

# Advances on Parameters in Discrete Models with Probabilities

ANR PACS Deliverable 5

Benoît Delahaye      Didier Lime      Laure Petrucci

June 28, 2019

## 1 Context and objectives

In real-life applications, probabilities are often used as a building block that allows abstracting from physical constraints or unknown environments. While parameters are used in order to *reason* in a formal way on unknown (or purposefully under-specified) aspects of a system, probabilities allow for abstracting some aspects in order to “simplify” the system under study: these aspects are *approximated* using probability distributions, often based on partial information obtained through empirical analysis. In a way, one could say that parameters target aspects of the system that need to be *studied* while probabilities target aspects of the system that are inconsequential or too complex to model in a formal way. By their complementary nature, probabilistic and parametric modelling/analysis can thus be combined in several ways. In Task 5, we focus on discrete probabilistic models with parameters that can either range on the discrete aspects of the system or on the probabilities themselves.

In the following, we therefore present our main contributions along these two axes. Worth to notice, a postdoctoral student (Paulin Fournier) has been hired to work on this task, and has been involved in several of the contributions presented hereafter. The main publications associated to those contributions are attached at the end of the document.

## 2 Probabilistic Models with Discrete Parameters

Our first contributions target the field of probabilistic models with discrete parameters. In a first line of work [[ADL17](#), [ADL19](#)], we have extended the Event-B modelling formalism, a modelling based on refinement which notably allows to describe parametric aspects of a system, to the probabilistic setting. In a second line of work [[DEB17](#)], we have developed a new method based on Statistical Model Checking in order to parametrize existing probabilistic models coming from other scientific fields. This method has in particular been successfully applied to the model of a jellyfish organism (article under review at the moment).

## 2.1 Probabilistic Event-B

Our first contribution has been to extend a modelling formalism called Event-B to the probabilistic setting. Event-B [Abr10] is a proof-based formal method used for discrete systems modelling. It is equipped with *Rodin* [ABH<sup>+</sup>10], an open toolset for modelling and proving systems. This toolset can easily be extended, which makes of Event-B a good candidate for introducing probabilistic reasoning in a proof-based modelling formalism. The development process in Event-B is based on refinement: systems are typically developed progressively using an ordered sequence of models, where each model contains more details than its predecessor. Refinement allows a step-by-step description of the behaviour of systems, providing an efficient way to give a detailed description of their behaviour. Notably, events in Event-B can be parameterized using discrete parameters. A few attempts had been previously made in order to incorporate probabilistic aspects in Event-B: In [MHA05], Abrial *et al.* have summarised the difficulties of embedding probabilities into Event-B. This paper suggests that probabilities need to be introduced as a refinement of *non-determinism*. In Event-B, non-determinism occurs in several places such as the choice between enabled events in a given state, the choice of the parameter values in a given event, and the choice of the value given to a variable through some non-deterministic assignments. To the best of our knowledge, the existing works on extending Event-B with probabilities have mostly focused on refining non-deterministic assignments into probabilistic assignments. Other sources of non-determinism have been left untouched. In [HH07], Hallersted *et al.* propose to focus on a qualitative aspect of probability. They refine non-deterministic assignments into *qualitative* probabilistic assignments where the actual probability values are not specified, and adapt the Event-B semantics and proof obligations to this new setting. In [Yil10], the same authors study the refinement of qualitative probabilistic Event-B models and propose a tool support inside Rodin. Other works [TTL09, TTL15, TTL10] have extended this approach by refining non-deterministic assignments into *quantitative* probabilistic assignments where, unlike in [HH07], the actual probability values are specified. This new proposition is then exploited in order to assess several system properties such as reliability and responsiveness.

In this setting, we have pursued these works by proposing a probabilistic extension of Event-B that allows to introduce probabilities in other places than probabilistic assignments. In particular, we have allowed to replace every source of non-determinism in Event-B (assignments, choice between events and parameter values) with probabilistic choices, as well as to produce mixed models that combine non-determinism in some places and probabilities in others. We have also studied the role of refinement and probabilistic refinement in this new setting and proposed operations for automatically transforming any (part of a) non-deterministic model with a probabilistic one. This line of work has led to several publications, in an international conference ([ADL17]) and in an international journal ([ADL19]).

## 2.2 Parameter synthesis using SMC

In a different setting, we have taken advantage of existing collaborations with colleagues from other scientific fields to propose a new method for parameterizing formal models of natural systems using Statistical Model Checking.

The modelling of the biological carbon pump (BCP) in the global ocean is still at its infancy because its biological component, which is known to be complex, is

extremely simplified. In such a context, models that require many unknown and non-linear parameters, allowing to scale from physiological features to ecosystems, appear as great tools to account for this complexity. Indeed, modelling the BCP must integrate parameters to fit both physiological and ecological behaviours that are together very difficult to determine. In particular, such a difficulty occurs for modelling the jellyfish *Pelagia noctiluca*. This gelatinous zooplanktonic species have high abundance in the Mediterranean Sea which could significantly contribute to the BCP via the production of detritus or faecal pellets that impact vertical fluxes of biogenic and particulate organic matter (POM). However, in spite of their important abundance and biomass, this planktonic compound remains vaguely represented in biogeochemical models because of global uncertainties about its ecophysiology. To overcome this issue, we proposed for the first time the use of the *Statistical Model Checking Engine* (SMCE) approach. Already standard in engineering, this probability-based computational framework considers sets of parameters as a whole and suggests a combinatorial search to decipher parameters that fit altogether distinct experimental data. Here, the SMCE was applied simultaneously on both exhaustive laboratory-culturing experiments of *P. noctiluca* and in situ cruise observations. Contrary to other parameter inference techniques, the SMCE, built on generic gelatinous ecophysiological constraints, seeks for sets of parameters that fit both local and global knowledge while considering uncertainties. While relying on intensive computing resources, SMCE allows an accurate estimation of ecophysiological rates of *P. noctiluca* as a function of temperature and prey concentrations, and under varying parameter inferences. In particular, the proposed model correctly reproduced jellyfish growth and degrowth in laboratory and environmental condition thanks to five fundamental physiological processes: filtration, respiration, reproduction, excretion and egestion. Moreover, besides accurate simulations, SMCE allowed to reason on multi-scale knowledge by showing necessary uncertainties on parameters to combine both laboratory and in situ conditions, which is of great help to underline missing knowledge regarding physiological processes. In the broader BCP context, this modelling framework was completed with new observations of remineralization rates and sinking speeds of particulate organic carbon (POC) produced by *P. noctiluca* after digestion. Those POC aggregates have high sinking speed ( $384 - 1329 m.d^{-1}$ ) and low remineralization rates ( $0.034 d^{-1}$ ). Such rates, coupled with jellyfish abundances in the northwestern Mediterranean Sea, allowed to estimate carbon fluxes between the jellyfish and its preys as well as its contribution to carbon export.

### 3 Probabilistic Models with Probabilistic Parameters

In another setting, we have also contributed to the field of probabilistic parametric models with parameters ranging on transition probabilities. On the one hand, we have proposed a new modelling formalism called *parametric Interval Markov Chains* [Del15], that extends Interval Markov Chains by allowing the endpoint of the intervals representing transition probabilities to be parametric. In this context, we have proposed methods for deciding the consistency of such models as well as for synthesising the set of parameter values ensuring such properties.

On the other hand, we have developed new formal verification techniques adapted to probabilistic systems with parameters ranging on the probabilities. This new method, called parametric Statistical Model Checking, is an extension of Sta-

tistical Model Checking to the parametric context, allowing to produce functions of the parameters that represent the (parametric) probability of satisfying given properties.

### 3.1 Parametric Interval Markov Chains

Discrete time Markov chains (MCs for short) are a standard probabilistic modelling formalism that has been extensively used in the literature to reason about software [WT94] and real-life systems [HDR10]. However, when modelling real-life systems, the exact value of transition probabilities may not be known precisely. Several formalisms abstracting MCs have therefore been developed. Parametric Markov chains [AHV93b] (pMCs for short) extend MCs by allowing parameters to appear in transition probabilities. In this formalism, parameters are variables and transition probabilities may be expressed as polynomials or rational functions over these variables. A given pMC represents a potentially infinite set of MCs, obtained by replacing each parameter by a given value. pMCs are particularly useful to represent systems where dependencies between transition probabilities are required. Indeed, a given parameter may appear in several distinct transition probabilities, which requires that the same value is given to all its occurrences. Interval Markov chains [JL91] (IMCs for short) extend MCs by allowing precise transition probabilities to be replaced by intervals, but cannot represent dependencies between distinct transitions. IMCs have mainly been studied with three distinct semantics interpretations. Under the *once-and-for-all* semantics, a given IMC represents a potentially infinite number of MCs where transition probabilities are chosen inside the specified intervals while keeping the same underlying graph structure. The *interval-Markov-decision-process* semantics (IMDP for short), such as presented in [CSH08, SVA06], does not require MCs to preserve the underlying graph structure of the original IMC but instead allows a finite “unfolding” of the original graph structure: new probability values inside the intervals can be chosen each time a state is visited. Finally, the *at-every-step* semantics, which was the original semantics given to IMCs in [JL91], does not preserve the underlying graph structure too while allowing to “aggregate” and “split” states of the original IMC in the manner of probabilistic simulation.

Model-checking algorithms and tools have been developed in the context of pMCs [DJJ<sup>+</sup>15, HHWZ10, KNP11] and IMCs with the once-and-for-all and the IMDP semantics [CK15, BLW13]. State of the art tools [DJJ<sup>+</sup>15] for pMC verification compute a rational function on the parameters that characterises the probability of satisfying a given property, and then use external tools such as SMT solving [DJJ<sup>+</sup>15] for computing the satisfying parameter values. For these methods to be viable in practice, the allowed number of parameters is quite limited. On the other hand, the model-checking procedure for IMCs presented in [BLW13] is adapted from machine learning and builds successive refinements of the original IMCs that optimise the probability of satisfying the given property. This algorithm converges, but not necessarily to a global optimum. It is worth noticing that existing model checking procedures for pMCs and IMCs strongly rely on their underlying graph structure (i. e., respect the once-and-for-all semantics). However, in [CSH08] the authors perform model checking of  $\omega$ -PCTL formulas on IMCs w.r.t. the IMDP semantics and they show that model checking of LTL formulas can be solved for the IMDP semantics by reduction to the model checking problem of  $\omega$ -PCTL on IMCs with the IMDP semantics. For all that, to the best of our knowledge, no solutions for model-checking IMCs with the at-every-step semantics have been proposed yet.

In a first line of work [Del15, DLP16], we have introduced a new formalism called Parametric interval Markov chains (pIMCs for short), that generalise both IMCs and pMCs by allowing parameters to appear in the endpoints of the intervals specifying transition probabilities, and studied basic properties such as consistency and reachability.

In a second line of work [BDL<sup>+</sup>17, BDF<sup>+</sup>18], we study the difference between pIMCs and other Markov Chain abstractions models and investigate three semantics for IMCs: once-and-for-all, interval-Markov-decision-process, and at-every-step. In particular, we prove that all three semantics agree on the maximal/minimal reachability probabilities of a given IMC. We then investigate solutions to several parameter synthesis problems in the context of pIMCs – consistency, qualitative reachability and quantitative reachability – that rely on constraint encodings. Finally, we propose a prototype implementation of our constraint encodings with promising results.

### 3.2 Parametric SMC

As made clear in the previous sections, modelling and abstracting are widely accepted as crucial steps in the understanding and study of real-life systems. In many cases, it is necessary to incorporate probabilities in the models to cope with uncertainty, to abstract complex behaviour, or to introduce randomness. Markov chains and Markov decision processes, in particular, have been widely studied.

Though exact methods are known for such models they usually require solving huge equation systems, and therefore have scalability issues with the biggest models. A way to avoid this complexity is to consider approximation techniques through simulation. In particular, Monte Carlo simulation techniques allow to infer the real behaviour of the system via independent simulations up to a computable precision.

The values given to probabilistic transitions can have a huge impact on the behaviour of the system. In the early stages of development, it may therefore be useful to have an insight on how the values of transition probabilities affect the system in order to be able to set the best value in terms of convergence speed for example. To this purpose, parametric Markov chains have been introduced in [AHV93a]. They allow to replace the probability values given to transitions by parameter variables, and therefore to be able to give guarantees on the system for all possible values of the parameters.

The aim of this line of work is to apply Monte Carlo simulation to parametric Markov chains in order to approximate the probability of the considered property as a polynomial function of the parameters. In addition, we also derive a confidence interval on the obtained probabilities as a polynomial function of the parameters. Aside from using parameterized models, which comes in particular with better flexibility in the modelling, robustness of the results, and usability at the earliest stages of conception, the expected benefits of our new approach are largely those of such simulation techniques for non-parameterized Markov chains: better scalability through a reduced memory footprint, a complexity that is largely independent of the model complexity (be it in terms of size of the state-space, or of features used as long as they are executable). More specific to our approach, since we derive polynomial function approximations, where exact methods lead to rational functions, these results should be easier to post-process. Finally the complexity of our approach is largely independent of the number of parameters.

This technique has been applied to a real-life case study in order to demonstrate

its usefulness and scalability. This has led to a publication in an international conference [BAD<sup>+</sup>19].

## References


- [ABH<sup>+</sup>10] Jean-Raymond Abrial, Michael Butler, Stefan Hallerstede, Thai Son Hoang, Farhad Mehta, and Laurent Voisin. Rodin: an open toolset for modelling and reasoning in event-b. *International journal on software tools for technology transfer*, 12(6):447–466, 2010.
- [Abr10] Jean-Raymond Abrial. *Modeling in Event-B: system and software engineering*. Cambridge University Press, 2010.
- [ADL17] Mohamed Amine Aouadhi, Benoît Delahaye, and Arnaud Lanoix. Moving from event-b to probabilistic event-b. In *Proceedings of the Symposium on Applied Computing, SAC 2017, Marrakech, Morocco, April 3-7, 2017*, pages 1348–1355, 2017.
- [ADL19] Mohamed Amine Aouadhi, Benoît Delahaye, and Arnaud Lanoix. Introducing probabilistic reasoning within event-b. *Software and System Modeling*, 18(3):1953–1984, 2019.
- [AHV93a] R. Alur, T. A. Henzinger, and M. Y. Vardi. Parametric real-time reasoning. In *STOC*, pages 592–601. ACM, 1993.
- [AHV93b] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. Parametric real-time reasoning. In *Symposium on Theory of Computing*, pages 592–601. ACM, 1993.
- [BAD<sup>+</sup>19] Ran Bao, J. Christian Attiogbé, Benoît Delahaye, Paulin Fournier, and Didier Lime. Parametric statistical model checking of UAV flight plan. In *Formal Techniques for Distributed Objects, Components, and Systems - 39th IFIP WG 6.1 International Conference, FORTE 2019, Held as Part of the 14th International Federated Conference on Distributed Computing Techniques, DisCoTec 2019, Kongens Lyngby, Denmark, June 17-21, 2019, Proceedings*, pages 57–74, 2019.
- [BDF<sup>+</sup>18] Anicet Bart, Benoît Delahaye, Paulin Fournier, Didier Lime, Eric Monfroy, and Charlotte Truchet. Reachability in parametric interval markov chains using constraints. *Theor. Comput. Sci.*, 747:48–74, 2018.
- [BDL<sup>+</sup>17] Anicet Bart, Benoît Delahaye, Didier Lime, Eric Monfroy, and Charlotte Truchet. Reachability in parametric interval Markov chains using constraints. In *International Conference on Quantitative Evaluation of Systems*, volume 10503 of *LNCS*, pages 173–189. Springer, 2017.
- [BLW13] Michael Benedikt, Rastislav Lenhardt, and James Worrell. LTL model checking of interval Markov chains. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 7795 of *LNCS*, pages 32–46. Springer, 2013.

- [CK15] Souymodip Chakraborty and Joost-Pieter Katoen. Model checking of open interval Markov chains. In *International Conference on Analytical and Stochastic Modelling Techniques and Applications*, volume 9081 of LNCS, pages 30–42, Cham, 2015. Springer.
- [CSH08] Krishnendu Chatterjee, Koushik Sen, and Thomas A. Henzinger. Model-checking omega-regular properties of interval Markov chains. In *International Conference on Foundations of Software Science and Computational Structures*, volume 4962 of LNCS, pages 302–317. Springer, 2008.
- [DEB17] Benoît Delahaye, Damien Eveillard, and Nicholas Bouskill. On the power of uncertainties in microbial system modeling: No need to hide them anymore. *mSystems*, 2(6), 2017.
- [Del15] Benoît Delahaye. Consistency for parametric interval Markov chains. In *International Workshop on Synthesis of Complex Parameters*, volume 44 of OASICS, pages 17–32. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.
- [DJJ<sup>+</sup>15] Christian Dehnert, Sebastian Junges, Nils Jansen, Florian Corzilius, Matthias Volk, Harold Bruintjes, Joost-Pieter Katoen, and Erika Ábrahám. Prophesy: A probabilistic parameter synthesis tool. In *International Conference on Computer Aided Verification*, volume 9207 of LNCS, pages 214–231, Cham, 2015. Springer.
- [DLP16] Benoît Delahaye, Didier Lime, and Laure Petrucci. Parameter synthesis for parametric interval Markov chains. In *International Conference on Verification, Model Checking, and Abstract Interpretation*, volume 9583 of LNCS, pages 372–390. Springer, 2016.
- [HDR10] Dirk Husmeier, Richard Dybowski, and Stephen Roberts. *Probabilistic Modeling in Bioinformatics and Medical Informatics*. Springer Publishing Company, Incorporated, 2010.
- [HH07] Stefan Hallerstede and Thai Son Hoang. Qualitative probabilistic modelling in event-b. In *Integrated Formal Methods*, pages 293–312. Springer, 2007.
- [HHWZ10] E. M. Hahn, H. Hermanns, B. Wachter, and L. Zhang. PARAM: A model checker for parametric Markov models. In *CAV*, volume 6174 of *Lecture Notes in Computer Science*, pages 660–664. Springer, 2010.
- [JL91] Bengt Jonsson and Kim Guldstrand Larsen. Specification and refinement of probabilistic processes. In *Symposium on Logic in Computer Science*, pages 266–277. IEEE, 1991.
- [KNP11] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *International Conference on Computer Aided Verification*, volume 6806 of LNCS, pages 585–591. Springer, 2011.
- [MHA05] Carroll Morgan, Thai Son Hoang, and Jean-Raymond Abrial. The challenge of probabilistic event b—extended abstract—. In *ZB 2005: Formal Specification and Development in Z and B*, pages 162–171. Springer, 2005.

- [SVA06] Koushik Sen, Mahesh Viswanathan, and Gul Agha. Model-checking Markov chains in the presence of uncertainties. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 3920 of *LNCS*, pages 394–410, Berlin, Heidelberg, 2006. Springer.
- [TTL09] Anton Tarasyuk, Elena Troubitsyna, and Linas Laibinis. Reliability assessment in event-b development. *NODES 09*, page 11, 2009.
- [TTL10] Anton Tarasyuk, Elena Troubitsyna, and Linas Laibinis. Towards probabilistic modelling in event-b. In *Integrated Formal Methods*, pages 275–289. Springer, 2010.
- [TTL15] Anton Tarasyuk, Elena Troubitsyna, and Linas Laibinis. Integrating stochastic reasoning into event-b development. *Formal Aspects of Computing*, 27(1):53–77, 2015.
- [WT94] James A Whittaker and Michael G Thomason. A Markov chain model for statistical software testing. *IEEE Transactions on Software engineering*, 20(10):812–824, 1994.
- [Yil10] Emre Yilmaz. *Tool support for qualitative reasoning in Event-B*. PhD thesis, Master Thesis ETH Zürich, 2010, 2010.



# Introducing probabilistic reasoning within Event-B

Mohamed Amine Aouadhi<sup>1</sup> · Benoît Delahaye<sup>1</sup>  · Arnaud Lanoix<sup>1</sup>

Received: 1 March 2017 / Revised: 5 September 2017 / Accepted: 17 September 2017 / Published online: 9 October 2017  
© Springer-Verlag GmbH Germany 2017

**Abstract** Event-B is a proof-based formal method used for discrete systems modelling. Several works have previously focused on the extension of Event-B for the description of probabilistic systems. In this paper, we propose an extension of Event-B that allows designing fully probabilistic systems as well as systems containing both probabilistic and non-deterministic choices. Compared to existing approaches which only focus on probabilistic assignments, our approach allows expressing probabilistic choices in all places where non-deterministic choices originally appear in a standard Event-B model: in the choice between enabled events, event parameter values and in probabilistic assignments. Furthermore, we introduce novel and adapted proof obligations for the consistency of such systems and introduce two key aspects to incremental design: probabilisation of existing events and refinement through the addition of new probabilistic events. In particular, we provide proof obligations for the almost-certain convergence of a set of new events, which is a required property in order to prove standard refinement in this context. Finally, we propose a fully detailed case study, which we use throughout the paper to illustrate our new constructions.

**Keywords** Event-B · Probabilistic systems · Markov chains

## 1 Introduction

As systems become more and more complex, with randomised algorithms [30], probabilistic protocols [3] or failing components, it is necessary to add new modelling features in order to take into account complex system properties such as reliability [39], responsiveness [14, 38], continuous evolution and energy consumption. One of these features is probabilistic reasoning to introduce uncertainty in a model or to mimic randomised behaviour. Probabilistic modelling formalisms have therefore been developed in the past, mainly extending automata-based formalisms [32, 34]. Abstraction [16, 26], refinement [25] and model-checking algorithms [8, 11] have been successfully studied in this context. However, the introduction of probabilistic reasoning in proof-based modelling formalisms has been, to the best of our knowledge, quite limited [5, 9, 17, 18, 21, 23, 24, 33]. Translations from proof-based models are always possible. However, the use of automata-based verification in this context is inconvenient due to the state space explosion in the translation.

Event-B [1] is a proof-based formal method used for discrete systems modelling. It is equipped with *Rodin* [2], an open toolset for modelling and proving systems. This toolset can easily be extended, which makes Event-B a good candidate for introducing probabilistic reasoning in a proof-based modelling formalism. The development process in Event-B is based on refinement: systems are typically developed progressively using an ordered sequence of models, where each model contains more details than its predecessor. Refinement allows a step-by-step description of the behaviour of systems, providing an efficient way to give a detailed description of their behaviour.

So far, several research works have focused on the extension of Event-B to allow the expression of probabilistic information in Event-B models. In [29], Abrial et al. have

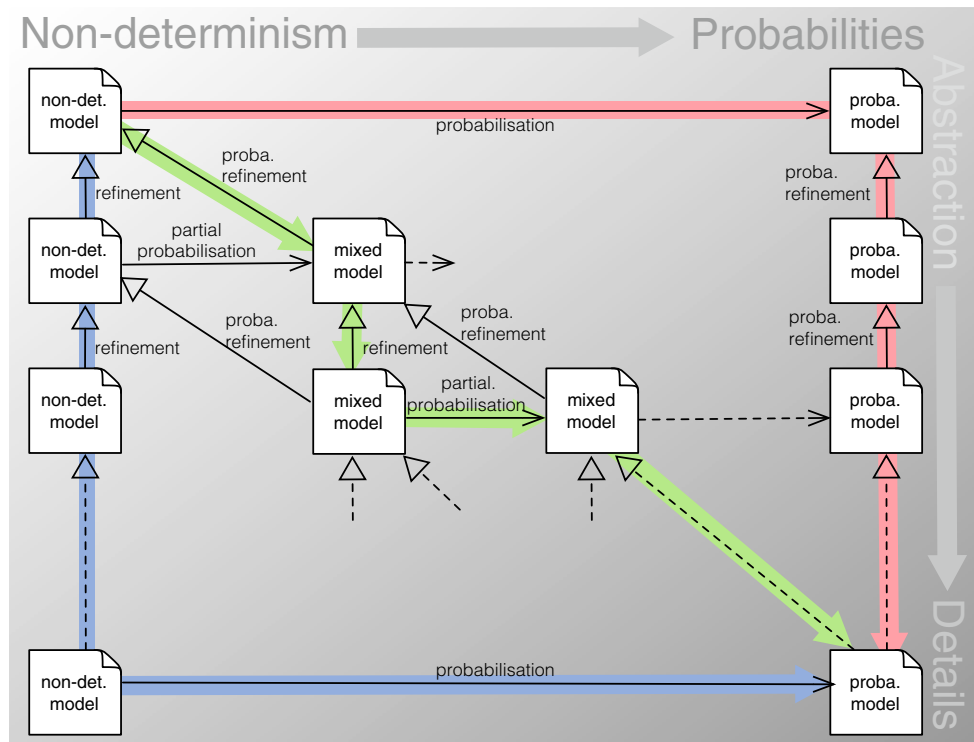
---

Communicated by Dr. Juergen Dingel.

---

Benoît Delahaye  
benoit.delahaye@univ-nantes.fr

<sup>1</sup> LS2N UMR CNRS 6004, University of Nantes, Nantes, France



**Fig. 1** How to introduce probabilities within Event-B?

summarised the difficulties in embedding probabilities into Event-B. This paper suggests that probabilities need to be introduced as a refinement of *non-determinism*. In Event-B, non-determinism occurs in several places such as the choice between enabled events in a given state, the choice of the parameter values in a given event, and the choice of the value given to a variable through some non-deterministic assignments. To the best of our knowledge, the existing works on extending Event-B with probabilities have mostly focused on refining non-deterministic assignments into probabilistic assignments. Other sources of non-determinism have been left untouched. In [19], Hallerstede et al. proposed to focus on a qualitative aspect of probability. They refined non-deterministic assignments into *qualitative* probabilistic assignments where the actual probability values are not specified, and adapted the Event-B semantics and proof obligations to this new setting. In [40], the same authors studied the refinement of qualitative probabilistic Event-B models and proposed a tool support inside Rodin. Other works [35–37] have extended this approach by refining non-deterministic assignments into *quantitative* probabilistic assignments where, unlike in [19], the actual probability values are specified. This new proposition is then exploited in order to assess several system properties such as reliability and responsiveness.

Unfortunately, other sources of non-determinism than assignments have been left untouched, although the authors argue that probabilistic choice between events or parameter

values can be achieved by transformations of the models that embed these choices inside probabilistic assignments. While this is unarguably true, such transformations are not trivial and greatly impede the understanding of Event-B models. Moreover, these transformations would need to be included in the refinement chain when designers need it, which would certainly be counterintuitive to engineers.

In this paper, we extend these works by proposing a probabilistic extension of Event-B and presenting some ways of introducing probabilistic reasoning within Event-B. As the design process within Event-B is based on refinement, we propose to provide a standard description of a system by a set of models related by refinement. According to this modelling process, probabilities can be introduced in several manners that are illustrated in Fig. 1.

In this figure, the vertical axis represents the introduction of more details in the model, while the horizontal axis represents the introduction of probabilistic information. Several types of models stand out:

- Models on the left side of the picture are *standard* Event-B models that only contain non-deterministic choices but no probabilistic information;
- Models on the right side of the picture are *fully probabilistic* Event-B models where all choices are probabilistic;
- Models in the centre of the picture are *mixed* Event-B models, where both non-deterministic and probabilistic choices are present.

Depending on the system that is being developed, the development process could always stay on the left side (when the system does not have any probabilistic aspect) and therefore end in the bottom left of the picture, or the development process could move to the right side and either end in the bottom right when the system is fully probabilistic or in the middle when the system contains both probabilistic and non-deterministic aspects.

In any case, there are many ways both to add standard (non-deterministic) details in the model and to add probabilistic information. Figure 1 provides three generic development processes (green, red and blue), which we detail below.

Assuming the model under development is fully probabilistic, one could consider starting with an abstract non-deterministic version of the model and then progressively refining it in a standard way until a satisfying level of details is achieved. Once enough details have been introduced, all non-deterministic choices can be refined into probabilistic choices in one shot (this last step is called *probabilisation*), as depicted in blue in Fig. 1.

Obviously, one could also consider starting with the probabilisation step, therefore obtaining an abstract fully probabilistic model. From this model, details can then be introduced through *probabilistic refinement*, as depicted in red in Fig. 1. While the fully probabilistic counterpart of the standard Event-B refinement still eludes us at this point, we, nevertheless, propose some restricted refinement steps for fully probabilistic systems through context refinement and the introduction of new probabilistic events.

Finally, the designer could decide to interleave the introduction of new details in the model with the introduction of probabilistic information. In this context, intermediate models are mixed models and one has to consider the standard refinement of mixed models, the *partial probabilisation* operation (that only turns some of the non-deterministic choices into probabilistic choices), the introduction of probabilistic events in a standard (non-deterministic) model and the introduction of standard (non-deterministic) events in a probabilistic or mixed model. That last development process is depicted in green in Fig. 1.

In this paper, we therefore provide the scientific foundations in order to allow all the design possibilities presented above in the Event-B framework. We therefore propose some new syntactic elements for writing probabilistic and mixed Event-B models in the Event-B framework. The consistency of such models is then expressed, as in standard Event-B, in terms of proof obligations. In order to prove the correctness of our approach, we show that the operational semantics of such models can be expressed in terms of (potentially infinite-state) Markov chains—for fully probabilistic models—and (potentially infinite-state) Markov decision processes—for

mixed Event-B models—therefore resembling the LTS operational semantics of standard Event-B models.

As explained above, we propose several operations for introducing details and probabilistic aspects in standard/mixed/probabilistic Event-B models. In particular, we focus on the introduction of new probabilistic events in a given model. In the standard Event-B setting, *convergence* is a required property for proving a refinement step as soon as new events are introduced in the model. The counterpart property in the probabilistic setting is *almost-certain convergence*, which has already been studied in [21,28] in the context of probabilistic programs and the standard B method, and in [19,22] in the context of non-deterministic Event-B models with only probabilistic assignments. While the authors of [21,28] propose hypotheses under which probabilistic while loops almost-certainly converge, these hypotheses cannot be directly applied to our setting as they would require a translation from the probabilistic Event-B setting to the standard probabilistic B setting which is not trivial. In addition, some new conditions would need to be exhibited in the probabilistic Event-B setting that ensure that the hypotheses on the standard probabilistic B setting are met. In the paper, we instead choose to exhibit conditions at the probabilistic Event-B level and show that these conditions ensure almost-certain convergence of the operational semantics of the model. On the other hand, [19,22] focused on almost-certain convergence at the probabilistic Event-B level for probabilistic Event-B models where probabilities only appear inside probabilistic assignments, but cannot appear in the choice between enabled events or in the choice of parameter values. However, we show that the proof obligations developed in this context are not sufficient for our models. We therefore propose new sufficient conditions, expressed in terms of proof obligations, for the almost-certain convergence of a set of fully probabilistic events. While the conditions we exhibit are more constrained than those from [19] concerning events and parameters, they are also less restrictive concerning probabilistic assignments.

Finally, some of the results presented in this paper are being implemented in a prototype plugin for Rodin, which we briefly present in the conclusion. Note that this paper is an extension of [4], where we have first introduced a fully probabilistic extension of Event-B. Compared to [4], which only focused on fully probabilistic Event-B, this paper provides a study of *mixed* Event-B models, where non-deterministic choices and probabilistic choices can be present, and explains how probabilistic information can be introduced during the development process in Event-B through probabilisation of standard events and introduction of new probabilistic events in all kind of Event-B models (standard, mixed, probabilis-

tic). This paper also provides new discussions, figures and examples illustrating in better details contributions from [4].

**Outline** The paper is structured as follows. Section 2 presents the scientific background of this paper in terms of transition systems and Event-B, and Sect. 3 introduces our running case study: a simple peer-to-peer protocol. In Sect. 4, we introduce the syntax of fully probabilistic models and then investigate the consistency and operational semantics of such models. Section 5 then provides a similar study in the context of mixed Event-B models. The introduction of probabilistic detailed information in standard/mixed/probabilistic Event-B models is explained in Sects. 6 and 7. Finally, Sect. 8 presents our extension to the Rodin platform and concludes the paper.

## 2 Background

In this section, we introduce notations for (probabilistic) transition systems as well as basic elements of the Event-B method that will be used throughout the paper.

### 2.1 Transition systems

In the following, let  $Dist(S)$  denote the set of distributions over a given set  $S$ , i.e. the set of functions  $\delta : S \rightarrow [0, 1]$  such that  $\sum_{s \in S} \delta(s) = 1$ .

**Labelled transition system** [8] A labelled transition system (LTS for short) is a tuple  $\mathcal{M}=(S, Acts, \rightarrow, s_0, AP, L)$  where  $S$  is a set of states,  $Acts$  is a set of actions,  $\rightarrow \subseteq S \times Acts \times S$  is a transition relation,  $s_0 \subseteq S$  is the initial state,  $AP$  is a set of atomic propositions, and  $L : S \rightarrow 2^{AP}$  is a labelling function.

**Probabilistic labelled transition system** [8] A probabilistic labelled transition system (PLTS for short) is a tuple  $\mathcal{M}=(S, Acts, P, s_0, AP, L)$  where  $S$  is a set of states,  $Acts$  is a set of actions,  $s_0 \in S$  is the initial state,  $AP$  is a set of atomic propositions,  $L : S \rightarrow AP$  is a labelling function, and  $P : S \times Acts \times S \rightarrow [0, 1]$  is the transition probability function. If for each state  $s \in S$  we have  $\sum_{s' \in S, a \in Acts} P(s, a, s') = 1$ , then the PLTS is a *discrete-time Markov chain* (DTMC for short).

**Markov decision process** [8] A Markov decision process (MDP for short) is a tuple  $\mathcal{M}=(S, Acts, P, l_{init}, AP, L)$  where  $S$  is a countable set of states,  $Acts$  is a set of actions,  $P \in S \times Acts \times Dist(S)$  is the transition probability function,  $l_{init} \in Dist(S)$  is the initial distribution,  $AP$  is a set of atomic propositions, and  $L : S \rightarrow 2^{AP}$  is a labelling function.

An action  $\alpha$  is enabled in state  $s$  if and only if there exists a distribution  $\delta \in Dist(S)$  such that  $(s, \alpha, \delta) \in P$ .

### 2.2 Event-B

Event-B [1] is a formal method used for the development of complex discrete systems. Systems are described in Event-B by means of models. For the sake of simplicity, we assume in the rest of the paper that an Event-B model is expressed by a tuple  $M=(\bar{v}, I(\bar{v}), V(\bar{v}), Evts, Init)$  where  $\bar{v} = \{v_1 \dots v_n\}$  is a set of variables,  $I(\bar{v})$  is an invariant,  $V(\bar{v})$  is an (optional) variant used for proving the convergence of the model,  $Evts$  is a set of events, and  $Init \in Evts$  is the initialisation event. The invariant  $I(\bar{v})$  is a conjunction of predicates over the variables of the system specifying properties that must always hold.

```

ei ≐
any  $\bar{t}$  where
  Gi( $\bar{t}, \bar{v}$ )
then
  Si( $\bar{t}, \bar{v}$ )
end
```

**Events** An event has the following structure (see Figure on the side), where  $e_i$  is the name of the event,  $\bar{t} = \{t_1 \dots t_n\}$  represents the (optional) set of parameters of the event,  $G_i(\bar{t}, \bar{v})$  is the (optional) guard of the event and  $S_i(\bar{t}, \bar{v})$  is the action of the event. An event is *enabled* in a given valuation of the variables (also called a configuration) if and only if there exists a parameter valuation such that its guard  $G_i(\bar{t}, \bar{v})$  is satisfied in this context. The action  $S_i(\bar{t}, \bar{v})$  of an event may contain several assignments that are executed in parallel. An assignment can be expressed in one of the following forms:

- **Deterministic assignment:**  $x := E(\bar{t}, \bar{v})$  means that the expression  $E(\bar{t}, \bar{v})$  is assigned to the variable  $x$ .
- **Predicate (non-deterministic) assignment:**  $x :| Q(\bar{t}, \bar{v}, x, x')$  means that the variable  $x$  is assigned a new value  $x'$  such that the predicate  $Q(\bar{t}, \bar{v}, x, x')$  is satisfied.
- **Enumerated (non-deterministic) assignment:**  $x := \{E_1(\bar{t}, \bar{v}) \dots E_n(\bar{t}, \bar{v})\}$  means that the variable  $x$  is assigned a new value taken from the set  $\{E_1(\bar{t}, \bar{v}) \dots E_n(\bar{t}, \bar{v})\}$ .

**Before–after predicate and semantics** The formal semantics of an assignment is described by means of a before–after predicate (BA)  $Q(\bar{t}, \bar{v}, x, x')$ . This BA describes the relationship between the values of the variable before ( $x$ ) and after ( $x'$ ) the execution of an assignment. Before–after predicates are as follows:

- the BA of a deterministic assignment is  $x' = E(\bar{t}, \bar{v})$ ,
- the BA of a predicate assignment is  $Q(\bar{t}, \bar{v}, x, x')$ , and

- the BA of an enumerated assignment is  $x' \in \{E_1(\bar{t}, \bar{v}) \dots E_n(\bar{t}, \bar{v})\}$ .

Recall that the action  $S_j(\bar{t}, \bar{v})$  of a given event  $e_j$  may contain several assignments that are executed in parallel. Assume that  $v_1 \dots v_i$  are the variables assigned in  $S_j(\bar{t}, \bar{v})$  – variables  $v_{i+1} \dots v_n$  are thus not modified – and let  $Q(\bar{t}, \bar{v}, v_1, v'_1) \dots Q(\bar{t}, \bar{v}, v_i, v'_i)$  be their corresponding BA. Then, the BA  $S_j(\bar{t}, \bar{v}, \bar{v}')$  of the event action  $S_j(\bar{t}, \bar{v})$  is:

$$S_j(\bar{t}, \bar{v}, \bar{v}') \cong Q(\bar{t}, \bar{v}, v_1, v'_1) \wedge \dots \wedge Q(\bar{t}, \bar{v}, v_i, v'_i) \wedge (v'_{i+1} = v_{i+1}) \wedge \dots \wedge (v'_n = v_n)$$

*Proof obligations* The consistency of a standard Event-B model is characterised by means of *proof obligations* (POs) which must be discharged. Discharging all the necessary POs allows to prove that the model is sound with respect to some behavioural semantics. Formal definitions of standard Event-B POs are given in [1]. In the following, we only recall the most important of them: **(event/INV)** for *invariant preservation*, which states that the invariant still holds after the execution of each event in the Event-B model  $M$ . Given an event  $e_i$  with guard  $G_i(\bar{t}, \bar{v})$  and action  $S_i(\bar{t}, \bar{v})$ , this PO is expressed as follows:

$$I(\bar{v}) \wedge G_i(\bar{t}, \bar{v}) \wedge S_i(\bar{t}, \bar{v}, \bar{v}') \vdash I(\bar{v}') \quad \text{(event/INV)}$$

*Operational semantics* As established in [10], the semantics of an Event-B model  $M = (\bar{v}, I(\bar{v}), V(\bar{v}), \text{Evts}, \text{Init})$  is expressed in terms of a labelled transition system (LTS)  $\mathcal{M} = (S, \text{Acts}, T, s_0, AP, L)$  where  $S$  is a set of states, each state in  $S$  being uniquely identified by its label;  $\text{Acts}$  is the set of actions (event names);  $s_0 \in S$  is the initial state obtained by executing the *Init* event;  $AP$  is the set of atomic propositions: a set of predicates that correspond to the valuations of  $\bar{v}$  and satisfy the invariant  $I(\bar{v})$ ;  $L: S \rightarrow AP$  is a labelling function that provides the valuations of the variables  $\bar{v}$  in a given state; and  $T \subseteq S \times \text{Acts} \times S$  is the transition relation corresponding to the actions of the events of  $M$ .

### 2.3 Refinement in Event-B

In Event-B, the process of modelling systems is based on the theory of refinement. Many research works have focused in the development of the theory of refinement [6, 7, 28]: it appears from the literature that refinement is used in two related concepts in computer science. The first one considers refinement as a top-down program development methodology when the system is firstly described by an abstract specification and progressively refined by other ones. Each abstract specification will be more detailed, and new details can be introduced during refinement. The second concept

concerns the preservation of correctness between abstract and refined specifications.

Event-B refinement supports both concepts: it is a mechanism for introducing details about the static and dynamic properties of a model while preserving correctness. For the static part, refinement in Event-B allows a detailed description of the state space by the introduction of new variables (i.e. data/context refinement [15, 20]). Concerning the dynamic aspects, refinement in Event-B also allows a more detailed description of the execution of the system by adding new events processing the new introduced variables or by giving more details on the events of an abstract model. For more details on the theory of refinement in Event-B, we refer the reader to the action systems formalism [6] which has inspired the development of Event-B.

In Event-B, under a number of conditions expressed as proof obligations, a concrete model  $N = (\bar{w}, J(\bar{v}, \bar{w}), \text{Evts}_c, \text{Init}_c)$  may refine an abstract model  $M = (\bar{v}, I(\bar{v}), V(\bar{v}), \text{Evts}, \text{Init})$ . In this case,  $\bar{w}$  is a set of variables containing some (preserved) variables of the abstract model and some new variables introduced by refinement,  $J(\bar{v}, \bar{w})$  is an invariant that must provide a relation between the (removed) abstract variables ( $\bar{v}$ ) and the new concrete variables ( $\bar{w}$ ),  $\text{Evts}_c$  is the set of concrete events that contains both the refined events ( $\text{Evts}_c \cap \text{Evts}_a$ ) and the new events introduced by refinement ( $\text{Evts}_c \setminus \text{Evts}_a$ ), and  $\text{Init}_c$  is the concrete initialisation event.

*Events* Each abstract event from the set  $\text{Evts}_a$  can be refined by one or more concrete events. Moreover, several events from the abstract set  $\text{Evts}_a$  can be refined a single one. The first case is called *splitting*, while the second one is called *merging*.

$e_a \cong$ <p><b>any</b> <math>\bar{t}</math> <b>where</b>  <math>G_a(\bar{t}, \bar{v})</math>  <b>then</b>  <math>S_a(\bar{t}, \bar{v})</math>  <b>end</b></p>	$e_c \cong$ <p><b>refines</b> <math>e_{i+1}</math>  <b>any</b> <math>\bar{u}</math> <b>where</b>  <math>G_c(\bar{u}, \bar{w})</math>  <b>with</b>  <math>W_i(\bar{t}, \bar{u}, \bar{v}, \bar{v}', \bar{w}, \bar{w}')</math>  <b>then</b>  <math>S_c(\bar{u}, \bar{w})</math>  <b>end</b></p>
--	--

In this paper, we do not treat the cases of event *splitting* and *merging*, and we only consider the refinement of an abstract event  $e_a$  by only one concrete event  $e_c$  as follows. We remark that the event  $e_c$  may contain one more component  $W_i(\bar{t}, \bar{u}, \bar{v}, \bar{v}', \bar{w}, \bar{w}')$  which denotes a witness. A witness links the abstract parameters  $\bar{t}$  and the abstract variables  $\bar{v}'$  to concrete parameters  $\bar{u}$  and concrete variables  $\bar{w}'$ .

As the Event-B refinement process allows a more detailed description of the execution of the system, it is necessary to introduce new events ( $\text{Evts}_c \setminus \text{Evts}_a$ ) which characterise the evolution of the new added variables during refinement.

*Proof obligations* In Event-B refinement, the behaviour of the concrete model must be compatible with the behaviour of the abstract one. This constraint is verified and maintained by some proofs obligations dedicated to refinement. All the refinement POs are presented in [1]. In the following, we recall only the most important ones.

For the refinement of an abstract event  $e_a$  by a concrete event  $e_c$ , the two following POs must be satisfied:

1. **Guard strengthening** The guard of  $e_c$  is as least as strong as the guard of  $e_a$ :

$$\begin{array}{l} I(\bar{v}) \wedge J(\bar{v}, \bar{w}) \wedge G_c(\bar{u}, \bar{w}) \wedge W_{i_c}(\bar{t}, \bar{u}, \bar{v}, \bar{v}', \bar{w}, \bar{w}') \\ \vdash \\ G_a(\bar{t}, \bar{v}) \\ \text{(grd/STRENGTH)} \end{array}$$

2. **Simulation** The action of  $e_c$  simulates the action of  $e_a$ :

$$\begin{array}{l} I(\bar{v}) \wedge J(\bar{v}, \bar{w}) \wedge G_c(\bar{u}, \bar{w}) \wedge S_c(\bar{u}, \bar{w}, \bar{w}') \vdash S_a(\bar{t}, \bar{v}, \bar{v}') \\ \text{(act/SIM)} \end{array}$$

The last ensures that when a concrete event is executed, what it does is not contradictory with what the abstract event does.

When introducing new events during refinement, it is then necessary to show that their introduction cannot prevent the system from behaving as specified in the abstract model. In particular, it is necessary to show that such new events are “convergent”, in the sense that they cannot keep control indefinitely: at some point the system has to stop executing new events in order to follow the behaviour specified in its abstract model.

