow \exists \mathbf{x} \in B \ \forall \varepsilon > 0 \ \exists \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : (l,\mathbf{y}) \in Reach^\varepsilon{}_i$ [Lemma 2.7]
$\Rightarrow \forall \varepsilon > 0 \ \exists \mathbf{y} \in B : (l,\mathbf{y}) \in Reach^\varepsilon_i$ [Property 2.1]
$\Rightarrow \forall \varepsilon > 0 : Reach^\varepsilon_i \cap (l,B) \neq \emptyset$ $\square$

[5] Note that $\mathbf{u_g} \in Grid$
[6] Here $R^*_i \cap (l,B) \neq \emptyset$ denotes $Z \wedge B$ being satisfiable for some $(l,Z) \in R^*_i$.

The above lemma implies that at any iteration depth i, if the set R_i^* intersects with a target state, then the corresponding perturbed reach-set under even the smallest of perturbations likewise intersects with the target state.

Lemma 2.9. *For any* even i, $l \in L$, $\mathbf{u_g} \in Grid \cap Inv(l)$, $\mathbf{v} \in \mathbb{R}^n_{\geq 0}$,

$$\begin{aligned}&(l, \mathbf{u_g}) \models R_i^* \wedge \exists l' \in L \ \exists t \in T : l \xrightarrow{t} l' \wedge \ \mathbf{v} \in N_k(\mathbf{u_g}) \cap Inv(l) \cap G(t)\\ \Rightarrow\ &\exists \mathbf{v_g} \in norm_k(timepass(\mathbf{u_g})) \cap grid(\mathbf{v}) \cap Inv(l) \cap G(t) :\\ &\quad \exists \mathbf{w_g} \in (Inv(l') \cap Res_t(\mathbf{v_g}) : (l', \mathbf{w_g}) \models R_{i+2}^*\end{aligned}$$

The proof is immediate from Lemma 2.5 and the definition of R_i^*. Here we assume, in addition to the guards and invariants being closed and diagonal-free, the following condition of *admissible target locations*, which ensures consistency between the invariants of a location and the guards of the transitions entering and leaving that location:
For any locations l and l', and any transition t with $l \xrightarrow{t} l'$:
$Inv(l) \cap G(t) \neq \emptyset \ \wedge \ Inv(l') \cap G(t) \neq \emptyset$.

Lemma 2.10. *For any* even i, *any* $l \in L$, *any* $\mathbf{x} \in \mathbb{R}^n_{\geq 0}$,
$\forall \mathbf{u_g} \in Grid \cap Inv(l) : (l, \mathbf{u_g}) \models R_i^*, \ \mathbf{x} \notin N_k(\mathbf{u_g}) \cap Inv(l)$
$\Rightarrow \exists \varepsilon_i > 0 \ \forall \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k, \forall \mathbf{u} \in Inv(l) : (l, \mathbf{u}) \in Reach_i^{\varepsilon_i}$:
$\mathbf{y} \notin timepass^{\varepsilon_i}(\mathbf{u}) \cap Inv(l)$

We prove Lemma 2.10 by contraposition:

Contraposition of Lemma 2.10

For any *even* i, any $l \in L$, any $\mathbf{x} \in \mathbb{R}^n_{\geq 0}$,
$\forall \varepsilon > 0 \ \exists \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k \ \exists \mathbf{u} \in Inv(l) : (l, \mathbf{u}) \in Reach_i^{\varepsilon} \wedge \mathbf{y} \in timepass^{\varepsilon}(\mathbf{u}) \cap Inv(l)$
$\Rightarrow \exists \mathbf{u_g} \in Grid \cap Inv(l) : (l, \mathbf{u_g}) \models R_i^* \ \wedge \ \mathbf{x} \in N_k(\mathbf{u_g}) \cap Inv(l)$

Proof. We prove by induction over even i

1. Base case $i = 0$:The proof follows from Lemma 2.3, and as
   $\forall \varepsilon > 0 : R_0^* = Reach_0^{\varepsilon} = \langle (l_0, \mathbf{0}) \rangle$
2. Induction Hypothesis: Assume that (the contraposition of) Lemma 2.10 holds for some $i > 0$
3. Induction Step: We show that (the contraposition of) Lemma 2.10 holds for $i + 2$. Now,
   $\forall \varepsilon > 0 \ \exists \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k \ \exists \mathbf{u} \in Inv(l) : (l, \mathbf{u}) \in Reach_{i+2}^{\varepsilon}$
   $\wedge \ \mathbf{y} \in timepass^{\varepsilon}(\mathbf{u}) \cap Inv(l)$
   $\Rightarrow \forall \varepsilon > 0 \ \exists \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k \ \exists l' \in L \ \exists t \in T : l' \xrightarrow{t} l \ \exists \mathbf{v} \in Inv(l') :$
   $(l', \mathbf{v}) \in Reach_i^{\varepsilon} \ \exists \mathbf{w} \in timepass^{\varepsilon}(\mathbf{v}) \cap Inv(l') \cap G(t)$
   $\exists \mathbf{u} \in Res_t(\mathbf{w}) \cap Inv(l) : \mathbf{y} \in timepass^{\varepsilon}(\mathbf{u}) \cap Inv(l)$ [definition of $Reach_i^{\varepsilon}$]
   $\Rightarrow \forall \varepsilon > 0 \ \exists \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k \ \exists l' \in L, \exists t \in T : l' \xrightarrow{t} l \ \exists \mathbf{v_g} \in Inv(l') \cap Grid$
   $: (l', \mathbf{v_g}) \models R_i^* \ \exists \mathbf{w} \in N_k(\mathbf{v_g}) \cap Inv(l') \cap G(t) \ \exists \mathbf{u} \in Res_t(\mathbf{w}) \cap Inv(l)$
   $: \mathbf{y} \in timepass^{\varepsilon}(\mathbf{u}) \cap Inv(l)$
   [Induction Hypothesis and Lemma 2.6]
   $\Rightarrow \exists l' \in L \ \exists t \in T : l' \xrightarrow{t} l \ \exists \mathbf{v_g} \in Inv(l') \cap Grid : (l', \mathbf{v_g}) \models R_i^*$

$\exists \mathbf{w_g} \in norm_k(timepass(\mathbf{v_g})) \cap Inv(l') \cap G(t) \cap Grid$
$\exists \mathbf{u_g} \in Res_t(\mathbf{w_g}) \cap Inv(l) : \mathbf{x} \in N_k(\mathbf{u_g}) \cap Inv(l)$
[From Lemma 2.5 and Lemma 2.9]
$\Rightarrow \exists \mathbf{u_g} \in Grid \cap Inv(l) : (l, \mathbf{u_g}) \models R^*_{i+2} \ \wedge \ \mathbf{x} \in N_k(\mathbf{u_g}) \cap Inv(l) \quad \square$

A consequence is that the following converse of Lemma 2.7 also holds:

Lemma 2.11. *Given any $i \in \mathbb{N}$, any $l \in L$, and any $\mathbf{x} \in \mathbb{R}^n_{\geq 0}$,*

$$(l, \mathbf{x}) \not\models R^*_i \Rightarrow \exists \varepsilon_i > 0 \ \forall \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : \ (l, \mathbf{y}) \notin Reach^{\varepsilon_i}_i$$

Proof. 1. Base case $i = 0$: The proof is immediate as
$\forall \varepsilon > 0 : R^*_0 = Reach^{\varepsilon}_0 = \langle (l_0, \mathbf{0}) \rangle$
2. Induction Hypothesis: Assume that Lemma 2.11 holds for some $i > 0$
3. Induction Step: We show that Lemma 2.11 holds for $i + 1$. Now,
$(l, \mathbf{x}) \not\models R^*_{i+1} \Leftrightarrow (l, \mathbf{x}) \not\models R^*_i \ \wedge \ (l, \mathbf{x}) \not\models Post(R^*_i)$

- i even: $(l, \mathbf{x}) \not\models Post(R^*_i) \Rightarrow \forall \mathbf{u_g} \in Grid \cap Inv(l)$
$: (l, \mathbf{u}) \models R^*_i \ \ \mathbf{x} \notin timepass(\mathbf{u}) \cap Inv(l)$
$\Rightarrow \exists \varepsilon_i > 0 \ \forall \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k \ \ \forall \mathbf{u} \in Inv(l)$
$: (l, \mathbf{u}) \in Reach^{\varepsilon_i}_i \ \ \mathbf{y} \notin timepass^{\varepsilon_i}(\mathbf{u}) \cap Inv(l)$ [Lemma 2.10]
$\Rightarrow \exists \varepsilon_i > 0 \ \forall \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : (l, \mathbf{y}) \notin Succ(Reach^{\varepsilon_i}_i)$—(1)
Thus $(l, \mathbf{x}) \not\models R^*_{i+1} \Rightarrow (l, \mathbf{x}) \not\models R^*_i \ \wedge \ (l, \mathbf{x}) \not\models Post(R^*_i)$
$\Rightarrow \exists \delta_i > 0 \ \forall \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : \ (l, \mathbf{y}) \notin Reach^{\delta_i}_i$
$\wedge \ \exists \varepsilon_i > 0 \ \forall \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : (l, \mathbf{y}) \notin Succ(Reach^{\varepsilon_i}_i)$
[Induction Hypothesis and the preceding (1)]
Let $\varepsilon_{i+1} = min\{\delta_i, \varepsilon_i\}$. As $Reach^{\varepsilon}_i$ is downward closed w.r.t ε,
$\exists \varepsilon_{i+1} > 0 \ \forall \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : (l, \mathbf{y}) \notin Reach^{\varepsilon_{i+1}}_{i+1}$
- i odd: $(l, \mathbf{x}) \not\models Post(R^*_i)$
$\Rightarrow \forall t \in T \ \forall l' \in L : l' \xrightarrow{t} l \ \forall \mathbf{u} \in Inv(l') \cap G(t)$
$: (l', \mathbf{u}) \models R^*_i : \mathbf{x} \notin Inv(l) \cap Res_t(\mathbf{u_g})$
$\Rightarrow \forall t \in T \ \forall l' \in L : l' \xrightarrow{t} l \ \forall \mathbf{u} \in Inv(l') \cap G(t)$
$: \forall \varepsilon > 0 \ \exists \mathbf{w} \in \lfloor \mathbf{u} \rfloor_k : (l', \mathbf{w}) \in Reach^{\varepsilon}_i : \mathbf{x} \notin Inv(l) \cap Res_t(\mathbf{u})$
[Lemma 2.7]
$\Rightarrow \forall \varepsilon > 0 \ \forall \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k \ \forall t \in T \ \forall l' \in L : l' \xrightarrow{t} l \ \forall \mathbf{w} \in Inv(l') \cap G(t)$
$: (l', \mathbf{w}) \in Reach^{\varepsilon}_i : \mathbf{y} \notin Inv(l) \cap Res_t(\mathbf{u})$ [Property 2.1]
$\Rightarrow \forall \varepsilon > 0 \ \forall \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : (l, \mathbf{y}) \notin Succ(Reach^{\varepsilon}_i)$
Thus, $(l, \mathbf{x}) \not\models R^*_{i+1} \Rightarrow (l, \mathbf{x}) \not\models R^*_i \ \wedge \ (l, \mathbf{x}) \not\models Post(R^*_i)$
$\Rightarrow \exists \varepsilon_i > 0 \ \forall \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : \ (l, \mathbf{y}) \notin Reach^{\varepsilon_i}_i$
$\wedge \ \forall \varepsilon > 0 \ \forall \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : (l, \mathbf{y}) \notin Succ(Reach^{\varepsilon}_i)$
Let $\varepsilon_{i+1} = \varepsilon_i$. It then follows that
$\exists \varepsilon_{i+1} > 0 \ \forall \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : (l, \mathbf{y}) \notin Reach^{\varepsilon_{i+1}}_{i+1} \quad \square$

Lemma 2.12. *For any $l \in L$, $i \in \mathbb{N}$, any diagonal-free k-zone B,*

$$R^*_i \cap (l, B) = \emptyset \Rightarrow \exists \varepsilon_i > 0 : Reach^{\varepsilon_i}_i \cap (l, B) = \emptyset.$$

Proof. $R^*_i \cap (l, B) = \emptyset$
$\Rightarrow \forall \mathbf{x} \in B : (l, \mathbf{x}) \not\models R^*_i$
$\Rightarrow \forall \mathbf{x} \in B \ \ \exists \varepsilon_i > 0 \ \forall \mathbf{y} \in \lfloor \mathbf{x} \rfloor_k : (l, \mathbf{y}) \notin Reach^{\varepsilon_i}_i$ [Lemma 2.11]
$\Rightarrow \exists \varepsilon_i > 0 \ \forall \mathbf{y} \in B : (l, \mathbf{y}) \notin Reach^{\varepsilon_i}_i$ [Property 2.1]
$\Rightarrow \exists \varepsilon_i > 0 : Reach^{\varepsilon_i}_i \cap (l, B) = \emptyset \quad \square$

The preceding lemma implies that at any iteration depth i, the set R_i^* does not intersect with a target state iff there exists a strictly positive value of the perturbation, such that the corresponding perturbed reach-set at that iteration depth likewise avoids the target state. The following corollary is then a direct consequence of Lemmas 2.8 and 2.12.

Corollary 2.2. *Given any $l \in L$, any diagonal-free k-zone B,* $R^* \cap (l, B) = \emptyset \Leftrightarrow \forall i \in \mathbb{N}\ \exists \varepsilon_i > 0 : Reach_i^{\varepsilon_i} \cap (l, B) = \emptyset$

Corollaries 2.1 and 2.2 lead us to the following theorem, which is a main result of this chapter.

Theorem 2.1. *Let R be the final reach-set computed by the standard zone-based FRA, for a TA with closed and diagonal-free guards and invariants. Then for any closed and diagonal-free k-zone B and any $l \in L$,* $R \cap (l, B) = \emptyset \Leftrightarrow \forall i \in \mathbb{N}\ \exists \varepsilon_i > 0 : Reach_i^{\varepsilon_i} \cap (l, B) = \emptyset$

It follows from this theorem that the standard zone-based FRA used in tools like UPPAAL is exact (sound and complete) while testing TA with closed guards and invariants for robust safety against closed targets.

The "$\Leftarrow$" part of Theorem 2.1 states that a closed target is reported as reachable by standard zone-based FRA only if it is also reachable in a finite number of iterations of the transition relation of the TA, under even the slightest of perturbations. This result is intuitively obvious, because even the smallest perturbed reach-set is a strict superset of its non-perturbed version. The "$\Rightarrow$" part of Theorem 2.1 states that a closed target is reported as unreachable by zone-based FRA only if for any given number of iterations i of the transition relation, there exists a strictly positive value of the perturbation ε_i that the automaton can tolerate and yet remains safe, in the sense that the corresponding perturbed reach-set $Reach_i^{\varepsilon_i}$ has an empty intersection with the (closed) target state. It must be noted here that this does not mean the existence of a homogeneous $\varepsilon > 0$ independent of the number of iterations, for which the unsafe state can be avoided, which is the notion considered in related works [Pur00, WDMR08, DK06, Dim07, JR11, KLMP14, San15]. Rather, as mentioned in this chapter's introduction, the magnitude of the tolerated perturbation ε_i could (but not necessarily) decrease with the number i of iterations, with ε_i potentially tending to 0 as i tends to ∞[7]. However, so long as we execute an *arbitrary, but finite number* of iterations, we are guaranteed a positive value of the tolerable perturbation for robust safety.

The analyses in [Pur00, WDMR08, DK06, Dim07], on the other hand, add states that can be reached in any (unbounded) number of iterations through the (progress) cycles of the automaton[8], for even the slightest perturbation. Therefore, a state

[7] For closed TA in which each cycle has at least one transition that resets all clocks simultaneously, the robust reach-sets computed by the algorithms in [Pur00, WDMR08, DK06, Dim07, JR11, KLMP14, San15] coincide with the standard reach-set computed by UPPAAL, as seen in the automaton of Fig. 2.1. Thus, a certificate of safety by standard UPPAAL for such TA w.r.t. closed targets implies a robust safety margin independent of iteration depth.

[8] We make no assumption on the cycles of the automaton.

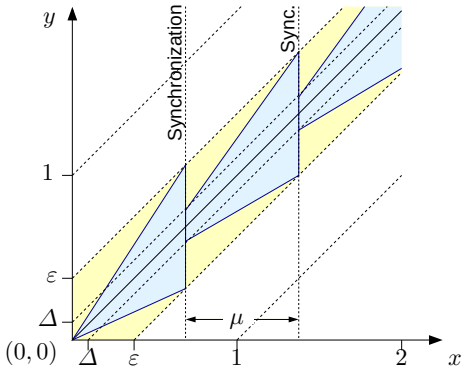

Fig. 2.2. Periodic resynchronization resulting in a bound ε on relative drift between clocks

$(l, \mathbf{x})$ is considered to be *robustly unreachable* in our sense (i.e., not included in R^*), but reachable in the sense of the works in [Pur00, dWDMR04, DK06, Dim07] iff $\lim_{\varepsilon \to 0} \min\{i \in \mathbb{N} \mid (l, \mathbf{x}) \in Reach_i^{\varepsilon} \ \} = \infty$.

2.4 Robustness against drifting clocks under resynchronization

In the previous section, we considered a model of drifting clocks where the relative drift between the clocks increases without bound with the passage of time, although the clock-slopes are themselves bounded according to the parameter ε. This is, however, rarely the case in practice, where the clocks, though subject to drift, are *regularly resynchronized* by diverse means, ranging from bit-stuffing in communication protocols to high-level clock synchronisation schemes. A parameter Δ characterizes the *post-synchronization-gap* and a parameter μ the longest possible gap between synchronizations. If the slopes of the clocks (w.r.t absolute time) are in the range $\left[\frac{1}{1+\theta}, 1+\theta\right]$ between synchronizations, such a resynchronization enforces a *uniform bound* given by

$$\varepsilon = \max\left(\Delta + \mu\left(\frac{1}{1+\theta}\right)^2, \Delta + \mu(1+\theta)^2\right) = \Delta + \mu(1+\theta)^2$$

on the relative drift between the clocks, irrespective of the extent of time-passage. The phenomenon is illustrated for two clocks x and y in Fig. 2.2. Throughout this section, we assume $0 < \varepsilon < 1$.

We incorporate such a resynchronization into TA by associating a *drift-offset* $\boldsymbol{\delta} \in [-\varepsilon, \varepsilon]^n$ for each clock valuation $\mathbf{x} \in \mathbb{R}^n_{\geq 0}$. This drift-offset keeps track of the extent to which the individual clocks in $\mathbf{x}$ have deviated from an implicit reference clock maintained by the synchronization scheme. The states of a TA in this semantics are thus tuples $(l, \mathbf{x}, \boldsymbol{\delta}) \in L \times \mathbb{R}^n_{\geq 0} \times [-\varepsilon, \varepsilon]^n$. As the deviation $\boldsymbol{\delta}$ is controlled by

the synchronization scheme such that it always remains below ε, the (perturbed) time-passage under synchronization is as follows:

Definition 2.11. *Given any* $\mathbf{x} \in \mathbb{R}^n_{\geq 0}$, *any* $\boldsymbol{\delta} \in [-\varepsilon, \varepsilon]^n$,

$$timepass^{\varepsilon}_{sync}(\mathbf{x}, \boldsymbol{\delta}) = \{(\mathbf{x'}, \boldsymbol{\delta'}) \mid \boldsymbol{\delta'} \in [-\varepsilon, \varepsilon]^n \wedge \exists d > 0 : \mathbf{x'} = \mathbf{x} - \boldsymbol{\delta} + d + \boldsymbol{\delta'}\}$$

A *run* of a perturbed TA subject to clock synchronization with accuracy ε is a sequence $\langle (l_0, \mathbf{x}_0, \boldsymbol{\delta}_0), (l_1, \mathbf{x}_1, \boldsymbol{\delta}_1), \ldots \rangle$ of states such that

1. l_0 is the initial location and $\mathbf{x}_0 = \boldsymbol{\delta}_0 = \mathbf{0}$,
2. For even i, $l_{i+1} = l_i$, $\mathbf{x}_{i+1} \in Inv(l_i)$
   $\wedge\ (\mathbf{x}_{i+1}, \boldsymbol{\delta}_{i+1}) \in \{(\mathbf{x_i}, \boldsymbol{\delta_i})\} \cup timepass^{\varepsilon}_{sync}(\mathbf{x}_i, \boldsymbol{\delta}_i)$ [9]
3. For odd i, $\exists t_i \in T : l_i \xrightarrow{t_i} l_{i+1} : \mathbf{x}_i \in Inv(l_i) \cap G(t_i)$,
   $\mathbf{x}_{i+1} \in Inv(l_{i+1}) \cap Res_{t_i}(\mathbf{x}_i)$, $\boldsymbol{\delta}_{i+1} = Res_{t_i}(\boldsymbol{\delta}_i)$.

Due to memorizing the current deviation $\boldsymbol{\delta}$ and adjusting it consistently to the constraint that the overall accuracy is better than ε, this semantics is subtly more constrained than the —superficially similar— semantics permitting an arbitrarily directed ε-deviation upon every time passage.

$SReach^{\varepsilon}$ is the corresponding perturbed reach-set, defined inductively as follows, with $SReach^{\varepsilon}_i$ denoting the perturbed reach-set in $i \in \mathbb{N}$ steps, starting from the initial state $(l_0, \mathbf{0}, \mathbf{0})$ and alternating between (perturbed) time-passage and (exact) discrete-location transitions:

- $SReach^{\varepsilon}_0 \equiv \{(l_0, \mathbf{0}, \mathbf{0})\}$
- For i even, $Succ(SReach^{\varepsilon}_i) = \{(l, \mathbf{x}, \boldsymbol{\delta}) \mid \mathbf{x} \in Inv(l)$
  $\wedge\ \exists \mathbf{x'} \in Inv(l), \exists \boldsymbol{\delta'} \in [-\varepsilon, \varepsilon]^n : (l', \mathbf{x'}, \boldsymbol{\delta'}) \in SReach^{\varepsilon}_i$
  $\wedge\ (\mathbf{x}, \boldsymbol{\delta}) \in timepass^{\varepsilon}_{sync}(\mathbf{x'}, \boldsymbol{\delta'})\}$
- For i odd, $Succ(SReach^{\varepsilon}_i) = \{(l, \mathbf{x}, \boldsymbol{\delta}) \mid \exists t \in T, l' \in L : l' \xrightarrow{t} l,$
  $\exists \mathbf{x'} \in Inv(l') \cap G(t)\ \ \exists \boldsymbol{\delta'} \in [-\varepsilon, \varepsilon]^n : (l', \mathbf{x'}, \boldsymbol{\delta'}) \in SReach^{\varepsilon}_i :$
  $\wedge\ \mathbf{x} \in Inv(l) \cap Res_t(\mathbf{x'})\ \ \wedge\ \ \boldsymbol{\delta} = Res_t(\boldsymbol{\delta'})\}$
- $\forall i \geq 0, SReach^{\varepsilon}_{i+1} = SReach^{\varepsilon}_i\ \ \cup\ \ Succ(SReach^{\varepsilon}_i)$
- $SReach^{\varepsilon} = \bigcup_{i \in \mathbb{N}} SReach^{\varepsilon}_i$

As before, we assume that all guards and invariants are closed and diagonal-free. Let $Reach$ denote the reach-set obtained by considering perfectly synchronous clocks ($\varepsilon = 0$), where $Reach_i$ denotes the reach-set at step i, as defined previously (cf. Definition 2.6). We establish the relationship between the sets $SReach^{\varepsilon}$ and $Reach$ through the following lemmas.

Lemma 2.13. *For any* $i \in \mathbb{N}$, $l \in L$, $\mathbf{x} \in \mathbb{R}^n_{\geq 0}$, $\boldsymbol{\delta} \in [-\varepsilon, \varepsilon]^n$,

$$(l, \mathbf{x}, \boldsymbol{\delta}) \in SReach^{\varepsilon}_i\ \ \Rightarrow\ \ \exists \mathbf{x_g} \in grid(\mathbf{x}) : (l, \mathbf{x_g}) \in Reach_i$$

Proof. By induction over i.

1. Base case $i = 0$: The proof is immediate as $SReach^{\varepsilon}_0 = \{(l_0, \mathbf{0}, \mathbf{0})\}$ and $Reach_0 = \{(l_0, \mathbf{0})\}$
2. Induction Hypothesis: Assume that Lemma 2.13 holds for some i

[9] By abuse of notation, the subscripts i here denote the sequence of tuples in a run, and not individual vector components.

3. Induction Step: Let $(l, \mathbf{x}, \boldsymbol{\delta}) \in SReach_{i+1}^{\varepsilon}$
   $\Rightarrow (l, \mathbf{x}, \boldsymbol{\delta}) \in SReach_{i}^{\varepsilon} \ \vee \ (l, \mathbf{x}, \boldsymbol{\delta}) \in Succ(SReach_{i}^{\varepsilon})$
   - $(l, \mathbf{x}, \boldsymbol{\delta}) \in SReach_{i}^{\varepsilon}$: The proof is immediate from the Induction Hypothesis, as $SReach_{i}^{\varepsilon} \subseteq SReach_{i+1}^{\varepsilon}$ for all i.
   - $(l, \mathbf{x}, \boldsymbol{\delta}) \in Succ(SReach_{i}^{\varepsilon})$ and i even
     $\Rightarrow \ \exists (l, \mathbf{x'}, \boldsymbol{\delta'}) \in SReach_{i}^{\varepsilon} : (\mathbf{x}, \boldsymbol{\delta}) \in timepass_{sync}^{\varepsilon}(\mathbf{x'}, \boldsymbol{\delta'}) \wedge \mathbf{x} \in Inv(l)$
     $\Rightarrow \exists \mathbf{x'_g} \in Grid : (l, \mathbf{x'_g}) \in Reach_i$
     $\wedge \ \exists \mathbf{x_g} \in grid(\mathbf{x}) \cap Inv(l) : \mathbf{x_g} \in timepass(\mathbf{x'_g})$
     [Induction Hypothesis, definitions of $grid(\mathbf{x})$, $timepass(\mathbf{x'_g})$, and $timepass_{sync}^{\varepsilon}(\mathbf{x'}, \boldsymbol{\delta'})$, and as $Inv(l)$ is closed]
     $\Rightarrow \exists \mathbf{x_g} \in grid(\mathbf{x}) : (l, \mathbf{x_g}) \in Succ(Reach_i)$
     $\Rightarrow \exists \mathbf{x_g} \in grid(\mathbf{x}) : (l, \mathbf{x_g}) \in Reach_{i+1}$
   - $(l, \mathbf{x}, \boldsymbol{\delta}) \in Succ(SReach_{i}^{\varepsilon})$ and i odd:
     $\Rightarrow \exists (l', \mathbf{x'}, \boldsymbol{\delta'}) \in SReach_{i}^{\varepsilon}, \exists t \in T : l' \xrightarrow{t} l, \ \mathbf{x'} \in Inv(l') \cap G(t)$
     $\wedge \ \mathbf{x} \in Inv(l) \cap Res_t(\mathbf{x'}) \ \wedge \boldsymbol{\delta} = Res_t(\boldsymbol{\delta'})$
     $\Rightarrow \exists \mathbf{x'_g} \in Grid : (l', \mathbf{x'_g}) \in Reach_i, \exists t \in T : l' \xrightarrow{t} l$
     $\wedge \ \mathbf{x'_g} \in Inv(l') \cap G(t) \ \wedge \ \exists \mathbf{x_g} \in grid(\mathbf{x}) \cap Inv(l) \cap Res_t(\mathbf{x'_g})$
     [Induction Hypothesis, definition of $grid(\mathbf{x})$, and as $Inv(l)$, $Inv(l')$, and $G(t)$ are closed]
     $\Rightarrow \exists \mathbf{x_g} \in grid(\mathbf{x}) : (l, \mathbf{x_g}) \in Succ(Reach_i)$
     $\Rightarrow \exists \mathbf{x_g} \in grid(\mathbf{x}) : (l, \mathbf{x_g}) \in Reach_{i+1}$ $\square$

The following corollary is an immediate consequence.

Corollary 2.3. *For any $i \in \mathbb{N}$, it holds that:*

$$sup_{s \in SReach_i^{\varepsilon}} \ dist(s, Reach_i) < 1 \ ,$$

where for $s = (l, \mathbf{x}, \boldsymbol{\delta}) \in SReach_i^{\varepsilon}$, $dist(s, Reach_i) = \inf_{(l, \mathbf{x'}) \in Reach_i} dist(\mathbf{x}, \mathbf{x'})$.

Corollary 2.3 intuitively means that irrespective of the iteration depth i, the perturbed reach-set $SReach_i^{\varepsilon}$ stays "close-enough" to the standard reach-set $Reach_i$, in the sense that even the "farthest" point in the perturbed reach-set is less than unit distance away from the standard reach-set.

Lemma 2.14. *For a TA with only closed and diagonal-free guards and invariants, and any closed target location-zone pair of the form (l, B):*

$$Reach \cap (l, B) = \emptyset \Leftrightarrow \forall 0 < \varepsilon < 1 : SReach^{\varepsilon} \cap (l, B) = \emptyset \ ,$$

where $SReach^{\varepsilon} \cap (l, B) = \emptyset$ denotes $\forall (l, \mathbf{x}, \boldsymbol{\delta}) \in SReach^{\varepsilon} : \mathbf{x} \notin B$.

The proof of "$\Leftarrow$" is obvious as $\forall \varepsilon > 0 : SReach^{\varepsilon} \supset Reach$, in the following sense: $\forall (l, \mathbf{x}) \in Reach : (l, \mathbf{x}, \mathbf{0}) \in SReach^{\varepsilon}$. The proof of "$\Rightarrow$" follows from Corollary 2.3, in conjuction with the fact that B is a closed zone, as are all the guards and invariants of the TA, and $0 < \varepsilon < 1$. This lemma, together with the soundness and completeness result for standard zone-based FRA [BY04], leads us to the following theorem, which is the second main result of this chapter:

Theorem 2.2. *For a TA with only closed and diagonal-free guards and invariants, any location l, and any closed, diagonal-free k-zone B :*

$$R \cap (l, B) = \emptyset \Leftrightarrow \forall 0 < \varepsilon < 1 : SReach^{\varepsilon} \cap (l, B) = \emptyset \ ,$$

where R is the symbolic reachable state-space that is ultimately computed by the standard zone-based FRA.

Theorem 2.2 thus establishes the exactness of standard zone-based forward analysis using a tool like UPPAAL for TA with closed guards and invariants, when testing for robust safety against closed targets, with drifting clocks subject to periodic resynchronizations that enforce accuracy better than 1. A certification of robust safety in this case implies that the target state could be avoided by all values of the perturbation ε that are strictly less than 1, independent of the depth of iteration, unlike the case for unbounded relative clock-drift that was considered in the previous section.

2.5 Conclusion

We have investigated reachability in TA subject to drifting clocks – a phenomenon that occurs in practical implementations of timed systems. We first considered the model of clock-drift introduced in [Pur00], and analyzed the reachability for TA with closed guards and invariants, but without the assumption of progress cycles, as was made in [Pur00, WDMR08, DK06, Dim07, JR11, KLMP14, San15]. We showed the exactness of the standard zone-based FRA of UPPAAL for such TA, under a notion of robustness weaker than that in [Pur00, dWDMR04, DK06, Dim07, JR11, KLMP14, San15], in the sense that we do not add states that require an unbounded number of iterations in order to be reached, under infinitesimally small clock-drift (cf. Theorem 2.1). Our notion is applicable to all systems having a finite life-time, where for any particular projected life-time, an appropriate worst-case clock drift enforcing behavior indistinguishable from the ideal can be chosen. For long life-times, the permissible clock drift may become extremely small. As technical realizations in many systems (like, e.g., bit-stuffing in communication protocols or the central-master synchronization incorporated in GPS-controlled systems) address this problem by regular clock resynchronization, thus bounding the relative drift within an set of clocks even over arbitrarily long life-times, we have also modelled and analyzed such synchronization schemes. We have shown that the standard zone-based analysis of UPPAAL is again exact while testing such models for robust safety, but now with the assertion of a uniform strictly positive robustness margin of 1, independent of system life-time.
Note that our definition of TA admits only *diagonal-free constraints* for the guards, invariants, and targets. This is because TA with diagonal constraints of the form $x - y \triangleright c$ have been shown to be incompatible with forward reachability analysis that employs standard k-normalization for termination, and a modified normalization that takes into account the diagonal constraints of the TA is in fact necessary for dealing with such cases [Bou04, BY04]. However, the techniques of this chapter extend quite naturally to TA with diagonal constraints and a suitably modified normalization operation. An extension of these techniques to Probabilistic TA [KNSS02] (TA with discrete probability distributions annotating transitions between locations) is considered in Chapter 6.

We finally wish to mention other works that investigate perturbations in TA while addressing questions other than (symbolic) reachability analysis – [GHJ97] imposes a topological closure on timed traces, which has been shown in [OW03] to affect digitization of TA. [BMR06] considers robust model-checking of LTL properties, while [BMR08] considers robustness analysis via channel machines. The PhD dissertation [San13] is a comprehensive body of work concerning *implementability, synthesis and game-theoretic analysis* of (closed) TA w.r.t guard enlargement, while [ABG$^+$14] considers language theoretic questions arising in a network of TA with independently evolving clocks.

3

(Un-)Decidability of Bounded Multi-Priced Timed Automata

3.1 Introduction

The (un-)decidability frontier between TA, for which location reachability and related properties are decidable, and Linear Hybrid Automata (LHA) [ACH+95], for which these properties happen to be undecidable, has been investigated through analysis of various moderate extensions of the original TA framework. Some of these extensions are interesting in their own right, as they provide valuable enhancements to the expressiveness of the TA framework, thus enabling the analysis and optimization of phenomena such as scheduling, which are beyond the scope of TA.

One such extension is that of Priced Timed Automata (PTA) or, synonymously, Weighted Timed Automata [ABH+01, BFH+01, LBB+01] for modelling real-time systems subject to some *budgetary constraints* on resource consumption. PTA have -in addition to the real-valued *clocks* of classical TA - a *cost-function* mapping locations and edges to non-negative integers, whereupon a certain cost is incurred by staying in a location, or by taking an edge. The *minimum (infimum) cost reachability problem* for PTA computes the minimum (infimum) cost of reaching a given *goal-location*. The minimum / infimum cost reachability problem for PTA has been shown to be decidable and computable [ABH+01, BFH+01, LBB+01], leading to efficient tool-support through UPPAAL CORA along with applications to real-time scheduling [BLR]. A key factor for the decidability of location reachability in PTA is that the cost variable is a monotonically increasing *observer* in the following sense: the cost variable cannot be reset, and testing the cost is forbidden in both guards of edges and invariants of locations, thereby restricting the expressive power of the model wrt. LHA, or equivalently, wrt. Stop-Watch Automata (SWA) [HKPV98, CL00]. This preservation of decidability has attracted an immense amount of research on PTA in recent years (see [BMR06] and Chapter 5 of [Bou09] for surveys), among which we take a closer look at the following enhancements to the original LPTA model:

- The *optimum reachability problem* is considered in [BBBR07] for PTA having a single cost variable, with *both positive and negative integer costs* being allowed on edges and locations. The optimality here refers to the computation of both *infimum and supremum* cost, which is shown to be PSPACE-COMPLETE, with

optimum paths of the underlying transition system consisting of time-transitions occurring at time instants arbitrarily close to integers.

- The optimum (conditional) reachability problem for *Multi-Priced Timed Automata* (MPTA) with multiple cost variables is considered in [LR08], with only *non-negative costs* being allowed on edges and locations. The decidability of the minimum- and maximum- cost reachability problems is shown through exact symbolic (zone-based) algorithms that are guaranteed to terminate. Termination of the symbolic algorithm for computing the maximum cost reachability is subject to a divergence condition on costs, where the accumulation of each of the costs diverges along all infinite paths of the underlying Multi-Priced Transition System (MPTS).
- MPTA with *both positive and negative costs* are considered in [BMR08]. More specifically, [BMR08] investigates *Dual-Priced Timed Automata (DPTA)* with two observers (one observer termed as cost and the other as reward) in the context of optimum infinite scheduling, where the reward takes on only non-negative rates and is "strongly diverging" in the following sense: the accumulated reward diverges along every infinite path in the underlying transition system of the equivalent closed DPTA (obtained as usual by making all inequalities in guards and invariants non-strict). There are no such restrictions on the cost observer, which can take on both positive and negative values. Optimum infinite schedules (that minimize or maximize the cost/reward ratio) are shown to be computable for such DPTA via *corner-point abstractions*.

Nevertheless, an exact characterization of the conditions for (un-)decidability and computation of the *optimum reachability problems* for MPTA having both positive and negative costs have remained unclear. We therefore attempt here to bridge the gap between the results of [BBBR07] and [LR08] by formulating conditions for (un-)decidability and computatibility wrt. the optimum reachability problems for such MPTA, through the following contributions:

1. We first show that *Stopwatch Automata* (SWA) [Č92, HKPV98] can be encoded using MPTA with two cost variables per stopwatch, allowing both positive and negative costs on edges and locations, with each of the costs being subject to individual upper and lower bounds that are to be respected along all viable paths of the underlying transition system. Since location reachability is undecidable in SWA with just one stop-watch, an immediate consequence of such an encoding is that even location reachability becomes undecidable for DPTA with two cost variables, admitting both postive and negative costs in locations and edges. Moreover, this undecidability result holds even when no costs are charged upon taking edges.
2. We then consider MPTA with both positive and negative costs on locations and edges, with individual bounds on each cost variable, and restrict the underlying MPTS such that a minimum absolute cost is incurred along all quasi-cyclic viable paths. Under such a restriction, we show that the reachability problem is decidable and that the optimum cost is computable for such MPTA. These results are derived from a reduction of the optimum reachability problem to the solution of a linear constraint system representing the path conditions over a finite number of viable paths, with the finiteness here being obtained from the boundedness and the (quasi-)cycle conditions on the costs.

Our contributions may thus be viewed as an additional step towards the precise characterization of the (un-)decidability frontiers between various semantic models for richer classes of real-time systems.

The remainder of this chapter is organized as follows: Section 3.2 introduces MPTA and MPTS. Section 3.3 illustrates the encoding of SWA through MPTA admitting both positive and negative (bounded) costs in locations / edges, thereby demonstrating that even location reachability is undecidable for such MPTA with as few as two cost variables. Section 3.4 describes the computation of optimum cost for MPTA with both positive and negative costs in locations and edges, but subject to the cost boundedness and (quasi-)cycle conditions mentioned above. Section 3.5 concludes the chapter. This chapter is a revision of the publication [3], with updated references to the literature that has appeared since its publication.

3.2 Multi- Priced Timed Automata (MPTA)

Given a finite set C of *clocks*, a *clock valuation* over C is a map $v : C \to \mathbb{R}_{\geq 0}$ that assigns a non-negative real value to each clock in C. If n is the number of clocks, a clock valuation is basically a point in $\mathbb{R}^n_{\geq 0}$, which we henceforth denote by $\mathbf{u}, \mathbf{v}$ etc.

Definition 3.1. *A* zone *over a set of clocks C is a system of constraints defined by the grammar $g ::= x \triangleright d \mid x - y \triangleright d \mid g \wedge g$, where $x, y \in C$, $d \in \mathbb{N}$, and $\triangleright \in \{<, \leq, >, \geq\}$. The set of zones over C is denoted $Z(C)$.*

A *closed zone* is one in which $\triangleright \in \{\leq, \geq\}$, and we denote the set of closed zones over C by $Z_c(C)$. A zone with no bounds on clock differences (i.e., with no constraint of the form $x - y \triangleright d$) is said to be *diagonal-free*, and we denote the corresponding set of zones by $Z_d(C)$. The set $Z_{cd}(C)$ denotes zones that are *both closed and diagonal-free*. The set $Z_{cdU}(C)$ denotes the set of *closed, diagonal-free zones having only upper bounds on the clocks*.

Definition 3.2. *An MPTA is a tuple $A = (L, C, (l_0, \mathbf{0}), E, I, \mathbf{P})$, with*

- *a finite set L of* locations *and a finite set C of* clocks*, with $|C| = n$.*
- *An* initial *location $l_0 \in L$ together with the initial clock-valuation $\mathbf{0}$ where all clocks are set to* 0*.*
- *a set $E \subseteq L \times Z_{cd}(C) \times 2^C \times L$ of possible* edges *between locations. An edge $e = (l, g, Y, l')$ between two locations l and l' is denoted $l \xrightarrow{e} l'$, and involves a* guard *$g = G(e) \in Z_{cd}(C)$, a* reset set *$Y = Res_e \subseteq C$.*
- *$I : L \to Z_{cdU}(C)$ assigns* invariants *to locations*
- *$\mathbf{P}$ is an indexed set of* prices *$\{p_1, \ldots, p_{n'}\}$ where each $p_i : (L \cup E) \to \mathbb{Z}$ assigns* price-rates *to locations and* prices *(or* costs*) to edges*

In the sequel, we will denote by m the *clock ceiling* of the MPTA A under investigation, which is the largest constant among the clock constraints of A. For ease of presentation, we assume that the guards and invariants of the automaton are *closed and diagonal-free zones*. We further assume that the clock-values are upper-bounded by m through the invariants at each location. These are not real restrictions, as every (P)TA can be transformed into an equivalent bounded and diagonal-free one (as in

Section 5.3 of [BBBR07]). Boundedness likewise does not confine the expressiveness of Stop-Watch Automata discussed in the next section.

The concrete semantics of such an MPTA is given by a corresponding *Multi-Priced Transition System (MPTS)* with states $(l, \mathbf{u}) \in (L, \mathbb{R}^n_{\geq 0})$ where $\mathbf{u} \models I(l)$, with initial state $(l_0, \mathbf{0})$, and a transition relation $\rightarrow$ defined as follows:

- *Time-transitions*: $(l, \mathbf{u}) \xrightarrow{\delta, \mathbf{c}} (l, \mathbf{v})$ if $\mathbf{c} = \mathbf{P}(l) \cdot \delta$ and $\forall 0 \leq t \leq \delta : \mathbf{u} + t \models I(l)$ where $\mathbf{u} + t$ denotes the addition of t to each component of $\mathbf{u}$.
- *Switch-Transitions*: $(l, \mathbf{u}) \xrightarrow{e, \mathbf{c}} (l', \mathbf{v})$ if $\exists e = (l, g, Y, l') \in E : \mathbf{u} \models g, \mathbf{v} = [Y \leftarrow 0]\mathbf{u}$, $\mathbf{v} \models I(l'), \mathbf{c} = \mathbf{P}(e)$

Definition 3.3. *A* canonical initialized path *[BBBR07] π of an MPTS is a (possibly infinite) sequence of states s_i (each state s_i being a location-plus-clock-valuation pair of the form $(l, \mathbf{u})$), which starts from the initial state and alternates between time- and switch-transitions $\pi = s_0 \xrightarrow{\delta, \mathbf{c^0}} s_1 \xrightarrow{e_1, \mathbf{c^1}} s_2 \ldots$. The set of all possible canonical initialized paths is denoted Π. For a finite path $\pi \in \Pi$ of length $|\pi| = k$, its* accumulated cost-vector *is defined as: $\mathbf{Cost(\pi)} = \sum_{i=0}^{k-1} \mathbf{c^i}$, with the summation here being performed component-wise for each cost-vector $\mathbf{c^i}$, with $Cost_j(\pi) = \sum_{i=0}^{k-1} c^i_j$ for each cost-component.*

For $\pi \in \Pi$, let π_k denote its finite prefix of length k. Then the corresponding accumulated cost along π is given by $\mathbf{Cost(\pi)} = lim_{k \to \infty} \mathbf{Cost(\pi_k)}$, if the latter exists.

The accumulated cost of $\pi \in \Pi$ wrt. a (set of) goal *state(s) G is defined as:*

$$\mathbf{Cost_G(\pi)} = \begin{cases} \boldsymbol{\infty} & \text{if } \forall i \geq 0 : s_i \notin G \ , \\ \sum_{i=0}^{k} \mathbf{c^i} & \text{if } \exists k \geq 0 : (s_k \in G \wedge \forall i < k : s_i \notin G) \ . \end{cases}$$

$\mathbf{Cost_G(\pi)}$ for $\pi \in \Pi$ therefore yields the accumulated cost-vector along the shortest prefix of π ending in a goal state.

Cost-Boundedness Constraint.

We assume in this chapter that the permissible *cost charging is bounded by budgetary constraints*, in the sense that paths of the MPTS exceeding this budget (e.g., exhausting the battery capacity) are considered unviable and thus irrelevant to the optimization problem, even if the budget is exceeded only temporarily. The *budgetary constraint* is given formally as follows: For each cost variable, there is a lower bound $L_j \in \mathbb{Z}_{\leq 0}$ and an upper bound $U_j \in \mathbb{Z}_{\geq 0}$ which all viable paths have to obey throughout. Thus, a path π is called *viable* iff

$$\forall \pi' \text{ non-empty canonical prefix of } \pi : \forall j \in \{1, \ldots, n'\} : L_j \leq Cost_j(\pi') \leq U_j$$

holds.

We further designate Ω as the linear objective function that we wish to *optimize* wrt. reaching a set of *goal locations* under such budgetary constraints. Ω can be an arbitrary linear combination of prices drawn from $\mathbf{P}$. The objective of this chapter is to formulate the conditions for (un-)decidability of reaching G under the budgetary constraints and for computatibility of the minimum value of Ω when viably reaching G. We call the latter the optimum-cost reachability problem, formally given below.

Problem 3.1. Given an MPTA $A = (L, C, (l_0, \mathbf{0}), E, I, \mathbf{P})$ having a set Π of canonical initialized paths in its corresponding MPTS, and given a set $G \subseteq L$ of goal locations plus a linear objective function Ω, as well as budgetary constraints (L_j, U_j) for the accumulation of each cost function p_j along all viable paths in Π, the *optimum-cost reachability problem* is to compute

$$\min\{\Omega(\mathbf{Cost}_{\mathbf{G}}(\pi)) \mid \pi \in \Pi, \pi \text{ viable}\} \ .$$

Note that as Ω is an arbitrary linear combination of the prices accumulated in A, this problem — despite being formulated as a minimization problem — incorporates maximum-cost reachability also.

