Distributed Synthesis in Continuous Time

Holger Hermanns[1], Jan Krčál[1], and Steen Vester[2(✉)]

[1] Computer Science, Saarland University, Saarbrücken, Germany
{hermanns,krcal}@cs.uni-saarland.de
[2] Technical University of Denmark, Kongens Lyngby, Denmark
stve@dtu.dk

Abstract. We introduce a formalism modelling communication of distributed agents strictly in *continuous-time*. Within this framework, we study the problem of synthesising local strategies for individual agents such that a specified set of goal states is reached, or reached with at least a given probability. The flow of time is modelled explicitly based on continuous-time randomness, with two natural implications: First, the non-determinism stemming from interleaving disappears. Second, when we restrict to a subclass of *non-urgent* models, the quantitative value problem for two players can be solved in EXPTIME. Indeed, the explicit continuous time enables players to communicate their states by delaying synchronisation (which is unrestricted for non-urgent models). In general, the problems are undecidable already for two players in the quantitative case and three players in the qualitative case. The qualitative undecidability is shown by a reduction to decentralized POMDPs for which we provide the strongest (and rather surprising) undecidability result so far.

1 Introduction

Distributed self-organising and self-maintaining systems are posing interesting design challenges, and have been subject to many practical [33] as well as theoretical [28,29] investigations. Distributed systems interact in real time, and one very general way to reason about their timing behaviour is to assume that arbitrary continuous probability distributions govern the timing of local steps as well as of communication steps. We are interested in how foundational properties of such distributed systems differ from models where timing is abstracted. As principal means of communication we consider symmetric handshake communication, since it can embed other forms of communication faithfully [2,24] including asynchronous and input/output-separated communication.

As an example, consider the problem of leaking a secret from a sandboxed malware to an attacker. The behaviour of attacker and malware (and possibly other components) are prescribed in terms of states, private transitions, labelled synchronisation transitions, and delay transitions which model both local computation times and synchronisation times. The delays are governed by arbitrary continuous probability distributions over real time. Handshake synchronisation is assumed to take place if all devices able to do so agree on the same transition

B. Jacobs and C. Löding (Eds.): FOSSACS 2016, LNCS 9634, pp. 353–369, 2016.
DOI: 10.1007/978-3-662-49630-5_21

label. Otherwise the components run fully asynchronously. The sandboxing can be thought of as restricting the set of labels allowed to occur on synchronisation transitions. The question we focus on is how to synthesise the component control strategies for malware and attacker so that they reach their target (of leaking the secret) almost surely or with at least a given probability p.

More precisely, we consider a parallel composition of n modules synchronizing via handshake communication. The modules are modelled by interactive Markov chains (IMCs) [16,18], a generalization of labelled transition systems and of continuous time Markov chains, equipped with a well-understood compositional theory. Each module may in each state enable private actions, as well as synchronisation actions. It is natural to view such a *distributed IMC* as a game with $n+1$ players, where the last player controls the interleaving of the modules. Each of the other n players controls the decisions in a single module, only based on its local timed history containing only transitions that have occurred within the module. On entering a state of its module, each player selects and commits to executing one of the actions enabled. A private action is executed immediately while a synchronisation action requires a CSP-style handshake [6], it is executed once all modules able to perform this action have committed to it.

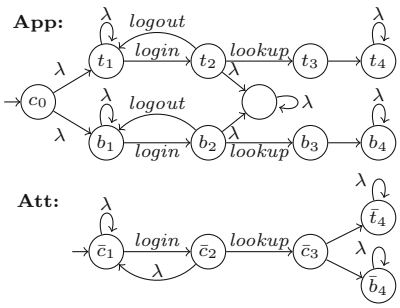
For representing delay distributions, we make one decisive and one technical restriction. First, we assume that each distribution is continuous. This for instance disallows deterministic delays of, say, 3 time units. It is an important simplification assumed along our explorations of continuous-time distributed control. Second, we restrict to exponential distributions. This is a pure technicality, since (a) our results can be developed with general continuous distributions, at the price of excessive notational and technical overhead, and (b) exponential distributions can approximate arbitrary continuous distributions arbitrarily close [25]. Together, these assumptions enable us to work in a setting close to interactive Markov chains.

Apart from running in continuous time, the concepts behind distributed IMCs are rather common. Closely related are models based on probabilistic automata [32] or (partially observable) Markov decision processes [3,26]. In these settings, the power of the interleaving player $n+1$ is a matter of ongoing debate [7,8,27]. The problem is that without additional (and often complicated) assumptions this player is too powerful to be realistic, and can for instance leak information between the other players. This is a problem, e.g. in the security context, making model checking results overly pessimistic [13].

In sharp contrast to the discrete-time settings, in our distributed IMCs the interleaving player $n+1$ does not have decisive influence on the resulting game. The reason is that the interleaving player can only affect the order of transitions between two delays, but neither *which transitions* are taken nor what the different players *observe*. This is rooted in the common alphabet synchronisation and especially the continuous-time nature of the game: the probability of two local modules changing state at the same time instant is zero, except if synchronising.

Example 1. We consider the model displayed on the right where the delay transitions are labelled by some rate λ. It displays a very simplistic malicious *App*

trying to communicate a secret to an outside *Att*acker, despite being sandboxed. Innocently looking action *login*, *logout* and *lookup* synchronise *App* and *Att*, while the unlabelled transitions denote some private actions of the respective module.



Initially, the *App* can only let time pass. The *Att*acker player has no other choice than committing to handshaking on action *login*. A race of the delay transitions will occur that at some point will lead to either state $(t_1, \bar{c}_1)$ or $(b_1, \bar{c}_1)$, with equal probability. Say in $(t_1, \bar{c}_1)$, the *App* player can only commit to action *login*. The synchronisation will happen immediately since the *Att*acker is committed to *login* already, leading to $(t_2, \bar{c}_2)$. Now the *App* player has either to commit to action *lookup* or *logout*. The latter will induce a deadlock due to a mismatch in players' commitments. Instead assuming the earlier, the state synchronously and immediately changes to $(t_3, \bar{c}_3)$. The *Att*acker player can now use its local timed history to decide which of the private actions to pick. Whatever it chooses, an interleaving of private actions of the two modules follows in zero time. Unless the reachability condition considers transient states such as $(t_3, \bar{t}_4)$ where no time is spent, the player resolving the interleaving has no influence on the outcome.

Now, assume the reachability condition is the state set $\{(t_4, \bar{t}_4), (b_4, \bar{b}_4)\}$. This corresponds to the *Att*acker player correctly determining the initial race of the *App*, and can be considered as a leaked secret. However, according to the explanations provided, it should be obvious that the probability of guessing correctly (by committing properly in state $\bar{c}_3$) is no larger than 0.5, just because the players are bound to decide only based on the local history. The crucial question is: is there an algorithm to compute such probabilities, in general?