In order to prove that a set of events is convergent in Event-B, we have to introduce a natural number expression  $V(\bar{v})$ , called a *variant*, and show that all convergent events strictly decrease the value of this variant. As a consequence, when the variant hits zero, it is guaranteed that no convergent event can be performed. That leads to two POs to be discharged:

1. **Numeric variant** Under the guard  $G_i(\bar{t}, \bar{v})$  of each convergent event  $e_i$ , the variant  $V(\bar{v})$  is greater or equal to 0.

$$I(\bar{v}) \wedge G_i(\bar{t}, \bar{v}) \vdash V(\bar{v}) \in \text{NAT} \quad \text{(event/var/NAT)}$$

2. **Convergence** The action  $S_i(\bar{t}, \bar{v})$  of each convergent event  $e_i$  must always decrease the variant  $V(\bar{v})$ .

$$I(\bar{v}) \wedge G_i(\bar{t}, \bar{v}) \vdash \forall \bar{v}'. S_i(\bar{t}, \bar{v}, \bar{v}') \Rightarrow V(\bar{v}') < V(\bar{v}) \quad \text{(event/VAR)}$$

### 3 Running example: simple peer-to-peer protocol

We now introduce our running example, based on a simplified scenario of a peer-to-peer protocol, based on BitTorrent [12]. The description of the complete case study can be found in [31]. The model considers a set of  $N$  clients trying to download a file that has been partitioned into  $K$  blocks. Initially, no block has been downloaded by any client. The protocol ends when all the clients have successfully downloaded all the blocks.

*Initial protocol model* The model  $P2P_1$  given in Fig. 2 presents an abstract Event-B specification of the protocol. It represents a general abstraction of the behaviour of the protocol with no details included. At this level, the state of the protocol is described by means of one variable  $DB$ . This variable contains a matrix which indicates for each client  $c \in 1..N$  and each block  $b \in 1..K$  whether the client has already downloaded the block ( $DB(c \mapsto b) = \text{finished}$ ) or not ( $DB(c \mapsto b) = \text{empty}$ ). Initially, no block has been downloaded by any client.

At this level of abstraction, we only consider one event (AllDL) describing in one statement the whole execution of the protocol.  $\text{magicDB}$  is a parameter chosen in such a way that for all client  $c \in 1..N$  and block  $b \in 1..K$ , we have  $\text{magicDB}(c \mapsto b) = \text{finished}$ . The substitution  $DB := \text{magicDB}$  corresponds to the (somehow magical) download of all the blocks by all the clients in one shot. Notice that in reality, the download of all the blocks of the file by all the clients is not done in one shot. It is instead made gradually by successive attempts. Introducing these attempts is the purpose of the first refinement, which we present hereafter.

*Step-by-step download* We now present a first refinement of the protocol. For this purpose, we enlarge the set of variables and events. The resulting model is presented in Fig. 3. We introduce a new variable  $DBin$  that contains a matrix which represents the state of download of each block at each iteration of the model. For each client and each block, the corresponding state could be *finished*—indicating that the client has successfully downloaded the block; *incoming*—indicating that the client is currently trying to download the block; or *empty*—indicating that the download of the block has not yet started. Initially, as in the abstract model, no block has been downloaded by any client, therefore  $DBin := (1..N \times 1..K) \times \{\text{empty}\}$ . Furthermore, we impose that each client  $c$  is not currently trying to download more than one block  $b$ , as indicated in the invariant in Fig. 3.

To model the download process in a step-by-step manner, we introduce two new events: *Start1DL* and *Finish1DL*.

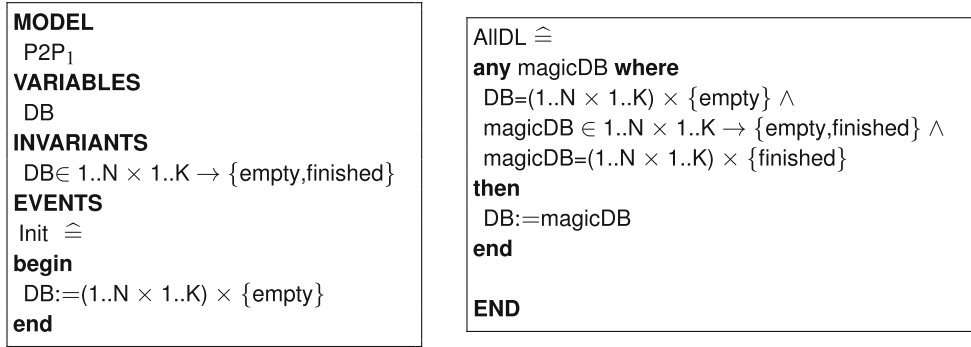


Fig. 2 Initial Event-B model of the simple P2P protocol

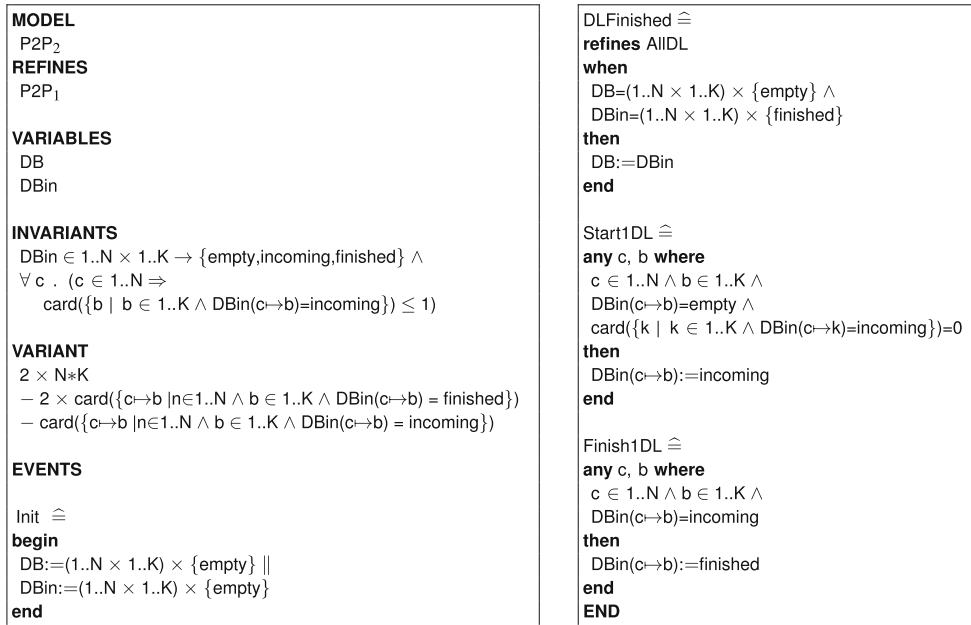


Fig. 3 First refinement of the simple P2P protocol: step-by-step download

In the event **Start1DL**, a client  $c$  and a block  $b$  are chosen in a non-deterministic manner in such a way that the download of the block  $b$  by client  $c$  has not yet started; furthermore, the considered client  $c$  is not currently trying to download another block  $k$ ; then, the client starts to download the block ( $DBin(c \rightarrow b) := incoming$ ). The event **Finish1DL** models that a client  $c$  terminates the downloading of a block  $b$  in a similar manner. The events **Start1DL** and **Finish1DL** are activated until we cannot find any pair  $(c,b)$  such that  $DBin(c \rightarrow b) = empty$ .

The event **DLFinished** refines the event **AllDL**. It is now enabled when  $DBin = (1..N \times 1..K) \times \{finished\}$ , i.e. when all the clients have successfully downloaded all the blocks. Then, it just substitutes the value of  $DBin$  to  $DB$  to realise the abstract attempted substitution from **AllDL**.

As we introduce two new events, we have to show their convergence by introducing the variant given in Fig. 3. Each

time **Start1DL** and **Finish1DL** are activated, their actions increase the numbers of *incoming* or *finished* in  $DBin$ ; therefore, the variant clearly decreases.

Figure 4 gives an extract of the operational semantics of this first refinement in terms of transition system, with  $N=2$  and  $K=2$ . In this figure, a small dot in the matrix indicates that the block has not been downloaded yet, while an empty (resp. filled) bullet represents that the block is incoming (respectively, successfully downloaded). For readability purposes, transitions in this TS have been annotated with the corresponding parameter values. The indicated  $v$  corresponds to the value of the variant in the corresponding state. As expected, the value of  $v$  decreases each time a new event is performed.

Remark that, at this point, any download attempt is ultimately successful. The purpose of the next refinement step is therefore to introduce failures in the block download process.

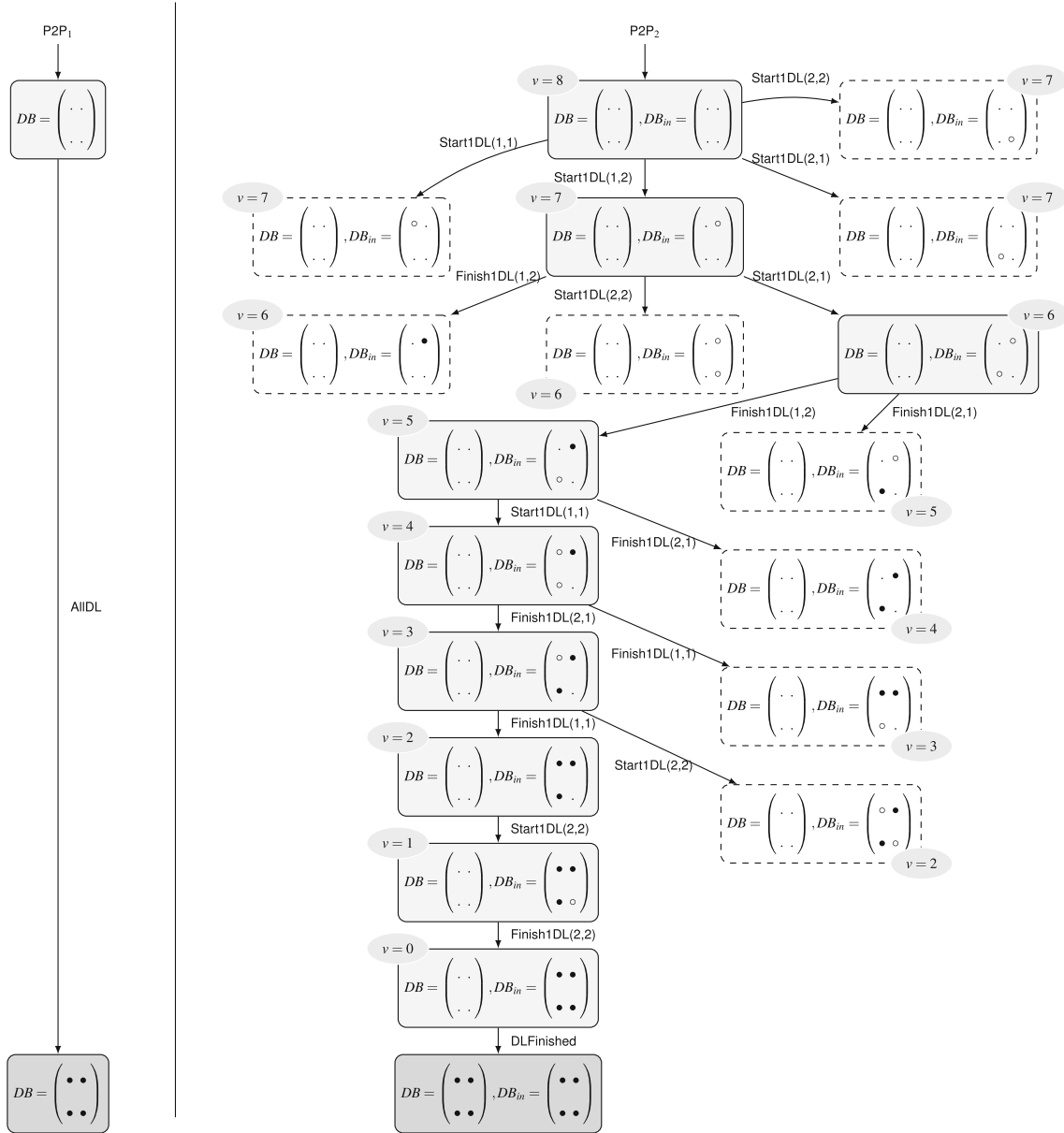


Fig. 4 Extract of the transition system of the first refinement of the simple P2P protocol, with N=2 and K=2

*Introducing failures* In this refinement  $P2P_3$ , we want to take into consideration some possible failures during the download process. More precisely, two possible failures can occur when a download is incoming: a failure can be critical (in this case, the download must be aborted) or not (in this case, the download continues). We therefore simply add to the previous Event-B model a new event **FailureDL**, given in Fig. 5. Its action non-deterministically chooses a value from  $\{\text{empty}, \text{incoming}\}$ : **empty** is chosen when the failure aborts the download, and **incoming** is chosen otherwise. This new event has the same guard as the event **FinishDB**. As a consequence, both events are enabled at the same time and the choice between them is non-deterministic, which mod-

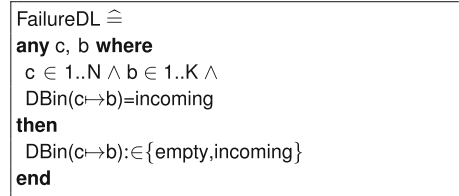


Fig. 5 New event introduced in the second refinement step

els the uncontrolled occurrence of failures. An extract of the operational semantics of this new model is provided in Fig. 6.

As we have introduced a new event, we need to prove its convergence. Unfortunately, due to the non-deterministic



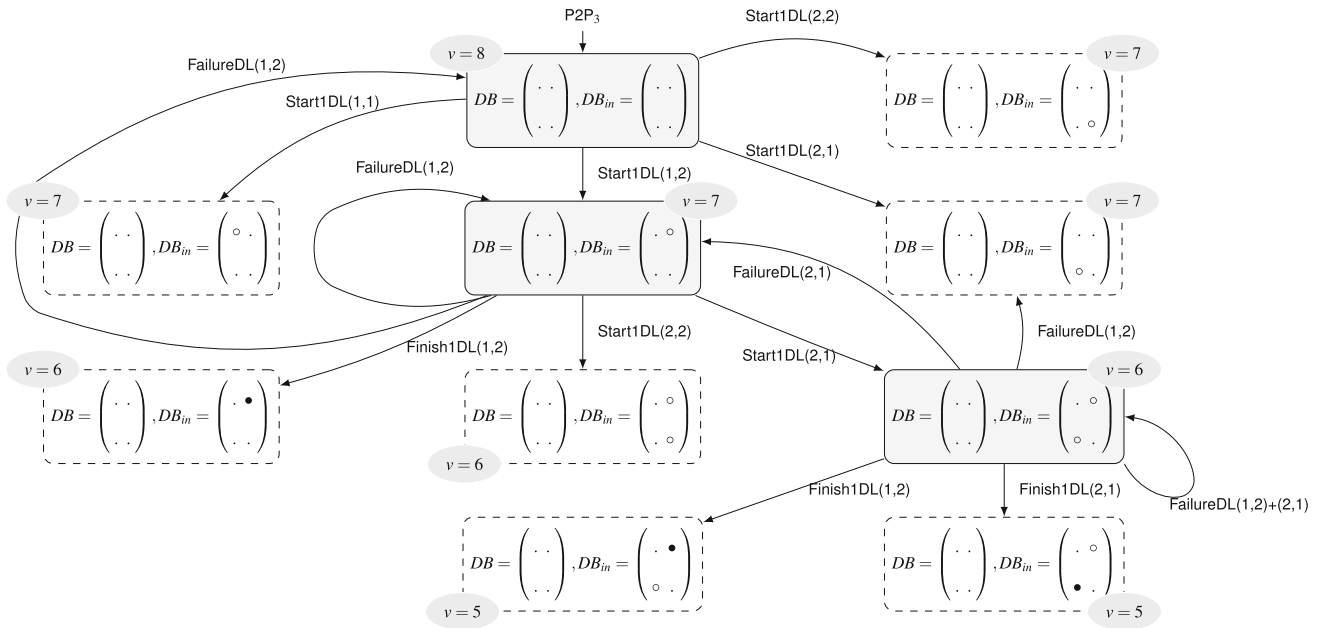


Fig. 6 Extract of the transition system of the second refinement of the simple P2P protocol, with N=2 and K=2

nature of the assignment in the event **FailureDL**, it is impossible to provide a variant that decreases regardless of its outcome. The refinement between this new model and the previous one is therefore not correct.

We will address the same problem in the probabilistic setting and prove that, in this case, the probabilistic version of **FailureDL** almost-certainly converges. As a consequence, unlike here, the refinement is correct in the probabilistic setting.

### 4 Fully probabilistic Event-B

We recall that our goal is to introduce probabilistic reasoning within Event-B. In this section, we present the basic elements of syntax and semantics for fully probabilistic Event-B models. We begin by presenting the sources of non-determinism in Event-B and explaining how they can be replaced by probabilistic choices in the context of fully probabilistic models. We then present the syntax of such fully probabilistic Event-B models. In order to ensure the consistency these models, we present the set of new POs specific to the introduced elements and how standard POs can be adapted in this context. Finally, in order to prove the correctness of our approach, we propose operational semantics of such models in terms of Markov chains.

#### 4.1 Introducing probabilistic choices

In Event-B, non-determinism can appear in three places: the choice of the enabled event to be executed, the choice of the parameter value to be taken and the choice of the value

to be assigned to a given variable in a non-deterministic assignment. To obtain a *fully probabilistic Event-B model*, we propose to replace all these non-deterministic choices with probabilistic ones. In the following, we go through these three sources of non-determinism and explain how to turn them into probabilistic choices.

*Choice of the enabled event* In standard Event-B, when several events are enabled in a given configuration, the event to be executed is chosen non-deterministically. In order to resolve this non-deterministic choice, we propose to equip each probabilistic event with a *weight*. In configurations where several probabilistic events are enabled, the probability of choosing one of them will therefore be computed as the ratio of its weight against the total value of the weights of all enabled events in this state. Using weights instead of actual probability values is convenient as the set of enabled events evolves with the configuration of the system. Using probability values instead would require to normalise them in all configurations. Moreover, for the sake of expressivity, we propose to express the weight  $W_i(\vec{v})$  of a probabilistic event  $\Theta_i$  as an expression over the variables  $\vec{v}$  of the fully probabilistic Event-B model. The probability of executing a given event can therefore evolve as the system progresses. A probabilistic event is therefore allowed to be executed only if *i*) its guards is fulfilled and *ii*) its weight is strictly positive.

*Choice of the parameter values* In standard Event-B, events can be equipped with parameters. In each configuration where this is possible, a valuation of the parameters is chosen

such that the guard  $G_i(\bar{t}, \bar{v})$  of the event is satisfied. When there are several such parameter valuations, one of them is selected non-deterministically. We therefore propose to replace this non-deterministic choice by a uniform choice over all parameter valuations ensuring that the guard of the event is satisfied. The uniform distribution is a default choice, but our results can be extended to any other discrete distribution.

*Non-deterministic assignments* Recall that non-deterministic assignments in Event-B are expressed in two forms: predicate non-deterministic assignments and enumerated non-deterministic assignments.

We propose to replace predicate non-deterministic assignments by *predicate probabilistic assignments* written

$$x := \oplus Q_x(\bar{t}, \bar{v}, x')$$

Instead of choosing non-deterministically among the values of  $x'$  such that the predicate  $Q_x(\bar{t}, \bar{v}, x')$  is true as in standard predicate non-deterministic assignments, we propose to choose this new value using an uniform distribution. For simplicity reasons, we enforce that this uniform distribution must be discrete and therefore that the set of values  $x'$  such that  $Q_x(\bar{t}, \bar{v}, x')$  is true must always be finite. As above, the uniform distribution we propose by default could be replaced by any other discrete distribution.

We propose to replace enumerated non-deterministic assignments by *enumerated probabilistic assignments* written

$$x := E_1(\bar{t}, \bar{v}) @ p_1 \oplus \dots \oplus E_m(\bar{t}, \bar{v}) @ p_m$$

In this structure, the variable  $x$  is assigned the expression  $E_i$  with probability  $p_i$ . In order to define a correct probability distribution, each  $p_i$  must be strictly positive and smaller or equal to 1, and they must sum up to 1. Although rational numbers are not natively handled in Event-B, we assume that the context of the model allows the use of rational numbers. In practice, that can be done by defining a ‘‘Rational’’ theory in Rodin using the theory plugin providing capabilities to define and use mathematical extensions to the Event-B language and the proving infrastructure [13].

Remark that standard deterministic assignments are retained, but can also be considered as enumerated probabilistic assignments where  $m=1$ .

$e_i \hat{=} \text{weight } W_i(\bar{v})$   
**any**  $\bar{t}$  **where**  
 $G_i(\bar{t}, \bar{v})$   
**then**  
 $SP_i(\bar{t}, \bar{v})$   
**end**

Refining all non-deterministic choices into probabilistic choices has side effects on the syntax of events and models. In probabilistic Event-B, we therefore propose to use the syntax above for a probabilistic event  $e_i$  where  $W_i(\bar{v})$  is the weight of the event,  $G_i(\bar{t}, \bar{v})$  is the guard of the event, and  $SP_i(\bar{t}, \bar{v})$  is a *probabilistic action*, i.e. an action consisting only of deterministic and *probabilistic* assignments which are executed in parallel. Remark that the before–after predicate  $SP_i(\bar{t}, \bar{v}, \bar{v}')$  of such a probabilistic event will be identical to the BA of its standard (non-deterministic) counterpart.

For simplicity reasons we impose, as in standard Event-B, that the initialisation event must be deterministic. The results we present in the rest of the paper can, nevertheless, easily be extended to probabilistic initialisation events.

**Definition 1** (*Fully probabilistic Event-B model*) A fully probabilistic Event-B model is a tuple  $M=(\bar{v}, I(\bar{v}), PEvs, Iinit)$  where  $\bar{v} = \{v_1 \dots v_n\}$  is a set of variables,  $I(\bar{v})$  is the invariant,  $PEvs$  is a set of *probabilistic* events, and  $Iinit$  is the initialisation event.

*Running example* A probabilistic version of the P2P model from Sect. 3 is given in Fig. 7. This model has the same variables, the same invariants and the same events as the Event-B model from Figs. 3 and 5.

The events of the model  $P2P_P$  are annotated with specific weights. The risk of download failures decreases with the number of successful downloads: each time a block is successfully downloaded, the weight of `Finish1DL` increases, whereas the weight of `FailureDL` decreases. The weight of `Start1DL` models that the probability of starting a new download decreases with the number of blocks being currently downloaded.

In case of failure, we fix the probability of aborting the download to 40%. This probability is introduced in the event `FailureDL` by using an enumerated probabilistic assignment instead of a non-deterministic one: the variable `DBin(c→b)` is assigned the value `empty` with a probability  $\frac{4}{10}$  (the download aborts) and the value `incoming` with a probability  $\frac{6}{10}$  (the download continues).

## 4.2 Consistency

As in standard Event-B, the consistency of a fully probabilistic Event-B model is defined by means of proof obligations (POs). In this section, we therefore introduce new POs specific to fully probabilistic Event-B and explain how we adapt standard Event-B POs in order to prove the consistency of fully probabilistic Event-B models.

### 4.2.1 Specific POs for fully probabilistic Event-B

We start by presenting new POs specific to fully probabilistic Event-B.

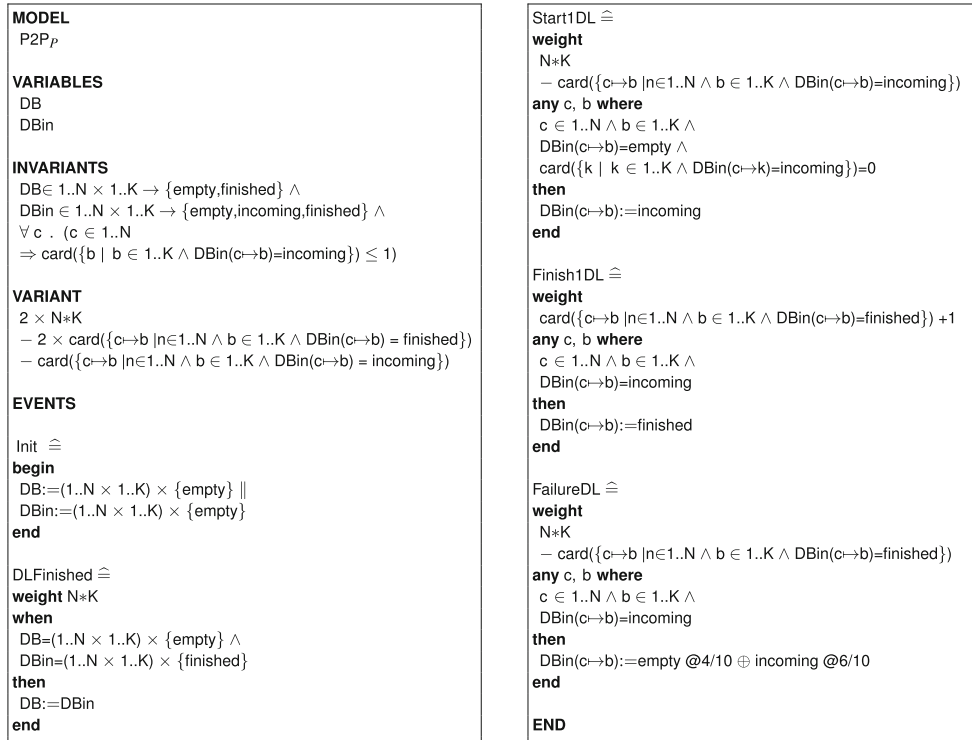


Fig. 7 Probabilistic version of the simple P2P protocol

*Numeric weight* For simplicity reasons, we impose that the expression  $W_i(\vec{v})$  representing the weight of a given probabilistic event must evaluate to natural numbers.

$$I(\vec{v}) \wedge G_i(\vec{t}, \vec{v}) \vdash W_i(\vec{v}) \in \text{NAT} \quad (\text{event/WGHT/NAT})$$

*Parameter values finiteness* In order to be able to use a discrete uniform distribution over the set of parameter valuations ensuring that the guard of a probabilistic event is satisfied, we impose that this set must be finite.

$$I(\vec{v}) \vdash \text{finite}(\{\vec{t} \mid G_i(\vec{t}, \vec{v})\}) \quad (\text{event/param/pWD})$$

*Enumerated probabilistic assignments well-definedness and feasibility* In all enumerated probabilistic assignments, it is necessary to ensure that the discrete probability values  $p_1 \dots p_n$  define a correct probability distribution. Formally, this leads to two POs:

1. Probability values  $p_i$  in enumerated probabilistic assignments are strictly positive and smaller or equal to 1.

$$\vdash 0 < p_i \leq 1 \quad (\text{event/assign/pWD1})$$

2. The sum of the probability values  $p_1 \dots p_n$  in enumerated probabilistic assignments must be equal to 1.

$$\vdash p_1 + \dots + p_n = 1 \quad (\text{event/assign/pWD2})$$

Feasibility of enumerated probabilistic assignments is trivial: as soon as at least one expression  $E_i(\vec{t}, \vec{v})$  is present and well defined, it always returns a value.

*Predicate probabilistic assignment well-definedness and feasibility* In order to define a discrete uniform distribution over the set of values of a variable  $x$  making the predicate  $Q_x(\vec{t}, \vec{v}, x')$  of the corresponding assignment satisfied, we impose that this set must be finite.

$$I(\vec{v}) \wedge G_i(\vec{t}, \vec{v}) \wedge W_i(\vec{v}) > 0 \vdash \text{finite}(\{x' \mid Q_x(\vec{t}, \vec{v}, x')\}) \quad (\text{event/assign/pWD3})$$

Feasibility of predicate probabilistic assignments is ensured by the standard feasibility PO [1] inherited from Event-B. It ensures that the set  $\{x' \mid Q_x(\vec{t}, \vec{v}, x')\}$  is not empty.

#### 4.2.2 Modifications to standard POs

In standard Event-B, if we want to prove that an event is enabled, we need to prove that its guard is satisfied. However, in fully probabilistic Event-B, we additionally need to prove that its weight is strictly positive. We therefore modify standard and optional Event-B POs as follows.

*Invariant preservation* The invariant must be preserved by all enabled probabilistic events.

$$I(\bar{v}) \wedge G_i(\bar{t}, \bar{v}) \wedge W_i(\bar{v}) > \mathbf{0} \wedge SP_i(\bar{t}, \bar{v}, \bar{v}') \vdash I(\bar{v}')$$

(event/pINV)

*Deadlock freedom* In all acceptable configurations, there must exist at least one enabled probabilistic event.

$$I(\bar{v}) \vdash (G_1(\bar{t}, \bar{v}) \wedge W_1(\bar{v}) > \mathbf{0}) \vee \dots \vee (G_n(\bar{t}, \bar{v}) \wedge W_n(\bar{v}) > \mathbf{0})$$

(model/pDLF)

For the sake of understanding, we hereby insist on the separation between the guard of an event, which reflects the classical notion of enabledness, and the fact that its weight must be strictly positive. Obviously, one could also automatically rewrite the guard of all probabilistic events in order to include the condition on its weight. This solution would allow conserving most of the standard Event-B consistency POs without modifications in the probabilistic setting.

### 4.2.3 Running example

Consider the fully probabilistic Event-B model P2P<sub>p</sub> given in Fig. 7: all the weight expressions are natural numbers (event/WGHT/NAT) and, for each event, the number of acceptable parameter valuations is finite (event/param/pWD). For the probabilistic enumerated assignment on the event FailureDL, each given probability is a rational *p* such that  $0 < p \leq 1$  (event/assign/pWD1) and their sum is clearly equal to 1 (event/assign/pWD2). The invariant is always preserved by each probabilistic version of the events (event/pINV). The model P2P<sub>p</sub> is therefore *consistent*.

## 4.3 Semantics

As explained in Sect. 2.2, the operational semantics of standard Event-B models is expressed in terms of labelled transition systems. In the following, we extend this work by presenting the operational semantics of fully probabilistic Event-B models in terms of discrete-time Markov chains (DTMCs).

Remark that our goal, unlike in [35,37], is not to translate our models into DTMCs and use standard model-checking techniques to verify them. Instead, we aim at reasoning directly on fully probabilistic Event-B models and benefiting from the symbolic proof mechanism that is the signature of the Event-B approach. The following DTMC semantics are, nevertheless, introduced as a demonstration of the correctness of our approach and results.

### 4.3.1 Notations

Let  $M=(\bar{v}, I(\bar{v}), PEvts, Iinit)$  be a fully probabilistic Event-B model and  $\sigma$  be a valuation of its variables. Given a variable  $x \in \bar{v}$ , we write  $[\sigma]x$  for the value of  $x$  in  $\sigma$ . Given an expression  $E(\bar{v})$  over variables in  $\bar{v}$ , we write  $[\sigma]E(\bar{v})$  (or  $[\sigma]E$  when clear from the context) for the evaluation of  $E(\bar{v})$  in the context of  $\sigma$ . Given an expression  $E(\bar{t}, \bar{v})$  over variables and parameters, we write  $[\sigma, \theta]E(\bar{t}, \bar{v})$  for the evaluation of  $E(\bar{t}, \bar{v})$  under parameter valuation  $\theta$  and variable valuation  $\sigma$ .

Given a probabilistic event  $e_i$  with a set of parameters  $\bar{t}$  and a valuation  $\sigma$  of the variables, we write  $T_\sigma^{e_i}$  for the set of parameter valuations  $\theta$  such that the guard of  $e_i$  evaluates to **true** in the context of  $\sigma$  and  $\theta$ . Formally,  $T_\sigma^{e_i} = \{\theta \mid [\sigma, \theta]G_i(\bar{t}, \bar{v}) = true\}$ . Recall that parameter valuations are chosen uniformly on this set. We write  $P_{T_\sigma^{e_i}}$  for the uniform distribution on the set  $T_\sigma^{e_i}$ .

Given a valuation  $\sigma$  of the variables and a probabilistic event  $e_i$ , we say that  $e_i$  is *enabled* in  $\sigma$  iff (a) the weight of  $e_i$  evaluates to a strictly positive value in  $\sigma$  and (b) either  $e_i$  has no parameter and its guard evaluates to **true** in  $\sigma$  or there exists at least one parameter valuation  $\theta$  such that the guard of  $e_i$  evaluates to **true** in the context of  $\sigma$  and  $\theta$ , i.e.  $T_\sigma^{e_i} \neq \emptyset$ .

Let  $e_i$  be a probabilistic event in **PEvts** and let  $x$  be a variable modified by  $e_i$ . Recall that  $x$  can be modified only by one assignment within the action of  $e_i$ . If  $x$  is modified by an enumerated probabilistic assignment ( $x := E_1(\bar{t}, \bar{v})@_{p_1} \oplus \dots \oplus E_m(\bar{t}, \bar{v})@_{p_m}$  ( $m \geq 1$ )), then we write  $\mathcal{E}_{e_i}(x)$  for the set of all expressions that can be assigned to the variable  $x$  by this assignment.

$$\mathcal{E}_{e_i}(x) = \{E_1(\bar{t}, \bar{v}), \dots, E_m(\bar{t}, \bar{v})\}$$

The probability of choosing an expression  $E_i$  among all others expressions is written  $P_x^{e_i}(E_i) = p_i$ .

Given a probabilistic event  $e_i$ , we write  $Var(e_i)$  for the set of variables in  $\bar{v}$  that are modified by the action of  $e_i$ , i.e. the variables that appear on the left side of an assignment in  $SP_i(\bar{t}, \bar{v})$ . Recall that a variable  $x \in Var(e_i)$  must be on the left side of either a predicate probabilistic assignment or an enumerated probabilistic assignment. Let  $e_i \in PEvts$  be a probabilistic event,  $x \in Var(e_i)$  be a variable,  $\sigma, \sigma'$  two valuations of the variables  $\bar{v}$  and  $\theta$  a valuation of the parameter values associated to the event  $e_i$  such that  $e_i$  is enabled in the context of  $\sigma$  and  $\theta$  and leads the system to  $\sigma'$ .

If  $x$  is modified by an enumerated probabilistic assignment of  $e_i$ , then we write  $\mathcal{E}_{e_i}(x)|_{\sigma, \theta}^{\sigma'}$  for the set of expressions in  $\mathcal{E}_{e_i}(x)$  such that their evaluation in the context of  $\sigma$  and  $\theta$  returns the value of  $x$  in the valuation  $\sigma'$ .

Formally,

$$\mathcal{E}_{e_i}(x)|_{\sigma, \theta}^{\sigma'} = \{E \in \mathcal{E}_{e_i}(x) \mid [\sigma, \theta](E(\bar{t}, \bar{v})) = [\sigma']x\}$$

If  $e_i$  is not equipped with parameters, then this subset is written  $\mathcal{E}_{e_i}(x)|_{\sigma'}^{\sigma}$ .

If  $x$  is modified by a predicate probabilistic assignment ( $x : \oplus Q_x(\bar{t}, \bar{v}, x')$ ), then we write  $\mathcal{V}_{\theta, \sigma}^{e_i}(x)$  for the set of values  $x'$  that make the predicate  $Q_x(\bar{t}, \bar{v}, x')$  true when evaluated in  $\sigma$  and  $\theta$ .

$$\mathcal{V}_{\theta, \sigma}^{e_i}(x) = \{x' \mid [\sigma, \theta]Q_x(\bar{t}, \bar{v}, x') = true\}$$

If  $e_i$  is not equipped with parameters, then this subset is written  $\mathcal{V}_{\sigma}^{e_i}(x)$ .

Let  $e_i$  be a probabilistic event and let  $x$  be a variable in  $Var(e_i)$ , given an original valuation  $\sigma$  of the variables, a valuation  $\theta$  of the parameters of  $e_i$  and a target valuation  $\sigma'$  of the variables, we write  $P_{\sigma, \theta}^{e_i}(x, \sigma')$  for the probability that  $x$  is assigned the new value  $[\sigma']x$  when executing  $e_i$  from the valuation  $\sigma$  and with parameter valuation  $\theta$ . If  $e_i$  is not equipped with parameters, this is written  $P_{\sigma}^{e_i}(x, \sigma')$ . In the following, we always use the more general notation and assume that it is replaced with the specific one when there are no parameters. Formally, this probability is given by:

1. if  $x$  is modified by an enumerated probabilistic assignment, then:

$$P_{\sigma, \theta}^{e_i}(x, \sigma') = \sum_{E \in \mathcal{E}_{e_i}(x)|_{\sigma, \theta}^{\sigma'}} P_x^{e_i}(E)$$

2. if  $x$  is modified by a predicate probabilistic assignment, then:

$$P_{\sigma, \theta}^{e_i}(x, \sigma') = \frac{1}{card(\mathcal{V}_{\theta, \sigma}^{e_i}(x))}$$

if  $[\sigma']x \in \mathcal{V}_{\theta, \sigma}^{e_i}(x)$  and 0 otherwise.

#### 4.3.2 DTMC operational semantics

Informally, the operational semantics of a fully probabilistic Event-B model  $M=(\bar{v}, l(\bar{v}), PEvts, linit)$  is a probabilistic LTS  $\llbracket M \rrbracket = (S, Acts, P, s_0, AP, L)$  where the states, labels, actions, atomic propositions and initial state are similarly obtained as for the standard LTS semantics of Event-B. The only difference with the standard LTS semantics is that the transitions are equipped with probabilities, which we explain below. In the following, we identify the states with the valuations of the variables defined in their labels.

Intuitively, the transition probabilities are obtained as follows: Let  $e_i \in PEvts$  be a probabilistic event,  $x \in \bar{v}$  be a variable and  $s, s'$  be two states of  $\llbracket M \rrbracket$  such that  $(s, e_i, s')$  is a transition in the standard LTS semantics, i.e. where  $e_i$  is enabled in  $s$  and there exists a parameter valuation  $\theta \in T_s^{e_i}$ , if any, such that the action of  $e_i$  may take the system from  $s$  to  $s'$ . The probability assigned to transition  $(s, e_i, s')$  is then

equal to the product of (1) the probability that the event  $e_i$  is chosen from the set of enabled events in state  $s$ , (2) the probability of choosing each parameter valuation  $\theta$ , and (3) the overall probability that each modified variable is assigned the value given in  $s'$  under parameter valuation  $\theta$ .

**Definition 2** (Fully probabilistic Event-B operational semantics) The operational semantics of a fully probabilistic Event-B model  $M=(\bar{v}, l(\bar{v}), PEvts, linit)$  is a PLTS  $\llbracket M \rrbracket = (S, Acts, P, s_0, AP, L)$  where  $S, Acts, s_0, AP,$  and  $L$  are defined as in the standard LTS semantics of Event-B models (see Sect. 2.2), and  $P : S \times Acts \times S \rightarrow [0, 1]$  is the transition probability function such that for a given state  $s$ , for all  $e_i, s' \in Acts \times S$ , we have  $P(s, e_i, s') = 0$  if  $e_i \notin Acts(s)$  or  $\exists x \in X \setminus \{Var(e_i)\}$  st  $[s]x \neq [s']x$  and otherwise

$$P(s, e_i, s') = \frac{[s]W_i(\bar{v})}{\underbrace{\sum_{e_j \in Acts(s)} [s]W_j(\bar{v})}_{(1)}} \times \sum_{\theta \in T_s^{e_i}} \underbrace{(P_{T_s^{e_i}}(\theta))}_{(2)} \times \underbrace{\prod_{x \in Var(e_i)} P_{s, \theta}^{e_i}(x, s')}_{(3)}$$

In the following proposition, we show that the semantics of a fully probabilistic Event-B model as defined above is indeed a DTMC.

**Proposition 1** The operational semantics of a fully probabilistic Event-B model  $M$  satisfying the POs given in Sect. 4.2.1 is a DTMC.

*Proof* We must prove that for each state  $s$  in  $\llbracket M \rrbracket$ , the sum of probabilities of the outgoing transitions from  $s$  is equal to one. Let  $M$  be a fully probabilistic Event-B model,  $\bar{v} = (x_1, x_2, \dots, x_n)$  the set of variables of  $M$  and  $s \in S$  a state of  $\llbracket M \rrbracket$ . We assume that each variable  $x_i$  in  $\bar{v}$  takes its value from a set  $X_i$ .

Recall that the probability of a transition  $(s, e_i, s')$  is 0 if  $e_i \notin Acts(s)$  or  $\exists x \in \bar{v} \setminus \{Var(e_i)\} \mid [s]x \neq [s']x$  and otherwise:

$$P(s, e_i, s') = \frac{[s]W_i(\bar{v})}{\sum_{e_j \in Acts(s)} [s]W_j(\bar{v})} \times \sum_{\theta \in T_s^{e_i}} (P_{T_s^{e_i}}(\theta)) \times \prod_{x \in Var(e_i)} P_{s, \theta}^{e_i}(x, s')$$

In order to prove that, for all  $s \in S$ ,  $P(s, \dots)$  is a probability distribution on  $Acts(s) \times S$ , we must therefore show that (1)  $P(s, e_i, s') \in [0, 1]$  for all  $(s, e_i, s')$ , and (2) for all  $s \in S$ ,  $\sum_{e_i \in Acts(s)} \sum_{s' \in S} P(s, e_i, s') = 1$ .

First, we observe that (1) is a direct consequence of POs (event/WGHT/NAT), (event/param/pWD), (event/assign/pWD1), (event/assign/pWD2) and (event/assign/pWD3). Indeed,

- (event/WGHT/NAT) ensures that  $0 \leq \frac{[s]W_i(\bar{v})}{\sum_{e_j \in \text{Acts}(s)} [s]W_j(\bar{v})} \leq 1$ ,
- (event/param/pWD) ensures that  $0 \leq P_{T_s^{e_i}}(\theta) \leq 1$  for all  $\theta \in T_s^{e_i}$  and that  $\sum_{\theta \in T_s^{e_i}} P_{T_s^{e_i}}(\theta) = 1$ , and
- (event/assign/pWD1), (event/assign/pWD2) and (event/assign/pWD3) ensure that  $0 \leq \prod_{x \in \text{Var}(e_i)} P_{s,\theta}^{e_i}(x, s') \leq 1$  for all  $\theta \in T_s^{e_i}$ .

Moreover, (2) is derived as follows:  
By definition,

$$\begin{aligned} & \sum_{s' \in S, e_i \in \text{Acts}(s)} P(s, e_i, s') \\ &= \sum_{e_i \in \text{Acts}(s)} \sum_{s' \in S} \frac{[s]W_i(\bar{v})}{\sum_{e_j \in \text{Acts}(s)} [s]W_j(\bar{v})} \\ & \times \sum_{\theta \in T_s^{e_i}} (P_{T_s^{e_i}}(\theta) \times \prod_{x \in \text{Var}(e_i)} P_{s,\theta}^{e_i}(x, s')) \end{aligned}$$

Since only  $P_{s,\theta}^{e_i}(x, s')$  depends on  $s'$ , this becomes

$$\begin{aligned} & \sum_{s' \in S, e_i \in \text{Acts}(s)} P(s, e_i, s') \\ &= \sum_{e_i \in \text{Acts}(s)} \frac{[s]W_i(\bar{v})}{\sum_{e_j \in \text{Acts}(s)} [s]W_j(\bar{v})} \\ & \times \sum_{\theta \in T_s^{e_i}} (P_{T_s^{e_i}}(\theta) \times \sum_{s' \in S} \prod_{x \in \text{Var}(e_i)} P_{s,\theta}^{e_i}(x, s')) \end{aligned}$$

Let  $S_1 = \{s' \in S \mid \forall x \in \bar{v} \setminus \text{Var}(e). [s]x = [s']x\}$ . Remark that  $P_{s,\theta}^{e_i}(x, s') = 0$  for all  $s' \notin S_1$ . As a consequence,

$$\begin{aligned} & \sum_{s' \in S, e_i \in \text{Acts}(s)} P(s, e_i, s') \\ &= \sum_{e_i \in \text{Acts}(s)} \frac{[s]W_i(\bar{v})}{\sum_{e_j \in \text{Acts}(s)} [s]W_j(\bar{v})} \\ & \times \sum_{\theta \in T_s^{e_i}} (P_{T_s^{e_i}}(\theta) \times \underbrace{\sum_{s' \in S_1} \prod_{x \in \text{Var}(e_i)} P_{s,\theta}^{e_i}(x, s')}_{\text{(A)}}) \end{aligned}$$

We now reduce the expression (A). For all  $x \in \text{Var}(e_i)$ , we recall that  $P_{s,\theta}^{e_i}(x, s') = \sum_{E \in \mathcal{E}_{e_i}(x)_{s,\theta}^{s'}} P_x^{e_i}(E)$  if  $x$  is modified by an enumerated probabilistic assignment and  $P_{s,\theta}^{e_i}(x, s') = \frac{1}{\text{card}(\mathcal{V}_{\theta,s}^{e_i}(x))}$  if  $x$  is modified by a predicate probabilistic assignment. As a consequence,  $P_{s,\theta}^{e_i}(x, s')$  does not really depend on  $s'$  but only depends on the value of  $x$  in  $s'$ :  $v'_x = [s']x \in X$ . Given  $x \in \bar{v}$  and  $v'_x = [s']x \in X$ , we therefore write  $F_x^{s,\theta,e_i}(v'_x) = P_{s,\theta}^{e_i}(x, s')$  if  $x \in \text{Var}(e_i)$ .

