AVACS – Automatic Verification and Analysis of
Complex Systems

REPORTS
of SFB/TR 14 AVACS

Editors: Board of SFB/TR 14 AVACS

Time-Bounded Reachability in Distributed Input/Output Interactive Probabilistic Chains

by
Georgel Calin Pepijn Crouzen Pedro R. D'Argenio
E. Moritz Hahn Lijun Zhang

AVACS Technical Report No. 64
June 2010
ISSN: 1860-9821

Publisher: Sonderforschungsbereich/Transregio 14 AVACS
(Automatic Verification and Analysis of Complex Systems)
Editors: Bernd Becker, Werner Damm, Martin Fränzle, Ernst-Rüdiger Olderog,
Andreas Podelski, Reinhard Wilhelm
ATRs (AVACS Technical Reports) are freely downloadable from `www.avacs.org`

Author(s) contact: Pepijn Crouzen (`crouzen@cs.uni-saarland.de`).

Time-Bounded Reachability in Distributed Input/Output Interactive Probabilistic Chains

Georgel Calin[1], Pepijn Crouzen[1], Pedro R. D'Argenio[2], E. Moritz Hahn[1], and Lijun Zhang[3]

[1] Department of Computer Science, Saarland University, Saarbrücken, Germany
[2] FaMAF, Universidad Nacional de Córdoba, Córdoba, Argentina
[3] DTU Informatics, Technical University of Denmark, Denmark

Abstract. We develop an algorithm to compute timed reachability probabilities for distributed models which are both probabilistic and nondeterministic. To obtain realistic results we consider the recently introduced class of (strongly) distributed schedulers, for which no analysis techniques are known.
Our algorithm is based on reformulating the nondeterministic models as parametric ones, by interpreting scheduler decisions as parameters. We then apply the PARAM tool to extract the reachability probability as a polynomial function, which we optimize using nonlinear programming.

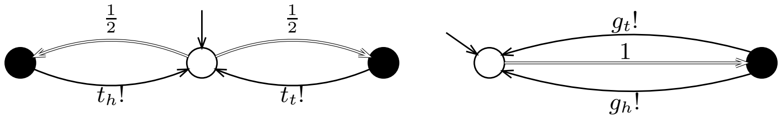
Key words: Distributed Systems, Probabilistic Models, Nondeterminism, Time-Bounded Reachability

1 Introduction

This paper considers the computation of reachability probabilities for compositional models with probabilistic and nondeterministic behavior. Such models arise, for instance, in the field of distributed algorithms, where probabilistic behavior is often used to break symmetries in the system. Nondeterminism may appear through the uncertain order of events occurring in different processes or to model freedom of design or unspecified behavior within a process.

Traditional analysis techniques for probabilistic models with nondeterminism compute the maximal and minimal probability to reach a set of configurations by considering all possible resolutions of the nondeterminism [2]. Recently it has been shown that this approach may lead to unrealistic results for models of distributed systems or algorithms [14]. The problem is that the traditional approach allows processes to use non-local information to influence their decisions.

As a running example we will use a simple coin-flip experiment. One player repeatedly flips a coin, while a second player (nondeterministically) guesses the outcome (See Fig. 1). We are interested in the probability that the second player manages to guess correctly at least once within t rounds. Intuitively this probability is $1 - \left(\frac{1}{2}\right)^t$, but it has been shown that standard analysis methods will produce a probability of 1 for any $t > 0$ [14]. The issue is that, from a *global* point of view, the optimal resolution of the nondeterministic guess simply uses the outcome of the coin-flip as a guide and this way always guesses correctly.



The coin-flip is depicted on the left-hand side and the coin-guess on the right-hand side. Initial states are indicated by incoming arrows; interactive transitions are labelled with their actions and probabilistic transitions $s \Rightarrow \mu$ are depicted by arrows from s to the support of μ, where each such arrow is labelled with the associated probability. Probabilistic transitions are required to synchronise in the composition.

Fig. 1: Basic I/O-IPC Models of the Repeated Coin-Flip Experiment.

Distributed schedulers restrict the resolution of nondeterminism by enforcing that *local* decisions of the processes are based only on local knowledge [14]. For our example, this means that the guesser is not allowed to base the guess on the outcome of the coin-flip. We will see that this leads to realistic results for the probability of attaining a correct guess. Strongly distributed schedulers, in addition, ensure that the relative probability of choosing between two different components does not change with time, provided these components remain idle and uninformed of the progress of the rest of the system.

When considering distributed (or strongly distributed) schedulers, bounds for reachability probabilities are both undecidable and unapproximable in general [13]. However *time-bounded* reachability probabilities, i.e., the probability to reach a set of configurations within a specified time-period, can be computed. For distributed schedulers, this is due to the fact that optimal solutions in this setting can be computed by only taking into account the subset of deterministic distributed schedulers, which is finite if the system under consideration is finite and acyclic. Nonetheless, the theoretical complexity of this problem is exponential in the number of states. The case of strongly distributed schedulers turns out to be more difficult. In this setting, optimal solutions may lie on pure probabilistic schedulers [14]. Therefore, exploring all possible solutions is not an option, and hence its decidability was unknown until now.

In this paper, we propose to reduce the problem of computing time-bounded reachability probabilities for distributed, probabilistic, and nondeterministic models, under distributed (or strongly distributed) schedulers to a nonlinear optimization problem, more precisely, to a polynomial optimization problem. Since polynomial programming is decidable [4], it turns out that time-bounded reachability probabilities under strongly distributed schedulers is also decidable. We use as our modeling vehicle the formalism of *input/output interactive probabilistic chains* (see Section 2). The computation of time-bounded reachability probabilities is achieved by reformulating the models as parametric Markov chains (see Section 5), where the parameters are the decisions of the schedulers and the distributed model is *unfolded* up to the specified time-point (see Section 6). The time-bounded reachability probability can now be expressed as a polynomial function and we can compute bounds for it by optimizing the function under certain constraints.

While for distributed schedulers the only restriction on the variables of the polynomials is that appropriately grouped they form a distribution (i.e. all variables take values between 0 and 1 and each group of variables sum up to 1), the case of strongly distributed schedulers requires some additional and more complex restrictions, the optimal value of the property being calculated through more involved nonlinear programming techniques.

In Sections 2 to 5 we recall the necessary background, lifted to the newly introduced setting of input/output interactive probabilistic chains. From Section 6 onwards we present our novel algorithm to compute time-bounded reachability for input/output interactive probabilistic chains.

2 Input/Output Interactive Probabilistic Chains

Interactive probabilistic chains (IPCs) are state-based models that combine discrete-time Markov chains and labelled transition systems [8]. IPCs can be used to compositionally model probabilistic systems. An important feature of the IPC formalism is that probabilistic transitions and action-labelled transitions are handled orthogonally. As our modeling vehicle we use input/output interactive probabilistic chains (I/O-IPCs), a restricted variant of IPCs with a strict separation between local and non-local behavior. The restriction of IPCs to I/O-IPCs follows the one of interactive Markov chains to I/O-IMCs in the continuous-time setting [3]. The separation between local and non-local behavior is achieved by partitioning the I/O-IPC actions in *input*, *output*, and *internal* actions. In this section we briefly introduce the necessary I/O-IPC definitions.

Let $\mathit{Dist}(X)$ be the set of all probability distributions over the finite set X.

Definition 1. *A **basic** I/O-IPC $\mathcal{P}$ is a quintuple $\langle S, A, \rightarrow_{\mathcal{P}}, \Rightarrow_{\mathcal{P}}, \hat{s}\rangle$, where:*

- *S is a finite set of states with $\hat{s} \in S$ the initial state;*
- *$A = A^I \cup A^O \cup A^{int}$ is a finite set of actions, consisting of disjoint sets of input actions (A^I), output actions (A^O), and internal actions (A^{int});*
- *$\rightarrow_{\mathcal{P}} \subseteq S \times A \times S$ is the set of interactive transitions;*
- *$\Rightarrow_{\mathcal{P}} : S \rightharpoonup \mathit{Dist}(S)$ is a partial function representing the set of probabilistic transitions.*

Input actions are suffixed by "?", output actions by "!" and we require that an I/O-IPC is input-enabled, i.e. for each state s and each input action a there is at least one state s' such that $(s, a, s') \in \rightarrow_{\mathcal{P}}$. We also require that the I/O-IPC is action-deterministic*, that is, for each state s and each action a there is at most one state s' such that $(s, a, s') \in \rightarrow_{\mathcal{P}}$. Finally we require that every state has at least one outgoing, internal, output, or probabilistic transition.*

We say that an I/O-IPC is *closed* if it has no input actions, i.e., $A^I = \emptyset$. Note that the requirement of action-determinism is introduced only to simplify the theoretical framework around schedulers. Nondeterministic choices between input transitions can be handled in a similar way as nondeterministic choices between output or internal transitions [14].