Our Contribution. This paper is the first to explore distributed cooperative reachability games with continuous-time flow modelled explicitly. The formalism we study is based on interactive Markov chains, which in turn has been applied across a wide range of engineering domains. We aim at synthesising local strategies for the players to reach with at least a given probability a specified set of goal states. If this probability is 1 we call the problem *qualitative*, otherwise *quantitative*. We consider *existential* problems, asking for the existence of strategies with these properties, and *value* problems, asking for strategies approximating the given probability value arbitrarily closely. We have three main results:

1. We show that, under mild assumptions on the winning condition, in continuous-time distributed synthesis the interleaving player has no power.
2. In general, we establish that the quantitative problems are undecidable for two or more players, the qualitative value problem is undecidable for two or more players and the qualitative existence problem is EXPTIME-hard for two players and undecidable for three or more players.

3. However, when focusing on the subclass of 2-player *non-urgent* distributed IMCs, the quantitative value problem can be solved in exponential time. Non-urgency enables changing the decisions committed to after some time. Thus, it empowers the players to reach a distributed consensus about the next handshake to perform by observing the only information they jointly have access to: the advance of time.

The qualitative undecidability comes from a novel result about decentralised partially observable Markov decision processes (DEC-POMDP), a multi-player extensions of POMDP. While qualitative existence is decidable for POMDP [1], we show that qualitative existence is undecidable for DEC-POMDP already for 2 players. It is to the knowledge of the authors the strongest undecidability result for DEC-POMDPs with infinite horizon which is of its own interest. By a reduction from DEC-POMDP to distributed IMCs that adds one player, we get undecidability of qualitative existence for 3 or more players in distributed IMCs. Due to space constraints, full proofs can be found in [17].

2 Distributed Interactive Markov Chains

We denote by $\mathbb{R}$, $\mathbb{R}_{\geq 0}$, $\mathbb{N}$, and $\mathbb{N}_0$ the sets of real numbers, non-negative real numbers, positive integers, and non-negative integers, respectively. Furthermore, for a finite set X, we denote by $\Delta(X)$ the set of *discrete probability distributions* over X, i.e. functions $f : X \to [0,1]$ such that $\sum_{x \in X} f(x) = 1$. Finally, for a tuple $\boldsymbol{x}$ from a product space $X_1 \times \cdots \times X_n$ and for $1 \leq i \leq n$, we use functional notation $\boldsymbol{x}(i)$ to denote the ith element of the tuple.

We first give a definition of a (local) module based on the formalism of Interactive Markov Chains (IMC). Then we introduce (global) distributed IMC.

Definition 1 ***(IMC).*** *An* IMC *(module) is a tuple* $(S, Act, \hookrightarrow, \rightsquigarrow, s^{in})$ *where*

- S *is a finite set of states with an initial state* s^{in}*,*
- Act *is a finite set of actions,*
- $\hookrightarrow \subseteq S \times Act \times S$ *is the* action *transition relation,*
- $\rightsquigarrow \subseteq S \times \mathbb{Q}_{>0} \times S$ *is the finite* delay *transition relation.*

We write $s \overset{a}{\hookrightarrow} s'$ when $(s, a, s') \in \hookrightarrow$ and $s \overset{\lambda}{\rightsquigarrow} s'$ when $(s, \lambda, s') \in \rightsquigarrow$ (λ being the *rate* of the transition). We say that action a is *available* in s if $s \overset{a}{\hookrightarrow} s'$ for some s'.

Definition 2 ***Distributed IMC.*** *A* distributed IMC *is a tuple*

$$\mathcal{G} = ((S_i, Act_i, \hookrightarrow_i, \rightsquigarrow_i, s_i^{in}))_{1 \leq i \leq n}$$

of modules for players $Plr = \{1, ..., n\}$*. Furthermore, by* $Act = \bigcup_i Act_i$ *we denote the set of all actions, and by* $S = S_1 \times ... \times S_n$ *the set of (global) states.*

Intuitively, a distributed IMC moves in continuous-time from a (global) state to a (global) state using transitions with labels from $Label = Act \cup Plr$:

- An action transition with label $a \in Act$ corresponds to *synchronous* communication of all players in $\mathrm{Sync}(a) := \{j \in Plr \mid a \in Act_j\}$ and can only be taken when it is *enabled*, i.e. when all these players choose their local transitions with action a at the same time. It is called a *synchronisation* action if $|\mathrm{Sync}(a)| \geq 2$ and a *private* action if $|\mathrm{Sync}(a)| = 1$.
- A delay transition of any player $j \in Plr$ is taken *independently* by the player after a random delay, i.e. the set of players that synchronise over label j is $\mathrm{Sync}(j) = \{j\}$.

Formally, the (local) *choices of player* j range over $\mathcal{C}_j = \hookrightarrow_j \cup \{\bot\}$. When in (local) state s, the player may pick only a choice *available in* s. That is, either an action transition of the form $s \overset{a}{\hookrightarrow} s'$ or $\bot$ if there is no such action transition. We define *global choices* as $\mathcal{C} = \mathcal{C}_1 \times \cdots \times \mathcal{C}_n$. A global choice $\boldsymbol{c}$ induces the set $\mathrm{En}(\boldsymbol{c}) = \{a \in Act \mid \forall j \in \mathrm{Sync}(a) : \boldsymbol{c}(j) = (\cdot, a, \cdot)\}$ of actions *enabled* in $\boldsymbol{c}$.

To avoid that time stops by taking infinitely many action steps in zero time, we pose a standard assumption prohibiting cycles [14,15,19–21]: we require that for every action $a \in Act$ there is a player $j \in \mathrm{Sync}(a)$ such that the labelled transition system $(S_j, \hookrightarrow_j)$ does not have any cycle involving action a.

The behaviour of a distributed IMC is a *play*, an infinite sequence

$$\rho = \boldsymbol{s_0 c_0} \xrightarrow{a_1,t_1} \boldsymbol{s_1 c_1} \xrightarrow{a_2,t_2} \boldsymbol{s_2 c_2} \cdots$$

where each $\boldsymbol{s_i} \in S$ is the state after i moves, $\boldsymbol{c_i} \in \mathcal{C}$ is the choice of the players in the state $\boldsymbol{s_i}$, and $a_{i+1} \in Label$ and $t_{i+1} \in \mathbb{R}_{\geq 0}$ are the label and the absolute time of the next transition taken. By *Play* we denote the set of all plays. Which play is taken depends on the *strategies* of the players, on the *scheduler* which resolves interleaving of communication whenever multiple actions are enabled, and on the rules (involving randomness) given later.