MPTA having two cost variables only are termed *Dual-Priced Timed Automata (DPTA)* in the remainder. We now proceed to show in Section 3.3 that DPTA with a boundedness condition on costs as above can be used to encode Stop-Watch Automata (SWA) with one stopwatch, for which even location reachability is undecidable [HKPV98]. It therefore follows that Problem 3.1 is undecidable for such cases. We however show in Section 3.4 that Problem 3.1 is decidable and computable even for MPTA when one imposes suitable conditions on viable quasi-cyclic paths of the corresponding MPTS.

3.3 Encoding of Stop-Watch Automata using Bounded MPTA

Stopwatch automata (SWA) are an extension of timed automata where advance of individual clocks can be stopped in selected locations. It has been shown in [Č92, HKPV98] that location reachability is undecidable even for simple SWA (in the sense of all clock constraints being diagonal-free), and even when both the clocks and the stopwatches are confined to bounded range. The result, which is based on encoding two-counter machines, applies to SWA as small as a single stopwatch and four clocks. In the sequel, we will provide an encoding of stopwatch automata with n bounded clocks and n'' bounded stopwatches by MPTA with $n + 1$ clocks and $2n''$ cost variables. This shows that location reachability is undecidable for bounded dually priced 5-clock MPTA.

As it suffices for our undecidability result, and as the generalization is straightforward, we demonstrate our reduction on 1-stopwatch SWA only. Let $m \in \mathbb{N}$ be a common upper bound on all clocks $C = \{x_1, \ldots, x_n\}$ and the single stopwatch sw occurring in the SWA A, i.e. m dominates the individual range bounds on clocks and sw. We construct an MPTA with two cost variables s and S, both with bounded range $[0, 2m]$, and $n + 1$ (bounded) clocks $C \cup \{h\}$, where h is a fresh helper clock. W.l.o.g, we assume that the SWA A to be encoded does not contain guard conditions on its stopwatch, as these can always be replaced by invariants imposed in urgent transient states.

The central idea of the encoding is that s watches the lower bounds while S watches the upper bounds imposed on sw. Therefore,

1. the prices s and S do generally evolve with the same rate as the stopwatch sw they simulate,

2. $s \leq S$ holds throughout,
3. when sw is subject to an invariant imposing a lower bound of $l \geq 0$ then $s = sw - l$,[1]
4. when sw is subject to an invariant imposing an upper bound of $u \leq m$ then $S = sw + 2m - u$.

Note that maintaining properties 3 and 4 leads to the bound $[0, 2m]$ on s and S enforcing the original invariant on sw, as $s = sw - l \wedge s \geq 0$ implies $sw \geq l$ and $S = sw + 2m - u \wedge S \leq 2m$ implies $sw \leq u$. Thus, the general bounds on s and S enforce the invariants on sw without any need for explicit invariants on s and S. All that has to be done is to, first, initialize s and S such that 3 and 4 hold, which is achieved by replacing the initial state by two states as in Fig. 3.1 and, second, update

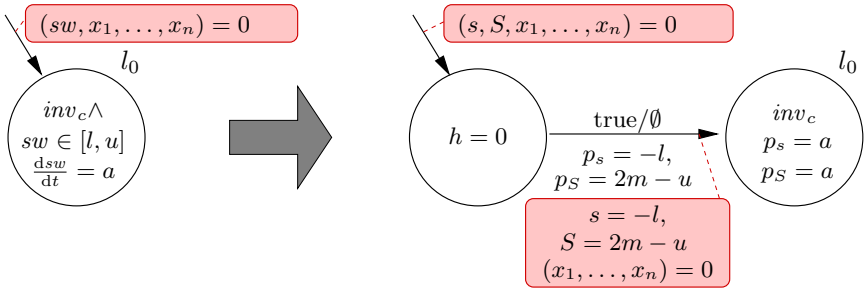

Fig. 3.1. Initializing the cost variables s and S simulating the stopwatch such that they enforce the invariant on the stopwatch sw. Here and in the remainder, inv_c (as well as inv'_c) refers to the parts of the invariant not dealing with the stopwatch. a is the slope of the stopwatch in l_0, which can be 0 or 1. Here and in all subsequent figures, formulae in shaded boxes are not part of the automaton, but collect invariant properties of the MPTS guaranteed along simulating runs.

them accordingly upon a change of the invariant mediated by a location change in the SWA, which is shown in Fig. 3.2. Note that in both cases in accordance with property 1, the cost rates p_s and p_S coincide to the slope a of sw.

Resetting the stopwatch requires a slightly more complex construction, as we need to force s to value 0 (assuming that the invariant of the location following the reset does not enforce a lower bound on the just reset stopwatch, which would render the switch transition infeasible) and S to value $2m - u'$, where u' is the upper bound on sw in the target location of the resetting switch transition. To achieve this, we simulate the (instantaneous) switch transition by a run of duration $2m$. Within this run, we let s and S run to value m, which we test by substracting $-2m$ in a subsequent switch transition. We then adjust the values as desired. Furthermore, we employ the wrapping automaton construction of [Č92, HKPV98] to preserve the clock values. The complete automaton fragment is depicted in Fig. 3.3.

[1] Note that whenever there is no explicit invariant enforcing a stronger lower bound, sw still is subject to the invariant $sw \geq 0$. Moreover, sw is always subject to an upper bound as it is generally confined to the range $[0, m]$. The invariants $s \geq 0$ (uniform over all locations) and $sw \geq l$ mutually reinforce each other.

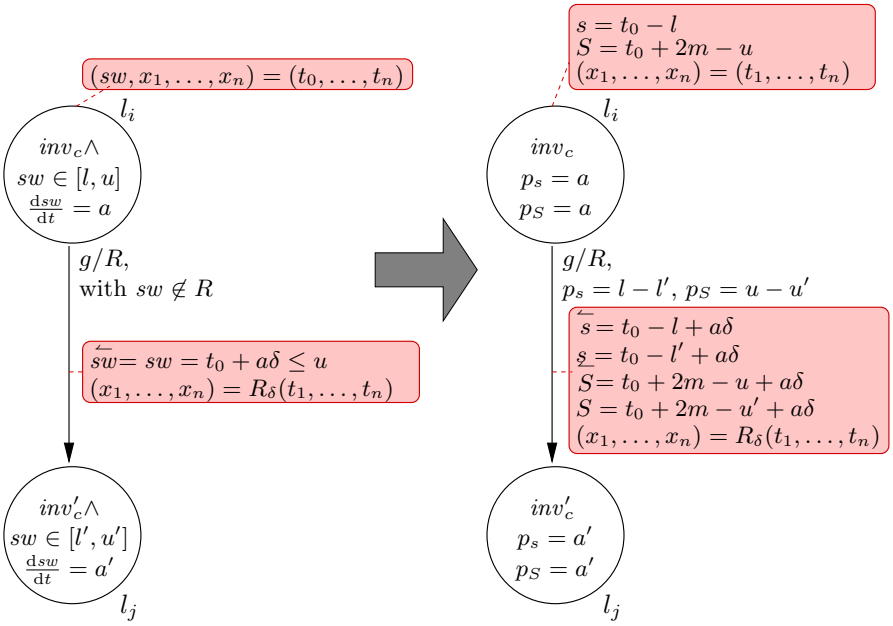

Fig. 3.2. Implementing change of invariant on the stopwatch in case the stopwatch is not reset by the switch transition. W.l.o.g., we assume that the guard g does not mention the stopwatch, as the pertinent conditions can be moved to invariants. $\overleftarrow{x}$ denotes the value of x before the transition while x denotes its value thereafter. δ represents the time spent in l_i and $R_\delta(t_1, \ldots, t_n)$ abbreviates the result of applying the reset R to $(x_1, \ldots, , x_n) = (t_1 + \delta, \ldots, t_n + \delta)$.

Glueing together the above MPTA fragments at the like-named locations, one obtains an MPTA which is equivalent to the encoded SWA wrt. location reachability. Due to the undecidability of location reachability for SWA [Č92, HKPV98], this reduction yields the following result:

Theorem 3.1. *Location reachability is undecidable for MPTA with $n \geq 1$ clocks and $\max(2, 14 - 2n)$ bounded cost variables. In particular, it is undecidable for MPTA with 6 clocks and 2 bounded cost variables, as well as for 1 clock and 12 bounded cost variables.*

Proof. The invariance properties mentioned in the shaded boxes in Figures 3.1 to 3.3, which are straightforward to establish based on the semantics of stopwatch automata and MPTA, show that the stopwatch automaton A has a path reaching location l_i with clock readings $(x_1, \ldots, x_n) = (t_1, \ldots, t_n)$ and stopwatch reading t_0 iff the encoding MPTA M has a viable path reaching location l_i with the same clock readings $(x_1, \ldots, x_n) = (t_1, \ldots, t_n)$ and costs $s = t_0 - l$ and $S = t_0 + 2m - u$. Hence, A can reach a given target location l_{target} iff M can reach the corresponding location.

According to [Č92, HKPV98], location reachability is undecidable for simple SWA with bounded clocks and stopwatches. The reduction to two-counter machines

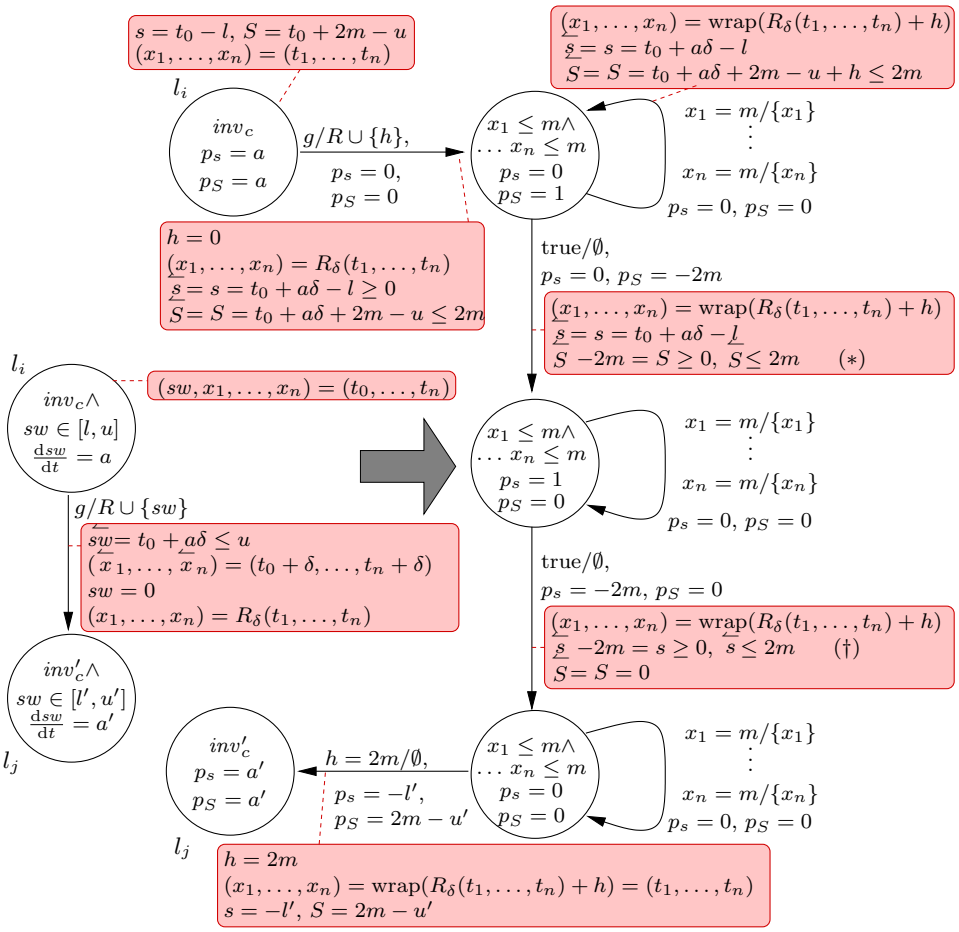

Fig. 3.3. Resetting the stopwatch, i.e. setting $s = -l$ and $S = 2m - u'$ while preserving the clocks. Here, $\mathrm{wrap}(x) := x - m\lfloor \frac{x}{m} \rfloor$. Note that $(*)$ and $(\dagger)$ imply $S = 0$ and $s = 0$.

used in their proof yields SWA with five clocks and one stopwatch. As clocks are special cases of stopwatches, location reachability is thus undecidable for SWA with $n \ge 0$ clocks and $max(1, 6 - n)$ stopwatches. Our reduction encodes such SWA by a bounded MPTA with $n \ge 1$ clocks and $max(2, 14 - 2n)$ bounded prices. □

An immediate consequence is

Corollary 3.1. *Optimum cost is not effectively computable for bounded MPTA.*

Proof. As the optimum cost is infinite iff the target location is unreachable, computing the optimum cost entails solving the reachability problem, which is undecidable for bounded PTA with more than one price according to Theorem 3.1. □

Note that these results can easily be strengthened wrt. the operations permitted on costs:

Corollary 3.2. *Location reachability remains undecidable and optimum cost uncomputable even if no costs can be charged on edges.*

Proof. It is straightforward to simulate an instantaneous price update by a durational update of fixed duration and slope using the wrapping construction to retain the clock values. □

When interpreting the above negative results concerning effectiveness, it should be noted, however, that the encoding relies crucially on closedness as well as on the universally binding character of the budgetary constraints. It is currently unclear whether similar undecidability results hold under an open (in the sense of strict bounds on the cost) budget or the permission to temporarily overdraw the budget by small amounts.

3.4 Optimum Cost Reachability for MPTA under Cost-Charging Quasi-Cycles

In this section, we investigate the optimum cost problem for bounded MPTA subject to the additional assumption that non-trivial cost is charged upon each quasi-cyclic viable path, where a quasi-cycle is a sequence of states returning close to its origin. This property is captured by the following definition:

Definition 3.4. *We call an MPTA* cost-charging on quasi-cycles *if there exists* $\varepsilon > 0$ *such that for each canonical viable path* $\pi = s_0 \xrightarrow{\delta_0, \mathbf{c^0}} s_1 \xrightarrow{e_1, \mathbf{c^1}} s_2 \ldots \xrightarrow{\delta_i, \mathbf{c}^i} s_i$ *with* $d(s_0, s_i) \leq \varepsilon$ *and path-length* $i \geq 2$*, i.e. the path contains at least one jump, it holds that* $|Cost_k(\pi)| \geq \varepsilon$ *for some cost-component. I.e., the MPTA is cost-charging on quasi-cycles iff there is no infinitesimally cheap return to a close vicinity of a state once this vicinity has been left.*

Hereby, we define the distance $d((l, \mathbf{u}), (l', \mathbf{u}'))$ *between two states* $(l, \mathbf{u})$ *and* $(\mathbf{l}', u)$ *to be*

$$d((l, \mathbf{u}), (l', \mathbf{u}')) = \begin{cases} \infty & \text{if } l \neq l', \\ ||\mathbf{u} - \mathbf{u}'|| & \text{if } l = l', \end{cases}$$

where $||\cdot||$ *is the maximum norm.*

Note that we neither demand a constant sign for the cost incurred nor fix the cost variable that incurs non-trivial cost upon quasi-return. Hence, some cost variable may well incur cost ε on some path from $(l, \mathbf{u})$ to some $(l, \mathbf{u}')$ in its ε-vicinity and cost $-\varepsilon$ on the same or another cost variable when proceeding from $(l, \mathbf{u}')$ to another $(l, \mathbf{u}'')$ in the ε-vicinity of $(l, \mathbf{u}')$.

Nevertheless, together with compactness of the state space as implied by boundedness, cost-charging on quasi-cycles is strong enough a condition of finiteness on all viable, i.e. bound-respecting, paths. [2]

[2] Note that any constant bound $\delta(\varepsilon) > 0$ suffices as a minimum lower bound on the absolute cost incurred along quasi-cyclic paths in order to obtain the finiteness condition. We have however chosen a single parameter ε for ease of presentation.

Lemma 3.1. *Let A be a bounded MPTA which is cost-charging on quasi-cycles. Then the length of canonical viable paths in A is finitely bounded.*

Proof. Let $\varepsilon > 0$ be the constant from Def. 3.4. Let L be the set of locations and $\{x_1, \ldots, x_n\}$ be the set of clocks in A and let D_i for $i = 1, \ldots, n$ be their respective bounded domains. Let $\{p_1, \ldots, p_{n'}\}$ be the set of cost variables in A and let P_i for $i = 1, \ldots, n'$ be their respective bounded domains. As the domains are bounded, the topological closure $\overline{V}$ of the combined clock-and-cost space $V = \prod_{i=1}^{n} D_i \times \prod_{j=1}^{n'} P_j$ is compact. Hence, V can only contain finitely many ε-separated points.

Let k be an upper bound on the maximum number of ε-separated points in V. Let $\pi = s_0 \xrightarrow{\delta_0, \mathbf{c^0}} s_1 \xrightarrow{e_1, \mathbf{c^1}} s_2 \ldots \xrightarrow{\delta_K, \mathbf{c}^K} s_K$ be a viable canonical path and let $\mathbf{C}^i = \sum_{j=0}^{i} \mathbf{c}^i$ be the accumulated costs until step i. As A is cost-charging on quasi-cycles, $||\mathbf{u}_i - \mathbf{u}_j|| \geq \varepsilon \vee ||\mathbf{C}_i - \mathbf{C}_j|| \geq \varepsilon$, which is equivalent to $||(\mathbf{u}_i, \mathbf{C}_i) - (\mathbf{u}_j, \mathbf{C}_j)|| \geq \varepsilon$, for each $i, j \leq K$ with $l_i = l_j$. As $(\mathbf{u}_i, \mathbf{C}_i) \in V$ for each $i \leq K$, it follows that $\forall l \in L : |\{(l_i, u_i, C_i) \mid i \leq K \wedge l_i = l\}| \leq k$. Consequently, $K \leq k \cdot |L|$ holds for the length K of the canonical viable path π. □

Given the fact expressed in Lemma 3.1 that all canonical viable paths have a uniform finite bound on their lengths, a consequence is that the optimum reachability problem for MPTA becomes an instance of *bounded model-checking* (BMC) [BCC+03] that is solvable for a rich class of hybrid systems. In particular, we encode the optimum bounded reachability problem up to depth k for MPTA as a Mixed Integer Linear Program (MILP). In the remainder of this section, we provide a corresponding algorithm, which is based upon reducing Problem 3.1 for cost-charging MPTA to a Mixed Integer Linear Program, along the lines of [PSE03] illustrating BMC for acyclic LPTA.

This MILP will then be used twofold: First, as it expresses feasibility of a path of length k, versions with increasingly larger k will be used to determine the upper bound K on path length. Once this has been found, a version of depth K equipped with the cost term as an objective function will be used for determining the optimum cost.

Given an MPTA $A = (L, C, (l_0, \mathbf{0}), E, I, \mathbf{P})$, a set $G \subseteq L$ of goal-locations, and an arbitrary linear combination Ω of prices, we generate the following MILP for the BMC problem of depth k:

- For each discrete location $l \in L$ we take $k + 1$ zero-one variables l^i, where each l^i takes on either of the values 0 or 1, with $0 \leq i \leq k$. The value of l^i encodes whether A is in location l in step i as follows: $l^i = 1$ iff A is in location l in step i. Thus, for any $i \leq k$, there should be exactly one $l \in L$ such that $l^i = 1$, which can be enforced by requiring $\sum_{l \in L} l^i = 1$ in the MILP for each $i \in \{0, \ldots, k\}$.
- For each edge $e \in E$ we take k zero-one variables e^i, with $1 \leq i \leq k$. The value of e^i encodes whether A's ith move in the run was transition e. Again, one enforces that exactly one transition is taken in each step by adding the constraint $\sum_{e \in E} l^i = 1$ in the MILP for each $i \in \{1, \ldots, k\}$.
- For each clock $c \in C$ we take k real-valued variables c^i, with $0 \leq i \leq k - 1$. The value of c^i encodes c's value immediately *after* the ith transition in the run.
- For each $i \leq k$ we take one real-valued variable δ^i representing the time spent in the ith location along the run.

- We add constraints describing the initial state, i.e. enforcing $l_0^0 = 1$ and $c^0 = 0$ for each $c \in C$.
- We add constraints describing the relationship between discrete locations and transitions, i.e. guaranteeing that $e^i = 1$ implies $l^{i-1} = 1$ and $\widetilde{l^i} = 1$ for $(l, \widetilde{l}) \in e$. This can be encoded as a linear constraint via

$$l^{i-1} \geq e^i \ \wedge \ l^i \geq e^i \ .$$

- We add constraints enforcing the location invariants, i.e. checking for each $i \leq k$ that $l^i = 1 \implies I(l^i)[c_1^i, \ldots, c_n^i/c_1, \ldots, c_n]$ and that $l^i = 1 \implies I(l^i)[c_1^i + \delta^i, \ldots, c_n^i + \delta^i/c_1, \ldots, c_n]$, where $c_1, \ldots, c_n$ are the clocks (i.e., $\{c_1, \ldots, c_n\} = C$) and $\phi[y/x]$ denotes substitution of y for x in ϕ.
  As all clocks are bounded by m, the implications can be realized using the *switch variable encoding*. E.g., for an upper bound $x \leq u$ in the invariant, $m \cdot l_i + x^i + \delta^i \leq m + u$ implements the implication $l_i = 1 \implies x^i + \delta^i \leq u$.
- Using the same encoding, we add constraints enforcing guards, i.e. guaranteeing for each $0 \leq i \leq k-1$ that

$$e^{i+1} = \mathit{active} \implies g(e)[c_1^i + \delta^i, \ldots, c_n^i + \delta^i/c_1, \ldots, c_n] \ ,$$

  where $g(e)$ denotes the guard of edge e.
- We add constraints dealing with resets, i.e. enforcing for each $0 \leq i \leq k-1$ that

$$e^{i+1} = \mathit{active} \implies \begin{cases} c^{i+1} = c^i + \delta^i & \text{iff } c \notin Y(e) \ , \\ c^{i+1} = 0 & \text{iff } c \in Y(e) \ , \end{cases}$$

  where $Y(e)$ is the reset map associated to edge e.
- For each price variable $p \in \mathbf{P}$, we define $k+1$ auxiliary variables p_t^i recording the step price incurred by p in step $i \leq k$ and $k+1$ auxiliary variables p_d^i recording the price incurred by staying in the location during step $i \leq k$. Using the switch variable encoding, we enforce

$$\bigwedge_{l \in L} \bigwedge_{i=0}^{k} l_i = 1 \implies p_d^i = p(l) \cdot \delta^i$$

  and

$$p_t^0 = 0 \wedge \bigwedge_{e \in E} \bigwedge_{i=1}^{k} e_i = 1 \implies p_t^i = p(e) \ ,$$

  where $p : E \cup L \to \mathbb{Z}$ is the cost assignment of A.
- Adding k further variables p^i for each price $p \in P$, we can record the price $Cost_p$ accumulated so far by defining

$$\begin{aligned} p^0 &= 0 \ , \\ p^{j+1} &= p^j + p_d^j + p_t^j \end{aligned}$$

  for each $j < k$. Viability of the paths is enforced by additionally demanding

$$\begin{aligned} L_p \leq p^j &\leq U_p \ , \\ L_p \leq p^j + p_d^j &\leq U_p \end{aligned}$$

  for each $j \leq k$.

This encoding, which is a standard MILP encoding suitable for bounded model-checking, can now be used in two ways:

1. For checking whether viable paths of length k exist, the above system is simply build for the desired depth k and checked for feasibility using an MILP solver. The resulting MILP is feasible iff paths of length $\geq k$ exist.
2. For determining the minimum cost for reaching the goal states within k steps, we, first, modify the goal states to become sinks by decorating them with cost-free and always enabled loops, second, build the above constraint system to depth k, third, add constraints enforcing the goal-locations to be visited by constraint
$$\sum_{i=0}^{k} \sum_{l \in G} l^i \geq 1$$
and, finally, use the linear expression
$$\Omega[\mathbf{P}^k/\mathbf{P}]$$
as an objective function to be minimized by the MILP solver. The MILP solver will either report the system to be infeasible, in which case the minimum cost of reaching G along canonical initialized paths of length at most k is infinite, or it will report the minimum cost of reaching G along canonical initialized paths of length at most k as the optimum value of its objective function. An optimum path can then be retrieved from the variables in the MILP that represent the MPTA state at the various steps.

Combining these two steps, we can solve Problem 3.1, i.e. the optimum-cost reachability problem, effectively by iteratively performing step 1 for increasing k until no viable path of length k^* exists and then performing step 2 for $k^* - 1$. Based on this procedure, we obtain the following positive result concerning effectiveness of cost optimal reachability in MPTA:

Theorem 3.2. *For bounded MPTA A which are cost-charging on quasi-cycles, the following two properties hold:*

1. *It is decidable whether A has a viable (i.e. obedient to the budgetary constraints) initialized path to some goal state.*
2. *The optimum cost for A reaching a goal state via a viable initialized path is computable for any linear cost function.* □

3.5 Conclusion and future research

We have investigated conditions for (un-)decidability and computability of the optimum reachability problem for MPTA admitting both positive and negative costs on locations and edges. Our encoding of SWA using cost-bounded MPTA however critically depends on the fact that the bounds on the cost variables are *closed* (i.e., non-strict), and that these bounds are to be respected for each cost variable along all viable paths of the underlying transition system. The question of whether this undecidability result holds even when cost bounds are strict, or when some path is allowed to temporarily overshoot the budget wrt. a cost variable, or when subject to other

forms of budgetary constraints (as motivated, for instance, in [BFL$^+$08, BLM14]) remains currently open. Moreover, given that the undecidability result for cost-bounded MPTA holds for $n \geq 1$ clocks and $max(2, 14 - 2n)$ bounded cost variables (cf. Theorem 3.1), another natural question would be the validity of this result when one correspondingly restricts the number of clocks and bounded cost variables. Another interesting research direction would be to investigate a possible extension of the language theoretic results of [Qua10] to the setting of MPTA.

The cost-charging (quasi-)cycle assumption used to validate the BMC procedure of Section 3.4 is related to the divergence assumptions made in [LR08, BMR08].

The presently proposed procedure of iteratively increasing the depth (for which BMC is performed) is not efficient, particularly for a small bound ε on the minimum cost incurred along quasi-cyclic viable paths. The question of whether other realistic path conditions might admit efficient (symbolic) algorithms, and with applications to decision problems for the Duration Calculus (DC) [CHR91] remains open. The work in [FH07] implements a decision procedure for a rich fragment of DC, allowing for *positive linear combinations* of accumulated durations with only *upper bound duration constraints*. As negatively weighted durations map to negative cost rates when extending the construction of [FH07], efficient symbolic algorithms for MPTA with both positive and negative costs would enable reasoning about richer DC-fragments involving *both positive and negative* linear combinations of accumulated durations, with *both upper- and lower-bound duration constraints*.

4

Structural Transformations for Extended Timed Automata

4.1 Introduction

Reasoning about networks of (real-time) systems is much easier when the execution of the system components is viewed sequentially, as opposed to corresponding distributed or concurrent representations. Transformations of distributed system representations to equivalent *layered* (i.e., sequential) representations were first explored in [EF82] through a notion of *communication closedness* between system components. Such a layered transformation was subsequently investigated in [Jan94] for a process algebra based on hierarchical graphs, with an operator for *layered composition* (intermediate between sequential and parallel composition) that formalized *equivalences* between the distributed and layered system representations through the *Communication Closed Layer* (CCL) laws, by exploiting *independence* between system components. Real-time extensions to this process algebra were presented in [JPXZ94], where CCL laws were shown to hold under certain *timing conditions*, even in the absence of cross-component independence.

We present in this chapter an enhancement of the layered transformation used in the *assertion-based* reasoning techniques of the above works for simplifying the *automatic verification* of (networks of) real-time systems. Our layered transformation is complemented by the transformation techniques of *separation* and *flattening*. This chapter is a revised version of [7] with the following structure:

Many works on partial order reduction for (networks of) timed automata (TA) (cf. [BJLY98, Min99, LNZ05, HP07]) assume *local time semantics*, where the clocks of each constituent TA evolve independently so as to reduce timing-based dependencies, but at the expense of extra *reference clocks* for resynchronization (cf. [BJLY98]). In this chapter, we instead work with networks of TA extended with shared data variables (termed extended timed automata (ETA)), having *synchronous clocks* across the constituent ETA, as supported by the well-established ETA model-checker UPPAAL [BDL04]. Dependencies in such ETA networks arise due to: (a) the read-write interference of the variables shared across the ETA, and (b) the global timing constraints induced by synchronous clocks.

Section 4.2 of this chapter reviews ETA and their compositional constructs, while Section 4.3 establishes communication closed equivalences that exploit the absence of dependencies due to (a) and (b) above. Notions of non-interference are introduced

for dealing with (a), while (b) is handled by *wrapped* ETA that mimic local time semantics in a network, even in the presence of globally synchronous clocks.

Section 4.4 of this chapter establishes communication closed equivalences for ETA with synchronous clocks that exploit *precedence relations*, in the presence of shared variable and clock dependencies between ETA.

The explicit passage of control (from one sequential phase of the system to the next) necessary for applying (non-interference- or precedence-based) layered transformations may however not be directly apparent from the system's structure, owing to multiple nested cycles that often arise while modelling reactive distributed real-time systems as ETA networks. We therefore introduce in Section 4.5 the transformations of *separation* and *flattening* as (reachability preserving) pre-processing steps that (under certain cycle conditions on the ETA) reduce the nesting depth and the number of cycles in ETA networks. Communication closedness (via appropriate non-interference and/or precedence conditions) may be easily investigated on such separated and flattened ETA, so that the verification of *layered reachability* properties may be rendered almost trivial.

The interplay of the three structural transformations (separation, flattening, and layering) is illustrated in Section 4.6 on an enhanced version of Fischer's real-time mutual exclusion protocol for two critical sections. Section 4.7 concludes the chapter.

4.2 ETA and Composition Operators

We now briefly review the *extended timed automata* (ETA) model for (networks of) real-time systems enriched with *shared data variables*. Semantics of ETA are given in terms of the underlying timed transition system and the associated regions. Various compositional operators for ETA are also introduced – namely those for *sequential*, *step*, *parallel*, and *layered* composition, along with the notions of *non-interference* (for dealing with shared variables) and *wrapping* (for dealing with globally synchronous clocks).

Extended timed automata.

We first introduce some notions that constitute the syntax of ETA, where the notation $(x, y, \cdots \in)\ X$ indicates that we denote by $x, y \cdots$ the typical elements of the set X. Let $(\alpha, \beta, \cdots \in)\ \Sigma$ be a finite *alphabet* of *channels*. For each channel $\alpha \in \Sigma$ there are two *actions*: $\alpha?$ denotes *input* on α and $\alpha!$ *output* on α, where $\alpha?, \alpha! \notin \Sigma$. We consider two different *internal* actions $\tau, \varepsilon \notin \Sigma$, where τ results only from *synchronization*. The set of all actions over Σ is denoted by $(a, b, \cdots \in)$ $\Sigma_{?!} = \{\alpha? \mid \alpha \in \Sigma\} \cup \{\alpha! \mid \alpha \in \Sigma\} \cup \{\tau, \varepsilon\}$. In the context of parallel composition, input and output are *complementary* actions that can synchronize yielding τ. For an action $a \in \Sigma_{?!} \setminus \{\tau, \varepsilon\}$, its complementary action is denoted by $\overline{a}$, i.e., $\overline{\alpha?} = \alpha!$, and vice-versa.

Let $(u, v, \cdots \in)\ V$ be a finite set of *data variables* ranging over a finite set D. The set $(\psi_D \in)\ \Psi(V)$ of *data expressions* over V is the set of expressions involving variables of V and the arithmetic operators $+, -, \cdots$, interpreted in the usual way for integer-valued data variables. The set $(\phi_D \in)\ \Phi(V)$ of *data constraints* over V is the set of Boolean constraints over variables in V involving the arithmetic $(+, -, \cdots)$

and relational ($<, \leq, >, \geq$) operators, interpreted in the usual way for integer-valued data variables.A *data valuation* assigns to each data variable in V a value in D. If $|V| = m$, we denote by $(\mathbf{u}, \mathbf{v}, \cdots \in)\ D^m$ the set of data valuations.

Let $(x, y, \cdots \in)\ C$ be a finite set of *clocks*. The set $(\phi \in)\ \Phi(C)$ of *clock constraints* over C has the following syntax: $\phi ::= x \bowtie k \mid \phi_1 \wedge \phi_2$, where $x \in C$, $k \in \mathbb{N}$, and $\bowtie \in \{<, \leq, \geq, >\}$. The subset of clock constraints involving only *upper bounds* $<, \leq$ on clocks is denoted by $\Phi_U(C)$ The set $(g \in)\ G(C, V)$ of *guards* has the syntax: $g ::= \phi \mid \phi_D \mid g_1 \wedge g_2$, where $\phi \in \Phi(C)$ and $\phi_D \in \Phi(V)$.

A *clock valuation* assigns a non-negative real value to each clock in C. If $|C| = n$, we identify the set of clock valuations with $(\mathbf{x}, \mathbf{y} \in)\ \mathbb{R}^n_{\geq 0}$. For a clock valuation $\mathbf{x}$ and a positive real number d, we write $\mathbf{x} + d$ to denote the addition of d uniformly to each component of $\mathbf{x}$. By $\mathbf{0}$ we denote the clock valuation where all clocks are set to 0, while $\mathbf{v}_0 \in D^m$ denotes a designated initial data valuation. A *reset* operation is an assignment $x := 0$ to a clock $x \in C$, or an assignment $v := \psi_D$ to a data-variable $v \in V$, where $\psi_D \in \Psi(V)$. By $(r \in)\ R(C, V)$ we denote the set of lists of reset operations. In the sequel, we shall identify constraints with the respective sets of clock and data valuations satisfying them, so as to enable the application of set-theoretic operations.

We then define an *extended timed automaton* (ETA) A as follows:

Definition 4.1 (Extended timed automaton). *An* extended timed automaton *(ETA) is a tuple*
$A = (L, \Sigma, C, V, l_0, l_F, Inv, E)$*, where*

- L *is a finite set of* locations*,*
- Σ *is a finite* alphabet *of* channels*,*
- C *is a finite set of* clocks *with* $|C| = n$*,*
- V *is a finite set of* data variables *with* $|V| = m$*,*
- $l_0 \in L$ *is the* initial *location,*
- $l_F \in L$ *is the* final *location,*
- $Inv : L \to \Phi_U(C)$ *assigns a* clock invariant *to each location, where we assume that* $Inv(l_F) = true$*,*
- $E \subseteq L \times \Sigma_{?!} \times G(C, V) \times R(C, V) \times L$ *is a finite set of directed* edges *between locations, where an edge* $e = (l, a, g, r, l')$ *from* l *to* l' *involves an* action $a \in \Sigma_{?!}$ *over the alphabet* Σ*, a* guard $g \in G(C, V)$*, and a list* r *of* reset operations *involving clocks in* C *and data variables in* V*.*

Note that clock invariants, by definition, admit only upper bounds on the clock values. An edge $e \in E$ is of the form $e = (l, a, g, r, l')$ with $l, l' \in L, a \in \Sigma_{?!}, g \in G(C, V)$, and r a list of reset operations. Define $target(e) = l'$ as the target location of the edge e, $act(e) = a$ as the *action of* e and $edges_A(a) = \{e \in E \mid act(e) = a\}$ as the set of edges in A with action a. For a pair of clock valuations $\mathbf{x}$ and $\mathbf{y}$ and a constant $k \in \mathbb{N}$, we denote by $\mathbf{x} \approx_k \mathbf{y}$ the *k-region-equivalence* between $\mathbf{x}$ and $\mathbf{y}$, defined as follows:

Definition 4.2 (k-region-equivalence). *The* k-region-equivalence relation $\approx_k$ *on two* n*-dimensional clock valuations* $\mathbf{x}$ *and* $\mathbf{y}$ *is defined by*

$$\mathbf{x} \approx_k \mathbf{y} \quad \textit{iff} \quad \forall i \leq n : \left(\begin{array}{c} (x_i > k) \wedge (y_i > k) \\ \vee\ (int(x_i) = int(y_i) \wedge (fr(x_i) = 0 \Leftrightarrow fr(y_i) = 0) \wedge \\ \forall j \leq n : (fr(x_i) \leq fr(x_j) \Leftrightarrow fr(y_i) \leq fr(y_j))) \end{array} \right),$$

Here, for a clock valuation $\mathbf{x} \in \mathbb{R}^n_{\geq 0}$, x_i denotes its i-th component, i.e., the value of the i-th clock, and $int(x_i)$ and $fr(x_i)$ denote the integer and fractional parts of x_i, respectively. By $[\mathbf{x}]_k$ we denote the *k-region* containing $\mathbf{x}$, which is the equivalence class induced by $\approx_k$.

The semantics of an ETA is given in terms of its *timed transition system*, which consists of a (potentially infinite) set of *states* of the form $(l, \mathbf{x}, \mathbf{v})$, where $l \in L$, $\mathbf{x} \in \mathbb{R}^n_{\geq 0}$, and $\mathbf{v} \in D^m$. The transitions between such states result in the formation of *paths* through the timed transition system, defined as follows:

Definition 4.3 (Path). *A* path π *through a timed transition system is a (possibly infinite) sequence* $\pi = \langle (l_0, \mathbf{0}, \mathbf{v}_0) \xrightarrow{d_0} (l_1, \mathbf{x}_1, \mathbf{v}_1) \xrightarrow{e_1} (l_2, \mathbf{x}_2, \mathbf{v}_2) \xrightarrow{d_2} (l_3, \mathbf{x}_3, \mathbf{v}_3) \xrightarrow{e_3} \ldots \rangle$ *of states with delays* $d_i \in \mathbb{R}_{\geq 0}$ *and edges* $e_i \in E$*, subject to the following* initiation *and* consecution *conditions:*

1. *Initiation:* l_0 *is the initial location, and* $\mathbf{0}$ *and* $\mathbf{v}_0$ *are the initial clock and data valuations, respectively:*
2. *Consecution (time-passage): for even* i *with* $(l_i, \mathbf{x}_i, \mathbf{v}_i) \xrightarrow{d_i} (l_{i+1}, \mathbf{x}_{i+1}, \mathbf{v}_{i+1})$ *we require* $l_{i+1} = l_i$, $\mathbf{x}_{i+1} = \mathbf{x}_i + d_i$ *with* $\mathbf{x}_{i+1} \in Inv(l_i)$, *and* $\mathbf{v}_{i+1} = \mathbf{v}_i$*, where we write* $\mathbf{x}_i + d_i$ *to denote the addition of* d_i *uniformly to each component of* $\mathbf{x}_i$*.*
3. *Consecution (edges): for odd* i *with* $(l_i, \mathbf{x}_i, \mathbf{v}_i) \xrightarrow{e_i} (l_{i+1}, \mathbf{x}_{i+1}, \mathbf{v}_{i+1})$ *we require that* e_i *is of the form* $e_i = (l_i, a_i, g_i, r_i, l_{i+1})$*, where* $(\mathbf{x}_i, \mathbf{v}_i) \in Inv(l_i) \cap g$, $\mathbf{x}_{i+1} = r(\mathbf{x}_i)$ *with* $\mathbf{x}_{i+1} \in Inv(l_{i+1})$, *and* $\mathbf{v}_{i+1} = r(\mathbf{v}_i)$*, where* $r(\mathbf{x})$ *denotes the clock valuation obtained from* $\mathbf{x}$ *after resetting all the clocks in* r*, while* $r(\mathbf{v})$ *denotes the data valuation obtained from* $\mathbf{v}$ *by suitably updating all the data variables in* r*.*

The reachable state space of the ETA A, denoted $Reach(A)$, is then given by the set of states reachable from the initial state through transitions of all paths, with $Reach_i(A)$ denoting the set of reachable states of A after i iterations of its transition relation, defined as follows:

Definition 4.4 (Reachable state space of ETA). $Reach(A) \subseteq L \times \mathbb{R}^n_{\geq 0} \times D^m$ *is the* reachable state space *of an ETA* A*, consisting of a (potentially infinite) set of states of the form* $(l, \mathbf{x}, \mathbf{v})$*, where* $l \in L$, $\mathbf{x} \in \mathbb{R}^n_{\geq 0}$, *and* $\mathbf{v} \in D^m$*. It is defined inductively as follows, with* $Reach_i(A)$ *denoting the reach-set under* $i \in \mathbb{N}$ *steps, starting from the initial state* $(l_0, \mathbf{0}, \mathbf{v}_0)$ *and alternating between time-passage and discrete transitions:*

- $Reach_0(A) = \{(l_0, \mathbf{0}, \mathbf{v}_0)\}$,
- $Reach_{i+1}(A) = Reach_i(A) \cup Succ(Reach_i(A))$,
  where if $i \geq 0$ *is even,*
  $$Succ(Reach_i(A)) = \left\{ (l, \mathbf{x}, \mathbf{v}) \,\middle|\, \begin{array}{l} \exists\, \mathbf{x}' \in Inv(l) \;\; \exists\, d \in \mathbb{R}^n_{\geq 0} : (l, \mathbf{x}', \mathbf{v}) \in Reach_i(A) \\ \wedge \;\; \mathbf{x} = \mathbf{x}' + d \;\; \wedge \;\; \mathbf{x} \in Inv(l) \end{array} \right\}$$

and if $i \geq 0$ *is odd,*
$$Succ(Reach_i(A)) = \left\{ (l, \mathbf{x}, \mathbf{v}) \,\middle|\, \begin{array}{l} \exists\, e = (l', a, g, r, l) \in E \\ \exists\, (\mathbf{x}', \mathbf{v}') \in Inv(l') \cap g \\ (l', \mathbf{x}', \mathbf{v}') \in Reach_i(A) \\ \wedge \;\; \mathbf{x} = r(\mathbf{x}') \\ \wedge \;\; \mathbf{v} = r(\mathbf{v}') \end{array} \right\} :$$

- $Reach(A) = \bigcup_{i \in \mathbb{N}} Reach_i(A)$.

This leads to the notion of *reachability equivalence* denoted by $\equiv$. Given two ETA A_1 and A_2, we define $A_1 \equiv A_2$ iff $\forall i \in \mathbb{N} : Reach_i(A_1) = Reach_i(A_2)$. Thus $\equiv$ requires equal sets of reachable states after every iteration of the transition relation.

ETA compositions.

So far we considered ETA operating in isolation. In practice, real-time systems *communicate* with each other and their environment. This results in *composite* systems with communicating components. The communication is via synchronizing actions drawn from a shared alphabet and via shared data variables. We now consider four operators for constructing composite systems: *sequential*, *step*, *parallel*, and *layered* composition defined for ETA $A_i = (L_i, \Sigma_i, C_i, V_i, l_{0i}, l_{Fi}, Inv_i, E_i)$, $i = 1, 2$, with disjoint locations: $L_1 \cap L_2 = \emptyset$.

For modeling that the execution of A_1 is followed by that of A_2, it is convenient to have two composition operators at hand. The *sequential composition* $A_1; A_2$ *amalgamates* the final location l_{F1} of A_1 with the initial location l_{02} of A_2, while the *step composition* $A_1 \rhd A_2$ links l_{F1} and l_{02} by an explicit *step transition* t. Formally, $A_1; A_2$ has the location set $L_1 \cup L_2 \cup \{\widetilde{l_{F1}}\} \setminus \{l_{F1}, l_{02}\}$, where $\widetilde{l_{F1}}$ is a new location obtained by amalgamating l_{F1} with l_{02}, as given below:

Definition 4.5 (Sequential ;-composition). *Let $l_{01} \neq l_{F1}$ and $l_{02} \neq l_{F2}$, with l_{F1} having no outgoing edges. Then the* sequential composition *of A_1 and A_2 is defined as the ETA*

$$A_1; A_2 = (L_1 \cup L_2 \cup \{\widetilde{l_{F1}}\} \setminus \{l_{F1}, l_{02}\}, \Sigma_1 \cup \Sigma_2, C_1 \cup C_2, V_1 \cup V_2, l_{01}, l_{F2}, Inv, E),$$

where $\widetilde{l_{F1}}$ is disjoint from $L_1 \cup L_2$, $Inv(l_i) = Inv_i(l_i)$ for $l_i \in L_i$ and $i = 1, 2$, $Inv(\widetilde{l_{F1}}) = Inv_1(l_{F1}) \wedge Inv_2(l_{02})$, and E is given by:

$$\begin{aligned}
E = &(E_1 \setminus \{(l_1, a_1, g_1, r_1, l_{F1}) \mid (l_1, a_1, g_1, r_1, l_{F1}) \in E_1\}) \\
&\cup \{(l_1, a_1, g_1, r_1, \widetilde{l_{F1}}) \mid (l_1, a_1, g_1, r_1, l_{F1}) \in E_1\} \\
&\cup (E_2 \setminus \{\{(l_{02}, a_2, g_2, r_2, l_2) \mid (l_{02}, a_2, g_2, r_2, l_2) \in E_2\} \cup \\
&\qquad \{(l_2, a_2, g_2, r_2, l_{02}) \mid (l_2, a_2, g_2, r_2, l_{02}) \in E_2\}\}) \\
&\cup \{(\widetilde{l_{F1}}, a_2, g_2, r_2, l_2) \mid (l_{02}, a_2, g_2, r_2, l_2) \in E_2 \ \wedge \ l_2 \neq l_{02}\} \\
&\cup \{(l_2, a_2, g_2, r_2, \widetilde{l_{F1}}) \mid (l_2, a_2, g_2, r_2, l_{02}) \in E_2 \ \wedge \ l_2 \neq l_{02}\} \\
&\cup \{(\widetilde{l_{F1}}, a_2, g_2, r_2, \widetilde{l_{F1}}) \mid (l_{02}, a_2, g_2, r_2, l_{02}) \in E_2\}.
\end{aligned}$$

The first two lines in the construction of E in $A_1; A_2$ deal with edges of A_1 leading to l_{F1}, while the last five lines deal with edges of A_2 that either leave or enter l_{02}. Note that we assume $Inv(l_{F1}) = true$ by Definition 4.1. We require that there is no outgoing edge from l_{F1}, as it would otherwise be possible to reenter A_1 from A_2 via incoming edges to l_{02} and outgoing edges from l_{F1}. The set of edges E is obtained by appropriately assigning $\widetilde{l_{F1}}$ as the target or source location for edges in E_1 entering l_{F1} and for edges in E_2 entering or leaving l_{02}. In the *step composition* $A_1 \rhd A_2$ defined below, the stepping transition t allows for l_{F1} to have outgoing transitions, as no location of A_1 will be re-entered once t has been executed.