Given an action a, we use the shorthand notation $s \xrightarrow{a}_{\mathcal{P}} s'$ for an interactive transition $(s, a, s') \in \rightarrow_{\mathcal{P}}$ of any I/O-IPC $\mathcal{P}$. Given a distribution μ over the states of $\mathcal{P}$ we use the shorthand notation $s \Rrightarrow_{\mathcal{P}} \mu$ for $(s,\mu) \in \Rrightarrow_{\mathcal{P}}$. We often leave out the subscript when it is clear from the context.

2.1 Parallel Composition

Distributed I/O-IPCs are obtained through parallelizing ("$\|$") simpler I/O-IPCs.

Definition 2. *Two I/O-IPCs $\mathcal{P}$ and $\mathcal{Q}$ are composable if $A^O_{\mathcal{P}} \cap A^O_{\mathcal{Q}} = A_{\mathcal{P}} \cap A^{int}_{\mathcal{Q}} = A^{int}_{\mathcal{P}} \cap A_{\mathcal{Q}} = \emptyset$. If $\mathcal{P}$ and $\mathcal{Q}$ are composable then $\mathcal{C} := \mathcal{P}\|\mathcal{Q}$ will be the I/O-IPC*

$$\langle S_{\mathcal{P}} \times S_{\mathcal{Q}}, A^I_{\mathcal{C}} \cup A^O_{\mathcal{C}} \cup A^{int}_{\mathcal{C}}, \rightarrow_{\mathcal{C}}, \Rrightarrow_{\mathcal{C}}, (\hat{s}_{\mathcal{P}}, \hat{s}_{\mathcal{Q}})\rangle,$$

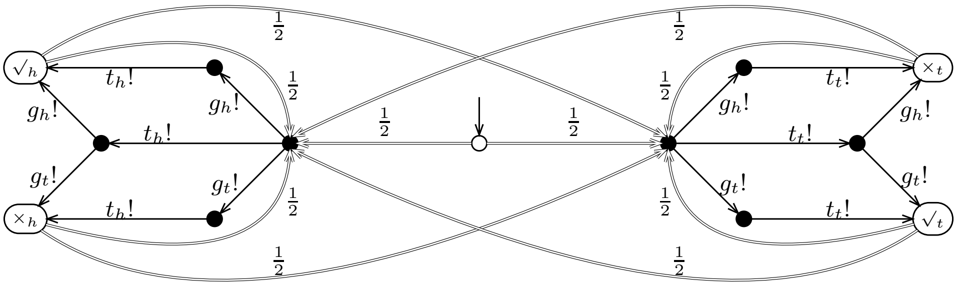
where $A^O_{\mathcal{C}} := A^O_{\mathcal{P}} \cup A^O_{\mathcal{Q}}$, $A^I_{\mathcal{C}} := \left(A^I_{\mathcal{P}} \cup A^I_{\mathcal{Q}}\right) \setminus A^O_{\mathcal{C}}$, $A^{int}_{\mathcal{C}} = A^{int}_{\mathcal{P}} \cup A^{int}_{\mathcal{Q}}$ and the transition relations are

$$\begin{aligned}
\rightarrow_{\mathcal{C}} = &\{(s,t) \xrightarrow{a}_{\mathcal{C}} (s',t) |\ s \xrightarrow{a}_{\mathcal{P}} s',\ a \in A_{\mathcal{P}} \setminus A_{\mathcal{Q}}\} \\
&\cup \{(s,t) \xrightarrow{a}_{\mathcal{C}} (s,t') |\ t \xrightarrow{a}_{\mathcal{Q}} t',\ a \in A_{\mathcal{Q}} \setminus A_{\mathcal{P}}\} \\
&\cup \{(s,t) \xrightarrow{a}_{\mathcal{C}} (s',t') |\ s \xrightarrow{a}_{\mathcal{P}} s',\ t \xrightarrow{a}_{\mathcal{Q}} t',\ a \in A_{\mathcal{P}} \cap A_{\mathcal{Q}}\} \\
\Rrightarrow_{\mathcal{C}} = &\{(s,t) \Rrightarrow_{\mathcal{C}} (\mu_s \times \mu_t) |\ s \Rrightarrow_{\mathcal{P}} \mu_s\ \wedge\ t \Rrightarrow_{\mathcal{Q}} \mu_t\}
\end{aligned}$$

with $\mu_s \times \mu_t$ denoting the product distribution on $S_{\mathcal{P}} \times S_{\mathcal{Q}}$.

Parallel composition can be extended to any finite set $\mathcal{C}$ of I/O-IPCs in the usual way. Let $\#\mathcal{C}$ denote the number of basic *components of $\mathcal{C}$.*

The result of synchronizing an input action with an output action through I/O-IPC parallelization will be an output action in the resulting model.



In the labelled states, the two I/O-IPCs in Fig. 1 distribute over the next states according to their possible combined choices. Otherwise, the output actions are interleaved. The flip matching the guess is represented by the $\surd$-labelled "goal" states.

Fig. 2: Distributed I/O-IPC Model of the Repeated Coin-Flip Experiment.

2.2 Vanishing and Tangible States

The use of distinct probabilistic and instantaneous transitions separates the concerns of time and interaction. In essence, it allows us to specify interactions between components which are instantaneous and do not have to be modeled with explicit time steps. We say that internal and output transitions are *immediate* and that probabilistic transitions are *timed*. We now assume that immediate transitions always take precedence over timed transitions. This assumption is known as the *maximal progress* assumption [18]. This separation between immediate and timed transitions is also reflected in the system states.

Definition 3. *A state is called vanishing if at least one outgoing immediate transition is enabled in it. If only probabilistic actions are enabled in a state then it is called tangible.*

The black-colored nodes in Fig. 1 and 2 are vanishing states, while the rest are all tangible states. For simplicity, in our current study we consider only models that do not exhibit Zeno behaviour (i.e. loops consisting of only immediate actions are not reachable/present in the distributed model).

2.3 Paths

An I/O-IPC *path* describes one possible *run* of the I/O-IPC. In such a run, we start in a particular state, follow a transition to another state, and so forth.

Definition 4. *Given an I/O-IPC* $\mathcal{P} = \langle S, A, \rightarrow, \Rightarrow, \hat{s} \rangle$*, a finite path of* $\mathcal{P}$ *of length* $n \in \mathbb{N}$ *is a sequence* $s_0 a_0 s_1 a_1 \ldots a_{n-1} s_n$ *where states* ($s_i \in S$ *for* $i \in [0, n]$) *and either actions or distributions* ($a_i \in A \cup Dist(S)$*, for* $i \in [0, n-1]$) *are interleaved. For consecutive states* s_i *and* s_{i+1} *we find that either* $a_i \in A$ *and* $(s_i, a_i, s_{i+1}) \in \rightarrow$ *or* $a_i \in Dist(S)$*,* s_i *is tangible,* $(s_i, a_i) \in \Rightarrow$*, and* $a_i(s_{i+1}) > 0$*. An infinite path of* $\mathcal{P}$ *is an infinite sequence* $s_0 a_0 s_1 a_1 \ldots$ *interleaving states and actions/distributions. We denote the last state of a finite path* σ *as* $last(\sigma)$*.*

For studying time-bounded reachability, we need a notion of *time*. We follow the definition of time in IPCs and say that only *probabilistic* transitions take time, while interactive transitions are considered to take place immediately [7].

Definition 5. *The* elapsed time *along a finite path* σ*, notation* $\mathsf{t}(\sigma)$*, is defined recursively, for states* s*, actions* a *and distributions* μ *over states:* $\mathsf{t}(s) = 0$*,* $\mathsf{t}(\sigma a s) = t(\sigma)$*, and* $\mathsf{t}(\sigma \mu s) = t(\sigma) + 1$*.*

2.4 Schedulers

We now wish to associate paths of an I/O-IPC with probabilities. The usual strategy is to define the probability of a path as the multiplication of the probabilities of its transitions. To define such a probability for paths in an I/O-IPC we need some way of resolving the nondeterministic choice between interactive transitions in vanishing states of an I/O-IPC. For all states $s \in S$, let $A^{en}_{s,\mathcal{P}} = \{a \in A^O \mid \exists s'.s \xrightarrow{a} s'\} \cup \{a \in A^{int} \mid \exists s'.s \xrightarrow{a} s'\}$ be the set of enabled immediate actions for s.

Definition 6. *A scheduler for an I/O-IPC* $\mathcal{P}$ *is a function* $\eta_{\mathcal{P}} : Paths(\mathcal{P}) \rightarrow Dist(A_{\mathcal{P}})$*, such that positive probabilities are assigned only to actions enabled in the last state of a path:* $\eta_{\mathcal{P}}(\sigma)(a) > 0$ *implies that* $a \in A^{en}_{last(\sigma),\mathcal{P}}$*.*

If $\mathcal{P}$ is closed, then a scheduler determines the probability to observe a certain path, which also allows us to define time-bounded reachability probabilities. We give the details, in the context of distributed schedulers, in Section 4.

3 I/O-IPC Nondeterminism Resolution

As we have seen, the probability of reaching a set of goal states in a distributed I/O-IPC depends on how the nondeterminism of chosing an action is handled. By assigning probabilities to the available actions, a scheduler can be seen as a refinement of the I/O-IPC such that the induced model becomes deterministic. It can thus be said that a scheduler enables us to determine reachability probabilities in a deterministic fashion.