2.1 Schedulers and Strategies

First we define strategies and schedulers basing their decision on the current local and global history, respectively. A *(global) history* is a finite prefix

$$h = \boldsymbol{s_0 c_0} \xrightarrow{a_1,t_1} \cdots \xrightarrow{a_i,t_i} \boldsymbol{s_i}$$

of a play ending with a state. For given h, we get the *local history* of player j as

$$\pi_j(h) = \boldsymbol{s'_0}(j)\boldsymbol{c'_0}(j) \xrightarrow{a'_1,t'_1} \cdots \xrightarrow{a'_\ell,t'_\ell} \boldsymbol{s'_\ell}(j)$$

where $vects'_0\boldsymbol{c'_0} \xrightarrow{a'_1,t'_1} \cdots \xrightarrow{a'_\ell,t'_\ell} \boldsymbol{s'_\ell}$ is the subsequence of h omitting all steps not visible for player j, i.e. all $\xrightarrow{a_m,t_m} \boldsymbol{s_m c_m}$ with $j \notin \mathrm{Sync}(a_m)$. The set of all global histories is denoted by *Hist*; the set of local histories of player j by $Hist_j$.

Example 2. Consider again Example 1. Let *App* be controlled by player 1 and *Att* by player 2. For the following history we get corresponding local histories

$$h = (c_0, \bar{c}_1)(\bot, login) \xrightarrow{1,0.42} (t_1, \bar{c}_1)(login, login) \xrightarrow{login,0.42} (t_2, \bar{c}_2),$$

$$\pi_1(h) = c_0 \ \bot \xrightarrow{1,0.42} t_1 \ login \xrightarrow{login,0.42} t_2, \qquad \pi_2(h) = \bar{c}_1 \ login \xrightarrow{login,0.42} \bar{c}_2$$

Note that the attacker can neither observe the Markovian transition nor the local state of the *App*. The *App* cannot observe the local state of the attacker either, but it can be deduced from the local history of the *App*.

A *strategy* for player j is a measurable function $\sigma : \mathit{Hist}_j \to \Delta(\mathcal{C}_j)$ that assigns to any local history h of player j a probability distribution over choices available in the last state of h. We say that a strategy σ for player j is *pure* if for all h we have $\sigma(h)(c) = 1$ for some c; and *memoryless* if for all h and h' with equal last local state we have $\sigma(h) = \sigma(h')$.

A *scheduler* is a measurable function $\delta : \mathit{Hist} \times \mathcal{C} \to \Delta(\mathit{Act}) \cup \{\bot\}$ that assigns to any global history h and global choice $\boldsymbol{c}$ a probability distribution over actions enabled in $\boldsymbol{c}$; or a special symbol $\bot$ again denoting that no action is enabled.

Example 3. The available local choices in $(t_2, \bar{c}_2)$, the last state of h from above, are $\{(t_2, \mathit{lookup}, t_3), (t_2, \mathit{logout}, t_1)\}$ for *App* and solely $\{(\bar{c}_2, \mathit{lookup}, \bar{c}_3)\}$ for *Att*. Let the strategy of *App* select either choice with equal probability. If $(t_2, \mathit{lookup}, t_3)$ is chosen, *lookup* is enabled and must be picked by the scheduler σ. If $(t_2, \mathit{logout}, t_1)$ is chosen, no action is enabled and δ must pick $\bot$, waiting for a delay transition.

2.2 Probability of Plays

Let us fix a *profile of strategies* $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n)$ for individual players, and a scheduler δ. The play starts in the initial state $\boldsymbol{s_0} = (s_1^{in}, \ldots, s_n^{in})$ and inductively evolves as follows. Let the current history be $h = \boldsymbol{s_0 c_0} \xrightarrow{a_1,t_1} \cdots \xrightarrow{a_i,t_i} \boldsymbol{s_i}$.

- For the next choice $\boldsymbol{c_i}$, only players $P_i := \mathrm{Sync}(a_i)$ involved in the last transition freely choose (we assume $P_0 := \mathit{Plr}$). Hence, independently for every $j \in P_i$, the choice $\boldsymbol{c_i}(j)$ is taken randomly according to $\sigma_j(\pi_j(h))$. All remaining players $j \notin P_i$ stick to the previous choice $\boldsymbol{c_i}(j) = \boldsymbol{c_{i-1}}(j)$ as for them, no observable event happened.
- After fixing $\boldsymbol{c_i}$, there are two types of transitions:
  1. If $\mathrm{En}(\boldsymbol{c_i}) \neq \emptyset$, the next synchronization action $a_{i+1} \in \mathrm{En}(\boldsymbol{c_i})$ is chosen randomly according to $\delta(h, \boldsymbol{c_i})$ and taken immediately at time $t_{i+1} := t_i$. The next state $\boldsymbol{s_{i+1}}$ satisfies for every $j \in \mathit{Plr}$:

$$\boldsymbol{s_{i+1}}(j) = \begin{cases} \mathrm{target}(\boldsymbol{c_i}(j)) & \text{if } j \in \mathrm{Sync}(a_{i+1}), \\ \boldsymbol{s_i}(j) & \text{if } j \notin \mathrm{Sync}(a_{i+1}). \end{cases}$$

     where $\mathrm{target}(\boldsymbol{c_i}(j))$ denotes the target of the transition chosen by player j. In other words, players involved in synchronisation move according to their choice, the remaining players stay in their previous states.
  2. If $\mathrm{En}(\boldsymbol{c_i}) = \emptyset$, a local delay transition is taken after a random delay, chosen as follows. Each delay transition $\boldsymbol{s_i}(j) \overset{\lambda}{\rightsquigarrow} \cdot$ outgoing from the current local state of any player j is assigned randomly a real-valued delay according to the exponential distribution with rate λ. This results in a collection of real numbers. The transition $\boldsymbol{s_i}(\ell) \overset{\lambda}{\rightsquigarrow} s$ with the minimum delay d in

this collection is taken. Hence, $a_{i+1} := \ell$ (denoting that player ℓ moves), $t_{i+1} := t_i + d$, and the next state $\boldsymbol{s_{i+1}}$ satisfies for every $j \in Plr$:

$$\boldsymbol{s_{i+1}}(j) = \begin{cases} s & \text{if } j \in \mathrm{Sync}(a_{i+1}) = \{\ell\}, \\ \boldsymbol{s_i}(j) & \text{if } j \notin \mathrm{Sync}(a_{i+1}). \end{cases}$$

All these rules induce a probability measure $\Pr^{\boldsymbol{\sigma},\delta}$ over the set of all plays by a standard cylinder construction.