Definition 4.6 (Step $\triangleright$-composition). *The* step composition *of A_1 and A_2, denoted $A_1 \triangleright A_2$, is the ETA $A_1 \triangleright A_2 = (L_1 \cup L_2, \Sigma_1 \cup \Sigma_2, C_1 \cup C_2, V_1 \cup V_2, l_{01}, l_{F2}, Inv, E)$, with $Inv(l_i) = Inv_i(l_i)$ for $l_i \in L_i$ and $i = 1, 2$, and $E = E_1 \cup E_2 \cup \{t\}$, where $t = (l_{F1}, \varepsilon, true, \emptyset, l_{02})$ steps from l_{F1} to l_{02}.*

Alternative definitions of sequential and step compositions for timed automata may be found in [BP99, DHQ+08]. Parallel composition $\|$ of ETA is in the CCS-style [Mil89], i.e., parallel ETA *synchronize* on common actions but also act autonomously on all actions – the latter is modelled by *interleaving*. In order to avoid any read-write and write-write conflicts w.r.t the shared variables in the parallel ETA, we require that edges with synchronizing actions are *non-interfering*, as defined below. For an edge $e = (l, a, g, r, l')$ of an ETA A its *write-set* $wr(e)$ is the set of all clocks and data variables appearing on the left-hand side of one of the reset operations in r, while its *read-set* $rd(e)$ is the set of all clocks and data variables appearing in the guard g or on the right-hand side of a reset operation in r.

Definition 4.7 (Non-interfering edges $\not\frown$). *Let E_1 and E_2 be sets of edges. The* non-interference relation $\not\frown \subseteq E_1 \times E_2$ *is defined for $e_1 \in E_1$ and $e_2 \in E_2$ by: $e_1 \not\frown e_2$ if $rd(e_1) \cap wr(e_2) = wr(e_1) \cap rd(e_2) = wr(e_1) \cap wr(e_2) = \emptyset$. If the latter condition does not hold, e_1 and e_2* interfere *and we write $e_1 \frown e_2$.*

The relation $\not\frown$ is canonically lifted to sets of edges (and consequently to ETA): $E_1 \not\frown E_2$ iff for all $e_1 \in E_1$ and $e_2 \in E_2$ we have $e_1 \not\frown e_2$. For two ETA A_1 and A_2 with respective edge-sets E_1 and E_2, we have that $A_1 \not\frown A_2$ when (1) $E_1 \not\frown E_2$, i.e., their edge-sets are non-interfering, and (2) $C_1 \cap C_2 = \emptyset$, i.e., their clock-sets are disjoint so as to eliminate timing-induced dependencies between A_1 and A_2 by the *wrapping* construction (cf. Definition 4.10). In the context of parallel composition $\|$, we require a more relaxed notion of *synchronized non-interference* on the constituent ETA A_1 and A_2 in order for $A_1 \| A_2$ to be well-formed.

Definition 4.8 (Synchronized non-interfering ETA $\not\frown_{sync}$). *ETA A_1 and A_2 over alphabets Σ_1 and Σ_2, respectively, are* synchronized non-interfering, *denoted $A_1 \not\frown_{sync} A_2$, if $\forall a \in \Sigma_{1?!} \setminus \{\tau, \varepsilon\} : \overline{a} \in \Sigma_{2?!} \implies edges_{A_1}(a) \not\frown edges_{A_2}(\overline{a})$.*

The relation $\not\frown_{sync}$ on ETA is only w.r.t synchronizing actions on common channels, and thus (unlike the more restrictive $\not\frown$ relation on ETA) does not preclude shared-variable and clock dependencies between actions on disjoint channels.

The *parallel composition* of two A_1 and A_2, with $A_1 \not\frown_{sync} A_2$, is then constructed according to a CCS-style synchronization and interleaving, as follows:

Definition 4.9 (Parallel $\|$-composition). *When $A_1 \not\frown_{sync} A_2$, their* parallel composition *is defined by $A_1 \| A_2 = (L_1 \times L_2, \Sigma_1 \cup \Sigma_2, C_1 \cup C_2, V_1 \cup V_2, (l_{01}, l_{02}), (l_{F1}, l_{F2}), Inv, E)$, where $\forall (l_1, l_2) \in L_1 \times L_2 : Inv(l_1, l_2) = Inv_1(l_1) \wedge Inv_2(l_2)$ and E given by:*

- *Synchronization: If $e_1 = (l_1, a_1, g_1, r_1, l_1') \in E_1$ and $e_2 = (l_2, a_2, g_2, r_2, l_2') \in E_2$ with $a_2 = \overline{a}_1$ then $((l_1, l_2), \tau, g_1 \wedge g_2, r_1 r_2, (l_1', l_2')) \in E$.*
- *Interleaving: If $e_1 = (l_1, a_1, g_1, r_1, l_1') \in E_1$ then $\forall\, l_2 \in L_2 : ((l_1, l_2), a_1, g_1, r_1, (l_1', l_2)) \in E$. Conversely, if $e_2 = (l_2, a_2, g_2, r_2, l_2') \in E_2$ then $\forall l_1 \in L_1 : ((l_1, l_2), a_2, g_2, r_2, (l_1, l_2')) \in E$.*

Note that in the synchronization case it is not relevant whether we take $r_1 r_2$ (first the list r_1 of reset operations and then r_2) as above or the reverse order $r_2 r_1$ because we assume the synchronized non-interference $e_1 \not\frown e_2$.

Our CCS-style interleaving allows A_1 and A_2 to act autonomously on all actions, thus leaving $A_1 \| A_2$ *open* to further (parallel) composition with an ETA A_3, with which the interleaving edges of $A_1 \| A_2$ may synchronize. This contrasts with the semantics of UPPAAL for ETA networks, where every discrete transition is represented as having a τ-labelled internal action (obtained by synchronization of complementary ! and ? actions) for the symbolic computation of the state-space of a parallel composition. Thus UPPAAL views $A_1 \| A_2$ as a *closed* network having only τ-labelled edges. Such a closed network semantics may be obtained for our parallel composition by means of *channel restriction* (cf. Definition 4.13 on p. 146 of [OD08]) for eliminating interleaving edges in $A_1 \| A_2$.

A real-time distributed system often consists of (sequential) phases that execute in parallel on multiple platforms, wherein a transition (edge) within a given phase can execute only after all *dependent* transitions (edges) in each preceding phase have been executed. It is clear that the non-interference relation of Definition 4.7 sufficiently captures (in-)dependence in the untimed setting, where dependencies are induced only by shared variables. In the timed setting of ETA, however, the clocks of the various system components evolve *synchronously*, resulting in *timing-induced* dependencies even in the presence of disjoint sets of clocks. In contrast to several works on the partial order reduction of TA (e.g., [BJLY98, Min99, LNZ05]) that deal with such timing-induced dependencies by imposing the semantic condition of *local time* (where, in addition to mutual disjointness, the clocks of the constituent components run entirely independent of each other), we retain here the synchrony between the clocks of the various components as in the UPPAAL model-checker, but decouple the timing influences between the components by *wrapping* an ETA with an initial location that admits idling for arbitrarily long periods before proceeding to its actual execution, as given below:

Definition 4.10 (Wrapped ETA). *An ETA* $A = (L, \Sigma, C, V, l_0, l_F, Inv, E)$ *is* wrapped *if* $Inv(l_0) = true$ *and every edge* $e \in E$ *leaving* l_0 *is of the form* $e = (l_0, \varepsilon, true, r, l)$*, where* r *resets all clocks in* C *to 0. If* A *is wrapped, we denote this by writing* $[A]$.

Remark 4.1. The arbitrary idling permitted in l_0 in Definition 4.10 mimics local time semantics when $[A]$ is considered in the context of a (parallel) composition. Intuitively, $[A]$ is protected against time influences from components working in parallel, as long as it stays in its initial location. We illustrate this by comparing the parallel composition of two ETA with that of their wrapped versions in Figure 4.1. Note that the ETA A and B therein do not share any variable, and have disjoint (but synchronous) clocks x and y. Thus $A \not\frown B$. The synchronous evolution of x and y however results in a *timing induced dependency* between the edges a and b, with a never occurring after b owing to the corresponding guards. Wrapping A and B decouples this timing induced dependency within a parallel composition (owing to the preceding clock resets).

We now introduce an asymmetric layered composition operator $\bullet$ (intermediate between parallel and sequential composition) that involves the non-interference relation on edges of ETA. The *layered composition* of A_1 and A_2 is given by $A_1 \bullet A_2$

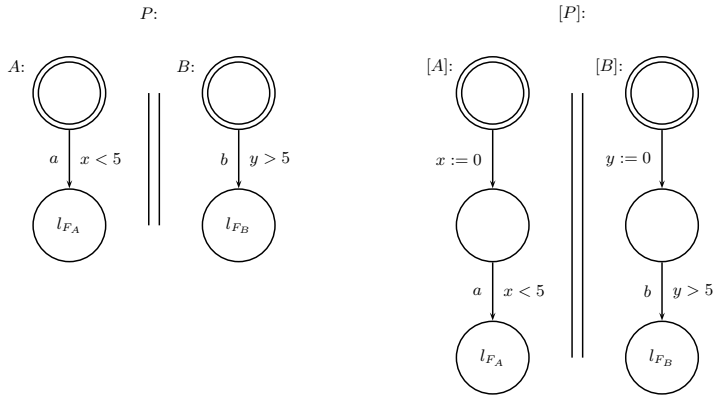

Fig. 4.1. Here, in $P = A \| B$ to the left, A and B are not wrapped. Thus, although $A \not\frown B$, the edge a can never occur after the edge b in P, as the clocks x and y are synchronous. In the wrapped version to the right, where $[P] = [A] \| [B]$, the preceding resets allow the edges a and b to occur in either order in $[P]$.

where $A_1 \not\frown_{sync} A_2$, and Inv is as in the parallel composition $A_1 \| A_2$, while E is a subset of the set of edges of $A_1 \| A_2$, as an edge of A_2 is allowed to execute in $A_1 \bullet A_2$ only after all *dependent* edges of A_1 have been executed. This layered composition is defined formally below:

Definition 4.11 (Layered •-composition). *When* $A_1 \not\frown_{sync} A_2$*, their* layered composition *is defined by*

$$A_1 \bullet A_2 = (L_1 \times L_2, \Sigma_1 \cup \Sigma_2, C_1 \cup C_2, V_1 \cup V_2, (l_{01}, l_{02}), (l_{F1}, l_{F2}), Inv, E),$$

where Inv *is as in the parallel composition* $A_1 \| A_2$*, while* E *now differs in the second part of the interleaving case:*

- *Synchronization: If* $e_1 = (l_1, a_1, g_1, r_1, l_1') \in E_1$ *and* $e_2 = (l_2, a_2, g_2, r_2, l_2') \in E_2$ *with* $a_2 = \overline{a}_1$ *then* $((l_1, l_2), \tau, g_1 \wedge g_2, r_1 r_2, (l_1', l_2')) \in E$.
- *Interleaving: If* $e_1 = (l_1, a_1, g_1, r_1, l_1') \in E_1$ *then* $\forall\, l_2 \in L_2 : ((l_1, l_2), a_1, g_1, r_1, (l_1', l_2)) \in E$.
  If $e_2 = (l_2, a_2, g_2, r_2, l_2') \in E_2$ *and* $\forall\, l_1, l_1^* \in L_1 : l_1 \xrightarrow{*} l_1^* \;\; \forall\, e_1 = ({l_1}^*, a_1, g_1, r_1, l_1') \in E_1 : e_1 \not\frown e_2$ *then* $((l_1, l_2), a_2, g_2, r_2, (l_1, l_2')) \in E$

where $l_1 \xrightarrow{*} l_1^*$ *expresses that* l_1^* *is reachable from* l_1 *in the syntactic structure of* A_1 *through an arbitrary sequence of edges.*

Thus only the second part of the interleaving case differs from parallel composition: an interleaving edge of A_2 is allowed to execute only after all *dependent* edges of A_1 (in the sense of the $\frown$-relation) have been executed. A natural setting for such asymmetric interleaving arises when A_1 edges write to variables that are then read by (dependent) A_2 edges.

The four operators for ETA compositions, namely those for sequential (;), step ($\triangleright$), parallel ($\|$), and layered ($\bullet$) compositions, are illustrated in Figure 4.2.

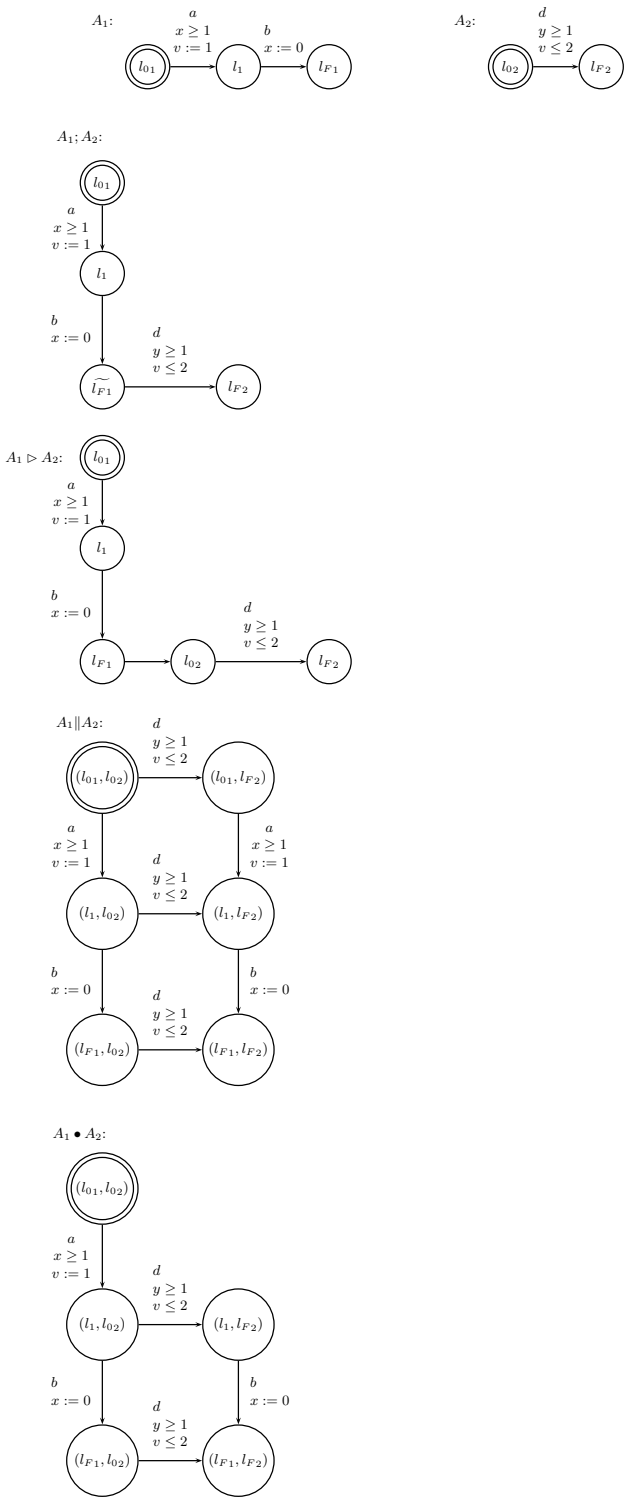

Fig. 4.2. Composition operators for two ETA A_1 and A_2 with clocks x, y, a shared data variable v, actions a, b, d, assumed to be non-complementary, and an interference $e_a \frown e_d$ of the edge labeled by a and the edge labeled by d induced by the variable v. In the step composition $A_1 \rhd A_2$, the ε-label on the stepping transition has been omitted. Initial locations are marked with an additionally inscribed circle.

4.3 CCL Laws and Equivalences

This section formalizes communication closed layer equivalences for ETA networks under suitable non-interference conditions. We begin with an adaptation of the CCL law of [Jan94] to the setting of ETA.

Theorem 4.1 (CCL with •-composition). *For all ETA A_1, A_2 and B_1, B_2 with $A_1 \not\frown B_2$ and $B_1 \not\frown A_2$ the following* communication closed layer *law holds:*

$$(A_1 \bullet A_2) \,\|\, (B_1 \bullet B_2) \;=\; (A_1 \| B_1) \bullet (A_2 \| B_2). \textit{(CCL)}$$

Proof. We show the CCL equality, where we identify location tuples $(l_{A_1}, l_{A_2}, l_{B_1}, l_{B_2})$ of the ETA of the LHS with reordered location tuples $(l_{A_1}, l_{B_1}, l_{A_2}, l_{B_2})$ of the ETA of the RHS.

Consider a location $l = (l_{A_1}, l_{A_2}, l_{B_1}, l_{B_2})$ of $(A_1 \bullet A_2) \| (B_1 \bullet B_2)$ and an edge $e = (l, a, g, r, l')$ starting at this location. By the definition of $\bullet$ and $\|$, the edge e could be due to either (a1) A_1, (a2) B_1, (a3) A_2, (a4) B_2 individually, or as a synchronization involving (b1) A_1 and A_2, (b2) A_1 and B_1, (b3) A_1 and B_2, (b4) A_2 and B_1, (b5) A_2 and B_2. We have to show that e also occurs at the location l of $(A_1 \| B_1) \bullet (A_2 \| B_2)$.

The cases (a1) and (a2) are relatively straightforward, whereas the cases (a3) and (a4) require more care. Of these we consider (a3) in detail. Here e is permitted in the layered composition $A_1 \bullet A_2$, i.e., from l_{A_1} there is no location syntactically reachable where an edge e_1 starts that interfers with e. Then e can occur at l in $A_2 \| B_2$ by the interleaving case and in $(A_1 \| B_1) \bullet (A_2 \| B_2)$ as it is permitted by the layered composition with $A_1 \| B_1$. This holds because e is permitted by A_1 and by B_1 due to the assumption $B_1 \not\frown A_2$. The case (a4) is symmetric.

Of the synchronization cases we consider (b5) in detail. It combines arguments for the individual cases (a3) and (a4). There are edges with complementary actions occurring in A_2 and B_2 individually that can synchronize to an edge labeled with τ in the context of $A_2 \| B_2$. The part of the τ-edge stemming from A_2 was permitted in $A_1 \bullet A_2$, and the part of the τ-edge stemming from B_2 was permitted in $B_1 \bullet B_2$. Furthermore, since $A_1 \not\frown B_2$ and $B_1 \not\frown A_2$, the part of the τ-edge stemming from A_2 is non-interfering with all edges in B_1, and the part of the τ-edge stemming from B_2 is non-interfering with all edges in A_1. Thus the τ-edge itself occurs in $(A_1 \| B_1) \bullet (A_2 \| B_2)$.

So indeed e occurs also at the location l of $(A_1 \| B_1) \bullet (A_2 \| B_2)$.

Vice versa, consider a location $l = (l_{A_1}, l_{B_1}, l_{A_2}, l_{B_2})$ and an edge $e = (l, a, g, r, l')$ starting at this location in $(A_1 \| B_1) \bullet (A_2 \| B_2)$. The same cases (a1)–(a4) and (b1)–(b5) need to be considered. □

Remark 4.2. 1. Since equality is a *congruence* w.r.t. parallel composition, the CCL law holds also in the context of an arbitrary parallel ETA C. For example, we have $((A_1 \bullet A_2) \| (B_1 \bullet B_2)) \| C = ((A_1 \| B_1) \bullet (A_2 \| B_2)) \| C$. Clearly, equality of ETA implies reachability equivalence. So the CCL law remains valid when we replace $=$ by $\equiv$. Note that reachability equivalence itself is not a congruence

w.r.t. parallel composition because it ignores the action labels of the edges, which are decisive for synchronization with a parallel ETA.

2. Theorem 4.1 holds also when the ETA appearing to the right of the $\bullet$-operator are specialized to wrapped ones, i.e., provided $A_1 \not\frown B_2$ and $B_1 \not\frown A_2$, we have $(A_1 \bullet [A_2]) \| (B_1 \bullet [B_2]) = (A_1 \| B_1) \bullet ([A_2] \| [B_2])$.
3. Theorem 4.1 may be generalized to multiple parallel and layered instances of ETA as follows: If there are *no cross-interferences*, i.e., $A_{i,j} \not\frown A_{k,l}$ for $i \neq k$ and $j \neq l$, then

$$\begin{array}{c||c||c} A_{1,1} & \cdots & A_{1,m} \\ \bullet & \cdots & \bullet \\ \cdots & \cdots & \cdots \\ \bullet & \cdots & \bullet \\ A_{n,1} & \cdots & A_{n,m} \end{array} \quad = \quad \begin{array}{ccc} (A_{1,1} & \| \cdots \| & A_{1,m}) \\ \bullet & & \bullet \\ \cdots & \cdots & \cdots \\ \bullet & & \bullet \\ (A_{n,1} & \| \cdots \| & A_{n,m}) \end{array}$$

The above equality between the parallel and layered versions is again upto a reordering of location tuples.

PO-equivalence.

We now formalize *partial order* (po-) equivalence as a means of relating step and layered compositions. For this relationship, we have to address the fact that in a layered composition $A_1 \bullet A_2$ there may be τ-edges arising from synchronization of complementary actions, whereas such τ-edges do not arise in the step composition $A_1 \rhd A_2$. For this purpose, we introduce for a path π of $A_1 \bullet A_2$ (or of $A_1 \| A_2$) the operation $split(\pi)$ that *splits* every synchronization edge of π (labelled with τ) into a sequence of its constituent input and output edges, which is possible owing to the synchronized non-interference assumed for edges labelled with complementary actions (see Definition 4.8). Note that this non-interference implies that the order in which synchronization edges are split into constituent input and output edges is irrelevant, such that we may always choose the (unique) representative path in which the output action always precedes the complementary input action with zero-delay in between. Consider a finite path π of $A_1 \bullet A_2$ or of $A_1 \| A_2$ with fragments π' and π'' of the form

$$\pi = \pi' \xrightarrow{d} ((l_{A_1}, l_{A_2}), \mathbf{x}, \mathbf{u}) \xrightarrow{\tau} ((l'_{A_1}, l'_{A_2}), \mathbf{x}'', \mathbf{u}'') \xrightarrow{d''} \pi'',$$

where the τ-action has resulted from synchronizing an action $a = \alpha! \in \Sigma_{1_{?!}}$ with $\overline{a} = \alpha? \in \Sigma_{2_{?!}}$. Then
$split(\pi) = \pi' \xrightarrow{d} ((l_{A_1}, l_{A_2}), \mathbf{x}, \mathbf{u}) \xrightarrow{\alpha!} ((l'_{A_1}, l_{A_2}), \mathbf{x}', \mathbf{u}') \xrightarrow{0}$
$((l'_{A_1}, l_{A_2}), \mathbf{x}', \mathbf{u}') \xrightarrow{\alpha?} ((l'_{A_1}, l'_{A_2})\mathbf{x}'', \mathbf{u}'') \xrightarrow{d''} \pi''$, where the intermediate valuations $\mathbf{x}'$ and $\mathbf{u}'$ are uniquely determined by the resets of the $\alpha!$-labelled edge of A_1 on $\mathbf{x}$ and $\mathbf{u}$. Following such a splitting of τ-edges, we may now define po-equivalence on paths of ETA.

Definition 4.12 (po-equivalence $\equiv_{po}$ of paths). *Let A_1 and A_2 be ETA sharing an alphabet Σ, with Π_1 and Π_2 denoting the corresponding sets of finite paths terminating in their final locations l_{F_1} resp. l_{F_2}. Let $\approx$ be a relation between the locations of A_1 and A_2. Then $\pi_1 \in \Pi_1$ is* po-equivalent *to $\pi_2 \in \Pi_2$, denoted $\pi_1 \equiv_{po} \pi_2$, relative*

to $\approx$ on the corresponding locations, $\approx_k$ on the corresponding clock-valuations (where k is the maximum constant of A_1 and A_2), and the identity on the corresponding data valuations, if split(π_2) can be obtained from split(π_1), time-abstractedly and by ignoring stuttering *ε-labelled edges (of the form $(l, \varepsilon, true, \emptyset, l')$, by repeated permutation of adjacent independent edges separated by only one time-passage.*

For illustration, consider two (split and ε-free) path fragments

$$\pi_1 = \langle (l_{01}, \mathbf{0}, \mathbf{v}_0) \xrightarrow{d_0} (l_1, \mathbf{x}_1, \mathbf{u}_1) \xrightarrow{e} (l_2, \mathbf{x}_2, \mathbf{u}_2) \xrightarrow{d_2} (l_3, \mathbf{x}_3, \mathbf{u}_3) \xrightarrow{f} (l_4, \mathbf{x}_4, \mathbf{u}_4) \rangle$$

and

$$\pi_2 = \langle (l_{02}, \mathbf{0}, \mathbf{v}_0) \xrightarrow{d'_0} (l'_1, \mathbf{y}_1, \mathbf{v}_1) \xrightarrow{f} (l'_2, \mathbf{y}_2, \mathbf{v}_2) \xrightarrow{d'_2} (l'_3, \mathbf{y}_3, \mathbf{v}_3) \xrightarrow{e} (l'_4, \mathbf{y}_4, \mathbf{v}_4) \rangle,$$

where $d_0, d_2, d'_0, d'_2 \in \mathbb{R}_{\geq 0}$. Then $\pi_1 \equiv_{po} \pi_2$ relative to $\approx$ with $l_{01} \approx l_{02}$, $\forall 1 \leq i \leq 4 : l_i \approx l'_i$, $\mathbf{x}_4 \approx_k \mathbf{y}_4, \mathbf{u}_4 = \mathbf{v}_4$, and $e \not\frown f$. Thus, π_1 and π_2 (relative to $\approx$ on their locations, region-equivalence on their clock valuations, and the identity on their data valuations) differ only in the (permutative) ordering of *independent* transitions. Note that we need the k-region-equivalence $\approx_k$ to relate the (final) clock valuations $\mathbf{x}_4$ and $\mathbf{y}_4$ of owing to the intermediate delays in one path being *time abstractedly* (and not exactly) matched in the other path. Note also that we ignore ε-labelled *stutter* edges for $\equiv_{po}$ as such edges are only of the form $(l, \varepsilon, true, \emptyset, l')$ (e.g., as encountered while stepping from A_1 to A_2, cf. Definition 4.6). As we ignore ε-transitions for $\equiv_{po}$, two po-equivalent paths may not necessarily have the same length. This definition of $\equiv_{po}$ is now lifted to ETA as follows:

Definition 4.13 (po-equivalence $\equiv_{po}$ of ETA). *For ETA A_1 and A_2 sharing a common alphabet Σ, with Π_1 and Π_2 denoting the corresponding sets of finite paths terminating in their respective final locations l_{F1} and l_{F2} respectively, we write $A_1 \equiv_{po} A_2$ iff $\forall \pi_i \in \Pi_i \;\; \exists \pi_{3-i} \in \Pi_{3-i} : \pi_i \equiv_{po} \pi_{3-i}$, where $i \in \{1, 2\}$.*

The following notion of *layered normal form* is then used for permuting paths of layered compositions.

Definition 4.14 (Layered normal form). *A (finite, terminating) path π of $A_1 \bullet A_2$ is in* layered normal form (LNF) *if it consists of consecutive edges from E_1 passing through l_{F1}, followed by consecutive edges from E_2 ending in l_{F2}.*

In a path π of $A_1 \bullet A_2$ in LNF, the A_2-transitions are *delayed* until all A_1-transitions have occurred. This may be too late because the clock constraints of the A_2-transitions may not be satisfied any more. To avoid this issue, we wrap A_2 so that starting $[A_2]$ resets all clocks of A_2. We thus mimic local time semantics for A_2. Now Proposition 4.1 states that every path of $A_1 \bullet [A_2]$ can be rewritten into a po-equivalent path in LNF, which leads to Proposition 4.2 establishing po-equivalence between layered and step compositions of ETA.

Proposition 4.1 ($\equiv_{po}$ and LNF). *Consider an ETA A_1 with the final location l_{F1} and a wrapped ETA $[A_2]$ with the final location l_{F2}. Let Π denote the set of all finite paths of $A_1 \bullet [A_2]$ terminating in (l_{F1}, l_{F2}), and $\Pi_L \subseteq \Pi$ the subset of the paths in LNF. Then we have that $\forall \pi \in \Pi \;\; \exists \pi' \in \Pi_L : \pi \equiv_{po} \pi'$.*

Proof. Take a path $\pi \in \Pi$ of $A_1 \bullet [A_2]$. Since there may be synchronization edges (labelled with τ) in π, we need to demonstrate that $split(\pi)$ may be suitably permuted into a path of $A_1 \bullet [A_2]$ in LNF. By construction, $split(\pi)$ has passed through the final location l_{F_1} of A_1, and contains only interleavings of edges from A_1 and A_2. If edges of A_2, say in the order $e_1, \ldots, e_n$, occur in $split(\pi)$ before A_1 has reached its final location l_{F1}, we permute these edges "to the right" so that they occur after l_{F1} has been reached, thereby generating a path $\pi' \in \Pi_L$ with $\pi \equiv_{po} \pi'$.

Such permutations are possible, as by Definition 4.11 of $\bullet$, the edges $e_1, \ldots, e_n$ satisfy $e \not\frown e_i$ for all edges e of A_1 that need to be permuted "to the left", before the occurrence of l_{F_1} in $split(\pi)$. Note that regarding the global time, the edges $e_1, \ldots, e_n$ (that now have been permuted "to the right" of l_{F_1}) are delayed, but this delay does not prevent e_1 (and subsequently $e_2, \ldots, e_n$) from occurring, because A_2 is wrapped. Moreover, as $Inv(l_{F1}) = true$ by Definition 4.1, it is possible for A_1 to spend any amount of time in l_{F1}, irrespective of whether l_{F1} is reached along π or π'. Hence the intitial edge e_1 can start after any delay, thereby resetting the clocks of A_2 (cf. Definition 4.10), thus enabling $e_2, \ldots, e_n$ to proceed according to their local time in A_2, as would be the case for the path $\pi' \in \Pi_L$. □

Proposition 4.2 ($\equiv_{po}$ between $\bullet$- and $\rhd$-compositions). *For an ETA A_1 and a wrapped ETA $[A_2]$, we have that $A_1 \bullet [A_2] \;\equiv_{po}\; A_1 \rhd [A_2]$.*

Proof. We provide a sketch of the proof here by first introducing a relation $\approx$ that relates locations of $A_1 \bullet [A_2]$ with those of $A_1 \rhd [A_2]$, defined by $\forall l_1 \in L_1 : (l_1, l_{02}) \approx l_1$ and $\forall l_2 \in L_2 : (l_{F1}, l_2) \approx l_2$.

Let $\Pi^{\rhd}$ (resp. $\Pi^{\bullet}$) be the set of all finite paths of $A_1 \rhd [A_2]$ (resp. $A_1 \bullet [A_2]$) terminating in l_{F2} (resp. (l_{F1}, l_{F2})). Then

- $\forall \pi \in \Pi^{\rhd} \quad \exists \pi' \in \Pi^{\bullet} : \pi' \equiv_{po} \pi$, where π' is in LNF.
- $\forall \pi' \in \Pi^{\bullet} \quad \exists \pi \in \Pi^{\rhd} : \pi \equiv_{po} \pi'$, where π' is either in LNF, or is po-equivalent to another path in $\Pi^{\bullet}$ that is in LNF (cf. Proposition 4.1).

The po-equivalence between π and π' above is relative to $\approx$ between their respective locations (when π' is in LNF), region equivalence (w.r.t the maximum of all constants of A_1 and A_2) between the respective clock valuations, and identity between the respective data valuations, modulo the ε-labelled stepping transition in π. The po-equivalence between $A_1 \rhd [A_2]$ and $A_1 \bullet [A_2]$ follows as an immediate consequence. □

Replacing $\bullet$ by $\rhd$ in the CCL-law of Theorem 4.1 yields po-equivalences for wrapped ETA $[A_2]$ and $[B_2]$.

Theorem 4.2 ($\equiv_{po}$-CCL with $\rhd$-composition). *For ETA A_1, B_1 and wrapped ETA A_2, B_2 with $A_1 \not\frown B_2$ and $B_1 \not\frown A_2$ the CCL law*

$$(A_1 \rhd [A_2]) \parallel (B_1 \rhd [B_2]) \;\equiv_{po}\; (A_1 \| B_1) \rhd ([A_2] \parallel [B_2]) \textit{(CCL-step-po)}$$

holds for po-equivalence.

Proof. Let $A = A_1 \rhd [A_2]$, $B = B_1 \rhd [B_2]$, $S_1 = A_1 \| B_1$, and $S_2 = [A_2] \| [B_2]$. We then write $S = S_1 \rhd S_2$ and $P = A \| B$, where it needs to be shown that $P \equiv_{po} S$. Let Π_P resp. Π_S denote the set of finite paths of P resp. S terminating in their final

location $(l_{F_{A_2}}, l_{F_{B_2}})$, where $l_{F_{A_2}}$ resp. $l_{F_{B_2}}$ is the final location of A_2 resp. B_2. Then clearly $\Pi_S \subseteq \Pi_P$. It then remains to show that $\forall \pi \in \Pi_P \ \exists \pi' \in \Pi_S : \pi \equiv_{po} \pi'$. As π may contain synchronized τ-edges, we need to demonstrate that $split(\pi)$ may be suitably permuted into a (split) path of S. By construction, $split(\pi)$ has passed through the final location $(l_{F_{A_1}}, l_{F_{B_1}})$ of S_1, and contains only interleavings of edges from A_1, B_1, A_2, and B_2. If edges of A_2, say in the order $e_{A_1}, \ldots, e_{A_n}$, and those of B_2, say in the order $e_{B_1}, \ldots, e_{B_m}$, occur in $split(\pi)$ before $(l_{F_{A_1}}, l_{F_{B_1}})$ has been reached, we permute such edges "to the right" so that they occur after $(l_{F_{A_1}}, l_{F_{B_1}})$, thereby generating a path $\pi' \in \Pi_S$ with $\pi \equiv_{po} \pi'$. Note that the po-equivalence in this case involves the identity relation on locations of P and S.

Such permutations are possible, as by the side-condition $B_1 \not\frown A_2$, the edges $e_{A_1}, \ldots, e_{A_n}$ satisfy $e_B \not\frown e_{A_i}$ for all edges e_B from B_1 that need to be permuted "to the left", before the occurrence of $(l_{F_{A_1}}, l_{F_{B_1}})$ in $split(\pi)$. Likewise, by the side-condition $A_1 \not\frown B_2$, the edges $e_{B_1}, \ldots, e_{B_m}$ satisfy $e_A \not\frown e_{B_i}$ for all edges e_A of A_1 that again need to be permuted "to the left", before the occurrence of $(l_{F_{A_1}}, l_{F_{B_1}})$ in $split(\pi)$. Note that regarding the global time, the edges $e_{A_1}, \ldots, e_{A_n}$ and $e_{B_1}, \ldots, e_{B_m}$ (that now have been permuted "to the right" of $(l_{F_{A_1}}, l_{F_{B_1}})$) are delayed, but these delays do not prevent e_{A_1} and e_{B_1} (and subsequently $e_{A_2}, \ldots, e_{A_n}$ and $e_{B_2}, \ldots, e_{B_m}$) from occurring, because both A_2 and B_2 are wrapped. Hence the intitial edges e_{A_1} and e_{B_1} can start after arbitrary delays, thereby resetting the clocks of A_2 resp. B_2 (cf. Definition 4.10), thus enabling $e_{A_2}, \ldots, e_{A_n}$ and $e_{B_2}, \ldots, e_{B_m}$ to proceed according to their local times in $[A_2]\|[B_2]$, as would be the case for the path $\pi' \in \Pi_S$. □

Remark 4.3. 1. As stated earlier in Remark 4.2, the equality induced by the CCL law of Theorem 4.1 is a *congruence* preserved in arbitrary parallel contexts, i.e, if $A_1 \not\frown B_2$ and $B_1 \not\frown A_2$, then for any ETA C, we have $((A_1 \bullet [A_2])\|(B_1 \bullet [B_2]))\|C = ((A_1\|B_1) \bullet ([A_2]\|[B_2]))\|C$. However, when we replace the $\bullet$ by $\rhd$, the resultant po-equivalences from Theorem 4.2 are in general not preserved in arbitrary parallel contexts, i.e, even if $A_1 \not\frown B_2$ and $B_1 \not\frown A_2$ it generally does not follow that $((A_1 \rhd [A_2])\|(B_1 \rhd [B_2]))\|C \equiv_{po} ((A_1\|B_1) \rhd ([A_2]\|[B_2]))\|C$, except when C is independent of all the other ETA. By contrast, the equalities of Theorem 4.1 and $\equiv_{po}$ of Theorem 4.2 are both preserved in arbitrary step contexts, i.e, for any ETA C, when $A_1 \not\frown B_2$ and $B_1 \not\frown A_2$, we have $((A_1 \bullet [A_2])\|(B_1 \bullet [B_2])) \rhd C = ((A_1\|B_1) \bullet ([A_2]\|[B_2])) \rhd C$, and $((A_1 \rhd [A_2])\|(B_1 \rhd [B_2])) \rhd C \equiv_{po} ((A_1\|B_1) \rhd ([A_2]\|[B_2])) \rhd C$.

2. Theorem 4.2 may also be generalized to multiple parallel and step instances of ETA: If there are *no cross-interferences*, i.e., $A_{i,j} \not\frown A_{k,l}$ for $i \neq k$ and $j \neq l$, and if all $A_{i,j}$ for $2 \leq i \leq n$ are wrapped, then

$$\begin{array}{c||c||c} A_{1,1} & \cdots & A_{1,m} \\ \rhd & \cdots & \rhd \\ \cdots & \cdots & \cdots \\ \rhd & \cdots & \rhd \\ [A_{n,1}] & \cdots & [A_{n,m}] \end{array} \quad \equiv_{po} \quad \begin{array}{c} (A_{1,1} \ \| \ \cdots \ \| \ A_{1,m}) \\ \rhd \\ \cdots \\ \rhd \\ ([A_{n,1}] \ \| \ \cdots \ \| \ [A_{n,m}]) \end{array}$$

3. The ETA equivalences seen so far are related thus: $= \ \Rightarrow \ \equiv \ \Rightarrow \ \equiv_{po}$.

Localized reachability in non-terminating ETA.

The po-equivalence $\equiv_{po}$ that relates parallel and step-compositions of (wrapped) ETA under suitable "cross-independence" conditions (cf. Theorem 4.2) is restrictive, in the sense that $\equiv_{po}$ examines only finite paths (terminating in the final locations) for possible permutations. In practice it is often necessary to examine ETA with non-terminating paths (typically arising in distributed reactive real-time systems that (networks of) ETA seek to model). To this end, we introduce a location-based equivalence $\equiv_{\mathcal{L}}$ that *localizes reachability within a layer*. As with $\equiv_{po}$, this *layered reachability equivalence* $\equiv_{\mathcal{L}}$ arises by the replacement of $\bullet$ by $\rhd$ within the CCL law. However, unlike $\equiv_{po}$, the equivalence $\equiv_{\mathcal{L}}$ may also be applied to ETA with possibly non-terminating paths, and suffices for the preservation of *mutual exclusion* properties, as will be shown in Section 4.6.

Definition 4.15 (Layered reachability equivalence $\equiv_{\mathcal{L}}$). *Let $Reachloc(A)$ denote the set of reachable locations of A. Consider ETA P and S, with both having sub-ETA A_1, A_2, B_1, B_2 with disjoint sets of locations. We define* layered reachability equivalence $P \equiv_{\mathcal{L}} S$ *as follows:*

$$\forall\, i \in \{1,2\}\ \forall\, l_a \in L_{A_i}, l_b \in L_{B_i} : (l_a, l_b) \in Reachloc(P) \text{ iff } (l_a, l_b) \in Reachloc(S).$$

The relation $\equiv_{\mathcal{L}}$ is an equivalence relation on ETA with the structure as given above, where the sub-ETA A_1, A_2, B_1, and B_2 are combined in P and S by means of parallel and step compositions. In the following theorem, we consider an ETA P with parallel composition as the top-most operator and an ETA S with the step composition as the top-most operator. In both ETA, the sub-ETA A_1, B_1 constitute one layer and the sub-ETA A_2, B_2 a second layer. Due to the localized property of $\equiv_{\mathcal{L}}$, wherein one considers only location reachability *within a layer*, the issue of (non-)termination is implicitly handled: for the two ETA P and S, with $P \equiv_{\mathcal{L}} S$ as in Definition 4.15, we consider only the cases that either both P and S are still in their first layer (with control residing at a location pair drawn from locations in A_1 and B_1), or that both P and S have terminated their first layer (after having passed through its final location $(l_{F_{A_1}}, l_{F_{B_1}})$) and are now in their second layer (with control residing at a location pair drawn from locations in A_2 and B_2). Cross-layer properties are not necessarily preserved between P and S. Indeed, for a location-pair $l_a \in L_{A_2},\ l_b \in L_{B_1}$, we may have $(l_a, l_b) \in Reachloc(P) \setminus Reachloc(S)$, because B_1 and hence the first layer $(A_1 \| B_1)$ in S may not terminate.

Theorem 4.3 below establishes $\equiv_{\mathcal{L}}$ between P and S when layered composition is replaced by step-composition in the CCL-law.

Theorem 4.3 ($\equiv_{\mathcal{L}}$-CCL with $\rhd$-composition). *Given ETA A_1, B_1, and wrapped ETA $[A_2], [B_2]$, with $A_1 \not\frown B_2$ and $B_1 \not\frown A_2$. Then the CCL law*

$$(A_1 \rhd [A_2]) \,\|\, (B_1 \rhd [B_2]) \ \equiv_{\mathcal{L}}\ (A_1 \| B_1) \rhd ([A_2] \,\|\, [B_2]) \quad \textit{(CCL-step-}\mathcal{L}\textit{)}$$

holds for the layered reachability equivalence $\equiv_{\mathcal{L}}$.

Proof. Let $P = (A_1 \rhd [A_2]) \,\|\, (B_1 \rhd [B_2])$ and $S = (A_1 \,\|\, B_1) \rhd ([A_2] \,\|\, [B_2])$. We show $P \equiv_{\mathcal{L}} S$. Consider a location pair $l_a \in L_{A_1},\ l_b \in L_{B_1}$ in the first layer consisting of A_1 and B_1. Then clearly $(l_a, l_b) \in Reachloc(P)$ iff $(l_a, l_b) \in Reachloc(S)$.

Now consider a location pair $l_a \in L_{A_2},\ l_b \in L_{B_2}$ in the second layer consisting of A_2 and B_2. If $(l_a, l_b) \in \mathit{Reachloc}(S)$ then clearly $(l_a, l_b) \in \mathit{Reachloc}(P)$ as any path of P can mimic the path of S reaching (l_a, l_b).

The only difficult case is $(l_a, l_b) \in \mathit{Reachloc}(P)$. It implies that individually A_1 and B_1 have terminated and taken their steps to the initial locations of A_2 and B_2 before reaching $l_a \in L_{A_2}$ and $l_b \in L_{B_2}$ inside P. Since $A_1 \not\frown B_2$ and $B_1 \not\frown A_2$, reaching $l_a \in L_{A_2}$ does not depend on events of B_2, and vice versa, reaching $l_b \in L_{B_2}$ does not depend on events of A_2. Furthermore, since A_2 and B_2 are wrapped, they may idle for an arbitrary time and then start taking one of their initial edges. Thus $(l_a, l_b) \in \mathit{Reachloc}(S)$ by first waiting for the termination of both A_1 and B_1, next taking the joint step to the initial locations of A_2 and B_2, and then continuing in A_2 and B_2 to reach l_a and l_b. □

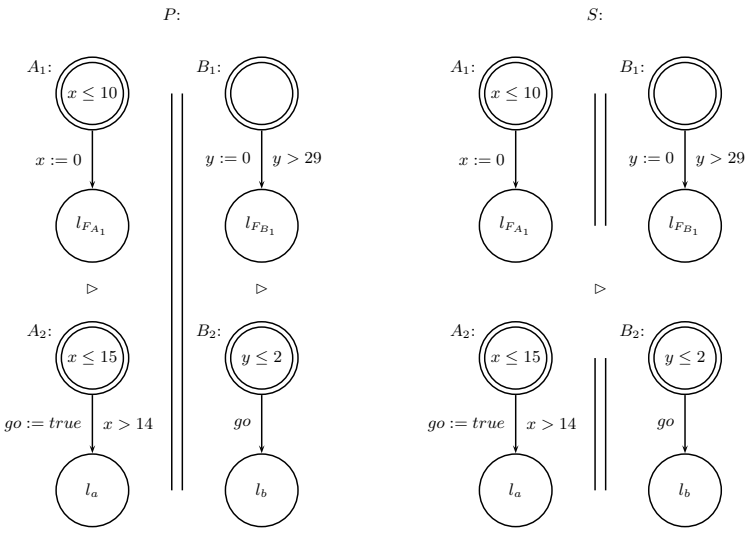

Fig. 4.3. Here A_2 and B_2 are not wrapped. Initially, the Boolean variable *go* is false. Then – unlike in Theorem 4.3 – P and S have different reachable locations within layer 2: $(l_a, l_b) \in \mathit{Reachloc}(P)$ but $(l_a, l_b) \notin \mathit{Reachloc}(S)$.

Remark 4.4. 1. We show that wrapped A_2 and B_2 are necessary for Theorem 4.3 to hold. In Figure 4.3 the ETA A_2 and B_2 are not wrapped. On the left-hand side the parallel composition P is displayed and on the right-hand side the step composition S. In P the transition of A_2 can be taken after at most 25 seconds global time, setting the Boolean variable *go* to true, and the transition of B_2 checks the guard *go* after more than 29 seconds global time, which is then found true. Hence $(l_a, l_b) \in \mathit{Reachloc}(P)$. However, in S the final location pair

$(l_{F_{A_1}}, l_{F_{B_1}})$ is reached only after 29 seconds global time (with $x > 19$ due to the local reset of x). Thus the invariant $x \leq 15$ in the initial location of A_2 is violated. So $(l_a, l_b) \notin \mathit{Reachloc}(S)$, violating Theorem 4.3.
Even if the clocks x and y were reset anew at the joint step in S, the invariant $y \leq 2$ of the initial location of B_2 would force the transition of B_2 to check its guard *go* when it is still false because the transition of A_2 setting *go* to true is taken only after 14 seconds. So $(l_a, l_b) \notin \mathit{Reachloc}(S)$, again violating Theorem 4.3.
However, wrapping A_2 and B_2 will add new initial locations with the invariant true to A_2 and B_2, and the transitions leaving this new location will reset the clocks x and y. Then $(l_a, l_b) \in \mathit{Reachloc}(P)$ and $(l_a, l_b) \in \mathit{Reachloc}(S)$, thus preserving Theorem 4.3.