However, the class of all schedulers for the model of a distributed system contains schedulers that are unrealistic in that they allow components of the system to use non-local information to guide their local decisions. To overcome this problem, *distributed* schedulers have been introduced, that restrict the possible choices of a scheduler to make them more realistic in a distributed setting [14]. Distributed schedulers have originally been introduced for (switched) probabilistic input/output automata [6] and we adapt them here for our input/output interactive probabilistic chains formalism.

To illustrate the necessity of distributed schedulers, consider the game described in Fig. 2 where an unbiased coin is repeatedly flipped and guessed by two independent entities at the same time. We are interested in the probability to reach the set of states labelled $\surd$ within a specified number t of timed (probabilistic) steps. This is exactly the probability that the guessing player guesses correctly within at most t tries. Intuitively, for each flip/guess turn, the guessing player should guess right with one half probability – the flip is probabilistic and its outcome should be hidden to the guesser.

However, it is clear that in the composed model there is a scheduler that arrives with probability one at a $\surd$ state within one timed step. This scheduler chooses the action g_h if the flip yields heads and g_t if the flip yields tails, thereby always winning. The purpose of distributed schedulers is to make sure that the decision between g_h and g_t is made only based on *local* information.

3.1 Distributed Schedulers

The main principle of distributed schedulers is to use a separate scheduler for each of the components of the system such that each has access only to their own scheduling history. To be able to reason about *local* information we first introduce path projections.

Given any distributed I/O-IPC $\mathcal{C} = \mathcal{P}_1 \| \ldots \| \mathcal{P}_n$ and a path $\sigma \in \mathit{Paths}(\mathcal{C})$, the projection $\sigma[\mathcal{P}_i]$ of σ on $\mathcal{C}$'s i-th basic component is given by:

- $(\hat{s}_{\mathcal{C}})[\mathcal{P}_i] = \pi_i(\hat{s}_{\mathcal{C}})$
- $(\sigma a s)[\mathcal{P}_i] = \begin{cases} \sigma[\mathcal{P}_i] & \text{if } a \notin A_{\mathcal{P}_i} \\ (\sigma[\mathcal{P}_i])a(\pi_i(s)) & \text{if } a \in A_{\mathcal{P}_i} \end{cases}$
- $(\sigma(\mu_1 \times \cdots \times \mu_n)s)[\mathcal{P}_i] = (\sigma[\mathcal{P}_i])\mu_i(\pi_i(s))$.

where $\pi_i\left((s_1, \ldots, s_n)\right) = s_i$ for all $(s_1, \ldots, s_n) \in S_{\mathcal{C}}$.

A *local* scheduler for $\mathcal{P}$ is simply any scheduler for $\mathcal{P}$ as given by Def. 6. A local scheduler resolves the nondeterminism arising from choices between enabled output and internal actions in one of the components. However, nondeterminism may also arise from the interleaving of the different components. In other words, if for some state in a distributed I/O-IPC, two or more components have enabled immediate actions, then it must be decided which component acts first. This decision is made by the *interleaving scheduler.*

Definition 7. *Given a distributed I/O-IPC* $C = \mathcal{P}_1 \| \ldots \| \mathcal{P}_n$*, an interleaving scheduler* $\mathcal{I} : Paths(\mathcal{C}) \to Dist(\{\mathcal{P}_1, \ldots, \mathcal{P}_n\})$ *is defined for paths* σ *such that* $last(\sigma)$ *is vanishing. The interleaving scheduler* $\mathcal{I}$ *chooses probabilistically an enabled component of the distributed system, i.e., we have that* $\mathcal{I}(\sigma)(\mathcal{P}_i) > 0$ *implies* $A^{en}_{last(\sigma[\mathcal{P}_i]),\mathcal{P}_i} \neq \varnothing$.

Local schedulers and an interleaving scheduler yield a *distributed scheduler.*

Definition 8. *Given a distributed I/O-IPC* $\mathcal{C} = \mathcal{P}_1 \| \ldots \| \mathcal{P}_n$*, local schedulers* $\eta_{\mathcal{P}_1}, \ldots, \eta_{\mathcal{P}_1}$*, and interleaving scheduler* $\mathcal{I}$*, the associated* distributed scheduler *is the function* $\eta_{\mathcal{C}} : Paths(\mathcal{C}) \to Dist\,(A_{\mathcal{C}})$ *such that, for all* $\sigma \in Paths(\mathcal{C})$ *with* $last(\sigma)$ *vanishing and for all* $a \in A_{\mathcal{C}}$*:*

$$\eta_{\mathcal{C}}(\sigma)(a) = \sum_{i=1}^{n} \mathcal{I}(\sigma)(\mathcal{P}_i) \cdot \eta_{\mathcal{P}_i}(\sigma[\mathcal{P}_i])(a)$$

We denote the set of all distributed schedulers as DS.

3.2 Strongly Distributed Schedulers

Although the class of distributed schedulers already realistically restricts the local decisions of processes in a distributed setting, in certain cases there exist distributed schedulers, where the interleaving schedulers are too powerful. In essence, the problem is that a distributed scheduler may use information from a component $\mathcal{P}_1$ to decide how to pick between components $\mathcal{P}_2$ and $\mathcal{P}_3$. In certain settings this is unrealistic. To counter this problem strongly distributed schedulers have been introduced [14].

Given any two components $\mathcal{P}_i$, $\mathcal{P}_j$ of a distributed I/O-IPC $\mathcal{C} = \mathcal{P}_1 \| \ldots \| \mathcal{P}_n$, consider the following property: for all σ, σ' such that $\sigma[\mathcal{P}_i] = \sigma'[\mathcal{P}_i]$ and $\sigma[\mathcal{P}_j] = \sigma'[\mathcal{P}_j]$, if $\mathcal{I}(\sigma)(\mathcal{P}_i) + \mathcal{I}(\sigma)(\mathcal{P}_j) \neq 0$ and $\mathcal{I}(\sigma')(\mathcal{P}_i) + \mathcal{I}(\sigma')(\mathcal{P}_j) \neq 0$ then

$$\frac{\mathcal{I}(\sigma)(\mathcal{P}_i)}{\mathcal{I}(\sigma)(\mathcal{P}_i) + \mathcal{I}(\sigma)(\mathcal{P}_j)} = \frac{\mathcal{I}(\sigma')(\mathcal{P}_i)}{\mathcal{I}(\sigma')(\mathcal{P}_i) + \mathcal{I}(\sigma')(\mathcal{P}_j)}. \tag{1}$$

Definition 9. *A scheduler* η *is strongly distributed if it is distributed and the restriction in Eq.* (1) *holds for the interleaving scheduler* $\mathcal{I}$ *of* η.

We denote the set of all strongly distributed schedulers as SDS. The intuition behind strongly distributed scheduler is that the choices the interleaving scheduler makes between two components $\mathcal{P}_i$, $\mathcal{P}_j$ should be consistent with respect to the local paths of $\mathcal{P}_i$, $\mathcal{P}_j$. If for two global paths, the local paths of $\mathcal{P}_i$, $\mathcal{P}_j$ are identical, then the probability of choosing $\mathcal{P}_i$ under the condition that we choose either $\mathcal{P}_i$ or $\mathcal{P}_j$ should be identical for both global paths.

Strongly distributed schedulers are useful depending on which system is considered for study [14]. When analyzing an auctioning protocol, for example, where each component models one of the bidders, then the order in which the bidders interact with the auctioneer should not leak information that can be used to the advantage of the other bidders. In such a situation, strongly distributed schedulers would provide more adequate worst-case/best-case probabilities.

However, if the interleaving scheduler should have access to the history of the components (as it might be the case for a kernel scheduler on a computer) then distributed schedulers should be considered, as the strongly distributed version might rule out valid possibilities.

4 Induced Probability Measure

When all the nondeterministic choices in a distributed I/O-IPC are scheduled, we end up with a probability measure on sets of paths of the I/O-IPC.

We define this probability measure in a similar way as is done for IPCs [22]. We fix a closed distributed I/O-IPC $\mathcal{C} = \mathcal{P}_1 \| \ldots \| \mathcal{P}_n$ with state space $S_\mathcal{C}$, actions $A_\mathcal{C}$, and initial state $\hat{s}$. The *cylinder* induced by the finite path σ is the set of infinite paths $\sigma^\uparrow = \{\sigma' \mid \sigma' \text{ is infinite and } \sigma \text{ is a prefix of } \sigma'\}$. Let the set of cylinders generate the σ-algebra on infinite paths of $\mathcal{C}$.