2.3 Distributed Synthesis Problem

We study the following two fundamental reachability problems for distributed IMCs. Let $\mathcal{G}$ be a distributed IMC, $T \subseteq S$ be a target set of states, and p be a rational number in $[0, 1]$. Denoting by $\Diamond T$ the set of plays ρ that reach a state in T and stay there for a non-zero amount of time, we focus on:

Existence Does there exist a strategy profile $\boldsymbol{\sigma}$ s.t. for all schedulers δ,

$$\Pr^{\boldsymbol{\sigma},\delta}(\Diamond T) \geq p \ ?$$

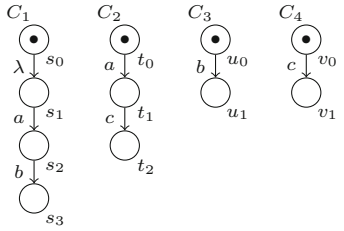
Value Can the value p be arbitrarily approached, i.e. do we have

$$\sup_{\boldsymbol{\sigma}} \inf_{\delta} \Pr^{\boldsymbol{\sigma},\delta}(\Diamond T) \geq p \ ?$$

We refer to the general problem with $p \in [0, 1]$ as *quantitative*. When we restrict to $p = 1$, we call the problem *qualitative*.

3 Schedulers Are Not that Powerful

The task of a scheduler is to choose among concurrently enabled transitions, thereby resolving the non-determinism conceptually caused by interleaving. In this section, we address the impact of the decisions of the scheduler. We show that despite having the ability to affect the order in which transitions are taken in the *global* play, the scheduler cannot affect what every player observes *locally*. Thus, the scheduler affects neither the choices of any player nor what synchronisation occurs. As a result, for winning objectives that are closed under *local observation equivalence*, the scheduler cannot affect the probability of winning.



Example 4. Consider the distributed IMC to the right. After the delay transition is taken in C_1 and there is synchronisation on action a, the scheduler can choose whether there will be synchronisation on b or c first. However, it can only affect the interleaving, not any of the local plays.

For a play $\rho = \boldsymbol{s_0 c_0} \xrightarrow{a_1,t_1} \boldsymbol{s_1 c_1} \xrightarrow{a_2,t_2} \boldsymbol{s_2 c_2} \cdots$ we define the local play $\pi_j(\rho)$ of player j analogously to local histories. We define *local observation* equivalence $\sim$ over plays by setting $\rho \sim \rho'$ if $\pi_j(\rho) = \pi_j(\rho')$ for all $j \in Plr$. Let us stress that two local observation equivalent plays have exactly the same action and delay transitions happening at the same moments of time; only the order of action transitions happening at the same time can differ. Finally, we say that a set E of plays is *closed under local observation equivalence* if for any $\rho \in E$ and any ρ' such that $\rho \sim \rho'$ we have $\rho' \in E$. It is now possible to show the following.

Theorem 1. *Let E be a measurable set of plays closed under local observation equivalence. For any strategy profile $\boldsymbol{\sigma}$ and schedulers δ and δ' we have*

$$\Pr^{\boldsymbol{\sigma},\delta}(E) = \Pr^{\boldsymbol{\sigma},\delta'}(E).$$

As a result, for the rest of the paper we write $\Pr^{\boldsymbol{\sigma}}(E)$ instead of $\Pr^{\boldsymbol{\sigma},\delta}(E)$ since the scheduler cannot affect the probability of the events we consider. Indeed, the reachability objectives defined in the previous section are closed under local observation equivalence.

Remark 1. The fact that interleaving does not have decisive impact in continuous time may seem natural and thus possibly unsurprising to experts. Yet, the result does not hold for many small variations of the setting we consider, e.g. neither for asymmetric communication nor when allowing cycles of action transitions.

4 Undecidability Results

In this section, we put distributed IMCs into context of other partial-observation models. As a result, we show that reachability quickly gets undecidable here.

Theorem 2. *For distributed IMCs we have that*

1. *the qualitative value, quantitative value, and quantitative existence problems are undecidable with $n \geq 2$ players; and*
2. *the qualitative existence problem is* ExpTime*-hard with $n = 2$ players and undecidable with $n \geq 3$ players.*

Theorem 2 is obtained by using two fundamental results. First, we provide a novel (and somewhat surprising) result for *decentralized POMDPs* (DEC-POMDPs) [3], an established multi-player generalization of POMDPs. We show that the qualitative existence problem for DEC-POMDPs is undecidable already for 2 players. This is, to the knowledge of the authors, currently the strongest known undecidability result for DEC-POMDPs. Second, we show that distributed IMCs are not only more expressive (w.r.t. reachability) than POMDPs but also more expressive than DEC-POMDPs. We show it by reducing reachability in DEC-POMDPs with n players to reachability in distributed IMCs with $n+1$ players. Theorem 2 follows from these two results and from known results about POMDPs [4,12,26]. For an overview, see Table 1.

Table 1. Undecidability results for reachability. Unreferenced results are shown here.

	POMDPs	DEC-POMDPs	Distributed IMCs
Qual. Existence	Dec. [1]	Undec. for ≥ 2 players	Undec. for ≥ 3 players
Value	Undec. [12]	Undec. for ≥ 1 player [12]	Undec. for ≥ 2 players
Quant. Existence	Undec. [26]	Undec. for ≥ 1 player [26]	Undec. for ≥ 2 players
Value	Undec. [4]	Undec. for ≥ 1 player [4]	Undec. for ≥ 2 players

4.1 Decentralized POMDP (DEC-POMDP)

We start with a definition of the related formalism of decentralized POMDP [3].

Definition 3. *A DEC-POMDP is a tuple* $(S, Plr, (Act_i, \mathcal{O}_i)_{1 \leq i \leq n}, P, O, s^{in})$ *where*

- S *is a finite set of global states with initial state* $s^{in} \in S$,
- $Plr = \{1, ..., n\}$ *is a finite set of players,*
- Act_i *is a finite set of local actions of player* i *with* $Act_i \cap Act_j = \emptyset$ *if* $j \neq i$, *(by* $Act = Act_1 \times \cdots \times Act_n$ *we denote the set of global actions),*
- $\mathcal{O}_i$ *is a finite set of local* observations *for player* i, *(by* $\mathcal{O} = \mathcal{O}_1 \times \cdots \times \mathcal{O}_n$ *we denote the set of global observations),*
- $P : S \times Act \to \Delta(S)$ *is the transition function which assigns to a state and a global action a probability distribution over successor states, and*
- $O : S \times Act \times S \to \Delta(\mathcal{O})$ *is the* observation function *which assigns to every transition a probability distribution over global observations.*

In contrast to distributed IMCs that capture flow of time explicitly, DEC-POMDP is a discrete-time formalism. A DEC-POMDP starts in the initial state s^{in}. Assuming that the current state is s, one discrete step of the process works as follows. First, each player j chooses an action a_j. Then the next state s' is chosen according to the probability distribution $P(s, \boldsymbol{a})$ where $\boldsymbol{a} = (a_1, \ldots, a_n)$. Then, each player j receives an observation $o_j \in \mathcal{O}_j$ such that the observations $\boldsymbol{o} = (o_1, ..., o_n)$ are chosen with probability $O(s, \boldsymbol{a}, s')(\boldsymbol{o})$. Repeating this forever, we obtain a *play* which is an infinite sequence $\rho = s_0 \boldsymbol{a_0} \boldsymbol{o_0} s_1 \boldsymbol{a_1} \boldsymbol{o_1} \cdots$ where $s_0 = s^{in}$ and for all $i \geq 0$ it holds that $s_i \in S$, $\boldsymbol{a_i} \in Act$, and $\boldsymbol{o_i} \in \mathcal{O}$. Note that the players can only base their decisions on the sequences of observations they receive rather than the actual sequence of states which is not available to them. For a more complete coverage of DEC-POMDPs, see [3].