For  $\bar{v} = \{x_1, \dots, x_n\}$ , we have  $S_1 = \{(v'_{x_1}, \dots, v'_{x_n}) \mid v'_{x_i} \in X_i \text{ if } x_i \in \text{Var}(e_i) \text{ and } v'_{x_i} = [s]x_i \text{ otherwise}\}$ . Assuming

that  $\text{Var}(e_i) = \{x_1, \dots, x_k\}$  with  $k \leq n$ , we can therefore rewrite (A) as follows:

$$\begin{aligned} \text{(A)} &= \sum_{s' \in S_1} \prod_{x_i \in \text{Var}(e_i)} F_x^{s,\theta,e_i}(v'_{x_i}) \\ &= \sum_{v'_{x_1} \in X_1} \sum_{v'_{x_2} \in X_2} \dots \sum_{v'_{x_k} \in X_k} \left( \prod_{i=1}^k F_x^{s,\theta,e_i}(v'_{x_i}) \right) \\ &= \sum_{v'_{x_1} \in X_1} \sum_{v'_{x_2} \in X_2} \dots \sum_{v'_{x_k} \in X_k} (F_{x_1}^{s,\theta,e_i}(v'_{x_1}) \cdot F_{x_2}^{s,\theta,e_i}(v'_{x_2}) \cdot \dots \cdot F_{x_k}^{s,\theta,e_i}(v'_{x_k})) \\ &= \sum_{v'_{x_1} \in X_1} F_{x_1}^{s,\theta,e_i}(v'_{x_1}) \cdot \sum_{v'_{x_2} \in X_2} F_{x_2}^{s,\theta,e_i}(v'_{x_2}) \cdot \dots \cdot \sum_{v'_{x_k} \in X_k} F_{x_k}^{s,\theta,e_i}(v'_{x_k}) \\ &= \prod_{x_i \in \text{Var}(e_i)} \sum_{v'_{x_i} \in X_i} F_{x_i}^{s,\theta,e_i}(v'_{x_i}) \end{aligned}$$

By (event/assign/pWD1), (event/assign/pWD2) and (event/assign/pWD3), we have  $\sum_{v'_{x_i} \in X_i} F_{x_i}^{s,\theta,e_i}(v'_{x_i}) = 1$  for all  $x_i \in \text{Var}(e_i)$ , therefore  $\text{(A)} = \prod_{x_i \in \text{Var}(e_i)} [\sum_{v'_{x_i} \in X_i} F_{x_i}^{s,\theta,e_i}(v'_{x_i})] = 1$

As a consequence,

$$\begin{aligned} \sum_{s' \in S, e_i \in \text{Acts}(s)} P(s, e_i, s') &= \sum_{e_i \in \text{Acts}(s)} \frac{[s]W_i(\bar{v})}{\sum_{e_j \in \text{Acts}(s)} [s]W_j(\bar{v})} \\ & \times \sum_{\theta \in T_s^{e_i}} (P_{T_s^{e_i}}(\theta)) \end{aligned}$$

Moreover, by construction, we have  $\sum_{\theta \in T_s^{e_i}} P_{T_s^{e_i}}(\theta) = 1$ . Therefore,

$$\begin{aligned} \sum_{s' \in S, e_i \in \text{Acts}(s)} P(s, e_i, s') &= \sum_{e_i \in \text{Acts}(s)} \frac{[s]W_i(\bar{v})}{\sum_{e_j \in \text{Acts}(s)} [s]W_j(\bar{v})} \\ &= 1 \end{aligned}$$

As a conclusion,  $P(s, \cdot, \cdot)$  is indeed a probability distribution on  $\text{Acts}(s) \times S$  for all  $s \in S$  and therefore  $\llbracket M \rrbracket$  is a DTMC.  $\square$

### 4.3.3 Running example

Figure 8 presents the first steps of the detailed construction of the DTMC corresponding to the fully probabilistic Event-B model given in Fig. 7, with the number of clients  $N$  and the numbers of blocks  $K$  fixed to 2. We only present this detailed construction to illustrate the operational semantics of our model as defined above. This DTMC will not be used within the design process in probabilistic Event-B.

We now explain how some of the probability values in the DTMC from Fig. 8 are computed. We only focus on interesting (and complex) examples and leave out the rest of the computation to the reader. From the state referenced (s1) on Fig. 8, three events are enabled:

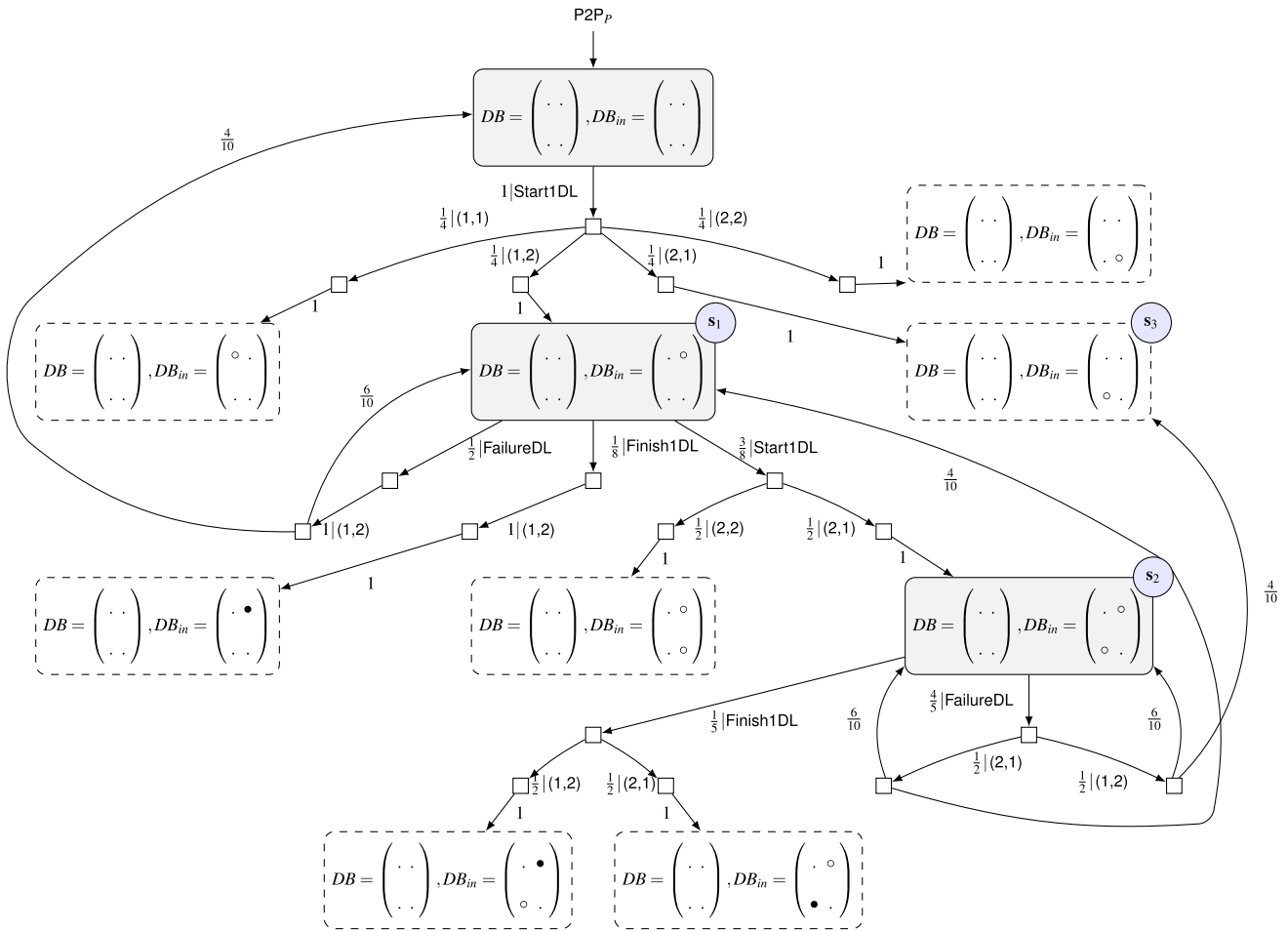


Fig. 8 Extract of the detailed construction of the DTMC of the simple P2P protocol, with N=2 and K=2

- FailureDL with a weight value of 4. The probability of choosing this event is therefore  $\frac{4}{4+3+1} = \frac{1}{2}$ ;
- Start1DL with a weight value of 3. The probability of choosing this event is therefore  $\frac{3}{4+3+1} = \frac{3}{8}$ ;
- Finish1DL with a weight value of 1. The probability of choosing this event is therefore  $\frac{1}{4+3+1} = \frac{1}{8}$ .

When choosing the event Finish1DL, only one valuation for the parameters is possible with a probability of 1. The corresponding action is deterministic and therefore executed with probability 1. The global probability of leaving the state (s<sub>1</sub>) using the event Finish1DL is therefore  $\frac{1}{8} \times 1 \times 1 = \frac{1}{8}$ .

When choosing the event Start1DL, two possible valuations for the parameters are possible, with a probability  $\frac{1}{2}$  for each of them. The action corresponding to event Start1DL is deterministic, and the parameter valuation (2, 1) allows to reach state (s<sub>2</sub>). As a consequence, the global probability of reaching (s<sub>2</sub>) from (s<sub>1</sub>) using the event Start1DL is  $\frac{3}{8} \times \frac{1}{2} \times 1 = \frac{3}{16}$ .

When choosing the event FailureDL, only one valuation for the parameters is possible. The corresponding action is probabilistic, leading to two different states (with probabilities  $\frac{6}{10}$  of going back to (s<sub>1</sub>) and  $\frac{6}{10}$  of going to another state). As a consequence, the global probability of looping on (s<sub>1</sub>) using the event FailureDL is  $\frac{1}{2} \times 1 \times \frac{6}{10} = \frac{3}{10}$ .

From the state referenced as (s<sub>2</sub>) on Fig. 8, we only focus on one interesting transition. Among the two events that can be enabled, we consider the event FailureDL: the probability of choosing this event is  $\frac{4}{5}$ . Two possible valuations for the parameters are then possible, with a probability of  $\frac{1}{2}$  for each of them. Then, for each parameter valuation, the corresponding action is probabilistic, leading to different states. What makes this transition interesting is that for different parameter valuations, some actions lead to the same state:

- the global probability of returning to (s<sub>1</sub>) is  $\frac{4}{5} \times \frac{1}{2} \times \frac{4}{10} = \frac{4}{25}$ ,
- the global probability of reaching (s<sub>3</sub>) is  $\frac{4}{5} \times \frac{1}{2} \times \frac{4}{10} = \frac{4}{25}$ ,

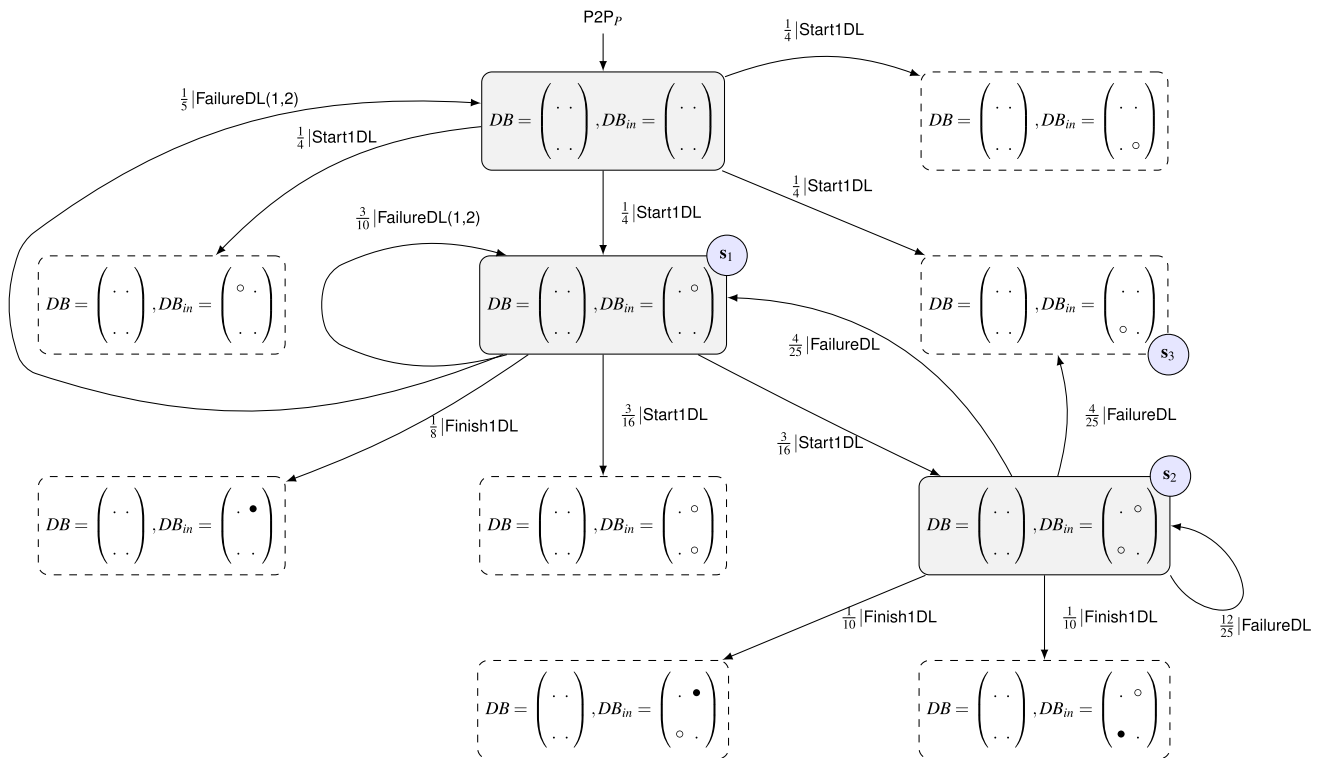


Fig. 9 Extract of the DTMC of the simple P2P protocol, with N=2 and K=2

- the global probability of looping on (s<sub>2</sub>) is  $\frac{4}{5} \times (\frac{1}{2} \times \frac{6}{10} + \frac{1}{2} \times \frac{6}{10}) = \frac{12}{25}$ , where (2, 1) and (1, 2) are the parameter valuations leading to these probabilistic choices.

After reduction, we obtain the DTMC given Fig. 9.

### 5 Mixed Event-B

After presenting *fully probabilistic Event-B models*, we now move to the context of *mixed Event-B models*, i.e. Event-B models containing both probabilistic and non-deterministic events. The syntax and operational semantics we propose combine those introduced in state of the art models [35–37] and in Sect. 4 by allowing probabilistic choice to be expressed in all places where non-determinism exists, instead of limiting its existence to probabilistic assignments, but also retaining non-determinism wherever needed.

#### 5.1 Description

Mixed Event-B models can be obtained through several means. While they can be produced as standalone models for describing systems containing both non-deterministic and

probabilistic behaviours, they can also be produced as intermediate models during the step-by-step refinement process. In this last setting, a mixed Event-B model can be obtained by adding probabilistic events within a standard Event-B model, by adding standard events within a fully probabilistic Event-B model, or by refining some standard events into probabilistic events in a standard Event-B model.

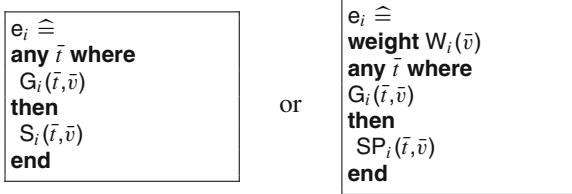
Regardless of how they are produced, mixed Event-B models are models that contain both standard (non-deterministic) events and probabilistic events. We distinguish two classes of mixed Event-B models and study them separately in the rest of this section: (i) *partially mixed Event-B models* and (ii) *fully mixed Event-B models*. The former are models where the standard and probabilistic events do not interact: their guards are necessarily disjoint. In *partially mixed Event-B models*, standard events cannot be enabled in the same valuations where probabilistic events are enabled, and vice-versa. On the other hand, *fully mixed Event-B models* allow probabilistic and non-deterministic events to be enabled in the same configurations (variable valuations).

#### 5.2 Syntax

Since mixed Event-B models contain both standard and probabilistic events, they combine the standard Event-B syntax and the probabilistic syntax introduced in Sect. 4. Therefore,



events in a mixed Event-B model have one of the forms presented below.



For simplicity reasons we impose, as in standard and probabilistic Event-B, that the initialisation event must be deterministic.

**Definition 3 (Mixed Event-B model)** A mixed Event-B model is a tuple  $M=(\bar{v}, I(\bar{v}), \text{MEvts}, \text{Init})$  where  $\bar{v}=\{v_1 \dots v_n\}$  is a set of variables,  $I(\bar{v})$  is the invariant, **MEvts** is a set containing both *standard* and *probabilistic* events, and **Init** is the initialisation event.

### 5.3 Consistency

As in standard Event-B, the consistency of a mixed Event-B model is defined by means of proof obligations (POs). In this section, we discuss specific POs for mixed Event-B models. In particular, we introduce new POs specific to such models and discuss how regular POs for standard and fully probabilistic Event-B can be adapted to this setting.

#### 5.3.1 Case of partially mixed Event-B models

We first consider partially mixed Event-B models. Recall that, in such models, the guards of non-deterministic and probabilistic events are necessarily disjoint. As a consequence, no non-deterministic events can be enabled in configurations (variable valuations) where probabilistic events are enabled, and vice-versa. In such models, it is therefore possible to partition the set of configurations into configurations where non-deterministic events are enabled and configurations where probabilistic events are enabled. Depending on which type of configuration is considered, we will therefore apply standard POs specific to non-deterministic events or POs dedicated to probabilistic events as introduced in Sect. 4.2.

Let  $M=(\bar{v}, I(\bar{v}), \text{MEvts}, \text{Init})$  be a partially mixed Event-B model.  $\text{MEvts} = \{e_1 \dots e_i \dots e_n\}$  is the set of events of  $M$ , which we partition into  $\{e_1 \dots e_i\}$ , the subset of standard events, and  $\{e_{i+1} \dots e_n\}$ , the subset of probabilistic events.

In order to ensure that a mixed Event-B model is *partially mixed*, we propose the two following POs.

- 1. Enabledness of standard events:** This PO states that in configurations where at least one standard event is enabled, no probabilistic event is enabled. It is written as follows:

$$I(\bar{v}) \wedge G_1(\bar{i}, \bar{v}) \vee \dots \vee G_i(\bar{i}, \bar{v}) \vdash \neg ((G_{i+1}(\bar{i}, \bar{v}) \wedge W_{i+1}(\bar{v}) > 0) \vee \dots \vee (G_n(\bar{i}, \bar{v}) \wedge W_n(\bar{v}) > 0)) \quad (\text{mixedEB/csrt1})$$

- 2. Enabledness of probabilistic events:** This PO states that in configurations where at least one probabilistic event is enabled, no non-deterministic event is enabled. It is written as follows:

$$I(\bar{v}) \wedge ((G_{i+1}(\bar{i}, \bar{v}) \wedge W_{i+1}(\bar{v}) > 0) \vee \dots \vee (G_n(\bar{i}, \bar{v}) \wedge W_n(\bar{v}) > 0)) \vdash \neg (G_1(\bar{i}, \bar{v}) \vee \dots \vee G_i(\bar{i}, \bar{v})) \quad (\text{mixedEB/csrt2})$$

As expected, a mixed Event-B model that satisfies these two POs is *partially mixed*: the set of configurations where standard events are enabled is disjoint from the set of configurations where probabilistic events are enabled.

The consistency of such a model is then ensured by applying standard POs [1] for non-deterministic events and the dedicated POs for probabilistic events that are introduced in Sect. 4.2.

#### 5.3.2 Case of fully mixed Event-B models

We now move to the more general context of fully mixed Event-B models. Recall that, in this context, probabilistic and non-deterministic events can be enabled in the same configurations. Luckily, most consistency POs as introduced in [1] and in Sect. 4.2 remain valid in this new setting: they can be used for their specific purpose in the same configuration. The only PO that requires modification due to the interaction between probabilistic and non-deterministic events is the deadlock freedom. While this PO is optional in Event-B, it is often used. We therefore present the required modifications hereafter.

**Deadlock freedom** In standard Event-B, this PO consists in proving that, in all acceptable configurations, there is always at least one enabled event. In standard Event-B, we recall that an event is enabled only if its guard is fulfilled. In probabilistic Event-B, an event is enabled if in addition to its guard being fulfilled, its weight is strictly positive. In the case of *Fully mixed Event-B models*, this PO will therefore ensure that, in a given configuration where both standard and probabilistic events are enabled, there is at least one standard event whose guard is fulfilled or a probabilistic event whose guard is fulfilled and whose weight is strictly positive.

We therefore rewrite it as follows :

$$I(\bar{v}) \vdash (G_1(\bar{i}, \bar{v}) \vee \dots \vee G_m(\bar{i}, \bar{v})) \vee (G_{m+1}(\bar{i}, \bar{v}) \wedge W_{m+1}(\bar{v}) > 0) \vee \dots \vee ((G_n(\bar{i}, \bar{v}) \wedge W_n(\bar{v}) > 0)) \quad (\text{model/mDLF})$$

### 5.4 Operational semantics

As presented, respectively, in Sects. 2.2 and 4.3, the operational semantics of standard Event-B models is expressed in terms of labelled transition systems, while the operational semantics of fully probabilistic Event-B models is expressed in terms of discrete-time Markov chains. In the following, we extend these constructions by presenting the operational semantics of mixed Event-B models in terms of Markov decision processes. As in Sect. 4.3, our goal is not to translate mixed models into MDPs and use standard model-checking techniques to verify them. Again, MDP semantics are only introduced as a demonstration of the correctness of our approach.

*MDP operational semantics* Informally, the semantics of a mixed Event-B model  $M=(\bar{v}, I(\bar{v}), MEvts, I_{init})$  is expressed by means of an MDP  $\llbracket M \rrbracket=(S, Acts, P, l_{init}, AP, L)$  where the states, labels and atomic propositions are obtained as for standard and probabilistic semantics of Event-B. For the initial distribution  $l_{init}$ , we recall that we only consider mixed Event-B models with deterministic initialisation event. As a consequence we have one initial state  $s_0$  obtained after the execution of the initialisation event and then we have  $l_{init}(s_0) = 1$ . The major difference between the semantics we introduced hereafter and both standard and probabilistic semantics concerns actions and transitions. We recall that in the LTS semantics of a standard Event-B model, the transitions are not equipped with probabilities and the choice between transitions enabled in a given state is done in a non-deterministic manner. In the DTMC semantics of a fully probabilistic Event-B model, the transitions are equipped with probabilities and the choice between transitions enabled in a given state is done in a probabilistic manner. In what follows, we explain how we construct transitions for both partially and fully mixed Event-B models. We then provide

the formal definition of the operational semantics of mixed Event-B models. Again, we start by considering partially mixed Event-B models and only then move to the more general setting of fully mixed Event-B models.

*Partially mixed Event-B models* Recall that, in the case of partially mixed Event-B models, the guards of probabilistic and non-deterministic events are disjoint. As a consequence, states in the operational semantics of such models will be divided into states where standard events can be performed and states where probabilistic events can be performed. In order to obtain a well-defined MDP semantics, we will thus introduce two types of probabilistic transitions. In the case of standard events, the corresponding transitions will consist in probabilistic transitions that assign a probability 1 to the target states. We will therefore conserve the non-deterministic choice between standard events in the resulting MDP as a non-deterministic choice between the corresponding probabilistic transitions. On the other hand, in the case of probabilistic events, the corresponding transitions will be computed as in the case of the fully probabilistic Event-B setting, i.e. resulting in a single probabilistic transition where the probability distribution encompasses the probabilistic choices introduced in all enabled probabilistic events. Since standard MDP notations require that transitions are equipped with an action name, we introduce a new action name *Prob* that will represent the execution of a probabilistic transition in states where one is enabled. We emphasise the fact that this new action name only appears in the MDP semantics but does not correspond to a merging of the probabilistic events.

*Example 1* Consider the very simple Event-B model and the corresponding MDP operational semantics given in Fig. 10. Obviously, it is a *partially mixed Event-B model*.

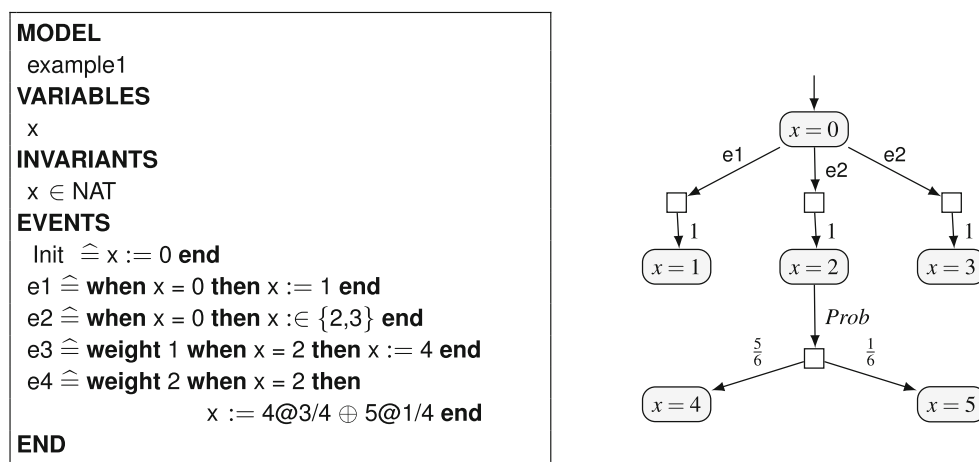


Fig. 10 Partially mixed Event-B model and MDP semantics for Example 1

- From the initial state ( $x = 0$ ) only two non-deterministic events are enabled:  $e1$  and  $e2$  which, respectively, lead to ( $x = 1$ ) (with  $e1$ ) and ( $x = 2$ ) or ( $x = 3$ ) (with  $e2$ ), which corresponds to the non-deterministic choice in the action of  $e2$ .
- From the state ( $x = 2$ ), two probabilistic events are enabled:  $e3$  and  $e4$  with respective weights 1 and 2 and corresponding probabilities  $\frac{1}{3}$  and  $\frac{2}{3}$ .  $e3$  leads to ( $x = 4$ ) with probability 1, whereas  $e4$  leads to ( $x = 4$ ) with a probability of  $\frac{3}{4}$  and to ( $x = 5$ ) with a probability of  $\frac{1}{4}$ , which corresponds to the probabilistic choice in the action of  $e4$ .

As a consequence, there are 3 probabilistic transitions from state ( $x = 0$ ), corresponding to all the non-deterministic choices present in this configuration. The choice between these transitions is non-deterministic. On the contrary, there will only be one probabilistic transition from state ( $x = 2$ ), encompassing all the probabilistic choices present in this configuration. The corresponding probability distribution is as follows.

- State ( $x = 4$ ) is reached with a probability of  $\underbrace{\left(\frac{1}{3} \times 1\right)}_{e3} + \underbrace{\left(\frac{2}{3} \times \frac{3}{4}\right)}_{e4} = \frac{5}{6}$ ,
- State ( $x = 5$ ) is reached with a probability of  $\frac{2}{3} \times \frac{1}{4} = \frac{1}{6}$ .

We remark that probabilistic event names do not appear in the MDP semantics presented above. Indeed, the new action name *Prob* is used in order to represent the single probabilistic transition encoding the behaviour of all probabilistic events at once. Since the probability distribution attached to this transition is on the states of the MDP, probabilistic event names are lost. It would be easy to extend our definition and notations in order to include probabilistic event names, e.g. by defining a probability distribution on pairs made of event

names and states, but we do not do it here as probabilistic event names are not used in the rest of this section.

*Fully mixed Event-B models* In the context of fully mixed Event-B models, probabilistic events may be enabled in the same configurations as standard events. As a consequence, states in the corresponding MDP semantics will allow transitions either corresponding to standard events or corresponding to probabilistic events. We propose to treat them in a similar manner as in the case of partially mixed Event-B models. Instead of having either a non-deterministic choice between the probabilistic transitions corresponding to standard events **or** a single probabilistic transitions encoding the behaviour of all probabilistic events, we propose to have a non-deterministic choice between all potential transitions, i.e. all probabilistic transitions corresponding to standard events **and** the single probabilistic transition encoding the behaviour of all probabilistic events. As in the case of partially mixed Event-B models, we introduce a new action name *Prob* for the probabilistic transition corresponding to probabilistic events. Remark we have deliberately chosen to resolve the non-deterministic choice between events before resolving the probabilistic choice. The same order is applied in the case of MDPs as well as in all existing approaches introducing probabilistic substitutions in Event-B [19,21,37].

*Example 2* We propose a slightly modified version of Example 1. Consider the Event-B model and its corresponding MDP operational semantics given in Fig. 11. Obviously, it is a *fully mixed Event-B model*. The only differences w.r.t. the model from Example 1 are that (1) all events are enabled from the initial state ( $x = 0$ ), and (2) the probabilistic action of event  $e4$  leads to states ( $x = 4$ ) and ( $x = 3$ ) instead of ( $x = 4$ ) and ( $x = 5$ ) (but the probabilities are untouched).

As a consequence, there are now 4 probabilistic transitions from state ( $x = 0$ ), 3 of them corresponding the non-deterministic choices from events  $e1$  and  $e2$ , and one that encompasses all the probabilistic choices from events  $e3$  and  $e4$ . The choice between these 4 transitions is non-

```

MODEL
example2
VARIABLES
x
INVARIANTS
x ∈ NAT
EVENTS
Init ≙ x := 0 end
e1 ≙ when x = 0 then x := 1 end
e2 ≙ when x = 0 then x ∈ {2,3} end
e3 ≙ weight 1 when x = 0 then x := 4 end
e4 ≙ weight 2 when x = 0 then
      x := 4@3/4 ⊕ 3@1/4 end
END
    
```

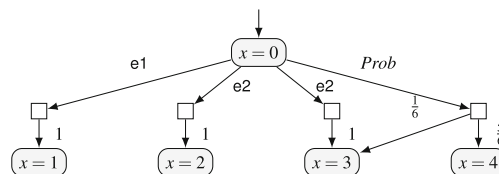


Fig. 11 Fully mixed Event-B model and MDP semantics for Example 2

deterministic. The computation of the actual probability distributions is similar to the one presented in Example 1, but remark that state ( $x = 3$ ) can now be reached through two distinct transitions: with probability 1 using one of the transitions labelled with  $e2$ , and with probability  $\frac{1}{6}$  using the transition labelled with  $Prob$ .

We now propose a formal definition for the MDP semantics of a mixed Event-B model (regardless of whether it is partially or fully mixed).

**Definition 4** (MDP semantics for mixed Event-B models)

The operational semantics of a mixed Event-B model  $M=(\bar{v}, l(\bar{v}), MEvts, l_{init})$  is an MDP  $\llbracket M \rrbracket = (S, Acts, T, l_{init}, AP, L)$  where the states  $S$ , the initial state  $l_{init}$ , the atomic propositions  $AP$  and labels  $L$  are defined as in the standard and fully probabilistic operational semantics of Event-B. As explained above, the action names  $Acts$  are the names of standard events, to which we add the new action name  $Prob$ . Formally,  $Acts = Acts_{nd} \cup \{Prob\}$ . Finally,  $T \subseteq S \times Acts \times Dist(S)$  is the transition relation such that  $(s, e, \delta) \in T$  iff either

- $e \in Acts_{nd}$  and there exists  $s' \in S$  such that  $(s, e, s')$  is a transition in the standard (non-deterministic) version of  $M$ , and  $\delta(s'') = 1$  if  $s'' = s'$  and 0 otherwise, or
- $e = Prob$  and  $\delta(s'') = \sum_{e_k \in Acts_p} P(s, e_k, s'')$ , with  $P$  defined as in Definition 2.

We remark that probabilistic event names do not appear in the above MDP semantics, as we only represent the overall transition probabilities. Nevertheless, it is easy to include these event names in the transition relation by extending it from a distribution on states to a distribution on pairs made of event names and states (as is done in Sect. 4.3). However, we choose to leave it out for now as it is an unnecessary feature for our purpose here.

In the following proposition, we show that, as expected, the operational semantics of a mixed Event-B model as defined above is indeed an MDP.

**Proposition 2** The operational semantics of a mixed Event-B model  $M$  satisfying the POs given in Sect. 5.3 is an MDP.

*Proof-sketch* Similarly to Proposition 1, the aim here is to prove that the transition function defined in Definition 4 leads to valid probability distributions. We must therefore prove that for all  $s \in S$  and  $(s, e, \delta) \in T$ , (1)  $\delta(s') \in [0, 1]$  for all  $s' \in S$  and (2)  $\sum_{s' \in S} \delta(s') = 1$ .

- If  $e \in Acts_{nd}$ , this clearly follows from the definition of  $\delta$ .
- Otherwise, we have  $e = Prob$  and therefore  $\delta(s') = \sum_{e_k \in Acts_p} P(s, e_k, s')$ , with  $P$  defined as in Definition 2. As a consequence,

$$\sum_{s' \in S} \delta(s') = \sum_{s' \in S} \sum_{e_k \in Acts_p} P(s, e_k, s')$$

By Proposition 1, we know that (1)  $\sum_{e_k \in Acts_p} P(s, e_k, s') \geq 0$ , and (2)  $\sum_{s' \in S} \sum_{e_k \in Acts_p} P(s, e_k, s') = 1$ , so  $\sum_{s' \in S} \delta(s') = 1$ , which also implies that  $\sum_{e_k \in Acts_p} P(s, e_k, s') \leq 1$ .  $\square$

**6 Introducing probabilities by refinement**

Our main goal is to enable modelling probabilistic behaviours within Event-B. As explained earlier (and illustrated in Fig. 1), this can be done in several ways while preserving the refinement-based approach inherent to Event-B. In this section, we present how to turn a standard (group of) event(s) into a probabilistic (group of) event(s). In the next section we explain how probabilistic information can be introduced earlier in the design process by introducing new probabilistic events through refinement.

The *probabilisation* process is a refinement-like process that consists in transforming a set of non-deterministic events in a given model into probabilistic events, while keeping the rest of the model untouched. Depending on the type of model from which these events are taken, the result of this operation could be a (fully or partially) mixed or a fully probabilistic Event-B model. This process could typically be used as a last step of the refinement chain, allowing to transform a fully detailed standard Event-B model into a fully probabilistic one. The resulting model has the same elements as the original one: we do not allow the introduction of new variables, constants or new events; each event keeps its elements; we do not add new parameters, reinforce the guards or add new assignments to the events. The only difference between the two models is the introduction of event weights and assignment probabilities for the considered set of events.

In order to probabilise a set of standard events, all the events in this set must satisfy some conditions, formalised by means of *probabilisation feasibility* POs:

1. **Parameter probabilisation.** For each concerned standard event, the set of values taken by the parameters such that the guard of the event is fulfilled must be finite.

$l(\bar{v}) \vdash \text{finite}(\{\bar{t} \mid G(\bar{t}, \bar{v})\})$	(event/param/proba)
---	---------------------

2. **Event probabilisation.** Depending on the type of assignment, several POs can be applied:
  - (a) **Enumerated Assignment probabilisation.** For each enumerated assignment  $\mathbf{x} : \in \{E_1(\bar{t}, \bar{v}), \dots, E_i(\bar{t}, \bar{v}), \dots, E_n(\bar{t}, \bar{v})\}$  which appears in the concerned events of

the standard Event-B model, the set of expressions assigned to the corresponding variable must be finite.

$$\vdash \text{finite}(\{E_1(\bar{i}, \bar{v}), \dots, E_i(\bar{i}, \bar{v}), \dots, E_n(\bar{i}, \bar{v})\}) \quad (\text{event/assign/proba})$$

- (a) **Predicate Assignment probabilisation.** For each predicate non-deterministic assignment  $x :| Q_x(\bar{i}, \bar{v}, x')$  which appears in the concerned events of the standard Event-B model, the set of values  $x'$  such that  $Q_x(\bar{i}, \bar{v}, x')$  is true must be finite.

$$\vdash \text{finite}(\{x' \mid Q_x(\bar{i}, \bar{v}, x')\}) \quad (\text{event/assign/proba})$$

When all the above conditions are fulfilled, the considered set of events can be probabilised. Then, the probabilisation process consists in:

1. Producing a new *probabilistic/mixed* Event-B model which **probabilises** the original model.

As the variables and the invariants are not impacted by the probabilisation, they are automatically included in the generated Event-B model;

2. Probabilising each event of the given set, *i.e.*
  - (a) Copying each event of the original model into the new probabilistic/mixed Event-B model;
  - (b) Annotating each event with a weight expression  $W(\bar{v})$ ;
  - (c) Replacing each enumerated assignment by an enumerated probabilistic assignment in the form:

$$x := E_1(\bar{i}, \bar{v})@p_1 \oplus \dots \oplus E_i(\bar{i}, \bar{v})@p_i \oplus \dots \oplus E_n(\bar{i}, \bar{v})@p_n$$

where the designer will have to precise the desired probability values  $p_1, \dots, p_i, \dots, p_n$ .

- (d) Replacing each predicate non-deterministic assignment by the corresponding predicate probabilistic assignment:

$$x: \oplus Q_x(\bar{i}, \bar{v}, x')$$

As default (proposed) values, event weights are set to 1 and the parameters of probabilistic assignments are uniform ( $p_i = \frac{1}{n}, i = 1..n$ ).

*Running example* The non-deterministic Event-B model  $P2P_3$  corresponding to the second level of the refinement on the case study presented in Sect. 3 can be probabilised. For illustration purposes, we apply a partial probabilisation on the selected events **Finish1DL** and **FailureDL**.

During this process, we need to input weights for both events. We propose expressions that imply that the number of failures decreases with the number of successful downloads. When probabilising the event **FailureDL**, we need to input probability values for the transformation of the non-deterministic choice  $DBin(c \mapsto b) \in \{\text{empty}, \text{incoming}\}$  to a probabilistic choice  $DBin(c \mapsto b) = \text{empty} @4/10 \oplus \text{incoming} @6/10$ .

Finally, we obtain a fully mixed event-B model  $P2P_M$  illustrated in Fig. 12 where the events **DLFinished** and **Start1DL** are still non-deterministic whereas **Finish1DL** and **FailureDL** are probabilistic.

When applying probabilisation on all the events of the non-deterministic Event-B model given in Fig. 5, we obtain the fully probabilistic Event-B model given in Fig. 7, in Sect. 4.1.

## 7 Introducing new probabilistic events by refinement

As previously explained, probabilistic information can be introduced in the design process by introducing new probabilistic events through (probabilistic) refinement. Obviously, this process could be applied to standard, mixed or even fully probabilistic models, therefore leading to mixed or fully probabilistic models.

One principal aspect of refinement in Event-B is the addition within a refinement step of new variables and new events acting on those variables. In this section, we explain how to introduce new probabilistic events in a given (abstract) model. Regardless of the type of model (non-deterministic, mixed or fully probabilistic), it is necessary to show that the introduction of these new events cannot prevent the system from behaving as specified in the abstract model. Recall that this is usually done by proving that the set of new events introduced in the refinement step converges, *i.e.* that events from this set cannot keep control indefinitely. As a consequence, at some point, the system must stop the execution of new events in order to execute the behaviour proposed in the abstract model.

We therefore propose a solution in order to prove that a given set of probabilistic events almost-certainly converges. As a first step, we only consider fully probabilistic Event-B models and propose a set of sufficient conditions, expressed as POs, that allow proving that a set of probabilistic events is almost-certainly convergent. We then explain how these POs can be extended to the more general setting.

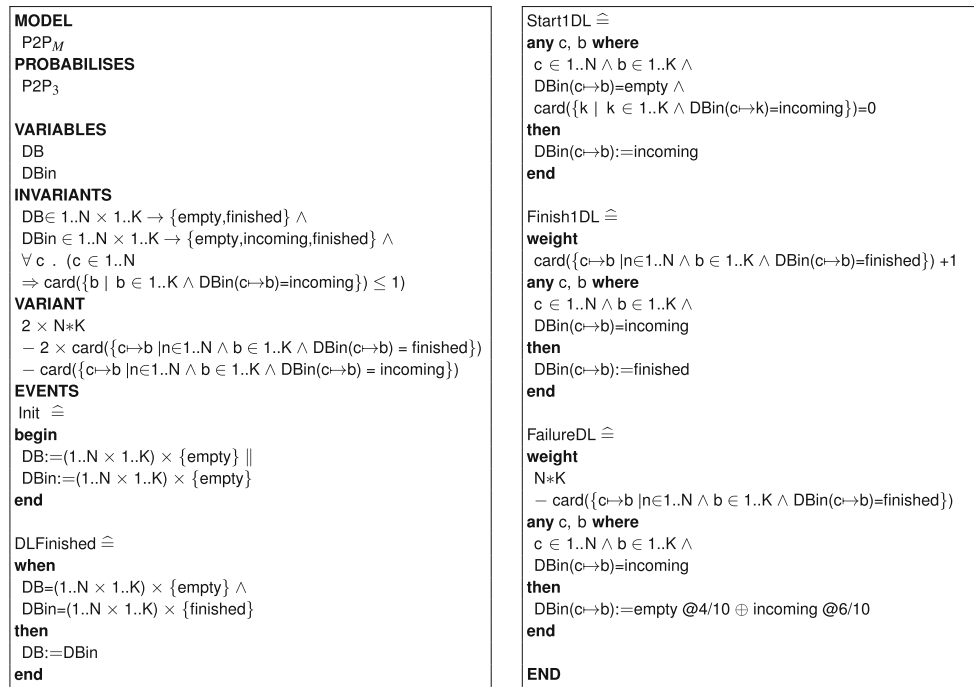


Fig. 12 A fully mixed Event-B version of the simple P2P protocol

Finally, we briefly consider the dual process: introducing standard (non-deterministic) events inside a probabilistic (or mixed) Event-B model.

### 7.1 Introducing new events in a fully probabilistic Event-B model

In standard Event-B refinement, it is required to show that a given set of events *always converges*. On the contrary, in probabilistic Event-B, it is only required to prove that a given set of probabilistic events *almost-certainly* converges. In other words, we are interested in showing that, in all states of the system where convergent probabilistic events can be executed, the probability of eventually taking a non-convergent event or reaching a deadlock is 1 (the probability of infinitely executing convergent events is 0).

This property has already been investigated in [19] and [22], in the context of events having probabilistic actions but where non-determinism is still present between events. In this context, Hallerstede and Hoang propose in [19] sufficient conditions for a set of events to almost-certainly converge. These conditions can be summarised as follows: As in standard Event-B, one needs to exhibit a natural number expression  $V(\bar{v})$  called a variant. Unlike in the standard setting, only one resulting valuation of the execution of *each* convergent event needs to decrease this variant. Indeed, in this case, the probability of decreasing the variant is strictly positive. Unfortunately, using such a permissive condition is not

sufficient in our context: there might also be a strictly positive probability of increasing the variant. Therefore, Hallerstede and Hoang require the introduction of another natural number expression  $U(\bar{v})$  which must maximise the variant  $V(\bar{v})$  and never increase. The proposition from [19] is refined in [22], where Hoang requires in addition that the probabilities considered in probabilistic assignments are bounded away from 0. This is ensured by requiring that the set of values that can be returned by a probabilistic assignment is finite.

#### 7.1.1 Adaptation to probabilistic events

We now show how to adapt the results proposed in [19] and [22] to new probabilistic events introduced in a fully probabilistic Event-B model. Since there are no non-deterministic choices between enabled events, it is not anymore necessary to require that *all* enabled events in a given configuration may decrease the variant. We therefore start by relaxing the condition proposed in [19]: we only require that, in all configurations where a convergent event is enabled, there is *at least one* convergent event for which at least one resulting valuation decreases the variant.

1. **Almost-certain convergence** In all configurations where at least one convergent event is enabled, there must exist at least one valuation  $\bar{v}'$  obtained after the execution of one of these enabled events which decreases the variant.

$$I(\bar{v}) \wedge ((G_i(\bar{t}, \bar{v}) \wedge W_i(\bar{v}) > 0) \vee \dots \vee (G_n(\bar{t}, \bar{v}) \wedge W_n(\bar{v}) > 0)) \vdash \quad (\text{model/pVar})$$

$$(\exists \bar{v}'. G_i(\bar{t}, \bar{v}) \wedge W_i(\bar{v}) > 0 \wedge SP_i(\bar{t}, \bar{v}, \bar{v}') \wedge V(\bar{v}') < V(\bar{v})) \vee \dots \vee (\exists \bar{v}'. G_n(\bar{t}, \bar{v}) \wedge W_n(\bar{v}) > 0 \wedge SP_n(\bar{t}, \bar{v}, \bar{v}') \wedge V(\bar{v}') < V(\bar{v}))$$

As in [19], we also require that convergent events can only be enabled when the variant is positive and that the variant is bounded above. In order to simplify the reasoning, we propose to use a constant bound  $U$ , as in [22].