Definition 10. *Let η be a scheduler for $\mathcal{C}$. The* probability measure induced by *η on the set of infinite paths is the unique probability measure P_η such that, for any state s in $S_\mathcal{C}$, any action a in $A_\mathcal{C}$ and any distribution $\mu \in \mathit{Dist}(S_\mathcal{C})$:*

$$P_\eta(s^\uparrow) = \begin{cases} 1 & \textit{if } s = \hat{s} \\ 0 & \textit{otherwise} \end{cases}$$

$$P_\eta(\sigma a s^\uparrow) = \begin{cases} P_\eta(\sigma^\uparrow) \cdot \eta(\sigma)(a) & \textit{if } \mathit{last}(\sigma) \textit{ is vanishing and } \mathit{last}(\sigma) \xrightarrow{a} s \\ 0 & \textit{otherwise} \end{cases}$$

$$P_\eta(\sigma \mu s^\uparrow) = \begin{cases} P_\eta(\sigma^\uparrow) \cdot \mu(s) & \textit{if } \mathit{last}(\sigma) \textit{ is tangible and } \mathit{last}(\sigma) \Rightarrow \mu \\ 0 & \textit{otherwise} \end{cases}$$

We are now ready to define time-bounded reachability for I/O-IPCs.

Definition 11. *Given an I/O-IPC $\mathcal{P}$ with an initial distribution, a set of goal states $\mathcal{G}$ and a time-bound $t \in \mathbb{N}$, we have that the probability to reach $\mathcal{G}$ within t time-steps, denoted $P_\eta(\Diamond^{\leq t}\mathcal{G})$, is:*

$$P_\eta(\Diamond^{\leq t}\mathcal{G}) = P_\eta(\bigcup\{\sigma^\uparrow \mid \mathsf{t}(\sigma) \leq t \textit{ and } \mathit{last}(\sigma) \in \mathcal{G}\})$$

5 Parametric Markov Models

To compute probabilities for time-bounded reachability we will transform distributed I/O-IPCs into parametric Markov models (see Section 6). In this section we give a brief overview of parametric Markov chains [9, 16, 15].

Let S be a finite set of states. We let $V = \{x_1, \ldots, x_n\}$ denote a set of variables with domain $\mathbb{R}$. An *assignment* ζ is a function $\zeta : V \to \mathbb{R}$. A *polynomial* g over V is a sum of monomials $g(x_1, \ldots, x_n) = \sum_{i_1,\ldots,i_n} a_{i_1,\ldots,i_n} x_1^{i_1} \cdots x_n^{i_n}$ where each $i_j \in \mathbb{N}_0$ and each $a_{i_1,\ldots,i_n} \in \mathbb{R}$. A *rational function* f over V is a fraction $f(x_1, \ldots, x_n) = f_1(x_1, \ldots, x_n)/f_2(x_1, \ldots, x_n)$ of two polynomials f_1, f_2 over V.

Let $\mathcal{F}_V$ denote the set of rational functions from V to $\mathbb{R}$. Given $f \in \mathcal{F}_V$ and an assignment ζ, we let $\zeta(f)$ denote the rational function obtained by substituting each occurrence of $x \in V$ with $\zeta(x)$.

Definition 12. *A parametric Markov chain (PMC) is a tuple $\mathcal{D} = (S, \hat{s}, \mathbf{P}, V)$ where S is a finite set of states, $\hat{s}$ is the initial state, $V = \{v_1, \ldots, v_n\}$ is a finite set of parameters and $\mathbf{P}$ is the probability matrix $\mathbf{P} : S \times S \to \mathcal{F}_V$.*

The generalization of the probability matrix $\mathbf{P}$ to k steps is given below.

Definition 13. *Given a PMC $\mathcal{D} = (S, \hat{s}, \mathbf{P}, V)$, the k-step probability matrix $\mathbf{P}_k$, $k \in \mathbb{N}$, is defined recursively for any $k' > 1$ and states $s, s' \in S$:*

$$\mathbf{P}_0(s, s') = \begin{cases} 1 \text{ if } s = s' \\ 0 \text{ if } s \neq s' \end{cases}$$

$$\mathbf{P}_{k'}(s, s') = \sum_{s'' \in S} \mathbf{P}_{k'-1}(s, s'') \cdot \mathbf{P}(s'', s')$$

6 Parametric Interpretation

By having the scheduler η fixed, P_η together with the scheduled I/O-IPC $\mathcal{C}$ would become deterministic. To be more specific, by treating the interleaving and local scheduler decisions as unknowns we arrive at analyzing parametric Markov chains, the parameters being precisely the decisions that the interleaving and local schedulers perform.

We have seen in Section 4 that fixing the scheduler of a distributed I/O-IPC induces a probability measure on paths. Our approach is now to fix the scheduler *parametrically*, i.e., by treating the probabilities chosen by the interleaving and local schedulers as parameters. We will show that this *unfolding* of the I/O-IPC induces a PMC (see Section 5) whose states are paths of the distributed I/O-IPC. To make sure the induced PMC is finite we generate it only for paths up to a specific time-bound t. We then prove that computing the probability to reach a set of states within t time-units for the I/O-IPC is equivalent to computing time-unbounded reachability for the induced PMC.

To give an idea of how this unfolding works, consider again the repeated coin-flip experiment, depicted in Fig. 1 and 2. Intuitively it should hold that $Pr(\Diamond^{\leq 2}\{\surd_h, \surd_t\}) = 3/4$ if we assume the guessing player has no information about the outcome of each coin-flip. Since the guessing player must win within two steps he has two chances to guess correctly. The first guess yields a probability of $\frac{1}{2}$ to win, while for the case that the first guess is incorrect (probability $\frac{1}{2}$) there is another opportunity to guess correctly with probability $\frac{1}{2}$. This brings the overall probability to win within two steps to $\frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} = \frac{3}{4}$.Fig. 3 describes the unfolding of the distributed I/O-IPC from Fig. 2 up to time-point 2. On the right-hand side we see the structure of the PMC for one time-step. The unfolding up to 2 time steps is shown schematically on the left-hand side, where each square represents a copy of the structure on the right-hand side.

The local scheduler decisions in this case for each repeating structure P_k are x_k^h, x_k^t s.t. $x_k^h + x_k^t = 1$ and the interleaving scheduler decisions are $i_k^g, i_k^f, j_k^g, j_k^f$

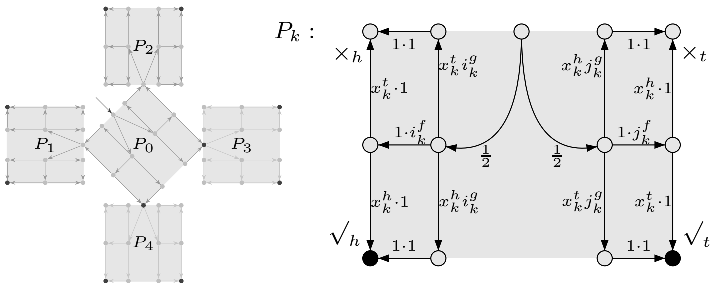


Fig. 3: Repeated Coin-Flip Experiment (PMC scheme up to time 2: all transitions are parametric. Interleaving is used for making the model more compact.)

s.t. $i_k^g + i_k^f = j_k^g + j_k^f = 1$. Here x_k^h, for example, denotes the probability assigned by the local scheduler for the guesser to pick "heads" for a local path ending in a "heads" vs. "tail" choice. The parameters i_k^g, i_k^f (as well as j_k^g, j_k^f) denote the probabilities the interleaving scheduler assigns to the "guessing" model and the "flipping" model respectively, for a given global path which enables them both.

Now, $Pr(\Diamond^{\leq 2}\{\surd_h, \surd_t\})$ can be computed as the sum of the cumulated probabilities on the paths leading to $\{\surd_h, \surd_t\}$ states by using the given unfolding in Fig. 3 and the above parameter restrictions:

$$\tfrac{1}{2}\Big(x_0^h\cdot i_0^g + i_0^f\cdot x_0^h + (x_0^t\cdot i_0^g + i_0^f\cdot x_0^t)\cdot\big[\tfrac{1}{2}(x_1^h\cdot i_1^g + i_1^f\cdot x_1^h) + \tfrac{1}{2}(x_1^t\cdot j_1^g + j_1^f\cdot x_1^t)\big]\Big) +$$
$$\tfrac{1}{2}\Big(x_0^t\cdot j_0^g + j_0^f\cdot x_0^t + (x_0^h\cdot j_0^g + j_0^f\cdot x_0^h)\cdot\big[\tfrac{1}{2}(x_2^h\cdot i_2^g + i_2^f\cdot x_2^h) + \tfrac{1}{2}(x_2^t\cdot j_2^g + j_2^f\cdot x_2^t)\big]\Big) =$$
$$\tfrac{1}{2}(x_0^h + x_0^t\cdot(\tfrac{1}{2}x_1^h + \tfrac{1}{2}x_1^t)) + \tfrac{1}{2}(x_0^t + x_0^h\cdot(\tfrac{1}{2}x_2^h + \tfrac{1}{2}x_2^t)) = \tfrac{3}{4}$$

We now define the above interpretation of choices via parameters formally.