4.2 Reduction from DEC-POMDP

First we present the reduction from a DEC-POMDP $\mathcal{P}$ to a distributed IMC $\mathcal{G}$. In this subsection, we write $\Pr^{\boldsymbol{\sigma}}_{\mathcal{P}}$ or $\Pr^{\boldsymbol{\sigma}}_{\mathcal{G}}$ instead of $\Pr^{\boldsymbol{\sigma}}$ to distinguish between the probability measure in the DEC-POMDP from the probability measure in the distributed IMC.

Proposition 1. *For a DEC-POMDP $\mathcal{P}$ with n players and a target set T of states of $\mathcal{P}$ we can construct in polynomial time a distributed IMC $\mathcal{G}$ with $n+1$ players and a target set T' of global states in $\mathcal{G}$ where:*

$$\exists \boldsymbol{\sigma} : \Pr_{\mathcal{G}}^{\boldsymbol{\sigma}}(\diamond T) = p \iff \exists \boldsymbol{\sigma'} : \Pr_{\mathcal{P}}^{\boldsymbol{\sigma'}}(\diamond T') = p.$$

Proof (Proof Sketch). Let us fix n and $\mathcal{P} = (S, Plr, (Act_i)_{1 \le i \le n}, \delta, (\mathcal{O}_i)_{1 \le i \le n}, O)$ where $Plr = \{1, ..., n\}$. Further, let $Act_i = \{a_{i1}, ..., a_{im_i}\}$ and $\mathcal{O}_i = \{o_{i1}, ..., o_{i\ell_i}\}$ for player $i \in Plr$. The distributed IMC $\mathcal{G}$ has $n+1$ modules, one module for each player in $\mathcal{P}$ and the *main* module responsible for their synchronisation. Intuitively,

- the module of every player i stores the last local observation in its state space. Every step of $\mathcal{P}$ is modelled as follows: The player *outputs* to the main module the action it chooses and then *inputs* from the main module the next observation.
- The main module stores the global state in its state space. Every step of $\mathcal{P}$ corresponds to the following: The main module *inputs* the actions of all players one by one, then it randomly picks the new state and new observations according to the rules of $\mathcal{P}$ based on the actions collected. The observations are lastly *output* to all players, again one by one.

We construct the distributed IMC so that only the outputting player chooses what action to output whereas the inputting player accepts whatever comes. The construction of modules for player i is illustrated in Fig. 1 along with constructions for input and output. The interesting part is how an action from the set $\{a_1, \ldots, a_r\}$ is input in a state s. Instead of waiting in s, the player travels by delay transitions in a round-robin fashion through a cycle of r states, where in the i-th state, only the action a_i is available. Thus, the player has no influence and must input the action that comes. By this construction, the main module has at most one action transition in every state such that the player cannot influence anything; other modules get no insight by observing time and thus the players have the same power as in the DEC-POMDP. □

4.3 Undecidability of Qualitative Existence in DEC-POMDP

Next, we show that the qualitative existence problem for DEC-POMDPs even with $n \ge 2$ players is undecidable. The proof has similarities with ideas from

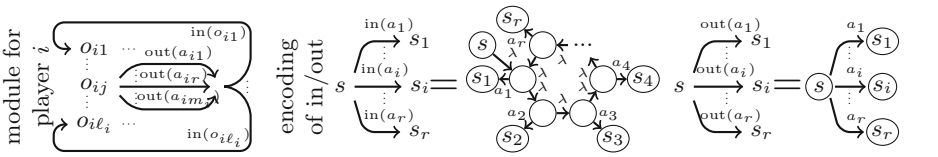


Fig. 1. Module for player i on the left. Input and output encoding to the right.

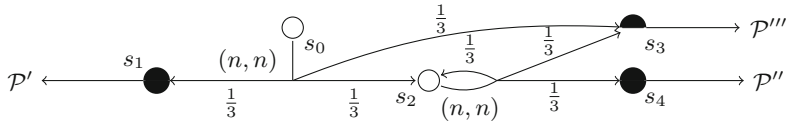


Fig. 2. Overall structure of $\mathcal{P}$ without details of $\mathcal{P}', \mathcal{P}''$ and $\mathcal{P}'''$.

[5] where it is shown that deciding existence of sure-winning strategies in safety games with 3 players and partial observation is undecidable. Using the randomness of DEC-POMDPs we show undecidability of the qualitative existence problem for reachability in 2-player DEC-POMDPs.

Theorem 3. *It is undecidable whether for a DEC-POMDP $\mathcal{P}$ with $n \geq 2$ players and a set T of target states in $\mathcal{P}$ if there exists a strategy profile $\boldsymbol{\sigma}$ such that* $\Pr^{\boldsymbol{\sigma}}_{\mathcal{P}}(\diamond T) = 1$.

Proof (Proof Sketch). We do a reduction from the non-halting problem of a deterministic Turing machine M that starts with a blank input tape. From M we construct a DEC-POMDP $\mathcal{P}$ with two players $Plr = \{1, 2\}$ such that M does not halt if and only if players 1 and 2 have strategies $\boldsymbol{\sigma} = (\sigma_1, \sigma_2)$ which ensure that the probability of reaching a target set T is 1. Figure 2 shows the overall structure of $\mathcal{P}$ without details of sub-modules $\mathcal{P}', \mathcal{P}''$ and $\mathcal{P}'''$.

Both players have two possible observations, black and white. We depict the observation of player 1 in the top-half and of player 2 in the bottom-half of every state. The play starts in s_0 and with probability 1, every player receives the black observation exactly once during the play. If the play goes to s_1 or s_4 the players will receive the observation at the same time and if the play goes to s_3 then player 2 will receive the observation in the step after player 1 does. The modules $\mathcal{P}', \mathcal{P}''$ and $\mathcal{P}'''$ are designed so that:

- In $\mathcal{P}'$, a target state is reached if and only if the sequence of actions played by both players encodes the initial configuration of M.
- In $\mathcal{P}''$, a target state is reached with probability 1 if and only if both players play the same infinite sequence of actions. Note that randomness is essential to build such a module.
- In $\mathcal{P}'''$, the target set is reached if and only if the sequences of actions played by player 1 and 2 encode two configurations C_1 and C_2 of M, respectively, such that C_1 is not an accepting configuration and C_2 is a successor configuration of C_1. This can be done since a finite automaton can be constructed that recognizes if one configuration is a successor of the other when reading both configurations at the same time. Note that it is possible because such configurations can only differ by a constant amount (the control state, the tape head position and in symbols in cells near the tape head).