2. **Numeric variant** Convergent events can only be enabled when the variant is greater or equal to 0.

$$I(\bar{v}) \wedge G_i(\bar{t}, \bar{v}) \wedge W_i(\bar{v}) > 0 \vdash V(\bar{v}) \in \text{NAT} \quad (\text{event/var/pNAT})$$

3. **Bounded variant** Convergent events can only be enabled when the variant is less or equal to  $U$ .

$$I(\bar{v}) \wedge G_i(\bar{t}, \bar{v}) \wedge W_i(\bar{v}) > 0 \vdash V(\bar{v}) \leq U \quad (\text{event/pBOUND})$$

Finally, the finiteness of the set of values that can be returned by a probabilistic assignment is already ensured by the syntax for enumerated probabilistic assignments and by PO (event/assign/pWD3) for predicate probabilistic assignments and their non-emptiness is ensured by the standard feasibility POs.

### 7.1.2 Inadequacy of adapted POs

Unfortunately, as we deal with potentially infinite-state systems, POs 1–3 presented above are not anymore sufficient for proving that the probability of eventually executing a non-convergent event or reaching a deadlock is 1. Indeed, although the probability of decreasing the variant is always strictly positive because of PO (model/pVar) and although the number of values that can be returned by a given probabilistic assignment is always finite, the combination of event weights and parameter choice can make this value infinitely small in some cases. In this case, it is well known that almost-certain reachability/convergence is not ensured. This problem is a direct consequence of the unboundedness of the weights of convergent events as well as of the number of acceptable parameter values, which, by getting arbitrarily big, cause the probability of decreasing the variant to get arbitrarily small. Two examples illustrating this fact are given below.

*Example 3* (Necessity of bounding event weights) In this example, we show by means of an example of a probabilistic Event-B model the necessity of bounding the weights of new probabilistic events in order to ensure almost-certain convergence.

Consider the probabilistic Event-B model **M1** and the corresponding DTMC semantics given in Fig. 13. This model has two variables:  $x$  and  $y$  and three events **evt1**, **evt2** and **evt3**, two of which (**evt1** and **evt2**) are convergent. The variant of this model is  $x$ , and the bound on the variant is clearly  $U = 2$ .

In states where  $x = 1$ , only convergent events **evt1** and **evt2** are enabled and the local probability of choosing **evt1** is  $\frac{1}{y}$ , while the local probability of choosing **evt2** is  $\frac{y-1}{y}$ . In states where  $x = 2$ , only **evt1** can be chosen with probability 1. In states where  $x = 0$ , the only enabled event is the (non-convergent) event **evt3**.

Clearly, the model **M1** satisfies proof obligations (model/pVar), (event/var/pNAT) and (event/pBOUND). However, as we show below, the probability of eventually taking a non-convergent event is strictly smaller than 1 from all states where  $x > 0$  because the probability of decreasing the variant, although strictly positive in all states, gets infinitely small from states where  $x = 1$  as  $y$  increases.

Without loss of generality, we compute the probability of eventually taking **evt3** from the initial state where  $x = 1$  and  $y = 2$ . The reasoning starting from other states is similar. This probability is equal to the sum of

- (1) the probability of directly taking **evt1** from (1, 2),
- (2) the probability of reaching (1, 4) and taking **evt1** from (1, 4),
- (3) the probability of reaching (1, 8) and taking **evt1** from (1, 8)
- (4) ...

Clearly, (1) is equal to  $\frac{1}{2}$ , (2) is equal to  $\frac{1}{2} \cdot \frac{1}{4} = \frac{1}{8}$ , (3) is equal to  $\frac{1}{2} \cdot \frac{3}{4} \cdot \frac{1}{8} < \frac{1}{16}$ , and in general, the probability of reaching state (1,  $2^i$ ) with  $i > 2$  and taking **evt1** from this state is strictly smaller than  $\frac{1}{2^{i+1}}$ .

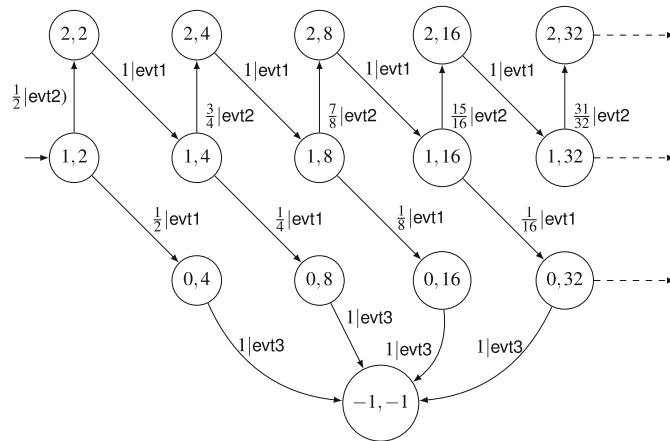
As a consequence, the probability of eventually taking **evt3** from the initial state is strictly smaller than

$$\frac{1}{2} + \sum_{i=2}^{\infty} \frac{1}{2^{i+1}} = \frac{3}{4}$$

Therefore, **M1** does not almost-certainly converge.

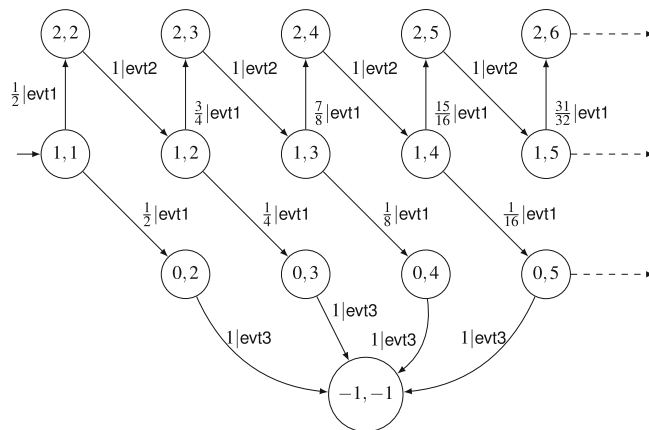
The behaviour we expose here is a direct consequence of the unboundedness of the weights of convergent events, which, by getting arbitrarily big, cause the probability of decreasing the variant to get arbitrarily small.

<b>MODEL</b> M1	$evt1 \triangleq$ <b>convergent</b> <b>weight 1</b> <b>when</b> $0 < x \leq 2$ <b>then</b> $x := x - 1 \parallel y := 2 * y$ <b>end</b>
<b>VARIABLES</b> $x$ $y$	
<b>INVARIANTS</b> $x \in \text{INT} \wedge$ $y \in \text{INT}$	$evt2 \triangleq$ <b>convergent</b> <b>weight</b> $y - 1$ <b>when</b> $0 < x \leq 1$ <b>then</b> $x := x + 1$ <b>end</b>
<b>VARIANT</b> $x$	
<b>EVENTS</b>  $Init \triangleq$ <b>begin</b> $x := 1 \parallel y := 2$ <b>end</b>	$evt3 \triangleq$ <b>weight 1</b> <b>when</b> $x = 0$ <b>then</b> $x := -1 \parallel y := -1$ <b>end</b>



**Fig. 13** Probabilistic Event-B model and DTMC semantics illustrating the necessity of bounding event weights to ensure almost-certain convergence

<b>MODEL</b> M2	$evt1 \triangleq$ <b>convergent</b> <b>weight 1</b> <b>any</b> $t$ <b>where</b> $t \in \{1..2^y\} \wedge 0 < x \leq 1$ <b>then</b> $x := ((t = 1 \wedge x' - x + t = 0) \text{ or } (2 \leq t \leq 2^y \wedge x' - x - 1 = 0))$ $\parallel y := y + 1$ <b>end</b>
<b>VARIABLES</b> $x$ $y$	
<b>INVARIANTS</b> $x \in \text{INT} \wedge$ $y \in \text{INT}$	$evt2 \triangleq$ <b>convergent</b> <b>weight 1</b> <b>when</b> $x = 2$ <b>then</b> $x := x - 1$ <b>end</b>
<b>VARIANT</b> $x$	
<b>EVENTS</b>  $Init \triangleq$ <b>begin</b> $x := 1 \parallel y := 1$ <b>end</b>	$evt3 \triangleq$ <b>weight 1</b> <b>when</b> $x = 0$ <b>then</b> $x := -1 \parallel y := -1$ <b>end</b>



**Fig. 14** Probabilistic Event-B model and DTMC semantics illustrating the necessity of bounding event parameter values to ensure almost-certain convergence

*Example 4* (Necessity of bounding event parameter values)  
 We now use a similar example to show the necessity of bounding the number of admissible parameter values in new probabilistic events in order to prove their almost-certain convergence. The probabilistic Event-B model M2 and its corresponding DTMC semantics, given in Fig. 14 are similar to the ones presented in Fig. 13.

In this case also, we observe that the probability of eventually executing non-convergent event  $evt3$  from the initial state is strictly smaller than  $3/4$ . The main difference is that, in M2, only the choice of parameter values is responsible for infinitely decreasing the probabilities of decreasing the variant.



7.1.3 Additional proof obligations

We therefore adapt classical results from infinite-state DTMC to our setting and propose sufficient conditions in terms of proof obligations to prove the almost-certain convergence of the set of new introduced events. Informally, the following POs ensure that the probability of decreasing the variant cannot get infinitely small by requiring that both the weights of convergent events and the number of potential values given to parameters in convergent events are bounded.

- 4. **Bounded weight** The weight of all convergent events must be bounded above by a constant upper bound BW.

$$I(\bar{v}) \wedge G_i(\bar{t}, \bar{v}) \vdash W_i(\bar{v}) \leq BW \quad (\text{event/wght/BOUND})$$

- 5. **Bounded parameters** The number of potential values for parameters in convergent events must be bounded above by a constant upper bound BP.

$$I(\bar{v}) \vdash \text{card}(\{\bar{t} \mid G_i(\bar{t}, \bar{v})\}) \leq BP \quad (\text{event/param/BOUND})$$

We now formally prove that the conditions presented above are sufficient for guaranteeing the almost-certain convergence of a given set of events in a probabilistic Event-B model.

**Theorem 1** *Let  $M=(\bar{v}, I(\bar{v}), V(\bar{v}), PEvts, Init)$  be a probabilistic Event-B model and  $PEvts_c \subseteq PEvts$  a set of convergent events. If  $M$  satisfies the above POs (1-5), then the set  $PEvts_c$  almost-certainly converges.*

*Proof* Let  $M = (\bar{v}, I(\bar{v}), V(\bar{v}), Evts, Init)$  be a probabilistic Event-B model.  $Evts = Evts_{nc} \cup Evts_c$  is the partition of the set of events  $Evts$  into non-convergent events  $Evts_{nc}$  and convergent events  $Evts_c$ .

We show that if  $M$  satisfies the following convergence POs:

- 1. event/var/pNAT

$$\forall e \in Evts_c. I(\bar{v}) \wedge W_e(\bar{v}) > 0 \wedge G_e(\bar{t}, \bar{v}) \vdash V(\bar{v}) \in NAT$$

- 2. event/pBOUND

$$\forall e \in Evts_c. I(\bar{v}) \wedge W_e(\bar{v}) > 0 \wedge G_e(\bar{t}, \bar{v}) \vdash V(\bar{v}) \leq U$$

- 3. event/wght/BOUND

$$\forall e \in Evts_c. I(\bar{v}) \wedge G_e(\bar{t}, \bar{v}) \vdash W(\bar{v}) \leq BW$$

- 4. event/param/BOUND

$$\forall e \in Evts_c. I(\bar{v}) \vdash \text{card}(\{\bar{t} \mid G_e(\bar{t}, \bar{v})\}) \leq BP$$

- 5. model/pVar

$$I(\bar{v}) \wedge (G_{i+1}(\bar{t}, \bar{v}) \vee \dots \vee G_n(\bar{t}, \bar{v})) \vdash (\exists \bar{v}'. W_{i+1}(\bar{v}) \wedge G_{i+1}(\bar{t}, \bar{v}) \wedge SP_{i+1}(\bar{t}, \bar{v}) \wedge V(\bar{v}') < V(\bar{v})) \vee \dots \vee (\exists \bar{v}'. W_n(\bar{v}) \wedge G_n(\bar{t}, \bar{v}) \wedge SP_n(\bar{t}, \bar{v}) \wedge V(\bar{v}') < V(\bar{v}))$$

then  $M$  almost-certainly converges (with probability 1).

Recall that almost-certain convergence of  $M$  consists in proving that, from all valuations of the variables of  $M$  where a convergent event is enabled, the probability of eventually taking a non-convergent event or reaching a deadlock is 1. In order to prove this result, we consider a slightly modified version of the DTMC semantics of  $M$  and use classical results on DTMCs in order to show that the probability of eventually reaching a given set of states is 1 from all states where non-convergent events are enabled.

In order to take into account the difference between convergent and non-convergent events, we propose the following slightly extended version of the DTMC semantics of  $M$ . In this version, all the states are replicated in order to “remember” the last event executed.

Formally, consider the probabilistic Event-B model  $M$  introduced above and let  $\llbracket M \rrbracket = (S, s_0, AP, L, Acts, P)$  be the DTMC semantics of  $M$  as introduced in Definition 2. We build the DTMC  $\llbracket M \rrbracket' = (T, t_0, AP, L', Acts, P')$  where

- $T \subseteq S \times (Acts \cup \{\epsilon\})$  is the set of extended states, consisting in pairs  $(s, a)$  where  $s$  is a state of  $\llbracket M \rrbracket$  and  $a$  is an action (event name),
- $t_0 = (s_0, \epsilon)$  is the initial state,
- $L'$  is such that  $L'((s, a)) = L(s)$  for all  $s \in S$  and  $a \in Acts$ , and
- $P'$  is such that  $P'((s, a), e, (s', b)) = P(s, e, s')$  if  $e = b$  and 0 otherwise for all action  $a$ .

It is easy to see that  $M$  almost-certainly converges iff the probability of eventually reaching either a deadlock state or an extended state of the form  $t = (s, e)$  where  $e$  is a non-convergent event is 1 in  $\llbracket M \rrbracket'$  from all (extended) states where convergent events are enabled.

Since  $\llbracket M \rrbracket$  has a potentially infinite set of states, showing such a result is not trivial. In order to prove it, we therefore exploit existing results from the theory of DTMCs. In particular, we focus on the global coarseness property introduced in [27], which is a sufficient condition for the “decisiveness” of infinite-state Markov chains. Formally, given a Markov chain  $\mathcal{M} = (S, \mathcal{P})$  and a target set of states  $\mathcal{F} \subseteq S$ , we

say that  $\mathcal{M}$  is globally coarse w.r.t.  $\mathcal{F}$  iff there exists some minimal bound  $\alpha > 0$  such that for all state  $s \in \mathcal{S}$ , the probability of eventually reaching  $\mathcal{F}$  from  $s$  is either 0 or greater or equal to  $\alpha$ . It is then shown in [27] that whenever a Markov chain  $\mathcal{M}$  is globally coarse w.r.t. the set  $\mathcal{F}$ , the probability of eventually reaching either  $\mathcal{F}$  or a set of states  $\tilde{\mathcal{F}}$  from which  $\mathcal{F}$  cannot be reached is 1 from any state of  $\mathcal{M}$ .

In the following, we will apply this result to the DTMC  $\llbracket M \rrbracket'$  in order to prove that  $M$  almost-certainly converges.

We therefore proceed as follows:

- (a) We start with introducing notations that will be used throughout the proof.
- (b) We then propose a partition of the extended states  $T$  of  $\llbracket M \rrbracket'$  and introduce our goal set  $F \subseteq T$ .
- (c) We show that all states from each partition of  $T$  satisfy the global coarseness property w.r.t.  $F$ .
- (d) We finally show that the set  $\tilde{F}$  is empty and conclude.

We now detail each step of this proof.

- (a) Consider the following notations:
  - In the DTMC  $\llbracket M \rrbracket'$ , we partition the set of actions (event names) as follows:  $\mathbf{Acts} = \mathbf{Acts}_{nc} \cup \mathbf{Acts}_c$ , where  $\mathbf{Acts}_{nc}$  is the set of non-convergent actions and  $\mathbf{Acts}_c$  is the set of convergent actions.
  - Given an extended state  $t$  and a set of states  $G \subseteq T$ , we write  $P(t \models \diamond G)$  for the probability of eventually reaching  $G$  from  $t$ .
  - Given a predicate  $P$  and an extended state  $t = (s, a)$  of  $\llbracket M \rrbracket'$ , we write  $P(t)$  for the evaluation of  $P$  in the state  $s$ .
  - Given an extended state  $t = (s, a) \in T$ , we write  $\mathbf{Acts}(t)$  for the set of events enabled in  $s$ . Similarly, we write  $\mathbf{Acts}_c(t)$  for the set of convergent events enabled in  $s$  and  $\mathbf{Acts}_{nc}(t)$  for the set of non-convergent events enabled in  $s$ .
  - Given a set of events  $E$  and a state  $t = (s, a) \in T$ , we write  $W^t(E)$  (or  $W^s(E)$  when clear from the context) for the sum of the weights of the events from  $E$  that are enabled in  $s$ .
  - Given a state  $t = (s, a) \in T$ , we write  $Succ(t)$  for the set of extended states that are reached from  $t$ :
 
$$Succ(t) = \{t' \in T \mid \exists e \in \mathbf{Acts}(t). P^t(t, e, t') > 0\}$$
  - Given a finite execution  $\sigma = t_0, e_0, t_1, \dots, t_{n-1}, e_{n-1}, t_n$  of  $\llbracket M \rrbracket'$ , the length of  $\sigma$  is written  $L(\sigma)$  and is equal to the number of transitions executed in  $\sigma$ . In the above example case,  $L(\sigma) = n$ .

- (b) We now introduce the following sets of extended states  $T$  :

- $T_1 = \{t = (s, a) \in T \mid \exists e \in \mathbf{Evts}_c, \exists \theta \in T_s^e, G_e(s, \theta) \wedge \forall e' \in \mathbf{Evts}_{nc}, \forall \theta \in T_s^{e'}, \neg G_{e'}(s, \theta)\}$  is the set of extended states where only convergent events are enabled.
- $T_2 = \{t = (s, a) \in T \mid \exists e \in \mathbf{Evts}_c, \exists \theta \in T_s^e, G_e(s, \theta) \wedge \exists e' \in \mathbf{Evts}_{nc}, \exists \theta \in T_s^{e'}, G_{e'}(s, \theta)\}$  is the set of states where both convergent and non-convergent events are enabled.
- $T_3 = \{t = (s, a) \in T \mid \forall e \in \mathbf{Evts}_c, \forall \theta \in T_s^e, \neg G_e(s, \theta)\}$  is the set of states where no convergent events are enabled.
- $T_4 = \{t = (s, a) \in T \mid a \in \mathbf{Evts}_{nc}\}$  is the set of states reached after performing a non-convergent event.

It is easy to see that  $T = T_1 \cup T_2 \cup T_3$  defines a partition of  $T$ . The convergence property for our probabilistic Event-B model  $M$  clearly concerns states from  $T_3$  and  $T_4$ . We therefore define our target set as  $F = T_3 \cup T_4$ . As in [27], we write  $\tilde{F}$  for the subset of states of  $T$  from which it is impossible to reach  $F$ . We show later that  $\tilde{F}$  is empty.

- (c) We now show that all extended states in  $T_1$  and  $T_2$  and  $T_3$  satisfy the global coarseness property w.r.t.  $F$ , i.e. that there exists a minimal bound  $\alpha > 0$  such that for each extended state  $t \in T$ , the probability of eventually reaching  $F$  is either 0 or greater or equal to  $\alpha$ .
  - We begin with states in  $T_2$ . Let  $t_2 = (s_2, a) \in T_2$ . Let  $F_2$  be the subset of states that are reached from  $t_2$  by non-convergent events. Obviously,  $F_2 \subseteq T_4 \subseteq F$ . Formally,

$$F_2 = \{t' = (s', a') \in T \mid t' \in Succ(t_2) \wedge a' \in \mathbf{Acts}_{nc}\}$$

By definition of  $T_2$ , at least one convergent event is enabled in  $t_2$ , therefore we have  $W^{t_2}(\mathbf{Acts}_c) > 0$ . Likewise, at least one non-convergent event can be enabled in  $t_2$ , thus  $W^{t_2}(\mathbf{Acts}_{nc}) > 0$ . Therefore,  $W^{t_2}(\mathbf{Acts}) > 0$ .

Recall from Sect. 4.3 that the probability of a transition  $(t_2, e, t')$  where  $e \in \mathbf{Acts}_{nc}(t_2)$  and  $t' = (s', e) \in F_2$  is given by:

$$P^t(t_2, e, t') = P(s_2, e, s') = \frac{W_e(s_2)}{W^{s_2}(\mathbf{Acts})} \times \sum_{\theta \in T_{s_2}^e} [P_{T_{s_2}^e}(\theta) \times \prod_{x \in Var(e)} P_{s_2, \theta}^e(x, s')]$$

By definition, all non-convergent events  $e$  take the system in states in  $F_2$  regardless of the probabilistic choice made inside the action of  $e$ . Therefore:

$$\sum_{e \in \mathbf{Acts}_{nc}(s_2), t' \in F_2} P(t_2, e, t') = \sum_{e \in \mathbf{Acts}_{nc}(s_2)} \frac{W_e(s_2)}{W^{s_2}(\mathbf{Acts})} \times 1$$

As a result, the probability of eventually reaching  $F_2$  from  $t_2$  is above  $\frac{W^{s_2}(\mathbf{Acts}_{nc})}{W^{s_2}(\mathbf{Acts})}$ .

We now show by contradiction that there exists  $\alpha_2 > 0$  s.t  $\forall t_2 \in T_2, P(t_2 \models \diamond F_2) \geq \alpha_2$ .

Assume the contrary, i.e.  $\forall \alpha_2 > 0, \exists t_2 \in T_2$  s.t  $P(t_2 \models \diamond F_2) < \alpha_2$ .

Let  $\alpha_2$  be such that  $(\frac{1}{\alpha_2} - 1) > BW \times card(\mathbf{Acts}_c)$ . There must exist  $t_2 = (s_2, a) \in T_2$  such that  $P(t_2 \models \diamond F_2) < \alpha_2$ . By the result above, we know that  $P(t_2 \models \diamond F_2) \geq \frac{W^{s_2}(\mathbf{Acts}_{nc})}{W^{s_2}(\mathbf{Acts})}$ . As a consequence, we must have:

$$\frac{W^{s_2}(\mathbf{Acts}_{nc})}{W^{s_2}(\mathbf{Acts})} < \alpha_2$$

Recall that  $W^{s_2}(\mathbf{Acts}) = W^{s_2}(\mathbf{Acts}_{nc}) + W^{s_2}(\mathbf{Acts}_c)$ . Therefore,

$$\frac{W^{s_2}(\mathbf{Acts})}{W^{s_2}(\mathbf{Acts}_{nc})} = 1 + \frac{W^{s_2}(\mathbf{Acts}_c)}{W^{s_2}(\mathbf{Acts}_{nc})} > \frac{1}{\alpha_2}$$

As a consequence,

$$W^{s_2}(\mathbf{Acts}_c) > W^{s_2}(\mathbf{Acts}_{nc}) \cdot \left(\frac{1}{\alpha_2} - 1\right)$$

By definition of  $T_2$ , we have  $W^{s_2}(\mathbf{Acts}_{nc}) \geq 1$ , therefore

$$W^{s_2}(\mathbf{Acts}_c) > \left(\frac{1}{\alpha_2} - 1\right)$$

Finally, by definition of  $\alpha_2$ , we have  $W^{s_2}(\mathbf{Acts}_c) > BW \times card(\mathbf{Acts}_c)$ , which is clearly in contradiction with PO event/wght/BOUND.

We therefore conclude that there exists  $\alpha_2 > 0$  such that  $\forall t_2 \in T_2, P(t_2 \models \diamond F_2) \geq \alpha_2$ .

- We now move to extended states in  $T_1$ : we show that there exists  $\alpha_1$  such that for all extended states  $t_1 \in T_1, P(t_1 \models \diamond F) \geq \alpha_1$ .

Recall that the probability function of  $\llbracket M \rrbracket'$  is expressed as follows: For all  $t_1 = (s_1, a) \in T_1, e \in \mathbf{Acts}$ , and  $t' = (s', a) \in T$ , we have

$$P(t_1, e, t') = P(s_1, e, s') = \frac{W_e(s_1)}{W^{s_1}(\mathbf{Acts})} \times \sum_{\theta \in T_{s_1}^e} [P_{T_{s_1}^e}(\theta) \times \prod_{x \in Var(e)} P_{s_1, \theta}^e(x, s')]$$

Since  $t_1 \in T_1$ , this expression can only be nonzero if  $e$  is a convergent event. In this case, PO event/wght/BOUND ensures that  $W^{s_1}(\mathbf{Acts}) \leq BW \cdot card(\mathbf{Acts}_c)$ . Therefore,

for all convergent events enabled in  $t_1$ , we have  $\frac{W_e(s_1)}{W^{s_1}(\mathbf{Acts})} \geq \frac{1}{BW \cdot card(\mathbf{Acts}_c)}$ .

Moreover, PO event/param/BOUND ensures that the number of parameter valuations satisfying the guard of  $e$  in  $s_1$  is bounded by  $BP$ . As a consequence,

$$\sum_{\theta \in T_{s_1}^e} [P_{T_{s_1}^e}(\theta) \times \prod_{x \in Var(e)} P_{s_1, \theta}^e(x, s')] \geq \frac{1}{BP} \times \sum_{\theta \in T_{s_1}^e} [\prod_{x \in Var(e)} P_{s_1, \theta}^e(x, s')]$$

Finally, since the probabilities inside each probabilistic assignment ( $P_x^e(E)$ ) are constant and in finite number, there is a minimal value  $\beta > 0$  (which we do not detail here) such that for all  $t_1 = (s_1, a) \in T_1, e \in \mathbf{Acts}_c$ , and  $t' = (s', e) \in T$ , whenever  $P(t_1, e, t') > 0$ , we have

$$\sum_{\theta \in T_{s_1}^e} [\prod_{x \in Var(e)} P_{s_1, \theta}^e(x, s')] \geq \beta$$

As a consequence, there exists a minimal value  $\gamma > 0$  such that  $P'(t_1, e, t') \geq \gamma$  for all  $t_1 \in T_1, e \in \mathbf{Acts}_c$ , and  $t' \in T$  such that  $P'(t_1, e, t') > 0$ .

Now, let  $t_0 = (s_0, a_0) \in T_1$  be an extended state. By definition of  $T_1$  and because of POs event/pBOUND, event/var/pNAT and model/pVar, the value of the variant in  $t_0$  is between 0 and  $U$  and there must exist a transition that leads the system to an extended state  $t_1 = (s_1, a_1)$  s.t.  $V(t_1) < V(t_0)$ . Necessarily, we have  $t_1 \in T_1$  or  $t_1 \in T_2$  or  $t_1 \in T_3$ ; therefore, there must exist a finite execution  $\sigma = t_0, e_0, t_1, \dots, t_{n-1}, t_{n-1}, t_n$  with  $t_n \in T_2 \cup T_3$  and  $\forall i < n, t_i \in T_1$  and  $L(\sigma) \leq U + 1$ .

If  $t_n \in T_3 \subseteq F$ , then  $P(t_0 \models \diamond F) \geq \gamma^{U+1}$ . Otherwise, we have  $t_n \in T_2$  and  $P(t_n \models \diamond F) \geq \alpha_2$ , therefore  $P(t_0 \models \diamond F) \geq \alpha_2 \cdot \gamma^{U+1}$ .

As a consequence, since  $\alpha_2 \leq 1$ , we have  $\gamma^{U+1} \geq \alpha_2 \cdot \gamma^{U+1}$  and there exists  $\alpha_1 = \alpha_2 \cdot \gamma^{U+1} > 0$  such that for all extended states  $t_1 \in T_1, P(t_1 \models \diamond F) \geq \alpha_1$ .

- Finally, since  $T_3 \subseteq F$ , we have  $P(t_3 \models \diamond F) = 1$  for all extended states  $t_3 \in T_3$ .

We therefore conclude that  $\llbracket M \rrbracket'$  is globally coarse w.r.t  $F$ . As a consequence,  $\forall t \in T, P(t \models \diamond F \vee \diamond \tilde{F}) = 1$ .

- (d) We have shown above that for all extended states either in  $T_1, T_2$  or  $T_3$ , we have  $P(t \models \diamond F) > 0$ . Since  $T = T_1 \cup T_2 \cup T_3, \tilde{F}$  is therefore necessarily empty.

Since  $\llbracket M \rrbracket'$  is globally coarse w.r.t  $F$  and  $\tilde{F}$  is empty, we have that for all extended state  $t \in T$ , the probability of eventually reaching the target set  $F$  is 1. As a consequence, the probability of eventually reaching either a deadlock state

or an extended state of the form  $t = (s, e)$  where  $e$  is a non-convergent event is 1 in  $\llbracket M \rrbracket'$  from all (extended) states where convergent events are enabled, which concludes our proof.  $\square$

*Remark* The additional POs imposing boundedness of weights and event parameters are therefore sufficient (in addition to the adaptation of standard convergence POs presented earlier) for proving the convergence of a given set of events. While these POs are certainly restrictive, they are easily provable and seem consistent with the requirement on the boundedness of the variant. Identifying less restrictive conditions in general is still an open question.

### 7.1.4 Running example

Recall that, as explained at the end of Sect. 3, we cannot prove that the peer-to-peer protocol (with failures) always terminates, because we cannot prove the convergence of the non-deterministic Event-B model  $P2P_3$ . Indeed, the event `FailureDL` fails to decrease the corresponding variant, therefore preventing convergence.

Now consider the fully probabilistic Event-B model  $P2P_P$  given in Fig. 7 from Sect. 4. To show that the protocol always terminates we have to show that the set of events  $\{\text{Start1DL}, \text{Finish1DL}, \text{FailureDL}\}$  almost-certainly converges (i.e. converges with probability 1).

- The variant is numeric (event/var/pNAT), and we can take the expression  $2 \times N \times K$  as a possible upper bound for the variant (event/pBOUND);
- The weights of the convergent events are bounded by  $N \times K + 1$  (event/wght/BOUND);
- The possible parameter values are bounded by  $N \times K$  (event/param/BOUND);
- Finally, in each possible configuration, the event `Finish1DL` decreases the variant with a positive probability (model/pVar).

Since the model  $P2P_P$  satisfies all the required POs, Theorem 1 ensures that the set  $\{\text{Start1DL}, \text{Finish1DL}, \text{FailureDL}\}$  almost-certainly converges.

In order to illustrate that the variant indeed decreases with a positive probability from all states using probabilistic event `Finish1DL`, we provide in Fig. 15 an extract of the MC semantics of  $P2P_P$  (for  $N=2$  and  $K=2$ ), where all states are labelled with the value of the variant.

## 7.2 Generalisation to introduction of new standard/probabilistic events in standard/mixed/probabilistic Event-B models

We now move to the general setting and explain how new events of any type can be introduced in all types of mod-

els. We show that the results presented above can be easily adapted to the more general settings that combine classical and probabilistic refinement on standard, mixed or fully probabilistic Event-B models.

### New probabilistic events in standard/mixed Event-B models

The results presented above can be adapted to the setting of standard or mixed Event-B models. In both cases, the result of introducing new probabilistic events will be a mixed Event-B model. Fortunately, since only the new probabilistic events need to be (almost-certainly) convergent, the required POs are identical to the ones presented in Sect. 7.1. Although the resulting semantics is an MDP instead of a Markov chain, the almost-certain convergence results still hold.

**Proposition 3** *Let  $M = (\bar{v}, I(\bar{v}), V(\bar{v}), \text{MEvts}, \text{Init})$  be a mixed Event-B model, where  $\text{MEvts} = \text{MEvts}_{pc} \cup \text{MEvts}_{pnc} \cup \text{MEvts}_{nd}$  is the partition of the set of events into probabilistic convergent events  $\text{MEvts}_{pc}$ , probabilistic non-convergent events  $\text{MEvts}_{pnc}$  and non-deterministic events  $\text{MEvts}_{nd}$ . If  $\text{MEvts}_{pc}$  satisfies the POs (model/pVar), (event/wght/BOUND), (event/pBOUND) and (event/var/pNAT) presented in Sect. 7.1, then  $\text{MEvts}_{pc}$  almost-certainly converges (with probability 1, regardless of the non-deterministic choices).*

*Proof-sketch* Since only probabilistic events need to be convergent, non-deterministic choices have no impact on the almost-certain convergence property. One can therefore easily adapt the proof of Theorem 1 to the MDP setting, where the role of schedulers will be trivial: regardless of the chosen scheduler, the probability of eventually taking a non-convergent event will be 1.

### Standard events in mixed/probabilistic Event-B models

After explaining how to introduce probabilistic events in standard/mixed/probabilistic Event-B models, we now consider the reverse operation, i.e. the introduction of new standard events in mixed or probabilistic Event-B models. Again, since only the new (standard) events need to be convergent, it is easy to see that standard convergence POs as presented in Sect. 2.3 are sufficient for proving almost-certain convergence. Indeed, since only standard (non-deterministic) events are convergent in the resulting model, it is necessary to show that every one of them decreases the variant.

### Standard and probabilistic events at once

Finally, we consider the introduction of both standard and probabilistic events in a single refinement step. Regardless of the type of the abstract model, the resulting model will be a mixed Event-B model. This is the most complex combination as both standard and probabilistic events need to converge at the same time.

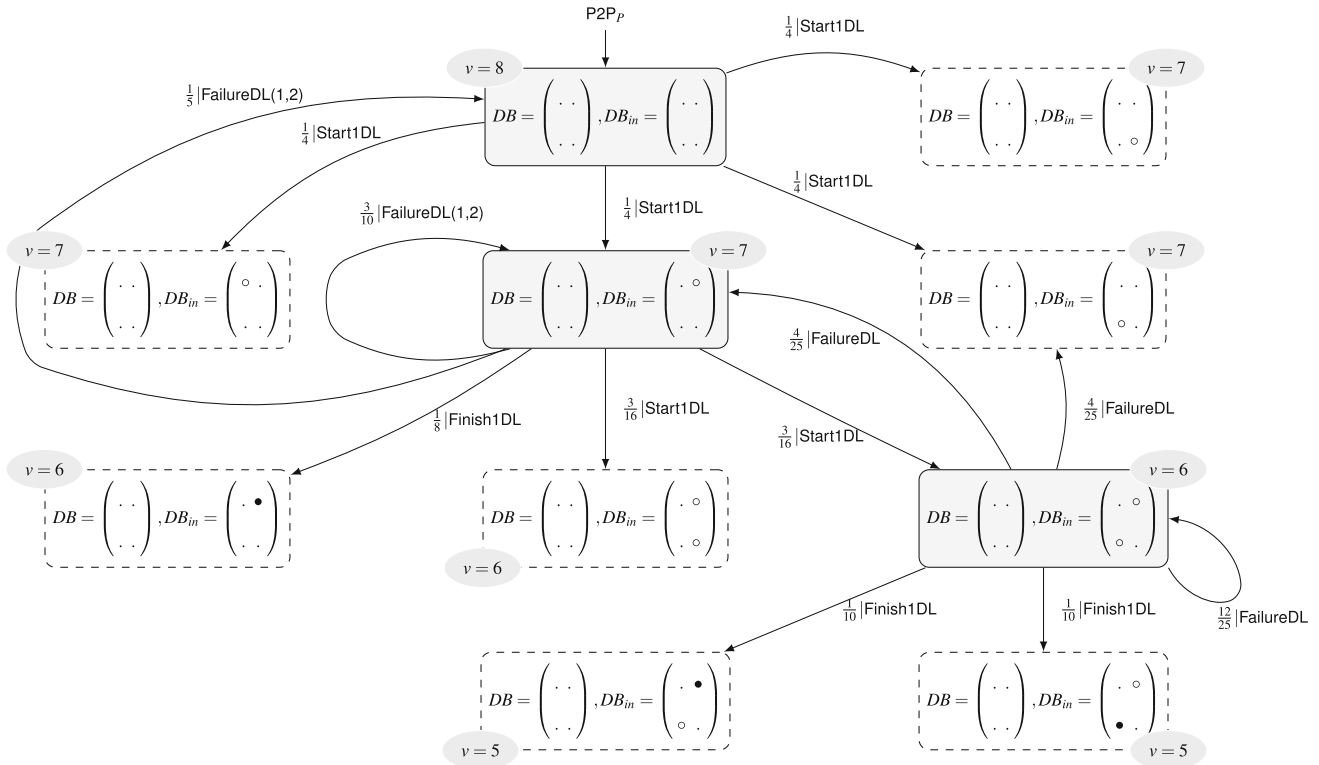


Fig. 15 Extract of the DTMC of the simple P2P protocol, with N=2 and K=2

Recall that proving the convergence of a set of new standard events boils down to proving in all configurations of the system where convergent events can be executed that each convergent event must decrease a given variant. On the other hand, we have shown that a set of probabilistic convergent events almost-certainly converges whenever, in all states where probabilistic convergent events can be executed, at least one of these events must decrease the variant.

When both standard and probabilistic events need to converge, we therefore propose to mix both approaches by requiring that, in all configurations where convergent events can be executed, at least one of the enabled probabilistic convergent events (if any) must decrease the variant with positive probability and all enabled standard convergent events (if any) must decrease it. Therefore, this boils down to proving standard convergence POs for standard convergent events and probabilistic almost-certain convergent POs for probabilistic convergent events. The only modification is that *all* convergent events (standard and probabilistic) need to respect the bound on the variant introduced in the probabilistic setting. We therefore need a new PO for standard convergent events: **Bounded variant for standard events** Standard convergent events can only be enabled when the variant is less or equal to  $\mathbf{U}$ .

$$\boxed{I(\bar{v}) \wedge G_i(\bar{r}, \bar{v}) \vdash V(\bar{v}) \leq \mathbf{U} \quad (\text{event/ndpBOUND})}$$

**Proposition 4** Let  $M = (\bar{v}, I(\bar{v}), V(\bar{v}), \text{Init}, \text{MEvts})$  be a fully mixed Event-B model where  $\text{MEvts} = \text{MEvts}_{pc} \cup \text{MEvts}_{pnc} \cup \text{MEvts}_{ndc} \cup \text{MEvts}_{ndnc}$  is the partition of the set of events into probabilistic convergent events  $\text{MEvts}_{pc}$ , probabilistic non-convergent events  $\text{MEvts}_{pnc}$ , standard convergent events  $\text{MEvts}_{ndc}$ , and standard non-convergent events  $\text{MEvts}_{ndnc}$ .

If all events from  $\text{MEvts}_{pc} \cup \text{MEvts}_{ndc}$  satisfy the PO (event/var/NAT) presented in Sect. 2.3, all events from  $\text{MEvts}_{ndc}$  satisfy the POs (event/var) (from Sect. 2.3) and (event/ndpbound) presented above, and all events from  $\text{MEvts}_{pc}$  satisfy the POs (event/wght/BOUND), (event/pBOUND) and (event/var/pNAT) presented in Sect. 7.1, then the set of events  $\text{MEvts}_{pc} \cup \text{MEvts}_{ndc}$  almost-certainly converges in  $M$ , i.e. converges with probability 1 in the worst case.

*Proof-sketch* Again, the proof of Theorem 1 can be easily adapted to this setting. One then needs to fix an arbitrary scheduler and show that, under this scheduler, the probability of eventually taking a non-convergent event is 1. Since all non-deterministic choices lead to either a convergent standard event (which decreases the variant with probability 1), a non-convergent event (which ends this proof) or the combined probabilistic transition (which either decreases the variant with a positive proba-

bility or is non-convergent itself), the conclusion is easily obtained.

### 8 Conclusion and future work

In this paper, we have presented an extension to the Event-B formalism that allows describing models with probabilistic aspects. We have focused on two types of models: *fully probabilistic* models, where all non-deterministic choices present in standard Event-B models are replaced with probabilistic choices; and *mixed* Event-B models that allow expressing both non-deterministic and probabilistic choices in the same model. We have provided proof obligations for the consistency of both types of models and expressed their operational semantics in terms of probabilistic transition systems. Moreover, we have also explained how the addition of probabilistic information can be done either as a standalone artefact (probabilisation of a standard model) or as a part of the design process that can be interleaved with standard refinement steps. In particular, we have focused on the addition of new probabilistic events in standard/mixed/probabilistic models and developed sufficient conditions in terms of proof obligations in order to show that a given set of probabilistic events is almost-certainly convergent, which is a required property in this context in standard Event-B.

Although we have considered the addition of new probabilistic events in a standard/mixed/probabilistic model, a complete counterpart to standard refinement for the probabilistic setting still eludes us. The problem mainly lies in two operations that are allowed in standard Event-B refinement: the split and merge operations.

- The split operation allows, in one refinement step, to transform one abstract event into multiple concrete events, while allowing the guards of these concrete events to be more restrictive than the guard of the abstract event. In the probabilistic setting, the problem mainly lies in the repartition of the weights of concrete events w.r.t the weight of the abstract event depending on which of the guards are satisfied. We have not found yet a satisfying solution that does not restrict in a too strict way the original split operation. Indeed, a simple but restrictive solution is to impose that the guards of the concrete events must be identical to the guard of the original abstract event. In this case, we only have to impose that the sum of the weights of the concrete events is equal to the weight of the abstract event.
- The merge operation, on the contrary, allows for a single concrete event to refine several abstract events at once. The same problem regarding the repartition of the weights of the original events arises in this setting, which has prevented us from finding a satisfying solution yet.

In the future, we plan on pursuing our work on probabilistic refinement, which would allow us to propose a modelling formalism that supports all the potential design strategies presented in Fig. 1 in the introduction. At the same time, we are pursuing our investigation of probabilistic properties and how to verify them using proof-based techniques. In the spirit of almost-certain convergence, we plan on providing a set of typical properties that can be easily discharged in probabilistic Event-B using predetermined proof obligations.

*Probabilistic plugin for Rodin* We have started the development of a probabilistic plugin for the Rodin platform (see Fig. 16). The Rodin tool [2] is an Eclipse-based IDE for designing models in Event-B. It allows the creation of Event-B models, the automatic generation of POs, and it incorporates some provers for discharging the necessary POs. Rodin is based on a set of plugins that facilitate its extension to support new functionalities.

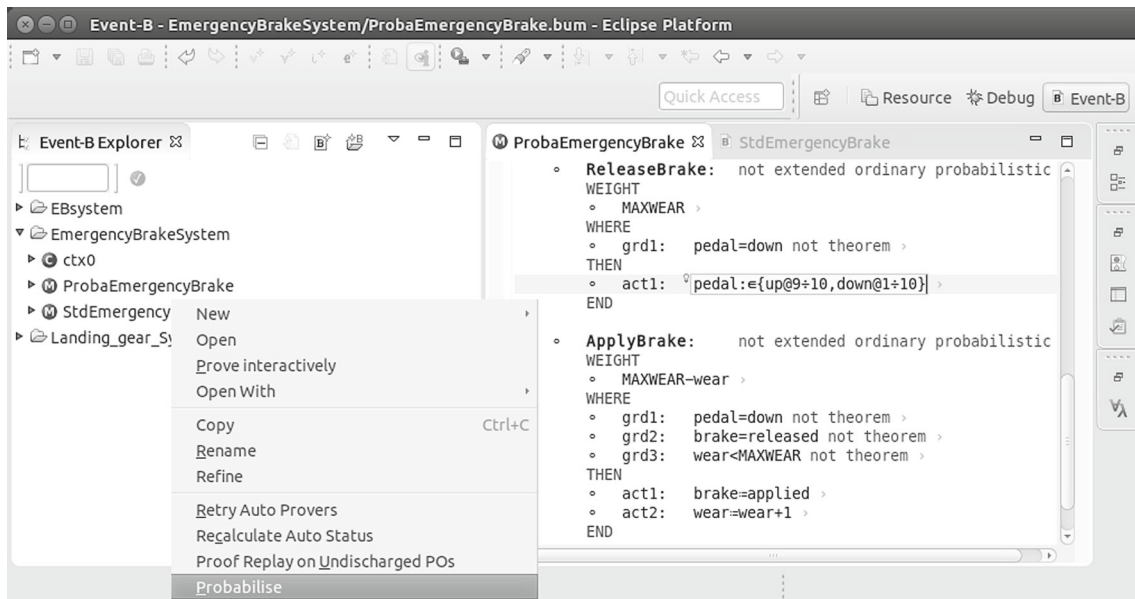
Our plugin is still under development, but it already supports the specification of fully probabilistic Event-B models and the generation of some dedicated POs presented in this paper.