Definition 14. *Let $S_{\mathcal{C}}^t \subseteq Paths(\mathcal{C})$ be the set of all paths with time-length $\leq t$ in a closed, distributed I/O-IPC $\mathcal{C} = \mathcal{P}_1\|\ldots\|\mathcal{P}_n$ which does not exhibit Zeno-behaviour. Define the parameters set V by*

$$\begin{aligned} V = \{\ & y_\sigma^i \mid \sigma \in S_{\mathcal{C}}^t, 1 \leq i \leq \#\mathcal{C}, A^{en}_{last(\sigma[\mathcal{P}_i]),\mathcal{P}_i} \neq \varnothing\} \cup \\ \{\ & x^a_{\sigma[\mathcal{P}_i]} \mid \sigma \in S_{\mathcal{C}}^t, 1 \leq i \leq \#\mathcal{C}, a \in A^{en}_{last(\sigma[\mathcal{P}_i]),\mathcal{P}_i}\} \end{aligned}$$

and let $\mathbf{P}$ match the induced probability measure, namely for any path $\sigma \in S_{\mathcal{C}}^t$, any state s of $\mathcal{C}$, any action a of $\mathcal{C}$ and any distribution μ over the states of $\mathcal{C}$:

$$\begin{array}{ll} \mathbf{P}(\sigma, \sigma a s) = y_\sigma^i \cdot x^a_{\sigma[\mathcal{P}_i]} & \textit{if } last(\sigma) \textit{ is vanishing, } last(\sigma) \xrightarrow{a} s,\ a \in A^{en}_{last(\sigma[\mathcal{P}_i]),\mathcal{P}_i} \\ \mathbf{P}(\sigma, \sigma\mu s) = \mu(s) & \textit{if } last(\sigma) \textit{ is tangible, } \mathsf{t}(\sigma) < t,\ last(\sigma) \Rightarrow \mu \\ \mathbf{P}(\sigma, \sigma) = 1 & \textit{if } last(\sigma) \textit{ is tangible, } \mathsf{t}(\sigma) = t. \end{array}$$

All other transition probabilities are zero. The unfolding of the I/O-IPC $\mathcal{C}$ up to time bound t is then the PMC $\mathcal{D} = (S_{\mathcal{C}}^t, \hat{s}_{\mathcal{C}}, \mathbf{P}, V)$. Given a set of states $\mathcal{G}$ of $\mathcal{C}$, we write $\overline{\mathcal{G}}$ for the paths in $S_{\mathcal{C}}^t$ that end in a state in $\mathcal{G}$, but never visit a state in $\mathcal{G}$ on the way.

The finiteness of $S^t_{\mathcal{C}}$ and V is guaranteed by the exclusion of infinite chains consisting of only immediate actions, as it implies that for each state in $\mathcal{C}$ a tangible state is reachable within a finite number of non-probabilistic steps.

The variables in Def. 14 can be restricted to ensure that they represent valid scheduler decisions in the following way:

$$\begin{cases} 0 \leq v \leq 1 & \text{if } v \in V \\ \sum_{a \in A} x^a_{\sigma[\mathcal{P}_i]} = 1 & \text{if } \sigma \in S^t_{\mathcal{C}};\ 1 \leq i \leq \#\mathcal{C} \text{ with } A = A^{en}_{last(\sigma[\mathcal{P}_i]),\mathcal{P}_i} \\ \sum_{i \in I} y^i_\sigma = 1 & \text{if } \sigma \in S^t_{\mathcal{C}};\ I = \{i \mid 1 \leq i \leq \#C, last(\sigma[P_i]) \text{ vanishing}\} \end{cases} \tag{2}$$

We write $\zeta \vdash (2)$ if the assignment $\zeta : V \to [0,1]$ satisfies (2) and by $\zeta(\mathbf{P}_k(\hat{s},\sigma))$ we denote $\mathbf{P}_k(\hat{s},\sigma)$ with all variables substituted according to the assignment ζ. I/O-IPC path probabilities are related to k-step transition probabilities of the induced PMC by the following.

Lemma 1. *For a closed, distributed I/O-IPC $\mathcal{C}$, let $\mathcal{D}$ be as in Def. 14. Then*

(i) For every distributed scheduler η there is an assignment $\zeta \vdash (2)$ such that for all $\sigma \in S^t_{\mathcal{C}}$: $P_\eta(\sigma^\uparrow) = \zeta(\mathbf{P}_k(\hat{s},\sigma))$ where k is the length of σ.

(ii) Reciprocally, for every assignment $\zeta : V \to [0,1]$ with $\zeta \vdash (2)$ there is a distributed scheduler η such that for all $\sigma \in S^t_{\mathcal{C}}$: $P_\eta(\sigma^\uparrow) = \zeta(\mathbf{P}_k(\hat{s},\sigma))$.

The proof of Lemma 1 can be found in Appendix A.

We can now reformulate the bounded reachability problem for I/O-IPCs under distributed schedulers as an unbounded reachability problem for the associated induced PMC – which is also bounded in a sense as it is acyclic and has precisely the same depth as the given time-bound.

Theorem 1. *Time-bounded reachability for an I/O-IPC $\mathcal{C}$ under distributed schedulers is equivalent to checking time-unbounded reachability on the PMC $\mathcal{D} = (S^t_{\mathcal{C}}, \hat{s}_{\mathcal{C}}, \mathbf{P}, V)$ as in Def. 14 for assignments that satisfy (2):*

$$\sup_{\eta \in DS} P_\eta(\Diamond^{\leq t}\mathcal{G}) = \sup_{\zeta \vdash (2)} \zeta(P_{\mathcal{D}}(\Diamond\,\overline{\mathcal{G}})) \ \textit{and}\ \inf_{\eta \in DS} P_\eta(\Diamond^{\leq t}\mathcal{G}) = \inf_{\zeta \vdash (2)} \zeta(P_{\mathcal{D}}(\Diamond\,\overline{\mathcal{G}})).$$

The proof of Theorem 1 can be found in Appendix A.

To extend this result to strongly distributed schedulers we must further restrict the variables of the induced PMC such that the allowed assignments match the strongly distributed schedulers. First we introduce new variables which represent the conditional probabilities in (1). For every i, j, $1 \leq i, j \leq \#\mathcal{C}$, $i \neq j$, and $\sigma \in S^t_{\mathcal{C}}$, define a new variable $z^{i,j}_{\sigma[\mathcal{P}_i],\sigma[\mathcal{P}_j]} \notin V$. Notice that two different $\sigma, \sigma' \in S^t_{\mathcal{C}}$ may induce the same variable if $\sigma[\mathcal{P}_i] = \sigma'[\mathcal{P}_i]$ and $\sigma[\mathcal{P}_j] = \sigma'[\mathcal{P}_j]$. We write V_z for the set of all such variables $z^{i,j}_{\sigma[\mathcal{P}_i],\sigma[\mathcal{P}_j]}$.

Using these new variables we impose new restrictions on the variables in the induced PMC of a distributed I/O-IPC.

$$z^{i,j}_{\sigma[\mathcal{P}_i],\sigma[\mathcal{P}_j]}(y^i_\sigma + y^j_\sigma) = y^i_\sigma \quad \text{if } 1 \leq i, j \leq \#\mathcal{C},\ i \neq j, \text{ and } \sigma \in S^t_{\mathcal{C}} \tag{3}$$

Theorem 2. *Time-bounded reachability for an I/O-IPC $\mathcal{C}$ under strongly distributed schedulers is equivalent to time-unbounded reachability of the parametric Markov chain $\mathcal{D} = (S^t_{\mathcal{C}}, \hat{s}_{\mathcal{C}}, \mathbf{P}, V \cup V_z)$ resulted through unfolding as in Def. 14 under the assumptions* (2) *and* (3)*:*

$$\sup_{\eta \in SDS} P_\eta(\Diamond^{\leq t}\mathcal{G}) = \sup_{\zeta \vdash (2),(3)} \zeta(P_{\mathcal{D}}(\Diamond\,\overline{\mathcal{G}})) \text{ and } \inf_{\eta \in SDS} P_\eta(\Diamond^{\leq t}\mathcal{G}) = \inf_{\zeta \vdash (2),(3)} \zeta(P_{\mathcal{D}}(\Diamond\,\overline{\mathcal{G}})).$$

The proof of Theorem 2 can be found in Appendix A.

7 Algorithm

Time-unbounded reachability probabilities for PMCs can be computed using the tool PARAM, which, since our PMC models are acyclic, results in analyzing a set of polynomial functions over the given variables. These polynomial functions can then be optimized under the constraints given by (2) and – for strongly distributed schedulers – (3) using standard numerical solvers.

We now present our algorithm to compute extremal time-bounded reachability probabilities for distributed I/O-IPCs. The following inputs are required: a closed, distributed I/O-IPC $\mathcal{C}$ which exhibits no Zeno-behavior, a time-bound t, and a set of goal states $\mathcal{G}$. The following steps are sequentially executed:

1. The I/O-IPC is unfolded up to time-bound t, yielding a PMC (see Def. 14). Additional linear constraints on the parameters are provided to ensure that all scheduler decisions lie in the interval $[0, 1]$ and that specific sets of parameters sum up to 1. In the case of strongly distributed schedulers, non-linear constraints are also generated as described in Thm. 2.
2. The time-unbounded reachability probability for the set of goal states $\mathcal{G}$ is calculated parametrically for the PMC generated in step 1 using the PARAM tool [15]. The result is a polynomial function.
3. The polynomial function generated in step 2 is optimized under the constraints generated in step 1 using non-linear programming. We have used the *active-set* algorithm [12, 17] provided by the *fmincon* function of Matlab[†], but any non-linear programming tool can in principle be used.