It can be shown by induction that if there are strategies σ_1, σ_2 that ensure reaching T with probability 1 then every σ_i has to play the encoding of the jth

configuration of M when it receives the black observation in the jth step. Further, it can be shown that these strategies do ensure reaching T with probability 1 if M does not halt on the empty input tape and do not ensure reaching T with probability 1 if M halts. □

5 Decidability for Non-urgent Models

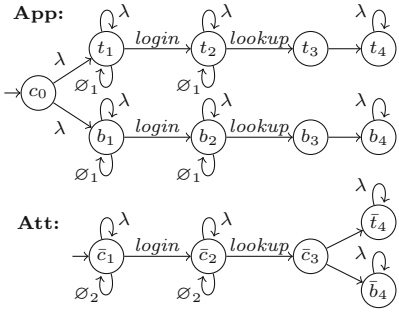
In this section, we turn our attention to a subclass of distributed IMCs, called *non-urgent*, that implies decidability for both the qualitative and quantitative value problems for 2 players.

Definition 4. *We call* $\mathcal{G} = ((S_i, Act_i, \hookrightarrow_i, \rightsquigarrow_i, s_{0i}))_{1 \le i \le n}$ non-urgent *if for every* $1 \le j \le n$*:*

1. *Every* $s \in S_j$ *is of one of the following forms:*
   (a) Synchronisation *state with at least one outgoing synchronisation action transition and exactly one outgoing delay transition which is a self-loop.*
   (b) Private *state with arbitrary outgoing delay transitions and private action transitions.*
2. *Player* j *has an action* $\varnothing_j \in Act_j$ *enabled in every synchronisation state from* S_j *that allows to "do nothing" and thus postpone the synchronisation. To this end,* $\varnothing_j$ *is also in* Act_k *for every other player* $k \neq j$ *but* $\varnothing_j$ *does not appear in* $\hookrightarrow_k$*. As a result,* j *does not take part in any synchronisation while choosing* $\varnothing_j$*.*

In a non-urgent distributed IMC, $\boldsymbol{s} \in S$ is called a (global) *synchronisation* state if it is the initial state or all $\boldsymbol{s}(j)$ are synchronisation states. We denote global synchronisation states by S'. All other global states $S \setminus S'$ are called *private*.

Example 5. Consider the non-urgent variant of Example 1 on the right. The "do nothing" actions are a natural concept; the only real modelling restriction is that one cannot model a communication time-out any more, the delay transitions from synchronisation states need to be self-loops.



Surprisingly, in this model, the secret can be leaked with probability 1 as follows. As before, the players reach the states $(t_2, \bar{c}_2)$ or $(b_2, \bar{c}_2)$ with equal probability. Now, the *App* player can arbitrarily postpone the *lookup* by committing to action $\varnothing_1$. Whenever the delay self-loop is taken, the player can re-decide to perform *lookup*. Since the self-loop is taken repetitively, the *App* player is flexible in choosing the timing of *lookup*. Thus, leaking the secret is simple, e.g. by performing *lookup* in an odd second when in t_2 and in an even second when in b_2.

For two players, we construct a general synchronisation scheme that (highly probably) shows the players the current global state after each communication.

Theorem 4. *The quantitative value problem for 2-player non-urgent distributed IMCs where the target set consists only of synchronisation states is in* ExpTime.

Being a special case, also the qualitative value problem is decidable. In essence, the problem becomes decidable because in the synchronisation states, the players can effectively exchange arbitrary information. This resembles the setting of [11]. The insight that observing global time provides an additional synchronization mechanism is not novel in itself, but it is obviously burdensome to formally capture in time-abstract models of asynchronous communication, and thus usually not considered. For distributed IMC, it still is non-trivial to develop; here it hinges on the non-urgency assumption. The results of [11] also indicate that for three or more players, additional constraints on the topology may be needed to obtain decidability.

In the rest of the section we prove Theorem 4, fixing a 2-player non-urgent distributed IMC $\mathcal{G} = ((S_i, Act_i, \hookrightarrow_i, \rightsquigarrow_i, s_{0i}))_{1 \leq i \leq 2}$, $p \in [0,1]$, and $T \subseteq S'$. We present the algorithm based on a reduction to a discrete-time Markov decision process and then discuss its correctness.

Markov decision process (MDP). An MDP is a tuple $\mathcal{M} = (S, A, P, s_0)$ where S is a finite set of states, A is a finite set of actions, $P : S \times A \to \Delta(S)$ is a partial probabilistic transition function, and s_0 is an initial state. An MDP is the special case of a DEC-POMDP with 1 player that has a unique observation for each state. A *play* in $\mathcal{M}$ is a sequence $\omega = s_0 s_1 \ldots$ of states such that $P(s_i, a_i)(s_{i+1}) > 0$ for some action a_i for every $i \geq 0$. A history is a prefix of a play. A *strategy* is a function π that to every history $h \cdot s$ assigns a probability distribution over actions such that if an action a is assigned a non-zero probability, then $P(s, a)$ is defined. A strategy π is *pure memoryless* if it assigns Dirac distributions to any history and its choice depends only on the last state of the history. When we fix a strategy π, we obtain a probability measure $\Pr^\pi$ over the set of plays. For further details, see [30].

The algorithm. It works by reduction to an MDP $\mathcal{M}_\mathcal{G} = (S', A, P, \boldsymbol{s_0})$ where

- $S' \subseteq S$ is the set of global synchronisation states;
- $A = \mathcal{C} \times \Sigma_1 \times \Sigma_2 \cup \{\bot\}$ where Σ_j is the set of pure memoryless strategies of player j in $\mathcal{G}$ that choose $\varnothing_j$ in every synchronisation state;
- For an arbitrary state (s_1, s_2), we define the transition function as follows:
  - For any $(\boldsymbol{c}, \sigma_1, \sigma_2) \in A$, the transition $P((s_1, s_2), (\boldsymbol{c}, \sigma_1, \sigma_2))$ is defined if $\boldsymbol{c}$ is available in (s_1, s_2) and the players agree in $\boldsymbol{c}$ on some action a, i.e. $\mathrm{En}(\boldsymbol{c}) = \{a\}$. If defined, the distribution $P((s_1, s_2), (\boldsymbol{c}, \sigma_1, \sigma_2))$ assigns to any successor state (s_1', s_2') the probability that in $\mathcal{G}$ the state (s_1', s_2') is reached from (s_1, s_2) via states in $S \setminus S'$ by choosing $\boldsymbol{c}$ and then using the pure memoryless strategy profile (σ_1, σ_2).
  - To avoid deadlocks, the transition $P((s_1, s_2), \bot)$ is defined iff no other transition is defined in (s_1, s_2) and it is a self-loop, i.e. it assigns probability 1 to (s_1, s_2).