The plugin also allows the *probabilisation* of a non-deterministic Event-B model: it automatically generates the corresponding probabilistic Event-B model. Once this latter is generated, the developer must complete the weight of each probabilistic event and probability values for each quantitative probabilistic assignment.

The plugin features are listed in Table 1 where ✓ denotes the supported functionalities and ~ the functionalities that

**Table 1** Plugin features

Writing probabilistic assignments	✓
Annotating events with weights	✓
Generating new PO (event/assign/pWD1)	✓
Generating new PO (event/assign/pWD2)	✓
Generating new PO (event/assign/pFIS)	~
Generating new PO (event/param/pWD)	~
Generating new PO (event/WGHT/NAT)	✓
Updating PO (event/pINV)	✓
Updating PO (model/pDLF)	✓
Reusing PO (event/pBOUND)	✓
Reusing PO (event/pBOUND1)	~
Reusing PO (event/var/pNAT)	✓
Reusing PO (event/pVAR)	✓
Integrating probabilisation process	✓
Checking condition (event/param/proba)	✓
Checking condition (model/assign/proba)	✓
Checking condition (event/assign/proba)	✓
Generating Probabilistic Event-B models	✓



**Fig. 16** Probabilistic plugin to the Rodin platform

are currently under development. As explained in Sect. 4.2.2, the modifications on the standard PO (event/pINV) consists in strengthening the guard by a predicate, indicating that the weight of the event needs to be strictly greater than 0. In order to implement these modifications in our plugin, one solution is to add the predicate  $W(\bar{v}) > 0$  to the event guard  $G(\bar{r}, \bar{v})$  of each event. We therefore obtain a new guard  $G'(\bar{r}, \bar{v}) \triangleq G(\bar{r}, \bar{v}) \wedge W(\bar{v}) > 0$ , which is used in order to generate and discharge the standard PO (event/INV).

In the future, we plan on pursuing our work on this plugin in order to integrate all the features presented in this paper. In particular, we will reuse works from [40], where another probabilistic plugin for Rodin is introduced in a slightly different context, in order to generate and discharge our new POs for almost-certain convergence.

## References

- Abrial, J.-R.: Modeling in Event-B: System and Software Engineering. Cambridge University Press, Cambridge (2010)
- Abrial, J.-R., Butler, M., Hallerstede, S., Hoang, T.S., Mehta, F., Voisin, L.: Rodin: an open toolset for modelling and reasoning in event-b. *Int. J. Softw. Tools Technol. Transf.* **12**(6), 447–466 (2010)
- Abrial, J.-R., Cansell, D., Méry, D.: A mechanically proved and incremental development of ieee 1394 tree identify protocol. *Form. Aspects Comput.* **14**(3), 215–227 (2003)
- Aouadhi, M.A., Delahaye, B., Lanoix, A.: Moving from event-b to probabilistic event-b. In: Proceedings of the 32nd Annual ACM Symposium on Applied Computing. ACM (2017, to appear)
- Audebaud, P., Paulin-Mohring, C.: Proofs of randomized algorithms in coq. *Sci. Comput. Program.* **74**(8), 568–589 (2009)
- Back, R.-J.: Refinement calculus, part ii: Parallel and reactive programs. In: Stepwise Refinement of Distributed Systems Models, Formalisms, Correctness, pp. 67–93. Springer (1990)
- Back, R.J., von Wright, J.: Refinement calculus, part i: Sequential nondeterministic programs. In: Stepwise Refinement of Distributed Systems Models, Formalisms, Correctness, pp. 42–66. Springer (1990)
- Baier, C., Katoen, J.-P., et al.: Principles of Model Checking, pp. 2620–2649. MIT Press, Cambridge (2008)
- Barthe, G., Fournet, C., Grégoire, B., Strub, P.-Y., Swamy, N., Zanella-Béguélin, S.: Probabilistic relational verification for cryptographic implementations. In: ACM SIGPLAN Notices, vol. 49, pp. 193–205. ACM (2014)
- Bert, D., Cave, F.: Construction of finite labelled transition systems from b abstract systems. In: Integrated Formal Methods, LNCS, vol. 1945, pp. 235–254. Springer (2000)
- Bianco, A., De Alfaro, L.: Model checking of probabilistic and nondeterministic systems. In: International Conference on Foundations of Software Technology and Theoretical Computer Science, pp. 499–513. Springer (1995)
- BitTorrent description. <http://www.bittorrent.com/lang/fr/>
- Butler, M., Maamria, I.: Practical theory extension in event-b. In: Theories of Programming and Formal Methods, LNCS, vol. 8051, pp. 67–81 (2013)
- Chu, W.W., Sit, C.-M.: Estimating task response time with contentions for real-time distributed systems. In: Real-Time Systems Symposium, 1988., Proceedings., pp. 272–281. IEEE (1988)
- De Roeber, W.-P., Engelhardt, K., Buth, K.-H.: Data Refinement: Model-Oriented Proof Methods and Their Comparison, vol. 47. Cambridge University Press, Cambridge (1998)
- Dehnert, C., Gebler, D., Volpato, M., Jansen, D.N.: On Abstraction of Probabilistic Systems, pp. 87–116. Springer Berlin Heidelberg, Berlin (2014)
- Goldreich, O.: Probabilistic proof systems. In: Modern Cryptography, Probabilistic Proofs and Pseudorandomness, pp. 39–72. Springer (1999)
- Haghighi, H., Afshar, M.: A z-based formalism to specify Markov chains. *Comput. Sci. Eng.* **2**(3), 24–31 (2012)
- Hallerstede, S., Hoang, T.S.: Qualitative probabilistic modelling in event-b. In: Integrated Formal Methods, pp. 293–312. Springer (2007)

20. He, J., Hoare, C., Sanders, J.W.: Data refinement refined resume. In: ESOP 86, pp. 187–196. Springer (1986)
21. Hoang, T.S.: The development of a probabilistic B-method and a supporting toolkit. PhD thesis, The University of New South Wales (2005)
22. Hoang, T.S.: Reasoning about almost-certain convergence properties using event-b. *Sci. Comput. Program.* **81**, 108–121 (2014)
23. Hurd, J.: Formal verification of probabilistic algorithms. PhD thesis, University of Cambridge, Computer Laboratory (2003)
24. Hurd, J., McIver, A., Morgan, C.: Probabilistic guarded commands mechanized in hol. *Electron. Notes Theor. Comput. Sci.* **112**, 95–111 (2005)
25. Jonsson, B., Larsen, K.G.: Specification and refinement of probabilistic processes. In: Proceedings of Sixth Annual IEEE Symposium on Logic in Computer Science, 1991. LICS'91, pp. 266–277. IEEE (1991)
26. Katoen, J.-P.: Abstraction of Probabilistic Systems, pp. 1–3. Springer Berlin Heidelberg, Berlin (2007)
27. Mayr, R., Henda, N.B., Abdulla, P.A.: Decisive Markov chains. *Log. Methods Comput. Sci.* **3**, 7 (2007)
28. McIver, A., Morgan, C.C.: Abstraction, Refinement and Proof for Probabilistic Systems. Springer, Berlin (2006)
29. Morgan, C., Hoang, T.S., Abrial, J.-R.: The challenge of probabilistic event b—extended abstract. In: ZB 2005: Formal Specification and Development in Z and B, pp. 162–171. Springer (2005)
30. Motwani, R., Raghavan, P.: Randomized Algorithms. Chapman & Hall/CRC, Boca Raton (2010)
31. Prism modelchecker description. <http://www.prismmodelchecker.org/>
32. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley, New York (2014)
33. Sere, K., Troubitsyna, E.: Probabilities in action systems. In: Proceedings of the 8th Nordic Workshop on Programming Theory, pp. 373–387 (1996)
34. Stoelinga, M.: An introduction to probabilistic automata. *Bull. EATCS* **78**(176–198), 2 (2002)
35. Tarasyuk, A., Troubitsyna, E., Laibinis, L.: Reliability assessment in event-b development. In: NODES 09, p. 11 (2009)
36. Tarasyuk, A., Troubitsyna, E., Laibinis, L.: Towards probabilistic modelling in event-b. In: Integrated Formal Methods, pp. 275–289. Springer (2010)
37. Tarasyuk, A., Troubitsyna, E., Laibinis, L.: Integrating stochastic reasoning into event-b development. *Form. Aspects Comput.* **27**(1), 53–77 (2015)
38. Trivedi, K.S., Ramani, S., Fricks, R.: Recent advances in modeling response-time distributions in real-time systems. *Proc. IEEE* **91**(7), 1023–1037 (2003)
39. Villemeur, A.: Reliability, Availability, Maintainability and Safety Assessment, Assessment, Hardware, Software and Human Factors, vol. 2. Wiley, New York (1992)
40. Yilmaz, E.: Tool support for qualitative reasoning in Event-B. PhD thesis, Master Thesis ETH Zürich, 2010 (2010)



**Mohamed Amine Aouadhi** is a Ph.D. student in Computer Science at the Université de Nantes and carries out his research activities in LS2N. He is working under the direction of Claude Jard and the supervision of Benoît Delahaye and Arnaud Lanoix. He holds a computer engineer diploma from the National School of Computer Sciences of Tunis (Tunisia).



**Benoît Delahaye** received his Ph.D. from the Université de Rennes in 2010. He is associate Professor since 2013 in the Computer Science Department at the Université de Nantes and carries out his research activities in LS2N. His main research interests concern formal abstraction and verification techniques of complex quantitative systems and in particular probabilistic systems.



**Arnaud Lanoix** received his Ph.D. from the University of Franche-Comté in 2005. After 3 years of postdoctorate at the LORIA (University of Lorraine), he is associate Professor since 2008 at the Université de Nantes and carries out his research activities in LS2N. His main research topics concern the use of formal methods and verification techniques, particularly in the context of component-based systems.





# On the Power of Uncertainties in Microbial System Modeling: No Need To Hide Them Anymore

 Benoit Delahaye,<sup>a</sup>  Damien Eveillard,<sup>a</sup> Nicholas Bouskill<sup>b</sup>

LS2N, UMR 6004, CNRS, Université de Nantes, ECN, IMTA, Nantes, France<sup>a</sup>; Lawrence Berkeley National Laboratory, Berkeley, California, USA<sup>b</sup>

**ABSTRACT** For decades, microbiologists have considered uncertainties as an undesired side effect of experimental protocols. As a consequence, standard microbial system modeling strives to hide uncertainties for the sake of deterministic understanding. However, recent studies have highlighted greater experimental variability than expected and emphasized uncertainties not as a weakness but as a necessary feature of complex microbial systems. We therefore advocate that biological uncertainties need to be considered foundational facets that must be incorporated in models. Not only will understanding these uncertainties improve our understanding and identification of microbial traits, it will also provide fundamental insights on microbial systems as a whole. Taking into account uncertainties within microbial models calls for new validation techniques. Formal verification already overcomes this shortcoming by proposing modeling frameworks and validation techniques dedicated to probabilistic models. However, further work remains to extract the full potential of such techniques in the context of microbial models. Herein, we demonstrate how statistical model checking can enhance the development of microbial models by building confidence in the estimation of critical parameters and through improved sensitivity analyses.

**KEYWORDS** modeling, simulation, uncertainty

Since the work of Monod (1), simple biological modeling has been prominent in microbiology. Because of their experimental tractability and purported simplicity, microbial experimental systems have fostered the rise of several cross-scale modeling approaches from the gene to the population level, which have been extended to test ecoevolutionary hypotheses. These modeling approaches proposed and addressed foundational hypotheses that developed into new biological paradigms such as growth rate or identification of functional units. The first microbial models were driven by reductionist assumptions (e.g., intracellular quota combined with kinetics mimicking biochemistry rules) yet demonstrated remarkable predictive power for portraying the growth of microbes in simple systems such as chemostats (1). Similar quota assumptions were used to model phytoplankton physiology (2) and later for modeling simplified global ocean ecosystems (3).

Reductionist modeling approaches have generally been parameterized from data gleaned from laborious bench experiments. However, contemporary next-generation sequencing (NGS) approaches provide unprecedented characterization of the diversity of microbial communities, yet because feedbacks between biotic and abiotic systems are inherently nonlinear and complex, mathematical models of microbial guilds interacting with their environment are required. Current models have been developed along the lines of systems biology approaches (4, 5) or trait-based models (6). Furthermore, NGS provides a large amount of data that represent one experiment alongside

**Published** 5 December 2017


**Citation** Delahaye B, Eveillard D, Bouskill N. 2017. On the power of uncertainties in microbial system modeling: no need to hide them anymore. *mSystems* 2:e00169-17. <https://doi.org/10.1128/mSystems.00169-17>.

**Editor** Nicholas Chia, Mayo Clinic

**Copyright** © 2017 Delahaye et al. This is an open-access article distributed under the terms of the [Creative Commons Attribution 4.0 International license](https://creativecommons.org/licenses/by/4.0/).

Address correspondence to Benoit Delahaye, [benoit.delahaye@univ-nantes.fr](mailto:benoit.delahaye@univ-nantes.fr).

B.D. and D.E. contributed equally to this work.

 On the power of uncertainties in microbial systems modeling: No need to hide them anymore

its associated uncertainties. It is also worth noting that increasing the data set provides a concomitant increase in the number of uncertainties that must be considered.

Within microbial models, these uncertainties can be accounted for by machine learning techniques (see Libbrecht and Noble [7] for a review) that produce automatically predictive (deterministic) models from experimental measurements. Uncertainties are accounted for in the modeling process (e.g., via averaging) but are hidden in the model itself. Moreover, despite the great predictive power of such modelings, the resulting models are not necessarily biologically meaningful. In particular, once parameterized, a model could be overfit to a data set without reflecting emergent properties and precluding or reducing knowledge discovery. Considering that a single microbial system could produce several distinct data sets through several experimental approaches, several different models are built accordingly (8, 9). All corresponding models must then be investigated via automatic learning and verification techniques to take into account their common properties rather than considering each model in isolation (10). Conversely, other probabilistic modelings consider uncertainties but make the parameterization difficult or advocate the use of multiple models that are difficult to validate (11). Despite the aforementioned challenges, uncertainties must be accounted for and integrated into microbial modeling approaches. Such an issue remains a general problem across biology, and even in ecology despite a long tradition of dealing with quantitative uncertainties (12). Herein, as previously done in engineering (13), we advocate that the proper use of dedicated verification techniques could support efforts to capture the complexity of microbial systems within models by promoting a computational convergence on uncertainties rather than simple simulations. Below we present a short overview of current modeling approaches for studying microbial systems, after which we discuss the computational challenges that must be overcome to better take into account uncertainties and verify the resulting models.

### **MICROBIAL STATE-OF-THE ART MODELINGS AND UNCERTAINTIES**

Initial biological modeling efforts were inspired by models of physical systems and formalized using nonlinear ordinary differential equations representing dynamic behaviors of gene activity or molecular concentrations. Interactions within these models are driven by reaction rates associated with particular mechanistic behaviors such as Michaelis-Menten or Hill functions. The need to use formal methods to analyze such models has been discussed extensively in the last 15 years (see De Jong [14] for a state-of-the-art review or Fisher and Henzinger [15]). However, their application mainly results in a discretization of all interactions, such that the model becomes qualitative. An advantage in this context is that the resulting qualitative models are computationally scalable and do not need to incorporate an extensive number of parameters that are mostly out of experimental reach, including parameters that show clear sensitivity to experimental conditions (16). Nonetheless, even if such models are sufficient to represent microbial gene regulatory networks, they are generally not sufficient for modeling quantitative microbial behaviors as needed in the context of simulation of microbial populations and communities, and subsequently, biogeochemical processes.

In order to represent complex quantitative microbial community responses to environmental constraints, several studies postulate that the development of genome-scale metabolic models that consider the whole set of metabolic reactions within a microbial strain are necessary (17). In this context, the metabolic network consists of stoichiometric coefficients and mass balance constraints. Herein, the rates are reduced to fluxes that can be estimated via flux balance analysis (FBA) (18), which negates the need for specific kinetic parameterization yet provides quantitative predictions. FBA models are based on constraints, and solving the metabolic network analysis requires the consideration of boundary conditions for all metabolic fluxes that take place within the cell (see Bordbar et al. [19] for a review). In particular, recently, Basler et al. (20) have shown that those boundaries that reflect uncertainties are necessary to better understand intracellular metabolic flux distribution.

In order to simulate microbial communities, one can consider them at equilibrium

by performing extensions of FBA (21, 22) or focus on dynamic quantitative behaviors by representing kinetics of several microbial populations via traits. These trait-based models use nonlinear differential equations and simplified complex dynamic behaviors with a single parameter. However, because of the inherent complexity of the population (23), estimating such a parameter is a difficult task without considering uncertainties and intensive statistical analyses. Despite several attempts to identify those traits from metagenomic experiments (24), estimating trait parameter values and automatically building predictive models from ecosystem experiments remain key challenges for the field.

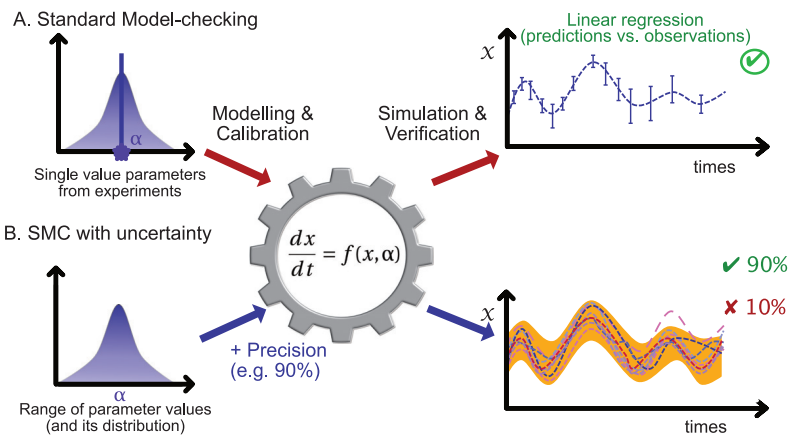
### NEED FOR DEDICATED MODEL-CHECKING TECHNIQUES

Following standard parameter estimation, whereby simulations satisfy the trends and trajectories of experimental data, models are usually tested through sensitivity analyses. Such analyses test the predictive accuracy of models across a wider range of parameter values. However, while such testing is commonly used in practice because of its computational scalability, sensitivity analysis does not provide a formal guarantee of the correctness of the model but rather a synthesis of extensive number of simulations. In several engineering fields, formal verification of models overcomes this shortcoming by performing model checking (25), which provides such guarantees. Unfortunately, because standard model checking was originally designed to study artificial systems, such as computer programs, it does not necessarily scale up to meet the demands of complexity of microbial systems, in particular when dealing with uncertainties. One must therefore combine model checking with sensitivity analyses, as fostered by recent statistical model-checking (SMC) methods (26–28). Like sensitivity analysis, SMC is based on simulations and executes the model several times to converge on a probability for a given property to be satisfied (see Fig. 1 for an illustration). In this case, the number of simulations that must be performed, and satisfied, is not arbitrarily fixed by the modeler but rather precomputed in order to ensure strong (formal) guarantees on the confidence and error levels of the analysis. Because SMC methods are based on simulations, they do not rely directly on the model structure (i.e., number of variables and constraints), only on the ability to run simulations, regardless of the formalization used (i.e., ordinary differential equations [ODEs], constraint-based, Boolean...). As a consequence, they are suitable for realistic modelings, ensuring strong formal guarantees (29) despite the complexity inherent to microbial systems.

### STATISTICAL MODEL CHECKING OF MICROBIAL MODELS

Because uncertainties are central to SMC methods, our hypothesis is that its use will be central for microbial model validation. SMC will indeed build formal confidence (trust) in the validation process while improving standard validation techniques. Standard validation is usually performed using a sufficiently large number of model simulations, but the precise number is left to the acumen of the modeler, which is not a satisfying guarantee with respect to the precision and correctness of the analyses. In contrast, the precision and correctness of SMC methods are formally certified by using statistical results to compute the required number of model simulations. Moreover, SMC methods are tailored for the analysis of models that incorporate uncertainties *per se* and therefore take into account parameter variations as standard characteristics of the models studied. Thus, two potential SMC applications in the context of microbial system validation could be emphasized.

**Model certification rather than sensitivity analysis.** Instead of fixing parameter values to their mean observed values and performing sensitivity analysis of one parameter at a time (Fig. 1A), we propose to embed the uncertainty of the parameter values into the models by assigning each parameter to a probability distribution based on its potential values informed by lab or field experiments (Fig. 1B). Consequently, a trait must then be considered a distribution on a range of values instead of a single value that represents multiple experiments. The distribution of values over a given



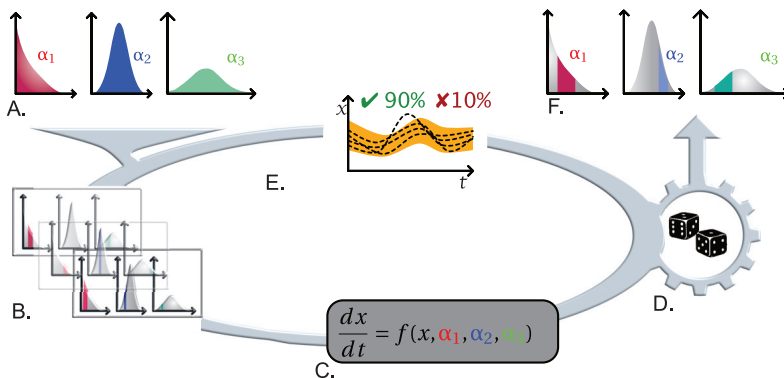
**FIG 1** Illustration of model checking without and with uncertainties. Following a range of experiments, standard data analysis highlights the distribution of values for a given parameter  $\alpha$ . (A) Along a standard model-checking protocol, one assumes a single value for  $\alpha$ , usually the mean. Such a value is then used for model calibration, which allows a simulation. Simulation results are then compared with observations for the sake of model verification (e.g., usually via linear regression between prediction and observations). (B) An example of a model-checking protocol that considers uncertainties *per se*. Instead of considering a single parameter value, one considers a range of values and precision guarantees and performs a range of simulations accordingly (one per color). Altogether, this SMC approach validates the models while taking into account intrinsic uncertainties and guarantees the desired precision (90% here).

range can thus be set as Gaussian, or as another distribution, in order to fit experimental uncertainties. In SMC, model simulations can be performed by picking parameter values within their attached distributions (i.e., by considering the variances of all parameters) and executing standard simulations. Following a simulation, deciding whether one prediction is valid can be done by computing the deviation of predicted values from existing observations. Since several simulations are performed (for several parameter values), SMC outputs a score representing the ratio of valid simulations. Thus, the SMC method performs a generalization of standard sensitivity analyses, not by analyzing the sensitivity of a single average simulation but by analyzing all feasible simulations and proposing general statistics of the whole; i.e., accurate statistical guarantees to perform predictive simulations while taking into account experimental uncertainties. By extension, considering uncertainties also allows us to certify all simulation behaviors (i.e., average of simulations) for the sake of model validation, rather than validating a single behavior (i.e., simulation of an average).

**Model parameter estimation with uncertainties.** Model parameters are often difficult to measure because microbial communities encapsulate an array of trait data leading to wide ranges in parameter values. Moreover, identifying a distribution of parameter values, for instance, trait parameter values for models of microbe-mediated biogeochemical processes remains a difficult task. One could overcome this difficulty by considering a slight modification of the SMC paradigm to decipher the global set of parameters (with uncertainties) that best fit the experimental data (Fig. 2). While the global range of potential parameter values is often known, the aim here is to identify, within this range, the “ideal” parameter values (i.e., those that produce the highest rate of valid simulations). In this context, a partition of the global range of parameter values (alongside the variance) is produced (Fig. 2B), and simulations are performed for each of the obtained “subranges” (Fig. 2C). SMC can then provide a certified score for each of those subranges, which represent an evaluation of their adequacy with respect to experimental data (Fig. 2D and E). When performed on all subranges from the partition, this method allows us to identify the parameter subranges that best fit the experimental data.

**CONCLUSION**

Uncertainties in parameter values are inherent to models built from experimental data. Instead of doing our best to rule out these uncertainties through deterministic



**FIG 2** Statistical model checking (SMC) for model parameter estimation with uncertainties. (A and B) Considering the distribution of parameter values (A), SMC will perform a partition of the global range of parameter values (B). Notably, all parameters will be identified as a whole, instead of identifying each parameter independently from others as in standard parameterization techniques. (C and D) Probabilistic simulations are then performed for each of the “subranges” obtained. (E) Simulations of all models are then compared to experimental data for the sake of adequacy estimation. (F) Iterated several times, this protocol allows us to identify parameter subranges that, considered altogether, best fit the experimental data.

modeling, we advocate that they should be incorporated into the models via the development of dedicated probabilistic modelings. Originally developed for software applications, statistical verification of such models will enhance the accuracy of model validation while also bringing formal evidence of the correctness of the approach. In addition, the future development of dedicated SMC methods will be the necessary steps to estimate parameter values with uncertainties, ensuring the satisfaction of desired properties, which represents the next methodological block for modeling quantitatively microbial systems from genes to ecosystems.

#### ACKNOWLEDGMENT

This work is supported by CNRS PICS 226589, ANR-14-CE28-0002.

#### REFERENCES

1. Monod J. 1949. The growth of bacterial cultures. *Annu Rev Microbiol* 3:371–394. <https://doi.org/10.1146/annurev.mi.03.100149.002103>.
2. Droop M. 1973. Some thoughts on nutrient limitation in algae. *J Phycol* 9:264–272.
3. Follows MJ, Dutkiewicz S. 2011. Modeling diverse communities of marine microbes. *Annu Rev Mar Sci* 3:427–451. <https://doi.org/10.1146/annurev-marine-120709-142848>.
4. Raes J, Bork P. 2008. Molecular eco-systems biology: towards an understanding of community function. *Nat Rev Microbiol* 6:693–699. <https://doi.org/10.1038/nrmicro1935>.
5. Thiele I, Heinken A, Fleming RMT. 2013. A systems biology approach to studying the role of microbes in human health. *Curr Opin Biotechnol* 24:4–12. <https://doi.org/10.1016/j.copbio.2012.10.001>.
6. Litchman E, Klausmeier CA. 2008. Trait-based community ecology of phytoplankton. *Annu Rev Ecol Evol Syst* 39:615–639. <https://doi.org/10.1146/annurev.ecolsys.39.110707.173549>.
7. Libbrecht MW, Noble WS. 2015. Machine learning in genetics and genomics. *Nat Rev Genet* 16:321–332. <https://doi.org/10.1038/nrg3920>.
8. Simão E, Remy E, Thieffry D, Chaouiya C. 2005. Qualitative modelling of regulated metabolic pathways: application to the tryptophan biosynthesis in *E. coli*. *Bioinformatics* 21(Suppl 2):ii190–ii196. <https://doi.org/10.1093/bioinformatics/bti1130>.
9. Salazar-Cavazos E, Santillán M. 2014. Optimal performance of the tryptophan operon of *E. coli*: a stochastic, dynamical, mathematical-modeling approach. *Bull Math Biol* 76:314–334. <https://doi.org/10.1007/s11538-013-9920-8>.
10. Vert J-P. 2010. Reconstruction of biological networks by supervised machine learning approaches, p 165–188. *In* Lodhi HM, Muggleton SH (ed), *Elements of computational systems biology*. John Wiley & Sons, Inc, Hoboken, NY.
11. Friedman N, Linial M, Nachman I, Pe'er D. 2000. Using Bayesian networks to analyze expression data. *J Comput Biol* 7:601–620. <https://doi.org/10.1089/106652700750050961>.
12. Mouquet N, Lagadeuc Y, Devictor V, Doyen L, Duputié A, Eveillard D, Faure D, Garnier E, Gimenez O, Huneman P, Jabot F, Jarne P, Joly D, Julliard R, Kéfi S, Kergoat GJ, Lavorel S, Le Gall L, Meslin L, Morand S, Morin X, Morlon H, Pinay G, Pradel R, Schurr FM, Thuiller W, Loreau M. 2015. Predictive ecology in a changing world. *J Appl Ecol* 52:1293–1310. <https://doi.org/10.1111/1365-2664.12482>.
13. Zuliani P, Platzer A, Clarke EM. 2013. Bayesian statistical model checking with application to Stateflow/Simulink verification. *Formal Methods Syst Des* 43:338–367. <https://doi.org/10.1007/s10703-013-0195-3>.
14. De Jong H. 2002. Modeling and simulation of genetic regulatory systems: a literature review. *J Comput Biol* 9:67–103. <https://doi.org/10.1089/10665270252833208>.
15. Fisher J, Henzinger TA. 2007. Executable cell biology. *Nat Biotechnol* 25:1239–1249. <https://doi.org/10.1038/nbt1356>.
16. Edwards JS, Palsson BO. 2000. Robustness analysis of the *Escherichia coli* metabolic network. *Biotechnol Prog* 16:927–939. <https://doi.org/10.1021/bp0000712>.
17. Thiele I, Palsson BØ. 2010. A protocol for generating a high-quality genome-scale metabolic reconstruction. *Nat Protoc* 5:93–121. <https://doi.org/10.1038/nprot.2009.203>.
18. Orth JD, Thiele I, Palsson BØ. 2010. What is flux balance analysis? *Nat Biotechnol* 28:245–248. <https://doi.org/10.1038/nbt.1614>.
19. Bordbar A, Monk JM, King ZA, Palsson BO. 2014. Constraint-based

- models predict metabolic and associated cellular functions. *Nat Rev Genet* 15:107–120. <https://doi.org/10.1038/nrg3643>.
20. Basler G, Nikoloski Z, Larhlimi A, Barabási AL, Liu YY. 2016. Control of fluxes in metabolic networks. *Genome Res* 26:956–968. <https://doi.org/10.1101/gr.202648.115>.
  21. Biggs MB, Papin JA. 2017. Managing uncertainty in metabolic network structure and improving predictions using Ensemble FBA. *PLoS Comput Biol* 13:e1005413. <https://doi.org/10.1371/journal.pcbi.1005413>.
  22. Budinich M, Bourdon J, Larhlimi A, Eveillard D. 2017. A multi-objective constraint-based approach for modeling genome-scale microbial ecosystems. *PLoS One* 12:e0171744. <https://doi.org/10.1371/journal.pone.0171744>.
  23. Loreau M, Naeem S, Inchausti P, Bengtsson J, Grime JP, Hector A, Hooper DU, Huston MA, Raffaelli D, Schmid B, Tilman D, Wardle DA. 2001. Biodiversity and ecosystem functioning: current knowledge and future challenges. *Science* 294:804–808. <https://doi.org/10.1126/science.1064088>.
  24. Raes J, Letunic I, Yamada T, Jensen LJ, Bork P. 2011. Toward molecular trait-based ecology through integration of biogeochemical, geographical and metagenomic data. *Mol Syst Biol* 7:473. <https://doi.org/10.1038/msb.2011.6>.
  25. Clarke EM, Jr, Grumberg O, Peled DA. 1999. *Model checking*. MIT Press, Cambridge, MA.
  26. Legay A, Delahaye B, Bensalem S. 2010. Statistical model checking: an overview, p 122–135. *In* Barringer H et al. (ed), *Runtime verification*. RV 2010. *Lecture Notes in Computer Science*, vol 6418. Springer, Berlin, Germany. [https://doi.org/10.1007/978-3-642-16612-9\\_11](https://doi.org/10.1007/978-3-642-16612-9_11).
  27. Koh CH, Palaniappan SK, Thiagarajan PS, Wong L. 2012. Improved statistical model checking methods for pathway analysis. *BMC Bioinformatics* 13:S15. <https://doi.org/10.1186/1471-2105-13-S17-S15>.
  28. Younes HLS, Simmons RG. 2006. Statistical probabilistic model checking with a focus on time-bounded properties. *Inform Comput* 204:1368–1409. <https://doi.org/10.1016/j.ic.2006.05.002>.
  29. Kwiatkowska M, Norman G, Parker D. 2008. Using probabilistic model checking in systems biology. *ACM SIGMETRICS Perform Eval Rev* 35:14–21.



## Reachability in parametric Interval Markov Chains using constraints



Anicet Bart<sup>a</sup>, Benoît Delahaye<sup>b,\*</sup>, Paulin Fournier<sup>b</sup>, Didier Lime<sup>c</sup>,  
Éric Monfroy<sup>b</sup>, Charlotte Truchet<sup>b</sup>

<sup>a</sup> Institut Mines-Télécom Atlantique, LS2N, UMR 6004, Nantes, France

<sup>b</sup> Université de Nantes, LS2N, UMR 6004, Nantes, France

<sup>c</sup> École Centrale de Nantes, LS2N, UMR 6004, Nantes, France

### ARTICLE INFO

#### Article history:

Received 24 July 2017

Received in revised form 16 May 2018

Accepted 6 June 2018

Available online 22 June 2018

Communicated by P. Aziz Abdulla

#### Keywords:

Markov chains

Abstract models

Reachability

Parameter synthesis

Constraint programming

### ABSTRACT

Parametric Interval Markov Chains (pIMCs) are a specification formalism that extend Markov Chains (MCs) and Interval Markov Chains (IMCs) by taking into account imprecision in the transition probability values: transitions in pIMCs are labelled with parametric intervals of probabilities. In this work, we study the difference between pIMCs and other Markov Chain abstractions models and investigate three semantics for IMCs: once-and-for-all, interval-Markov-decision-process, and at-every-step. In particular, we prove that all three semantics agree on the maximal/minimal reachability probabilities of a given IMC. We then investigate solutions to several parameter synthesis problems in the context of pIMCs – consistency, qualitative reachability and quantitative reachability – that rely on constraint encodings. Finally, we propose a prototype implementation of our constraint encodings with promising results.

© 2018 Elsevier B.V. All rights reserved.

## 0. Introduction

Discrete time Markov chains (MCs for short) are a standard probabilistic modelling formalism that has been extensively used in the literature to reason about software [1] and real-life systems [2]. However, when modelling real-life systems, the exact value of transition probabilities may not be known precisely. Several formalisms abstracting MCs have therefore been developed. Parametric Markov chains [3] (pMCs for short) extend MCs by allowing parameters to appear in transition probabilities. In this formalism, parameters are variables and transition probabilities may be expressed as polynomials or rational functions over these variables. A given pMC represents a potentially infinite set of MCs, obtained by replacing each parameter by a given value. pMCs are particularly useful to represent systems where dependencies between transition probabilities are required. Indeed, a given parameter may appear in several distinct transition probabilities, which requires that the same value is given to all its occurrences. Interval Markov chains [4] (IMCs for short) extend MCs by allowing precise transition probabilities to be replaced by intervals, but cannot represent dependencies between distinct transitions. IMCs have mainly been studied with three distinct semantics interpretations. Under the *once-and-for-all* semantics, a given

\* Corresponding author.

E-mail address: benoit.delahaye@univ-nantes.fr (B. Delahaye).

IMC represents a potentially infinite number of MCs where transition probabilities are chosen inside the specified intervals while keeping the same underlying graph structure. The *interval-Markov-decision-process* semantics (IMDP for short), such as presented in [5,6], does not require MCs to preserve the underlying graph structure of the original IMC but instead allows a finite “unfolding” of the original graph structure: new probability values inside the intervals can be chosen each time a state is visited. Finally, the *at-every-step* semantics, which was the original semantics given to IMCs in [4], does not preserve the underlying graph structure too while allowing to “aggregate” and “split” states of the original IMC in the manner of probabilistic simulation.

Model-checking algorithms and tools have been developed in the context of pMCs [7–9] and IMCs with the once-and-for-all and the IMDP semantics [10,11]. State of the art tools [7] for pMC verification compute a rational function on the parameters that characterizes the probability of satisfying a given property, and then use external tools such as SMT solving [7] for computing the satisfying parameter values. For these methods to be viable in practice, the allowed number of parameters is quite limited. On the other hand, the model-checking procedure for IMCs presented in [11] is adapted from machine learning and builds successive refinements of the original IMCs that optimize the probability of satisfying the given property. This algorithm converges, but not necessarily to a global optimum. It is worth noticing that existing model checking procedures for pMCs and IMCs strongly rely on their underlying graph structure (*i.e.*, respect the once-and-for-all semantics). However, in [5] the authors perform model checking of  $\omega$ -PCTL formulas on IMCs w.r.t. the IMDP semantics and they show that model checking of LTL formulas can be solved for the IMDP semantics by reduction to the model checking problem of  $\omega$ -PCTL on IMCs with the IMDP semantics. For all that, to the best of our knowledge, no solutions for model-checking IMCs with the at-every-step semantics have been proposed yet.

In this paper, we focus on Parametric interval Markov chains [12] (pIMCs for short), that generalize both IMCs and pMCs by allowing parameters to appear in the endpoints of the intervals specifying transition probabilities, and we provide four main contributions.

First, we formally compare abstraction formalisms for MCs in terms of succinctness: we show in particular in Proposition 2 that pIMCs are (*strictly*) *more succinct* than both pMCs and IMCs when equipped with the right semantics. In other words, everything that can be expressed using pMCs or IMCs can also be expressed using pIMCs while the reverse does not always hold.

Second, we prove in Theorem 1 that the once-and-for-all, the IMDP, and the at-every-step semantics are equivalent w.r.t. reachability properties, both in the IMC and in the pIMC settings. Notably, this result gives theoretical backing to the generalization of existing works on the verification of IMCs to the at-every-step semantics.

Third, we study the parametric verification of fundamental properties at the pIMC level: consistency, qualitative reachability, and quantitative reachability. Given the expressivity of the pIMC formalism, the risk of producing a pIMC specification that is incoherent and therefore does not model any concrete MC is high. Consistency is therefore of paramount importance. In order to provide solutions to consistency and reachability, we model these problems using constraint encodings and use state of the art constraint solvers for solving them. Constraints are first order logic predicates used for modelling and solving combinatorial problems [13]. A problem is described with a list of variables, each in a given domain of possible values, together with a list of constraints over these variables. Such problems are then sent to solvers which decide whether the problem is satisfiable, *i.e.*, if there exists a valuation of the variables satisfying all constraints, and in this case computes a solution. We therefore propose, in Section 3.1 and Proposition 3, constraint encodings for deciding whether a given pIMC is consistent and, if so, synthesizing parameter values ensuring consistency. We then extend, in Section 3.2 and Proposition 4, these encodings to qualitative reachability, *i.e.*, ensuring that given state labels are reachable in *all* (resp. *none*) of the MCs modelled by a given pIMC. Finally, in Section 4.2 and Theorem 2, we focus on the quantitative reachability problem, *i.e.*, synthesizing parameter values such that the probability of reaching given state labels satisfies fixed bounds in *at least one* (resp. *all*) MCs modelled by a given pIMC. While consistency and qualitative reachability for pIMCs have already been studied in [12], the constraint encodings we propose are significantly smaller (linear instead of exponential). To the best of our knowledge, our results provide the first solution to the quantitative reachability problem for pIMCs.

Our last contribution, presented in Section 5, is the implementation of all our verification algorithms in a prototype tool that generates the required constraint encodings and can be plugged to any SMT solver for their resolution.

This paper is an extended version of [14]. Firstly, [14] only considers the once-and-for-all and the at-every-step semantics. In this paper we also consider the IMDP semantics. Thus, we extend all results from [14] that only held for the once-and-for-all and the at-every-step semantics to the IMDP semantics. Secondly, [14] answers quantitative reachability properties over pIMCs and concludes with the perspective of verifying PCTL\* formulas over pIMCs. In this extended version, we show that PCTL\* properties are not equivalent w.r.t. to the different semantics. Thus, the proposed verification processes in [14] used for verifying qualitative and quantitative reachability properties over pIMCs cannot be trivially extended to the verification of PCTL\* properties over pIMCs. Finally, we extend the experimental evaluation proposed in [14]. In the context of linear parametric pIMC transitions, we propose in this extended version MILP [15] and SMT [16] implementations of the qualitative encodings, and SMT implementations of the quantitative encodings. We show that in the qualitative case the MILP formulation offers an exponential gain in term of resolution time compared to the SMT formulation. Also, experiments show that we are able to handle pIMCs with more than 15 000 states and transitions in the qualitative context, and more than 250 states and transitions in the quantitative context.



## 1. Background

In this section we introduce notions and notations that will be used throughout this paper. Given a finite set of variables  $X = \{x_1, \dots, x_k\}$  we call *domain* a set of values associated to a variable in  $X$ . We write  $D_x$  for the domain associated to the variable  $x \in X$  and  $D_X$  for the set of domains associated to the variables in  $X$ . A *valuation*  $v$  over  $X$  is a set  $v = \{(x, d) \mid x \in X, d \in D_x\}$  of elementary valuations  $(x, d)$  where for each  $x \in X$  there exists a unique pair of the form  $(x, d)$  in  $v$ . When clear from the context, we write  $v(x) = d$  for the value given to variable  $x$  according to valuation  $v$ . A *rational function*  $f$  over  $X$  is a division of two (multivariate) polynomials  $g_1$  and  $g_2$  over  $X$  with rational coefficients, i.e.,  $f = g_1/g_2$ . We write  $\mathbb{Q}$  for the set of rational numbers and  $\mathbb{Q}_X$  for the set of rational functions over  $X$ . The evaluation  $v(g)$  of a polynomial  $g$  under the valuation  $v$  replaces each variable  $x \in X$  by its value  $v(x)$ .

Given a finite set of states  $S$ , we write  $\text{Dist}(S)$  for the set of *probability distributions* over  $S$ , i.e., the set of functions  $\mu: S \rightarrow [0, 1]$  such that  $\sum_{s \in S} \mu(s) = 1$ . We write  $\mathbb{I}$  for the set containing all interval subsets of  $[0, 1]$ . In the following, we consider a universal set of symbols  $A$  that we use for labelling the states of our structures. We call these symbols *atomic propositions*. We will use Latin alphabet in state context and Greek alphabet in atomic proposition context.

**Constraints.** Let  $X$  be a finite set of variables. An *atomic constraint* over  $X$  is a Boolean expression of the form  $f(X) \bowtie g(X)$ , with  $\bowtie \in \{\leq, \geq, <, >, =\}$  and  $f$  and  $g$  two functions over variables in  $X$ . An atomic constraint is *linear* if the functions  $f$  and  $g$  are linear. A *constraint* over  $X$  is a Boolean combination of atomic constraints over  $X$ . *Constraints* are therefore first order logic predicates over a given set of variables. Informally, a *Constraint Satisfaction Problem* (CSP) consists of a set of variables associated to given domains and subject to constraints. A CSP is *satisfiable* if there exists a valuation of the variables that satisfies all constraints. Checking satisfiability of constraint problems is difficult in general, as the space of all possible valuations has a size exponential in the number of variables.

**Definition 1.1** (*Constraint Satisfaction Problem*). A *Constraint Satisfaction Problem* (CSP) is a tuple  $\Omega = (X, D, C)$  where  $X$  is a finite set of variables,  $D = D_X$  is the set of all domains associated to the variables from  $X$ , and  $C$  is a set of constraints over  $X$ .

We say that a valuation over  $X$  satisfies  $\Omega$  if and only if it satisfies all constraints in  $C$ . We write  $v(C)$  for the satisfaction result of the valuation of the constraints  $C$  according to  $v$  (i.e., true or false). In the following we call CSP *encoding* a scheme for formulating a given problem into a CSP. The size of a CSP corresponds to the number of variables and atomic constraints appearing in the problem. Note that, in constraint programming, having less variables or less constraints during the encoding does not necessarily imply faster solving time of the problems.

### Discrete Time Markov Chains.

**Definition 1.2** (*Discrete Time Markov Chain*). A Discrete Time Markov Chain (MC for short) is a tuple  $\mathcal{M} = (S, s_0, p, V)$ , where  $S$  is a finite set of states containing the initial state  $s_0$ ,  $V: S \rightarrow 2^A$  is a labelling function, and  $p: S \times S \rightarrow [0, 1]$  is a probabilistic transition function such that for all  $s \in S$ ,  $p(s, \cdot) \in \text{Dist}(S)$ .

The labelling function  $V$  of a MC  $\mathcal{M} = (S, s_0, p, V)$  associates to each state  $s \in S$  a set of atomic propositions. Given a state  $s \in S$ ,  $V(s)$  is called the *label* of  $s$ . We write  $\text{MC}$  for the set containing all discrete time Markov chains.

A Markov Chain can be represented as a directed graph where the nodes correspond to the states of the MC and the edges are labelled with the probabilities given by the transition function of the MC. In this representation, a missing transition between two states represents a transition probability of zero. As usual, given a MC  $\mathcal{M}$ , we call a *path* of  $\mathcal{M}$  a sequence of states obtained from executing  $\mathcal{M}$ , i.e., a sequence  $\omega = s_1, s_2, \dots$  such that the probability of taking the transition from  $s_i$  to  $s_{i+1}$  is strictly positive,  $p(s_i, s_{i+1}) > 0$ , for all  $i$ . A path  $\omega$  is finite if and only if it belongs to  $S^*$ , i.e., it represents a finite sequence of transitions from  $\mathcal{M}$ .