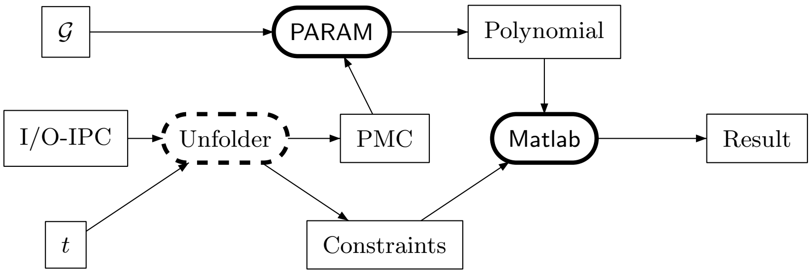


Fig. 4: The Envisioned Tool-Chain: ellipses represent tools, boxes represent data, where dashed ellipses represent tools that are currently in development.

[†] See http://www.mathworks.com

An overview of this tool-chain is presented in Fig. 4. The tool which unfolds the I/O-IPC and generates a PMC together with linear and non-linear constraints is still under development. We have, however generated PMC models and constraints semi-automatically for several case studies which we present next.

8 Case Studies

In this section we apply our algorithm to three case studies. Since the unfolder tool, which translates distributed I/O-IPCs into PMCs and constraints, is still under development we have generated PMCs and constraints for these cases in a semi-automatic way. The PARAM tool has been run on a computer with a 3 Ghz processor and 1 GB of memory, while Matlab was run on a computer with two 1.2 Ghz processors and 2 GB of memory. All I/O-IPC and PMC models are available from the authors.

8.1 Mastermind

In the game of Mastermind [1] one player, the *guesser*, tries to find out a *code*, generated by the other player, the *encoder*. The code consists of a number of tokens of fixed positions, where for each token one color (or other labelling) out of a prespecified set is chosen. Colors can appear multiple times.

Each round, the guesser guesses a code. This code is compared to the correct one by the encoder who informs the guesser on a) how many tokens were of the correct color *and* at the correct place and b) how many tokens were *not* at the correct place, *but* have a corresponding token of the same color in the code.

Notice that the decisions of the encoder during the game are deterministic, while the guesser has the choice between all valid codes. We assume that the encoder chooses the code probabilistically with a uniform distribution over all options. The guesser's goal is to find the code as fast as possible and ours to compute the maximal probability for this to happen within t rounds.

We formalize the game as follows: we let n be the number of tokens of the code and m the number of colors. This means there are m^n possible codes. Let $\mathcal{O}$ denote the set of all possible codes. We now informally describe the I/O-IPCs which represent the game. The guesses are described by actions $\{g_o \mid o \in \mathcal{O}\}$, whereas the answers are described by actions $\{a_{(x,y)} \mid x, y \in [0, n]\}$.

The guesser $\mathcal{G}$ repeats the following steps: From the initial state, $s_{\mathcal{G}}$ it first takes a probabilistic step to state $s'_{\mathcal{G}}$ and afterwards the guesser returns to the initial state via one of m^n transitions, each labelled with an output action $g_o!$. In both states the guesser receives answers $a_{(x,y)}?$ from the encoder and for all answers the guesser simply remains in the same state, except for the answer $a_{(n,n)}$ which signals that the guesser has guessed correctly. When the guesser receives this action it moves to the absorbing state $s''_{\mathcal{G}}$.

The encoder $\mathcal{E}$ is somewhat more complex. It starts by picking a code probabilistically, where each code has the same probability $\frac{1}{m^n}$. Afterwards the encoder repeats the following steps indefinitely. First it receives a guess from the guesser, then it replies with the appropriate answer and then it takes a probabilistic transition. This probabilistic step synchronizes with the probabilistic step of the guesser, which allows us to record the number of rounds the guesser needs to find the code.

The Mastermind game is the composition $\mathcal{C} := \mathcal{G} \| \mathcal{E}$ of the two basic I/O IPCs. Using the tool described in Section 7 we can now reason about the maximal probability $Pr(\Diamond^{\leq t} s''_{\mathcal{G}})$ to break the code within a prespecified number t of guesses. We consider here the set of all distributed schedulers as we obviously want that the guesser uses only local information to make its guesses. If we were to consider the set of all schedulers the maximum probability would be 1 for any time-bound as the guesser would immediately choose the correct code with probability 1. If only two components are considered, condition (1) has no effect and every distributed scheduler is also strongly distributed. Therefore we omit the analysis under strongly distributed schedulers for this case study.

Settings			PMC			PARAM			NLP	
n	m	t	$\#S$	$\#T$	$\#V$	Time(s)	Mem(MB)	$\#V$	Time(s)	Pr
2	2	2	197	248	36	0.0492	1.43	17	0.0973	0.750
2	2	3	629	788	148	0.130	2.68	73	0.653	1.00
3	2	2	1545	2000	248	0.276	5.29	93	1.51	0.625
3	2	3	10953	14152	2536	39.8	235	879	1433	1.00
2	3	2	2197	2853	279	0.509	6.14	100	2.15	0.556

Table 1: Results of Mastermind Case Study

Results are given in Table 1. In addition to the model parameters (n, m), the time bound (t) and the result (Pr) we provide statistics for the various phases of the algorithm. For the unfolded PMC we give the number of states ($\#S$), transitions ($\#T$), and variables ($\#V$). For the PARAM tool we give the time needed to compute the polynomial, the memory required, and the number of variables that remain in the resulting polynomial. Finally we give the time needed for Matlab to optimize the polynomial provided by PARAM under the linear constraints that all scheduler decisions lie between 0 and 1. For this case study we generated PMC models and linear constraints semi-automically given the parameters n, m, and t.

8.2 Dining Cryptographers

The dining cryptographers problem is a classical anonymity problem [5]. The cryptographers must work together to deduce a particular piece of information using their local knowledge, but at the same time each cryptographers' local knowledge may not be discovered by the others. The problem is as follows: three cryptograpers have just finished dining in a restaurant when their waiter arrives to tell them their bill has been paid anonymously. The cryptographers now decide they wish to respect the anonimity of the payer, but they wonder if one of the cryptographers has paid or someone else. They resolve to use the following protocol to discover whether one of the cryptographers paid, without revealing which one.

We depict part of the I/O-IPC models in Fig. 5. On the right-hand side we have the I/O-IPC $\mathcal{F}$ that simply decides who paid (actions p_i) and then starts the protocol. Each cryptographer has a probability of $\frac{1}{6}$ to have paid and there is a probability of $\frac{1}{2}$ that none of them has paid. On the left-hand side of Fig. 5 we see part of the I/O-IPC $\mathcal{G}_1$ for the first cryptographer. Each cryptographer flips a fair coin such that the others cannot see the outcome. In Fig. 5 we only show the case where cryptographer one flips heads. Each cryptographer now shows his coin to himself and his right-hand neighbour (actions h_i for heads and t_i for tails). This happens in a fixed order. Now, again in a fixed order, they proclaim whether or not the two coins they have seen were the same or different (actions s_i for same and d_i for different). However, if a cryptographer has paid he or she will *lie* when proclaiming whether the two coins were identical or not. In Fig. 5 we show the case where cryptographer one has not paid, so he proclaims the truth. Now we have that if there is an even number of "different" proclamations, then all of the cryptographers told the truth and it is revealed that someone else paid. If, on the other hand, there is an odd number of "different" proclamations, one of the cryptographers must have paid the bill, but it has been shown that there is no way for the other two cryptographers to know which one has paid. In our model the cryptographer first attempts to guess whether or not a cryptographer has paid (actions c_i to guess that a cryptographer has paid, action n_i if not). In case the cryptographer decides a cryptographer has paid, he guesses which one (action $g_{i,j}$ denotes that cryptographer i guesses cryptographer j has paid.

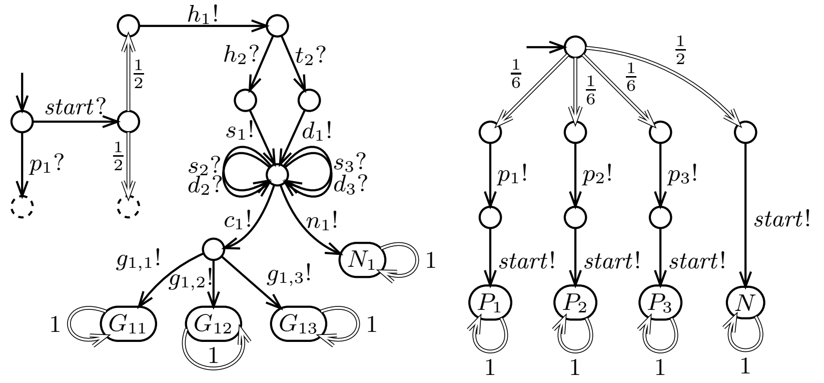


Fig. 5: Part of the I/O-IPC model $\mathcal{G}_1$ (left) of the first dining cryptographer and the I/O-IPC $\mathcal{F}$ (right) that probabilistically decides who has actually paid.