2. We may generalize Theorem 4.3 to multiple parallel and step instances of ETA, as with Theorems 4.1 and 4.2. Thus, if there are *no cross-interferences*, i.e., $A_{i,j} \not\frown A_{k,l}$ for $i \neq k$ and $j \neq l$, and if all $A_{i,j}$ for $2 \leq i \leq n$ are wrapped, then

$$\begin{array}{c||c||c} A_{1,1} & \cdots & A_{1,m} \\ \triangleright & \cdots & \triangleright \\ \cdots & \cdots & \cdots \\ \triangleright & \cdots & \triangleright \\ [A_{n,1}] & \cdots & [A_{n,m}] \end{array} \quad \equiv_{\mathcal{L}} \quad \begin{array}{ccccc} (A_{1,1} & \| & \cdots & \| & A_{1,m}) \\ & & \triangleright & & \\ \cdots & & \cdots & & \cdots \\ & & \triangleright & & \\ ([A_{n,1}] & \| & \cdots & \| & [A_{n,m}]) \end{array}$$

4.4 Precedence CCL

We now introduce a semantic condition termed *precedence* and demonstrate its use in establishing equivalences analogous to Theorems 4.1, 4.2, and 4.3 for ETA networks that do not respect the non-interference conditions discussed in the previous section.

Definition 4.16 (Precedence $\prec$ in ETA). *For ETA A_1, A_2 and C_1, C_2, where the final location $l_{F_{A_1}}$ of A_1 has no outgoing edge, we say that A_1* precedes *A_2 in the* parallel context *of C_1 and C_2, denoted $A_1 \prec_{C_1,C_2} A_2$ if for each state $((l_{A_1}, l_{C1}, l_{A2}, l_{C2}), \mathbf{x}, \mathbf{u}) \in \mathit{Reach}(A_1 \| C_1 \| A_2 \| C_2)$ and each edge e_2 of A_2 that is enabled at this state, $l_{A_1} = l_{F_{A_1}}$ holds, i.e., A_1 is at its final location.*

Informally, events of A_2 depend on the completion of A_1, even in the parallel context with C_1 and C_2. In practice, for this semantic property sufficient syntactic conditions on the data or clock parts of the guards of edges are checked. For example, a *timed precedence* might be formalized by considering a single clock x shared between A_1 and A_2, constrained at every location of A_1 except for l_{F_1} by the invariant $x \leq 10$, with every edge of A_1 being unguarded and without resets on x, and with every initial edge of A_2 guarded by the clock condition $x > 10$. Thus, in this setting A_1 needs at most 10 seconds to terminate, and A_2 needs at least 10 seconds to start, which then ensures that A_1 precedes A_2 in any parallel context. A *data precedence* might be formalized using a Boolean variable *go* that is initialized with false. If every initial edge of A_2 checks the Boolean condition *go* and only those edges leading to the final location of A_1 have the update $go := true$ then A_2 can only start after A_1 has terminated, provided the parallel context with C_1 and C_2 does not set *go* to true.

While the po- and layered reachability equivalences of the preceding section exploit non-interference together with (wrapping-simulated) local time, precedence exploits the implicit synchronization due to global time or the write-read order due to interference from having shared data variables, giving the stronger reachability equivalence $\equiv$ under certain "cross-precedence" conditions in compositions of ETA, as indicated by the following theorem.

Theorem 4.4 ($\prec$-CCL with $\bullet$-composition). *For ETA A_1, A_2 and B_1, B_2 with the precedence conditions $A_1 \prec_{A_2,B_1} B_2$ and $B_1 \prec_{A_1,B_2} A_2$ the* precedence communication closed layer *law*

$$(A_1 \bullet A_2) \| (B_1 \bullet B_2) \;\equiv\; (A_1 \| B_1) \bullet (A_2 \| B_2) \textit{(PrecCCL)}$$

holds for the reachability equivalence $\equiv$.

Proof. We show $(A_1 \bullet A_2) \| (B_1 \bullet B_2) \equiv (A_1 \| B_1) \bullet (A_2 \| B_2)$, i.e., the sets of reachable states are equal at every iteration i of their transition relations, where we identify location tuples $(l_{A_1}, l_{A_2}, l_{B_1}, l_{B_2})$ appearing in the states of the LHS with reordered location tuples $(l_{A_1}, l_{B_1}, l_{A_2}, l_{B_2})$ appearing in the states of the RHS.

The containment $Reach_i((A_1 \| B_1) \bullet (A_2 \| B_2)) \subseteq Reach_i((A_1 \bullet A_2) \| (B_1 \bullet B_2))$ is easy to check, as the parallel composition operator dominates on the right-hand side and the layered composition operator on the left-hand side. The dominance of layered composition induces fewer interleavings on the basis of the respective dependencies, as seen from the earlier definitions.

We now show $Reach_i((A_1 \bullet A_2) \| (B_1 \bullet B_2)) \subseteq Reach_i((A_1 \| B_1) \bullet (A_2 \| B_2))$ by induction over i.

Induction Basis. This case $i = 0$ is obvious.

Assume that the containment holds for some i.

Induction Step. Consider $((l_{A_1}, l_{A_2}, l_{B_1}, l_{B_2}), \mathbf{x}, \mathbf{u}) \in Reach_{i+1}((A_1 \bullet A_2) \| (B_1 \bullet B_2))$.

If $((l_{A_1}, l_{A_2}, l_{B_1}, l_{B_2}), \mathbf{x}, \mathbf{u}) \in Reach_i((A_1 \bullet A_2) \| (B_1 \bullet B_2))$ the proof is immediate from the induction hypothesis. We now examine $((l_{A_1}, l_{A_2}, l_{B_1}, l_{B_2}), \mathbf{x}, \mathbf{u}) \in Succ(Reach_i((A_1 \bullet A_2) \| (B_1 \bullet B_2))$. If i is even, the preceding transition corresponds to time-passage, which is possible also in $(A_1 \| B_1) \bullet (A_2 \| B_2)$, and the proof then follows immediately. If i is odd, the preceding transition could have been performed either by (a1) A_1, (a2) B_1, (a3) A_2, (a4) B_2 individually, or as a synchronization involving A_1 and A_2, A_1 and B_1, A_1 and B_2, A_2 and B_1, A_2 and B_2.

The cases (a1) and (a2) are relatively straightforward. The cases (a3) and (a4) require more care. We consider case (a3) in detail. There exist $((l_{A_1}, l_{A_2}, l'_{B_1}, l_{B_2}), \mathbf{x}', \mathbf{u}') \in Reach_i((A_1 \bullet A_2) \| (B_1 \bullet B_2))$ and an edge $e_2 = (l'_{A_2}, a, g, r, l_{A_2})$ of A_2 which can occur at the location $(l_{A_1}, l_{A_2}, l'_{B_1}, l_{B_2})$ of $((A_1 \bullet A_2) \| (B_1 \bullet B_2))$ such that $(\mathbf{x}', \mathbf{u}') \in Inv(l_{A_1}, l_{A_2}, l'_{B_1}, l_{B_2}) \cap g$, with $\mathbf{x} = r(\mathbf{x}')$, $\mathbf{u} = r(\mathbf{u}')$. Hence e_2 is permitted by the layered composition with A_1, i.e., from l_{A_1} there is no location syntactically reachable where an edge e_1 with $e_1 \frown e_2$ starts. Moreover, the precedence condition $B_1 \prec_{A_1,B_2} A_2$ implies that $l_{B_1} = l_{F_{B_1}}$, i.e., B_1 is at its final location. Therefore e_2 can also occur at $((l_{A_1}, l_{B_1}, l'_{A_2}, l_{B_2}), \mathbf{x}', \mathbf{u}') \in Reach_i((A_1 \| B_1) \bullet (A_2 \| B_2))$, yielding $((l_{A_1}, l_{B_1}, l_{A_2}, l_{B_2}), \mathbf{x}, \mathbf{u}) \in Reach_{i+1}((A_1 \| B_1) \bullet (A_2 \| B_2))$.

Each of the synchronization cases combines two individual edges labeled with complementary actions yielding the label τ. For the individual edges the argument is the same as above. $\square$

The following theorem looks at the Precedence CCL with layered composition replaced by sequential composition.

Theorem 4.5 ($\prec$-CCL with ;-composition). *Theorem 4.4 remains valid when replacing $\bullet$ by ; in the ETA expressions:*

$$(A_1; A_2) \| (B_1; B_2) \;\equiv\; (A_1 \| B_1); (A_2 \| B_2). \textit{(PrecCCL-seq)}$$

Proof. We show $(A_1; A_2) \| (B_1; B_2) \equiv (A_1 \| B_1); (A_2 \| B_2)$, i.e., the sets of reachable states are equal at every iteration i of their transition relations, where the locations appearing in the states on both sides are now of the form (l_A, l_B). The proof has the same structure as the one of Theorem 4.4, except for changes taking ; instead of $\bullet$ in account. We focus on the induction step $i \longrightarrow i+1$ of the proof of (now) $Reach_i((A_1; A_2) \| (B_1; B_2)) \subseteq Reach_i((A_1 \| B_1); (A_2 \| B_2))$.

Consider $((l_A, l_B), \mathbf{x}, \mathbf{u}) \in Reach_{i+1}((A_1; A_2) \| (B_1; B_2))$ and case (a3) as in the proof of Theorem 4.4, where the transitions stems from A_2. There exist $((l'_A, l_B), \mathbf{x}', \mathbf{u}') \in Reach_i((A_1; A_2) \| (B_1; B_2))$ and an edge $e_2 = (l'_A, a, g, r, l_A)$ of A_2 which can occur at the location (l'_A, l_B) of $((A_1; A_2) \| (B_1; B_2))$ such that $(\mathbf{x}', \mathbf{u}') \in Inv(l'_A, l_B) \cap g$, with $\mathbf{x} = r(\mathbf{x}')$, $\mathbf{u} = r(\mathbf{u}')$. Clearly, e_2 occurs after termination of A_1. Moreover, the precedence condition $B_1 \prec_{A_1, B_2} A_2$ implies here that e_2 occurs after termination of B_1. Therefore e_2 can also occur at $((l'_A, l_B), \mathbf{x}', \mathbf{u}') \in Reach_i((A_1 \| B_1); (A_2 \| B_2))$, yielding $((l_A, l_B), \mathbf{x}, \mathbf{u}) \in Reach_{i+1}((A_1 \| B_1); (A_2 \| B_2))$. □

Remark 4.5.

1. Unlike the CCL law of Theorem 4.1, the Precedence CCL law of Theorems 4.4 does not hold for the equality =. Note also that Theorem 4.5 does not require the ETA A_2 and B_2 to be wrapped, in contrast to Theorems 4.2 and 4.3. This is because due to the precedence relation, even in $(A_1; A_2) \| (B_1; B_2)$ the ETA A_2 and B_2 can only start when both A_1 and B_1 have terminated, and then A_2 and B_2 can start at the same global time as in $(A_1 \| B_1); (A_2 \| B_2)$. As $\equiv$ is not a congruence w.r.t. parallel composition, the Precedence CCL laws do not yield equivalences in an arbitrary parallel context.
2. Generalizations of Theorems 4.4 and 4.5 to multiple parallel, layered, and sequential instances of ETA require strong precedence conditions. Thus, if $A_{i,j} \prec_{\{A_{p,q}\}} A_{k,l}$ for all $i < k$ and $j < l$ in the parallel context of all other $A_{p,q}$, then:

$$\begin{array}{c||c||c} A_{1,1} & \cdots & A_{1,m} \\ \bullet & \cdots & \bullet \\ \cdots & \cdots & \cdots \\ \bullet & \cdots & \bullet \\ A_{n,1} & \cdots & A_{n,m} \end{array} \equiv \begin{array}{ccccc} (A_{1,1} & \| & \cdots & \| & A_{1,m}) \\ & & \bullet & & \\ & & \cdots & & \\ & & \bullet & & \\ (A_{n,1} & \| & \cdots & \| & A_{n,m}) \end{array}$$

This also holds also under identical precedence conditions with $\bullet$ replaced by ; .

4.5 Separation and Flattening

In this section, we consider two transformations on cycles in ETA. *Separation* reduces the nesting of cycles, while *flattening* reduces the number of cycles. These

transformations are sound (in the sense of reachability preservation) under the assumption that the ETA involved have *memoryless cycles*. Such an assumption is justified for protocols where each cycle performs some service, and there is no need to carry over some information from one service cycle to the next. Separation was studied in [Coh00] in the setting of Kleene algebras, where, under certain conditions, a nondeterministic iteration of the form $(a+b)^*$ could be separated into a sequence a^*b^* of iterations, with a and b being regular expressions for programs in a Kleene algebra. Iteration is implicit in ETA, as cycling through locations is permitted in the model (except possibly through the final location in some cases, cf. Definition 4.5). The + operator for non-deterministic choice on regular expressions is adapted to the setting of ETA as follows:

Definition 4.17 (Choice +-composition). *For ETA* $A_i = (L_i, \Sigma_i, C_i, V_i, l_{0_i}, l_{F_i}, Inv_i, E_i)$, $i \in \{1,2\}$, *with* $L_1 \cap L_2 = \emptyset$, *the* choice composition *of* A_1 *and* A_2 *is defined as the ETA* $A_1 + A_2 = (L_1 \cup L_2 \cup \{l_0, l_F\}, \Sigma_1 \cup \Sigma_2, C_1 \cup C_2, V_1 \cup V_2, l_0, l_F, Inv, E)$,, *where* $l_0, l_F \notin L_1 \cup L_2$, $Inv(l_i) = Inv_i(l_i)$ *for* $l_i \in L_i$ *and* $i = 1, 2$,
$Inv(l_0) = Inv_1(l_{0_1}) \wedge Inv_2(l_{0_2})$, *and*

$$E = \{(l_0, a, g, r, l) \mid (l_{0_1}, a, g, r, l) \in E_1 \vee (l_{0_2}, a, g, r, l) \in E_2\} \cup \{(l_{F_i}, \varepsilon, true, \emptyset, l_F) \mid i = 1, 2\} \cup E_1 \cup E_2.$$

This operator + for choice composition of ETA implements the corresponding rules of operational semantics in CCS [Mil89], where $A_1 \xrightarrow{a} B_1$ and $A_2 \xrightarrow{b} B_2$ imply that $A_1 + A_2 \xrightarrow{a} B_1$ and $A_1 + A_2 \xrightarrow{b} B_2$. In the above definition, the new location l_0 mimics the process algebraic expression $A_1 + A_2$. While the initial locations l_{0_1} and l_{0_2} may be re-entered in A_1 resp. A_2, the initial transitions of $A_1 + A_2$ from the new location l_0 unfold initial cycles, such that once an initial transition of A_1 has been executed in $A_1 + A_2$, the other component A_2 will never be executed, and vice-versa. Such a non-deterministic choice between the execution of component ETA may be enforced in practice by appropriate invariants and guards on the clocks and data variables, as will be seen in the next section.

In some applications one component may have to be (re-)entered even after the other has been executed, as will again be seen in the next section. To this end, the following *union* operator (adapted from the UNITY program notation [CM88]) is needed to suitably glue the two component ETA together, where the initial and final locations of a given ETA are identical.

Definition 4.18 (Union $\cup$). *For ETA* $A_i = (L_i, \Sigma_i, C_i, V_i, l_{0_i}, l_{0_i}, Inv_i, E_i)$, $i \in \{1,2\}$, *with* $L_1 \cap L_2 = \emptyset$, *the* union *of* A_1 *and* A_2 *is defined by the ETA*

$$A_1 \cup A_2 = (L_1 \cup L_2 \cup \{l_0\} \setminus \{l_{01}, l_{02}\}, \Sigma_1 \cup \Sigma_2, C_1 \cup C_2, V_1 \cup V_2, l_0, l_0, Inv, E)$$

where l_0 *is disjoint from* $L_1 \cup L_2$, $Inv(l_i) = Inv_i(l_i)$ *for* $l_i \in L_i$ *and* $i = 1, 2$, $Inv(l_0) = Inv_1(l_{0_1}) \wedge Inv_2(l_{0_2})$, *and* E *is given by*

$$
\begin{aligned}
E = (&E_1 \setminus \{\{(l_{01}, a_1, g_1, r_1, l_1) \mid (l_{01}, a_1, g_1, r_1, l_1) \in E_1\} \cup \\
&\qquad \{(l_1, a_1, g_1, r_1, l_{01}) \mid (l_1, a_1, g_1, r_1, l_{01}) \in E_1\}\}) \\
\cup\ &\{l_0, a_1, g_1, r_1, l_1) \mid (l_{01}, a_1, g_1, r_1, l_1) \in E_1 \ \wedge\ l_1 \neq l_{01}\} \\
\cup\ &\{(l_1, a_1, g_1, r_1, l_0) \mid (l_1, a_1, g_1, r_1, l_{01}) \in E_1 \ \wedge\ l_1 \neq l_{01}\} \\
\cup\ &\{(l_0, a_1, g_1, r_1, l_0) \mid (l_{01}, a_1, g_1, r_1, l_{01}) \in E_1\}) \\
\cup\ &(E_2 \setminus \{\{(l_{02}, a_2, g_2, r_2, l_2) \mid (l_{02}, a_2, g_2, r_2, l_2) \in E_2\} \cup \\
&\qquad \{(l_2, a_2, g_2, r_2, l_{02}) \mid (l_2, a_2, g_2, r_2, l_{02}) \in E_2\}\}) \\
\cup\ &\{l_0, a_2, g_2, r_2, l_2) \mid (l_{02}, a_2, g_2, r_2, l_2) \in E_2 \ \wedge\ l_2 \neq l_{02}\} \\
\cup\ &\{(l_2, a_2, g_2, r_2, l_0) \mid (l_2, a_2, g_2, r_2, l_{02}) \in E_2 \ \wedge\ l_2 \neq l_{02}\} \\
\cup\ &\{(l_0, a_2, g_2, r_2, l_0) \mid (l_{02}, a_2, g_2, r_2, l_{02}) \in E_2\}.
\end{aligned}
$$

The first five lines in the construction of E in $A_1 \cup A_2$ deal with edges of A_1 that either leave or enter l_{01}, while the last five lines deal with edges of A_2 that either leave or enter l_{02}. The set of edges E is obtained by appropriately assigning l_0 as the target or source location for edges in E_1 entering or leaving l_{01} and for edges in E_2 entering or leaving l_{02}. While in the sequential composition $A_1; A_2$ we require that $l_{01} \neq l_{F1}$ and $l_{02} \neq l_{F2}$ with no outgoing edges from l_{F1}, which is amalgamated with l_{02} into $\widetilde{l_{F1}}$, the union $A_1 \cup A_2$ requires that $l_{01} = l_{F1}$ and $l_{02} = l_{F2}$, which are then amalgamated into l_0. Whereas in the union $A_1 \cup A_2$ possible cycles of A_1 and A_2 are glued together in their initial location, in $A_1 \rhd A_2$ the new step transition t separates the A_1 cycles from the A_2 cycles so that all A_1 cycles are performed before the A_2 cycles, while in $A_1 + A_2$ either the cycles of A_1 are performed or those of A_2, but not both. The subtle difference between the $\cup$ and $+$ operators is illustrated in Figure 4.4.

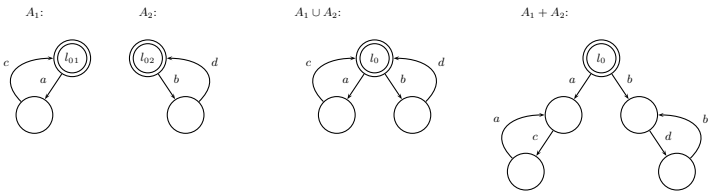

Fig. 4.4. The difference between $\cup$ and $+$ becomes apparent when there are loops at the initial locations. While $\cup$ glues them simply together, $+$ unfolds them before joining. Data and timing aspects have been abstracted away for simplicity.

An alternate definition of the $\cup$ operator for TA may be found in [BP99]. The Separation Theorems 4.6 and 4.7 examine (cyclic) behaviours of the $\cup$ and $\rhd$ operators on ETA under *memoryless* initial locations, where the notion of a location being memoryless is as defined below:

Definition 4.19 (Memoryless locations in ETA). *A location l of an ETA A is said to be* memoryless *if l is always entered with the same valuations of the clocks and the data variables.*

A sufficient syntactic condition for a location l to be memoryless is that all cycles through l have strong resets. A cycle of an ETA A through a location l is said to have *strong resets* if every transition entering l resets all clocks and all data variables to their initial valuations. To simplify the reasoning about cycles, we wish to transform the union of ETA with memoryless cycles into their step composition. The Separation Theorem 4.6 shows that this transformation respects a weak reachability equivalence $\equiv_r$ sufficient for the preservation of safety properties, as seen next.

Definition 4.20 (Weak reachability equivalence $\equiv_r$). *Consider ETA $A = (L, \Sigma, C, V, l_0, l_F, Inv, E)$ and*
$A' = (L', \Sigma, C, V, l_0', l_F', Inv', E')$ defined over the same alphabet Σ and the same sets of clocks C and data variables V, with $|C| = n$ and $|V| = m$. We define the weak reachability equivalence *between A and A', denoted by $A \equiv_r A'$, relative to a relation $\approx \subseteq L \times L'$, if*

$$\forall\, l \in L\, \forall\, l' \in L'\, \forall \mathbf{x} \in \mathbb{R}^n_{\geq 0}\ \forall \mathbf{v} \in D^m :$$
$$1.\ (l, \mathbf{x}, \mathbf{v}) \in \mathit{Reach}(A) \ \Rightarrow\ \exists\, l' \in L' : l \approx l' \wedge (l', \mathbf{x}, \mathbf{v}) \in \mathit{Reach}(A')$$
$$2.\ (l', \mathbf{x}, \mathbf{v}) \in \mathit{Reach}(A') \ \Rightarrow\ \exists\, l \in L : l \approx l' \wedge (l, \mathbf{x}, \mathbf{v}) \in \mathit{Reach}(A)$$

where $\approx$ preserves location invariants, i.e., if $l \approx l'$ then $Inv(l) = Inv'(l')$.

Definition 4.20 of $\equiv_r$ is a relaxation of the notion of "emulation" introduced in Definition 2 of [CJ99]. The following theorem shows the preservation of $\equiv_r$ by separation.

Theorem 4.6 (Separation). *Consider ETA A_1 and A_2 as in Definition 4.18 with memoryless initial locations l_{01} and l_{02}, both of which are (re-)entered with the designated initial clock and data valuations $\mathbf{0}$ and $\mathbf{v}_0$, respectively. Then $A_1 \cup A_2 \equiv_r A_1 \rhd A_2$ relative to the relation $\approx$ between the locations of A_1 and A_2, given by $\forall l \in (L_1 \cup L_2) \setminus \{l_0\} : l \approx l$ and $l_0 \approx l_{01}$ and $l_0 \approx l_{02}$. See Figure 4.5.*

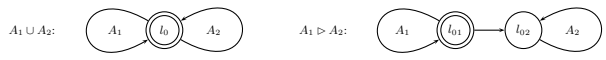

Fig. 4.5. This figure illustrates the effect of separation. Data and timing aspects have been abstracted away for simplicity.

Proof. Let $A = A_1 \cup A_2$ and $\Pi(A)$ denote the set of all paths of A. Then a path $\pi \in \Pi(A)$ may repeatedly cycle through A_1 and $A2$ in any order. We show that such a path π may be transformed into one of $A' = A_1 \rhd A_2$, consisting of consecutive

paths of A_1 followed by consecutive paths of A_2 plus possibly some extra final part, while preserving weak reachability of configurations in the sense of $\equiv_r$ relative to the location-relation $\approx$.

We have that $Reach(A) = \bigcup_{\pi \in \Pi(A)} Reach(\pi)$, where $Reach(\pi)$ denotes the states of A that are reachable along π. Consider a typical path $\pi \in \Pi(A)$, for example of the form $\pi = \{l_0\}\pi_1\{l_0\}\pi_2\{l_0\}\tilde{\pi}_1\{l_0\}\tilde{\pi}_2\{l_0\}\pi_{fin1}$, where the assertion $\{l_0\}$ indicates that the control resides in the initial location l_0. The paths π_1 and $\tilde{\pi}_1$ represent cycles through A_1, the paths π_2 and $\tilde{\pi}_2$ represent cycles through A_2, and π_{fin1} represents the (possibly empty) final part of π, say inside A_1, that does not reach l_0 again. So π alternates twice between A_1 and A_2 before finishing inside A_1.

Since both l_{01} and l_{02} are memoryless, and owing to the preservation by $\approx$ of (downward closed) location invariants (cf. Definitions 4.1 and 4.20), the path π can be transformed into two paths of $A' = A_1 \rhd A_2$, namely
$\pi' = \{l_{0_1}\}\pi_1\{l_{0_1}\}\tilde{\pi}_1\{l_{0_1}\}t\{l_{0_2}\}\pi_2\{l_{0_2}\}\tilde{\pi}_2\{l_{0_2}\}$ and $\pi'_{fin1} = \{l_{0_1}\}\pi_{fin1}$, where t refers to the stepping transition between A_1 and A_2 in $A_1 \rhd A_2$.

For Condition 1 of $\equiv_r$, we calculate for any $(l, \mathbf{x}, \mathbf{v}) \in Reach(\pi)$ the following:

$(l, \mathbf{x}, \mathbf{v}) \in Reach(\pi)$
$\Rightarrow \quad (l, \mathbf{x}, \mathbf{v}) \in Reach(\pi_1) \cup Reach(\pi_2) \cup Reach(\tilde{\pi}_1) \cup Reach(\tilde{\pi}_2) \cup Reach(\pi_{fin1})$
$\Rightarrow \quad \exists l' : l \approx l' \wedge$
$\quad (l', \mathbf{x}, \mathbf{v}) \in Reach(\pi_1) \cup Reach(\tilde{\pi}_1) \cup Reach(t \cdot \pi_2) \cup Reach(\tilde{\pi}_2) \cup Reach(\pi_{fin1})$
$\Rightarrow \quad \exists l' : l \approx l' \wedge (l', \mathbf{x}, \mathbf{v}) \in Reach(\pi') \cup Reach(\pi'_{fin1})$

Thus altogether $(l, \mathbf{x}, \mathbf{v}) \in Reach(A) \;\Rightarrow\; \exists l' : l \approx l' \wedge (l', \mathbf{x}, \mathbf{v}) \in Reach(A')$ as desired. The converse Condition 2 is shown similarly. □

While $\equiv_r$ is not a congruence w.r.t. parallel composition in general, the following theorem states its preservation by parallel instances of separation under $\not\frown$-conditions similar to those of Theorems 4.2 and 4.3.

Theorem 4.7 (Separation in parallel context). *For ETA A_1, B_1, A_2, B_2 with memoryless initial locations, with A_2 and B_2 wrapped and satisfying $A_1 \not\frown B_2$ and $B_1 \not\frown A_2$, it holds that $(A_1 \cup [A_2]) \| (B_1 \cup [B_2]) \equiv_r (A_1 \rhd [A_2]) \| (B_1 \rhd [B_2])$.*

Proof. Let $L = (A_1 \cup A_2) \| (B_1 \cup B_2)$ and $R = (A_1 \rhd A_2) \| (B_1 \rhd B_2)$. In L a path π may repeatedly cycle through A_1 and $A2$ in any order, interleaved with cycles through B_1 and B_2 in any order. We have to show that π can be transformed into a path π' of R, where all cycles through A_1 and B_1 precede the cycles through A_2 and B_2.

By way of example, consider a typical part in a path

$$\pi = \cdots \{l_{0A}\}\{l_{0B}\}\pi_{A_{21}}\pi_{B_{21}}\pi_{A_{22}}\{l_{0A}\}\pi_{B_{22}}\{l_{0B}\}\pi_{A_{11}}\pi_{B_{11}}\pi_{A_{12}}\{l_{0A}\}\pi_{B_{12}}\{l_{0B}\}\cdots$$

of L, where the path $\pi_{A_{21}}\pi_{A_{22}}$ represents a cycle through A_2 and $\pi_{A_{11}}\pi_{A_{12}}$ a cycle through A_1, and analogously the path $\pi_{B_{21}}\pi_{B_{22}}$ represents a cycle through B_2 and $\pi_{B_{11}}\pi_{B_{12}}$ a cycle through B_1. In π the cycles through A_2 and A_1 are shown interleaved with the cycles through B_2 and B_1. The assertion $\{l_{0A}\}$ indicates that the control resides in the joint initial location l_{0A} of A_1 and A_2, and $\{l_{0B}\}$ indicates that the control resides in the joint initial location l_{0B} of B_1 and B_2.

We claim that π can be transformed into a path

$$\pi' = \cdots \{l_{0A}\}\{l_{0B}\}\pi_{A_{11}}\pi_{B_{11}}\pi_{A_{12}}\{l_{0A}\}\pi_{B_{12}}\{l_{0B}\}\pi_{A_{21}}\pi_{B_{21}}\pi_{A_{22}}\{l_{0A}\}\pi_{B_{22}}\{l_{0B}\}\cdots$$

of R. Indeed, after the second $\{l_{0A}\}$ the cycle $\pi_{A_{21}}\pi_{A_{22}}$ can start and after the second $\{l_{0B}\}$ the cycle $\pi_{B_{21}}\pi_{B_{22}}$. Since A_2 and B_2 are wrapped, these cycles need not start immediately after $\{l_{0A}\}$ and $\{l_{0B}\}$. For instance, $\pi_{A_{21}}\pi_{A_{22}}$ starts only after the second $\{l_{0B}\}$ in π' above. Since $B_1 \not\frown A_2$, no part of $\pi_{A_{21}}\pi_{A_{22}}$ interferes with any part of $\pi_{B_{11}}\pi_{B_{12}}$, and vice versa, since $A_1 \not\frown B_2$, no part of $\pi_{B_{21}}\pi_{B_{22}}$ interferes with any part of $\pi_{A_{11}}\pi_{A_{12}}$. Thus the cycles $\pi_{A_{21}}\pi_{A_{22}}$ and $\pi_{B_{21}}\pi_{B_{22}}$ could be pushed back in π'. □

Remark 4.6. Generalization of Theorem 4.7 to multiple parallel, union, and step instances of ETA is possible under side-conditions similar to those required for the generalizations of Theorems 4.2 and 4.3, cf. Remarks 4.3 and 4.4. Thus, if there are *no cross-interferences*, i.e., $A_{i,j} \not\frown A_{k,l}$ for $i \neq k$ and $j \neq l$, and if all $A_{i,j}$ for $2 \leq i \leq n$ are wrapped, then

$$\begin{array}{c|c|c} A_{1,1} & \cdots & A_{1,m} \\ \cup & \cdots & \cup \\ \cdots & \cdots & \cdots \\ \cup & \cdots & \cup \\ [A_{n,1}] & \cdots & [A_{n,m}] \end{array} \equiv_r \begin{array}{c|c|c} A_{1,1} & \cdots & A_{1,m} \\ \rhd & \cdots & \rhd \\ \cdots & \cdots & \cdots \\ \rhd & \cdots & \rhd \\ [A_{n,1}] & \cdots & [A_{n,m}] \end{array}$$

The next theorem states that an ETA with a memoryless location l can be *flattened* into one that contains fewer cycles through l, while preserving $\equiv_r$.

Theorem 4.8 (Flattening). *Consider an ETA $A^* = (L, \Sigma, C, V, l_0, l_F, Inv, E^*)$ with a memoryless location $l \in L$. Then $A_l = (L, \Sigma, C, V, l_0, l_F, Inv, E_l)$ satisfies $A^* \equiv_r A_l$, where*

$$E_l = E^* \setminus \{\ e\ \mid \mathit{target}(e) = l \text{ and no cycle-free syntactic path from } l_0 \text{ to } l \text{ in } A^* \text{ contains } e\},$$

Proof. By construction, A_l keeps all edges that are needed to reach l from l_0 along a cycle-free syntactic path. Consider a path π of A^*, say of the form $\pi = \{l_0\}\pi_1\{l\}\cdots\pi_n\{l\}\pi_{fin}$, where the assertion $\{l_0\}$ (resp. $\{l\}$) indicates that the control resides in the location l_0 (resp. l). As l is first reached along π_1, each π_i with $i = 2, \ldots, n$ represents a subsequent cycle of A^* through l, and π_{fin} represents the (possibly empty) final part of π that does not reach l again. Since l is memoryless, every state that is reachable along the path π in A^* is also reachable in A_l along the following *set* of paths: $\pi_{1,2} = \{l_0\}\pi_1\{l\}\pi'_2$, …, $\pi_{1,n} = \{l_0\}\pi_1\{l\}\pi'_n$, $\pi_{1,fin} = \{l_0\}\pi_1\{l\}\pi_{fin}$, where π'_i is the path π_i without the last transition reentering the location l, for $i = 2, \ldots, n$. Thus $A^* \equiv_r A_l$ with identity as the location relation. □

If $E_l \subset E^*$ then A_l is a *flattened* version of A^* with a reduced number of cycles through l. Note that flattening at l_0 will remove every edge e with $\mathit{target}(e) = l_0$ because no edge is needed to reach the initial location l_0. So A_{l_0} is cycle-free at l_0. Next, we consider flattening of A^* in the context of a parallel composition $A^*\|B$ and state sufficient conditions for the preservation of *location reachability*.

Theorem 4.9 (Flattening in parallel context). *Suppose that for A^* and B, where A^* is memoryless at $l_a \in L_A$, the following holds within $A^*\|B$:*

1. *Every location of A^* is reachable from its initial location l_{0A} without visiting l_a more than once, while B stays in its initial location l_{0B}.*
2. *Every location of B is reachable from l_{0B}, while A^* stays in l_a.*
3. *If a transition entering l_a enables a transition of B with target l_b then every location of A^* is reachable from l_a without visiting l_a again, while B is in l_b.*

Then $Reachloc(A^*\|B) = Reachloc(A_{l_a}\|B)$.

Proof. Clearly, $Reachloc(A_{l_a}\|B) \subseteq Reachloc(A^*\|B)$ as all edges of A_{la} are present in A^*. Thus it suffices to show the reverse inclusion. We prove by induction on the transition steps that for every location pair: $(l_1, l_2) \in Reachloc(A^*\|B)$ implies $(l_1, l_2) \in Reachloc(A_{l_a}\|B)$.

Induction Basis. Clearly, the claim holds for the pair (l_{0A}, l_{0B}) of initial locations.

Induction Hypothesis. Suppose the claim holds for (l_1, l_2).

Induction Step. For location reachability, it suffices to consider the interleaving of A^*- and B-transitions.

Case 1. Consider an A^*-transition t_a leading from l_1 to l_1'.
The only interesting subcase is that t_a has been removed in A_{l_a}. By construction of A_{l_a}, we have $l_1' = l_a$, and there exists on the path from l_{0A} to l_1 in A^* a first occurrence of l_a that is reached also in A_{l_a}. Then $(l_1', l_2) = (l_a, l_2)$ is reachable in A_{l_a} as follows. First, proceed from (l_{0A}, l_{0B}) to (l_a, l_{0B}) by taking transitions in A_{l_a} only. This is possible due to Condition 1. Second, proceed from (l_a, l_{0B}) to (l_a, l_2) by taking transitions in B only. This is possible due to Condition 2.

Case 2. Consider an B-transition t_b leading from l_2 to l_2'.
The only interesting subcase is that a transition t_a entering l_a enables t_b but t_a has been removed in A_{l_a}. Then (l_1, l_2') is reachable in A_{l_a} as follows. First, proceed from (l_{0A}, l_{0B}) to (l_a, l_{0B}) by taking transitions in A_{l_a} only. This is possible due to Condition 1. Next, proceed from (l_a, l_{0B}) to (l_a, l_2') by taking transitions in B only. This is possible due to Condition 2. Then proceed from (l_a, l_2') to (l_1, l_2') by taking transitions in A_{l_a} only. This is possible due to Condition 3. □

Remark 4.7. In contrast to all the other transformations, flattening in a parallel context does not easily generalize to multiple parallel instances, and the three itemized conditions of Theorem 4.9 above require an exploration of the reachable state space. Such an exploration however does not entail a complete resolution of the $\|$ operator in $A^*\|B$, as each of the above conditions reduces to a *local reachability check of A^* resp. B*, with control residing at a fixed location of B resp. A^*. For the case where B is itself composed of multiple parallel ETA, a limited resolution of $\|$ within B may be necessary in order to verify the second reachability condition of Theorem 4.9. The above reachability checks may nonetheless be *localized within a layer* if the flattening is performed subsequent to separation and layering, as will be shown in the next section.

4.6 Example: Real-Time Mutual Exclusion

Consider two processes A and B competing for two critical sections $cs1$ and $cs2$. We safeguard these sections by a *double Fischer protocol DF* obtained by taking for

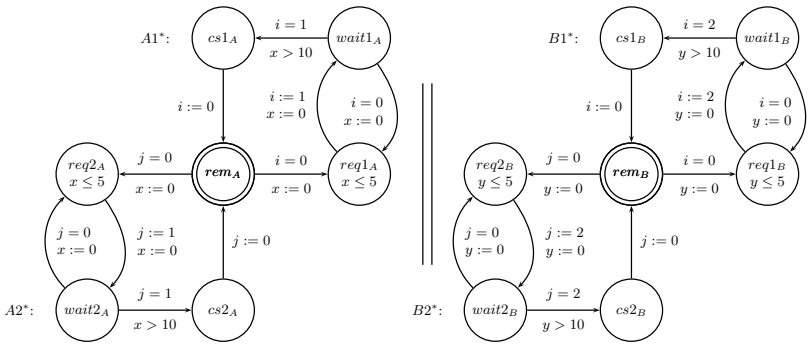

Fig. 4.6. Double Fischer protocol $DF = (A1^* \cup [A2^*]) \parallel (B1^* \cup [B2^*])$ for processes A and B accessing two critical sections $cs1$ and $cs2$. *Left*: ETA $A1^* \cup [A2^*]$; *right*: ETA $B1^* \cup [B2^*]$. In this and all subsequent figures, we omit the ε-labels of all edges.

each process the union of two copies of Fischer's real-time protocol. We represent DF by the following composition of ETA:

$$DF = (A1^* \cup [A2^*]) \parallel (B1^* \cup [B2^*]),$$

where $A1^*, A2^*$ are the two copies of Fischer's protocol used by process A, and $B1^*, B2^*$ the two copies used by process B, cf. Fig. 4.6. The stars * indicate the presence of cycles. In locations $cs1_A$ and $cs2_A$ process A accesses the critical sections $cs1$ and $cs2$, respectively, and in $cs1_B$ and $cs2_B$ process B does so. Initially, A and B need not access the critical sections and are busy with *remaining* activities in the initial locations rem_A and rem_B, whose conditions permit the wrapping of all ETA, and in particular $A2^*$ and $B2^*$. In $req1_A, req2_A$ and $req1_B, req2_B$ the processes A and B *request* access to $cs1, cs2$. The locations $wait1_A, wait2_A$ and $wait1_B, wait2_B$ represent *waiting* of A and B for $cs1, cs2$. The parallel ETA in DF use disjoint (but synchronous) clocks x and y, while sharing the data variables i and j that range over $0, 1, 2$ and are initialized with 0. These values indicate whether (0) none of the processes, (1) process A, or (2) process B wants to access $cs1$ or $cs2$, respectively. In these ETA, all edges are labelled by ε-actions. Synchronization between the ETA takes place via guards checking the values of the shared variables i and j.

We wish to prove that DF satisfies the *double mutual exclusion* property

$$DMX = \Box\neg(cs1_A \wedge cs1_B) \wedge \Box\neg(cs2_A \wedge cs2_B).$$

Thus, DMX is a conjunction of two *layered reachability* properties $MX_1 = \Box\neg(cs1_A \wedge cs1_B)$ and $MX_2 = \Box\neg(cs1_A \wedge cs1_B)$ for the ETA composition DF, whose first layer consists of $A1^*$ and $B1^*$, and whose second layer consists of $A2^*$ and $B2^*$.

We simplify the verification task now by a series of structural transformations so that it finally becomes almost trivial.

1. Separation.

We apply the separation transformation to DF and obtain two single versions of Fischer's protocol separated by an extra transition, cf. Fig. 4.7. This is possible due to $A1^* \not\frown B2^*$ and $B1^* \not\frown A2^*$. The result is

$$SDF = (A1^* \rhd [A2^*]) \parallel (B1^* \rhd [B2^*]).$$

By the Separation Theorem 4.7, we have $DF \equiv_r SDF$.

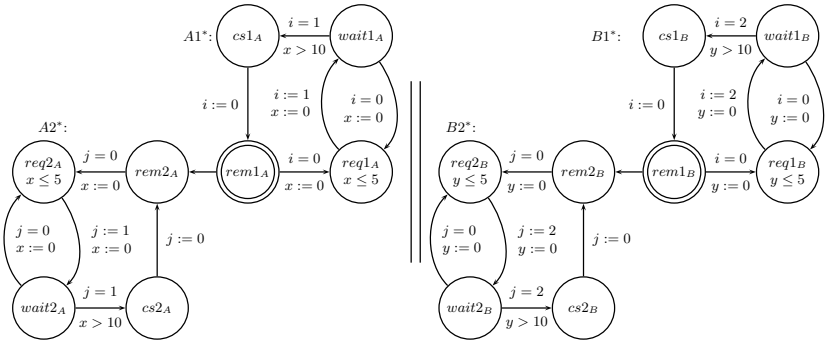

Fig. 4.7. Separated version of double Fischer: $SDF = (A1^* \rhd [A2^*]) \parallel (B1^* \rhd [B2^*])$.

2. Layering.

To apply layering to SDF, we calculate:

$$\begin{aligned} SDF = \quad & (A1^* \rhd [A2^*]) \parallel (B1^* \rhd [B2^*]) \\ \equiv_{\mathcal{L}} \quad & \{ \text{ Theorem 4.3, using } A1^* \not\frown B2^* \text{ and } B1^* \not\frown A2^* \} \\ & (A1^* \parallel B1^*) \rhd ([A2^*] \parallel [B2^*]) \quad = \quad LDF \end{aligned}$$

LDF stands for layered double Fischer. The component ETA of LDF are obtained from the ETA shown in Fig. 4.7 by cutting these at $rem2_A$ and $rem2_B$. The result is shown in Fig. 4.8.

To prove that DF satisfies DMX, it suffices to do this for LDF. Since step composition is the top operator in LDF, it suffices to show that both step components, $L_1 = A1^* \parallel B1^*$ and $L_2 = [A2^*] \parallel [B2^*]$, individually satisfy DMX. As $DMX = MX_1 \wedge MX_2$, where MX_1 is confined to L_1 and MX_2 is confined to L_2 (cf. the formulation of DMX earlier in this section), this further reduces to showing separately that L_1 satisfies MX_1 and that L_2 satisfies MX_2.

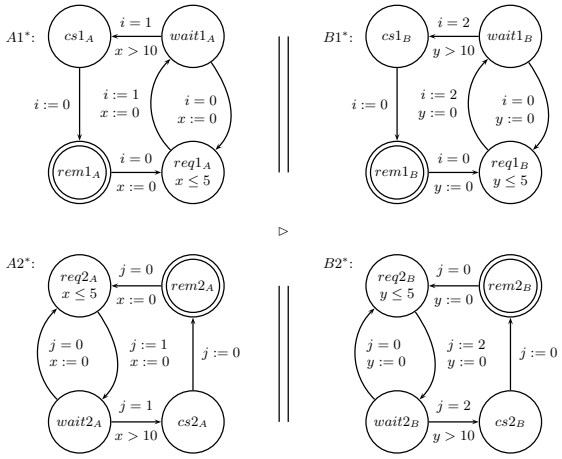

Fig. 4.8. Layered version of double Fischer: $LDF = (A1^* \parallel B1^*) \rhd ([A2^*] \parallel [B2^*])$.

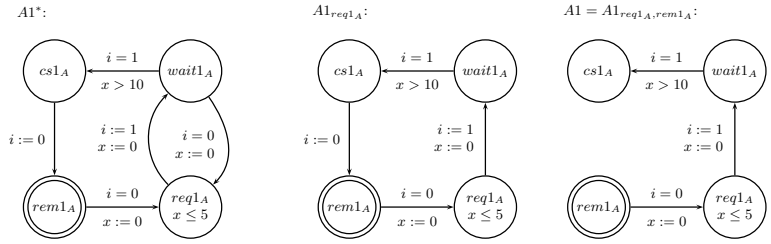

Fig. 4.9. Flattening $A1^*$ at location $req1_A$ yields $A1_{req1_A}$ and flattening this ETA at location $rem1_A$ yields the cycle-free ETA $A1 = A1_{req1_A, rem1_A}$.

3. Flattening.

We remove all cycles in $A1^*, B1^*, A2^*, B2^*$ and show this in detail for $A1^*$ in Fig. 4.9. Flattening $A1^*$ at $req1_A$ is possible, since whenever $req1_A$ is entered, $i = 0$ and $x = 0$ holds. Flattening the resulting $A1_{req1_A}$ at $rem1_A$ does not seem possible at first sight because only the data variable i is reset to 0. However, we may safely add $x := 0$ because this clock reset occurs when $rem1_A$ is left, and because x occurs neither in the invariant of $rem1_A$ nor in the guard of its outgoing transition. Note also that the additional three conditions of Theorem 4.9 hold for $A1^*$ at the locations $req1_A$ and $rem1_A$. Hence, we arrive at $A1 = A1_{req1_A, rem1_A}$ without any cycles. We may similarly flatten $B1^*$ at $req1_B$ and $rem1_B$, yielding a corresponding cycle-free ETA $B1$. It thus remains to show that two cycle-free versions of Fischer's protocol, $A1 \parallel B1$ and $A2 \parallel B2$, satisfy MX_1 and MX_2 respectively, where we have,

for $i \in \{1,2\}$, $Reachloc(Ai \| Bi) = Reachloc(Ai^* \| Bi^*)$ by Theorem 4.9, which is sufficient for preserving MX_1 and MX_2.