The MDP $\mathcal{M}_{\mathcal{G}}$ has size exponential in $|\mathcal{G}|$. Note that all target states T are included in S'. Slightly abusing notation, let $\diamond T$ denote the set of plays in $\mathcal{M}_{\mathcal{G}}$ that reach the set T. From standard results on MDPs [30], there exists an optimal pure memoryless strategy π^*, i.e. a strategy satisfying $\Pr^{\pi^*}(\diamond T) = \sup_\pi \Pr^\pi(\diamond T)$. Furthermore, such a strategy π^* and the value $v := \Pr^{\pi^*}(\diamond T)$ can be computed in time polynomial in $|\mathcal{M}_{\mathcal{G}}|$. Finally, the algorithm returns TRUE if $v \geq p$ and FALSE, otherwise.

Correctness of the algorithm Let us explain why the approach is correct.

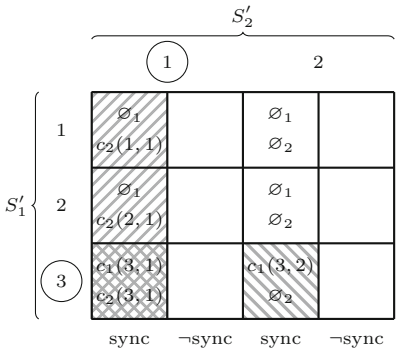
Proposition 2. *The value of $\mathcal{G}$ is equal to the value of $\mathcal{M}_{\mathcal{G}}$, i.e.*

$$\sup_{\boldsymbol{\sigma}} \Pr^{\boldsymbol{\sigma}}(\diamond T) \quad = \quad \sup_{\pi} \Pr^{\pi}(\diamond T).$$

Proof Sketch. As regards the $\leq$ inequality, it suffices to show that any strategy profile $\boldsymbol{\sigma}$ can be mimicked by some strategy π. This is simple as π in $\mathcal{M}_{\mathcal{G}}$ has always knowledge of the global state. Much more interesting is the $\geq$ inequality. We argue that for any strategy π there is a sequence of local strategy profiles $\boldsymbol{\sigma}^1, \boldsymbol{\sigma}^2, \ldots$ such that

$$\lim_{i \to \infty} \Pr^{\boldsymbol{\sigma}^i}(\diamond T) = \Pr^{\pi}(\diamond T).$$

The crucial idea is that a strategy profile communicates correctly (with high probability) the current global state in a synchronisation state by *delaying* as follows. The time is divided into phases, each of $|S'_1| \cdot 2|S'_2|$ slots (where S'_i are the synchronisation states of player i). We depict a phase by the table on the right where the time flows from top to bottom and from left to right (as reading a book). Players 1 and 2 try to synchronise in the row and column, respectively, corresponding to their current states (in circle) and in each slot take the choice $c_i(s_1, s_2)$ optimal given the current global state is (s_1, s_2); in the remaining slots they choose to do nothing. Since the players can change their choice only at random moments of time, their synchronising choice always stretches a bit into the successive silent slot (in a ¬sync column). The more we increase the size of each slot, the lower is the chance that a synchronisation choice of a player stretches to the next *synchronisation* slot. Thus, the lower is the chance of an erroneous synchronisation. We define the size of the slot to increase with i and also along the play so that for any fixed i the probability of at least one erroneous synchronisation is bounded by $\kappa_i < 1$ and for $i \to \infty$, we have $\kappa_i \to 0$. □



6 Discussion and Conclusion

This paper has introduced a foundational framework for modelling and synthesising distributed controllers interacting in continuous time via handshake communication. The continuous time nature of the model induces that the interleaving scheduler has in fact very little power. We studied cooperative reachability problems for which we presented a number of undecidability results, while for non-urgent models we established decidability of both quantitative and qualitative problems for the two-player case. In the framework considered, the restriction to exponential distributions is a technical vehicle, the results could have been developed in general continuous time, e.g. by using the model of stochastic automata [9].

Distributed IMCs can be considered as an attractive base model especially in the context of information flow and other security-related studies. This is because in contrast to the discrete time setting, the power of the interleaving scheduler is no matter of debate, it can leak no essential information.

From a more general perspective, distributed synthesis of control algorithms has received considerable attention in its entirety [22,28,29]. The asynchronous setting with handshake synchronisation has been considered in [23]. Notably, our assumption that players stay committed to a particular action choice for the time in between state changes implies the necessity to let the players explicitly solve distributed consensus problems. As done in [23], one can overcome this by letting local players pick sets of enabled actions (or letting them change choice with infinite speed), and then let some built-in magic pick a valid action from the intersection, implying that whenever possible a consensus is reached for sure. Such a change would however reintroduce the scheduler.

We should point out that distributed IMCs are not fully compositional: We are assuming a fixed vector of modules, and do not discuss that modules themselves may be vectors. Otherwise, we would face the phenomenon of auto-concurrency [10], where transitions with identical synchronisation actions might get enabled concurrently, despite not synchronising. This in turn would again re-introduce distinguishing power of the scheduler.

Distributed Markov chains [31] constitute another recent discrete-time approach where interleaving nondeterminism is tamed successfully via assumptions on the communication structure. The observation that continuous time reduces the power of the interleaving scheduler is not entirely new. Though not explicitly discussed, it underpins the model of probabilistic I/O automata (PIOA) [34] which uses I/O communication with input-enabledness and output-determinism. In that setting, output-determinism implies that local players have no decisive power, and hence a continuous time Markov chain arises. We can approximate I/O-based communication by distributed IMCs without the need for output-determinism. The approximation is linked to arbitrarily small but non-zero delays needed to cycle through synchronising action sets. A profound investigation of the continuous-time particularities of this and other synchronisation disciplines is considered an interesting topic for future work.

Acknowledgements. This work is supported by the EU 7th Framework Programme projects 295261(MEALS) and 318490 (SENSATION), by the Czech Science Foundation project P202/12/G061, the DFG Transregional Collaborative Research Centre SFB/TR 14 AVACS, and by the CDZ project 1023 (CAP).