We can see that a "run" of the distributed I/O-IPC $\mathcal{C} = \mathcal{F} \| \mathcal{G}_1 \| \mathcal{G}_2 \| \mathcal{G}_3$ takes two time-units, since there is one probabilistic step to determine who paid and one probabilistic step where all coins are flipped simultaneously. We are interested in two properties of this algorithm: first, all cryptographers should be able to determine whether someone else has paid or not. We can express this property, for example for the first cryptographer, as a reachability probability property:

$$P(\Diamond^{\leq 2}\{P_1, P_2, P_3\} \times \{G_{11}, G_{12}, G_{13}\} \times S_2 \times S_3 \cup \{N\} \times \{N_1\} \times S_2 \times S_3) = 1. \quad (4)$$

S_2 and S_3 denote the complete I/O-IPC state spaces of the second and third cryptographer. For the other cryptographers we find similar properties. Secondly, we must check that the payer remains anonymous. This means that, in the case that a cryptographer pays, the other two cryptographers cannot guess this fact. We can formulate this as a conditional reachability probability:

$$\frac{P(\Diamond^{\leq 2}\{P_2\} \times \{G_{12}\} \times S_2 \times S_3 \cup \{P_3\} \times \{G_{13}\} \times S_2 \times S_3)}{P(\Diamond^{\leq 2}\{P_2, P_3\} \times S_1 \times S_2 \times S_3)} = \frac{1}{2}. \tag{5}$$

I.e., the probability for the first cryptographer to guess correctly who paid – under the condition that one of the other cryptographers paid – is one half.

	PMC			PARAM			NLP	
Property	$\#S$	$\#T$	$\#V$	Time(s)	Mem(MB)	$\#V$	Time(s)	Pr
(4)	294	411	97	9.05	4.11	24	0.269	1.00
(5), top	382	571	97	9.03	4.73	16	0.171	0.167
(5), bottom	200	294	97	8.98	4.14	0	N/A	1/3

Table 2: Results of Dining Cryptographers Case Study.

Table 2 shows the results for the dining cryptographers case study. We compute the conditional probability in (5) by computing the top and bottom of the fraction separately. We can see that both properties (4) and (5) are fulfilled. Table 2 also lists statistics on the tool performances and model sizes as described for Table 1. Note especially that the third reachability probability was computed directly by PARAM. I.e., this probability is independent of the scheduler decisions and PARAM was able to eliminate all variables.

8.3 Randomized Scheduler Example

For the class of strongly distributed schedulers it may be the case that the maximal or minimal reachability probability can not be attained by a deterministic scheduler, i.e., a scheduler that always chooses one action/component with probability one. As our final case study we use a small example of such an I/O-IPC as depicted by Fig. 4 in [14]. In this example the maximal reachability probability for deterministic strongly distributed schedulers is $\frac{1}{2}$, while there exists a randomized strongly distributed scheduler with reachability probability $\frac{13}{24}$.

Table 3 shows the result of applying our tool chain to this example. We see that we can find a scheduler with maximal reachability probability 0.545, which is even greater than $\frac{13}{24}$. Note that we can express the maximal reachability probability as a time-bounded property because the example is acyclic. However, for this case, the result from Matlab depends on the initial assignment given to the solver. For certain initial assignments the solver returns a maximal probability of only 0.500. This indicates that further investigation is required in the appropriate nonlinear programming tool for our algorithm.

PMC			PARAM			NLP	
$\#S$	$\#T$	$\#V$	Time(s)	Mem(MB)	$\#V$	Time(s)	Pr
13	23	12	0.00396	1.39	11	0.241	0.545†

Table 3: Results of Randomized Scheduler Case Study.
(‡ For certain settings, Matlab reports a maximal probability of 0.500)

9 Related Work

The problem that global schedulers may be too optimistic or pessimistic in the verification of distributed, probablistic, and nondeterministic systems has been noted in several different settings [19–21, 14]. One approach to resolve this issue is to use *partial-information* schedulers [10]. Using partial-information schedulers allows the hiding of information that a global scheduler should not realistically use. However, this approach still assumes there is only one global scheduler, instead of several local schedulers as presented in this paper. For the class of memoryless partial-information schedulers, the extremal long-run average outcomes of tasks can be calculated by reformulating the problem as a non-linear programming problem [10]. A testing pre-order for distributed models with probabilistic and nondeterministic choices has been suggested which is aimed at realistically representing the power of schedulers in a distributed setting [11]. In this context, reachability probabilities (under a particular scheduler) are defined in a similar way as in our paper, but no algorithm to compute extremal probabilities or to compute the pre-order is given. It would be very interesting to study whether this pre-order [11] indeed preserves extremal time-bounded reachability probabilities when lifted to the setting of I/O-IPCs.

10 Conclusion

In this paper we have presented an algorithm to compute maximal and minimal time-bounded reachability probabilities for I/O-IPCs under distributed schedulers or strongly distributed schedulers. The core principle of our algorithm is to reformulate the problem as a polynomial optimization problem under linear and, in the case of strongly distributed schedulers, polynomial constraints.

The main drawback of our approach is that the PMC induced in our algorithm grows exponentially with the size of the original model and the specified time-bound, as the state space of the PMC consists of all paths of the original model, up to a time-bound. However, no other algorithm exists that can compute properties of distributed models under (strongly) distributed schedulers.

In several areas improvements can be made. First, it can be investigated if special purpose algorithms can be used for the specific type of non-linear programming problems we encounter in our context. Secondly, the memory-usage may be optimized by using the fact that in our setting we see only polynomial functions and do not make use of rational polynomial functions.

Acknowledgments. This work was partly supported by the German Research Council (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS) (see www.avacs.org for more information), the DAAD-MinCyT project "Quantitative Techniques for Dependable Distributed Systems (QTDDS)", and the ANPCyT-PICT 26135 "Verification of Probabilistic Distributed Systems". Lijun Zhang received partial support from *MT-LAB*, a VKR Centre of Excellence.

References

1. L. H. Ault. *Das Mastermind-Handbuch.* Ravensburger Buchverlag, 1982.

2. A. Bianco and L. de Alfaro. Model Checking of Probabalistic and Nondeterministic Systems. In *FOSSACS*, pages 499–513, 1995.
3. H. Boudali, P. Crouzen, and M. Stoelinga. A Compositional Semantics for Dynamic Fault Trees in Terms of Interactive Markov Chains. In *ATVA*, pages 441–456, 2007.
4. J. Canny. Some Algebraic and Geometric Computations in PSPACE. In *STOC*, pages 460–469, 1988.
5. D. Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.
6. L. Cheung, N. Lynch, R. Segala, and F. Vaandrager. Switched Probabilistic I/O Automata. In *ICTAC*, pages 494–510, 2004.
7. N. Coste, H. Garavel, H. Hermanns, R. Hersemeule, Y. Thonnart, and M. Zidouni. Quantitative Evaluation in Embedded System Design: Validation of Multiprocessor Multithreaded Architectures. In *DATE*, pages 88–89, 2008.
8. N. Coste, H. Hermanns, E. Lantreibecq, and W. Serwe. Towards Performance Prediction of Compositional Models in Industrial GALS Designs. In *CAV*, pages 204–218, 2009.
9. C. Daws. Symbolic and Parametric Model Checking of Discrete-Time Markov Chains. In *ICTAC*, pages 280–294, 2004.
10. L. de Alfaro. The Verification of Probabilistic Systems under Memoryless partial–information Policies is Hard. In *Proceedings of the workshop on probabilistic methods in verification*, 1999.
11. S. Georgievska and S. Andova. Retaining the Probabilities in Probabilistic Testing Theory. In *FOSSACS*, pages 79–93, 2010.
12. P. Gill, W. Murray, and M. Wright. *Practical Optimization*. Academic Press, London, 1981.
13. S. Giro and P. R. D'Argenio. Quantitative Model Checking Revisited: Neither Decidable Nor Approximable. In *FORMATS*, pages 179–194, 2007.
14. S. Giro and P. R. D'Argenio. On the Expressive Power of Schedulers in Distributed Probabilistic Systems. *ENTCS*, 253(3):45–71, 2009.
15. E. M. Hahn, H. Hermanns, B. Wachter, and L. Zhang. PARAM: A Model Checker for Parametric Markov Models. In *CAV*, 2010.
16. E. M. Hahn, H. Hermanns, and L. Zhang. Probabilistic Reachability for Parametric Markov Models. *STTT*, 2010.
17. S. Han. A Globally Convergent Method for Nonlinear Programming. *Journal of Optimization Theory and Applications*, 22, 1977.
18. H. Hermanns. *Interactive Markov Chains: The Quest for Quantified Quality*, volume 2428 of *LNCS*. Springer-Verlag, 2002.
19. G. Lowe. Representing Nondeterministic and Probabilistic Behaviour in Reactive Processes. Technical Report PRG-TR-11-93, Oxford Univ. Comp. Labs, 1993.
20. C. Morgan, A. McIver, K. Seidel, and M. Massink. Refinement-oriented probability for CSP. *Formal aspects of computing*, 8:617–647, 1996.
21. R. Segala. *Modeling and Verification of Randomized Distributed Real–time Systems*. PhD thesis, MIT, 1995.
22. L. Zhang and M. R. Neuhäußer. Model Checking Interactive Markov Chains. In *TACAS*, pages 53–68, 2010.