4. Precedence layering.

We prove that $A1 \| B1$ satisfies MX_1. Consider the ETA A_{01}, A_{11}, A_{21}, B_{01}, B_{11}, and B_{21} shown in Fig. 4.10. As before, x and y are clocks, and i is a shared data variable ranging over 0, 1, 2, initialized with 0. The ETA A_{01}, A_{11}, A_{21} represent three phases of $A1$ and B_{01}, B_{11}, B_{21} those of $B1$, such that $A1 \| B1 = (A_{01}; A_{11}; A_{21}) \,\|\, (B_{01}; B_{11}; B_{21})$.

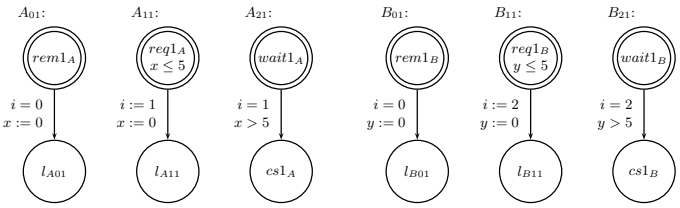

Fig. 4.10. Six ETA for building Fischer's protocol for single mutual exclusion of two processes: $A1 = A_{01}; A_{11}; A_{21}$ and $B1 = B_{01}; B_{11}; B_{21}$.

We explore the interleavings of $A_{01}; A_{11}; A_{21}$ with $B_{01}; B_{11}; B_{21}$ by (partially) *expanding* (as in CCS, cf. [Mil89], and as in [BS00]) the parallel composition in $A1 \| B1$. After A_{11} neither B_{01} nor B_{21} can occur due to the i-guard, and vice versa, after B_{11} neither A_{01} nor A_{21} can occur. After $A_{01} \| B_{01}$ we observe that timewise (by the synchronous evolution of the clocks x and y) B_{21} cannot proceed from $wait1_B$ to $cs1_B$ (due to the clock guard $y > 5$) before A_{11} has left $req1_A$ (with clock invariant $x \leq 5$) to reach its final location l_{A11}, and vice versa, A_{21} cannot proceed to $cs1_A$ before B_{11} reaches its final location l_{B11}. Thus A_{11} resp. B_{11} precedes B_{21} resp. A_{21} in any parallel context. In particular, the precedences $A_{11} \prec_{A_{21}, B_{11}} B_{21}$ and $B_{11} \prec_{A_{11}, B_{21}} A_{21}$ hold. Thus expansion and the law (PrecCCL-seq) of Theorem 4.5 yield

$$\begin{aligned} A1 \| B1 &\equiv_r (A_{01}; A_{11}; A_{21}) + (B_{01}; B_{11}; B_{21}) + (A_{01} \| B_{01}); ((A_{11}; A_{21}) \| (B_{11}; B_{21})) \\ &\equiv (A_{01}; A_{11}; A_{21}) + (B_{01}; B_{11}; B_{21}) + (A_{01} \| B_{01}); (A_{11} \| B_{11}); (A_{21} \| B_{21}) = SF_1, \end{aligned}$$

where $+$ denotes a non-deterministic choice operator on ETA (cf. Definition 4.17), and SF_1 stands for sequential Fischer. Clearly, SF_1 satisfies MX_1 because after $A_{11} \| B_{11}$ the data variable i stores either 1 or 2, and thus either A_{21} or B_{21} (but not both) can proceed to their critical section. Since each of the equivalences induced by our transformations (namely, $\equiv$, $\equiv_r$, $\equiv_{\mathcal{L}}$, and equality w.r.t *Reachloc*) is clearly sufficient for MX_1, we then conclude that MX_1 holds for SF_1 (and consequently for L_1). One may easily conclude by symmetry that MX_2 holds for L_2. Thus, altogether, DMX holds for DF as required.

Remark 4.8. While the example of this section considers two parallel timed processes competing for access to two critical sections, it may be generalized to multiple critical sections and multiple parallel instances. The corresponding complex protocol however admits analysis via generalizations of our transformations to multiple step and parallel instances (cf. Remarks 4.2, 4.3, 4.4, 4.5, 4.6 and 4.7).

4.7 Conclusion

We conclude this chapter with a discussion on additional references to the literature. Early work on distributed systems considered "regularity" conditions [Boc79, Boc88], under which the system behaviour is independent of communication delays, thus simplifying its (semi-formal) design and analysis. The regularity conditions therein (specified in a discrete-time setting) roughly correspond to our notion of wrapping, where timing influences from other parallel components are likewise decoupled.

A constraint-based decompositional proof methodology was illustrated in [LSW96] on the standard Fischer's protocol, formalized as a *timed modal specification*. More recently, an analysis of TA networks with "disjoint phases of activity" has been carried out in [MWP12], where it has been shown that the parallel composition of two TA (without shared data variables) is bisimilar to their sequential composition, if the TA exhibit certain *periodic but non-overlapping* behaviours. A transformation approach (roughly corresponding to a non-local version of our flattening, complemented by a notion of edge-redirection) for the simplified assume-guarantee verification of ETA networks, with application to Fischer's mutual exclusion protocol, has been presented in [MWF14].

In [CJ99] it was shown that any TA (possibly containing nested cycles, but again without shared data variables) may be transformed into one that is flat (in the sense that each location is part of at most one cycle), while preserving the reachability relation between states. Their (non-local) transformation, while applicable to all TA, is however not preserved in the context of parallel composition, and suffers from an exponential blow-up in the number of locations in the flattened TA, cf. Lemma 3 of [CJ99]. Our (local) separation and flattening transformations, on the other hand, are applicable (in the context of parallel composition) to the data-enriched setting of ETA networks, and maintain the same number of locations, while reducing the nesting depth and deleting those transitions that (re-)enter memoryless locations, cf. Theorems 4.7 and 4.9.

A layered transformation for distributed algorithms with (predominantly synchronous) *message passing* was presented in [SdR94]. *Round-based communication closedness* was considered in [CSCBM09] for fault-tolerant distributed algorithms with *asynchronous message passing*, with messages being considered only in the rounds during which they were sent. Consensus algorithms in such a setting were then brought under the scope of automatic verification, by means of "reduction theorems", cf. [CSCBM09].

Our example of real-time mutual exclusion is small but instructive; it served to illustrate the interplay of the structural transformations of separation, layering, and flattening.

For future work, it would be interesting to investigate whether the Gear Production Stack case-study in [PM09] admits state space reduction by our flattening transformation. Another possible direction could be the development of a *structured extension* to the Slicing Abstractions (SLAB) model-checker [DKFW10], which would perform a pre-processing of the model according to our transformation rules, thus simplifying the model's subsequent verification.

In the next chapter, we present a layered transformation for randomized distributed algorithms (modelled as compositions of probabilistic automata), with application to simplyfing a randomized distributed algorithm for mutual exclusion.

5

Layered Transformations for Networks of Probabilistic Automata

5.1 Introduction

Probabilistic automata [SL95, Seg00] (PA) constitute an operational framework for the modelling and analysis of discrete systems that exhibit both nondeterministic and randomized behaviour, such as randomized distributed algorithms. An I/O-variant of PA (PIOA) has been used to successfully analyze intricate randomized distributed algorithms such as the Aspnes-Herlihy randomized consensus algorithm [PSL00] and the IEEE Firewire protocol [SV99]. Extensions of PIOA have been used for specifying and verifying security protocols [CCK$^+$08]. In the context of concurrency theory, PA are used as semantic models for stochastic process algebras, and have been equipped with (bi)simulation notions [SL95].

Despite the presence of modular verification techniques for PA [Seg00], the correctness proofs of randomized distributed algorithms remain difficult and require substantial human ingenuity. This chapter attempts to simplify their reasoning by enriching probabilistic automata with the concept of *layering*. The main underlying idea is that the computations of randomized distributed algorithms often exhibit a *sequential* (i.e., layered) structure. The idea of using such sequential structure to simplify the verification of distributed algorithms was originally proposed in eighties by Elrad and Francez [EF82], and has been extended, formalized [SdR94], and applied to intricate distributed algorithms such as the minimal spanning tree algorithm [JZ92] about a decade later. Layered reasoning (though in a different way as in this chapter) has been recently used to obtain tighter bounds for asynchronous randomized consensus [AC08]; earlier work on applying layering to bound analysis appeared in [MR02]. We study in this chapter layered reasoning for randomized distributed algorithms, with PA (enriched with shared data variables) as the underlying operational model.

For simplifying the formal reasoning of randomized distributed algorithms, we introduce layered composition $\mathcal{P} \bullet \mathcal{Q}$ of PA $\mathcal{P}$ and $\mathcal{Q}$; the PA $\mathcal{P} \bullet \mathcal{Q}$ behaves like $\mathcal{P}||\mathcal{Q}$, the parallel composition of $\mathcal{P}$ and $\mathcal{Q}$, except that for all actions a of $\mathcal{P}$ and b of $\mathcal{Q}$ that depend on each other (e.g., as both actions affect the same shared variable), a is executed before b. Layered composition is thus a kind of asymmetric parallel composition. We obtain a probabilistic version of the *communication closed layer* (CCL) law that allows us to identify $(\mathcal{P}_1 \bullet \mathcal{P}_2)||(\mathcal{Q}_1 \bullet \mathcal{Q}_2)$ and $(\mathcal{P}_1||\mathcal{Q}_1) \bullet (\mathcal{P}_2||\mathcal{Q}_2)$, provided

$\mathcal{P}_1, \mathcal{Q}_2$ and $\mathcal{P}_2, \mathcal{Q}_1$ respect pair-wise certain *independence* or *precedence* conditions. This CCL law enables us to transform a randomized distributed algorithm into an equivalent layered one so as to permit easier verification. This verification is technically enabled using a partial-order (po) equivalence on probabilistic automata;

The CCL law together with the po-equivalence of $\bullet$ and ; allows the transformation of $(\mathcal{P}_1; \mathcal{P}_2)||(\mathcal{Q}_1; \mathcal{Q}_2)$ –via intermediate representations $(\mathcal{P}_1 \bullet \mathcal{P}_2)||(\mathcal{Q}_1 \bullet \mathcal{Q}_2)$ and $(\mathcal{P}_1||\mathcal{Q}_1) \bullet (\mathcal{P}_2||\mathcal{Q}_2)$– finally to $(\mathcal{P}_1\|\mathcal{P}_2); (\mathcal{Q}_1\|\mathcal{Q}_2)$ under the aforementioned independence or precedence conditions. This should yield a *syntactic (partial order) state-space reduction* that may be applied prior to automated model checking of randomized distributed algorithms. However, Luis Maria Ferrer Fioriti of Saarland University observed that the unrestricted (adversarial) resolution of non-determinism could exploit the outcome of a preceding probabilistic choice, leading to a violation of the po-equivalence between $(\mathcal{P}_1; \mathcal{P}_2)||(\mathcal{Q}_1; \mathcal{Q}_2)$ and $(\mathcal{P}_1\|\mathcal{P}_2); (\mathcal{Q}_1\|\mathcal{Q}_2)$ in the general case. We conjecture that an *oblivious adversarial resolution* of the non-determinism (that is not allowed to exploit the outcomes of prior probabilistic choices) might resolve this problem. Moreover, the necessary termination conditions on $\mathcal{P}_1$ and $\mathcal{Q}_1$ for obtaining po-equivalence remain open.

Notwithstanding these technical issues, we illustrate the feasibility of probabilistic layering on the randomized mutual exclusion algorithm of Kushilevitz and Rabin [KR92], which, as shown in [MGCM08], has oblivious adversarial resolution built into its specification. This algorithm improves an earlier version proposed by Rabin [Rab82], whose correctness proof was demonstrated as being flawed in [Sai92], due to the comparison of two probabilities that were not defined within the same probability space [Seg00].

This chapter is a revised version of [5] with the following structure: Section 5.2 recalls the PA framework and the notion of trace distributions. It also defines sequential (denoted by ;) and parallel composition (denoted by ||). Section 5.3 defines the notion of layered composition (denoted by $\bullet$), precedence relations for PA, and formulates the probabilistic CCL laws. Section 5.4 defines po-equivalence $\equiv^*_{po}$ when $\bullet$ is replaced by ; within the CCL laws. Section 5.5 shows the applicability of our approach to the randomized mutual exclusion algorithm of [KR92]. Section 5.6 concludes the chapter with additional perspectives on our results in relation to the analysis in [MGCM08].

5.2 Probabilistic Automata

In this section, we introduce *Probabilistic Automata* (PA) [Seg00, Sto02] enriched with *shared data variables* as operational models for randomized distributed algorithms. Such PA exhibit both *probabilistic* and *non-deterministic* behaviour. The non-deterministic behaviours of PA are resolved using *adversaries*. *Trace distributions* characterize the *probability spaces* associated to such adversaries. The behaviours of PA are compared in terms of their trace distributions. To enable *modular reasoning* of PA, we also introduce the concepts of *sequential* and *parallel composition*. The *non-deterministic selection* of a component during distributed execution is modelled by a corresponding operator on PA.

Actions, data variables, and probability distributions.

Let Σ be a finite *alphabet* of (communication) *channels*. A typical element of Σ is denoted $\alpha, \beta, \cdots$. There are two *actions* for each channel $\alpha \in \Sigma$: $\alpha?$ denotes an *input* on α, while $\alpha!$ denotes the corresponding *output* on α, where $\alpha?, \alpha! \notin \Sigma$. We denote by τ an action resulting from *synchronization*, with $\tau \notin \Sigma$. By $\Sigma_{?!} = \{\alpha? \mid \alpha \in \Sigma\} \cup \{\alpha! \mid \alpha \in \Sigma\} \cup \{\tau\}$, we denote the set of all actions over the alphabet Σ. A typical element of $\Sigma_{?!}$ is denoted $a, b, \cdots$. In the context of parallel composition, input and output are *complementary* actions that can synchronize yielding τ. For an action $a \in \Sigma_{?!} \setminus \{\tau\}$, its complementary action is denoted by $\overline{a}$, i.e., $\overline{\alpha?} = \alpha!$ and vice-versa. We stipulate that τ-actions may result only from the synchronization of complementary input and output actions.

Let V be a finite set of *data variables* (ranged over by v) that take values in some finite range D. By $\Psi(V)$ we denote the set of *data updates* with typical element ψ. The left-hand side of an update is a variable in V whereas its right-hand side is an expression involving the variables of V and the usual arithmetic operators $+, -, \cdots$. For each $\psi \in \Psi(V)$ each variable occurs at most once on the left-hand side of an update. The set $\Phi(V)$ of *data constraints* over V with typical element ϕ is the set of Boolean constraints over variables in V involving the usual arithmetic $(+, -, \cdots)$ and relational $(<, \leq, >, \geq)$ operators. A *data valuation* assigns a value in D to each data variable in V. If $|V| = m$, a data valuation is identified with a point in D^m, denoted typically by $\mathbf{u}, \mathbf{v}$ etc. Applying an update $\psi \in \Psi(V)$ on data valuation $\mathbf{u}$ yields the data valuation $\psi(\mathbf{u})$. For valuation $\mathbf{u}$ and boolean constraint ϕ, let $\mathbf{u} \models \phi$ denote that the valuation $\mathbf{u}$ satisfies the data constraint ϕ.

Let S be a countable set. The function $\mu : S \to [0, 1]$ is a *distribution* on S if $\sum_{s \in S} \mu(s) = 1$. Let $Dist(S)$ denote the set of distributions on S and $supp(\mu) = \{s \in S \mid \mu(s) > 0\}$ be the *support* of μ.

Definition 5.1 (Probabilistic automata (PA)). *PA are tuples* $\mathcal{P} = \langle L, l_0, l_f, V, \Sigma, prob \rangle$, *where:*

- L *is a finite, non-empty set of* locations*, ranged over by* l, l', $\cdots$.
- $l_0 \in L$ *is the* start location.
- $l_f \in L$ *is the* final location *with* $l_0 \neq l_f$.
- V *is a finite set of* data variables *taking on values in a finite range* D.
- Σ *is a finite* alphabet *of* channels.
- $prob \subseteq L \setminus \{l_f\} \times \Sigma_{?!} \times \Phi(V) \times Dist(\Psi(V) \times L)$ *is a* probabilistic transition relation.

In the above definition, *prob* relates a location $l \neq l_f$, action a and guard $\phi \in \Phi(V)$ to a *distribution* over $\Psi(V) \times L$, i.e., an update and a target location. The intuitive operational meaning of $(l, a, \phi, \mu) \in prob$ is as follows. Given the current location l, action a, and a data valuation in l satisfying the guard ϕ, with probability $\mu(\psi, l')$ a transition to location l' is made while updating the data variables according to the update ψ. In case $(l, a, \phi, \mu) \in prob$ and $(l, a, \phi, \nu) \in prob$, on action a and satisfaction of guard ϕ a non-deterministic choice between distributions μ and ν is made. Tuples $(l, a, \phi, \mu) \in prob$ are called *edges* of the PA. Let $edges(l, a, \phi)$ denote the set of edges $\{(l, a, \phi, \mu) \in prob\}$. Note that by the above definition, $edges(l_f, a, \phi) = \emptyset$ for all $a \in \Sigma_{?!}$ and all $\phi \in \Phi(V)$.

Remark 5.1. According to the above definition, *multiple* distributions can be associated with a given location l, action a, and guard ϕ, as *prob* is a relation. This enables a conceptually cleaner notion of non-deterministic choice between probabilistic automata, as will be defined later (cf. Definition 5.13).

The semantics of a PA is given in terms of a *probabilistic transition system* (PTS). A PTS is basically a labelled transition system where the target of labelled transitions are distributions over states, rather than just simply states. A *state* of a PA with $|V| = m$ is a pair $(l, \mathbf{u}) \in L \times D^m$, denoted typically by s, consisting of a location l and a data valuation $\mathbf{u}$.

Definition 5.2 (Induced probabilistic transition system). *The PA* $\mathcal{P} = \langle L, l_0, l_f, V, \Sigma, prob \rangle$ *with*
$|V| = m$ *induces the PTS* $[\mathcal{P}] = \langle S, s_0, \Sigma, \Delta \rangle$*, where*

- $S = L \times D^m$ *is the state space.*
- $s_0 = (l_0, \mathbf{v_0})$*, where* $\mathbf{v_0} \in D^m$ *denotes a designated initial data valuation.*
- $\Delta \subseteq S \times \Sigma_{?!} \times Dist(S)$ *is the transition relation defined by:* $((l, \mathbf{u}), a, \nu) \in \Delta$ *iff* $(l, a, \phi, \mu) \in prob$ *and* $\mathbf{u} \models \phi$*, with*
  $\nu((l', \mathbf{v})) = \sum\{\!|\, \mu(\psi, l') \mid \mathbf{v} = \psi(\mathbf{u}) \,|\!\}$*, where* $\{\!|\ldots|\!\}$ *denotes a multi-set.*

The transition relation Δ thus contains triples (s, a, ν) with $s = (l, \mathbf{u})$ whenever there is an edge (l, a, ϕ, μ) in the PA such that the valuation $\mathbf{u}$ satisfies the guard ϕ. The probability to move to the state $s' = (l', \mathbf{v})$ is the cumulative probability $\mu(\psi, l')$ where the valuation $\mathbf{v}$ is obtained from $\mathbf{u}$ by an update according to ψ. The multi-set is needed as there may be several branches of the edge (l, a, ϕ, μ) that lead with the same likelihood to the next state $s' = (l', \mathbf{v})$. A transition $(s, a, \mu) \in \Delta$ is denoted $s \xrightarrow{a} \mu$.

Remark 5.2. Note that the induced PTS considered in this chapter are all *finite state*, owing to the range D of the data variables being finite. The notation $[\mathcal{P}]$ in this chapter indicates the induced probabilistic transition system of the PA $\mathcal{P}$, and thus contrasts with a similar notation for wrapping in the previous chapter, cf. Definition 4.10. Note however that wrapping does not apply here as the PA framework is in the untimed setting.

Paths and traces.

The PA model thus incorporates both *non-determinism* and *probabilistic choice*, and a possible behaviour reflected in the corresponding PTS results from the resolution of non-deterministic and probabilistic choices, described in terms of paths. A *path* π of a PTS $[\mathcal{P}] = \langle S, s_0, \Sigma, \Delta \rangle$ is a (possibly infinite) sequence of the form $\pi = s_0 a_1 \mu_1 s_1 a_2 \mu_2 s_2 a_3 \mu_3 s_3 \ldots$ where $\forall n : s_n \xrightarrow{a_{n+1}} \mu_{n+1}$, and $\mu_{n+1}(s_{n+1}) > 0$. Let $last(\pi)$ denote the last state of π (if π is finite), $\pi(n)$ the n^{th} state of π, and $|\pi|$ the length (i.e., number of actions) of π. For a given PTS $[\mathcal{P}]$, let $Path^*([\mathcal{P}])$ be the set of all finite paths of $[\mathcal{P}]$, and $Path^\omega([\mathcal{P}])$ the set of all (possibly infinite) paths of $[\mathcal{P}]$. Also, we denote by $Path^n([\mathcal{P}])$ the set of paths $[\mathcal{P}]$ of length upto n. For a given path $\pi = s_0 a_1 \mu_1 s_1 a_2 \mu_2 s_2 a_3 \mu_3 s_3 \ldots$, the n^{th} transition is $s_{n-1} \xrightarrow{a_n} \mu_n$, and its *trace* is given by $trace(\pi) = a_1 a_2 a_3 \ldots$, obtained by omitting all states and distributions from π.

Definition 5.3 (Adversary). *For a given PTS* $[\mathcal{P}] = \langle S, s_0, \Sigma, \Delta \rangle$*, an* adversary $\mathcal{A}$ *of* $[\mathcal{P}]$ *is a function* $\mathcal{A} : Path^*([\mathcal{P}]) \mapsto (\Sigma_{?!} \times Dist(S)) \cup \{\bot\}$ *that maps every finite path* π *of* $[\mathcal{P}]$ *to a pair* (a, μ) *or to* $\bot$*, such that if* $\mathcal{A}(\pi) = (a, \mu)$ *for some* $a \in \Sigma_{?!}$ *and* $\mu \in Dist(S)$*, then* $(last(\pi), a, \mu) \in \Delta$ *and if there is no* $a \in \Sigma_{?!}$ *and* $\mu \in Dist(S)$ *such that* $(last(\pi), a, \mu) \in \Delta$ *then* $\mathcal{A}(\pi) = \bot$*, where* $\bot$ *denotes a special action for termination.*

Thus, an adversary resolves all non-deterministic choices of the PTS $[\mathcal{P}]$, so that under a given adversary $\mathcal{A}$ of $[\mathcal{P}]$, the behaviour of $\mathcal{P}$ is *purely probabilistic*. In this chapter, we restrict the class of adversaries to be *admissible* and *memoryless* [Seg95]. An adversary $\mathcal{A}$ of a PTS $[\mathcal{P}]$ is said to be *memoryless* if for any two finite paths π and π' of $[\mathcal{P}]$, $last(\pi) = last(\pi') \Rightarrow \mathcal{A}(\pi) = \mathcal{A}(\pi')$. An *admissible* adversary is one that schedules bismilar transitions for any two finite paths having identical traces and ending in bisimilar states [Seg95]. We denote by $Adv([\mathcal{P}])$ the set of all *memoryless admissible adversaries* of the PTS $[\mathcal{P}]$.

A path $\pi = s_0 a_1 \mu_1 s_1 a_2 \mu_2 s_2 \ldots \in Path^\omega([\mathcal{P}])$ under an adversary $\mathcal{A}$ is such that $\mathcal{A}(pref_n(\pi)) = (a_{n+1}, \mu_{n+1})$ and $\mu_{n+1}(s_{n+1}) > 0$ for all $0 \leq n \leq |\pi|$. Here, $pref_n(\pi)$ denotes the prefix of π of length n. We denote by $Path^*_{\mathcal{A}}([\mathcal{P}])$ (resp. $Path^\omega_{\mathcal{A}}([\mathcal{P}])$) the set of all finite (resp. possibly infinite) paths of the PTS $[\mathcal{P}]$ under a given adversary $\mathcal{A}$. Similarly, $Path^n_{\mathcal{A}}([\mathcal{P}])$ denotes the set of paths upto length n of $[\mathcal{P}]$ under $\mathcal{A}$. Note that states having the final location l_f of the PA $\mathcal{P}$ as their location component do not admit further actions, so that $\forall \mathcal{A} \in Adv([\mathcal{P}]), \forall \pi \in Path^*_{\mathcal{A}}([\mathcal{P}]) : last(\pi) = (l_f, \mathbf{u})$ we have that $\mathcal{A}(\pi) = \bot$.

A path π is said to be *maximal* under a given adversary $\mathcal{A}$ if either π is an infinite path in $Path^\omega_{\mathcal{A}}([\mathcal{P}])$, or if π is a finite path in $Path^*_{\mathcal{A}}([\mathcal{P}])$ and $\mathcal{A}(\pi) = \bot$. That is to say, a maximal path under adversary $\mathcal{A}$ is a path that cannot be prolonged under $\mathcal{A}$. We denote by $Path^{max}_{\mathcal{A}}([\mathcal{P}])$ the set of all maximal paths in $[\mathcal{P}]$ under the adversary $\mathcal{A}$.

We now formulate a notion of *equivalence on adversaries* for comparing the behaviours of PA that are defined over the same alphabet, the same state space, and the same set of distributions. Such PA arise when considering the probabilistic communication closed equivalences that will be detailed later.

Definition 5.4 (Equivalence $\equiv$ on adversaries). *Given PA* $\mathcal{P}_1$ *and* $\mathcal{P}_2$ *defined over the same alphabet* Σ*, the same state space* S*, and the same set of distributions* $Dist(S)$*, with* $\mathcal{A}_i \in Adv([\mathcal{P}_i])$*. Then* $\mathcal{A}_1 \equiv \mathcal{A}_2$ *iff* $\forall \pi_i \in Path^*_{\mathcal{A}_i}([\mathcal{P}_i])\ \exists \pi_{3-i} \in Path^*_{\mathcal{A}_{3-i}}([\mathcal{P}_{3-i}]) : |\pi_i| = |\pi_{3-i}| \wedge last(\pi_i) = last(\pi_{3-i}) \wedge \mathcal{A}_{3-i}(\pi_{3-i}) = \mathcal{A}_i(\pi_i)$ *where* $i \in \{1, 2\}$.

Thus, equivalent adversaries induce the same possible non-deterministic choices for all same-length finite paths with identical last-states. The *probability* that a given resolution of non-determinism —as specified by an adversary $\mathcal{A}$ of $[\mathcal{P}]$— results in a path $\pi \in Path^*([\mathcal{P}])$ is given by a function $\mathbf{Q}_{\mathcal{A}} : Path^*([\mathcal{P}]) \mapsto [0, 1]$. It is defined inductively as follows: $\mathbf{Q}_{\mathcal{A}}(s_0) = 1$ and if $\mathcal{A}(\pi) = (a, \mu)$ for some $a \in \Sigma_{?!}$ and $\mu \in Dist(S)$, then $\mathbf{Q}_{\mathcal{A}}(\pi a \mu s) = \mathbf{Q}_{\mathcal{A}}(\pi) \cdot \mu(s)$, and if $\mathcal{A}(\pi) = \bot$, then $\mathbf{Q}_{\mathcal{A}}(\pi \bot) = \mathbf{Q}_{\mathcal{A}}(\pi)$.

This probability is embedded within a *probability space* associated to the given adversary $\mathcal{A}$. Note that it is necessary to reason about the (probabilistic) behaviour

of adversaries in terms of their respective *probability spaces* —the mere assignment of probabilities to paths via distributions does not suffice. [1]

Definition 5.5 (Probability space). *A* probability space $\langle \Omega, \mathcal{F}, \mathbf{P} \rangle$ *consists of*

- Ω*, the* sample space*.*
- $\mathcal{F} \subseteq 2^{\Omega}$*, a* σ-field*, i.e.,* $\mathcal{F}$ *contains* Ω*, and is closed under countable union and complementation.*
- $\mathbf{P} : \mathcal{F} \mapsto [0,1]$*, a* probability measure *on* $\mathcal{F}$*, such that* $\mathbf{P}(\Omega) = 1$*, and* $\mathbf{P}(\cup_i X_i) = \sum_i \mathbf{P}(X_i)$*, where the* X_i *are pair-wise disjoint subsets of* $\mathcal{F}$*.*

This leads to the notion of a probability space associated to an adversary $\mathcal{A}$, by considering for each finite path π generated by $\mathcal{A}$ the corresponding *cylinder* $cyl(\pi)$ containing all *maximal* paths with π as prefix.

Definition 5.6 (Probability space associated to an adversary). *The* probability space *associated to an adversary* $\mathcal{A}$ *of a PTS* $[\mathcal{P}]$ *is* $\langle \Omega_{\mathcal{A}}, \mathcal{F}_{\mathcal{A}}, \mathbf{P}_{\mathcal{A}} \rangle$*, where*

- $\Omega_{\mathcal{A}} = Path^{max}_{\mathcal{A}}([\mathcal{P}])$
- $\mathcal{F}_{\mathcal{A}}$ *is the smallest* σ*-field containing the* cylinder sets $\{\, cyl(\pi) \mid \pi \in Path^{*}_{\mathcal{A}}([\mathcal{P}]) \}$*, where* $cyl(\pi) = \{\pi' \in \Omega_{\mathcal{A}} \mid \pi \text{ prefix of } \pi'\}$
- $\mathbf{P}_{\mathcal{A}}$ *is the unique measure on* $\mathcal{F}_{\mathcal{A}}$ *such that* $\mathbf{P}_{\mathcal{A}}(cyl(\pi)) = \mathbf{Q}_{\mathcal{A}}(\pi)$ *for all* $\pi \in Path^{*}_{\mathcal{A}}([\mathcal{P}])$*.*

Measure-theoretic arguments ensure that $\langle \Omega_{\mathcal{A}}, \mathcal{F}_{\mathcal{A}}, \mathbf{P}_{\mathcal{A}} \rangle$ is indeed a probability space. We are now positioned to characterize the semantics of a given PA $\mathcal{P}$ in terms of the *trace distribution* for some adversary $\mathcal{A}$ of its PTS $[\mathcal{P}]$. Such a trace distribution is obtained from the probability space associated to paths under the adversary $\mathcal{A}$ by omitting all states and distributions.

Definition 5.7 (Trace distribution). *The* trace distribution $\mathcal{T} = trdist_{\mathcal{A}}([\mathcal{P}])$ *of an adversary* $\mathcal{A}$ *of a PTS* $[\mathcal{P}] = \langle S, s_0, \Sigma, \Delta \rangle$ *is the probability space* $\langle \Omega_{\mathcal{T}}, \mathcal{F}_{\mathcal{T}}, \mathbf{P}_{\mathcal{T}} \rangle$*, where*

- $\Omega_{\mathcal{T}} = \Sigma^{*}_{?!} \cup \Sigma^{\omega}_{?!}$*.*
- $\mathcal{F}_{\mathcal{T}}$ *is the smallest* σ*-field containing* $\{cyl(w) \mid w \in \Sigma^{*}_{?!}\}$*, where* $cyl(w) = \{w' \in \Omega_{\mathcal{T}} \mid w \text{ prefix of } w'\}$*.*
- $\mathbf{P}_{\mathcal{T}}(X) = \mathbf{P}_{\mathcal{A}}(\{\pi \in Path^{max}_{\mathcal{A}}([\mathcal{P}]) \mid trace(\pi) \in X\})$ *for all* $X \in \mathcal{F}_{\mathcal{T}}$*.*

The sample space of $trdist_{\mathcal{A}}([\mathcal{P}])$ are sets of finite and infinite action-sequences, referred to as traces. Measurable elements are sets of traces obtained via a standard cylinder construction. The measure of set X is the probability of the set of maximal paths of PTS $[\mathcal{P}]$ under adversary $\mathcal{A}$ yielding a trace in X. Measure-theoretic arguments ensure the well-definedness of the above probability space. We denote by $trdist([\mathcal{P}])$ the set of the trace distributions of the PTS $[\mathcal{P}]$ under all possible (memoryless and admissible) adversarial resolutions.

Thus: $trdist([\mathcal{P}]) = \{\, trdist_{\mathcal{A}}([\mathcal{P}]) \mid \mathcal{A} \in Adv([\mathcal{P}]) \,\}$.

[1] In fact, the comparison of two probabilities that were not defined in the same probability space resulted in an erroneous proof of correctness [Seg00] for the earliest randomized algorithm for mutual exclusion [Rab82].

Definition 5.8 (Trace distribution equivalence). $\mathcal{P}_1 \equiv_{TD} \mathcal{P}_2$ *iff* $trdist([\mathcal{P}_1]) = trdist([\mathcal{P}_2])$.

This *trace distribution equivalence* $\equiv_{TD}$ enables the comparison of the behaviours of PA via the following proposition that gives a sufficient condition relating $\equiv_{TD}$ to the adversaries of the PA being compared.

Proposition 5.1 (From $\equiv$ on adversaries to $\equiv_{TD}$). *Given PA* $\mathcal{P}_1$ *and* $\mathcal{P}_2$ *over the same alphabet* Σ*, the same state space* S*, and the same set of distributions* $Dist(S)$*. If* $\forall \mathcal{A}_i \in Adv([\mathcal{P}_i])\ \exists \mathcal{A}_{3-i} \in Adv([\mathcal{P}_{3-i}]) : \mathcal{A}_i \equiv \mathcal{A}_{3-i}$ *(where* $i \in \{1,2\}$*), then* $\mathcal{P}_1 \equiv_{TD} \mathcal{P}_2$.

Proof. The PA $\mathcal{P}_1$ and $\mathcal{P}_2$ have the same alphabet Σ, the same state space S, and the same set of distributions $Dist(S)$, where each of Σ, S, and $Dist(S)$ is *finite*, and thus the induced PTS $[\mathcal{P}_1]$ and $[\mathcal{P}_2]$ are also finite (cf. Definition 5.1 and Remark 5.2). This finiteness rules out the existence of *non-cyclic infinite paths* in both $Path^{max}_{\mathcal{A}_1}([\mathcal{P}_1])$ and $Path^{max}_{\mathcal{A}_2}([\mathcal{P}_2])$, which might otherwise arise owing to König's Lemma [Koe36] for infinite graphs. Thus, reasoning about maximal paths in $Path^{max}_{\mathcal{A}_1}([\mathcal{P}_1])$ and $Path^{max}_{\mathcal{A}_2}([\mathcal{P}_2])$ reduces to reasoning about their finite prefixes of arbitrary length in $Path^{*}_{\mathcal{A}_1}([\mathcal{P}_1])$ and $Path^{*}_{\mathcal{A}_2}([\mathcal{P}_2])$, owing to all infinite paths in $Path^{max}_{\mathcal{A}_1}([\mathcal{P}_1])$ and $Path^{max}_{\mathcal{A}_2}([\mathcal{P}_2])$ being necessarily *cyclic*. In particular, two equivalent adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ –with $\mathcal{A}_1 \in Adv([\mathcal{P}_1])$ and $\mathcal{A}_2 \in Adv([\mathcal{P}_2])$– will induce the same set of maximal paths, i.e., $Path^{max}_{\mathcal{A}_1}([\mathcal{P}_1]) = Path^{max}_{\mathcal{A}_2}([\mathcal{P}_2])$ whenever $\mathcal{A}_1 \equiv \mathcal{A}_2$. Now, the trace distribution of a PTS for a given adversary is uniquely specified by the channel alphabet and the set of maximal paths under that adversary (Definition 5.7). Equivalent adversaries will therefore result in equal trace distributions, i.e., $\mathcal{A}_1 \equiv \mathcal{A}_2 \ \Rightarrow trdist_{\mathcal{A}_1}([\mathcal{P}_1]) = trdist_{\mathcal{A}_2}([\mathcal{P}_2])$. Further, we have that $\forall \mathcal{A}_i \in Adv([\mathcal{P}_i])\ \exists \mathcal{A}_{3-i} \in Adv([\mathcal{P}_{3-i}]) : \mathcal{A}_i \equiv \mathcal{A}_{3-i}$ (where $i \in \{1,2\}$), thus entailing $trdist([\mathcal{P}_1]) = trdist([\mathcal{P}_2])$. By Definition 5.8, this means $\mathcal{P}_1 \equiv_{TD} \mathcal{P}_2$. □

Composing PA.

We have thus far considered the semantics of PA that operate in isolation. In practice, however, PA need to be able to *communicate* with each other in order to effectively model the inter-component interactions within a randomized distributed system. We now define sequential, parallel, and choice composition of PA (denoted ;, ‖, and +, respectively) for assembling PA into a composite system.

Definition 5.9 (Sequential composition). *Given PA* $\mathcal{P}_i = \langle L_i, l_{0i}, l_{f_i}, V_i, \Sigma_i, prob_i \rangle$*, where* $i \in \{1,2\}$ *with* $L_1 \cap L_2 = \emptyset$*. Their* sequential composition*, denoted* $\mathcal{P}_1;\mathcal{P}_2$*, is the PA* $\langle L, l_0, l_f, V_1 \cup V_2, \Sigma_1 \cup \Sigma_2, prob \rangle$*, where* $L = (L_1 \setminus \{l_{f_1}\}) \cup L_2$ *with* $l_0 = l_{01}, l_f = l_{f_2}$ *and* $prob = prob'_1 \cup prob_2$.

Here $prob'_1 = prob_1[l_{02} \leftarrow l_{f_1}]$ is defined by $(l, a, \phi, \mu) \in prob_1$ iff $(l, a, \phi, \nu) \in prob'_1$ with

$$\nu(\psi, l') = \mu(\psi, l') \quad \text{if } l' \neq l_{f_1}, \text{ and}$$
$$\nu(\psi, l_{02}) = \mu(\psi, l_{f_1}) \text{ otherwise.}$$

The PA $\mathcal{P}_1;\mathcal{P}_2$ behaves first like $\mathcal{P}_1$ and subsequently like $\mathcal{P}_2$. To establish this, the final location l_{f_1} of $\mathcal{P}_1$ is amalgamated with the initial location l_{02} of $\mathcal{P}_2$. This is

reflected in the construction of *prob*, where basically all edges to l_{f_1} are redirected to l_{0_2}. This construction models sequential composition, since the final location l_{f_1} of $\mathcal{P}_1$ has no outgoing edges, cf. Definition 5.1.

While sequential composition describes the evolution of one PA followed by that of another, parallel composition $\|$ captures the concurrent evolution of two PA. We adopt the CCS-style composition [Mil89], i.e., parallel PA *synchronize* on common actions and act autonomously on all other actions —the latter is modelled by *interleaving*. In order to avoid any read-write and write-write conflicts w.r.t. the shared variables in the parallel PA, we require that edges corresponding to synchronizing actions are *non-interfering*. This notion is defined as follows. Consider edge $e = (l, a, \phi, \mu)$. Let the *write-set* of e, denoted $wr(e)$, be the set of variables occurring on the left-hand side of an update ψ with $\mu(\psi, l') > 0$ for some l'. The *read-set* of edge e, denoted $rd(e)$, consists of all data variables that appear in the guard ϕ or on the right-hand side of an update ψ with $\mu(\psi, l') > 0$ for some l'.

Definition 5.10 (Non-interfering edges). *Let E_1, E_2 be sets of edges with $e_1 \in E_1$ and $e_2 \in E_2$. The* non-interference relation *$\not\frown \subseteq E_1 \times E_2$ is defined by:*

$$e_1 \not\frown e_2 \text{ iff } rd(e_1) \cap wr(e_2) = wr(e_1) \cap rd(e_2) = wr(e_1) \cap wr(e_2) = \emptyset.$$

Thus, two edges are non-interfering whenever it is excluded that some variable that may change in one edge is read (or may change) in the other edge. The relation $\not\frown$ is then canonically lifted to sets of edges: $E_1 \not\frown E_2$ iff for all $e_1 \in E_1$ and $e_2 \in E_2$ we have $e_1 \not\frown e_2$. In the following, let $edges(a) = \{(l, a, \phi, \mu) \in prob \mid \exists l \in L\}$ denote the set of a-labelled edges emanating from some location $l \in L$. Two PA are now called non-interfering if their *synchronized* edges are non-interfering with each other.

Definition 5.11 (Non-interfering PA). *PA $\mathcal{P}_1$ and $\mathcal{P}_2$ over alphabet Σ_1 and Σ_2 respectively, are* non-interfering*, denoted $\mathcal{P}_1 \not\frown_{sync} \mathcal{P}_2$, if*

$$\forall a : a \in \Sigma_{i?!} \ \wedge \ \overline{a} \in \Sigma_{3-i?!}, \quad \text{it holds that } edges_i(a) \not\frown edges_{3-i}(\overline{a}),$$

where $edges_i(a)$ is the set of a-edges in PA $\mathcal{P}_i$, for $i \in \{1, 2\}$.

Remark 5.3. The above notion of non-interference is only w.r.t *synchronizing* actions on common channels. Note that this does not preclude shared-variable dependencies between actions on disjoint channels.

We now define parallel composition for such non-interfering PA.

Definition 5.12 (Parallel composition). *Let PA $\mathcal{P}_1$ and $\mathcal{P}_2$ with $\mathcal{P}_i = \langle L_i, l_{0_i}, l_{f_i}, V_i, \Sigma_i, prob_i \rangle$ for $i \in \{1, 2\}$ with $\mathcal{P}_1 \not\frown_{sync} \mathcal{P}_2$ and $L_1 \cap L_2 = \emptyset$. The* parallel composition *of $\mathcal{P}_1$ and $\mathcal{P}_2$, denoted $\mathcal{P}_1 \| \mathcal{P}_2$, is the PA $\langle L, l_0, l_f, V_1 \cup V_2, \Sigma_1 \cup \Sigma_2, prob \rangle$, where*

- *$L = L_1 \times L_2$ with $l_0 = (l_{0_1}, l_{0_2})$ and $l_f = (l_{f_1}, l_{f_2})$.*
- *$((l_1, l_2), a, \phi, \mu) \in prob$ iff $(l_i, a_i, \phi_i, \mu_i) \in prob_i$, $i \in \{1, 2\}$ and either:*
  1. Synchronization
     $a_i \in \Sigma_{i?!} \ \wedge \ \overline{a_i} \in \Sigma_{3-i?!}$, $a = \tau$, $\phi = \phi_1 \wedge \phi_2$, *and*
     $\mu(\psi, (l_1', l_2')) = \mu_1(\psi\lceil_{V_1}, l_1') \cdot \mu_2(\psi\lceil_{V_2}, l_2')$, *or*

2. Interleaving
$a_i \in \Sigma_{i?!}$, $a = a_i$, $\phi = \phi_i$, *and*
$$\mu(\psi,(l_1',l_2')) = \begin{cases} \mu_i(\psi\lceil_{V_i}, l_i') & \text{if } \psi\lceil_{V_{3-i}} = id_{V_{3-i}} \wedge l_{3-i}' = l_{3-i}, \\ 0 & \text{otherwise} \end{cases}$$

where $\psi\lceil_{V_i}$ *restricts* ψ *to the domain* V_i*, while* id_{V_i} *denotes the identity valuation over* V_i*, for* $i \in \{1,2\}$*.*

The parallel composition of $\mathcal{P}_1$ and $\mathcal{P}_2$ is thus the product of these PA, where the probability of synchronizing on a common channel is the product of the individual probabilities of performing the corresponding input/output actions in $\mathcal{P}_1$ and $\mathcal{P}_2$. Note that (as in CCS) we always allow each component to perform autonomous actions, where the other component idles with unit probability.

During the execution of randomized distributed algorithms, one often encounters non-deterministic selection between various components that are to be subsequently executed. Such non-deterministic selection is modelled by the operator + on PA.

Definition 5.13 (Non-deterministic choice). *Let PA* $\mathcal{P}_i = \langle L_i, l_{0i}, l_{f_i}, V_i, \Sigma_i, prob_i\rangle$*, for* $i \in \{1,2\}$ *with* $(L_1 \setminus \{l_{01}, l_{f_1}\}) \cap (L_2 \setminus \{l_{02}, l_{f_2}\}) = \emptyset$*,* $l_{01} = l_{02} = l_0$ *and* $l_{f_1} = l_{f_2} = l_f$*. The* non-deterministic choice *of* $\mathcal{P}_1$ *and* $\mathcal{P}_2$*, denoted* $\mathcal{P}_1 + \mathcal{P}_2$*, is the PA* $\langle L_1 \cup L_2, l_0, l_f, V_1 \cup V_2, \Sigma_1 \cup \Sigma_2, prob_1 \cup prob_2\rangle$*.*

Note that in the above definition, we require the initial and final locations of the component PA to be identical, while imposing disjointness of all other locations.

The compositional constructs $\|$ and + are both symmetric, and do not impose the precedence of one component over the other based on the dependencies between the individual PA.

In the next section, we will examine an asymmetric compositional operator and some relations that exploit such dependencies.

5.3 Action Independence, Precedence, Layering

A randomized distributed algorithm often consists of (sequential) phases that execute in parallel on different components, wherein a transition within a given phase can execute only after all *dependent* transitions in each preceding phase have been executed. In this section, we introduce the notion of *action independence* in PA and its corresponding PTS along the lines of action independence in Markov Decision Processes (MDPs) proposed in [BGC04, DN04]. This notion forms the basis for a *layered composition* operator, denoted $\bullet$, on PA that is intermediate between parallel and sequential composition. The $\bullet$ -operator is then used to formulate the communication-closed layer (CCL) laws in a probabilistic setting. For a PTS $[\mathcal{P}] = \langle S, s_0, \Sigma, \Delta\rangle$, let $act(s) = \{a \in \Sigma_{?!} \mid \exists \mu : (s,a,\mu) \in \Delta\}$ denote the set of enabled actions in state s.

Definition 5.14 (Action independence). *Let* $[\mathcal{P}] = \langle S, s_0, \Sigma, \Delta\rangle$ *be a PTS. The actions* $a, b \in \Sigma_{?!}$ *are* independent *in* $[\mathcal{P}]$*, denoted* $a \nsim b$*, iff for all states* $s \in S$ *with* $a, b \in act(s)$ *it holds that:*

1. *For any* $s' \in S$ *: if* $s \xrightarrow{a} \mu$ *and* $\mu(s') > 0$*, then* $b \in act(s')$*.*

2. *For any* $s' \in S$ *: if* $s \xrightarrow{b} \nu$ *and* $\nu(s') > 0$*, then* $a \in act(s')$.
3. *For any* $s'' \in S$ *:*

$$\sum\{\mu(s') \cdot \nu(s'') \mid s \xrightarrow{a} \mu \wedge s' \xrightarrow{b} \nu\} = \sum\{\mu(s') \cdot \nu(s'') \mid s \xrightarrow{b} \mu \wedge s' \xrightarrow{a} \nu\}.$$

Stated in words, actions a and b are independent whenever for every state s in which both actions are enabled, (1.) the occurrence of a does not disable b, (2.) and vice versa. Moreover, (3.) the total probability of reaching s'' from s by either performing a followed by b, or by performing b followed by a, coincides. Two distinct actions a and b are dependent, denoted $a \sim b$, iff they are not independent. The relation $\nsim$ is lifted to sets of actions in the standard manner. Notice that action independence is a semantic notion as it is defined on the underlying PTS $[\mathcal{P}]$ of the PA $\mathcal{P}$. The following proposition shows that the non-interference relation $\not\frown$, which can be determined by a simple syntactic analysis of $\mathcal{P}$, is a sufficient condition for action independence. Let $a \not\frown b$ whenever $edges(a) \not\frown edges(b)$.