**Example 1.** Fig. 1 illustrates the Markov chain  $\mathcal{M}_1 = (S, s_0, p, V) \in \text{MC}$  where the set of states  $S$  is given by  $\{s_0, s_1, s_2, s_3, s_4\}$ , the atomic propositions are restricted to  $\{\alpha, \beta\}$ , the initial state is  $s_0$ , and the labelling function  $V$  corresponds to  $\{(s_0, \emptyset), (s_1, \{\alpha\}), (s_2, \{\beta\}), (s_3, \{\alpha, \beta\}), (s_4, \{\alpha\})\}$ . The sequences of states  $(s_0, s_1, s_3)$ ,  $(s_0, s_2)$ , and  $(s_0, s_2, s_2, s_2)$ , are three (finite) paths from the initial state  $s_0$  to the states  $s_3$  and  $s_2$ , respectively.

In order to prove some of our results, we need a notion of bisimulation between Markov chains. We therefore use the classical notion of probabilistic bisimulation.

**Definition 1.3** (*Probabilistic bisimulation for Markov chains [17]*). Let  $\mathcal{M} = (S, s_0, p, V)$  be a Markov chain. A *probabilistic bisimulation* on  $\mathcal{M}$  is an equivalence relation  $\mathcal{R}$  on  $S$  such that for all states  $(s_1, s_2) \in \mathcal{R}$ , we have

- $V(s_1) = V(s_2)$ , and
- $p(s_1, T) = p(s_2, T)$  for each equivalence class  $T \in S/\mathcal{R}$ .

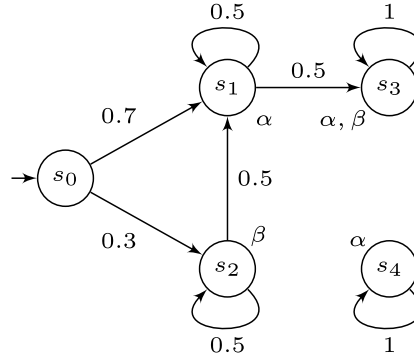


Fig. 1. MC  $\mathcal{M}_1$ .

Given two MCs  $\mathcal{M}_1$  and  $\mathcal{M}_2$ , we say that  $\mathcal{M}_1$  and  $\mathcal{M}_2$  are *bisimilar* if and only if there exists a probabilistic bisimulation  $\mathcal{R}$  on  $\mathcal{M}_1 \cup \mathcal{M}_2$  with  $(s_0^1, s_0^2) \in \mathcal{R}$ .

**Reachability.** A Markov chain  $\mathcal{M}$  defines a unique probability measure  $\mu^{\mathcal{M}}$  over the set of infinite sequences  $s_0, s_1, \dots \in S^\omega$  (see [17] for details). According to this measure, the probability of a *finite* sequence of states  $\omega = s_0, s_1, \dots, s_n$ , written  $\mathbb{P}^{\mathcal{M}}(\omega)$  is the product of the probabilities of the transitions involved in this sequence, i.e.,  $\mathbb{P}^{\mathcal{M}}(\omega) = p(s_0, s_1) \cdot p(s_1, s_2) \cdot \dots \cdot p(s_{n-1}, s_n)$ . Naturally, if  $\omega$  is not a path of  $\mathcal{M}$ , we have  $\mathbb{P}^{\mathcal{M}}(\omega) = 0$ .

Given a MC  $\mathcal{M}$ , the overall probability of reaching a given state  $s$  from the initial state  $s_0$  is called the *reachability probability* and written  $\mathbb{P}_{s_0}^{\mathcal{M}}(\diamond s)$  or  $\mathbb{P}^{\mathcal{M}}(\diamond s)$  when clear from the context. This probability is computed as the sum of the probabilities of all finite paths starting in the initial state and reaching the state  $s$  for the first time. Formally, let  $\text{reach}_{s_0}(s) = \{\omega \in S^* \mid \omega = s_0, \dots, s_n \text{ with } s_n = s \text{ and } s_i \neq s \forall 0 \leq i < n\}$  be the set of such paths. We then define  $\mathbb{P}^{\mathcal{M}}(\diamond s) = \sum_{\omega \in \text{reach}_{s_0}(s)} \mathbb{P}^{\mathcal{M}}(\omega)$  if  $s \neq s_0$  and 1 otherwise. This notation naturally extends to the reachability probability of a state  $s$  from a state  $t$  that is not  $s_0$ , written  $\mathbb{P}_t^{\mathcal{M}}(\diamond s)$  and to the probability of reaching a state label  $\Gamma \subseteq A$  written  $\mathbb{P}_{s_0}^{\mathcal{M}}(\diamond \Gamma)$ . One could also extend it to the reachability probability of an atomic proposition  $\alpha$  by considering all state labels containing  $\alpha$ . In the following, we say that a state  $s$  (resp. a label  $\Gamma \subseteq A$ ) is *reachable* in  $\mathcal{M}$  if and only if the reachability probability of this state (resp. label) from the initial state is strictly positive.

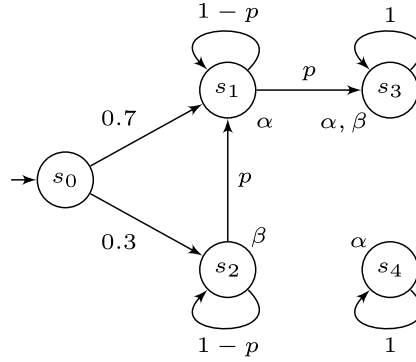
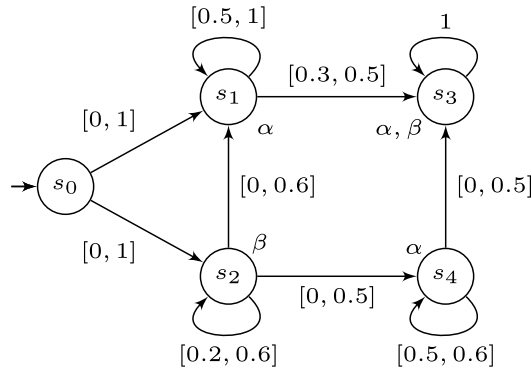
**Example 2** (Example 1 continued). In Fig. 1 the probability of the path  $(s_0, s_2, s_1, s_1, s_3)$  is  $0.3 \cdot 0.5 \cdot 0.5 \cdot 0.5 = 0.0375$  and the probability of reaching the state  $s_1$  from  $s_0$  is  $\mathbb{P}_{s_0}^{\mathcal{M}_1}(\diamond s_1) = p(s_0, s_1) + \sum_{i=0}^{+\infty} p(s_0, s_2)^i \cdot p(s_2, s_1) = p(s_0, s_1) + p(s_0, s_2) \cdot p(s_2, s_1) \cdot (1/(1 - p(s_2, s_2))) = 1$ . Furthermore, the probability of reaching  $\{\beta\}$  corresponds to the probability of reaching the state  $s_2$ , which is 0.3 here.

## 2. Markov chain abstractions

Modelling an application as a Markov Chain requires knowing the exact probability for each possible transition of the system. However, this can be difficult to compute or to measure in the case of a real-life application (e.g., because of precision errors or limited knowledge). In this section, we start with a generic definition of Markov chain abstraction models. Then we recall three abstraction models from the literature, respectively pMC, IMC, and pIMC, and finally we present a comparison of these existing models in terms of succinctness.

**Definition 2.1** (Markov chain abstraction model). A Markov chain abstraction model (an abstraction model for short) is a pair  $(\mathbb{L}, \models)$  where  $\mathbb{L}$  is a nonempty set and  $\models$  is a relation between MC and  $\mathbb{L}$ . Let  $\mathcal{P}$  be in  $\mathbb{L}$  and  $\mathcal{M}$  be in MC. We say that  $\mathcal{M}$  implements  $\mathcal{P}$  if and only if  $(\mathcal{M}, \mathcal{P})$  belongs to  $\models$  (i.e.,  $\mathcal{M} \models \mathcal{P}$ ). When the context is clear, we do not mention the satisfaction relation  $\models$  and only use  $\mathbb{L}$  to refer to the abstraction model  $(\mathbb{L}, \models)$ .

A *Markov chain abstraction model* is a specification theory for MCs. It consists of a set of abstract objects, called *specifications*, each of which representing a (potentially infinite) set of MCs – *implementations* – together with a satisfaction relation defining the link between implementations and specifications. As an example, consider the powerset of MC (i.e., the set containing all possible sets of Markov chains). Clearly,  $(2^{\text{MC}}, \in)$  is a Markov chain abstraction model, which we call the *canonical abstraction model*. This abstraction model has the advantage of representing all possible sets of Markov chains but it also has the disadvantage that some Markov chain abstractions are only representable by an infinite extension representation. Indeed, recall that there exist subsets of  $[0, 1] \subseteq \mathbb{R}$  which cannot be represented in a finite space (e.g., the Cantor set [18]). We now present existing MC abstraction models from the literature.

Fig. 2. pMC  $\mathcal{I}'$ .Fig. 3. IMC  $\mathcal{I}$ .

### 2.1. Existing MC abstraction models

**Parametric Markov Chain** is a MC abstraction model from [3] where a transition can be annotated by a rational function over *parameters*. We write  $\text{pMC}$  for the set containing all parametric Markov chains.

**Definition 2.2 (Parametric Markov Chain).** A Parametric Markov Chain (pMC for short) is a tuple  $\mathcal{I} = (S, s_0, P, V, Y)$  where  $S$ ,  $s_0$ , and  $V$  are defined as for MCs,  $Y$  is a set of variables (parameters) ranging over  $[0, 1]$ , and  $P: S \times S \rightarrow \mathbb{Q}_Y$  associates with each potential transition a parameterized probability.

Let  $\mathcal{M} = (S, s_0, p, V)$  be a MC and  $\mathcal{I} = (S', s'_0, P, V', Y)$  be a pMC. The satisfaction relation  $\models_{\text{p}}$  between MC and pMC is defined by  $\mathcal{M} \models_{\text{p}} \mathcal{I}$  if and only if  $S = S'$ ,  $s_0 = s'_0$ ,  $V = V'$ , and there exists a valuation  $v$  of  $Y$  such that  $p(s, s')$  equals  $v(P(s, s'))$  for all  $s, s'$  in  $S$ .

**Example 3.** Fig. 2 shows a pMC  $\mathcal{I}' = (S, s_0, P, V, Y)$  where  $S$ ,  $s_0$ , and  $V$  are identical to those of the MC  $\mathcal{M}$  from Fig. 1, the set  $Y$  contains only one variable  $p$ , and the parametric transitions in  $P$  are given by the edge labelling (e.g.,  $P(s_0, s_1) = 0.7$ ,  $P(s_1, s_3) = p$ , and  $P(s_2, s_2) = 1 - p$ ). Note that the pMC  $\mathcal{I}'$  is a specification containing the MC  $\mathcal{M}_1$  from Fig. 1.

**Interval Markov Chains** extend MCs by allowing to label transitions with intervals of possible probabilities instead of precise probabilities. We write  $\text{IMC}$  for the set containing all interval Markov chains.

**Definition 2.3 (Interval Markov Chain [4]).** An Interval Markov Chain (IMC for short) is a tuple  $\mathcal{I} = (S, s_0, P, V)$ , where  $S$ ,  $s_0$ , and  $V$  are defined as for MCs, and  $P: S \times S \rightarrow \mathbb{I}$  associates with each potential transition an interval of probabilities.

**Example 4.** Fig. 3 illustrates IMC  $\mathcal{I} = (S, s_0, P, V)$  where  $S$ ,  $s_0$ , and  $V$  are similar to the MC given in Fig. 1. By observing the edge labelling we see that  $P(s_0, s_1) = [0, 1]$ ,  $P(s_1, s_1) = [0.5, 1]$ , and  $P(s_3, s_3) = [1, 1]$ . On the other hand, the intervals of probability for missing transitions are reduced to  $[0, 0]$ , e.g.,  $P(s_0, s_0) = [0, 0]$ ,  $P(s_0, s_3) = [0, 0]$ ,  $P(s_1, s_4) = [0, 0]$ .

In the literature, IMCs have been mainly used with three distinct semantics: *once-and-for-all*, *interval-Markov-decision-process*, and *at-every-step*. These semantics are associated with distinct satisfaction relations which we now introduce.

The *once-and-for-all* IMC semantics [7,19,20] is alike to the semantics for pMC, as introduced above. The associated satisfaction relation  $\models_{\perp}^0$  is defined as follows.

**Definition 2.4** (Once-and-for-all satisfaction). A MC  $\mathcal{M} = (T, t_0, p, V^M)$  satisfies an IMC  $\mathcal{I} = (S, s_0, P, V^I)$  with the once-and-for-all semantics, written  $\mathcal{M} \models_{\perp}^o \mathcal{I}$ , if and only if  $(T, t_0, V^M) = (S, s_0, V^I)$  and for all reachable states  $s$  in  $\mathcal{M}$  and all states  $s' \in S$ ,  $p(s, s') \in P(s, s')$ .

In this sense, we say that MC implementations using the once-and-for-all semantics need to have the same structure as the IMC specification. Remark that this definition only targets *reachable* states in the MC. Indeed, some states of the IMC could have inconsistent outgoing transitions. While these states cannot be “satisfied” in MC implementations, satisfaction should still be possible if they are unreachable.

Next, the *interval-Markov-decision-process* IMC semantics (IMDP for short) [5,6] operates as an “unfolding” of the original IMC by picking each time a state is visited a possibly new probability distribution which respects the intervals of probabilities. Thus, this semantics allows one to produce MCs satisfying IMCs with a different structure. Formally, the associated satisfaction relation  $\models_{\perp}^d$  is defined as follows.

**Definition 2.5** (Interval-Markov-decision-process satisfaction). A MC  $\mathcal{M} = (T, t_0, p, V^M)$  satisfies an IMC  $\mathcal{I} = (S, s_0, P, V^I)$  with the IMDP semantics, written  $\mathcal{M} \models_{\perp}^d \mathcal{I}$ , if and only if there exists a mapping  $\pi$  from  $T$  to  $S$  such that

1.  $\pi(t_0) = s_0$ ,
2.  $V^I(\pi(t)) = V^M(t)$  for all states  $t \in T$ , and
3.  $p(t, t') \in P(\pi(t), \pi(t'))$  for all pairs of states  $t, t'$  in  $T$ , where  $t$  is reachable in  $\mathcal{M}$ .

Thus, we have that  $\models_{\perp}^d$  is more general than  $\models_{\perp}^o$  (i.e., whenever  $\mathcal{M} \models_{\perp}^o \mathcal{I}$  we also have  $\mathcal{M} \models_{\perp}^d \mathcal{I}$  for the identity mapping  $\pi$ ). Note that in [5,6] the authors allow the Markov chains satisfying the IMCs w.r.t.  $\models_{\perp}^d$  to have an infinite state space. In this work we consider Markov chains with a finite state space only, therefore MC implementations with respect to the IMDP semantics consist in a finite unfolding of the specification IMC, which corresponds to a transient IMDP semantics followed by a once-and-for-all semantics.

Finally, the *at-every-step* IMC semantics, first introduced in [4], operates as a simulation relation based on the transition probabilities and state labels, and therefore allows MC implementations to have a different structure than the IMC specification. Compared to the previous semantics, in addition to the unfoldings this one allows to “aggregate” and “split” states from the original IMC. Formally, the associated satisfaction relation  $\models_{\perp}^a$  is defined as follows.

**Definition 2.6** (At-every-step satisfaction). A MC  $\mathcal{M} = (T, t_0, p, V^M)$  satisfies an IMC  $\mathcal{I} = (S, s_0, P, V^I)$  with the at-every-step semantics, written  $\mathcal{M} \models_{\perp}^a \mathcal{I}$  if and only if there exists a relation  $\mathcal{R} \subseteq T \times S$  such that  $(t_0, s_0) \in \mathcal{R}$ , and whenever  $(t, s) \in \mathcal{R}$ , we have

1. the labels of  $s$  and  $t$  correspond:  $V^M(t) = V^I(s)$ ,
2. there exists a correspondence function  $\delta_{(t,s)} : T \rightarrow (S \rightarrow [0, 1])$  such that
  - (a) For all  $t' \in T$  if  $p(t, t') > 0$  then  $\delta_{(t,s)}(t')$  is a distribution on  $S$ ,
  - (b) For all  $s' \in S$ ,  $(\sum_{t' \in T} p(t, t') \cdot \delta_{(t,s)}(t')(s')) \in P(s, s')$ , and
  - (c) For all  $(t', s') \in T \times S$ , if  $\delta_{(t,s)}(t')(s') > 0$ , then  $(t', s') \in \mathcal{R}$ .

Note that reachability of the states from  $\mathcal{M}$  is not required for the at-every-step satisfaction relation. Indeed, in this case, states that are not reachable in  $\mathcal{M}$  can be absent from the satisfaction relation, and therefore do not impact at-every-step satisfaction.

Example 5 illustrates the three IMC semantics and Proposition 1 compares them. We say that an IMC semantics  $\models_1$  is more general than another IMC semantics  $\models_2$  if and only if for all IMCs  $\mathcal{I}$  and for all MCs  $\mathcal{M}$  if  $\mathcal{M} \models_2 \mathcal{I}$  then  $\mathcal{M} \models_1 \mathcal{I}$ . Also,  $\models_1$  is strictly more general than  $\models_2$  if and only if  $\models_1$  is more general than  $\models_2$  and  $\models_2$  is not more general than  $\models_1$ .

**Example 5** (Example 4 continued). Consider the IMC  $\mathcal{I}$  from Fig. 3, the MC  $\mathcal{M}_1$  from Fig. 1, the MC  $\mathcal{M}_2$  from Fig. 4, and the MC  $\mathcal{M}_3$  from Fig. 5. We have that  $\mathcal{M}_1$  satisfies  $\mathcal{I}$  w.r.t.  $\models_{\perp}^o$  and we say that  $\mathcal{M}_1$  has the same structure as  $\mathcal{I}$ . Trivially, we also have that  $\mathcal{M}_1$  satisfies  $\mathcal{I}$  w.r.t.  $\models_{\perp}^d$  and  $\models_{\perp}^a$ .

Regarding  $\mathcal{M}_2$ , note that two probability distributions have been chosen for the state  $s_1$  from  $\mathcal{I}$ . This produces two states  $t_1$  and  $t'_1$  in  $\mathcal{M}_2$  and changes the structure of the graph. Thus,  $\mathcal{M}_2 \not\models_{\perp}^o \mathcal{I}$ . On the other hand, we have that  $\mathcal{M}_2$  satisfies  $\mathcal{I}$  w.r.t.  $\models_{\perp}^d$  with the mapping  $\pi(t_i) = s_i$  for all  $t_i$  and  $\pi(t'_1) = s_1$ . Remark that there is no state  $t_4$  here, but having an unreachable state  $t_4$  and defining  $\pi(t_4) = s_4$  would not prevent satisfaction. Trivially, we also have  $\mathcal{M}_2 \models_{\perp}^a \mathcal{I}$  with the relation  $\mathcal{R}_2 = \{(t, s) \mid \pi(t) = s\}$ .

Finally, in the MC  $\mathcal{M}_3$  with state space  $T_3$  the state  $s_3$  from  $\mathcal{I}$  has been “split” into two states  $t_3$  and  $t'_3$  and the state  $t_{14}$  “aggregates” the states  $s_1$  and  $s_4$  from  $\mathcal{I}$ . The relation  $\mathcal{R}_3 \subseteq T_3 \times S$  containing the pairs  $(t_0, s_0)$ ,  $(t_{14}, s_1)$ ,  $(t_{14}, s_4)$ ,  $(t_2, s_2)$ ,  $(t_3, s_3)$ , and  $(t'_3, s_3)$  is a satisfaction relation between  $\mathcal{M}_3$  and  $\mathcal{I}$  such as defined by  $\models_{\perp}^a$ . For instance, consider the pair  $(t_2, s_2) \in \mathcal{R}$ . For this pair, define the correspondence function  $\delta_{(t_2, s_2)}$  such that  $\delta_{(t_2, s_2)}(t_2)(s_2) = 1$ ,  $\delta_{(t_2, s_2)}(t_{14})(s_1) = 0.5$ ,  $\delta_{(t_2, s_2)}(t_{14})(s_4) = 0.5$ , and  $\delta_{(t_2, s_2)}(t')(s') = 0$  otherwise. We can verify the following:

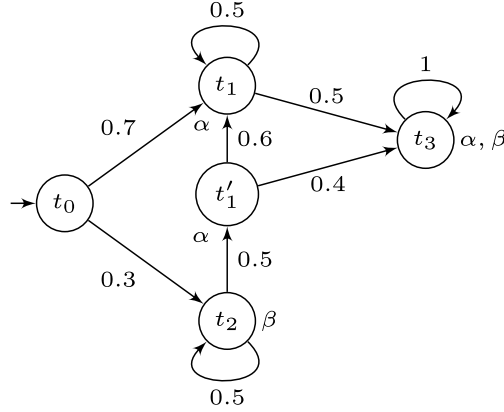


Fig. 4. MC  $\mathcal{M}_2$  satisfying the IMC  $\mathcal{I}$  from Fig. 3 w.r.t.  $\models_{\perp}^d$ .

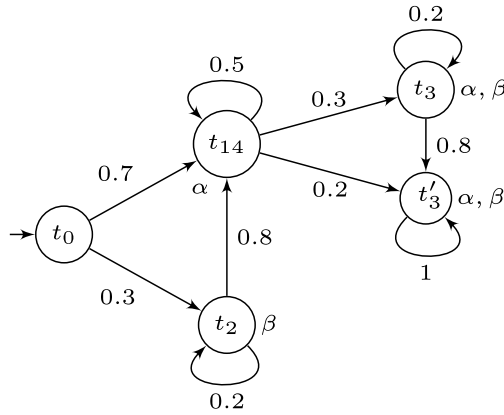


Fig. 5. MC  $\mathcal{M}_3$  satisfying the IMC  $\mathcal{I}$  from Fig. 3 w.r.t.  $\models_{\perp}^a$ .

1. For all  $t'$  such that  $p_3(t_2, t') > 0$ ,  $\delta_{(t_2, s_2)}(t')$  is a distribution on  $S$ . This is the case for  $t_2$  and  $t_{14}$ .
2. For all  $s' \in S$ ,  $(\sum_{t' \in T_3} p_3(t_2, t') \cdot \delta_{(t_2, s_2)}(t')(s')) \in P(s_2, s')$ :

$$[\mathbf{s}_1]: \quad (\sum_{t' \in T_3} p_3(t_2, t') \cdot \delta_{(t_2, s_2)}(t')(s_1)) = 0.8 * 0.5 = 0.4 \in [0, 0.6] = P(s_2, s_1)$$

$$[\mathbf{s}_2]: \quad (\sum_{t' \in T_3} p_3(t_2, t') \cdot \delta_{(t_2, s_2)}(t')(s_2)) = 0.2 * 1 = 0.2 \in [0.2, 0.6] = P(s_2, s_2)$$

$$[\mathbf{s}_4]: \quad (\sum_{t' \in T_3} p_3(t_2, t') \cdot \delta_{(t_2, s_2)}(t')(s_4)) = 0.8 * 0.5 = 0.4 \in [0, 0.5] = P(s_2, s_4)$$

3. For all  $(t', s') \in T_3 \times S$ , if  $\delta_{(t_2, s_2)}(t')(s') > 0$ , then  $(t', s') \in \mathcal{R}_3$ . This is the case for  $(t_2, s_2)$ ,  $(t_{14}, s_1)$ , and  $(t_{14}, s_4)$ .

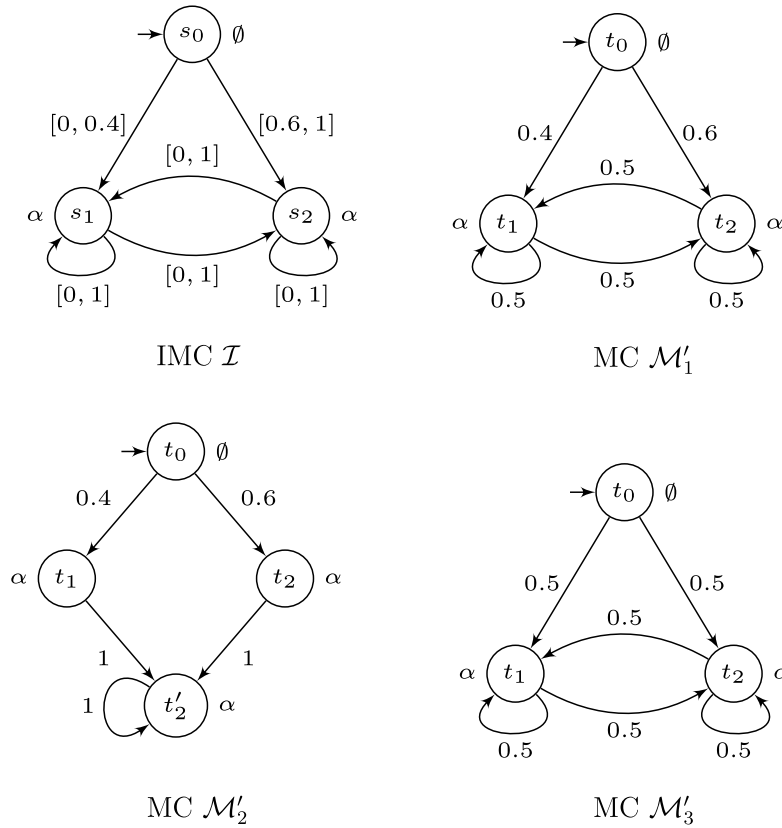
For all other pairs  $(t', s') \in \mathcal{R}_3$ , the correspondence functions can be defined trivially as Dirac distributions where  $\delta_{(t', s')}(t_{14})$  is either a Dirac on  $s_1$ , for  $(t', s') = (t_0, s_0)$  and  $(t_{14}, s_1)$ , or a Dirac on  $s_4$  for  $(t', s') = (t_{14}, s_4)$ .

Thus,  $\mathcal{M}_3 \models_{\perp}^a \mathcal{I}$ . On the other hand,  $\mathcal{M}_3 \not\models_{\perp}^d \mathcal{I}$  since there exist probabilities on transitions that cannot belong to their respective interval of probabilities on the IMC (e.g.,  $p_3(t_2, t_{14}) = 0.8 \notin [0, 0.6] = P(s_2, s_1)$ ).

**Proposition 1.** *The at-every-step satisfaction relation is (strictly) more general than the interval-Markov-decision-process satisfaction relation which is (strictly) more general than the once-and-for-all satisfaction relation.*

**Proof.** Let  $\mathcal{I} = (S, s_0, P, V)$  be an IMC and  $\mathcal{M} = (T, t_0, p, V')$  be a MC. We show that (1)  $\mathcal{M} \models_{\perp}^o \mathcal{I} \Rightarrow \mathcal{M} \models_{\perp}^d \mathcal{I}$ ; (2)  $\mathcal{M} \models_{\perp}^d \mathcal{I} \Rightarrow \mathcal{M} \models_{\perp}^a \mathcal{I}$ ; (3) in general  $\mathcal{M} \models_{\perp}^d \mathcal{I} \not\Rightarrow \mathcal{M} \models_{\perp}^o \mathcal{I}$ ; (4) in general  $\mathcal{M} \models_{\perp}^a \mathcal{I} \not\Rightarrow \mathcal{M} \models_{\perp}^d \mathcal{I}$ . This will prove that  $\models_{\perp}^a$  is strictly more general than  $\models_{\perp}^d$  and that  $\models_{\perp}^d$  is strictly more general than  $\models_{\perp}^o$ . At the same time, note that the following examples also illustrate that even if a Markov chain satisfies an IMC with the same graph representation w.r.t. the  $\models_{\perp}^a$  relation it may not verify the  $\models_{\perp}^o$  relation.

- (1) If  $\mathcal{M} \models_{\perp}^o \mathcal{I}$  then by definition of  $\models_{\perp}^o$  we have that  $T = S$ ,  $t_0 = s_0$ ,  $V(s) = V'(s)$  for all  $s \in S$ , and  $p(s, s') \in P(s, s')$  for all reachable states  $s \in S$  and all states  $s' \in S$ . The mapping  $\pi$  from  $T = S$  to  $S$  being the identity function is such as required by definition of  $\models_{\perp}^d$ :  $\pi(t_0) = t_0 = s_0$ ,  $V'(s) = V(s) = V(\pi(s))$  for all states  $s \in S$ , and  $p(s, s') \in P(\pi(s), \pi(s'))$  since  $P(\pi(s), \pi(s')) = P(s, s')$  and  $p(s, s') \in P(s, s')$  for all reachable states  $s \in S$  and all states  $s' \in S$ . Thus,  $\mathcal{M} \models_{\perp}^d \mathcal{I}$ .
- (2) If  $\mathcal{M} \models_{\perp}^d \mathcal{I}$  then there exists a mapping  $\pi$  from  $T$  to  $S$  such that  $\pi(t_0) = s_0$ ,  $V'(t) = V(\pi(t))$  for all states  $t \in T$ , and  $p(t, t') \in P(\pi(t), \pi(t'))$  for all pairs of states  $t, t' \in T$  where  $t$  is reachable in  $\mathcal{M}$ . The relation  $\mathcal{R} = \{(t, \pi(t)) \mid t \in T\}$



**Fig. 6.** IMC  $\mathcal{I}$ , MCs  $\mathcal{M}'_1$ ,  $\mathcal{M}'_2$ , and  $\mathcal{M}'_3$  such that  $\mathcal{M}'_1 \models_{\pm}^a \mathcal{I}$ ,  $\mathcal{M}'_1 \models_{\pm}^d \mathcal{I}$  and  $\mathcal{M}'_1 \not\models_{\pm}^o \mathcal{I}$ ;  $\mathcal{M}'_2 \models_{\pm}^d \mathcal{I}$  and  $\mathcal{M}'_2 \not\models_{\pm}^o \mathcal{I}$ ;  $\mathcal{M}'_3 \models_{\pm}^a \mathcal{I}$  and  $\mathcal{M}'_3 \not\models_{\pm}^d \mathcal{I}$ ; the graph representation of  $\mathcal{I}$ ,  $\mathcal{M}'_1$ , and  $\mathcal{M}'_3$  are isomorphic.

is such as required by definition of  $\models_{\pm}^a$  (consider for each state in  $T$  the correspondence function  $\delta: T \rightarrow (S \rightarrow [0, 1])$  such that  $\delta(t)(s) = 1$  if  $\pi(t) = s$  and  $\delta(t)(s) = 0$  otherwise). Thus  $\mathcal{M} \models_{\pm}^a \mathcal{I}$ .

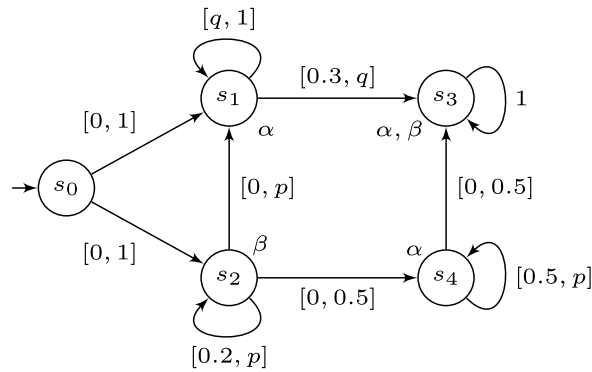
- (3) Consider IMC  $\mathcal{I}$  and MC  $\mathcal{M}'_2$  from Fig. 6. By definition of  $\models_{\pm}^d$  we have that  $\mathcal{M}'_2 \models_{\pm}^d \mathcal{I}$ . Indeed, consider the mapping  $\pi$  such that  $\pi(t_0) = s_0$ ,  $\pi(t_1) = s_1$ ,  $\pi(t_2) = s_2$ , and  $\pi(t'_2) = s_2$ . Let  $p$  be the transition function of  $\mathcal{M}'_2$  and  $P$  be the interval probability transition function of  $\mathcal{I}$ . Clearly, we have that  $p(\pi(t), \pi(t')) \in P(\pi(t), \pi(t'))$ . On the other hand, it is clear that  $\mathcal{M}'_2 \not\models_{\pm}^o \mathcal{I}$  since  $\mathcal{M}'_2$  and  $\mathcal{I}$  do not share the same state space.
- (4) Consider IMC  $\mathcal{I}$  and MC  $\mathcal{M}'_3$  from Fig. 6. By definition of  $\models_{\pm}^a$  we have that  $\mathcal{M}'_3 \models_{\pm}^a \mathcal{I}$ . Indeed, the relation  $\mathcal{R}$  containing  $(t_0, s_0)$ ,  $(t_1, s_1)$ ,  $(t_1, s_2)$ ,  $(t_2, s_1)$ , and  $(t_2, s_2)$  is a satisfaction relation between  $\mathcal{I}$  and  $\mathcal{M}'_3$ . Consider the correspondence function  $\delta$  from  $T$  to  $(S \rightarrow [0, 1])$  such that  $\delta(t_1)(s_1) = 4/5$ ,  $\delta(t_1)(s_2) = 1/5$ ,  $\delta(t_2)(s_2) = 1$ ,  $\delta(t_0)(s_0) = 1$ , and  $\delta(t)(s) = 0$  otherwise. On the other hand, since the outgoing probabilities from state  $t_0$  in  $\mathcal{M}'_3$  do not belong to their respective interval on probabilities in  $\mathcal{I}$ , we have that  $\mathcal{M}'_3 \not\models_{\pm}^d \mathcal{I}$ .  $\square$

**Parametric Interval Markov Chains**, as introduced in [12], abstract IMCs by allowing (combinations of) parameters to be used as interval endpoints in IMCs. Under a given parameter valuation the pIMC yields an IMC as introduced above. pIMCs therefore allow the representation, in a compact way and with a finite structure, of a potentially infinite number of IMCs. Note that one parameter can appear in several transitions at once, requiring the associated transition probabilities to depend on one another. Let  $Y$  be a finite set of parameters and  $v$  be a valuation over  $Y$ . By combining notations used for IMCs and pIMCs the set  $\mathbb{I}(\mathbb{Q}_Y)$  contains all parameterized intervals over  $[0, 1]$ , and for all  $I = [f_1, f_2] \in \mathbb{I}(\mathbb{Q}_Y)$ ,  $v(I)$  denotes the interval  $[v(f_1), v(f_2)]$  if  $0 \leq v(f_1) \leq v(f_2) \leq 1$  and the empty set otherwise.<sup>1</sup> We write pIMC for the set containing all parametric interval Markov chains.

**Definition 2.7** (Parametric Interval Markov Chain [12]). A Parametric Interval Markov Chain (pIMC for short) is a tuple  $\mathcal{P} = (S, s_0, P, V, Y)$ , where  $S$ ,  $s_0$ ,  $V$ , and  $Y$  are defined as for pIMCs, and  $P: S \times S \rightarrow \mathbb{I}(\mathbb{Q}_Y)$  associates with each potential transition a (parametric) interval.

In [12] the authors introduced pIMCs where parametric interval endpoints are limited to linear combination of parameters. In our contribution we extend the pIMC model by allowing rational functions over parameters as endpoints of

<sup>1</sup> Indeed, when  $0 \leq v(f_1) \leq v(f_2) \leq 1$  is not respected, the interval is inconsistent and therefore empty.

Fig. 7. pIMC  $\mathcal{P}$ .

parametric intervals. Given a pIMC  $\mathcal{P} = (S, s_0, P, V, Y)$  and a valuation  $v$ , we write  $v(\mathcal{P})$  for the IMC  $(S, s_0, P_v, V)$  obtained by replacing the transition function  $P$  from  $\mathcal{P}$  with the function  $P_v: S \times S \rightarrow \mathbb{I}$  defined by  $P_v(s, s') = v(P(s, s'))$  for all  $s, s' \in S$ . The IMC  $v(\mathcal{P})$  is called an *instance* of pIMC  $\mathcal{P}$ . Finally, depending on the semantics chosen for IMCs, three satisfaction relations can be defined between MCs and pIMCs. They are written  $\models_{\text{pI}}^a$ ,  $\models_{\text{pI}}^d$ , and  $\models_{\text{pI}}^o$  and defined as follows:  $\mathcal{M} \models_{\text{pI}}^a \mathcal{P}$  (resp.  $\models_{\text{pI}}^d$ ,  $\models_{\text{pI}}^o$ ) if and only if there exists an IMC  $\mathcal{I}$  instance of  $\mathcal{P}$  such that  $\mathcal{M} \models_{\text{I}}^a \mathcal{I}$  (resp.  $\models_{\text{I}}^d$ ,  $\models_{\text{I}}^o$ ).

**Example 6.** Consider the pIMC  $\mathcal{P} = (S, s_0, P, V, Y)$  given in Fig. 7. The set of states  $S$  and the labelling function are the same as in the MC and the IMC presented in Figs. 1 and 3 respectively. The set of parameters  $Y$  has two elements  $p$  and  $q$ . Finally, the parametric intervals from the transition function  $P$  are given by the edge labelling (e.g.,  $P(s_1, s_3) = [0.3, q]$ ,  $P(s_2, s_4) = [0, 0.5]$ , and  $P(s_3, s_3) = [1, 1]$ ). Note that the IMC  $\mathcal{I}$  from Fig. 3 is an instance of  $\mathcal{P}$  (by assigning the value 0.6 to the parameter  $p$  and 0.5 to  $q$ ). Furthermore, as said in Example 5, the Markov chains  $\mathcal{M}_1$  and  $\mathcal{M}_2$  (from Figs. 1 and 4 respectively) satisfy  $\mathcal{I}$  w.r.t.  $\models_{\text{I}}^a$ , therefore  $\mathcal{M}_1$  and  $\mathcal{M}_2$  satisfy  $\mathcal{P}$  w.r.t.  $\models_{\text{pI}}^a$ .

In the following, we consider that the size of a pMC, IMC, or pIMC  $\mathcal{L}$ , written  $|\mathcal{L}|$ , corresponds to its number of states plus its number of transitions not reduced to 0,  $[0, 0]$  or  $\emptyset$ . We will also often need to consider the predecessors ( $\text{Pred}$ ), and the successors ( $\text{Succ}$ ) of some given states. Given a pIMC with a set of states  $S$ , a state  $s$  in  $S$ , and a subset  $S'$  of  $S$ , we write:

- $\text{Pred}(s) = \{s' \in S \mid P(s', s) \notin \{\emptyset, [0, 0]\}\}$
- $\text{Succ}(s) = \{s' \in S \mid P(s, s') \notin \{\emptyset, [0, 0]\}\}$
- $\text{Pred}(S') = \bigcup_{s' \in S'} \text{Pred}(s')$
- $\text{Succ}(S') = \bigcup_{s' \in S'} \text{Succ}(s')$

## 2.2. Abstraction model comparisons

IMC, pMC, and pIMC are three Markov chain abstraction models. In order to compare their expressiveness and compactness, we introduce the comparison operators  $\sqsubseteq$  and  $\equiv$ . Let  $(\mathbb{L}_1, \models_1)$  and  $(\mathbb{L}_2, \models_2)$  be two Markov chain abstraction models containing respectively  $\mathcal{L}_1$  and  $\mathcal{L}_2$ . We say that  $\mathcal{L}_1$  is entailed by  $\mathcal{L}_2$ , written  $\mathcal{L}_1 \sqsubseteq \mathcal{L}_2$ , if and only if all MCs satisfying  $\mathcal{L}_1$  satisfy  $\mathcal{L}_2$  modulo bisimilarity. (i.e.,  $\forall \mathcal{M} \models_1 \mathcal{L}_1, \exists \mathcal{M}' \models_2 \mathcal{L}_2$  such that  $\mathcal{M}$  is bisimilar to  $\mathcal{M}'$ ). We say that  $\mathcal{L}_1$  is (semantically) equivalent to  $\mathcal{L}_2$ , written  $\mathcal{L}_1 \equiv \mathcal{L}_2$ , if and only if  $\mathcal{L}_1 \sqsubseteq \mathcal{L}_2$  and  $\mathcal{L}_2 \sqsubseteq \mathcal{L}_1$ . Definition 2.8 introduces succinctness based on the sizes of the abstractions.

**Definition 2.8 (Succinctness).** Let  $(\mathbb{L}_1, \models_1)$  and  $(\mathbb{L}_2, \models_2)$  be two Markov chain abstraction models.  $\mathbb{L}_1$  is at least as succinct as  $\mathbb{L}_2$ , written  $\mathbb{L}_1 \leq \mathbb{L}_2$ , if and only if there exists a polynomial  $p$  such that for every  $\mathcal{L}_2 \in \mathbb{L}_2$ , there exists  $\mathcal{L}_1 \in \mathbb{L}_1$  such that  $\mathcal{L}_1 \equiv \mathcal{L}_2$  and  $|\mathcal{L}_1| \leq p(|\mathcal{L}_2|)$ . Moreover,  $\mathbb{L}_1$  is strictly more succinct than  $\mathbb{L}_2$ , written  $\mathbb{L}_1 < \mathbb{L}_2$ , if and only if  $\mathbb{L}_1 \leq \mathbb{L}_2$  and  $\mathbb{L}_2 \not\leq \mathbb{L}_1$ .

We start with a comparison of the succinctness of the pMC and IMC abstractions. Since pMCs allow the expression of dependencies between the probabilities assigned to distinct transitions while IMCs allow all transitions to be independent, it is clear that there are pMCs without any equivalent IMCs (regardless of the IMC semantics used), therefore  $(\text{IMC}, \models_{\text{I}}^o) \not\leq (\text{pMC}, \models_{\text{p}})$ ,  $(\text{IMC}, \models_{\text{I}}^d) \not\leq (\text{pMC}, \models_{\text{p}})$ , and  $(\text{IMC}, \models_{\text{I}}^a) \not\leq (\text{pMC}, \models_{\text{p}})$ . On the other hand, IMCs with the IMDP and at-every-step semantics allow unbounded unfolding of the state space with different probabilistic choices, which implies that pMCs would need infinitely many states to represent the same set of MC implementations. Finally, we show that the set of MC implementations of an IMC with the once-and-for-all semantics can also be represented with a pMC of the same size as the original IMC. As a consequence, pMCs are strictly more succinct than IMCs with the once-and-for-all semantics. This is formalized in the following lemma.

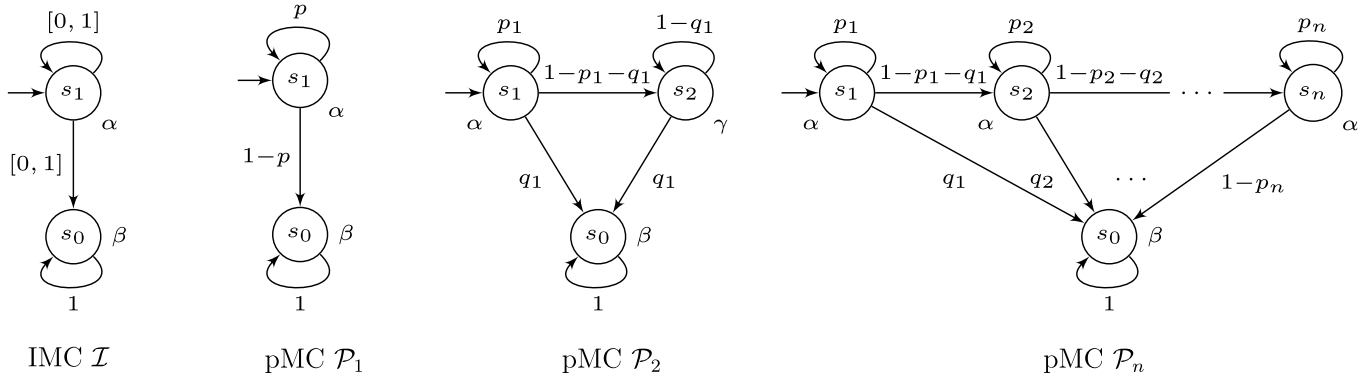


Fig. 8. IMC  $\mathcal{I}$  with three pMCs  $\mathcal{P}_1$ ,  $\mathcal{P}_2$ , and  $\mathcal{P}_n$ .