A Proofs

A.1 Proof of Lemma 1

Proof. For both (i) and (ii) set the assignment ζ, respectively the distributed scheduler η such that for $1 \leq i \leq \#\mathcal{C}$: $\zeta(y^i_\sigma) = \mathcal{I}(\sigma)(\mathcal{P}_i)$ if $A^{en}_{last(\sigma[\mathcal{P}_i]),\mathcal{P}_i} \neq \varnothing$ and $\zeta(x^a_{\sigma[\mathcal{P}_i]}) = \eta_{\mathcal{P}_i}(\sigma[\mathcal{P}_i])(a)$ if $a \in A^{en}_{last(\sigma[\mathcal{P}_i]),\mathcal{P}_i}$. This gives indentical mappings from assignments to distributed schedulers and back for (i) and (ii), which means we can prove both simultaneously.

For a distributed scheduler η and its associated assignment ζ, we now show that $P_\eta(\sigma^\uparrow) = \zeta(\mathbf{P}_k(\hat{s}, \sigma))$ by induction on the length k of σ:

- for paths of length 0 we have that

$$P_\eta(\hat{s}^\uparrow) = 1 = \zeta(\mathbf{P}_0(\hat{s}, \hat{s})) \text{ and, for } s \neq \hat{s},\ P_\eta(s^\uparrow) = 0 = \zeta(\mathbf{P}_0(\hat{s}, s)).$$

- for the inductive step, let the induction hypothesis (IH) be the following: *Given a path $\sigma \in S^t_\mathcal{C}$ of length $k > 0$, for any path σ' of length $k - 1$:*

$$P_\eta(\sigma') = \zeta(\mathbf{P}_{k-1}(\hat{s}, \sigma')).$$

  By case distinction:

  1. For *last*(σ) vanishing we have:

$$\begin{aligned} P_\eta(\sigma^\uparrow) &\overset{\text{Def 10}}{=} \sum_{\sigma=\sigma' as} P_\eta(\sigma'^\uparrow) \cdot \eta(\sigma')(a) \overset{IH}{=} \sum_{\sigma=\sigma' as} \zeta(\mathbf{P}_{k-1}(\hat{s}, \sigma')) \cdot \eta(\sigma')(a) \\ &\overset{\text{Def 8}}{=} \sum_{\sigma=\sigma' as} \zeta(\mathbf{P}_{k-1}(\hat{s}, \sigma')) \cdot \sum_{i=1,e(i,a,\sigma)}^{n} \mathcal{I}(\sigma)(\mathcal{P}_i) \cdot \eta_{\mathcal{P}_i}(\sigma[\mathcal{P}_i])(a) \\ &\overset{\text{Def 13}}{=} \sum_{\sigma=\sigma' as} \zeta(\mathbf{P}_{k-1}(\hat{s}, \sigma')) \cdot \zeta(\mathbf{P}(\sigma', \sigma)) = \zeta(\mathbf{P}_k(\hat{s}, \sigma)). \end{aligned}$$

  2. For *last*(σ) tangible we have:

$$\begin{aligned} P_\eta(\sigma^\uparrow) &\overset{\text{Def 10}}{=} \sum_{\sigma=\sigma' \mu s} P_\eta(\sigma'^\uparrow) \cdot \mu(s) \\ &\underset{IH}{\overset{\text{Def 13}}{=}} \sum_{\sigma=\sigma' \mu s} \zeta(\mathbf{P}_{k-1}(\hat{s}, \sigma')) \cdot \zeta(\mathbf{P}(\sigma', \sigma)) = \zeta(\mathbf{P}_k(\hat{s}, \sigma)). \end{aligned}$$

A.2 Proof of Theorem 1

Proof. For a distributed scheduler η with associated assignment ζ, as defined in Lemma 1, we have $P_\eta(\lozenge^{\leq t}\mathcal{G}) = P_\eta(\bigcup\{\sigma^\uparrow \mid \mathtt{t}(\sigma) \leq t \text{ and } last(\sigma) \in \mathcal{G}\})$.

Recall that the set of paths $\overline{\mathcal{G}} \subset S^t_\mathcal{C}$ is the set of paths that end in a state in $\mathcal{G}$, but do not pass through $\mathcal{G}$ under way. It is obvious that the cylinders induced by these paths do not overlap and their union is the set of all paths reaching $\mathcal{G}$ within t time-units.

We then have:

$$
\begin{aligned}
P_\eta(\Diamond^{\leq t}\mathcal{G}) = P_\eta\Big(\bigcup\{\sigma^\uparrow \mid \sigma \in \overline{\mathcal{G}}\}\Big) &= \sum_{\sigma\in\overline{\mathcal{G}}} P_\eta(\sigma^\uparrow) = \sum_{\sigma\in\overline{\mathcal{G}},|\sigma|=k} \zeta(\mathbf{P}_k(\hat{s},\sigma)) \\
&= \zeta\Big(\sum_{\sigma\in\overline{\mathcal{G}},|\sigma|=k} \mathbf{P}_k(\hat{s},\sigma)\Big) = \zeta(\mathbf{P}(\Diamond\,\overline{\mathcal{G}})).
\end{aligned}
$$

The last equality stems from the fact that a path σ of length k can only be reached for the first time after exactly k steps in $\mathcal{D}$.

We now have a one-to-one correspondence between distributed schedulers of $\mathcal{C}$ and assignments of $\mathcal{D}$ which satisfy (2) that preserves the probability to reach $\mathcal{G}$, respectively $\bar{\mathcal{G}}$. It immediately follows that the infimum and supremum probabilities over these schedulers respectively assignments must be equal.

A.3 Proof of Theorem 2

Proof. We associate strongly distributed schedulers η to assignments ζ following Lemma 1. For the extra variables in V_z we choose

$$
\zeta(z^{i,j}_{\sigma[\mathcal{P}_i],\sigma[\mathcal{P}_j]}) := \begin{cases} \zeta\left(\frac{y^i_\sigma}{y^i_\sigma+y^j_\sigma}\right) & \text{if } \zeta(y^i_\sigma + y^j_\sigma) > 0 \\ 1 & \text{otherwise.} \end{cases}
$$

Note that the value 1 is chosen arbitrarily here.

To prove Theorem 2 we must now show that any assignment that satisfies (3) is associated to a strongly distributed scheduler and that any strongly distributed scheduler is associated to an assigment that satisfies (3).

First, notice that for a path σ ending in a vanishing state and any distinct I/O-IPCs $\mathcal{P}_i$ and $\mathcal{P}_j$ that have immediate actions enabled after σ, we have that if $\zeta(y^i_\sigma) = 0 = \zeta(y^j_\sigma)$ then $z^{i,j}_{\sigma[\mathcal{P}_i],\sigma[\mathcal{P}_j]}(y^i_\sigma + y^j_\sigma) = y^i_\sigma$ holds, regardless the value of $\zeta(z^{i,j}_{\sigma[\mathcal{P}_i],\sigma[\mathcal{P}_j]})$.

Now, consider an assignment ζ with associated distributed scheduler η, and suppose we find two paths σ, σ' as above with $\zeta(y^i_\sigma + y^j_\sigma) \neq 0 \neq \zeta(y^i_{\sigma'} + y^j_{\sigma'})$, $\sigma[\mathcal{P}_i] = \sigma'[\mathcal{P}_i]$ and $\sigma[\mathcal{P}_j] = \sigma'[\mathcal{P}_j]$. Then (3) gives us that:

$$
\zeta\left(\frac{y^i_\sigma}{y^i_\sigma + y^j_\sigma}\right) = \zeta(z^{i,j}_{\sigma[\mathcal{P}_i],\sigma[\mathcal{P}_j]}) = \zeta(z^{i,j}_{\sigma'[\mathcal{P}_i],\sigma'[\mathcal{P}_j]}) = \zeta\left(\frac{y^i_{\sigma'}}{y^i_{\sigma'} + y^j_{\sigma'}}\right)
$$

As $\zeta(y^i_\sigma)$ corresponds to $\mathcal{I}(\sigma)(\mathcal{P}_i)$ in η – and similar correspondences exist for y^j_σ, $y^i_{\sigma'}$, $y^j_{\sigma'}$ – we have that now Equation. (1) holds for η, which means that η is indeed strongly distributed.

Since we have a one-to-one correspondence between distributed schedulers and assigments, this also proves the reverse, that the assigment associated to a strongly distributed scheduler satisfies (3). The remainder of the proof follows that of Theorem 1.