Proposition 5.2 (Sufficient condition for action independence). *Let PA* $\mathcal{P}_1$ *and* $\mathcal{P}_2$ *be over the alphabets* Σ_1 *and* Σ_2*, respectively. Then for* $a_i \in \Sigma_{i?!}$*,* $i \in \{i, 2\}$*,* $a_1 \not\frown a_2$ *implies* $a_1 \nsim a_2$ *in* $[\mathcal{P}_1 \| \mathcal{P}_2]$.

We may now define the *layered composition*, denoted $\bullet$, of two PA. The operational interpretation of $\mathcal{P}_1 \bullet \mathcal{P}_2$ is that it behaves like the parallel composition of $\mathcal{P}_1$ and $\mathcal{P}_2$ except that an action a in $\mathcal{P}_2$ can only occur if all the actions in $\mathcal{P}_1$ on which a depends (in the sense of $\sim$) have already occurred. Stated differently, dependent actions in $\mathcal{P}_1$ and $\mathcal{P}_2$ are treated as in the sequential composition $\mathcal{P}_1; \mathcal{P}_2$ whereas independent actions are handled as in interleaving. For location l in PA $\mathcal{P}$, let $l \xrightarrow{*} l'$ denote that location l' is *syntactically reachable* from l through an arbitrary finite sequence of edges. Let $act(l) = \{a \in \Sigma_{?!} \mid (l, a, \phi, \mu)$ is an edge in $\mathcal{P}\}$ denote the set of enabled actions in location l.

Definition 5.15 (Layered composition). *Given two PA* $\mathcal{P}_i = \langle L_i, l_{0i}, l_{f_i}, V_i, \Sigma_i, prob_i \rangle$*, where* $i \in \{1, 2\}$*, with* $L_1 \cap L_2 = \emptyset$ *and* $\mathcal{P}_1 \not\frown_{sync} \mathcal{P}_2$*. The* layered composition *of* $\mathcal{P}_1$ *and* $\mathcal{P}_2$*, denoted* $\mathcal{P}_1 \bullet \mathcal{P}_2$ *is the PA* $\langle L, l_0, l_f, V_1 \cup V_2, \Sigma_1 \cup \Sigma_2, prob \rangle$*, where*

- $L = L_1 \times L_2$ *with* $l_0 = (l_{01}, l_{02})$ *and* $l_f = (l_{f_1}, l_{f_2})$.
- $((l_1, l_2), a, \phi, \mu) \in prob$ *iff* $(l_i, a_i, \phi_i, \mu_i) \in prob_i$*,* $i \in \{1, 2\}$ *and either:*
  1. Synchronization
     $a_i \in \Sigma_{i?!} \wedge \overline{a_i} \in \Sigma_{3-i?!}$*,* $a = \tau$*,* $\phi = \phi_1 \wedge \phi_2$*, and* $\mu(\psi, (l_1', l_2')) = \mu_1(\psi\lceil_{V_1}, l_1') \cdot \mu_2(\psi\lceil_{V_2}, l_2')$*, or*
  2. Interleaving $\mathcal{P}_1$
     $a = a_1 \in \Sigma_{1?!}$*,* $\phi = \phi_1$*, and* $\mu(\psi, (l_1', l_2')) = \begin{cases} \mu_1(\psi\lceil_{V_1}, l_1') & \text{if } \psi\lceil_{V_2} = id_{V_2} \wedge l_2' = l_2, \\ 0 & \text{otherwise} \end{cases}$,
     or
  3. Interleaving $\mathcal{P}_2$
     $a = a_2 \in \Sigma_{2?!}$*,* $\phi = \phi_2$*, and*
     $\mu(\psi, (l_1', l_2')) = \begin{cases} \mu_2(\psi\lceil_{V_2}, l_2') & \text{if } \psi\lceil_{V_1} = id_{V_1} \wedge l_1' = l_1 \wedge \\ & \quad \forall l_1^* : l_1 \xrightarrow{*} l_1^* : act(l_1^*) \nsim a \text{ in } [\mathcal{P}_1 \| \mathcal{P}_2], \\ 0 & \text{otherwise.} \end{cases}$

In the above definition, the first two clauses are the same as for synchronization and interleaving in parallel composition, whereas the third clause restricts the autonomous execution of actions by $\mathcal{P}_2$ to actions that are ensured to be independent of those in $\mathcal{P}_1$. PA $\mathcal{P}_1$ and $\mathcal{P}_2$ are *independent*, denoted $\mathcal{P}_1 \nsim \mathcal{P}_2$, iff for all $a_1 \in \Sigma_{1?!}$ and for all $a_2 \in \Sigma_{2?!}$ it holds that $a_1 \nsim a_2$ in the PTS $[\mathcal{P}_1 \| \mathcal{P}_2]$. Otherwise, $\mathcal{P}_1$ and $\mathcal{P}_2$ are dependent, denoted $\mathcal{P}_1 \sim \mathcal{P}_2$. In the presence of a shared variable dependence between $\mathcal{P}_1$ and $\mathcal{P}_2$, a related notion is that of *precedence* for PA, defined as follows:

Definition 5.16 (Precedence $\prec$). *For PA $\mathcal{P}_1$ and $\mathcal{P}_2$, $\mathcal{P}_1$* precedes *$\mathcal{P}_2$ in the parallel context of PA $\mathcal{C}_1$ and $\mathcal{C}_2$, denoted $\mathcal{P}_1 \prec_{\mathcal{C}_1,\mathcal{C}_2} \mathcal{P}_2$ if $(\mathcal{P}_1 \| \mathcal{P}_2 \| \mathcal{C}_1 \| \mathcal{C}_2) \equiv_{TD} (\mathcal{P}_1; \mathcal{P}_2) \| (\mathcal{C}_1 \| \mathcal{C}_2)$.*

The relation $\prec$ is transitive and enforces a precedence of $\mathcal{P}_1$ over $\mathcal{P}_2$ in the parallel context of $\mathcal{C}_1$ and $\mathcal{C}_2$ and by requiring that $\mathcal{P}_1$ and $\mathcal{P}_2$ do not synchronize on common channels, and that $trdist([(\mathcal{P}_2; \mathcal{P}_1) \| (\mathcal{C}_1 \| \mathcal{C}_2)]) = \emptyset$, which is ensured at the semantic level by appropriate guards querying the shared data variables. This is illustrated in the analysis of the randomized mutual exclusion algorithm in in the next section. The following proposition conjectures the behaviour of the operator $+$ w.r.t. the independence $\nsim$ and precedence $\prec$ relations for PA.

Proposition 5.3 (Relating $+$ with $\nsim$ and $\prec$). *For PA $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{C}_1$, and $\mathcal{C}_2$, with with $\mathcal{P}_1 = \mathcal{R}_1 + \mathcal{U}_1$ and $\mathcal{P}_2 = \mathcal{R}_2 + \mathcal{U}_2$,*

- $\mathcal{R}_1 \nsim \mathcal{R}_2 \ \wedge \ \mathcal{R}_1 \nsim \mathcal{U}_2 \ \wedge \ \mathcal{U}_1 \nsim \mathcal{R}_2 \ \wedge \ \mathcal{U}_1 \nsim \mathcal{U}_2$ *iff* $\mathcal{P}_1 \nsim \mathcal{P}_2$
- $\mathcal{R}_1 \prec_{\mathcal{C}_1,\mathcal{C}_2} \mathcal{R}_2 \ \wedge \ \mathcal{R}_1 \prec_{\mathcal{C}_1,\mathcal{C}_2} \mathcal{U}_2 \ \wedge \ \mathcal{U}_1 \prec_{C_1,C_1} \mathcal{R}_2 \ \wedge \ \mathcal{U}_1 \prec_{C_1,C_2} \mathcal{U}_2$ *iff* $\mathcal{P}_1 \prec_{\mathcal{C}_1,\mathcal{C}_2} \mathcal{P}_2$.

We then use layered composition and appropriate *independence* and *precedence* side-conditions for formulating the following communication closed layer (CCL) equivalences for PA.

Theorem 5.1 (CCL laws for PA). *For PA $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{Q}_1$, and $\mathcal{Q}_2$, with ($\mathcal{P}_1 \nsim \mathcal{Q}_2$ or $\mathcal{P}_1 \prec_{\mathcal{Q}_1,\mathcal{P}_2} \mathcal{Q}_2$) and ($\mathcal{Q}_1 \nsim \mathcal{P}_2$ or $\mathcal{Q}_1 \prec_{\mathcal{P}_1,\mathcal{Q}_2} \mathcal{P}_2$), the following* communication closed layer *(CCL) equivalences hold:*

1. $\mathcal{P}_1 \bullet \mathcal{Q}_2 \ \equiv_{TD} \ \mathcal{P}_1 \| \mathcal{Q}_2$ (IND)
2. $(\mathcal{P}_1 \bullet \mathcal{P}_2) \| \mathcal{Q}_2 \ \equiv_{TD} \ \mathcal{P}_1 \bullet (\mathcal{P}_2 \| \mathcal{Q}_2)$ (CCL-L)
3. $(\mathcal{P}_1 \bullet \mathcal{P}_2) \| \mathcal{Q}_1 \ \equiv_{TD} \ (\mathcal{P}_1 \| \mathcal{Q}_1) \bullet \mathcal{P}_2$ (CCL-R)
4. $(\mathcal{P}_1 \bullet \mathcal{P}_2) \| (\mathcal{Q}_1 \bullet \mathcal{Q}_2) \ \equiv_{TD} \ (\mathcal{P}_1 \| \mathcal{Q}_1) \bullet (\mathcal{P}_2 \| \mathcal{Q}_2)$ (CCL)

Proof. We attempt a proof of the law CCL-L. Given PA $\mathcal{P}_1, \mathcal{P}_2, \mathcal{Q}_2$, with $\mathcal{P}_1 \nsim \mathcal{Q}_2$ or $\mathcal{P}_1 \prec \mathcal{Q}_2$, we aim to show $\mathcal{R} \equiv_{TD} \mathcal{U}$, where $\mathcal{R} = (\mathcal{P}_1 \bullet \mathcal{P}_2) \| \mathcal{Q}_2$ and $\mathcal{U} = \mathcal{P}_1 \bullet (\mathcal{P}_2 \| \mathcal{Q}_2)$. Note that $\mathcal{R}$ and $\mathcal{U}$ are both defined over the same channel alphabet Σ, state space S, and set of distributions $Dist(S)$, each of which is finite. It therefore suffices by Proposition 5.1 to show that $\forall \mathcal{A} \in Adv([\mathcal{R}]) \ \exists \mathcal{A}' \in Adv([\mathcal{U}]) : \mathcal{A} \equiv \mathcal{A}'$, and vice versa. We show such adversarial equivalence by induction on path lengths n for finite paths in $Path^*_{\mathcal{A}}([\mathcal{R}])$ and $Path^*_{\mathcal{A}'}([\mathcal{U}])$. The existence of an equivalent adversary $\mathcal{A}$ in $Adv([\mathcal{R}])$ for each adversary $\mathcal{A}'$ in $Adv([\mathcal{U}])$ is not hard to see intuitively, as the parallel composition operator $\|$ dominates in $\mathcal{R}$ and the layered composition operator $\bullet$ in $\mathcal{U}$. The dominance of $\bullet$ induces fewer interleavings in $[\mathcal{U}]$ on the basis of the respective dependencies or precedences.

We now show: $\forall \mathcal{A} \in Adv([\mathcal{R}])\ \exists \mathcal{A}' \in Adv([\mathcal{U}]) : \mathcal{A} \equiv \mathcal{A}'$. By Definition 5.4, this amounts to showing: $\forall \mathcal{A} \in Adv([\mathcal{R}])\ \forall \pi \in Path^*_{\mathcal{A}}([\mathcal{R}])\ \exists \mathcal{A}' \in Adv([\mathcal{U}])\ \exists \pi' \in Path^*_{\mathcal{A}'}([\mathcal{U}])$: $|\pi| = |\pi'|\ \wedge\ last(\pi) = last(\pi')\ \wedge\ \mathcal{A}'(\pi') = \mathcal{A}(\pi)$.

For an adversary $\mathcal{A} \in Adv([\mathcal{R}])$ we proceed by induction on the length n of a path π generated by $\mathcal{A}$.

Induction Basis. For the case $n = 0$, we have $\pi = s_0$, where $s_0 = ((l_{0\,P1}, l_{0\,P2}, l_{0\,Q2}), \mathbf{v_0})$ [2] is the initial state of both $\mathcal{R}$ and $\mathcal{U}$. Thus we take $\pi' = \pi = s_0$ and choose $\mathcal{A}' = \mathcal{A}$, as $\mathcal{R}$ and $\mathcal{U}$ have the same alphabet Σ, the same state-space S, and the same set of distributions $Dist(S)$.

Induction Step. Consider a path $\pi_{n+1} \in Path^{n+1}_{\mathcal{A}}([\mathcal{R}])$ (of length $n+1$) with $pref_n(\pi_{n+1}) = \pi_n$ such that $last(\pi_n) = s_n = ((l_{P1}, l_{P2}, l_{Q2}), \mathbf{u})$ and $\mathcal{A}(\pi_n) = (a_n, \mu_n)$ for $a_n \in \Sigma_{?!}$ and $\mu_n \in Dist(S)$. This choice (a_n, μ_n) could have been performed in $\mathcal{R}$ by either (a1) $\mathcal{P}_1$, (a2) $\mathcal{P}_2$, (a3) $\mathcal{Q}_2$ individually, or as a synchronization involving either (b1) $\mathcal{P}_2$ and $\mathcal{Q}_2$, (b2) $\mathcal{P}_1$ and $\mathcal{Q}_2$, or (b3) $\mathcal{P}_1$ and $\mathcal{P}_2$.

The cases (a1) and (a2) are relatively straightforward. We now consider case (a3). This means that there exists an edge $e = (l_{Q2}, a_n, \phi, \mu_n)$ in $\mathcal{Q}_2$ with $\mu_n(\psi, l'_{Q2}) > 0$ for some $l'_{Q2} \in L_{Q2}$ and $\mathbf{u} \models \phi$, leading to a state $s_{n+1} = ((l_{P1}, l_{P2}, l'_{Q2}), \mathbf{u}')$ where $\mathbf{u}' = \psi(\mathbf{u})$. From the induction hypothesis, there exists $\mathcal{A}' \in Adv([\mathcal{U}])$ resulting in a path $\pi'_n \in Path^n_{\mathcal{A}'}([\mathcal{U}])$ with $last(\pi'_n) = s_n$ and $\mathcal{A}'(\pi_n) = (a_n, \mu_n)$. As either $\mathcal{P}_1 \nsim \mathcal{Q}_2$ or $\mathcal{P}_1 \prec_{\mathcal{Q}_1, \mathcal{P}_2} \mathcal{Q}_2$, this necessarily means the existence of the same edge e in $\mathcal{U}$ leading to the same state s_{n+1}. We have thus shown that $last(\pi_{n+1}) = last(\pi'_{n+1}) = s_{n+1}$ where π'_{n+1} is a path in $Path^{n+1}_{\mathcal{A}'}([\mathcal{U}])$ obtained by continuing from π'_n along the adversarial choice (a_n, μ_n) of $\mathcal{A}'$. Now let $\mathcal{A}(\pi_{n+1}) = (a_{n+1}, \mu_{n+1})$. Then the possibilities for the choice (a_{n+1}, μ_{n+1}) are again as in (a1)–(a3) and (b1)–(b3). Thus for the case (a3) (and also straightforwardly for (a1) and (a2)), using again the fact that either $\mathcal{P}_1 \nsim \mathcal{Q}_2$ or $\mathcal{P}_1 \prec_{\mathcal{Q}_1, \mathcal{P}_2} \mathcal{Q}_2$, we see that $(last(\pi'_{n+1}), a_{n+1}, \mu_{n+1}) \in \Delta_{\mathcal{U}}$, and thus take $\mathcal{A}'(\pi'_{n+1}) = (a_{n+1}, \mu_{n+1})$.

On the other hand, when $s_n = ((l_{f\,P1}, l_{f\,P2}, l_{f\,Q2}), \mathbf{u})$, we have $\mathcal{A}(\pi_n) = \mathcal{A}'(\pi'_n) = \perp$.

In (b1) there are edges with complementary actions enabled in $\mathcal{P}_2$ and $\mathcal{Q}_2$ individually that can synchronize to an edge labeled with τ in the context of $\mathcal{P}_2 \| \mathcal{Q}_2$. The part of the τ-edge stemming from $\mathcal{P}_2$ was possible in $\mathcal{P}_1 \bullet \mathcal{P}_2$, so it is possible also in $\mathcal{U} = \mathcal{P}_1 \bullet (\mathcal{P}_2 \| \mathcal{Q}_2)$. When $\mathcal{P}_1 \nsim \mathcal{Q}_2$ or $\mathcal{P}_1 \prec \mathcal{Q}_2$, the part of the τ-edge stemming from $\mathcal{Q}_2$ is also enabled in $\mathcal{U}$, and when executed in $\mathcal{U}$ yields the same state as when executed in $\mathcal{R} = (\mathcal{P}_1 \bullet \mathcal{P}_2) \| \mathcal{Q}_2$, with the same possible further adversarial choices.

The case (b2) reduces to (a1) and (a3), and the case (b3) reduces to (a1) and (a2). □

Remark 5.4. Note that each of the equivalences of Theorem 5.1 in fact relates *isomorphic* PA, as expressed by the equivalence $\equiv$ on adversaries at each transition step.

[2] $l_{0\,P1}$ is the initial location of $\mathcal{P}_1$, etc. We identify nested pairs $((x, y), z)$ and $(x, (y, z))$ of locations with tuples (x, y, z).

5.4 Partial Order Equivalence

We introduce in this section *partial order equivalence* ($\equiv^*_{po}$) for PA. As we are interested in comparing layered and sequential compositions of PA w.r.t $\equiv^*_{po}$, we first need to eliminate paths having τ-edges in the layered composition that have resulted from the synchronization of complementary actions.

For PA $\mathcal{P}_1$ and $\mathcal{P}_1$, let $\mathcal{P} = \mathcal{P}_1 \bullet \mathcal{P}_2$. Let $Path^*_{\setminus\{\tau\}}([\mathcal{P}])$ denote the set of finite paths of $\mathcal{P}$ with no τ-labelled transitions, i.e., $Path^*_{\setminus\{\tau\}}([\mathcal{P}]) = \{\pi \in Path^*([\mathcal{P}]) \mid \pi = s_0 a_1 \mu_1 s_1 a_2 \mu_2 s_3 \cdots \ \wedge \ \forall i : a_i \neq \tau\}$. Similarly, we denote by $Path^*{}_{\mathcal{A}\setminus\{\tau\}}([\mathcal{P}])$ the set of all finite paths without τ-edges under some adversary $\mathcal{A} \in Adv([\mathcal{P}])$. We then have the following proposition that relates the (probabilistic) behaviour of paths of $\mathcal{P}$ with those that do not contain τ-labelled edges.

Proposition 5.4 (Ignoring paths with τ-edges). *For PA $\mathcal{P}_1$ and $\mathcal{P}_1$, let $\mathcal{P} = \mathcal{P}_1 \bullet \mathcal{P}_2$. Then we have $\forall \mathcal{A} \in Adv([\mathcal{P}])\ \forall \pi \in Path^*_{\mathcal{A}}([\mathcal{P}]) : \exists \mathcal{A}_i \in Adv([\mathcal{P}])\ \exists \pi_i \in Path^*{}_{\mathcal{A}_i\setminus\{\tau\}}([\mathcal{P}]) : \mathbf{P}_{\mathcal{A}}(cyl(\pi)) = \mathbf{P}_{\mathcal{A}_i}(cyl(\pi_i))$, where $i \in \{1,2\}$.*

Proof. Let $\pi \in Path^*_{\mathcal{A}}([\mathcal{P}])$ be a finite path (under some adversary $\mathcal{A}$ of the layered composition $\mathcal{P}$) containing a τ-labelled edge. Then π is of the form $\pi = \pi' \tau \mu \pi''$, where π' and π'' are finite path-fragments. Then from Definition 5.15 and from our stipulation that τ-edges may only result from synchronization along complementary actions, there necessarily exist adversaries $\mathcal{A}_i$ and corresponding finite paths $\pi_i \in Path^*_{\mathcal{A}_i\setminus\{\tau\}}([\mathcal{P}])$ such that $\pi_1 = \pi' a \mu' s_1 \overline{a} \mu'' \pi''$ and $\pi_2 = \pi' \overline{a} \mu'' s_2 a \mu' \pi''$, where $a \in \Sigma_{i?!}$, $\overline{a} \in \Sigma_{3-i?!}$ ($i \in \{1,2\}$). Owing to the layered composition $\bullet$ admitting a CCS-style interleaving on all actions, and owing to the non-interference condition imposed on complementary actions (cf. Definition 5.15), we have that $\mu'(s_1) \cdot \mu''(\pi''(0)) = \mu''(s_2) \cdot \mu'(\pi''(0)) = \mu(\pi''(0))$, where $\pi''(0)$ denotes the starting state of the path-fragment π''. As in the proof of Proposition 5.1, reasoning about the path probabilities along π, π_1 and π_2 may then be extended to reasoning about their possibly infinite continuations via the cylinder constructions, thus yielding $\mathbf{P}_{\mathcal{A}}(cyl(\pi)) = \mathbf{P}_{\mathcal{A}_1}(cyl(\pi_1)) = \mathbf{P}_{\mathcal{A}_2}(cyl(\pi_2))$. $\square$

Note that π_1 and π_2 appearing in (the proof of) Proposition 5.4 differ only in the permutative ordering of the independent transitions corresponding to a and $\overline{a}$, and they both preserve the validity of properties that are insensitive to superfluous intermediate states (such as s_1 and s_2). For such (stutter invariant, or next-free) properties, it therefore suffices to consider paths in $Path^*_{\setminus\{\tau\}}([\mathcal{P}])$. For relating paths in $Path^*_{\setminus\{\tau\}}([\mathcal{P}])$ with those of sequential composition, we introduce the partial order equivalence $\equiv^*_{po}$ on (finite paths of) PA.

Definition 5.17 (po-equivalence $\equiv^*_{po}$ on finite paths). *For PA $\mathcal{P}_1$ and $\mathcal{P}_2$, let $\pi_1 \in Path^*_{\setminus\{\tau\}}([\mathcal{P}_1])$ and $\pi_2 \in Path^*_{\setminus\{\tau\}}([\mathcal{P}_2])$. Then $\pi_1 \equiv_{po} \pi_2$ iff there exist finite path fragments π, π' such that $\pi_1 = \pi a \mu s_1 b \mu' \pi'$ and $\pi_2 = \pi b \mu' s_2 a \mu \pi'$, where $a \nsim b$. The* partial order equivalence $\equiv^*_{po}$ *on finite paths without τ-labelled edges is then the reflexive, transitive closure of $\equiv_{po}$.*

Thus two paths related via $\equiv^*_{po}$ may be obtained from each other by repeated permutation of adjacent independent actions, modulo some superfluous intermediate states (such as s_1 and s_2 in the above definition). Note that by the independence $a \nsim b$, we have $\mu(s_1) \cdot \mu'(\pi'(0)) = \mu'(s_2) \cdot \mu(\pi'(0))$. We then introduce a notion of *layered normal form* to relate via $\equiv^*_{po}$ the τ-eliminated paths of a layered composition.

Definition 5.18 (Paths of $\bullet$ in LNF). $\pi \in Path^*_{\setminus\{\tau\}}([\mathcal{P}_1 \bullet \mathcal{P}_2])$ *is said to be in* layered normal form *(LNF) iff* $\pi = s_0 a_1 \mu_1 s_1 \ldots$ *such that* $\exists n : loc(s_n) = l_{f_1}$ *and* $\forall i \leq n : a_i \in \Sigma_{1?!} \ \wedge \ \forall j > n : a_j \in \Sigma_{2?!}$.

Here, $loc(s_n)$ denotes the location-component of the state s_n.

Thus, a path (in the PTS) of a layered composition $\mathcal{P}_1 \bullet \mathcal{P}_2$ is in LNF if it first consecutively executes actions of $\mathcal{P}_1$ leading to l_{f_1} and thereafter consecutively executes actions of $\mathcal{P}_2$. We denote by $Path^*_{LNF}([\mathcal{P}_1 \bullet \mathcal{P}_2])$ the set of all paths of $[\mathcal{P}_1 \bullet \mathcal{P}_2]$ that are in LNF. The definition of the layered composition $\bullet$ permits the execution of $\mathcal{P}_2$ actions only after the execution of all dependent $\mathcal{P}_1$ actions, thus leading to the following proposition.

Proposition 5.5 (Relating paths in LNF via $\equiv^*_{po}$). *For PA* $\mathcal{P}_1$ *and* $\mathcal{P}_2$*, where* $[\mathcal{P}_1]$ *is guaranteed to terminate in* l_{f_1} *with probability* 1*, the following holds:* $\forall \pi \in Path^*_{\setminus\{\tau\}}([\mathcal{P}_1 \bullet \mathcal{P}_2]) \ \exists \pi' \in Path^*_{LNF}([\mathcal{P}_1 \bullet \mathcal{P}_2]) : \pi \equiv^*_{po} \pi'$.

Proof. This proposition follows from the definition of $\bullet$ and the construction of paths in LNF, given that an action of $\mathcal{P}_2$ is allowed to execute in $\mathcal{P}_1 \bullet \mathcal{P}_2$ only after all dependent actions of $\mathcal{P}_1$ have been executed, and any action of $\mathcal{P}_2$ in a path of $\mathcal{P}_1 \bullet \mathcal{P}_2$ is thus necessarily independent of all $\mathcal{P}_1$ actions it precedes, and may therefore be permuted repeatedly to yield a path of $\mathcal{P}_1 \bullet \mathcal{P}_2$ in LNF. $\square$

We now lift the notion of $\equiv^*_{po}$ from paths to PA.

Definition 5.19 ($\equiv^*_{po}$ for PA). *For PA* $\mathcal{P}_1$ *and* $\mathcal{P}_2$*, we write* $\mathcal{P}_1 \equiv^*_{po} \mathcal{P}_2$ *iff the following holds for* $i \in \{1,2\}$*:* $\forall \mathcal{A}_i \in Adv([\mathcal{P}_i]) \ \forall \pi_i \in Path^*_{\mathcal{A}i\setminus\{\tau\}}([\mathcal{P}_i]) \ \exists \mathcal{A}_{3-i} \in Adv([\mathcal{P}_{3-i}]) \ \exists \pi_{3-i} \in Path^*_{\mathcal{A}3-i\setminus\{\tau\}}([\mathcal{P}_{3-i}]) : \pi_i \equiv^*_{po} \pi_{3-i}$

Remark 5.5. Proposition 5.4 ensures that it is sufficient for $\equiv^*_{po}$ to consider paths in $Path^*_{\mathcal{A}i\setminus\{\tau\}}([\mathcal{P}_i])$. As in Proposition 5.4, we have for the po-equivalent paths π_1 and π_2 above: $\mathbf{P}_{\mathcal{A}_1}(cyl(\pi_1)) = \mathbf{P}_{\mathcal{A}_2}(cyl(\pi_2))$.

A consequence of Definition 5.19 and Proposition 5.5 is that $\equiv^*_{po}$ should hold between the compositions $\bullet$ and ; .

Theorem 5.2 ($\equiv^*_{po}$ between $\bullet$ and ;). *For PA* $\mathcal{P}_1$ *and* $\mathcal{P}_2$ *where* $[\mathcal{P}_1]$ *is guaranteed to terminate in* l_{f_1} *with probability* 1*,* $\mathcal{P}_1 \bullet \mathcal{P}_2 \ \equiv^*_{po} \ \mathcal{P}_1; \mathcal{P}_2$.

Proof. Let $\pi \in Path^*_{LNF}([\mathcal{P}_1 \bullet \mathcal{P}_2])$. Then by Definition 5.18, there exists a path $\pi' \in Path^*([\mathcal{P}_1; \mathcal{P}_2])$ such that $\forall n \geq 0 : loc(\pi(n)) \approx loc(\pi'(n)) \ \wedge \ val(\pi(n)) = val(\pi'(n)) \ \wedge \ \pi(n) \xrightarrow{a_{n+1}} \mu_{n+1} \ \Rightarrow \ \pi'(n) \xrightarrow{a_{n+1}} \mu_{n+1} \ \wedge \ \mu_{n+1}(\pi(n+1)) = \mu_{n+1}(\pi'(n+1))$, where $\pi(n)$ is the n^{th} state of π, $loc(\pi(n))$ is the location component of that state, $val(\pi(n))$ is the vector of data valuations in that state, and $\approx$ is a relation defined as follows on the location spaces $L_1 \times L_2$ of $\mathcal{P}_1 \bullet \mathcal{P}_2$ and $(L_1 \cup L_2) \setminus \{l_{f_1}\}$ of $\mathcal{P}_1; \mathcal{P}_2$: $\forall l_1 \in L_1$ with $l_1 \neq l_{f_1} : (l_1, l_{02}) \approx l_1$ and $\forall l_2 \in L_2 : (l_{f_1}, l_2) \approx l_2$. Every path of $\mathcal{P}_1 \bullet \mathcal{P}_2$ in LNF is therefore exactly mimicked by a path of $\mathcal{P}_1; \mathcal{P}_2$ (modulo the relation $\approx$ on the location spaces of $\mathcal{P}_1 \bullet \mathcal{P}_2$ and $\mathcal{P}_1; \mathcal{P}_2$), and vice-versa. We also have from Proposition 5.5 that every finite (τ-eliminated) path of $\mathcal{P}_1 \bullet \mathcal{P}_2$ is either in LNF, or is po-equivalent to another path in LNF. $\square$

As the PA related by Theorem 5.1 are isomorphic, with the equivalence $\equiv$ on adversaries being stronger than $\equiv^*_{po}$, we then have the following corollary from Theorems 5.1 and 5.2.

Corollary 5.1. *Replacement of* $\bullet$ *by* ; *in the CCL laws yields po-equivalences, when* $[\mathcal{P}_1]$ *and* $[\mathcal{Q}_1]$ *are guaranteed to terminate in their respective final locations with probability* 1.

The paper [5] had originally stated that 5.1 would result in the probabilistic preservation of stutter invariant properties expressible in the Linear Temporal Logic (LTL) without the next operator. However, Luis Maria Ferrer Fioriti of Saarland University found this to be false when no restrictions were placed on the allowable adversarial resolutions of non-determinism. Based on our analysis in the next section (confirmed by experimental evaluation in [Sha15]) and the analysis in Section 8 of [MGCM08], we conjecture that Corollary 5.1 preserves *probabilistic reachability* properties as in [Sha15] under *memoryless, admissible, and oblivious* resolutions of non-determinism, when the PA are either acyclic,or where the (transition systems of the) PA of the first layer, i.e., $[\mathcal{P}_1]$ and $[\mathcal{Q}_1]$ terminate in their respective final locations with probability 1.

Methodology for layered separation.

We conclude this section with an outline of the intended methodology for layered transformation, illustrated on a schematic example. Given PA $\mathcal{P}_1, \mathcal{P}_2, \mathcal{Q}_1$ such that $\mathcal{Q}_1 \nsim \mathcal{P}_2$ or $\mathcal{Q}_1 \prec_{\mathcal{P}_1} \mathcal{P}_2$ we can reduce the state space of the PA system by applying to it the equivalences $\equiv_{TD}$ and $\equiv^*_{po}$, as shown below.

$$
\begin{array}{ll}
 & (\mathcal{P}_1; \mathcal{P}_2) \| \mathcal{Q}_1 \\
\equiv^*_{po} & \quad \{ \text{ Corollary 5.1 } \} \\
 & (\mathcal{P}_1 \bullet \mathcal{P}_2) \| \mathcal{Q}_1 \\
\equiv_{TD} & \quad \{ \text{ CCL-R } \} \\
 & (\mathcal{P}_1 \| \mathcal{Q}_1) \bullet \mathcal{P}_2 \\
\equiv^*_{po} & \quad \{ \text{ Corollary 5.1 } \} \\
 & (\mathcal{P}_1 \| \mathcal{Q}_1); \mathcal{P}_2.
\end{array}
$$

If each of $\mathcal{P}_1, \mathcal{P}_2, \mathcal{Q}_1$ has 10 locations then –from the definitions of the compositional constructs $\|$, $\bullet$, and ;– we see that the original PA system has 190 locations while the transformed system via layered separation (using $\equiv_{TD}$ and $\equiv^*_{po}$) has 109 locations. These reductions in the number of discrete locations to be explored become more pronounced as the number of system components increases, as will be illustrated next.

5.5 Case Study : Randomized Mutual Exclusion

To demonstrate the benefit of the layered separation achieved by means of the CCL laws and the associated po-equivalences, we consider the (revised) randomized

mutual exclusion algorithm $MUTEX$ for $N \geq 2$ processes by Kushilevitz and Rabin [KR92]. The authors in [KR92] describe their algorithm informally, partly in the running text and partly by pieces of pseudo code. This makes it difficult to obtain a complete picture of the algorithm. We present here (our view of) the essential ingredients of the algorithm in terms of PA composed sequentially, nondeterministically and in parallel. Our presentation is influenced by the algebraic analysis in the work of McIver, Gonzalia, Cohen, and Morgan [MGCM08] of parts of the algorithm in terms of separation theorems expressed in a probabilistic Kleene Algebra. We however analyze the algorithm of [KR92] in this section by means of layered separation.

The algorithm proceeds in rounds. To avoid keeping a record of the unbounded round numbers, the algorithm uses a randomized round number chosen by the process entering the critical section. During a round, every process $\mathcal{P}_i$ with $i \in \{1, \ldots, N\}$ cycles through four phases: *drawing* or *voting* $\mathcal{V}_i$, *notification* or *testing* $\mathcal{T}_i$, *critical section* $\mathcal{C}_i$, and *remainder* $\mathcal{R}_i$. In the voting phase, the processes participate in a lottery where they use a geometric distribution to pick numbers in the set $\{1, \ldots, B\}$, where $B = log_2 N + 4$. The value k is drawn with probability $1/2^k$ for $k \leq B - 1$, and with probability $1/2^{k-1}$ for $k = B$. The process that has drawn the highest number in the lottery will be notified as the winner in the subsequent notification round and will next enter the critical section. It stays thereafter in its remainder phase until it decides to participate in another voting phase.

The algorithm distinguishes between *even* and *odd* rounds, with different pseudocode for voting, notification, and critical section. Between an access to the critical section in an even round (called even critical section) and the access to the critical section in the subsequent odd round (called odd critical section) both an odd notification phase (for entering the odd critical section) and an even voting phase (for the next even critical section) will occur, and vice versa, with the roles of even and odd exchanged.

The processes $\mathcal{P}_1, \ldots, \mathcal{P}_N$ share several variables: b_even and b_odd (maximal lottery value in even and odd rounds, respectively), r_even and r_odd (random round number bit in even and odd rounds, respectively), s (binary semaphore guarding the critical section), p (parity bit serving as guards of even and odd phases), and w (indicator that the winner of lottery is notified). Whereas b_even, b_odd range over $\{0, \ldots, B\}$, the variables s, p, w range over $\{0, 1\}$, and r_even, r_odd range over $\{nil, 0, 1\}$. Additionally, each process $\mathcal{P}_i$ manipulates some local variables: $b(i)$ (lottery value drawn by $\mathcal{P}_i$), $d(i)$ (difference contributed by $\mathcal{P}_i$ to the maximum lottery value), $r(i)$ (round number), $w(i)$ (indicator that $\mathcal{P}_i$ knows it has won the lottery), and $pc(i)$ (program counter). Here $b(i)$ and $d(i)$ range over $\{0, \ldots, B\}$, $r(i)$ ranges over $\{nil, 0, 1\}$, $w(i)$ over $\{0, 1\}$, and $pc(i)$ over $\{nil, rem, it\}$.

In Fig. 5.1–5.4 we show for process $\mathcal{P}_i$ the PA representing the even phases $\mathcal{V}_i, \mathcal{T}_i, \mathcal{C}_i$ (and explain how PA for the odd phases are obtained) as well as PA representing $\mathcal{R}_i$ and idling versions of voting and notification. In these PA, all edges are labelled uniquely by corresponding actions, which we omit in the graphic representation. Synchronization between the processes $\mathcal{P}_1, \ldots, \mathcal{P}_N$ does not take place via these actions, but via guards checking the values of the shared variables.

The algorithm is modeled by the following parallel composition $MUTEX$, where each process $\mathcal{P}_i$ is a loop (represented by $*$) built up from sequences of nondeterministic choices of the phases defined above:

$$MUTEX = (\mathcal{P}_1 \| \ldots \| \mathcal{P}_N) \text{ with } P_i = \left(\begin{array}{c} odd\ \mathcal{V}_i + \mathcal{IV}_i + even\ \mathcal{C}_i \\ ; \\ odd\ \mathcal{T}_i + \mathcal{IT}_i + \mathcal{R}_i \\ ; \\ even\ \mathcal{V}_i + \mathcal{IV}_i + odd\ \mathcal{C}_i \\ ; \\ even\ \mathcal{T}_i + \mathcal{IT}_i + \mathcal{R}_i \end{array} \right)^*$$

The algorithm starts with the following initial values: $b\_even = b\_odd = 0$, $r\_even = r\_odd = nil$, $s = 0$, $p = 1$, $w = 1$, and for all $i \in \{1, \ldots, N\}$ we assume $b(i) = 0$, $d(i) = 0$, $r(i) = 0$, $pc(i) = nil$. Furthermore, we stipulate w.l.o.g that $w(1) = 1$ and $w(i) = 0$ for $i \in \{2, \ldots, N\}$.

The nondeterministic choices in $\mathcal{P}_i$ are restricted by the following sequencing constraints enforced via the local program counter $pc(i)$: once idle voting $\mathcal{IV}_i$ is chosen, only idle notification $\mathcal{IT}_i$ can follow due to the guard $pc(i) = it$; once the critical section $\mathcal{C}_i$ is chosen, only the remainder $\mathcal{R}_i$ can follow due to the guard $pc(i) = rem$. Once voting $\mathcal{V}_i$ is chosen and it ends with a contribution $d(i) > 0$, only notification $\mathcal{T}_i$ can follow. If $\mathcal{V}_i$ ends with $d(i) = 0$, only idle notification $\mathcal{IT}_i$ can follow.

Transformation into a layered representation.

We now show that $MUTEX$ can be transformed into a layered representation. Since the CCL laws stated in the previous section do not involve the $*$-operator on PA, we argue by considering finite, initial unfoldings of the $*$ and establish the layering for these unfoldings. Inferring a layered representation of the loops from a layered representation of the unfoldings is correct if the processes $\mathcal{P}_i$ of $MUTEX$ are prevented from pursuing infinite executions of idle voting and idle notification by a *fairness* assumption.

In Table 5.1 we display the unfolded $MUTEX$ up to round 4. To refer uniquely to different instances of the phases of each process $\mathcal{P}_i$ in the unfolding we attach a *round number* $r \in \mathbb{N}$ to them and thus write $\mathcal{V}_i^r$, $\mathcal{T}_i^r$, $\mathcal{C}_i^r$, $\mathcal{R}_i^r$, $\mathcal{IV}_i^r$, and $\mathcal{IT}_i^r$. The instances are defined as follows: $\mathcal{V}_i^r = even\ \mathcal{V}_i$ if r is even and $\mathcal{V}_i^r = odd\ \mathcal{V}_i$ if r is odd, and analogously for $\mathcal{T}_i^r$ and $\mathcal{C}_i^r$. Furthermore, for all $r \in \mathbb{N}$, $\mathcal{R}_i^r = \mathcal{R}_i$, $\mathcal{IV}_i^r = \mathcal{IV}_i$, and $\mathcal{IT}_i^r = \mathcal{IT}_i$. The initial unfolding up to round 4 is presented by the parallel composition in Table 1, with one sequential thread for each of the processes $\mathcal{P}_1, \ldots, \mathcal{P}_N$. The nondeterminism represented by $+$ is resolved during execution so that in each round r there is exactly one $i \in \{1, \ldots, N\}$ such that the critical section $\mathcal{C}_i^r$ is entered (thus guaranteeing mutual exclusion). Exactly for that i the remainder $\mathcal{R}_i^r$ will follow. All the other processes $j \neq i$ are already in their next round $r + 1$ performing voting $\mathcal{V}_j^{r+1}$ and then notification $\mathcal{T}_j^{r+1}$. For a proper initialization of the algorithm we stipulate an abridged round 0 in which one process, say $\mathcal{P}_1$, enters its critical section $\mathcal{C}_1^0$ without previous voting and testing.

We now establish crucial independence and precedence relations between the phases.

Proposition 5.6. *The phases of the unfolded $MUTEX$ satisfy the following independence and precedence conditions, where $r, r1, r2 > 0$ and $i, j \in \{1, \ldots, N\}$.*

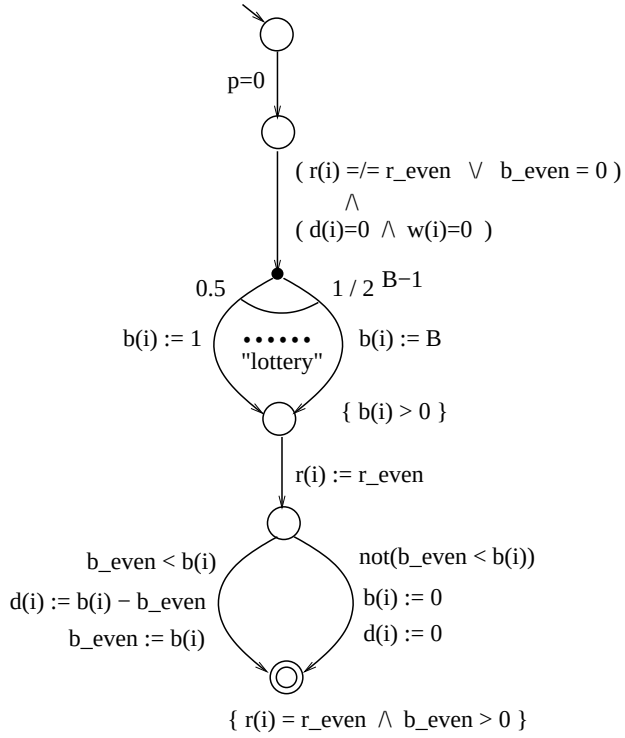

Fig. 5.1. The PA for *even* voting $\mathcal{V}_i$ with guard $p = 0$. In the PA for *odd* voting this guard is replaced by $p = 1$, and *r_even* and *b_even* are replaced by *r_odd* and *b_odd*, respectively. Note that the lottery has B branches with the last two branches having the same probability $1/2^{B-1}$.

1. *Remainder $\mathcal{R}_i^r$, idle voting $\mathcal{IV}_i^r$ and idle notification $\mathcal{IT}_i^r$ are* independent *($\nsim$) of any other phase.*
2. *If one of the rounds $r1$ and $r2$ is even and the other odd, voting $\mathcal{V}_i^{r1}$ and notification $\mathcal{T}_i^{r1}$ are* independent *of both voting $\mathcal{V}_j^{r2}$ and notification $\mathcal{T}_j^{r2}$, so $\mathcal{V}_i^{r1} \nsim \mathcal{V}_j^{r2}$, $\mathcal{V}_i^{r1} \nsim \mathcal{T}_j^{r2}$, $\mathcal{T}_i^{r1} \nsim \mathcal{V}_j^{r2}$, and $\mathcal{T}_i^{r1} \nsim \mathcal{T}_j^{r2}$.*
3. *A critical section $\mathcal{C}_i^r$ can only start if notification $\mathcal{T}_i^r$ has been completed, which in turn can only start if voting $\mathcal{V}_i^r$ has been completed: $\mathcal{V}_i^r \prec \mathcal{T}_i^r$ and $\mathcal{T}_i^r \prec \mathcal{C}_i^r$.*
4. *A critical section $\mathcal{C}_i^r$ can only start if all processes j that contributed $d(j) > 0$ to b_even or b_odd, respectively, in their voting $\mathcal{V}_j^r$ have completed their notification $\mathcal{T}_j^r$, so $\mathcal{T}_j^r \prec \mathcal{C}_i^r$ for all these j.*
5. *Notification $\mathcal{T}_j^{r+1}$ and voting $\mathcal{V}_j^{r+2}$ can only start if some critical section $\mathcal{C}_i^r$ has been completed: $\mathcal{C}_i^r \prec \mathcal{T}_j^{r+1}$ and $\mathcal{C}_i^r \prec \mathcal{V}_j^{r+2}$.*
6. *The critical section $\mathcal{C}_1^0$ and all votings $\mathcal{V}_j^1$ with $j \in \{2, \ldots, N\}$ are not preceded by any other phase and can thus start immediately.*

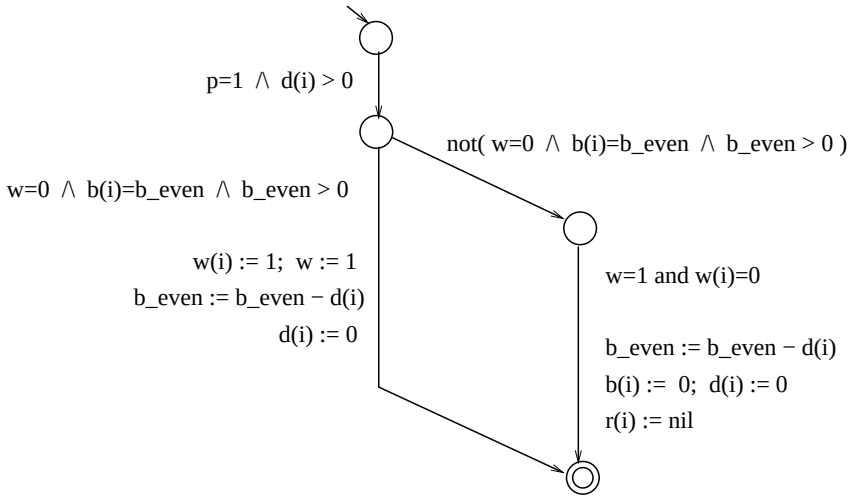

Fig. 5.2. The PA for *even* notification $\mathcal{T}_i$. Note that the guard is $p = 1 \wedge d(i) > 0$. In the PA for *odd* notification this guard is replaced by $p = 0 \wedge d(i) > 0$ and *b_even* by *b_odd*. The right branch is taken when $\mathcal{P}_i$ is not the winner, where it waits in the intermediate state until notified of the winner $\mathcal{P}_j$ via an update $w := 1$ in the left branch of $\mathcal{T}_j$.