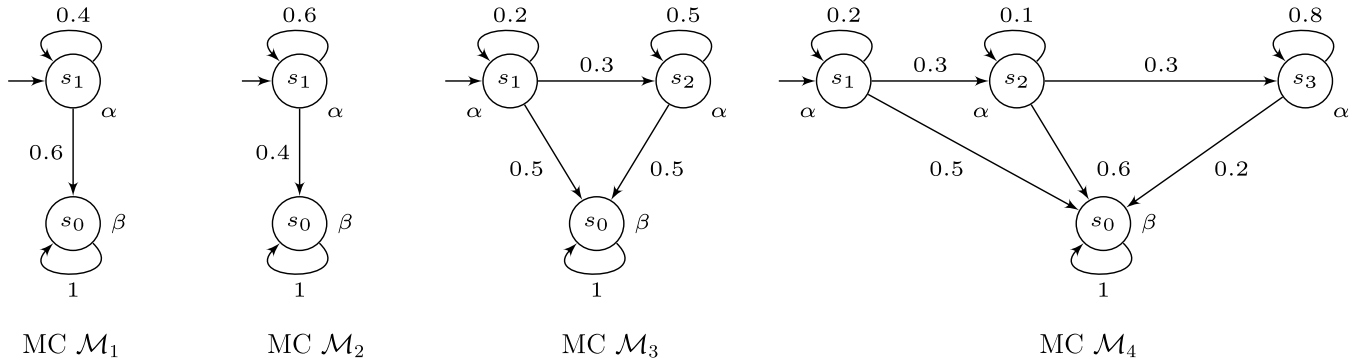


Fig. 9. MCs satisfying the IMC and/or one of the pMCs of Fig. 8.

**Lemma 1.** pMC and IMC abstraction models are mostly not comparable in terms of succinctness: (1)  $(\text{pMC}, \models_{\text{p}}) \not\leq (\text{IMC}, \models_{\text{I}}^a)$ , (2)  $(\text{pMC}, \models_{\text{p}}) \not\leq (\text{IMC}, \models_{\text{I}}^d)$ , (3)  $(\text{IMC}, \models_{\text{I}}^a) \not\leq (\text{pMC}, \models_{\text{p}})$ , and (4)  $(\text{IMC}, \models_{\text{I}}^d) \not\leq (\text{pMC}, \models_{\text{p}})$ . However, pMCs are strictly more succinct than IMCs with the once-and-for-all semantics: (5)  $(\text{pMC}, \models_{\text{p}}) < (\text{IMC}, \models_{\text{I}}^o)$ .

**Proof.** Let  $(L_1, \models_1)$  and  $(L_2, \models_2)$  be two Markov chain abstraction models. Recall that according to the succinctness definition (cf. Definition 2.8)  $L_1 \not\leq L_2$  if there exists  $\mathcal{L}_2 \in L_2$  such that either  $\mathcal{L}_1 \not\equiv \mathcal{L}_2$  for all  $\mathcal{L}_1 \in L_1$ , or the only way to have  $\mathcal{L}_1 \equiv \mathcal{L}_2$  is with  $\mathcal{L}_1$  exponentially larger than  $\mathcal{L}_2$ . Here, we provide examples of IMCs (resp. pMCs) that have no equivalent pMCs (resp. IMCs). We finish by providing a construction that produces a pMC representing the same set of MC implementations as a given IMC with the once-and-for-all semantics.

- (1)–(2) Consider the IMC  $\mathcal{I}$  from Fig. 8. Clearly, all of the MCs given in Fig. 9 satisfy  $\mathcal{I}$  with  $\models_{\text{I}}^a$  and  $\models_{\text{I}}^d$ . Remark that from every state  $s_i$ ,  $i > 0$  in  $\mathcal{M}_4$ , there is a different probability of going to  $s_0$ . As a consequence, the states are not bisimilar, and therefore building a pMC that matches these probabilities will require as many states as there are in the MC. Since  $\models_{\text{I}}^a$  and  $\models_{\text{I}}^d$  allow to represent MCs with an unbounded number of states (and thus an unbounded number of different probability values for going to  $s_0$ ), we would need pMCs such as  $\mathcal{P}_n$  from Fig. 8 with infinitely many states to represent the same set of MC implementations.
- (3)–(4) Consider pMC  $\mathcal{P}_2$  from Fig. 8. Because of the semantics of pMCs, it enforces the transitions from  $s_1$  to  $s_0$  and the transition from  $s_2$  to  $s_0$  to have the same probability. Since IMCs do not allow such dependencies between transitions, the set of MCs satisfying  $\mathcal{P}_2$  cannot be matched by any IMC semantics.
- (5) First observe that the arguments above (3)–(4) also apply to  $(\text{IMC}, \models_{\text{I}}^o)$ . As a consequence it is clear that  $(\text{IMC}, \models_{\text{I}}^o) \not\leq (\text{pMC}, \models_{\text{p}})$ . Now consider an IMC  $\mathcal{I} = (S, s_0, P, V)$ . We build a pMC  $\mathcal{I}' = (S, s_0, P', V, Y)$  that has the same set of MC implementations as  $\mathcal{I}$ . The intuition is that each transition equipped with an interval of the form  $[l, u]$ , with  $0 \leq l \leq u \leq 1$  can be matched with a transition with probability  $l + (u - l) \cdot p$ , where  $p$  is a fresh parameter. As a consequence, when  $p$  ranges over  $[0, 1]$ ,  $(l + (u - l) \cdot p)$  ranges over the interval  $[l, u]$ . Formally, we define  $Y = \{p_{(s,s')} \mid (s, s') \in S^2\}$  and for all  $s, s' \in S$ ,  $P'(s, s') = l_{(s,s')} + (u_{(s,s')} - l_{(s,s')}) \cdot p_{(s,s')}$ , where  $P(s, s') = [l_{(s,s')}, u_{(s,s')}]$ . It is then easy to see that  $\mathcal{I}'$  has exactly the same set of MC implementations as  $\mathcal{I}$  under the once-and-for-all semantics. Moreover, the size of  $\mathcal{I}'$  is equal to the size of  $\mathcal{I}$ . As a consequence,  $(\text{pMC}, \models_{\text{p}}) < (\text{IMC}, \models_{\text{I}}^o)$ .  $\square$

We now compare pMCs and IMCs to pIMCs. We show that  $(\text{pIMC}, \models_{\text{pI}}^o)$  is equivalent to  $(\text{pMC}, \models_{\text{p}})$  and pIMC is strictly more succinct than IMC for the three semantics.



Our comparison results are presented in the following Proposition.

**Proposition 2.** *The Markov chain abstraction models can be ordered as follows w.r.t. succinctness: (1)  $(\text{pIMC}, \models_{\text{pI}}^o) < (\text{IMC}, \models_{\text{I}}^o)$ , (2)  $(\text{pIMC}, \models_{\text{pI}}^d) < (\text{IMC}, \models_{\text{I}}^d)$ , (3)  $(\text{pIMC}, \models_{\text{pI}}^a) < (\text{IMC}, \models_{\text{I}}^a)$ , (4)  $(\text{pIMC}, \models_{\text{pI}}^d) < (\text{pMC}, \models_{\text{p}})$ , (5)  $(\text{pIMC}, \models_{\text{pI}}^a) < (\text{pMC}, \models_{\text{p}})$ , and (6)  $(\text{pIMC}, \models_{\text{pI}}^o) \equiv (\text{pMC}, \models_{\text{p}})$ .*

**Proof.** First, recall that the  $\text{pIMC}$  model is a Markov chain abstraction model allowing to declare parametric interval transitions, while the  $\text{pMC}$  model allows only parametric transitions (without intervals), and the  $\text{IMC}$  model allows interval transitions without parameters. Clearly, any  $\text{pMC}$  and any  $\text{IMC}$  can be translated into a  $\text{pIMC}$  with the right semantics (once-and-for-all for  $\text{pMC}$ s and the chosen  $\text{IMC}$  semantics for  $\text{IMC}$ s). This means that  $(\text{pIMC}, \models_{\text{pI}}^o)$  is more succinct than  $(\text{pMC}, \models_{\text{p}})$  and  $\text{pIMC}$  is more succinct than  $\text{IMC}$  for the three semantics. Furthermore, since  $\text{pMC}$  and  $\text{IMC}$  equipped with  $\models_{\text{I}}^a$  and  $\models_{\text{I}}^d$  are not comparable due to the above results, we have that the  $\text{pIMC}$  abstraction model equipped with  $\models_{\text{I}}^a$  and  $\models_{\text{I}}^d$  is strictly more succinct than the  $\text{pMC}$  abstraction model and that the  $\text{pIMC}$  abstraction model is strictly more succinct than the  $\text{IMC}$  abstraction model for all three semantics. This proves items (1)–(5).

Regarding item (6), we propose a construction that derives, from any given  $\text{pIMC}$   $\mathcal{P}$ , a  $\text{pMC}$   $\mathcal{I}$  that represents the same set of MC implementations up to bisimilarity.

Let  $\mathcal{P} = (S, s_0, P, V, Y)$  be a  $\text{pIMC}$  and consider the  $\text{pMC}$   $\mathcal{I} = (S', s'_0, P', V', Y')$  such that

- $S' = \{s^b \mid s \in S\} \cup \{s^a \mid s \in S\} \cup \{s^r \mid s \in S\}$ ,
- $s'_0 = s_0^b$ ,
- $V'(s^k) = V(s)$  for all  $k \in \{a, b, r\}$ ,
- $Y' = Y \cup \{p_{(s,t)} \mid s, t \in S\}$ , and
- $P'$  is such that for all  $s^k \in S'$  (regardless of  $k$ ) and for all  $t \in S$  such that  $P(s, t) = [l_{(s,t)}, u_{(s,t)}]$ , where  $l_{(s,t)}$  and  $u_{(s,t)}$  can be rational functions over  $Y$ ,
  - $P'(s^k, t^b) = p_{(s,t)} \cdot (p_{(s,t)} - l_{(s,t)})$ ,
  - $P'(s^k, t^a) = p_{(s,t)} \cdot (1 - (p_{(s,t)} - l_{(s,t)})) \cdot (u_{(s,t)} - p_{(s,t)})$ , and
  - $P'(s^k, t^r) = p_{(s,t)} \cdot (1 - (p_{(s,t)} - l_{(s,t)})) \cdot (1 - (u_{(s,t)} - p_{(s,t)}))$ .

The intuition in this construction is that we build 3 bisimilar copies of each state in  $S$  and use one parameter  $p_{(s,t)}$  to represent the probability of the transition between  $s$  and  $t$  in an implementation of  $\mathcal{P}$ . From any copy of  $s$ , the transition leading to the copy of  $t$  labelled with  $b$  enforces that  $p_{(s,t)} \geq l_{(s,t)}$ . The transition leading to the copy of  $t$  labelled with  $a$  enforces that  $p_{(s,t)} \leq u_{(s,t)}$ , and as a consequence that  $l_{(s,t)} \leq u_{(s,t)}$ . Finally, the transition leading to the copy of  $t$  labelled with  $r$  ensures that the overall probability of going from a given copy of  $s$  to all copies of  $t$  is  $p_{(s,t)}$ . Since all copies of a given state have the same outgoing transitions and the same label, they are all bisimilar. Therefore, any MC implementation of  $\mathcal{I}$  is bisimilar to a MC with a single copy of each state aggregating all outgoing transitions (with a total probability of  $p_{(s,t)}$  for each transition  $(s, t)$ ), but an implementation can only exist if the constraints on the parameters and intervals are satisfied. On the other hand, given an implementation of  $\mathcal{P}$  with the once-and-for-all semantics, all values of  $l_{(s,t)}$ ,  $u_{(s,t)}$ , and  $p_{(s,t)}$  are known for this implementation. It is therefore easy to build a new MC following the same construction as above. This new MC is bisimilar to the original one and trivially satisfies  $\mathcal{I}$ .

Finally, we remark that the size of  $\mathcal{I}$  is linear in the size of  $\mathcal{P}$  (there are 3 copies of each state and each transition is split in 3). As a consequence, we can conclude that  $(\text{pIMC}, \models_{\text{pI}}^o) \equiv (\text{pMC}, \models_{\text{p}})$ .  $\square$

### 3. Qualitative properties

As seen above,  $\text{pIMC}$ s are a succinct abstraction formalism for MCs. The aim of this section is to investigate qualitative properties for  $\text{pIMC}$ s, *i.e.*, properties that can be evaluated at the specification ( $\text{pIMC}$ ) level, but that entail properties on its MC implementations.  $\text{pIMC}$  specifications are very expressive as they allow the abstraction of transition probabilities using both intervals and parameters. Unfortunately, as it is the case for  $\text{IMC}$ s, this allows the expression of incorrect specifications. In the  $\text{IMC}$  setting, this is the case either when some intervals are ill-formed or when there is no probability distribution matching the interval constraints of the outgoing transitions of some reachable state. In this case, no MC implementation exists that satisfies the  $\text{IMC}$  specification. Deciding whether an implementation that satisfies a given specification exists is called the *consistency* problem. In the  $\text{pIMC}$  setting, the consistency problem is made more complex because of the parameters which can also induce inconsistencies in some cases. One could also be interested in verifying whether there exists an implementation that reaches some target states/labels, and if so, propose a parameter valuation ensuring this property. This problem is called the *consistent reachability* problem. Both the consistency and the consistent reachability problems have already been investigated in the  $\text{IMC}$  and  $\text{pIMC}$  setting [21,12]. In this section, we briefly recall these problems and propose new solutions based on CSP encodings. Our encodings are linear in the size of the original  $\text{pIMC}$ s whereas the algorithms from [21,12] are exponential.

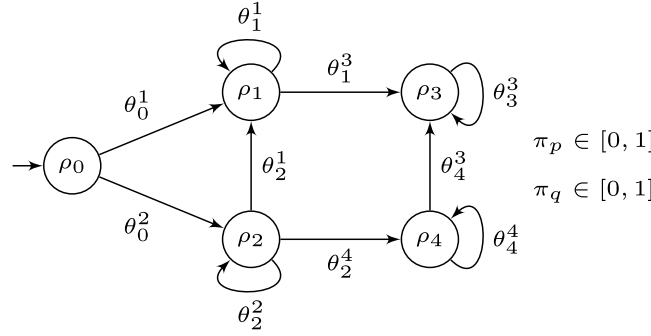


Fig. 10. Variables in the CSP produced by  $C_{\exists c}$  for the pIMC  $\mathcal{P}$  from Fig. 7.

### 3.1. Existential consistency

A pIMC  $\mathcal{P}$  is existentially consistent if and only if there exists a MC  $\mathcal{M}$  satisfying  $\mathcal{P}$  (i.e., there exists a MC  $\mathcal{M}$  satisfying an IMC  $\mathcal{I}$  instance of  $\mathcal{P}$ ). As seen in Section 2, pIMCs are equipped with three semantics: once-and-for-all ( $\models_{\text{pI}}^o$ ), IMDP ( $\models_{\text{pI}}^d$ ), and at-every-step ( $\models_{\text{pI}}^a$ ). Recall that  $\models_{\text{pI}}^o$  imposes that implementations need to have the same graph structure as (or a substructure of) the corresponding specification, up to renaming. In contrast,  $\models_{\text{pI}}^d$  and  $\models_{\text{pI}}^a$  allow implementations to have a different graph structure. It therefore seems that some pIMCs could be inconsistent w.r.t.  $\models_{\text{pI}}^o$  while being consistent w.r.t.  $\models_{\text{pI}}^a$ . On the other hand, checking the consistency w.r.t.  $\models_{\text{pI}}^o$  seems easier because of the fixed graph structure.

In [21], the author firstly proved that  $\models_{\text{pI}}^a$  and  $\models_{\text{pI}}^o$  semantics are equivalent w.r.t. existential consistency, and proposed a CSP encoding for verifying this property which is exponential in the size of the pIMC. Together with Proposition 1, this result ensures that the three semantics  $\models_{\text{pI}}^d$ ,  $\models_{\text{pI}}^a$ , and  $\models_{\text{pI}}^o$  are equivalent w.r.t. existential consistency. Based on this result of semantics equivalence we propose a new CSP encoding, written  $C_{\exists c}$ , for verifying the existential consistency property for pIMCs.

Let  $\mathcal{P} = (S, s_0, P, V, Y)$  be a pIMC. We write  $C_{\exists c}(\mathcal{P})$  for the CSP produced by  $C_{\exists c}$  according to  $\mathcal{P}$ . Any solution of  $C_{\exists c}(\mathcal{P})$  will correspond to a MC satisfying  $\mathcal{P}$ . In  $C_{\exists c}(\mathcal{P})$ , we use one variable  $\pi_p$  with domain  $[0, 1]$  per parameter  $p$  in  $Y$ ; one variable  $\theta_s^{s'}$  with domain  $[0, 1]$  per transition  $(s, s')$  in  $\{\{s\} \times \text{Succ}(s) \mid s \in S\}$ ; and one Boolean variable  $\rho_s$  per state  $s$  in  $S$ . These Boolean variables will indicate for each state whether it appears in the MC solution of the CSP (i.e., in the MC satisfying the pIMC  $\mathcal{P}$ ). For each state  $s \in S$ , constraints are as follows:

- (1)  $\rho_s$ , if  $s = s_0$
- (2)  $\neg \rho_s \Leftrightarrow \sum_{s' \in \text{Pred}(s) \setminus \{s\}} \theta_s^{s'} = 0$ , if  $s \neq s_0$
- (3)  $\neg \rho_s \Leftrightarrow \sum_{s' \in \text{Succ}(s)} \theta_s^{s'} = 0$
- (4)  $\rho_s \Leftrightarrow \sum_{s' \in \text{Succ}(s)} \theta_s^{s'} = 1$
- (5)  $\rho_s \Rightarrow \theta_s^{s'} \in P(s, s')$ , for all  $s' \in \text{Succ}(s)$

Recall that given a pIMC  $\mathcal{P}$  the objective of the CSP  $C_{\exists c}(\mathcal{P})$  is to construct a MC  $\mathcal{M}$  satisfying  $\mathcal{P}$ . Constraint (1) states that the initial state  $s_0$  appears in  $\mathcal{M}$ . Constraint (2) ensures that for each non-initial state  $s$ , variable  $\rho_s$  is set to false if and only if  $s$  is not reachable from its predecessors. Constraint (4) ensures that if a state  $s$  appears in  $\mathcal{M}$ , then its outgoing transitions form a probability distribution. On the contrary, constraint (3) propagates non-appearing states (i.e., if a state  $s$  does not appear in  $\mathcal{M}$  then all its outgoing transitions are set to zero). Finally, constraint (5) states that, for all appearing states, the outgoing transition probabilities must be selected inside the specified intervals.

**Example 7.** Consider the pIMC  $\mathcal{P}$  given in Fig. 7. Fig. 10 describes the variables in  $C_{\exists c}(\mathcal{P})$ : one variable per transition (e.g.,  $\theta_0^1, \theta_0^2, \theta_1^1$ ), one Boolean variable per state (e.g.,  $\rho_0, \rho_1$ ), and one variable per parameter ( $\pi_p$  and  $\pi_q$ ). The following constraints correspond to the constraints (2), (3), (4), and (5) generated by our encoding  $C_{\exists c}$  for the state 2 of  $\mathcal{P}$ :

$$\begin{aligned} \neg \rho_2 &\Leftrightarrow \theta_0^2 = 0 & \rho_2 &\Rightarrow 0 \leq \theta_2^1 \leq \pi_p \\ \neg \rho_2 &\Leftrightarrow \theta_2^1 + \theta_2^2 + \theta_2^4 = 0 & \rho_2 &\Rightarrow 0.2 \leq \theta_2^2 \leq \pi_q \\ \rho_2 &\Leftrightarrow \theta_2^1 + \theta_2^2 + \theta_2^4 = 1 & \rho_2 &\Rightarrow 0 \leq \theta_2^4 \leq 0.5 \end{aligned}$$

Finally, Fig. 11 describes a solution for the CSP  $C_{\exists c}(\mathcal{P})$ . Note that given a solution of a pIMC encoded by  $C_{\exists c}$ , one can construct a MC satisfying the given pIMC w.r.t.  $\models_{\text{pI}}^o$  by keeping all states  $s$  such that  $\rho_s$  is equal to true and considering the transition function given by the probabilities in the  $\theta_s^{s'}$  variables. We now show that our encoding works as expected.

**Proposition 3.** A pIMC  $\mathcal{P}$  is existentially consistent if and only if  $C_{\exists c}(\mathcal{P})$  is satisfiable.

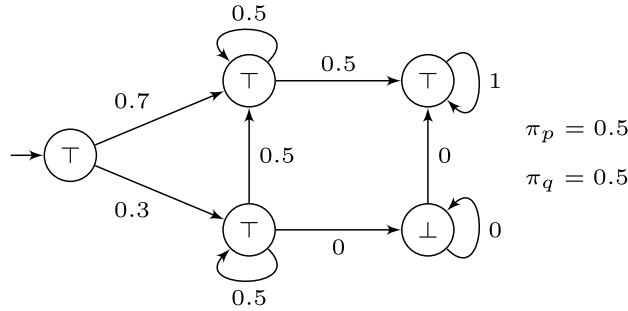


Fig. 11. A solution to the CSP  $C_{\exists c}(\mathcal{P})$  for the pIMC  $\mathcal{P}$  from Fig. 7.

**Proof.** Let  $\mathcal{P} = (S, s_0, P, V, Y)$  be a pIMC.

[ $\Leftarrow$ ] The CSP  $C_{\exists c}(\mathcal{P}) = (X, D, C)$  being satisfiable implies that there exists a valuation  $v$  of the variables in  $X$  satisfying all constraints in  $C$ . Consider the MC  $\mathcal{M} = (S, s_0, p, V)$  such that  $p(s, s') = v(\theta_s^{s'})$ , for all  $\theta_s^{s'} \in X$ , and  $p(s, s') = 0$  otherwise.

First, we show by induction that for any state  $s$  in  $S$ : “if  $s$  is reachable in  $\mathcal{M}$  then  $v(\rho_s)$  equals to true”. This is correct for the initial state  $s_0$  thanks to the constraint (1). Let  $s$  be a state in  $S$  and assume that the property is correct for at least one of its predecessors. By the constraint (2),  $v(\rho_s)$  equals true if there exists at least one predecessor  $s'' \neq s$  reaching  $s$  with a non-zero probability (i.e.,  $v(\theta_{s''}^s) \neq 0$ ). Let  $s''$  be the one satisfying the induction property. By the constraint (3), this is only possible if  $v(\rho_{s''})$  equals true. Thus  $v(\rho_s)$  equals true if there exists one reachable state  $s''$  such that  $v(\theta_{s''}^s) \neq 0$ . Therefore, all reachable states  $s$  in  $\mathcal{M}$  are such that  $v(\rho_s)$  equals true. As a consequence, the constraint (4) ensures that the probabilities of the outgoing transitions of all reachable states sum up to 1.

We now show that  $\mathcal{M}$  satisfies the pIMC  $\mathcal{P}$  w.r.t.  $\models_{\perp}^o$ . We proved above that for all reachable states  $s$  in  $\mathcal{M}$ , we have  $v(\rho_s)$  equals to true. By the constraint (5) it implies that for all reachable states  $s$  in  $\mathcal{M}$ :  $p(s, s') \in P(s, s')$  for all  $s'$ .<sup>2</sup> As a consequence,  $\mathcal{M} \models_{\perp}^o \mathcal{P}$ , and therefore  $\mathcal{M} \models_{\perp}^a \mathcal{P}$ .

[ $\Rightarrow$ ] The pIMC  $\mathcal{P}$  being consistent implies by the Theorem 4 from [12] stating that  $\models_{\perp}^a$  and  $\models_{\perp}^o$  are equivalent w.r.t. qualitative reachability, that there exists an implementation of the form  $\mathcal{M} = (S, s_0, p, V)$  where, for all reachable states  $s$  in  $\mathcal{M}$ , it holds that  $p(s, s') \in P(s, s')$  for all  $s'$  in  $S$ . Consider  $\mathcal{M}' = (S, s_0, p', V)$  such that for each non reachable state  $s$  in  $S$ :  $p'(s, s') = 0$ , for all  $s' \in S$ . The valuation  $v$  is such that  $v(\rho_s)$  equals true if and only if  $s$  is reachable in  $\mathcal{M}$ ,  $v(\theta_s^{s'}) = p'(s, s')$ , and for each parameter  $y \in Y$  a valid value can be selected according to  $p$  and  $P$  when considering reachable states. Finally, by construction,  $v$  satisfies the CSP  $C_{\exists c}(\mathcal{P})$ .  $\square$

Our existential consistency encoding is linear in the size of the pIMC instead of exponential for the encoding from [12] which enumerates the powerset of the states in the pIMC resulting in deep nesting of conjunctions and disjunctions.

### 3.2. Qualitative reachability

Let  $\mathcal{P} = (S, s_0, P, V, Y)$  be a pIMC and  $\Gamma \subseteq A$  be a state label. We say that  $\Gamma$  is *existentially reachable* in  $\mathcal{P}$  if and only if there exists an implementation  $\mathcal{M}$  of  $\mathcal{P}$  where  $\Gamma$  is reachable (i.e.,  $\mathbb{P}^{\mathcal{M}}(\diamond \Gamma) > 0$ ). In a dual way, we say that  $\Gamma$  is *universally reachable* in  $\mathcal{P}$  if and only if  $\Gamma$  is reachable in any implementation  $\mathcal{M}$  of  $\mathcal{P}$ . As for existential consistency, we use a result from [21] that states that the  $\models_{\perp}^a$  and the  $\models_{\perp}^o$  pIMC semantics are equivalent w.r.t. existential (and universal) reachability. As for the consistency problem, we get by Proposition 1 that the three IMC semantics are equivalent w.r.t. existential (and universal) reachability. Note first that in our  $C_{\exists c}$  encoding each  $\rho_s$  variable indicates if the state  $s$  appears in the constructed Markov chain. However, the  $\rho_s$  variable does not indicate if the state  $s$  is reachable from the initial state, but only if it is reachable from at least one other state (i.e., possibly different from  $s_0$ ). Indeed, if the graph representation of the constructed Markov chain has strongly connected components (SCCs for short), then all  $\rho_s$  variables in one SCC may be set to true while this SCC may be unreachable from the initial state. This is not an issue in the case of the consistency problem. Indeed, if a Markov chain containing an unreachable SCC is proved consistent, then it is also consistent without this unreachable SCC. However, in the case of the reachability problem, these SCCs are an issue. The following encoding therefore takes into account these isolated SCCs such that  $\rho_s$  variables are set to true if and only if the corresponding states are all reachable from the initial state. This encoding will solve the qualitative reachability problems (i.e., checking qualitative reachability from the initial state). We propose a new CSP encoding, written  $C_{\exists r}$ , that extends  $C_{\exists c}$ , for verifying these properties. Formally, CSP  $C_{\exists r}(\mathcal{P}) = (X \cup X', D \cup D', C \cup C')$  is such that  $(X, D, C) = C_{\exists c}(\mathcal{P})$ ,  $X'$  contains one integer variable  $\omega_s$  with domain  $[0, |S|]$  per state  $s$  in  $S$ ,  $D'$  contains the domains of these variables, and  $C'$  is composed of the following constraints for each state  $s \in S$ :

<sup>2</sup> As illustrated in Example 7,  $\mathcal{M}$  might not be a well formed MC since some unreachable states do not respect the probability distribution property. However, one can correct it by simply setting one of its outgoing transition to 1 for each unreachable state, which does not impact satisfaction.

- (6)  $\omega_s = 1$ , if  $s = s_0$   
 (7)  $\omega_s \neq 1$ , if  $s \neq s_0$   
 (8)  $\rho_s \Leftrightarrow (\omega_s \neq 0)$   
 (9)  $\omega_s > 1 \Rightarrow \bigvee_{s' \in \text{Pred}(s) \setminus \{s\}} (\omega_s = \omega_{s'} + 1) \wedge (\theta_{s'}^s > 0)$ , if  $s \neq s_0$   
 (10)  $\omega_s = 0 \Leftrightarrow \bigwedge_{s' \in \text{Pred}(s) \setminus \{s\}} (\omega_{s'} = 0) \vee (\theta_{s'}^s = 0)$ , if  $s \neq s_0$

Recall first that CSP  $\mathbf{C}_{\exists c}(P)$  constructs a Markov chain  $\mathcal{M}$  satisfying  $\mathcal{P}$  w.r.t.  $\models_{\perp}^0$ . Informally, for each state  $s$  in  $\mathcal{M}$  the constraints (6), (7), (9), and (10) in  $\mathbf{C}_{\exists r}$  ensure that  $\omega_s = k$  if and only if there exists in  $\mathcal{M}$  a path from the initial state to  $s$  of length  $k - 1$  with non zero probability; and state  $s$  is not reachable in  $\mathcal{M}$  from the initial state  $s_0$  if and only if  $\omega_s$  equals to 0. Finally, constraint (8) enforces the Boolean reachability indicator variable  $\rho_s$  to be set to true if and only if there exists a path with non zero probability in  $\mathcal{M}$  from the initial state  $s_0$  to  $s$  (i.e.,  $\omega_s \neq 0$ ).

Let  $S_{\Gamma}$  be the set of states from  $\mathcal{P}$  labelled with  $\Gamma$ . Recall that  $\mathbf{C}_{\exists r}(\mathcal{P})$  produces a Markov chain satisfying  $\mathcal{P}$  where reachable states  $s$  are such that  $\rho_s = \text{true}$ . As a consequence,  $\Gamma$  is existentially reachable in  $\mathcal{P}$  if and only if  $\mathbf{C}_{\exists r}(\mathcal{P})$  admits a solution such that  $\bigvee_{s \in S_{\Gamma}} \rho_s$ ; and  $\Gamma$  is universally reachable in  $\mathcal{P}$  if and only if  $\mathbf{C}_{\exists r}(\mathcal{P})$  admits no solution such that  $\bigwedge_{s \in S_{\Gamma}} \neg \rho_s$ . This is formalized in the following proposition.

**Proposition 4.** Let  $\mathcal{P} = (S, s_0, P, V, Y)$  be a pIMC,  $\Gamma \subseteq A$  be a state label,  $S_{\Gamma} = \{s \mid V(s) = \Gamma\}$ , and  $(X, D, C)$  be the CSP  $\mathbf{C}_{\exists r}(\mathcal{P})$ .

- CSP  $(X, D, C \cup \bigvee_{s \in S_{\Gamma}} \rho_s)$  is satisfiable if and only if  $\Gamma$  is existentially reachable in  $\mathcal{P}$
- CSP  $(X, D, C \cup \bigwedge_{s \in S_{\Gamma}} \neg \rho_s)$  is unsatisfiable if and only if  $\Gamma$  is universally reachable in  $\mathcal{P}$

**Proof.** Let  $\mathcal{P} = (S, s_0, P, V, Y)$  be a pIMC,  $\Gamma \subseteq A$  be a state label,  $S_{\Gamma} = \{s \mid V(s) = \Gamma\}$ , and  $(X, D, C)$  be the CSP  $\mathbf{C}_{\exists r}(\mathcal{P})$ . Recall first, that by Proposition 3 the constraints (1) to (5) in  $\mathbf{C}_{\exists r}(\mathcal{P})$  are satisfied if and only if there exists a MC  $\mathcal{M}$  satisfying  $\mathcal{P}$  w.r.t.  $\models_{\perp}^a$ .

- $[\Rightarrow]$  If CSP  $(X, D, C \cup \bigvee_{s \in S_{\Gamma}} \rho_s)$  is satisfiable then there exists a valuation  $v$  solution of this CSP and a corresponding MC  $\mathcal{M}$  satisfying  $\mathcal{P}$  w.r.t.  $\models_{\perp}^a$  such as presented in the proof of Proposition 3. Furthermore, the constraints (6) to (10) ensure by induction that for all states  $s \in S$ :  $v(\omega_s) = k$  with  $k \geq 1$  if there exists a path from the initial state  $s_0$  to the state  $s$  of size  $k - 1$  with non zero probability in  $\mathcal{M}$ , and  $v(\omega_s) = 0$  otherwise. By constraint (8) we have that  $v(\rho_s) = \text{true}$  if and only if state  $s$  is reachable in  $\mathcal{M}$  from the initial state  $s_0$ . Finally, constraint  $\bigvee_{s \in S_{\Gamma}} \rho_s$  ensures that at least one state labelled with  $\Gamma$  must be reachable in  $\mathcal{M}$ . Thus,  $\Gamma$  is existentially reachable in  $\mathcal{P}$ .
- $[\Leftarrow]$  If  $\Gamma$  is existentially reachable in  $\mathcal{P}$ , then by [12] there exists a MC  $\mathcal{M}$  satisfying  $\mathcal{P}$  w.r.t.  $\models_{\perp}^a$  such that  $\Gamma$  is reachable in  $\mathcal{M}$ . By construction of our encoding, one can easily construct from  $\mathcal{M}$  a valuation  $v$  satisfying all constraints in  $C \cup \bigvee_{s \in S_{\Gamma}} \rho_s$  such that  $v(\omega_s)$  contains the size (plus one) of an existing path in  $\mathcal{M}$  from the initial state to the state  $s$  with a non zero probability, and  $v(\omega_s) = 0$  if  $s$  is not reachable in  $\mathcal{M}$ .
- Note that  $\Gamma$  is universally reachable in  $\mathcal{P}$  if and only if there is no MC  $\mathcal{M}$  satisfying  $\mathcal{P}$  w.r.t.  $\models_{\perp}^a$  such that none of the states labelled with  $\Gamma$  is reachable in  $\mathcal{M}$ . “CSP  $(X, D, C \cup \bigwedge_{s \in S_{\Gamma}} \neg \rho_s)$  is unsatisfiable” encodes this statement.  $\square$

As for the existential consistency problem, we have an exponential gain in terms of size of the encoding compared to [12]: the number of constraints and variables in  $\mathbf{C}_{\exists r}$  is linear in terms of the size of the encoded pIMC.

**Remark 1.** In  $\mathbf{C}_{\exists r}$  constraints (2) inherited from  $\mathbf{C}_{\exists c}$  are entailed by constraints (8) and (10) added to  $\mathbf{C}_{\exists r}$ . Thus, in a practical approach one may ignore constraints (2) from  $\mathbf{C}_{\exists c}$  if they do not improve the solver performance.

#### 4. Quantitative properties

We now move to the verification of quantitative reachability properties in pIMCs. Quantitative reachability has already been investigated in the context of pMCs and IMCs with the once-and-for-all semantics. In this section, we propose our main theoretical contribution: a theorem showing that the three IMC semantics are equivalent with respect to quantitative reachability, which allows the extension of all results from [19,11] to the at-every-step semantics. Based on this result, we also extend the CSP encodings introduced in Section 3 in order to solve quantitative reachability properties on pIMCs regardless of their semantics.

##### 4.1. Equivalence of $\models_{\perp}^0$ , $\models_{\perp}^d$ , and $\models_{\perp}^a$ w.r.t. quantitative reachability

Given an IMC  $\mathcal{I} = (S, s_0, P, V)$  and a state label  $\Gamma \subseteq A$ , a quantitative reachability property on  $\mathcal{I}$  is a property of the type  $\mathbb{P}^{\mathcal{I}}(\diamond \Gamma) \sim p$ , where  $0 \leq p \leq 1$  and  $\sim \in \{\leq, <, >, \geq\}$ . Such a property is verified if and only if there exists a MC  $\mathcal{M}$  satisfying  $\mathcal{I}$  (with the chosen semantics) such that  $\mathbb{P}^{\mathcal{M}}(\diamond \Gamma) \sim p$ . While the techniques we propose here work for all values of  $p$ , the techniques for *qualitative* reachability properties are usually more efficient when  $p = 0$  or 1.

As explained above, existing techniques and tools for verifying quantitative reachability properties on IMCs only focus on the once-and-for-all and the IMDP semantics. However, to the best of our knowledge, there are no works addressing the same problem with the at-every-step semantics or showing that addressing the problem in the once-and-for-all and IMDP setting is sufficiently general. The following theorem fills this theoretical gap by proving that the three IMC semantics are equivalent w.r.t. quantitative reachability. In other words, for all MCs  $\mathcal{M}$  such that  $\mathcal{M} \models_{\perp}^a \mathcal{I}$  or  $\mathcal{M} \models_{\perp}^d \mathcal{I}$  and for all state labels  $\Gamma$ , there exist MCs  $\mathcal{M}_{\leq}$  and  $\mathcal{M}_{\geq}$  such that  $\mathcal{M}_{\leq} \models_{\perp}^o \mathcal{I}$ ,  $\mathcal{M}_{\geq} \models_{\perp}^o \mathcal{I}$ , and  $\mathbb{P}^{\mathcal{M}_{\leq}}(\diamond\Gamma) \leq \mathbb{P}^{\mathcal{M}}(\diamond\Gamma) \leq \mathbb{P}^{\mathcal{M}_{\geq}}(\diamond\Gamma)$ . Informally, MC implementations for the at-every-step semantics may contain several copies of the original IMC states, each with a different probability of eventually reaching  $\Gamma$ . We therefore build  $\mathcal{M}_{\leq}$  (resp.  $\mathcal{M}_{\geq}$ ) by choosing as sole representative of a given IMC state the one from  $\mathcal{M}$  that has the lowest (resp. highest) probability of eventually reaching  $\Gamma$ . This way we ensure that  $\mathcal{M}_{\leq}$  (resp.  $\mathcal{M}_{\geq}$ ) satisfies the original IMC with the once-and-for-all semantics and has a lower (resp. higher) probability of eventually reaching  $\Gamma$  than  $\mathcal{M}$ . This is formalized in the following theorem.

**Theorem 1.** *Let  $\mathcal{I} = (S, s_0, P, V)$  be an IMC,  $\Gamma \subseteq A$  be a state label,  $\sim \in \{\leq, <, >, \geq\}$ , and  $0 < p < 1$ .  $\mathcal{I}$  satisfies  $\mathbb{P}^{\mathcal{I}}(\diamond\Gamma) \sim p$  with the once-and-for-all semantics if and only if  $\mathcal{I}$  satisfies  $\mathbb{P}^{\mathcal{I}}(\diamond\Gamma) \sim p$  with the IMDP semantics if and only if  $\mathcal{I}$  satisfies  $\mathbb{P}^{\mathcal{I}}(\diamond\Gamma) \sim p$  with the at-every-step semantics.*

The proof presented in the following is constructive: we use the structure of the relation  $\mathcal{R}$  from the definition of  $\models_{\perp}^a$  in order to build the MCs  $\mathcal{M}_{\leq}$  and  $\mathcal{M}_{\geq}$ .

In the following, when it is not specified the IMC satisfaction relation considered is the *at-every-step* semantics (i.e., the  $\models_{\perp}^a$  satisfaction relation). As said previously, we use the structure of the relation  $\mathcal{R}$  from the definition of  $\models_{\perp}^a$  in order to build the MCs  $\mathcal{M}_{\leq}$  and  $\mathcal{M}_{\geq}$  presented in Theorem 1. Thus, we introduce some notations relative to  $\mathcal{R}$ . Let  $\mathcal{I} = (S, s_0, P, V^I)$  be an IMC and  $\mathcal{M} = (T, t_0, p, V^M)$  be a MC such that  $\mathcal{M} \models_{\perp}^a \mathcal{I}$ . Let  $\mathcal{R} \subseteq T \times S$  be a satisfaction relation between  $\mathcal{M}$  and  $\mathcal{I}$ . For all  $t \in T$  we write  $\mathcal{R}(t)$  for the set  $\{s \in S \mid t \mathcal{R} s\}$ , and for all  $s \in S$  we write  $\mathcal{R}^{-1}(s)$  for the set  $\{t \in T \mid t \mathcal{R} s\}$ . Furthermore we say that  $\mathcal{M}$  satisfies  $\mathcal{I}$  with degree  $n$  if  $\mathcal{M}$  satisfies  $\mathcal{I}$  with a satisfaction relation  $\mathcal{R}$  such that each state  $t \in T$  is associated by  $\mathcal{R}$  to at most  $n$  states from  $S$  (i.e.,  $|\mathcal{R}(t)| \leq n$ );  $\mathcal{M}$  satisfies  $\mathcal{I}$  with the same structure as  $\mathcal{I}$  if  $\mathcal{M}$  satisfies  $\mathcal{I}$  with a satisfaction relation  $\mathcal{R}$  such that each state  $t \in T$  is associated to at most one state from  $S$  and each state  $s \in S$  is associated to at most one state from  $T$  (i.e.,  $|\mathcal{R}(t)| \leq 1$  for all  $t \in T$  and  $|\mathcal{R}^{-1}(s)| \leq 1$  for all  $s \in S$ ).

**Proposition 5.** *Let  $\mathcal{I}$  be an IMC. If a MC  $\mathcal{M}$  satisfies  $\mathcal{I}$  with degree  $n \in \mathbb{N}$  then there exists a MC  $\mathcal{M}'$  satisfying  $\mathcal{I}$  with degree 1 such that  $\mathcal{M}$  and  $\mathcal{M}'$  are bisimilar.*

The main idea for proving Proposition 5 is that if a MC  $\mathcal{M}$  with state space  $T$  satisfies an IMC  $\mathcal{I}$  with state space  $S$  according to a satisfaction relation  $\mathcal{R}$ , then each state  $t$  related by  $\mathcal{R}$  to many states  $s_1, \dots, s_n$  (with  $n > 1$ ) can be split in  $n$  states  $t^1, \dots, t^n$ . The derived MC  $\mathcal{M}'$  will satisfy  $\mathcal{I}$  with a satisfaction relation  $\mathcal{R}'$  where each  $t_i$  is only associated by  $\mathcal{R}'$  to the state  $s_i$  ( $i \leq n$ ). This  $\mathcal{M}'$  will be bisimilar to  $\mathcal{M}$  and it will satisfy  $\mathcal{I}$  with degree 1. Note that by construction the size of the resulting MC is in  $O(|\mathcal{M}| \times |\mathcal{I}|)$ .

**Proof of Proposition 5.** Let  $\mathcal{I} = (S, s_0, P, V^I)$  be an IMC and  $\mathcal{M} = (T, t_0, p, V)$  be a MC. If  $\mathcal{M}$  satisfies  $\mathcal{I}$  (with degree  $n$ ) then there exists a satisfaction relation  $\mathcal{R}$  verifying the  $\models_{\perp}^a$  satisfaction relation. For each association  $(t, s) \in \mathcal{R}$ , let  $\delta_{(t,s)}$  be the correspondence function chosen for this pair of states.  $\mathcal{M}$  satisfies  $\mathcal{I}$  with degree  $n$  means that each state in  $\mathcal{M}$  is associated by  $\mathcal{R}$  to at most  $n$  states in  $\mathcal{I}$ . To construct a MC  $\mathcal{M}'$  satisfying  $\mathcal{I}$  with degree 1 we create one state in  $\mathcal{M}'$  per association  $(t, s)$  in  $\mathcal{R}$ . Formally, let  $\mathcal{M}'$  be equal to  $(U, u_0, p', V')$  such that  $U = \{u_t^s \mid (t, s) \in \mathcal{R}\}$ ,  $u_0 = u_{t_0}^{s_0}$ ,  $V'$  is such that  $V'(u_t^s) = V(t)$ , and  $p'(u_t^s, u_{t'}^{s'}) = p(t, t') \times \delta_{(t,s)}(t')(s')$ . The following computation shows that the outgoing probabilities given by  $p'$  form a probability distribution for each state in  $\mathcal{M}'$ , and thus that  $\mathcal{M}'$  is a MC. Let  $u_t^s \in U$ .

$$\begin{aligned} \sum_{u_{t'}^{s'} \in U} p'(u_t^s, u_{t'}^{s'}) &= \sum_{(t', s') \in \mathcal{R}} p'(u_t^s, u_{t'}^{s'}) = \sum_{(t', s') \in \mathcal{R}} p(t, t') \times \delta_{(t,s)}(t')(s') \\ &= \sum_{t' \in T} \left( p(t, t') \times \sum_{s' \in S} \delta_{(t,s)}(t')(s') \right) = \sum_{t' \in T} p(t, t') \times 1 = 1 \end{aligned}$$

Finally, by construction of  $\mathcal{M}'$ , since  $\mathcal{M} \models_{\perp}^a \mathcal{I}$ , we get that  $\mathcal{R}' = \{(u_t^s, s) \mid t \in T, s \in S\}$  is a satisfaction relation between  $\mathcal{M}'$  and  $\mathcal{I}$ . Furthermore, for all  $u \in U$ ,  $|\{s \mid (u, s) \in \mathcal{R}'\}| \leq 1$ . Thus, we get that  $\mathcal{M}'$  satisfies  $\mathcal{I}$  with degree 1.