References

1. Baier, C., Größer, M., Bertrand, N.: Probabilistic ω-automata. J. ACM **59**(1), 1 (2012)
2. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press, Cambridge (2008)
3. Bernstein, D.S., Givan, R., Immerman, N., Zilberstein, S.: The complexity of decentralized control of Markov decision processes. Math. Oper. Res. **27**(4), 819–840 (2002)
4. Bertoni, A., Mauri, G., Torelli, M.: Some recursively unsolvable problems relating to isolated cutpoints in probabilistic automata. In: Degano, P., Gorrieri, R., Marchetti-Spaccamela, A. (eds.) ICALP 1997. LNCS, vol. 1256. Springer, Heidelberg (1997)
5. Berwanger, D., Kaiser, L.: Information tracking in games on graphs. J. Logic Lang. Inform. **19**(4), 395–412 (2010)
6. Brookes, S.D., Hoare, C.A.R., Roscoe, A.W.: A theory of communicating sequential processes. J. ACM **31**(3), 560–599 (1984)
7. Canetti, R., Cheung, L., Kaynar, D.K., Liskov, M., Lynch, N.A., Pereira, O., Segala, R.: Analyzing security protocols using time-bounded task-PIOAs. Discrete Event Dyn. Syst. **18**(1), 111–159 (2008)
8. Cheung, L., Lynch, N.A., Segala, R., Vaandrager, F.W.: Switched PIOA: parallel composition via distributed scheduling. Theoret. Comput. Sci. **365**(1–2), 83–108 (2006)
9. D'Argenio, P.R., Katoen, J.-P.: A theory of stochastic systems part I: stochastic automata. Inf. Comput. **203**(1), 1–38 (2005)
10. Droste, M., Gastin, P.: Asynchronous cellular automata for pomsets without autoconcurrency. In: Sassone, V., Montanari, U. (eds.) CONCUR 1996. LNCS, vol. 1119. Springer, Heidelberg (1996)
11. Genest, B., Gimbert, H., Muscholl, A., Walukiewicz, I.: Asynchronous games over tree architectures. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) ICALP 2013, Part II. LNCS, vol. 7966, pp. 275–286. Springer, Heidelberg (2013)
12. Gimbert, H., Oualhadj, Y.: Probabilistic automata on finite words: decidable and undecidable problems. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6199, pp. 527–538. Springer, Heidelberg (2010)
13. Giro, S., D'Argenio, P.R., Fioriti, L.M.F.: Distributed probabilistic input/output automata: Expressiveness, (un)decidability and algorithms. Theoret. Comput. Sci. **538**, 84–102 (2014)
14. Guck, D., Han, T., Katoen, J.-P., Neuhäußer, M.R.: Quantitative timed analysis of interactive markov chains. In: Goodloe, A.E., Person, S. (eds.) NFM 2012. LNCS, vol. 7226, pp. 8–23. Springer, Heidelberg (2012)
15. Hermanns, H., Johr, S.: May we reach it? Or must we? In what time? With what probability? In: MMB, pp. 125–140. VDE Verlag (2008)
16. Hermanns, H.: Interactive Markov Chains and The Quest for Quantified Quality. Lecture Notes in Computer Science, vol. 2428. Springer, New York (2002)

17. Hermanns, H., Vester, S., Krčál, J.: Distributed synthesis in continuous time. CoRR abs/1601.01587 (2016)
18. Hermanns, H., Katoen, J.-P.: The how and why of interactive markov chains. In: Boer, F.S., Bonsangue, M.M., Hallerstede, S., Leuschel, M. (eds.) FMCO 2009. LNCS, vol. 6286, pp. 311–338. Springer, Heidelberg (2010)
19. Hermanns, H., Krčál, J., Křetínský, J.: Compositional verification and optimization of interactive markov chains. In: D'Argenio, P.R., Melgratti, H. (eds.) CONCUR 2013 – Concurrency Theory. LNCS, vol. 8052, pp. 364–379. Springer, Heidelberg (2013)
20. Katoen, J.-P., Klink, D., Neuhäußer, M.R.: Compositional abstraction for stochastic systems. In: Ouaknine, J., Vaandrager, F.W. (eds.) FORMATS 2009. LNCS, vol. 5813, pp. 195–211. Springer, Heidelberg (2009)
21. Katoen, J.-P., Zapreev, I.S., Hahn, E.M., Hermanns, H., Jansen, D.N.: The ins and outs of the probabilistic model checker MRMC. Perform. Eval. **68**(2), 90–104 (2011)
22. Madhusudan, P., Thiagarajan, P.S.: Distributed controller synthesis for local specifications. In: Orejas, F., Spirakis, P.G., van Leeuwen, J. (eds.) ICALP 2001. LNCS, vol. 2076, pp. 396–407. Springer, Heidelberg (2001)
23. Madhusudan, P., Thiagarajan, P.S.: A decidable class of asynchronous distributed controllers. In: Brim, L., Jančar, P., Křetínský, M., Kučera, A. (eds.) CONCUR 2002. LNCS, vol. 2421, pp. 145–160. Springer, Heidelberg (2002)
24. Milner, R.: Calculi for synchrony and asynchrony. Theoret. Comput. Sci. **25**, 267–310 (1983)
25. Neuts, M.F.: Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach. Courier Corporation, New York (1981)
26. Paz, A.: Introduction to Probabilistic Automata. Academic Press Inc., London (1971)
27. Pelozo, S.S., D'Argenio, P.R.: Security analysis in probabilistic distributed protocols via bounded reachability. In: Palamidessi, C., Ryan, M.D. (eds.) TGC 2012. LNCS, vol. 8191, pp. 182–197. Springer, Heidelberg (2013)
28. Pnueli, A., Rosner, R.: On the synthesis of an asynchronous reactive module. In: Ausiello, G., Dezani-Ciancaglini, M., Della Rocca, S.R. (eds.) ICALP. LNCS, vol. 372, pp. 652–671. Springer, Heidelberg (1989)
29. Pnueli, A., Rosner, R.: Distributed reactive systems are hard to synthesize. In: FOCS, pp. 746–757. IEEE Computer Society (1990)
30. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons, New York (2009)
31. Saha, R., Esparza, J., Jha, S.K., Mukund, M., Thiagarajan, P.S.: Distributed markov chains. In: D'Souza, D., Lal, A., Larsen, K.G. (eds.) VMCAI 2015. LNCS, vol. 8931, pp. 117–134. Springer, Heidelberg (2015)
32. Segala, R.: Modeling and Verification of Randomized Distributed Real-Time Systems. Ph.D. thesis, Massachusetts Institute of Technology (1995)
33. Sinopoli, B., Sharp, C., Schenato, L., Schaffert, S., Sastry, S.S.: Distributed control applications within sensor networks. Proc. IEEE **91**, 1235–1246 (2003)
34. Sue-Hwey, W., Smolka, S.A., Stark, E.W.: Composition and behaviors of probabilistic I/O automata. Theoret. Comput. Sci. **176**(1–2), 1–38 (1997)