Proof. Claim 1. This is clear because $\mathcal{R}_i^r$, $\mathcal{IV}_i^r$, $\mathcal{IT}_i^r$ do not access any of the shared variables.

Claim 2. Let r_1 be even and r_2 be odd. Of the shared variables, voting and notification both read p, and moreover, even voting $\mathcal{V}_i^{r1}$ accesses *r_even*, *b_even*, and even notification $\mathcal{T}_i^{r1}$ accesses *b_even* and w, whereas odd voting $\mathcal{V}_j^{r2}$ accesses *r_odd*, *b_odd*, and odd notification $\mathcal{T}_j^{r2}$ accesses *b_odd* and w. However, both $\mathcal{T}_i^{r1}$ and $\mathcal{T}_j^{r2}$ have the same effect on the shared variable w, namely updating w to 1 (via the left branch on Figure 2) in case the corresponding process has won the lottery in the preceding voting phase. So even voting $\mathcal{V}_i^{r1}$ and notification $\mathcal{T}_i^{r1}$ are independent of odd voting $\mathcal{V}_j^{r2}$ and notification $\mathcal{T}_j^{r2}$. Similar arguments apply when r_1 is odd and r_2 is even.

Claim 3. An even critical section $\mathcal{C}_i^r$ can only start if in round r process i left its notification $\mathcal{T}_i^r$ with $w(i) = 1$, which in turn is only possible if in the prior voting phase $\mathcal{V}_i^r$ process i drew the highest number $b(i)$ yielding $b(i) = b\_even$. The corresponding precedence relation holds for an odd critical section.

Claim 4. An even critical section $\mathcal{C}_i^r$ can only start if $b\_even = 0$ holds, which is only possible if in round r all processes j that increased *b_even* by contributing $d(j) > 0$ to it in their voting phase $\mathcal{V}_j^r$ have deducted the value $d(j)$ again from *b_even* in their notification phase $\mathcal{T}_j^r$. Thus all these processes j must have completed their notification $\mathcal{T}_j^r$ before $\mathcal{C}_i^r$ can start. The corresponding precedence relation holds for an odd critical section.

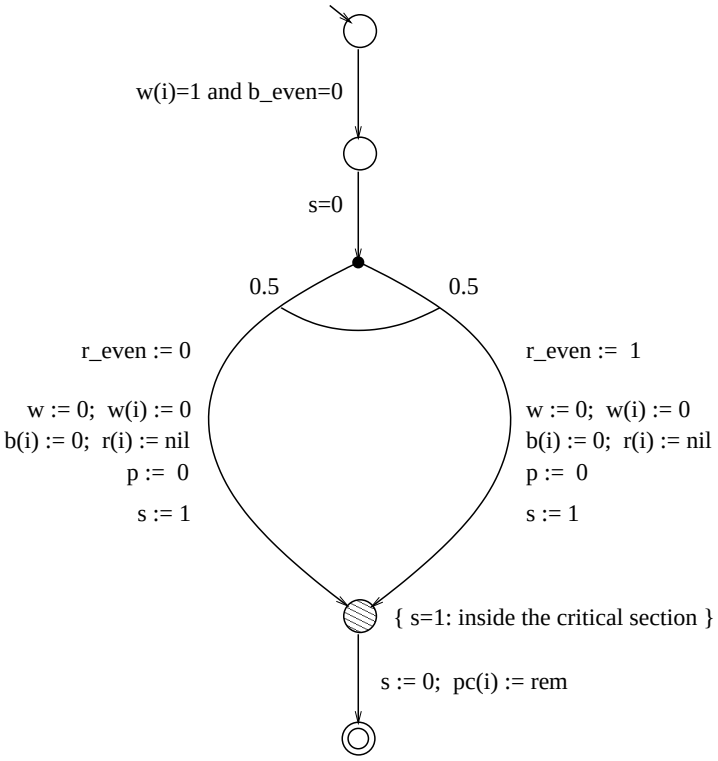

Fig. 5.3. The PA for *even* critical section $\mathcal{C}_i$. In the PA for *odd* critical section the update $p := 0$ is replaced by $p := 1$, and r_even by r_odd, and b_even by b_odd.

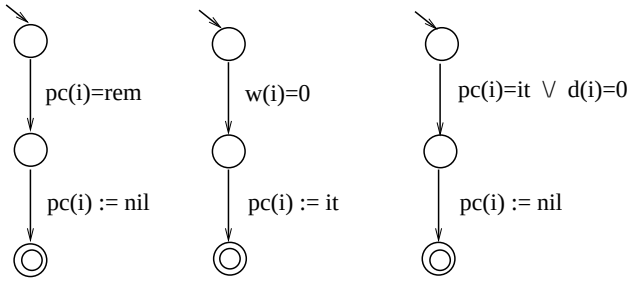

Fig. 5.4. The PA for remainder $\mathcal{R}_i$ (left), idle voting $\mathcal{IV}_i$ (middle), and idle notification $\mathcal{IT}_i$ (right).

$$
\begin{array}{rl}
\text{odd } (r=1): & \{p=1\}\mathcal{V}_1^1 + \mathcal{IV}_1^1 + \mathcal{C}_1^0 \;\|\; \mathcal{V}_2^1 + \mathcal{IV}_2^1 + \mathcal{C}_2^0 \;\|\; \ldots \;\|\; \mathcal{V}_N^1 + \mathcal{IV}_N^1 + \mathcal{C}_N^0 \\
& \quad ; \qquad\qquad ; \qquad\qquad \ldots \qquad ; \\
& \{p=0\}\mathcal{T}_1^1 + \mathcal{IT}_1^1 + \mathcal{R}_1^0 \;\|\; \mathcal{T}_2^1 + \mathcal{IT}_2^1 + \mathcal{R}_2^0 \;\|\; \ldots \;\|\; \mathcal{T}_N^1 + \mathcal{IT}_N^1 + \mathcal{R}_N^0 \\
& \quad ; \qquad\qquad ; \qquad\qquad \ldots \qquad ; \\
\text{even } (r=2): & \{p=0\}\mathcal{V}_1^2 + \mathcal{IV}_1^2 + \mathcal{C}_1^1 \;\|\; \mathcal{V}_2^2 + \mathcal{IV}_2^2 + \mathcal{C}_2^1 \;\|\; \ldots \;\|\; \mathcal{V}_N^2 + \mathcal{IV}_N^2 + \mathcal{C}_N^1 \\
& \quad ; \qquad\qquad ; \qquad\qquad \ldots \qquad ; \\
& \{p=1\}\mathcal{T}_1^2 + \mathcal{IT}_1^2 + \mathcal{R}_1^1 \;\|\; \mathcal{T}_2^2 + \mathcal{IT}_2^2 + \mathcal{R}_2^1 \;\|\; \ldots \;\|\; \mathcal{T}_N^2 + \mathcal{IT}_N^2 + \mathcal{R}_N^1 \\
& \quad ; \qquad\qquad ; \qquad\qquad \ldots \qquad ; \\
\text{odd } (r=3): & \{p=1\}\mathcal{V}_1^3 + \mathcal{IV}_1^3 + \mathcal{C}_1^2 \;\|\; \mathcal{V}_2^3 + \mathcal{IV}_2^3 + \mathcal{C}_2^2 \;\|\; \ldots \;\|\; \mathcal{V}_N^3 + \mathcal{IV}_N^3 + \mathcal{C}_N^2 \\
& \quad ; \qquad\qquad ; \qquad\qquad \ldots \qquad ; \\
& \{p=0\}\mathcal{T}_1^3 + \mathcal{IT}_1^3 + \mathcal{R}_1^2 \;\|\; \mathcal{T}_2^3 + \mathcal{IT}_2^3 + \mathcal{R}_2^2 \;\|\; \ldots \;\|\; \mathcal{T}_N^3 + \mathcal{IT}_N^3 + \mathcal{R}_N^2 \\
& \quad ; \qquad\qquad ; \qquad\qquad \ldots \qquad ; \\
\text{even } (r=4): & \{p=0\}\mathcal{V}_1^4 + \mathcal{IV}_1^4 + \mathcal{C}_1^3 \;\|\; \mathcal{V}_2^4 + \mathcal{IV}_2^4 + \mathcal{C}_2^3 \;\|\; \ldots \;\|\; \mathcal{V}_N^4 + \mathcal{IV}_N^4 + \mathcal{C}_N^3 \\
& \quad ; \qquad\qquad ; \qquad\qquad \ldots \qquad ; \\
& \{p=1\}\mathcal{T}_1^4 + \mathcal{IT}_1^4 + \mathcal{R}_1^3 \;\|\; \mathcal{T}_2^4 + \mathcal{IT}_2^4 + \mathcal{R}_2^3 \;\|\; \ldots \;\|\; \mathcal{T}_N^4 + \mathcal{IT}_N^4 + \mathcal{R}_N^3
\end{array}
$$

Table 5.1. Unfolding of the parallel $MUTEX$ algorithm up to round 4. For clarity, we have indicated the round numbers r and as assertions the p-values required in the guards of voting $\mathcal{V}_1^r$ and notification $\mathcal{T}_1^r$ of process $\mathcal{P}_1$ in each round. The voting and notification phases $\mathcal{V}_i^r$ and $\mathcal{T}_i^r$ of the other processes $\mathcal{P}_i$ have the same guards. Note that + binds stronger than ; and $\|$, and that ; binds stronger than $\|$.

Claim 5. Every even notification $\mathcal{T}_j^{r+1}$ in round $r+1$ and every odd voting $\mathcal{V}_j^{r+2}$ in round $r+2$ is guarded by $p=1$, which can only be established by the update $p := 1$ of a previous odd critical section $\mathcal{C}_i^r$ in round r. Analogously, every odd notification $\mathcal{T}_j^{r+1}$ and every even voting $\mathcal{V}_j^{r+2}$ is guarded by $p=0$, which can only be established by the update $p := 0$ of a previous even critical section $\mathcal{C}_i^r$.

Claim 6. Initially, we assume $w(1)=1$ and $b\_even=0$, so $\mathcal{C}_1^0$ can start immediately. Since initially also $p=0$ and $w(j)=0$, $d(j)=0$ holds for all $j \in \{2,\ldots,N\}$, voting $\mathcal{V}_j^1$ can start for these j. □

Note that the precedences here are in parallel contexts.

We are now prepared for the layered restructuring.

Proposition 5.7. *The unfolding of the parallel $MUTEX$ algorithm shown in Table 5.1 is po-equivalent ($\equiv_{po}^*$) to the unfolding of the layered $MUTEX$ algorithm shown in Table 5.2.*

Proof. We check that the claims of Proposition 5.6 imply the necessary independence and precedence relations between phases in different rounds and processes so that Theorem 5.1 and Corollary 5.1 can be applied. Since by Claim 1, the phases for remainder, idle voting, and idle notification are independent of any other phase, it suffices to analyze voting, notification, and critical section. Of these we pick phases

ph_j^r with a high round number r and argue that phases with a round number lower than r are either independent of ph_j^r or precede ph_j^r.

Case 1. Consider a voting $\mathcal{V}_j^4$ in round 4. Then in round 3, we have $\mathcal{V}_j^4 \nsim \mathcal{T}_i^3$ and $\mathcal{V}_j^4 \nsim \mathcal{V}_i^3$ by Claim 2. In round 2, we have $\mathcal{C}_i^2 \prec \mathcal{V}_j^4$ for the critical section of one process i due to Claim 5. Moreover, by Claim 4, we have $\mathcal{T}_k^2 \prec \mathcal{C}_i^2$ for all processes k that contributed $d(k) > 0$ to the maximum lottery value in their voting $\mathcal{V}_k^2$ before $\mathcal{T}_k^2$. All other processes l either abstained from voting before $\mathcal{C}_i^2$, which we can simulate by an independent idle voting $\mathcal{IV}_l^2$, or their voting $\mathcal{V}_l^2$ occurred before $\mathcal{C}_i^2$ but yielded only $d(l) = 0$. In both cases only an independent idle notification $\mathcal{IT}_l^2$ can follow. By Claim 5, any notification $\mathcal{T}_k^2$ in round 2 must be preceded by one critical section in round 1, say $\mathcal{C}_m^1 \prec \mathcal{T}_k^2$. Similar arguments apply when comparing phases in lower even-numbered rounds with $\mathcal{V}_j^4$.

Case 2. Consider a notification $\mathcal{T}_j^3$ in round 3. Then in round 2, we have $\mathcal{C}_i^2 \prec \mathcal{T}_j^3$ for the critical section of one process i due to Claim 5. Note that the votings $\mathcal{V}_l^3$ of round 3 must have occurred before $\mathcal{C}_i^2$ as this changes the value of p. After $\mathcal{C}_i^2$ only idle voting $\mathcal{IV}_l^3$ of round 3 remain possible. Furthermore, in round 2 we have $\mathcal{T}_j^3 \nsim \mathcal{V}_k^2$ and $\mathcal{T}_j^3 \nsim \mathcal{T}_k^2$ due to Claim 2. By Claim 5, any notification $\mathcal{T}_k^2$ in round 2 must be preceded by one critical section in round 1, say $\mathcal{C}_m^1 \prec \mathcal{T}_k^2$. For $\mathcal{C}_m^1$ we argue similar to Case 1, starting with $\mathcal{C}_i^2$ in round 2.

Case 3. Consider a critical section $\mathcal{C}_i^3$ in round 3. Again, we argue similar to Case 1, starting with $\mathcal{C}_i^2$ in round 2. □

$$
\begin{array}{c}
(\mathcal{V}_1^1 + \mathcal{IV}_1^1 + \mathcal{C}_1^0 \parallel \mathcal{V}_2^1 + \mathcal{IV}_2^1 + \mathcal{C}_2^0 \parallel \cdots \parallel \mathcal{V}_N^1 + \mathcal{IV}_N^1 + \mathcal{C}_N^0) \\
; \\
(\mathcal{T}_1^1 + \mathcal{IT}_1^1 + \mathcal{R}_1^0 \parallel \mathcal{T}_2^1 + \mathcal{IT}_2^1 + \mathcal{R}_2^0 \parallel \cdots \parallel \mathcal{T}_N^1 + \mathcal{IT}_N^1 + \mathcal{R}_N^0) \\
; \\
(\mathcal{V}_1^2 + \mathcal{IV}_1^2 + \mathcal{C}_1^1 \parallel \mathcal{V}_2^2 + \mathcal{IV}_2^2 + \mathcal{C}_2^1 \parallel \cdots \parallel \mathcal{V}_N^2 + \mathcal{IV}_N^2 + \mathcal{C}_N^1) \\
; \\
(\mathcal{T}_1^2 + \mathcal{IT}_1^2 + \mathcal{R}_1^1 \parallel \mathcal{T}_2^2 + \mathcal{IT}_2^2 + \mathcal{R}_2^1 \parallel \cdots \parallel \mathcal{T}_N^2 + \mathcal{IT}_N^2 + \mathcal{R}_N^1) \\
; \\
(\mathcal{V}_1^3 + \mathcal{IV}_1^3 + \mathcal{C}_1^2 \parallel \mathcal{V}_2^3 + \mathcal{IV}_2^3 + \mathcal{C}_2^2 \parallel \cdots \parallel \mathcal{V}_N^3 + \mathcal{IV}_N^3 + \mathcal{C}_N^2) \\
; \\
(\mathcal{T}_1^3 + \mathcal{IT}_1^3 + \mathcal{R}_1^2 \parallel \mathcal{T}_2^3 + \mathcal{IT}_2^3 + \mathcal{R}_2^2 \parallel \cdots \parallel \mathcal{T}_N^3 + \mathcal{IT}_N^3 + \mathcal{R}_N^2) \\
; \\
(\mathcal{V}_1^4 + \mathcal{IV}_1^4 + \mathcal{C}_1^3 \parallel \mathcal{V}_2^4 + \mathcal{IV}_2^4 + \mathcal{C}_2^3 \parallel \cdots \parallel \mathcal{V}_N^4 + \mathcal{IV}_N^4 + \mathcal{C}_N^3) \\
; \\
(\mathcal{T}_1^4 + \mathcal{IT}_1^4 + \mathcal{R}_1^3 \parallel \mathcal{T}_2^4 + \mathcal{IT}_2^4 + \mathcal{R}_2^3 \parallel \cdots \parallel \mathcal{T}_N^4 + \mathcal{IT}_N^4 + \mathcal{R}_N^3)
\end{array}
$$

Table 5.2. Unfolding of the layered $MUTEX$ algorithm up to round 4.

Proposition 5.7 holds for unfoldings up to any round number. This enables us to conclude that (under the assumption of fairness) the parallel $MUTEX$ is po-equivalent ($\equiv_{po}^*$) to the following layered version:

$$layered\_MUTEX = \left(\begin{array}{c} (odd\ \mathcal{V}_1 + \mathcal{IV}_1 + even\ \mathcal{C}_1 \parallel \cdots \parallel odd\ \mathcal{V}_N + \mathcal{IV}_N + even\ \mathcal{C}_N) \\ ; \\ (odd\ \mathcal{T}_1 + \mathcal{IT}_1 + \mathcal{R}_1 \quad \parallel \cdots \parallel \quad odd\ \mathcal{T}_N + \mathcal{IT}_N + \mathcal{R}_N) \\ ; \\ (even\ \mathcal{V}_1 + \mathcal{IV}_1 + odd\ \mathcal{C}_1 \parallel \cdots \parallel even\ \mathcal{V}_N + \mathcal{IV}_N + odd\ \mathcal{C}_N) \\ ; \\ (even\ \mathcal{T}_1 + \mathcal{IT}_1 + \mathcal{R}_1 \quad \parallel \cdots \parallel \quad even\ \mathcal{T}_N + \mathcal{IT}_N + \mathcal{R}_N) \end{array} \right)^*$$

For this layered version, we establish the following reduction in the number of locations. The number of locations $|L_P|$ for any of the constituent PA $\mathcal{P}$ is taken from Fig. 5.1–5.4. For nondeterministic choice we calculate $|L_{P+Q}| = |L_P| + |L_Q| - 2$, for sequential composition $|L_{P;Q}| = |L_P| + |L_Q| - 1$, and for parallel composition $|L_{P \parallel Q}| = |L_P| \cdot |L_Q|$, due to the definitions of these operators.

	number of locations in ...		
N ...	$MUTEX$ ...	$layered\_MUTEX$	reduction factor
3	15,625	1,453	≥ 10
4	390,625	10,781	≥ 36
5	9,765,625	81,085	≥ 120

Such reductions will speed up model checking the algorithm, as confirmed experimentally in [Sha15].

5.6 Conclusion

This chapter adopted the concept of communication-closed layers to probabilistic automata, a popular operational framework for the specification and verification of randomized distributed algorithms. The focus was on the theoretical underpinnings of incorporating layered separation into the framework of PA, for the analysis of complex, distributed compositions of PA. While errors remain in our analysis, and the precise choice of termination conditions and allowable resolutions of nondeterminism remain open, the application of layered separation has been nonetheless been shown in the modelling and analysis of a randomized mutual exclusion algorithm by Kushilevitz and Rabin [KR92], whose soundness has been experimentally confirmed in [Sha15].

McIver, Gonzalia, Cohen, and Morgan in [MGCM08] distilled from the description of this algorithm *algebraic axioms* that enabled them to simplify its reasoning by separation laws established for a *probabilistic Kleene Algebra* (pKA). We start operationally, formalizing parts of the mutual exclusion algorithm of [KR92] in terms of probabilistic automata and combine them by nondeterministic, sequential and parallel composition. Then we check whether suitable independence and precedence relations allow us to restructure parallel computations into layered ones, aiming (via a state-space reduction) at a simpler verification.

It is interesting to ask whether our formalization satisfies the four axioms of [MGCM08], which we cite here in their informal versions:

(1) Voting and notification commute.
(2) Notification occurs when the critical section is free.
(3) Voting occurs when the critical section is busy.
(4) It's more likely to lose, the later the vote.

Following Kushilevitz and Rabin, we distinguish even and odd rounds. This difference is not addressed in [MGCM08]. Regarding (1), we show in Proposition 5.6 the *independence* ($\sim$) of voting and notification, but only when voting is performed in an even round and notification in an odd round (or vice versa). Of course, independence implies commutativity. If voting $\mathcal{V}_i^{r1}$ and notification $\mathcal{T}_j^{r2}$ both occur in even rounds $r1, r2$ they interfere by writing to the shared variable *b_even*. Interestingly, $\mathcal{V}_i^{r1}$ and $\mathcal{T}_j^{r2}$ still commute, but in a degenerate sense: since $\mathcal{V}_i^{r1}$ is guarded by $p = 0$ and $\mathcal{T}_j^{r}$ by $p = 1$, and p is not changed by $\mathcal{V}_i^{r1}$ or $\mathcal{T}_j^{r2}$, both $\mathcal{V}_i^{r1}; \mathcal{T}_j^{r2}$ and $\mathcal{T}_j^{r2}; \mathcal{V}_i^{r1}$ end in a deadlock. The analogous statement is true for odd rounds. So axiom (1) is indeed satisfied.

Regarding (2) and (3), the unfolding of the layered $MUTEX$ in Table 5.2 shows that notification $\mathcal{T}_i^{r}$ takes place when no process is in its critical section and voting $\mathcal{V}_i^{r}$ takes place when one process is in its critical section. So axioms (2) and (3) are also satisfied.

Regarding (4), consider a sequence of even votings (cf. Fig. 5.1) by different processes in a given even round r, say $\mathcal{V}_1^r; \ldots; \mathcal{V}_m^r$ for $m \leq N$. Then after $\mathcal{V}_1^r$, process 1 will be the (temporary) winner with some value k drawn with probability $1/2^k$ for $1 \leq k \leq B-1$, and with probability $1/2^{k-1}$ for $k = B$, where $B = log_2 N + 4$, according to the geometric distribution assumed in the lottery. For process 2 to outperform process 1, it must draw a value $l > k$, but this is only possible with a probability of $1/2^l < 1/2^k$, etc. The same argument is true for odd votings. So axiom (4) is satisfied.

The satisfaction of the axioms (1)–(4) thus enables the application of the algebraic analysis of [MGCM08] to our operational model of Kushilevitz and Rabin's randomized mutual exclusion algorithm [KR92] in terms of probabilistic automata.

The pKA approach in [MGCM08] compares to our layering as follows:

- Central to the pKA approach in [MGCM08] is the exploitation of the *separation* theorems introduced earlier in [Coh00] for the non-randomized setting. Such separation theorems simplify the (algebraic) reasoning of (randomized) distributed systems by reducing complex interleavings into "separated" behaviours that admit individual analysis [MGCM08]. Our layering principle is similar in spirit: we exploit structural properties (formalized by means of suitable independence and precedence conditions that induce communication closedness) in complex randomized distributed systems so as to admit a *layered separation* that consequently simplifies verification.
- The work in [MGCM08] gives an *axiomatic* specification of some key properties of the randomized mutual exclusion algorithm of Kushilevitz and Rabin [KR92]. These axioms then enable simplified reasoning via separation theorems expressed in pKA. While the randomized mutual exclusion algorithm of [KR92] is likewise considered in this chapter for demonstrating the applicability and potential of our layered separation, we however provide an *operational* perspective of the

algorithm's intricate via suitable PA models. An analysis of these PA models then enables us to *derive* properties of the algorithm that were algebraically specified in Figure 7 of [MGCM08], as a consequence of our layered separation (cf. Proposition 5.7).

- Our layered separation and the pKA-based separation in [MGCM08] differ in their respective operational models and the verification techniques that they consequently admit. While the pKA-based separation aims for easier *assertional reasoning* or *interactive theorem proving* using pGCL in which probabilities can be treated as parameters, our layered separation is focussed on Segala's PA, aiming to achieve a *syntactic (partial order) state space reduction* prior to *model checking*.

Other related work.

Partial-order reductions based on ample-set constructions for Markov Decision Processes (MDPs) have been investigated in [BGC04, DN04] and shown to preserve next-free probabilistic linear- and branching-time properties. The work in [GDF09] discusses partial order reduction in the setting of distributed adversaries. A symbolic on-the-fly partial-order reduction (termed confluence reduction) has been proposed in [TSvdP11] for the preservation of probabilistic branching bisimulation, based on a process-algebraic framework for data-enriched PA [KvST12]. More recently, layering was investigated for (probabilistic) transition systems exhibiting may / must modalities [Sha15].

Open issues.

As mentioned earlier, the results of this chapter have not been fully worked out, and issues remain concerning the soundness of the results, especially with regard to the necessary termination conditions, the allowable resolutions of non-determinism, and the class of properties preserved. We have nonetheless presented the results in their current form, as our analysis of the mutual exclusion algorithm of [KR92] and its experimental confirmation in [Sha15] might provide useful insights into fixing these issues.

6

Robustness of Closed Perturbed Probabilistic Timed Automata

6.1 Introduction

The widely studied timed automaton (TA) model [AD94] augments finite automata with real-valued clocks, and provides for the modelling of *non-deterministic* real-time systems, by having transition-guards and location-invariants that the clock values need to satisfy. It however may be desirable to quantitatively express the relative *likelihood* of the system exhibiting certain behaviour, which is particularly relevant when evaluating real-time systems w.r.t. parameters such as reliability, availability, and mean-time- to- / between- failures.

Probabilistic Timed Automata (pTA) [KNSS02] are an extension of TA that model real-time systems in terms of discrete probability distributions annotating the transitions between locations, thereby expressing both *non-deterministic and probabilistic* behaviour. The pTA model has been used to check system requirements expressed as *probabilistic reachability* properties, as well as properties expressible in pTCTL (Probabilistic Timed Computation Tree Logic) [KNSS02, KNSW07] - an extension of TCTL to deal with probability. *Expected Reachability* properties are considered in [KNPS06]. Model-checking of pTA against such property classes is feasible, owing to a notion of *region equivalence* [KNSS02] and *digitization* [KNPS06], the latter being useful for the efficient computation of *expected reachability* properties, but whose application is restricted to the (realistic) subclass of pTA that are *closed and diagonal-free*, i.e., the guards and the invariants admit only non-strict comparisons of individual clocks with constants.

As with TA, the number of regions is exponential in the number of clocks, rendering region-based model-checking of pTA impractical. Efficient symbolic algorithms that manipulate Markov Decision Processes in the form of *Zone-Graphs* are presented in [KNSS02, KNSW07, KNP09]. The symbolic probabilistic reachability algorithm in [KNSS02] is fully forward, and is an adaptation of the standard zone-based algorithm used in TA model-checkers such as UPPAAL. It computes (a safe approximation of) the maximum probability of reaching an unsafe target state, or equivalently, the minimum probability of avoiding the unsafe state. This technique is particularly useful in the verification of quantitative *invariance* or *safety* properties, where we are interested in the probability of reaching an unsafe target. The probability of time-bounded reachability computed symbolically in [KNSS02] is

however not exact, but rather an upper bound on the true probability of reachability. More recently, [KNSW07, KNP09] have presented symbolic algorithms for pTA that entail backward analysis / stochastic game-based abstraction-refinement, for the exact computation of (max. / min.) probabilistic reachability. Given a pTA and a (bad) target state given as a location-zone pair (l, B), the symbolic algorithms in [KNSW07, KNP09] essentially compute the maximum / minimum probability (w.r.t. all possible resolutions of the non-deterministic choices in the pTA's underlying *probabilistic timed structure*, which is in fact an infinite-state *Markov Decision Process*) with which the target state could be reached.

However, all the above existing analysis techniques for pTA consider an idealized behaviour of the clocks, in the sense that they are assumed to be *fully synchronous*, unlike in practice, where the clocks could actually *drift*. The effect of such imprecisions, modelled by a parameter $\varepsilon > 0$ characterizing the extent of the potentially enhanced non-determinism has been investigated in Chapter 2, where it has been shown that an enlarged robust reach-set computation is not necessary in order to decide on robust safety for TA subject to realistic models of drifting clocks and system behaviour, either by limiting the life-time of the system to be finite, or by subjecting the drifting clocks to regular resynchronization.

In this chapter, we consider pTA with drifting clocks that could occur in practice, and investigate the adversarial role that such perturbations could possibly play in enhancing the non-deterministic behaviour of the system. In contrast with pTA that have perfectly synchronous clocks, pTA under perturbations exhibit greater non-determinism by having clocks that could drift relative to each other. We attempt to show that such enhanced non-determinism does not actually influence the computation of the minimum / maximum probabilities of avoiding / reaching a target state under arbitrarily small perturbations in the case of pTA having closed guards and invariants, when one considers the following two realistic models of drifting clocks and system behaviour, as in Chapter 2 for TA:

1. pTA models of finite life-time systems with the clock-rates being in an $\varepsilon > 0$ interval around 1, with the relative drift between clocks increasing without bound with the passage of time.
2. pTA models of regularly resynchronized system, with potentially infinite life-time, but where the relative drift between clocks is always bounded by $0 < \varepsilon < 1$ irrespective of the extent of time-passage.

We thus attempt to show that pTA (under certain restricted, yet realistic dynamics) are robust against clock-drift, and are thus amenable to the symbolic techniques used for computing probabilistic reachability in (unperturbed) pTA, thereby enabling an efficient computation of probabilistic performance measures on realistic system models.

The rest of the chapter is organized as follows: Section 6.2 discusses the pTA model, and introduces our notion of equivalence of adversaries and probabilistic timed structures that is later used to show preservation of probabilistic reachability. Section 6.3 considers pTA having relative drifts between the clocks, and shows the preservation of probabilistic reachability under arbitrarily small perturbations, for systems having finite life-time. Section 6.4 shows that probabilistic reachability is again preserved when one considers pTA where the relative drift between the clocks is always bounded through regular resynchronization, irrespective of the extent of time passage, with no restrictions on the system's lifetime. Section 6.5 concludes the

chapter. The results of this chapter, while appearing to be straightforward probabilistic extensions to the corresponding results for timed automata in Chapter 2, have not been fully worked out.

6.2 Probabilistic Timed Automata and Equivalences

Given a finite set C of *clocks*, a *clock valuation* over C is a map $v : C \mapsto \mathbb{R}_{\geq 0}$ that assigns a non-negative real value to each clock in C. If n is the number of clocks, a clock valuation is basically a point in $\mathbb{R}^n_{\geq 0}$, which we henceforth denote by $\mathbf{u}, \mathbf{v}$ etc. A *zone* over a set of clocks C is generated by the grammar $g ::= x \triangleright d \mid x - y \triangleright d \mid g \wedge g$, where $x, y \in C$, $d \in \mathbb{N}$, and $\triangleright \in \{<, \leq, >, \geq\}$. The set of zones over C is denoted $Z(C)$. A *closed zone* is one in which $\triangleright \in \{\leq, \geq\}$, and we denote the set of closed zones over C by $Z_c(C)$. A zone with no bounds on clock differences (i.e., with no constraint of the form $x - y \triangleright d$) is said to be *diagonal-free*, and we denote the corresponding set of zones by $Z_d(C)$. The set $Z_{cd}(C)$ denotes zones that are *both closed and diagonal-free*. The set $Z_{cdU}(C)$ denotes the set of *closed, diagonal-free zones having no lower bounds on the clocks*.

Definition 6.1 (Probabilistic Timed Automata). *[KNSS02] A* Probabilistic Timed Automaton *(pTA) is a tuple* $G = (L, C, l_0, inv, prob, < g_l >_{l \in L})$*, with*

- *a finite set* L *of* locations *and a finite set* C *of* clocks*, with* $|C| = n$
- *An* initial *location* $l_0 \in L$
- $inv : L \mapsto Z_{cdU}(C)$ *assigns* invariants *to locations*
- $prob : L \mapsto 2^{\mu(L \times 2^C)}$ *assigns to each location a (finite, non-empty) set* μ *of discrete probability distributions on* $L \times 2^C$
- *a family of functions* $< g_l >_{l \in L}$*, where, for any* $l \in L$*,* $g_l : prob(l) \mapsto Z_{cd}(C)$ *assigns to each* $p \in prob(l)$ *an enabling* guard $g_l(p)$*. An* edge *between two locations* (l, l') *involving a probability distribution* $p \in prob(l)$*, an enabling guard* $g_l(p)$*, and a* reset set $X \subseteq C$ *is denoted* $l \overset{p, g_l(p), X}{\longrightarrow} l'$*.*

Note that we assume that the enabling guards of the pTA are *closed and diagonal-free zones*. Location invariants in addition have only *upper-bounds* on clocks.

As for TA, the semantics of pTA include (invariant consistent) *time-passage* transitions within a given discrete location, and instantaneous *switch* transitions by taking an (enabled) edge between discrete-locations. However, in contrast with TA whose semantics involve switch transitions that are only non-deterministic, the switch transitions in pTA semantics are both *non-deterministic and probabilistic*, and consist of the following two steps performed in succession, starting from some location l: The system first makes a *non-deterministic choice* between the set of distributions $p \in prob(l)$, whose corresponding guard is satisfied by the current values of the clocks. Next, assuming that such a probability distribution p is chosen, the system then makes a corresponding *probabilistic transition*. For $l' \in L$ and $X \subseteq C$, the probability that the system executes a transition from l to l', with all clocks in X being reset to 0 is given by $p(l', X)$. The pTA semantics is described

formally in terms of its corresponding *Probabilistic Timed Structure* (pTS), defined as follows:

Definition 6.2 (Probabilistic Timed Structure). *[KNSS02] A* Probabilistic Timed Structure *(pTS) is a tuple $\mathcal{M} = (Q, Steps)$, where Q is a set of states, $Steps : Q \mapsto 2^{\mathbb{R}_{\geq 0} \times \mu(Q)}$ is a function that maps each state $q \in Q$ to a set $Steps(q)$ of pairs of the form (t, p), where $t \in \mathbb{R}_{\geq 0}$ and $p \in \mu(Q)$.*

$Steps(q)$ is the set of transitions that can be nondeterministically chosen in state q. Each transition takes the form (t, p), where t represents the duration of the transition, and p is the probability distribution used to determine the possible succesor states. Given a non-deterministic choice $(t, p) \in Steps(q)$ in state q, a probabilistic transition is then made to state q' with probability $p(q')$, after t time units.

Note that such a pTS is a (potentially infinite) *Markov Decision Process*. A path of a pTS $\mathcal{M} = (Q, Steps)$ is a (non-empty and possibly infinite) sequence $\pi = q_0 \overset{t_0,p_0}{\rightarrow} q_1 \overset{t_1,p_1}{\rightarrow} q_2 \cdots$, where $q_i \in Q, (t_i, p_i) \in Steps(q_i)$ and $0 < p_i(q_{i+1}) \leq 1$ for all $0 \leq i \leq |\pi|$. Paths in a pTS arise by the resolution of non-determinism by means of *adversaries*, defined as follows:

Definition 6.3 (Adversary of a Probabilistic Timed Structure). *[KNSS02] An* Adversary *A of a pTS $\mathcal{M} = (Q, Steps)$ resolves all its non-deterministic choices, by mapping every* finite path *π of $\mathcal{M}$ to a pair (t, p), such that $A(\pi) = (t, p) \in Steps(last(\pi))$, where $last(\pi)$ denotes the last state of π.*

We denote by $\mathcal{A}$ the set of all adversaries of the pTS $\mathcal{M}$. Similarly, an adversary A_i of history i resolves non-deterministic choices in $\mathcal{M}$ upto i steps, by considering every finite path π of $\mathcal{M}$ with $|\pi| \leq i$. The set of all adversaries of $\mathcal{M}$ of history i is denoted $\mathcal{A}_i$.

Under a given an adversary A of a pTS $\mathcal{M}$, the behaviour of the pTS is *purely probabilistic*, and corresponds to a *Sequential Markov Chain* defined over the set $Path^A_{fin}$ of *finite paths* under A of the pTS. This Sequential Markov Chain is given by $MC^A = (Path^A_{fin}, \mathbf{P^A})$, with :

$$\mathbf{P^A}(\pi_1, \pi_2) = \begin{cases} p(q) & \text{if } A(\pi_1) = (t, p) \ \wedge \ \pi_2 = \pi_1 \overset{t,p}{\rightarrow} q, \\ 0 & \text{otherwise.} \end{cases}$$

It is then possible to define a probability measure on the set of finite paths $Path^A_{fin}$ of a pTS under a given adversary that uniquely depends on the corresponding Sequential Markov Chain MC^A (see [KNSS02] for further details).

This then leads to our notion of *equivalent* adversaries between two probabilistic timed structures as follows:

Definition 6.4 (Equivalence of Adversaries). *Let $\mathcal{M}$ and $\mathcal{M}'$ be two pTS, and A resp. A' be some adversary of $\mathcal{M}$ resp. $\mathcal{M}'$. Let $MC^A = (Path^A_{fin}, \mathbf{P^A})$ be the Sequential Markov Chain corresponding to $\mathcal{M}$ under A, and $MC^{A'} = (Path^{A'}_{fin}, \mathbf{P^{A'}})$ be the Sequential Markov Chain corresponding to $\mathcal{M}'$ under A'. Then A is said to be* equivalent *to A', denoted $A \equiv A'$ if*

- $\forall \pi \in Path^A_{fin} \ [A(\pi) = (t, p) \ \wedge \ last(\pi) \overset{t,p}{\rightarrow} q \ \Rightarrow \ \exists \pi' \in Path^{A'}_{fin} : A'(\pi') = (t', p') \ \wedge \ last(\pi') \overset{t',p'}{\rightarrow} q' \ \wedge \ p(q) = p'(q')]$

- $\forall \pi \in Path^{A'}_{fin}\ [A'(\pi) = (t,p)\ \wedge\ last(\pi) \xrightarrow{t,p} q\ \Rightarrow\ \exists \pi' \in Path^{A}_{fin} : A(\pi') = (t',p')\ \wedge\ last(\pi') \xrightarrow{t',p'} q'\ \wedge\ p(q) = p'(q')]$

The following lemma then establishes a useful consequence of such equivalent adversaries:

Lemma 6.1. *Let $\mathcal{M}$ and $\mathcal{M}'$ be two pTS, and A and A' be their corresponding adversaries. Let $MC^A = (Path^A_{fin}, \mathbf{P^A})$ be the Sequential Markov Chain corresponding to $\mathcal{M}$ under A, and $MC^{A'} = (Path^{A'}_{fin}, \mathbf{P^{A'}})$ be the Sequential Markov Chain corresponding to $\mathcal{M}'$ under A'. If $A \equiv A'$, then:*

- $\forall \pi_1, \pi_2 \in Path^A_{fin}\ \exists {\pi_1}', {\pi_2}' \in Path^{A'}_{fin} : \mathbf{P^A}(\pi_\mathbf{1}, \pi_\mathbf{2}) = \mathbf{P^{A'}}({\pi_\mathbf{1}}', {\pi_\mathbf{2}}')$
- $\forall \pi_1, \pi_2 \in Path^{A'}_{fin}\ \exists {\pi_1}', {\pi_2}' \in Path^{A}_{fin} : \mathbf{P^{A'}}(\pi_\mathbf{1}, \pi_\mathbf{2}) = \mathbf{P^{A}}(\pi'_\mathbf{1}, \pi'_\mathbf{2})$

The proof is immediate from the definitions of a Sequential Markov Chain and equivalence of adversaries.

This lemma intuitively tells us that equivalent adversaries induce exactly the same underlying Sequential Markov Chain by mapping corresponding paths to *equivalent probability distributions* having *identical next-state transition probabilities*, but delaying for possibly different time instants. Our equivalence of adversaries thus induces a *time abstract* preservation of next-state transition probabilities of all enabled probability distributions.

The definition of equivalence of adversaries is now lifted to pTS (resp. sets of adversaries) as follows:

Definition 6.5 (Equivalence of pTS). *Let $\mathcal{M}$ and $\mathcal{M}'$ be two pTS, with $\mathcal{A}$ and $\mathcal{A}'$ denoting the corresponding sets of all possible adversaries. Then $\mathcal{M}$ and $\mathcal{M}'$ (resp. $\mathcal{A}$ and $\mathcal{A}'$) are said to be equivalent, denoted $\mathcal{M} \equiv \mathcal{M}'$ (resp. $\mathcal{A} \equiv \mathcal{A}'$) if:*

- $\forall A \in \mathcal{A}\ \exists A' \in \mathcal{A}' : A \equiv A'$
- $\forall A \in \mathcal{A}'\ \exists A' \in \mathcal{A} : A \equiv A'$

We now derive the pTS $\mathcal{M}_G$ corresponding to a given pTA G, in order to formally describe its semantics. For a given pTA $G = (L, C, l_0, inv, prob, < g_l >_{l \in L})$, the corresponding pTS is an (infinite-state) Markov Decision Process given by $\mathcal{M}_G = (Q_G, Steps_G)$, consisting of a set Q_G of (invariant consistent) states, and a transition relation (incorporating time-passage and location switches) that is both non-deterministic and probabilistic in nature.

- A state $q \in Q_G$ is a pair $(l, \mathbf{u}) \in L \times \mathbb{R}^n_{\geq 0}$ such that $\mathbf{u} \models inv(l)$.
- The function $Steps_G : Q_G \mapsto 2^{\mathbb{R}_{\geq 0} \times \mu(Q_G)}$ assigns to each state in Q_G a set of transitions, each of which take the form of a pair $(t, \tilde{p})$, consisting of a time duration $t \in \mathbb{R}_{\geq 0}$ and a probability distribution $\tilde{p} \in \mu(Q_G)$, over the set of states Q_G. For each $(l, \mathbf{u}) \in Q_G$, transitions are defined as follows:

  1. $(t, \tilde{p}) \in Steps_G(l, \mathbf{u})$ if $\exists p \in prob(l)$ such that:
     - $\mathbf{u} + t \models g_l(p) \wedge inv(l)$. Here $\mathbf{u} + t$ denotes the addition of t to each component of $\mathbf{u}$.
     - For any $(l', \mathbf{u'}) \in Q_G$, with $\mathbf{u'} = (\mathbf{u} + t)[X := 0]$ for an edge $l \xrightarrow{p, g_l(p), X} l'$, $\tilde{p}(l', \mathbf{u'}) = p(l', X)$

2. $(t, \tilde{p}) \in Steps_G(l, \mathbf{u})$ if
   - $\mathbf{u} + t \models inv(l)$
   - for any $(l', \mathbf{u'}) \in Q_G$:

$$\tilde{p}(l', \mathbf{u'}) = \begin{cases} 1, & \text{if } (l', \mathbf{u'}) = (l, \mathbf{u} + t) \\ 0 & \text{otherwise} \end{cases}$$

Thus, only action-distributions of the above PTS are probabilistic: delay-distribution pairs always comprise a *point distribution* [KNPS06]. We assume here for simplicity that each probabilistic switch (from a given distribution) is mapped to a unique target location. [1]

A *canonical initialized path* π of such a pTS is an alternating sequence of non-deterministic time-passage and probabilistic location switches, starting from the initial state, given as: $\pi = (l_0, \mathbf{0}) \overset{t_0, p_0}{\rightarrow} (l_1, \mathbf{x_1}) \overset{t_1, p_1}{\longrightarrow} (l_2, \mathbf{x_2}) \overset{t_2, p_2}{\rightarrow}$ ..... The set $\mathcal{A}^G$ of adversaries of M_G may then be defined as previously, over its finite, canonical initialized paths.

The above formulation of a pTS and its adversaries for a given pTA G leads us to the definition of the *probabilistic reachability problem* as follows: Let G be a pTA, $\mathcal{M}_G$ be its corresponding pTS having a set $\mathcal{A}^G$ of adversaries. For a given adversary $A \in \mathcal{A}^G$, we can define a probability measure over the set of paths from a state q of $\mathcal{M}_G$, and in particular, the probability $p_q^A(F)$ of reaching a target set of locations $F \subseteq L$ starting from q, under the given adversary A. For a given set F of target locations, probabilistic reachability then involves the computation of :

$$p^{min}{}_{\mathcal{M}_G}(F) = inf_{A \in \mathcal{A}_G} \; p_{q_0}^A(F)$$

and

$$p^{max}{}_{\mathcal{M}_G}(F) = sup_{A \in \mathcal{A}_G} \; p_{q_0}^A(F)$$

where $q_0 = (l_0, \mathbf{0})$. Practically feasible techniques exist for the computation of these probabilities, as given in [KNSS02, KNSW07, KNPS06, KNP09]. For the rest of this chapter, we will only consider the computation of $p_{\mathcal{M}_G}^{max}(F)$, as we are mainly interested in *probabilistic safety*, which corresponds to the minimum probability of avoiding a target location, or equivalently, the maximum probability of reaching a target location. Our results however easily extend to $p^{min}{}_{\mathcal{M}_G}(F)$.

Note that the pTA semantics discussed so far assumes *perfectly synchronous clocks*, unlike in practice, where the clocks could actually drift relative to each other. Such drifts increase the available non-deterministic choices for an adversary of the underlying pTS, resulting in a corresponding *perturbed* pTS. We however attempt to show in the rest of this chapter that such enhanced non-determinism in the perturbed pTS does not really play a role in the computation of $p^{max}{}_{\mathcal{M}_G}(F)$ for pTA having closed enabling guards and invariants, when one places realistic restrictions on either the system's life-time, or on the extent of the relative drifts between clocks. We aim to establish conditions wherein the *equivalence* as defined in this section is preserved between the probabilistic timed structures obtained from a given pTA by considering perfectly synchronous clocks on the one hand, and drifting clocks on the other. Such equivalence between the pTS crucially depends on the *closedness* of the guards and invariants in the syntactic structure of the pTA, and leads in turn to the preservation of probabilistic reachability, as will be elaborated next.

[1] We would otherwise have $\tilde{p}(l', \mathbf{u'}) = \sum_{X \in C \wedge (\mathbf{u}+t)[X:=0] = \mathbf{u'}} p(l', X)$ for switching transitions.

6.3 Robustness against drifting clocks under finite life-time

As in Section 2.3, the phenomenon of drifting clocks is modelled by introducing a parameter $\varepsilon > 0$ that characterizes the relative drift between the clocks. The rates of the clocks are assumed to be within the range $\left[\frac{1}{1+\varepsilon}, 1+\varepsilon\right]$. Note that, athough the clock-rates are bounded by an ε-interval around 1, the actual relative drift between the clocks may become arbitarily large with increasing delay, and therefore manifests as *unbounded* in systems having an infinite life-time. This effect has been exploited for TA in [Pur00, WDMR08, DK06, Dim07, JR11, KLMP14, San15] to show that even the smallest of drifts can lead to extra reachable states in TA that were previously unreachable under perfect clocks, even for closed TA that additionally satisfy the progress cycle assumption, wherein each clock of the TA is reset at least once in each cycle.

We will however attempt to show here that closed pTA (with no restrictions on the cycles) are "robust" relative to disturbances that manifest as unbounded relative drifts between the clocks, so long as the life-time of the system is finite. This result extends the corresponding result for TA in Section 2.3.

Such drifting clocks for a given pTA G under a drift parameter ε then result in a corresponding perturbed pTS $\mathcal{M}_G^\varepsilon = (Q_G^\varepsilon, Steps_G^\varepsilon)$, obtained as follows:

- A state $q \in Q_G^\varepsilon$ is a pair $(l, \mathbf{u}) \in L \times \mathbb{R}^n_{\geq 0}$ such that $\mathbf{u} \models inv(l)$.
- The function $Steps_G^\varepsilon : Q_G^\varepsilon \mapsto 2^{\mathbb{R}_{\geq 0} \times \mu(Q_G^\varepsilon)}$ assigns to each state in Q_G^ε a set of transitions, each of which take the form of a pair $(t, \tilde{p})$, consisting of a time duration $t \in \mathbb{R}_{\geq 0}$ and a probability distribution $\tilde{p} \in \mu(Q_G^\varepsilon)$, over the set of states Q_G^ε. For each $(l, \mathbf{u}) \in Q_G^\varepsilon$, transitions are defined as follows:
  1. $(t, \tilde{p}) \in Steps_G^\varepsilon(l, \mathbf{u})$ if $\exists p \in prob(l)$ such that

     $\exists \mathbf{u}' \in [\mathbf{u} + \frac{t}{1+\varepsilon}, \mathbf{u} + t(1+\varepsilon)] : \mathbf{u}' \models g_l(p) \wedge inv(l)$, and
     for any $(l'', \mathbf{u''}) \in Q_G^\varepsilon$, with $\mathbf{u''} = \mathbf{u}'[X := 0]$ for an edge $l \xrightarrow{p, g_l(p), X} l''$,
     $\tilde{p}(l'', \mathbf{u}'') = p(l'', X)$
  2. $(t, \tilde{p}) \in Steps_G^\varepsilon(l, \mathbf{u})$

     $\exists \mathbf{u}' \in [\mathbf{u} + \frac{t}{1+\varepsilon}, \mathbf{u} + t(1+\varepsilon)]$ s.t $\mathbf{u}' \models inv(l)$, and
     for any $(l'', \mathbf{u}'') \in Q_G^\varepsilon$:

$$\tilde{p}(l'', \mathbf{u}'') = \begin{cases} 1, & (l'', \mathbf{u}'') = (l, \mathbf{u}') \\ 0 & \text{otherwise} \end{cases}$$

As for the non-perturbed case, one can then define canonical initialized paths through such a perturbed pTS, and adversaries corresponding to such finite canonical initialized paths. We are particularly interested in adversaries of history $i \in \mathbb{N}$, corresponding to canonical initialized paths of length i, which we use in establishing the following theorem concerning the *equivalence* as defined in the previous section between the probabilistic timed structures of a given pTA with closed guards and invariants, having perturbed clocks on the one hand, and perfectly synchronized clocks on the other.

Theorem 6.1. *Given a pTA G (having closed guards and invariants), let $\mathcal{M}$ (resp. $\mathcal{M}^\varepsilon$) denote the corresponding pTS (resp. perturbed pTS with perturbation ε) as described in Section 2 (resp. Section 3)*[2]*. Let $\mathcal{A}_i$ (resp. $\mathcal{A}_i^\varepsilon$) denote the set of adversaries of $\mathcal{M}$ (resp. $\mathcal{M}^\varepsilon$) of history i defined over the corresponding sets of canonical initialized paths Π_i (resp. Π_i^ε) of length i. It then holds that $\forall i \in \mathbb{N}\ \exists \varepsilon_i > 0 : \mathcal{A}_i \equiv \mathcal{A}_i^{\varepsilon_i}$*

Proof. (Sketch) As the perturbed pTS exhibits greater non-determinism, it always includes behaviour that is possible without perturbation, i.e., every adversary of the non-perturbed pTS is equivalent (as defined in the previous section) to some adversary of the perturbed pTS. It therefore suffices to show that one can similarly find an equivalent adversary in the non-perturbed pTS for every adversary in the perturbed pTS. We define for this purpose the set of *grid-points* in $\mathbb{N}^n$ surrounding a given valuation $\mathbf{u} \in \mathbb{R}^n_{\geq 0}$ as $Grid(\mathbf{x}) = \{\mathbf{x_g} \in \mathbb{N}^n \mid dist(\mathbf{x}, \mathbf{x_g}) < 1\}$, where $dist(\mathbf{x}, \mathbf{x_g}) = max_{1 \leq i \leq n} |x_i - x_{g_i}|$. Induction over i should give: $\forall i \in \mathbb{N}\ \exists \varepsilon_i > 0\ \forall A \in \mathcal{A}_i^{\varepsilon_i}\ \forall \pi \in \Pi_i^{\varepsilon_i} : A(\pi) = (t, p)\ \wedge\ last(\pi) \xrightarrow{t,p} (l, \mathbf{x}) \Rightarrow\ \exists\ \pi' \in \Pi_i,\ A' \in \mathcal{A}_i,\ \mathbf{x'} \in Grid(\mathbf{x}) : A'(\pi') = (t', p')\ \wedge\ last(\pi') \xrightarrow{t',p'} (l, \mathbf{x'})\ \wedge\ p'(l, \mathbf{x'}) = p(l, \mathbf{x})$.
Note that the probabilistic choices here are preserved with identical next-state transition probabilities, while the non-deterministic delays are *time abstracted*. This is feasible owing to the fact that the guards and invariants in our case are closed, and are thus always enabled by some grid point. The proof then follows from the above result in conjunction with Definitions 6.4 and 6.5. □

Theorem 6.1 therefore implies that, for pTA having closed guards and invariants, it is always possible to choose an appropriate (strictly positive) value of the perturbation parameter ε depending on the computational history i, such that the enhanced non-determinism induced by the perturbation does not really affect the available *probabilistic choices*, leading to a preservation of probabilistic reachability properties. The following corollary is then an immediate consequence of Theorem 6.1 and Lemma 6.1.

Corollary 6.1. *For a given pTA G and a perturbation parameter $\varepsilon > 0$, and for a set of target locations F, let $p^{max} = sup_{A \in \mathcal{A}}\ p^A_{q_0}(F)$ denote the maximum probability of reaching F under all possible adversarial resolutions of the underlying pTS, while $p_i^\varepsilon = sup_{A \in \mathcal{A}_i^\varepsilon}\ p^A_{q_0}(F)$ denotes the maximum probability of reaching F under all possible adversarial resolutions of history i of the perturbed pTS corresponding to perturbation $\varepsilon > 0$. It then holds that:*

$$lim_{i \to \infty}\ lim_{\varepsilon \to 0}\ p_i^\varepsilon = p^{max}$$

Corollary 6.1 essentially states the preservation of the maximum probabilistic reachability under possibly unbounded relative drift between the clocks of the pTA, by choosing appropriately small perturbation parameters $\varepsilon > 0$ for increasingly large computational histories i, i.e., the larger the computational history, the smaller will be the corresponding admissible perturbation. From Theorem 6.1, one may also infer similar results for the minimum probability of reaching a target state under all possible adversial resolutions. Note that the limits in this corollary may not be swapped, as doing so would make the allowable perturbation independent of the history, which, in this pertubation model entailing unbounded relative drift between the clocks, would not preserve the maximum probabilistic reachability.

[2] We drop the reference to G here for ease of notation.

6.4 Robustness against drifting clocks under resynchronization

As in Section 2.4, we incorporate a *clock resynchronization scheme* into pTA by associating a *drift-offset* $\boldsymbol{\delta} \in [-\varepsilon, \varepsilon]^n$ for each clock valuation $\mathbf{x} \in \mathbb{R}^n$. This drift-offset keeps track of the extent to which the individual clocks in $\mathbf{x}$ have deviated from an implicit reference clock maintained by the synchronization scheme. The states of the underlying PTS in this semantics are thus tuples $(l, \mathbf{u}, \boldsymbol{\delta}) \in L \times \mathbb{R}^n \times [-\varepsilon, \varepsilon]^n$. As the deviation $\boldsymbol{\delta}$ is controlled by the synchronization scheme such that it always remains below ε, the corresponding *perturbed* PTS $\mathcal{M}_G^\varepsilon = (Q_G^\varepsilon, Steps_G^\varepsilon)$ for a given $\varepsilon > 0$ is obtained as follows:

- A state $q \in Q_G^\varepsilon$ is a triple $(l, \mathbf{u}, \boldsymbol{\delta}) \in L \times \mathbb{R}^n \times [-\varepsilon, \varepsilon]^n$ such that $\mathbf{u} \models inv(l)$.
- The function $Steps_G^\varepsilon : Q_G^\varepsilon \mapsto 2^{\mathbb{R} \times \mu(Q_G^\varepsilon)}$ assigns to each state in Q_G^ε a set of transitions, each of which take the form of a pair $(t, \tilde{p})$, consisting of a time duration $t \in \mathbb{R}$ and a probability distribution $\tilde{p} \in \mu(Q_G^\varepsilon)$, over the set of states Q_G^ε. For each $(l, \mathbf{u}, \boldsymbol{\delta}) \in Q_G^\varepsilon$, transitions are defined as follows:
  1. $(t, \tilde{p}) \in Steps_G^\varepsilon(l, \mathbf{u}, \boldsymbol{\delta})$ if $\exists p \in prob(l)$, $\exists \boldsymbol{\delta'} \in [-\varepsilon, \varepsilon]^n$ such that:
     - $\mathbf{u} + t - \boldsymbol{\delta} + \boldsymbol{\delta'} \models g_l(p) \wedge inv(l)$
     - For any $(l'', \mathbf{u''}, \boldsymbol{\delta''}) \in Q_G^\varepsilon$, with $\mathbf{u''} = (\mathbf{u} + t - \boldsymbol{\delta} + \boldsymbol{\delta'})[X := 0]$ and $\boldsymbol{\delta''} = \boldsymbol{\delta'}[X := 0]$ for an edge $l \overset{p, g_l(p), X}{\longrightarrow} l''$,
       $\tilde{p}(l'', \mathbf{u''}, \boldsymbol{\delta''}) = p(l'', X)$
  2. $(t, \tilde{p}) \in Steps_G^\varepsilon(l, \mathbf{u})$ if
     - $\exists \boldsymbol{\delta'} \in [-\varepsilon, \varepsilon]^n$ such that $\mathbf{u} + t - \boldsymbol{\delta} + \boldsymbol{\delta'} \models inv(l)$
     - for any $(l'', \mathbf{u''}, \boldsymbol{\delta''}) \in Q_G^\varepsilon$:
       $\tilde{p}(l'', \mathbf{u''}, \boldsymbol{\delta''}) = 1$ if
       $(l'', \mathbf{u''}, \boldsymbol{\delta''}) = (l, \mathbf{u} + t - \boldsymbol{\delta} + \boldsymbol{\delta'}, \boldsymbol{\delta'})$
       and $\tilde{p}(l'', \mathbf{u''}, \boldsymbol{\delta''}) = 0$ otherwise

A result stronger than Theorem 6.1 of the previous section may then be proven concerning the behaviour of the perturbed PTS under such a synchronization scheme.

Theorem 6.2. *Given a pTA G (having closed guards and invariants), let $\mathcal{M}$ (resp. $\mathcal{M}^\varepsilon$) denote the corresponding PTS (resp. perturbed PTS with perturbation ε due to resynchronization) as described in Section 6.2 (resp. Section 6.4). Let $\mathcal{A}_i$ (resp. $\mathcal{A}_i^\varepsilon$) denote the set of adversaries of $\mathcal{M}$ (resp. $\mathcal{M}^\varepsilon$) of history i defined over the corresponding sets of canonical initialized paths Π_i (resp. Π_i^ε) of length i. It then holds that* $\forall 0 < \varepsilon < 1 \;\; \forall i \in \mathbb{N} : \mathcal{A}_i \equiv \mathcal{A}_i^\varepsilon$

Proof. (Sketch) Similar to Theorem 6.1, induction over i should give
$\forall 0 < \varepsilon < 1 \;\; \forall i \in \mathbb{N} \;\; \forall A \in \mathcal{A}_i^\varepsilon \;\; \forall \pi \in \Pi_i^\varepsilon : A(\pi) = (t, p) \;\wedge\; last(\pi) \overset{t,p}{\rightarrow} (l, \mathbf{x}) \;\Rightarrow$
$\exists\; \pi' \in \Pi_i,\; A' \in \mathcal{A}_i,\; \mathbf{x'} \in Grid(\mathbf{x}) : A'(\pi') = (t', p') \;\wedge\; last(\pi') \overset{t',p'}{\rightarrow} (l, \mathbf{x'}) \;\wedge$
$p'(l, \mathbf{x'}) = p(l, \mathbf{x})$ The proof then follows from the above result in conjunction with Definitions 6.4 and 6.5. □

A consequence of Theorem 6.1 and Lemma 6.1 is that, for pTA having closed guards and invariants, and subject to perturbations that are controlled by a regular resynchronization scheme, the enhanced non-determinism induced by the perturbation again does not affect the available probabilistic choices, so long as the

synchronization scheme maintains the perturbation parameter ε strictly below 1, but irrespective of the computational history i. This in turn implies an equivalence between the probabilistic timed structures $\mathcal{M}$ and $\mathcal{M}^{\varepsilon}$, i.e., $\mathcal{M} \equiv \mathcal{M}^{\varepsilon}$, and leads to a stricter preservation of probabilistic reachability properties, as stated in the following corollary:

Corollary 6.2. *For a given pTA G and a perturbation parameter $0 < \varepsilon < 1$ controlled by a synchronization scheme, let $\mathcal{M}$ (resp. $\mathcal{M}^{\varepsilon}$) denote the corresponding PTS (resp. perturbed PTS). For a set of target states F, let $p^{max} = sup_{A \in \mathcal{A}}\ p^{A}_{q_0}(F)$ denote the maximum probability of reaching F under all possible adversarial resolutions of the underlying PTS, while $p^{\varepsilon} = sup_{A \in \mathcal{A}^{\varepsilon}}\ p^{A}_{q_0}(F)$ denotes the maximum probability of reaching F under all possible adversarial resolutions of the perturbed PTS corresponding to perturbation $0 < \varepsilon < 1$ controlled by the synchronization scheme. It then holds that:*

$$\forall 0 < \varepsilon < 1 : p^{\varepsilon} = p^{max}$$

Corollary 6.2 states a stronger result concerning the preservation of the maximum probabilistic reachability under clock drifts appropriately bounded by a synchronization scheme, with no restrictions now on the life-time of the system. As was the case with Corollary 6.1, Corollary 6.2 also applies for the computation of the minimum probability of reaching a target state under all possible adversial resolutions.

6.5 Conclusion

We considered the (minimum and maximum) probabilistic reachability properties for the well-studied formalism of Probabilistic Timed Automata under the realistic assumption of closed guards and invariants - closed and diagonal- free pTA have been shown to be very expressive in the modelling and analysis of large scale real-world applications such as the dynamic configuration protocol of IPv4 link-local addresses, the IEEE 802.11 medium access control protocol for wireless local area networks, and the IEEE 1394 FireWire root contention protocol [KNPS06]. We have attempted to render such analysis more realistic by considering the possible adversarial effects of perturbations in the form of drifting clocks that could occur in practice. We have attempted to show that such perturbations (and the resultant enhanced non-determinism) do not in fact affect the available probabilistic choices under the following realistic restrictions:

- The system has a finite life-time, such that for each computational history i, an appropriate small perturbation parameter ε_i can be chosen such that probabilistic reachability properties are preserved (cf. Theorem 6.1, Corollary 6.1).
- The system is subjected to a regular resynchronization scheme that always maintains a uniformly bounded relative drift ε between the clocks, leading again to the preservation of probabilistic reachability properties, but now for $0 < \varepsilon < 1$ irrespective of the systems life-time (cf. Theorem 6.2, Corollary 6.2).

These results may be viewed as a step towards enabling the (re-)usage of the existing rich machinery of algorithms and tools described in [KNSS02, KNPS06, KNSW07, KNP09] for the verification of a more realistic class of real-time system models that exhibit both non-deterministic and probabilistic behaviour. Subsequent to a full development of these results, future work includes their extensions to a richer class of properties, such as expected reachability properties [KNPS06] that are relevant for fault-tolerant performance measures such as the mean-time-(to/between)- failures of components.

7

Conclusion

We now summarize the results of the preceding chapters, together with some perspectives on classifying the various models presented in this dissertation within the unifying framework of *symbolic probabilistic systems* [KNS01].

1. Chapter 2 contributes to the quantitative axis of this dissertation, by demonstrating the robust safety for timed automata with closed guards and invariants against perturbations manifesting as drifts in the clocks. Our notions of robust safety required either (a) the allowable drift to be dependent on the number of iterations while increasing without bound with time-passage, yielding a strictly positive robust safety margin for systems having an arbitrary, but finite lifetime (cf. Section 2.3), or (b) the clocks to be regularly resynchronized such that extent of the drifts always remained bounded, yielding a robust safety margin of 1 irrespective of the extent of time passage (cf. Section 2.4).
2. Chapter 3 is a second contribution to the quantitative axis of this dissertation, with an investigation of the optimum reachability problem for Multi-Priced Timed Automata that admit both positive and negative costs on edges and locations, thus bridging the gap between the results of [BBBR07] and of [LR08]. We showed in Section 3.3 that even the location reachability problem is undecidable for MPTA equipped with both positive and negative costs, provided the costs are subject to a bounded budget, in the sense that paths of the underlying Multi-Priced Transition System (MPTS) that operationally exceed the budget are considered as not being viable. This undecidability result follows from an encoding of Stop-Watch Automata using such MPTA, and applies to MPTA with as few as two cost variables, and even when no costs are incurred upon taking edges. In Section 3.4, we then restricted the MPTA such that each viable quasi-cyclic path of the underlying MPTS incurs a minimum absolute cost. Under such a condition, the location reachability problem is shown to be decidable and the optimum cost is shown to be computable for MPTA with positive and negative costs and a bounded budget. These results follow from a reduction of the optimum reachability problem to the solution of a linear constraint system representing the path conditions over a finite number of viable paths of bounded length.
3. Chapter 4 is our main contribution to the structural axis of this dissertation. We investigate a series of transformations that aim at easier subsequent verifi-

cation of real-time systems with shared data variables, modelled as networks of extended timed automata (ETA). Our contributions to this end are the following: (1) We equip ETA with a library of composition operators, including one for layered composition, intermediate between parallel and sequential composition. Under certain non-interference and/or precedence conditions imposed on the structure of the ETA networks, the communication closed layer (CCL) laws and associated partial-order (po-) and (layered) reachability equivalences are shown to hold, cf. Sections 4.3 and 4.4. (2) Next, we investigate (under certain cycle conditions on the ETA) the (reachability preserving) transformations of separation and flattening aimed at reducing the number of cycles of the ETA, cf. Section 4.5. (3) We then show that our separation and flattening in Section 4.5 may be applied together with the CCL laws in Sections 4.3 and 4.4, in order to restructure ETA networks such that the verification of layered reachability properties is rendered easier. This interplay of the three structural transformations (separation, flattening, and layering) is demonstrated in Section 4.6 on an enhanced version of Fischers real-time mutual exclusion protocol for access to multiple critical sections.

4. Chapter 5 attempts to contribute to both the quantitative and structural axes of this dissertation, with an adaptation of the CCL concept to the formal reasoning of randomized distributed algorithms. We do so by enriching probabilistic automata with a layered composition operator, as in Chapter 4 for the non-randomized real-time setting. Layered composition is used in Section 5.3 to establish probabilistic counterparts of the CCL laws that exploit independence and/or precedence conditions between the constituent PA. The probabilistic CCL laws are then used to enable partial order (po-) equivalence when layered composition is replaced by sequential composition. The conditions for termination and the allowable adversarial resolutions of non-determinism, as well as the full class of properties preserved by our layered transformations in the probabilistic setting remain unclear. We have nonetheless demonstrated the feasibility of our layered transformations on the randomized mutual exclusion algorithm of [KR92], complementing an algebraic approach for analyzing this algorithm in [MGCM08]. The preservation of a large class of probabilistic temporal properties by our layered transformation of the algorithm of [KR92] has been confirmed by extensive experimental evaluation in [Sha15], with our transformation entailing a state-space reduction by a factor of 3. We speculate later in this chapter on the insights that might be drawn from our layered transformation of the algorithm of [KR92].
5. Chapter 6 attempts to contribute to the quantitative axis of this dissertation by exploring extensions of the results of Chapter 2 to the setting of probabilistic timed automata. While these probabilistic extensions appear to be relatively straightfoward as the discrete probability distributions in this setting annotate only the discrete transitions between locations (and are thus orthogonal to the clocks), the full (inductive) proofs of the main results in this chapter (Theorems 6.1 and 6.2) have not been presented. We have nonetheless included these preliminary results in this dissertation, as they very naturally extend the corresponding results for (closed) timed automata. Thus, as in Section 2.3, we show in Section 6.3 that limit of the max / min probability of reaching a target location in a closed probabilistic timed automaton remains unchanged for infinitesimally small perturbations that depend on the iteration depth. As in Section 2.4, we

show in Section 2.5 that that the max / min reachability probability again remains unchanged for perturbation parameters strictly less than 1 when the drifting clocks are subject to regular resynchronization.

The usage of "closedness" in Chapters 4 and 5 that contribute to the structural axis of this dissertation is in the setting of compositions (primarily in a parallel context) and refers to the (absence of the) interferences that might arise due to shared variables and / or synchronization actions. In Chapters 2, 3, and 6 that contribute to the quantitative axis of this dissertation, "closedness" refers to topological closure of the guards and invariants manifesting as non-strict constraints in the models considered.

Symbolic Probabilistic Systems

Each of the models considered in this dissertation is a strict subclass of *probabilistic hybrid automata* [Spr01], and each subclass may be classified as a *symbolic probabilistic system* (SPS) [KNS01]. We now attempt a classification scheme for SPS in order to provide a unifying scheme for the models considered in this dissertation, along the lines of a probabilistic counterpart to the classification scheme in [HMR05].

- The (closed) perturbed timed automata of Section 2.3 with (unbounded) drifting clocks characterized by slopes in the range $[\frac{1}{1+\varepsilon}, 1+\varepsilon]$ for a given perturbation parameter $\varepsilon > 0$ do not in general yield to a (time-abstract) bisimulation quotient on the state-space. Works that do give decision procedures for reachability in perturbed timed automata largely address the equivalent problem modelled by a perturbation parameter $\Delta > 0$ characterizing the guard enlargement, under the progress cycle and/or flatness assumptions [Pur00, WDMR08, DK06, JR11, KLMP14, San15]. As (closed) perturbed timed automata under a given perturbation $\varepsilon > 0$ characterizing the extent of the (rectangular) clock-drift appear to be a subclass of rectangular hybrid automata, we classify the perturbed timed automata of Section 2.3 in the class STS3, cf. Theorem 3C of [HMR05]. Note that the decision procedure implied by Theorem 2.1 and Corollary 2.1 involve perturbation parameters that depend on the depth of iteration.
- As the (closed) perturbed timed automata of Section 2.4 that are subject to regular clock resynchronization admit a decision procedure for every perturbation parameter ε strictly less than 1, such models may be classified in STS1, analogous to timed automata, cf. Theorem 1C of [HMR05].
- As the (bounded) Multi-Priced Timed Automata of Chapter 3 may be encoded as Stop-Watch Automata, thus yielding undecidability, while nonetheless admitting a decision procedure under the cost-charging assumption, and as the boundedness condition seems to play a crucial role in both cases, the classification of such models that lie at the fine (un-)decidability frontier between timed and linear hybrid automata remains unclear.
- In Chapter 3, we consider networks of extended timed automata in the context of a finite number of parallel compositions. As the data variables here have a finite range, they may be classified in STS1, analogous to timed automata, cf. Theorem 1C of [HMR05].
- For the probabilistic models of Chapters 5 and 6, we propose an analogous classification scheme for symbolic probabilistic systems. Thus the finite state PA

(and finite compositions thereof) fall into the class SPS1, as do the (closed) perturbed probabilistic timed automata of Section 6.4 subject to regular clock resynchronization for values of the perturbation parameter ε upto 1. The (closed) perturbed probabilistic timed automata of Section 6.3 that are subject to unbounded rectangular clock drifts fall however into the class SPS3.

References

ABBL03. L. Aceto, P. Bouyer, A. Burgueno, and K. Larsen. The power of reachability testing for timed automata. *Theoretical Computer Science*, 300:411–475, 2003.

ABG+14. S. Akshay, Benedikt Bollig, Paul Gastin, Madhavan Mukund, and K. Narayan Kumar. Distributed timed automata with independently evolving clocks. *Fundam. Inform.*, 130(4):377–407, 2014.

ABH+01. R. Alur, R. K. Brayton, T. A. Henzinger, S. Qadeer, and S. K. Rajamani. Partial-order reduction in symbolic state-space exploration. *FMSD*, 18:97–116, 2001.

AC08. Hagit Attiya and Keren Censor. Tight bounds for asynchronous randomized consensus. *J. ACM*, 55(5), 2008.

ACH+95. R. Alur, C. Courcoubetis, N. Halbwachs, T. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theor. Comput. Sci.*, 138(1):3–34, 1995.

AD94. R. Alur and D. Dill. A theory of timed automata. *TCS*, 126(2):183–235, 1994.

BBBR07. Patricia Bouyer, Thomas Brihaye, Véronique Bruyère, and Jean-François Raskin. On the optimal reachability problem of weighted timed automata. *Formal Methods in System Design*, 31(2):135–175, 2007.

BCC+03. Armin Biere, Alessandro Cimatti, Edmund M. Clarke, Ofer Strichman, and Yunshan Zhu. Bounded model checking. *Advances in Computers*, 58:117–148, 2003.

BDL04. G. Behrmann, A. David, and K. G. Larsen. A tutorial on Uppaal. In *Formal Methods for the Design of Real-Time Systems*, volume 3185 of *LNCS*, pages 200–236. Springer, 2004.

BFH+01. Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim Guldstrand Larsen, Paul Pettersson, Judi Romijn, and Frits W. Vaandrager. Minimum-cost reachability for priced timed automata. In M.-D. Di Benedetto and A. Sangiovanni-Vincentelli, editors, *HSCC*, volume 2034 of *LNCS*, pages 147–161. Springer, 2001.

BFL+08. Patricia Bouyer, Ulrich Fahrenberg, Kim Guldstrand Larsen, Nicolas Markey, and Jirí Srba. Infinite runs in weighted timed automata with energy constraints. In Franck Cassez and Claude Jard, editors,

FORMATS, volume 5215 of *Lecture Notes in Computer Science*, pages 33–47. Springer, 2008.

BGC04. C. Baier, M. Größer, and F. Ciesinski. Partial order reduction for probabilistic systems. In *Quantitative Evaluation of Systems (QEST)*, pages 230–239. IEEE CS Press, 2004.

BJLY98. J. Bengtsson, B. Jonsson, J. Lilius, and W. Yi. Partial order reductions for timed systems. In D. Sangiorgi and R. de Simone, editors, *CONCUR*, volume 1466 of *LNCS*, pages 485–500. Springer, 1998.

BLM14. Patricia Bouyer, Kim G. Larsen, and Nicolas Markey. Lower-bound-constrained runs in weighted timed automata. *Perform. Eval.*, 73:91–109, 2014.

BLR. G. Behrmann, K. G. Larsen, and J. I. Rasmussen. Optimal scheduling using priced timed automata. *SIGMETRICS Performance Evaluation Review*, 32.

BMR06. P. Bouyer, N. Markey, and P.-A. Reynier. Robust model-checking of linear-time properties in timed automata. In J. R. Correa, A. Hevia, and M. Kiwi, editors, *LATIN*, volume 3887 of *LNCS*, pages 238–249. Springer, 2006.

BMR08. P. Bouyer, N. Markey, and P.-A. Reynier. Robust analysis of timed automata via channel machines. In R. Armadio, editor, *FoSSaCS*, volume 4962 of *LNCS*, pages 157–171. Springer, 2008.

Boc79. G.v. Bochmann. Distributed synchronization and regularity. *Computer Networks*, 3:36–43, 1979.

Boc88. G.v. Bochmann. Delay-independent design for distributed systems. *IEEE Trans. Software Eng.*, 14(8):1229–1237, 1988.

Bou04. P. Bouyer. Forward analysis of updatable timed automata. *Formal Methods in System Design*, 24:281–320, 2004.

Bou09. P. Bouyer. *From Qualitative to Quantitative Analysis of Timed Systems*. 2009. Mémoire d'habilitation, Université Paris 7.

BP99. P. Bouyer and A. Petit. Decomposition and composition of timed automata. In J. Wiedermann, P. van Emde Boas, and M. Nielsen, editors, *ICALP*, volume 1644 of *LNCS*, pages 210–219. Springer, 1999.

BS00. S. Bornot and S. Sifakis. An algebraic framework for urgency. *Inf. Comput.*, 163:172–202, 2000.

BY04. J. Bengtsson and W. Yi. Timed automata: Semantics, algorithms, and tools. In *Lectures on Concurrency and Petri Nets*, volume 3098 of *LNCS*, pages 87–124. Springer, 2004.

CCK+08. Ran Canetti, Ling Cheung, Dilsun Kirli Kaynar, Moses Liskov, Nancy A. Lynch, Olivier Pereira, and Roberto Segala. Analyzing security protocols using time-bounded task-PIOAs. *Discrete Event Dynamic Systems*, 18(1):111–159, 2008.

CHR91. Zhou Chaochen, C. A. R. Hoare, and Anders P. Ravn. A calculus of durations. *Inf. Process. Lett.*, 40(5):269–276, 1991.

CJ99. H. Comon and Y. Jurski. Timed automata and the theory of real numbers. In J. C. M. Baeten and S. Mauw, editors, *CONCUR*, volume 1664 of *LNCS*, pages 242–257. Springer, 1999.

CL00. Franck Cassez and Kim Guldstrand Larsen. The impressive power of stopwatches. In Catuscia Palamidessi, editor, *CONCUR*, volume 1877 of *LNCS*, pages 138–152. Springer, 2000.

CM88. K. M. Chandy and J. Misra. *Parallel Program Design – A Foundation*. Addison Wesley, 1988.

Coh00. E. Cohen. Separation and reduction. In R. C. Backhouse and J. N. Oliveira, editors, *Mathematics of Program Construction*, volume 1837 of *LNCS*, pages 45–59. Springer, 2000.

CSCBM09. M. Chaouch-Saad, B. Charron-Bost, and S. Merz. A reduction theorem for the verification of round-based distributed algorithms. In O. Bournez and I. Potapov, editors, *Reachability Problems (RP)*, volume 5797 of *LNCS*, pages 93–106. Springer, 2009.

dAHM03. L. de Alfaro, T. A. Henzinger, and R. Majumdar. Discounting the future in systems theory. In J. C. M. Baeten, J. K. Lenstra, J. Parrow, and G. J. Woeginger, editors, *ICALP*, volume 2719 of *LNCS*, pages 1022–1037. Springer, 2003.

DHQ+08. J. S. Dong, P. Hao, S. Qin, J. Sun, and W. Yi. Timed automata patterns. *IEEE Trans. Software Eng.*, 34(6):844–859, 2008.

Dim07. Catalin Dima. Dynamical properties of timed automata revisited. In Jean-François Raskin and P. S. Thiagarajan, editors, *FORMATS*, volume 4763 of *Lecture Notes in Computer Science*, pages 130–146. Springer, 2007.

DK06. C. Daws and P. Kordy. Symbolic robustness analysis of timed automata. In *FORMATS*, volume 4202 of *LNCS*, pages 143–155. Springer, 2006.

DKFW10. K. Dräger, A. Kupriyanov, B. Finkbeiner, and H. Wehrheim. Slab: A certifying model checker for infinite-state concurrent systems. In J. Esparza and R. Majumdar, editors, *TACAS*, volume 6015 of *LNCS*, pages 271–274, 2010.

DN04. P. R. D'Argenio and P. Niebert. Partial order reduction on concurrent probabilistic programs. In *Quantitative Evaluation of Systems (QEST)*, pages 240–249. IEEE CS Press, 2004.

dWDMR04. M. de Wulf, L. Doyen, N. Markey, and J.-F. Raskin. Robustness and implementability of timed automata. In Y. Lakhnech and S. Yovine, editors, *FORMATS-FTRTFT*, volume 3253 of *LNCS*, pages 118–133. Springer, 2004.

EF82. T. Elrad and N. Francez. Decomposition of distributed programs into communication-closed layers. *Sci. Comput. Program.*, 2:155–173, 1982.

FH07. Martin Fränzle and Michael R. Hansen. Deciding an interval logic with accumulated durations. In Orna Grumberg and Michael Huth, editors, *Proceedings of the 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 4424 of *Lecture Notes in Computer Science*, pages 201–215. Springer, 2007.

GDF09. S. Giro, P. R. D'Argenio, and L. M. F. Fioriti. Partial order reduction for probabilistic systems: A revision for distributed schedulers. In M. Bravetti and G. Zavattaro, editors, *CONCUR*, volume 5710 of *LNCS*, pages 338–353. Springer, 2009.

GHJ97. V. Gupta, T. A. Henzinger, and R. Jagadeesan. Robust timed automata. In O. Maler, editor, *Hybrid and Real-Time Systems*, volume 1201 of *LNCS*, pages 331–345. Springer, 1997.

HKPV98. Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? *J. Comput. Syst. Sci.*, 57(1):94–124, 1998.

HMR05. T. Henzinger, R. Majumdar, and J.-F. Raskin. A classification of symbolic transition systems. *ACM Transactions on Computational Logic*, 6(1):1–32, 2005.

HP07. J. Haakansson and P. Pettersson. Partial order reduction for verification of real-time components. In J.-F. Raskin and P. S. Thiagarajan, editors, *FORMATS*, volume 4763 of *LNCS*, pages 211–226. Springer, 2007.

Jan94. W. Janssen. *Layered Design of Parallel Systems*. PhD thesis, U. Twente, 1994.

JPXZ94. W. Janssen, M. Poel, Q. Xu, and J. Zwiers. Layering of real-time distributed processes. In H. Langmaack, W. P. de Roever, and J. Vytopil, editors, *FTRTFT*, volume 863 of *LNCS*, pages 393–417. Springer, 1994.

JR11. Rémi Jaubert and Pierre-Alain Reynier. Quantitative robustness analysis of flat timed automata. In Martin Hofmann, editor, *FOSSACS*, volume 6604 of *Lecture Notes in Computer Science*, pages 229–244. Springer, 2011.

JZ92. Wil Janssen and Job Zwiers. From sequential layers to distributed processes: Deriving a distributed minimum weight spanning tree algorithm. In *Principles of Distributed Computing (PODC)*, pages 215–227. ACM Press, 1992.

KLMP14. Piotr Kordy, Rom Langerak, Sjouke Mauw, and Jan Willem Polderman. A symbolic algorithm for the analysis of robust timed automata. In Cliff B. Jones, Pekka Pihlajasaari, and Jun Sun, editors, *Formal Methods*, volume 8442 of *Lecture Notes in Computer Science*, pages 351–366. Springer, 2014.

KNP09. Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Stochastic games for verification of probabilistic timed automata. In *FORMATS*, pages 212–227, 2009.

KNPS06. Marta Z. Kwiatkowska, Gethin Norman, David Parker, and Jeremy Sproston. Performance analysis of probabilistic timed automata using digital clocks. *Formal Methods in System Design*, 29(1):33–78, 2006.

KNS01. M. Z. Kwiatkowska, G. Norman, and J. Sproston. Symbolic computation of maximal probabilistic reachability. In K. G. Larsen and M. Nielsen, editors, *CONCUR*, volume 2154 of *LNCS*, pages 169–183. Springer, 2001.

KNSS02. M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theoretical Computer Science*, 282:101–150, 2002.

KNSW07. Marta Z. Kwiatkowska, Gethin Norman, Jeremy Sproston, and Fuzhi Wang. Symbolic model checking for probabilistic timed automata. *Inf. Comput.*, 205(7):1027–1077, 2007.

Koe36. D. Koenig. *Theorie der Endlichen und Unendlichen Graphen: Kombinatorische Topologie der Streckenkomplexe*. Leipzig: Akad. Verlag, 1936.

KR92. E. Kushilevitz and M. O. Rabin. Randomized mutual exclusion algorithms revisited. In *PODC*, pages 275–283, 1992.

KvST12. J. P. Katoen, J. C. van de Pol, M. I. A. Stoelinga, and M. Timmer. A linear process-algebraic format with data for probabilistic automata. *Theor. Comput. Sci.*, 413(1):36–57, 2012.

LBB+01. Kim Guldstrand Larsen, Gerd Behrmann, Ed Brinksma, Ansgar Fehnker, Thomas Hune, Paul Pettersson, and Judi Romijn. As cheap as possible: Efficient cost-optimal reachability for priced timed automata. In Gérard Berry, Hubert Comon, and Alain Finkel, editors, *CAV*, volume 2102 of *LNCS*, pages 493–505. Springer, 2001.

LNZ05. D. Lugiez, P. Niebert, and S. Zennou. A partial order semantics approach to the clock explosion problem of timed automata. *Theor. Comput. Sci.*, 345:27–59, 2005.

LPY01. M. Lindahl, P. Pettersson, and W. Yi. Formal design and analysis of a gear controller. *Software Tools for Technology Transfer*, 3:353–368, 2001.

LR08. Kim Guldstrand Larsen and Jacob Illum Rasmussen. Optimal reachability for multi-priced timed automata. *Theor. Comput. Sci.*, 390(2-3):197–213, 2008.

LSW96. K. G. Larsen, B. Steffen, and C. Weise. Fischer's protocol revisited: A simple proof using modal constraints. In R. Alur, T. A. Henzinger, and E. D. Sontag, editors, *Hybrid Systems*, volume 1066 of *LNCS*, pages 604–615. Springer, 1996.

MGCM08. A. K. McIver, C. Gonzalia, E. Cohen, and C. C. Morgan. Using probabilistic Kleene algebra pKA for protocol verification. *J. Log. Algebr. Program.*, 76(1):90–111, 2008.

Mil89. R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.

Min99. M. Minea. Partial order reduction for model checking of timed automata. In J. C. M. Baeten and S. Mauw, editors, *CONCUR*, volume 1664 of *LNCS*, pages 431–436. Springer, 1999.

MR02. Yoram Moses and Sergio Rajsbaum. A layered analysis of consensus. *SIAM J. Comput.*, 31(4):989–1021, 2002.

MWF14. Ahmed Mahdi, Bernd Westphal, and Martin Fränzle. Transformations for compositional verification of assumption-commitment properties. In Joël Ouaknine, Igor Potapov, and James Worrell, editors, *Reachability Problems*, volume 8762 of *LNCS*, pages 216–229. Springer, 2014.

MWP12. M. Muniz, B. Westphal, and A. Podelski. Timed automata with disjoint activity. In M. Jurdzinski and D. Nickovic, editors, *FORMATS*, volume 7595 of *LNCS*, pages 188–203. Springer, 2012.

OD08. E.-R. Olderog and H. Dierks. *Real-time systems - formal specification and automatic verification.* Cambridge University Press, 2008.

OW03. J. Ouaknine and J. Worrell. Revisiting digitization, robustness, and decidability for timed automata. In *LICS*, pages 198–207. IEEE Computer Society, 2003.

PM09. H.-J. Peter and R. Mattmüller. Component-based abstraction refinement for timed controller synthesis. In *RTSS*, pages 364–374. IEEE Computer Society, 2009.

PSE03. Sebastian Panek, Olaf Stursberg, and Sebastian Engell. Optimization of timed automata models using mixed-integer programming. In Kim Guldstrand Larsen and Peter Niebert, editors, *FORMATS*, volume 2791 of *Lecture Notes in Computer Science*, pages 73–87. Springer, 2003.

PSL00. Anna Pogosyants, Roberto Segala, and Nancy A. Lynch. Verification of the randomized consensus algorithm of Aspnes and Herlihy: a case study. *Distributed Computing*, 13(3):155–186, 2000.

Pur00. A. Puri. Dynamical properties of timed automata. *Discrete Event Dynamic Systems*, 10:87–113, 2000.

Qua10. Karin Quaas. *Kleene-Schützenberger and Büchi Theorems for Weighted Timed Automata.* PhD thesis, University of Leipzig, 2010.

Rab82. M. O. Rabin. n-process mutual exclusion with bounded waiting by $4 \log n$ shared variables. *J. Comput. Syst. Sci.*, 25(1):66–75, 1982.

Sai92. I. Saias. Proving probabilistic correctness statements: the case of Rabin's algorithm for mutual exclusion. In *Principles of Distributed Computing (PODC)*, pages 263–274. ACM Press, 1992.

San13. Ocan Sankur. *Robustness in Timed Automata: Analysis, Synthesis, Implementation.* PhD thesis, ENS Cachan, 2013.

San15. Ocan Sankur. Symbolic quantitative robustness analysis of timed automata. In Christel Baier and Cesare Tinelli, editors, *TACAS*, volume 9035 of *Lecture Notes in Computer Science*, pages 484–498. Springer, 2015.

SdR94. F. A. Stomp and W.-P. de Roever. A principle for sequential reasoning about distributed algorithms. *Formal Asp. Comput.*, 6(6):716–737, 1994.

Seg95. R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems.* PhD thesis, Massachusetts Institute of Technology, 1995.

Seg00. Roberto Segala. Verification of randomized distributed algorithms. In Ed Brinksma, Holger Hermanns, and Joost-Pieter Katoen, editors, *Formal Methods and Performance Analysis*, volume 2090 of *LNCS*, pages 232–260. Springer, 2000.

Sha15. Arpit Sharma. *Reduction techniques for Nondeterministic and Probabilistic Systems.* PhD thesis, RWTH Aachen University, 2015.

SL95. R. Segala and N. Lynch. Probabilistic simulations for probabilistic processes. *Nordic J. Computing*, 2(2):250–273, 1995.

Spr01. Jeremy Sproston. *Model Checking for Probabilistic Timed and Hybrid Systems.* PhD thesis, University of Birmingham, 2001.

Sto02. Mariëlle Stoelinga. An introduction to probabilistic automata. *Bulletin of the EATCS*, 78:176–198, 2002.

SV99. Mariëlle Stoelinga and Frits W. Vaandrager. Root contention in IEEE 1394. In Joost-Pieter Katoen, editor, *AMAST Workshop on Real-Time and Probabilistic Systems (ARTS)*, volume 1601 of *LNCS*, pages 53–74. Springer, 1999.

TSvdP11. Mark Timmer, Mariëlle Stoelinga, and Jaco van de Pol. Confluence reduction for probabilistic systems. In P. A. Abdulla and K. R. M. Leino, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 6605 of *LNCS*, pages 311–325. Springer, 2011.

Č92. K. Čerāns. *Algorithmic Problems in Analysis of Real Time System Specifications.* PhD thesis, University of Latvia, 1992.

WDMR08. M De Wulf, L. Doyen, N. Markey, and J.-F Raskin. Robust safety of timed automata. *Formal Methods in System Design*, 33:45–84, 2008.

Yov97. S. Yovine. KRONOS: A verification tool for real-time systems. *International Journal on Software Tools for Technology Transfer*, 1(1-2):123–133, 1997.

List of Publications

1. M. Swaminathan and M. Fränzle, "A Symbolic Decision Procedure for Robust Safety of Timed Systems", 14^{th} *Intl. Symposium on Temporal Representation and Reasoning* (TIME'07), p. 192, IEEE Computer Society (2007).
2. M. Swaminathan, M. Fränzle, and J.-P. Katoen, "The Surprising Robustness of (Closed) Timed Automata against Clock-Drift", 5^{th} *IFIP Intl. Conf. on Theoretical Computer Science* (IFIP TCS'08), Vol. 273, *Intl. Federation for Information Processing*, pp. 537–553, Springer (2008).
3. M. Fränzle and M. Swaminathan, "Revisiting Decidability & Optimum Reachability for Multipriced Timed Automata", 7^{th} *Intl. Conf. on Formal Modeling and Analysis of Timed Systems* (FORMATS'09), Vol. 5813, *Lecture Notes in Computer Science*, pp. 149–163, Springer (2009).
4. E.-R. Olderog and M. Swaminathan, "Layered Composition for Timed Automata", 8^{th} *Intl. Conf. on Formal Modeling and Analysis of Timed Systems* (FORMATS'10), Vol. 6246, *Lecture Notes in Computer Science*, pp. 228–242, Springer (2010).
5. M. Swaminathan, J.-P. Katoen, and E.-R. Olderog, "Layered Reasoning for Randomized Distributed Algorithms", *Formal Aspects of Computing*, Vol. 24, Nr. 4–6, pp. 477–496, Springer (2012).
6. E.-R. Olderog and M. Swaminathan, "Structural Transformations for Data-Enriched Real-Time Systems", 10^{th} *Intl. Conf. on Integrated Formal Methods* (iFM'13), Vol. 7940, *Lecture Notes in Computer Science*, pp. 378–393, Springer (2013).
7. E.-R. Olderog and M. Swaminathan, "Structural Transformations for Data-Enriched Real-Time Systems", *Formal Aspects of Computing*, Vol. 27. Nr. 4, pp. 727–750, Springer (2015).