Consider the relation  $\mathcal{B}' = \{(u_t^s, t) \subseteq U \times T \mid (t, s) \in \mathcal{R}\}$ . Let  $\mathcal{B}$  be the closure of  $\mathcal{B}'$  by transitivity, reflexivity, and symmetry (i.e.,  $\mathcal{B}$  is the minimal equivalence relation based on  $\mathcal{B}'$ ). We prove that  $\mathcal{B}$  is a bisimulation relation between  $\mathcal{M}$  and  $\mathcal{M}'$ . By construction each equivalence class from  $\mathcal{B}$  contains exactly one state  $t$  from  $T$  and all states  $u_t^s$  such that  $(t, s) \in \mathcal{R}$ . Let  $(u_t^s, t)$  be in  $\mathcal{B}$ ,  $t'$  be a state in  $T$ , and  $B$  be the equivalence class from  $\mathcal{B}$  containing  $t'$  (i.e.,  $B$  is the set  $\{t'\} \cup \{u_{t'}^{s'} \in U \mid s' \in S \text{ and } (t', s') \in \mathcal{R}\}$ ). Firstly note that by construction the labels agree on  $u_t^s$  and  $t$ :  $V'(u_t^s) = V(t)$ . Secondly the following computation shows that  $p'(u_t^s, B \cap U)$  equals to  $p(t, B \cap T)$ , and thus that  $u_t^s$  and  $t$  are bisimilar:

$$\begin{aligned}
p'(u_t^s, B \cap U) &= \sum_{u_{t'}^{s'} \in B \cap U} p'(u_t^s, u_{t'}^{s'}) = \sum_{u_{t'}^{s'} \in B \cap U} p(t, t') \times \delta_{(t,s)}(t')(s') \\
&= \sum_{\{s' \in S \mid (s', t') \in \mathcal{R}\}} p(t, t') \times \delta_{(t,s)}(t')(s') = p(t, t') \times \sum_{\{s' \in S \mid (s', t') \in \mathcal{R}\}} \delta_{(t,s)}(t')(s') \\
&= p(t, t') \times 1 = p(t, \{t'\}) = p(t, B \cap T)
\end{aligned}$$

Moreover, we have by construction that  $(u_{t_0}^{s_0}, t_0) \in \mathcal{B}'$ , therefore  $\mathcal{M}'$  and  $\mathcal{M}$  are bisimilar.  $\square$

**Remark 2.** Note that whenever a MC satisfies an IMC with degree 1, all correspondence functions involved in the satisfaction relation are reduced to Dirac distributions, i.e., whenever  $\delta(t)(s) > 0$  for some  $s$ , we have  $\delta(t)(s) = 1$  and  $\delta(t)(s') = 0$  for all  $s' \neq s$ .

**Corollary 1.** Let  $\mathcal{I}$  be an IMC,  $\mathcal{M}$  be a MC satisfying  $\mathcal{I}$ , and  $\phi$  be a PCTL\* formula. There exists a MC  $\mathcal{M}'$  satisfying  $\mathcal{I}$  with degree 1 such that the probability  $\mathbb{P}^{\mathcal{M}'}(\phi)$  equals the probability  $\mathbb{P}^{\mathcal{M}}(\phi)$ .

Corollary 1 is derived from Proposition 5 joined with the probability preservation of the PCTL\* formulae on bisimilar Markov chains (see [17], Theorem 10.67, p. 813). Corollary 1 allows one to reduce to one the number of states in the pIMC  $\mathcal{I}$  satisfied by each state in the MC  $\mathcal{M}$  while preserving probabilities. Thus, one can construct from a MC  $\mathcal{M}$  satisfying an IMC  $\mathcal{I}$  another MC  $\mathcal{M}'$  satisfying the same IMC  $\mathcal{I}$  where the states in  $\mathcal{M}'$  are related to at most one state in  $\mathcal{I}$ . However, some states in  $\mathcal{I}$  may still be related to many states in  $\mathcal{M}'$ . The objective of Lemma 2 is to reduce these relations to an “at most one” in both directions ( $\mathcal{I}$  to  $\mathcal{M}'$  and  $\mathcal{M}'$  to  $\mathcal{I}$ ).

**Lemma 2.** Let  $\mathcal{I} = (S, s_0, P, V^I)$  be an IMC,  $\mathcal{M} = (T, t_0, p, V^{\mathcal{M}})$  be a MC satisfying  $\mathcal{I}$  with degree 1, and  $\Gamma \subseteq A$  be a proposition. If  $\mathcal{M}$  does not have the same structure as  $\mathcal{I}$  then there exists a MC  $\mathcal{M}_1 = (S_1, s_0^1, p_1, V_1^{\mathcal{M}_1})$  (resp.  $\mathcal{M}_2$ ) satisfying  $\mathcal{I}$  with degree 1, and such that  $S_1 \subseteq S$  and  $\mathbb{P}^{\mathcal{M}_1}(\diamond\Gamma) \leq \mathbb{P}^{\mathcal{M}}(\diamond\Gamma)$  (resp.  $S_2 \subseteq S$  and  $\mathbb{P}^{\mathcal{M}_2}(\diamond\Gamma) \geq \mathbb{P}^{\mathcal{M}}(\diamond\Gamma)$ ).

Lemma 2 reduces the number of states in  $\mathcal{M}$  while enhancing the maximal or minimal reachability probability. This lemma has a constructive proof. The main idea of the proof is that, whenever several states in  $\mathcal{M}$  satisfy the same state of  $\mathcal{I}$ , we only keep the state minimizing (or maximizing) the probability of eventually reaching  $\Gamma$ . All transitions leading to removed states are redirected to this “optimal” state. Since all removed states satisfy the same state in  $\mathcal{I}$  as the one we keep, satisfaction is preserved. Moreover, because the remaining states minimize (resp. maximize) the probability of eventually reaching  $\Gamma$ , the resulting MC verifies  $\mathbb{P}^{\mathcal{M}_1}(\diamond\Gamma) \leq \mathbb{P}^{\mathcal{M}}(\diamond\Gamma)$  (resp.  $\mathbb{P}^{\mathcal{M}_2}(\diamond\Gamma) \geq \mathbb{P}^{\mathcal{M}}(\diamond\Gamma)$ ). Before proving Lemma 2, we introduce some notations and Lemma 3 which will be used for proving Lemma 2.

Let  $S$  be a set of states. Consider two subsets  $S_1$  and  $S_2$  of  $S$ , a state  $s \in S$  and an integer  $n \geq 1$ . We write  $s \xrightarrow{n \times S_1} S_2$  for the set of finite state sequences initiating in  $s$ , ending when they first visit  $S_2$  (again, if  $s \in S_2$ ), and visiting  $S_1$  exactly  $n$  times. Formally,

$$s \xrightarrow{n \times S_1} S_2 = \{s_0, \dots, s_k \in S^* \mid k \geq 1, s_0 = s, s_k \in S_2, |\{i \in [0, k] \mid s_i \in S_1\}| = n, \text{ and } s_i \notin S_2 \text{ for all } 0 < i < k\}.$$

In the following, we slightly abuse notations and sometimes use state labels  $\Gamma \subseteq A$  to represent the set of states labelled with  $\Gamma$ .

**Lemma 3.** Let  $\mathcal{M} = (S, s_0, p, V)$  be a MC,  $\Gamma \subseteq A$  be a state label, and  $s$  be a state from  $S$  such that  $\mathbb{P}_s^{\mathcal{M}}(\diamond\Gamma) > 0$ . Then  $\mathbb{P}_s^{\mathcal{M}}(\diamond\Gamma) = 1$  if  $V(s) = \Gamma$  and

$$\mathbb{P}_s^{\mathcal{M}}(\diamond\Gamma) = \frac{\mathbb{P}_s^{\mathcal{M}}(s \xrightarrow{1 \times \{s\}} \Gamma)}{1 - \mathbb{P}_s^{\mathcal{M}}(s \xrightarrow{0 \times \Gamma} \{s\})} \text{ otherwise.}$$

**Proof.** We focus on the case where  $V(s) \neq \Gamma$ . First, we remark that the above quotient is well-defined. Indeed, since  $\mathbb{P}_s^{\mathcal{M}}(\diamond\Gamma) > 0$ , we have  $\mathbb{P}_s^{\mathcal{M}}(s \xrightarrow{0 \times \Gamma} \{s\}) \neq 1$  and therefore  $1 - \mathbb{P}_s^{\mathcal{M}}(s \xrightarrow{0 \times \Gamma} \{s\}) \neq 0$ .

By definition, we have

$$\mathbb{P}_s^{\mathcal{M}}(\diamond\Gamma) = \sum_{n=1}^{+\infty} \mathbb{P}_s^{\mathcal{M}}(s \xrightarrow{n \times \{s\}} \Gamma)$$

By construction of  $s \xrightarrow{n \times \{s\}} \Gamma$ , we have

$$\begin{aligned}
\mathbb{P}_s^{\mathcal{M}}(s \xrightarrow{n \times \{s\}} \Gamma) &= \mathbb{P}_s^{\mathcal{M}}(\{s_0, \dots, s_k \mid s_0 = s_k = s \text{ and } s_i \neq s \wedge V(s_i) \neq \Gamma, 0 < i < k\}) \times \mathbb{P}_s^{\mathcal{M}}(s \xrightarrow{(n-1) \times \{s\}} \Gamma) \\
&= \mathbb{P}_s^{\mathcal{M}}(s \xrightarrow{0 \times \Gamma} \{s\}) \times \mathbb{P}_s^{\mathcal{M}}(s \xrightarrow{(n-1) \times \{s\}} \Gamma)
\end{aligned}$$

As a consequence,  $\mathbb{P}_s^{\mathcal{M}}(\diamond\Gamma)$  is the sum of the terms of a geometric series of first term  $\mathbb{P}_s^{\mathcal{M}}(s \xrightarrow{1 \times \{s\}} \Gamma)$  and common ratio  $\mathbb{P}_s^{\mathcal{M}}(s \xrightarrow{0 \times \Gamma} \{s\})$ , and therefore

$$\mathbb{P}_s^{\mathcal{M}}(\diamond\Gamma) = \sum_{n=1}^{+\infty} \mathbb{P}_s^{\mathcal{M}}(s \xrightarrow{n \times \{s\}} \Gamma) = \frac{\mathbb{P}_s^{\mathcal{M}}(s \xrightarrow{1 \times \{s\}} \Gamma)}{1 - \mathbb{P}_s^{\mathcal{M}}(s \xrightarrow{0 \times \Gamma} \{s\})} \quad \square$$

**Proof of Lemma 2.** Let  $\mathcal{I} = (S, s_0, P, V^I)$  be an IMC and  $\mathcal{M} = (T, t_0, p, V^{\mathcal{M}})$  be a MC satisfying  $\mathcal{I}$  with degree 1. Let  $\mathcal{R}$  be the satisfaction relation between  $\mathcal{M}$  and  $\mathcal{I}$  with degree 1. The following proves in 4 steps that there exists  $\mathcal{M}_1$  satisfying  $\mathcal{I}$  with degree 1, with a set of states  $S_1$  such that  $S_1 \subseteq S$ , and such that  $\mathbb{P}^{\mathcal{M}_1}(\diamond\Gamma) \leq \mathbb{P}^{\mathcal{M}}(\diamond\Gamma)$ . In the first step, we build a new MC  $\mathcal{M}'$  that has fewer states than  $\mathcal{M}$ . In the second step, we show that  $\mathcal{M}'$  satisfies  $\mathcal{I}$  with degree 1. The third step is dedicated to showing that  $\mathbb{P}^{\mathcal{M}'}(\diamond\Gamma) \leq \mathbb{P}^{\mathcal{M}}(\diamond\Gamma)$ . Finally, the fourth step concludes by showing how to iterate this procedure in order to build  $\mathcal{M}_1$ .

1. We would like to construct a MC  $\mathcal{M}'$  satisfying  $\mathcal{I}$  with fewer states than  $\mathcal{M}$  such that  $\mathbb{P}^{\mathcal{M}'}(\diamond\Gamma) \leq \mathbb{P}^{\mathcal{M}}(\diamond\Gamma)$ . Since  $\mathcal{R}$  is of degree 1, each state  $t$  in  $T$  is associated to at most one state  $s$  in  $S$ . Furthermore, since  $\mathcal{M}$  does not have the same structure as  $\mathcal{I}$ , there exists at least one state  $s \in S$  such that  $|\mathcal{R}^{-1}(s)| \geq 2$ . Let  $\bar{s} \in S$  be such a state, and let  $T_{\bar{s}} = \{t_1, \dots, t_n\} = \mathcal{R}^{-1}(s)$ , with  $n \geq 2$ . Since this set is finite, there exists at least one state  $\bar{t} \in T_{\bar{s}}$  such that  $\mathbb{P}_{\bar{t}}^{\mathcal{M}}(\diamond\Gamma)$  is minimal among the states from  $T_{\bar{s}}$ . Let  $U = T_{\bar{s}} \setminus \{\bar{t}\}$ .

We build  $\mathcal{M}'$  from  $\mathcal{M}$  by replacing all transitions going to a state  $t \in U$  by transitions going to  $\bar{t}$  instead, and by removing all states from  $U$ . If the initial state was in  $T_{\bar{s}}$  and has been removed, then  $\bar{t}$  becomes the new initial state. Formally  $\mathcal{M}' = (T', t'_0, p', V')$  such that  $T' = (T \setminus U)$ ,  $V'$  is the restriction of  $V^{\mathcal{M}}$  on  $T'$ ,  $t'_0 = \bar{t}$  if  $t_0 \in T_{\bar{s}}$  and  $t'_0 = t_0$  otherwise, and for all  $t, t' \in T'$ :  $p'(t, t') = p(t, t')$  if  $t' \neq \bar{t}$  and  $p'(t, \bar{t}) = \sum_{t' \in T_{\bar{s}}} p(t, t')$ . The following computation shows that

the outgoing probabilities given by  $p'$  in  $\mathcal{M}'$  are well defined (i.e. form correct probability distributions).

$$\begin{aligned} \sum_{t' \in T'} p'(t, t') &= \sum_{t' \in T' \setminus \{\bar{t}\}} p'(t, t') + p'(t, \bar{t}) \\ &\stackrel{(1)}{=} \sum_{t' \in T \setminus T_{\bar{s}}} p(t, t') + \sum_{t' \in T_{\bar{s}}} p(t, t') \\ &= \sum_{t' \in T \setminus T_{\bar{s}} \cup T_{\bar{s}}} p(t, t') = \sum_{t' \in T} p(t, t') = 1 \end{aligned}$$

Note that the first step comes from the definition of  $p'$  with respect to  $p$ .

2. We now prove that  $\mathcal{M}'$  satisfies  $\mathcal{I}$  with degree 1.

Let  $\mathcal{R}' \subseteq T' \times S$  be the restriction of  $\mathcal{R}$  to  $T'$ . First, observe that since  $\mathcal{R}$  is of degree 1, so is  $\mathcal{R}'$ . We now prove that  $\mathcal{R}'$  is a satisfaction relation between  $\mathcal{M}'$  and  $\mathcal{I}$ . For each pair  $(t, s) \in \mathcal{R}$  let  $\delta_{(t,s)}$  be the correspondence function given by the satisfaction relation  $\mathcal{R}$ . Recall that, as said in Remark 2, all correspondence functions  $\delta_{(t,s)}$  are reduced to Dirac distributions.

Let  $(t, s)$  be in  $\mathcal{R}'$  and  $\delta'_{(t,s)}: T' \rightarrow (S \rightarrow [0, 1])$  be such that  $\delta'_{(t,s)}(t')(s') = \delta_{(t,s)}(t')(s')$  if  $t' \neq \bar{t}$ , and  $\delta'_{(t,s)}(\bar{t})(s') = 1$  if  $s' = \bar{s}$  and 0 otherwise.  $\delta'_{(t,s)}$  is a correspondence function for the pair  $(t, s)$  in  $\mathcal{R}'$  such as required by the  $\models_{\perp}^a$  satisfaction relation:

- (a) Let  $t'$  be in  $T'$  such that  $p'(t, t') > 0$ . If  $t' \neq \bar{t}$  then  $\delta'_{(t,s)}(t')$  is equivalent to  $\delta_{(t,s)}(t')$ , and since  $p'(t, t') = p(t, t') > 0$  in this case,  $\delta_{(t,s)}(t')$  is a distribution on  $S$  by construction. Otherwise,  $t' = \bar{t}$  and by construction  $\delta'_{(t,s)}(\bar{t})$  is a Dirac distribution on  $S$ .
- (b) Let  $s'$  be in  $S$ . We show that  $\sum_{t' \in T'} p'(t, t') \times \delta'_{(t,s)}(t')(s') \in P(s, s')$ .

First remark that since  $\mathcal{R}$  is of degree 1 and  $T_{\bar{s}} = \mathcal{R}^{-1}(\bar{s})$ , then for all  $t' \in T_{\bar{s}}$ , we have  $\delta_{(t,s)}(t')(s') = 0$  for all  $s' \neq \bar{s}$ , and  $\delta_{(t,s)}(t')(\bar{s}) = 1$  whenever  $p(t, t') > 0$ .

As a consequence,  $p'(t, \bar{t}) \times \delta'_{(t,s)}(\bar{t})(s') = \sum_{t' \in T_{\bar{s}}} (p(t, t') \times \delta_{(t,s)}(t')(s'))$  for all  $s'$ .

Therefore, we have the following computation.

$$\begin{aligned} &\sum_{t' \in T'} p'(t, t') \times \delta'_{(t,s)}(t')(s') \\ &= \sum_{t' \in T' \setminus \{\bar{t}\}} p'(t, t') \times \delta'_{(t,s)}(t')(s') + p'(t, \bar{t}) \times \delta'_{(t,s)}(\bar{t})(s') \end{aligned}$$

$$\begin{aligned}
&= \sum_{t' \in T \setminus T_{\bar{s}}} p(t, t') \times \delta_{(t,s)}(t')(s') + \sum_{t' \in T_{\bar{s}}} p(t, t') \times \delta_{(t,s)}(t')(s') \\
&= \sum_{t' \in T} p(t, t') \times \delta_{(t,s)}(t')(s') \\
&\in P(s, s') \text{ by definition of } \delta_{(t,s)}
\end{aligned}$$

(c) Let  $t'$  be in  $T'$  and  $s'$  be in  $S$ . We have by construction of  $\mathcal{R}'$  from  $\mathcal{R}$  that if  $\delta'_{(t,s)}(t')(s') > 0$  then  $(t', s') \in \mathcal{R}$ .

Therefore,  $\mathcal{R}'$  is a satisfaction relation of degree 1 between  $\mathcal{M}$  and  $\mathcal{I}$ . Moreover, since  $t'_0$  is either  $t_0$  if  $t_0 \notin T_{\bar{s}}$  or  $\bar{t}$  otherwise, we have  $(t'_0, s_0) \in \mathcal{R}'$ , which allows to conclude.

3. We now prove that the probability of reaching  $\Gamma$  from  $\bar{t}$  is lower in  $\mathcal{M}'$  than in  $\mathcal{M}$ . We consider the MC  $\mathcal{M}''$ , obtained from  $\mathcal{M}$  by replacing all states labelled with  $\Gamma$  by absorbing states. Formally  $\mathcal{M}'' = (T, t_0, p'', V^{\mathcal{M}'})$  such that for all  $t, t' \in T$ :  $p''(t, t') = p(t, t')$  if  $V^{\mathcal{M}'}(t) \neq \Gamma$ ,  $p''(t, t') = 1$  if  $V^{\mathcal{M}'}(t) = \Gamma$  and  $t = t'$ , and  $p''(t, t') = 0$  otherwise. By definition of the reachability property we get that  $\mathbb{P}_{\bar{t}}^{\mathcal{M}''}(\diamond\Gamma)$  equals to  $\mathbb{P}_{\bar{t}}^{\mathcal{M}}(\diamond\Gamma)$  for all states  $t$  in  $T$ .

If  $V^{\mathcal{M}}(\bar{t}) = \Gamma$ , then we trivially have  $\mathbb{P}_{\bar{t}}^{\mathcal{M}'}(\diamond\Gamma) = \mathbb{P}_{\bar{t}}^{\mathcal{M}}(\diamond\Gamma) = 1$ . Otherwise, if  $\mathbb{P}_{\bar{t}}^{\mathcal{M}'}(\diamond\Gamma) = 0$  then trivially  $\mathbb{P}_{\bar{t}}^{\mathcal{M}'}(\diamond\Gamma) \leq \mathbb{P}_{\bar{t}}^{\mathcal{M}}(\diamond\Gamma)$ . We therefore assume that  $\mathbb{P}_{\bar{t}}^{\mathcal{M}'}(\diamond\Gamma) > 0$  and  $V^{\mathcal{M}}(\bar{t}) \neq \Gamma$ . The following computation concludes.

From Lemma 3, we have

$$\mathbb{P}_{\bar{t}}^{\mathcal{M}'}(\diamond\Gamma) \stackrel{(1)}{=} \frac{\mathbb{P}_{\bar{t}}^{\mathcal{M}'}(\bar{t} \xrightarrow{1 \times \{\bar{t}\}} \Gamma)}{1 - \mathbb{P}_{\bar{t}}^{\mathcal{M}'}(\bar{t} \xrightarrow{0 \times \Gamma} \{\bar{t}\})}$$

By construction of  $\mathcal{M}'$  from  $\mathcal{M}$ , we obtain

$$\stackrel{(2)}{=} \frac{\mathbb{P}_{\bar{t}}^{\mathcal{M}}(\bar{t} \xrightarrow{1 \times T_{\bar{s}}} \Gamma)}{1 - \mathbb{P}_{\bar{t}}^{\mathcal{M}}(\bar{t} \xrightarrow{0 \times \Gamma} T_{\bar{s}})}$$

Therefore, by construction of  $\mathcal{M}''$  from  $\mathcal{M}$  where states labelled with  $\Gamma$  are absorbing states,

$$\stackrel{(3)}{=} \frac{\mathbb{P}_{\bar{t}}^{\mathcal{M}''}(\bar{t} \xrightarrow{1 \times T_{\bar{s}}} \Gamma)}{1 - \mathbb{P}_{\bar{t}}^{\mathcal{M}''}(\bar{t} \xrightarrow{0 \times \Gamma} T_{\bar{s}})}$$

Since  $\mathbb{P}_{\bar{t}}^{\mathcal{M}''}(\diamond\Gamma)$  is equal to  $\mathbb{P}_{\bar{t}}^{\mathcal{M}''}(\bar{t} \xrightarrow{1 \times T_{\bar{s}}} \Gamma) + \sum_{t' \in T_{\bar{s}}} (\mathbb{P}_{\bar{t}}^{\mathcal{M}''}(\bar{t} \xrightarrow{2 \times T_{\bar{s}}} \{t'\}) \times \mathbb{P}_{t'}^{\mathcal{M}''}(\diamond\Gamma))$  (recall that states labelled with  $\Gamma$  are absorbing in  $\mathcal{M}''$ ), we have

$$\stackrel{(4)}{=} \frac{\mathbb{P}_{\bar{t}}^{\mathcal{M}''}(\diamond\Gamma) - \sum_{t' \in T_{\bar{s}}} (\mathbb{P}_{\bar{t}}^{\mathcal{M}''}(\bar{t} \xrightarrow{2 \times T_{\bar{s}}} \{t'\}) \times \mathbb{P}_{t'}^{\mathcal{M}''}(\diamond\Gamma))}{1 - \mathbb{P}_{\bar{t}}^{\mathcal{M}''}(\bar{t} \xrightarrow{0 \times \Gamma} T_{\bar{s}})}$$

Moreover,  $\mathbb{P}_{t'}^{\mathcal{M}}(\diamond\Gamma) \geq \mathbb{P}_{t'}^{\mathcal{M}'}(\diamond\Gamma)$  for all  $t' \in T_{\bar{s}}$ , which is also correct in  $\mathcal{M}''$  by construction. Therefore,

$$\begin{aligned}
&\stackrel{(5)}{\leq} \frac{\mathbb{P}_{\bar{t}}^{\mathcal{M}''}(\diamond\Gamma) - \sum_{t' \in T_{\bar{s}}} (\mathbb{P}_{\bar{t}}^{\mathcal{M}''}(\bar{t} \xrightarrow{2 \times T_{\bar{s}}} \{t'\}) \times \mathbb{P}_{t'}^{\mathcal{M}''}(\diamond\Gamma))}{1 - \mathbb{P}_{\bar{t}}^{\mathcal{M}''}(\bar{t} \xrightarrow{0 \times \Gamma} T_{\bar{s}})} \\
&= \frac{\mathbb{P}_{\bar{t}}^{\mathcal{M}''}(\diamond\Gamma) \times (1 - \sum_{t' \in T_{\bar{s}}} \mathbb{P}_{\bar{t}}^{\mathcal{M}''}(\bar{t} \xrightarrow{2 \times T_{\bar{s}}} \{t'\}))}{1 - \mathbb{P}_{\bar{t}}^{\mathcal{M}''}(\bar{t} \xrightarrow{0 \times \Gamma} T_{\bar{s}})}
\end{aligned}$$

Finally, we have  $\sum_{t' \in T_{\bar{s}}} \mathbb{P}_{\bar{t}}^{\mathcal{M}''}(\bar{t} \xrightarrow{2 \times T_{\bar{s}}} \{t'\}) = \mathbb{P}_{\bar{t}}^{\mathcal{M}''}(\bar{t} \xrightarrow{0 \times \Gamma} T_{\bar{s}})$ . Indeed  $\bar{t} \xrightarrow{0 \times \Gamma} T_{\bar{s}}$  represents all finite state sequences starting in  $\bar{t}$  and ending in  $T_{\bar{s}}$  without visiting  $\Gamma$  or  $T_{\bar{s}}$  inbetween. Therefore, since states labelled with  $\Gamma$  are absorbing in  $\mathcal{M}''$ , we have  $\bar{t} \xrightarrow{0 \times \Gamma} T_{\bar{s}} = \cup_{t' \in T_{\bar{s}}} \bar{t} \xrightarrow{2 \times T_{\bar{s}}} \{t'\}$ . As a consequence,

$$\begin{aligned}
&\stackrel{(6)}{=} \frac{\mathbb{P}_{\bar{t}}^{\mathcal{M}''}(\diamond\Gamma) \times (1 - \mathbb{P}_{\bar{t}}^{\mathcal{M}''}(\bar{t} \xrightarrow{0 \times \Gamma} T_{\bar{s}}))}{1 - \mathbb{P}_{\bar{t}}^{\mathcal{M}''}(\bar{t} \xrightarrow{0 \times \Gamma} T_{\bar{s}})} \\
&= \mathbb{P}_{\bar{t}}^{\mathcal{M}''}(\diamond\Gamma) \\
&= \mathbb{P}_{\bar{t}}^{\mathcal{M}}(\diamond\Gamma)
\end{aligned}$$



From all states in  $t' \in T' \setminus \bar{t}$ , we trivially have

$$\mathbb{P}_{t'}^{\mathcal{M}'}(t' \xrightarrow{0 \times \{\bar{t}\}} \Gamma) = \mathbb{P}_{t'}^{\mathcal{M}}(t' \xrightarrow{0 \times T_{\bar{s}}} \Gamma).$$

Moreover,

$$\mathbb{P}_{t'}^{\mathcal{M}'}(t' \xrightarrow{0 \times \Gamma} \bar{t}) = \mathbb{P}_{t'}^{\mathcal{M}}(t' \xrightarrow{0 \times \Gamma} T_{\bar{s}}).$$

Finally, observe that

$$\mathbb{P}_{t'}^{\mathcal{M}'}(\diamond \Gamma) = \mathbb{P}_{t'}^{\mathcal{M}'}(t' \xrightarrow{0 \times \{\bar{t}\}} \Gamma) + \mathbb{P}_{t'}^{\mathcal{M}'}(t' \xrightarrow{0 \times \Gamma} \bar{t}) \times \mathbb{P}_{\bar{t}}^{\mathcal{M}'}(\diamond \Gamma).$$

As a consequence, we clearly have  $\mathbb{P}_{t'}^{\mathcal{M}'}(\diamond \Gamma) \leq \mathbb{P}_{t'}^{\mathcal{M}}(\diamond \Gamma)$  for all  $t' \in T'$ .

4. In order to conclude the proof, the above procedure has to be repeated for all states  $\bar{s} \in S$  such that  $|\mathcal{R}^{-1}(\bar{s})| \geq 2$ , and then rename the states  $\bar{t}$  to the corresponding  $\bar{s}$ . Since there are finitely many such classes and since the procedure removes one of them each time it is applied, this converges to a MC  $\mathcal{M}_1$  that satisfies the desired properties (and in particular  $S_1 \subseteq S$ ).

The same method can be used for building a MC  $\mathcal{M}_2$  satisfying  $\mathcal{I}$  with degree 1, and such that  $S_2 \subseteq S$  and  $\mathbb{P}^{\mathcal{M}_2}(\diamond \Gamma) \geq \mathbb{P}^{\mathcal{M}}(\diamond \Gamma)$ . In this case, some care has to be taken in order to choose the state  $\bar{t}$  that is selected among the set  $T_{\bar{s}} = \mathcal{R}^{-1}(\bar{s})$ . Indeed, it does not suffice anymore to select any  $\bar{t}$  such that  $\mathbb{P}_{\bar{t}}^{\mathcal{M}}(\diamond \Gamma)$  is maximal. In order to ensure that Lemma 3 can be applied, we need to have  $\mathbb{P}_{\bar{t}}^{\mathcal{M}'}(\diamond \Gamma) > 0$ , which could be wrong for instance if all direct successors of  $\bar{t}$  in  $\mathcal{M}$  are in  $T_{\bar{s}}$  (even if  $\bar{t}$  is one of the states with maximal probability of eventually reaching  $\Gamma$ ).

Luckily, whenever there is at least one state  $t \in T_{\bar{s}}$  such that  $\mathbb{P}_t^{\mathcal{M}}(\diamond \Gamma) > 0$ , a simple graph analysis can show that, among all states  $T_{\bar{s}}^{\max} \subseteq T_{\bar{s}}$  that have maximal reachability of eventually reaching  $\Gamma$ , there must be at least one state  $\bar{t}$  such that the probability of reaching  $\Gamma$  from  $\bar{t}$  without visiting  $T_{\bar{s}}$  is strictly positive. Formally if there exists  $t \in T_{\bar{s}}$  such that  $\mathbb{P}_t^{\mathcal{M}}(\diamond \Gamma) > 0$ , then  $|\{\bar{t} \in T_{\bar{s}} \mid \mathbb{P}_{\bar{t}}^{\mathcal{M}}(\bar{t} \xrightarrow{1 \times T_{\bar{s}}} \Gamma) > 0 \text{ and } \mathbb{P}_{\bar{t}}^{\mathcal{M}}(\diamond \Gamma) \geq \mathbb{P}_{t'}^{\mathcal{M}}(\diamond \Gamma) \forall t' \in T_{\bar{s}}\}| \geq 1$ .

Choosing such a state  $\bar{t}$  therefore ensures that Lemma 3 can be used and the rest of the proof follows directly by replacing all occurrences of “ $\leq$ ” with “ $\geq$ ”.  $\square$

Next, Lemma 4 is a consequence of Corollary 1 and Lemma 2 and states that the maximal and the minimal probability of reaching a given state label is realized by Markov chains with the same structure as the IMC.

**Lemma 4.** *Let  $\mathcal{I} = (S, s_0, P, V)$  be an IMC,  $\mathcal{M}$  be a MC satisfying  $\mathcal{I}$  w.r.t.  $\models_{\perp}^a$ , and  $\Gamma \subseteq A$  be a state label. There exist MCs  $\mathcal{M}_1$  and  $\mathcal{M}_2$  satisfying  $\mathcal{I}$  w.r.t.  $\models_{\perp}^o$  such that  $\mathbb{P}^{\mathcal{M}_1}(\diamond \Gamma) \leq \mathbb{P}^{\mathcal{M}}(\diamond \Gamma) \leq \mathbb{P}^{\mathcal{M}_2}(\diamond \Gamma)$ .*

**Proof.** Let  $\mathcal{I}$  be an IMC and  $\mathcal{M}$  be a MC satisfying  $\mathcal{I}$  w.r.t.  $\models_{\perp}^a$ . Let  $\mathcal{M}'$  be the MC satisfying  $\mathcal{I}$  with degree 1 given by Corollary 1.

Consider the MCs  $\mathcal{M}_1$  and  $\mathcal{M}_2$  given by applying Lemma 2 to  $\mathcal{M}'$ . By construction, we have  $\mathcal{M}_1 \models_{\perp}^a \mathcal{I}$  with degree 1,  $\mathbb{P}^{\mathcal{M}_1}(\diamond \Gamma) \leq \mathbb{P}^{\mathcal{M}'}(\diamond \Gamma)$  and  $\mathbb{P}^{\mathcal{M}'}(\diamond \Gamma) = \mathbb{P}^{\mathcal{M}}(\diamond \Gamma)$ , therefore  $\mathbb{P}^{\mathcal{M}_1}(\diamond \Gamma) \leq \mathbb{P}^{\mathcal{M}}(\diamond \Gamma)$ . Moreover, since  $S_1 \subseteq S$  and  $\mathcal{M}_1 \models_{\perp}^a \mathcal{I}$ , it directly follows that  $\mathcal{M}_1 \models_{\perp}^o \mathcal{I}$ .

The same reasoning applies to  $\mathcal{M}_2$ , concluding the proof.  $\square$

Finally, the following proves our Theorem 1 using Lemma 4 and Proposition 1.

**Proof of Theorem 1.** Let  $\mathcal{I} = (S, s_0, P, V)$  be an IMC,  $\Gamma \subseteq A$  be a state label,  $\sim \in \{\leq, <, >, \geq\}$ , and  $0 < p < 1$ . Recall that according to an IMC satisfaction relation the property  $\mathbb{P}^{\mathcal{I}}(\diamond \Gamma) \sim p$  holds if and only if there exists a MC  $\mathcal{M}$  satisfying  $\mathcal{I}$  (with the chosen semantics) such that  $\mathbb{P}^{\mathcal{M}}(\diamond \Gamma) \sim p$ .

1. We first prove the equivalence w.r.t.  $\models_{\perp}^o$  and  $\models_{\perp}^a$ .

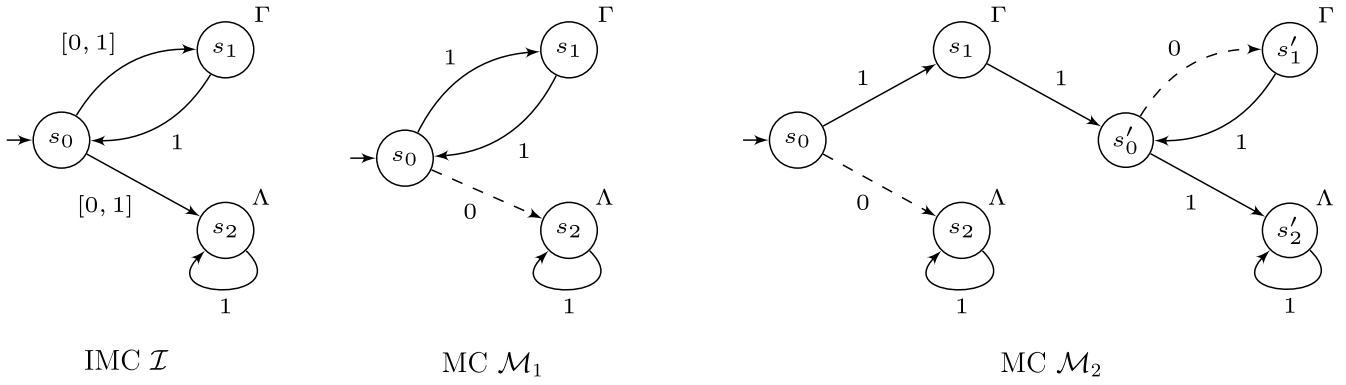
[ $\Rightarrow$ ] Since  $\models_{\perp}^a$  is more general than  $\models_{\perp}^o$ , for all MCs  $\mathcal{M}$ , if  $\mathcal{M} \models_{\perp}^o \mathcal{I}$  then  $\mathcal{M} \models_{\perp}^a \mathcal{I}$  (Proposition 1).

[ $\Leftarrow$ ] Since  $\mathbb{P}^{\mathcal{I}}(\diamond \Gamma) \sim p$  with the at-every-step semantics implies that there exists a MC  $\mathcal{M}$  such that  $\mathcal{M} \models_{\perp}^a \mathcal{I}$  and  $\mathbb{P}^{\mathcal{M}}(\diamond \Gamma) \sim p$ . Thus by Lemma 4 we get that there exists a MC  $\mathcal{M}'$  such that  $\mathcal{M}' \models_{\perp}^o \mathcal{I}$  and  $\mathbb{P}^{\mathcal{M}'}(\diamond \Gamma) \sim p$ .

2. We now prove the equivalence w.r.t.  $\models_{\perp}^o$  and  $\models_{\perp}^d$ .

[ $\Rightarrow$ ] Direct from the fact that  $\models_{\perp}^d$  is more general than  $\models_{\perp}^o$  (Proposition 1).

[ $\Leftarrow$ ] Since  $\models_{\perp}^a$  is more general than  $\models_{\perp}^d$ ,  $\mathbb{P}^{\mathcal{I}}(\diamond \Gamma) \sim p$  with the IMDP semantics implies that  $\mathbb{P}^{\mathcal{I}}(\diamond \Gamma) \sim p$  with the at-every-step semantics. Thus by Lemma 4 we get that there exists a MC  $\mathcal{M}'$  such that  $\mathcal{M}' \models_{\perp}^o \mathcal{I}$  and  $\mathbb{P}^{\mathcal{M}'}(\diamond \Gamma) \sim p$ .  $\square$



**Fig. 12.** IMC  $\mathcal{I}$ , MC  $\mathcal{M}_1$ , and MC  $\mathcal{M}_2$  such that property  $\mathbb{P}_{=1}(X(\Gamma \wedge \mathbb{P}_{=1}(\diamond \Lambda)))$  holds for  $\mathcal{M}_2$ , does not hold for  $\mathcal{M}_1$ , holds for  $\mathcal{I}$  w.r.t.  $\models_{\perp}^a$ , and does not hold for  $\mathcal{I}$  w.r.t.  $\models_{\perp}^o$ .

Thus, by Theorem 1 the three semantics  $\models_{\perp}^o$ ,  $\models_{\perp}^d$ , and  $\models_{\perp}^a$  are equivalent with respect to the minimal and the maximal quantitative reachability properties. However, note that the equivalence of reachability properties does not trivially extend to more general properties. For instance, consider the formula  $\mathbb{P}_{=1}(X(\Gamma \wedge \mathbb{P}_{=1}(\diamond \Lambda)))$ , where  $\Gamma$  and  $\Lambda$  are two state labels. This property states that the probability of visiting a state labelled with  $\Gamma$  in the next step and almost certainly reaching a state labelled with  $\Lambda$  afterwards is 1. Fig. 12 contains an IMC  $\mathcal{I}$  such that this property does not hold w.r.t.  $\models_{\perp}^o$  while it holds w.r.t.  $\models_{\perp}^a$ . Indeed, under the  $\models_{\perp}^o$  semantics, enforcing that the probability of visiting a state labelled with  $\Gamma$  as the next state is 1 imposes that the branch leading to  $\Lambda$  is given a probability 0, hence preventing from almost certainly visiting a state labelled with  $\Lambda$  afterwards (cf. the MC  $\mathcal{M}_1$  in Fig. 12). On the contrary, the  $\models_{\perp}^a$  semantics allows one to unfold the structure, therefore to first assign a probability 1 to the transition leading to  $\Gamma$  and then give a probability 1 to the transition leading to  $\Lambda$  (cf. the MC  $\mathcal{M}_2$  in Fig. 12). Note that the same would be possible under the  $\models_{\perp}^d$  semantics.

#### 4.2. Constraint encodings

Note that the result from Theorem 1 naturally extends to pIMCs. We therefore exploit this result to construct a CSP encoding for verifying quantitative reachability properties in pIMCs. As in Section 3, we extend the CSP  $\mathbf{C}_{\exists\mathbf{r}}$ , that produces a correct MC implementation for the given pIMC, by imposing that this MC implementation satisfies the given quantitative reachability property. In order to compute the probability of reaching state label  $\Gamma$  at the MC level, we use standard techniques from [17] that require the partitioning of the state space into three sets  $S_{=1}$ ,  $S_{=0}$ , and  $S_{?}$  that correspond to a subset of states reaching  $\Gamma$  with probability 1, a subset of states from which  $\Gamma$  cannot be reached, and the remaining states, respectively. Once this partition is chosen, the reachability probabilities of all states in  $S_{?}$  are computed as the unique solution of a linear equation system (see [17], Theorem 10.19, p. 766). We now explain how we identify states from  $S_{=1}$ ,  $S_{=0}$  and  $S_{?}$ , and how we encode the linear equation system, which leads to the resolution of quantitative reachability.

Let  $\mathcal{P} = (S, s_0, P, V, Y)$  be a pIMC and  $\Gamma \subseteq A$  be a state label. We start by setting  $S_{=1} = \{s \mid V(s) = \Gamma\}$ . We then extend  $\mathbf{C}_{\exists\mathbf{r}}(\mathcal{P})$  in order to identify the set  $S_{=0}$ . Let  $\mathbf{C}'_{\exists\mathbf{r}}(\mathcal{P}, \Gamma) = (X \cup X', D \cup D', C \cup C')$  be such that  $(X, D, C) = \mathbf{C}_{\exists\mathbf{r}}(\mathcal{P})$ .  $X'$  contains one Boolean variable  $\lambda_s$  and one integer variable  $\alpha_s$  with domain  $[0, |S|]$  per state  $s$  in  $S$ ,  $D'$  contains the domains of these variables, and  $C'$  is composed of the following constraints for each state  $s \in S$ :

- (11)  $\alpha_s = 1$ , if  $\Gamma = V(s)$
- (12)  $\alpha_s \neq 1$ , if  $\Gamma \neq V(s)$
- (13)  $\lambda_s \Leftrightarrow (\rho_s \wedge (\alpha_s \neq 0))$
- (14)  $\alpha_s > 1 \Rightarrow \bigvee_{s' \in \text{succ}(s) \setminus \{s\}} (\alpha_s = \alpha_{s'} + 1) \wedge (\theta_s^{s'} > 0)$ , if  $\Gamma \neq V(s)$
- (15)  $\alpha_s = 0 \Leftrightarrow \bigwedge_{s' \in \text{succ}(s) \setminus \{s\}} (\alpha_{s'} = 0) \vee (\theta_s^{s'} = 0)$ , if  $\Gamma \neq V(s)$

Note that variables  $\alpha_s$  play a symmetric role to variables  $\omega_s$  from  $\mathbf{C}_{\exists\mathbf{r}}$ : instead of indicating the existence of a path from  $s_0$  to  $s$ , they characterize the existence of a path from  $s$  to a state labelled with  $\Gamma$ . In addition, due to constraint (13), variables  $\lambda_s$  are set to true if and only if there exists a path with non zero probability from the initial state  $s_0$  to a state labelled with  $\Gamma$  visiting  $s$ . Thus,  $\Gamma$  cannot be reached from states such that  $\lambda_s = \text{false}$ . Therefore,  $S_{=0} = \{s \mid \lambda_s = \text{false}\}$ , which is formalized in Proposition 6.

**Proposition 6.** Let  $\mathcal{P} = (S, s_0, P, V, Y)$  be a pIMC and  $\Gamma \subseteq A$  be a state label. There exists a MC  $\mathcal{M} \models_{\perp}^a \mathcal{P}$  if and only if there exists a valuation  $v$  solution of the CSP  $\mathbf{C}'_{\exists\mathbf{r}}(\mathcal{P}, \Gamma)$  such that for each state  $s \in S$ :  $v(\lambda_s)$  is equal to true if and only if  $\mathbb{P}_s^{\mathcal{M}}(\diamond \Gamma) \neq 0$  and  $s$  is reachable from  $s_0$ .

**Proof.** Constraints (1)–(10) given before ensure that  $\rho_s$  is true if and only if  $s$  is reachable from  $s_0$ . Moreover, by constraints (11)–(12) and (14)–(15),  $\alpha_s > 0$  if and only if  $\Gamma$  is reachable from  $s$ . Therefore, by constraint (13),  $\lambda_s$  is true if and only if  $\mathbb{P}_s^{\mathcal{M}}(\diamond\Gamma) \neq 0$  and  $s$  is reachable from  $s_0$ .  $\square$

Finally, we encode the equation system from [17] in a last CSP encoding that extends  $\mathbf{C}'_{\exists\Gamma}$ . Let  $\mathbf{C}_{\exists\Gamma}(\mathcal{P}, \Gamma) = (X \cup X', D \cup D', C \cup C')$  be such that  $(X, D, C) = \mathbf{C}'_{\exists\Gamma}(\mathcal{P}, \Gamma)$ .  $X'$  contains one variable  $\gamma_s$  per state  $s$  in  $S$  with domain  $[0, 1]$ ,  $D'$  contains the domains of these variables, and  $C'$  is composed of the following constraints for each state  $s \in S$ :

- (16)  $\neg\lambda_s \Rightarrow \gamma_s = 0$   
(17)  $\lambda_s \Rightarrow \gamma_s = 1$ , if  $\Gamma = V(s)$   
(18)  $\lambda_s \Rightarrow \gamma_s = \sum_{s' \in \text{Succ}(s)} \gamma_{s'} \theta_s^{s'}$ , if  $\Gamma \neq V(s)$

As a consequence, variables  $\gamma_s$  encode the probability with which state  $s$  eventually reaches  $\Gamma$  when  $s$  is reachable from the initial state and 0 otherwise.

**Proposition 7.** Let  $\mathcal{P} = (S, s_0, P, V, Y)$  be a pIMC and  $\Gamma \subseteq A$  be a state label. There exists a MC  $\mathcal{M} \models_{\text{pI}}^a \mathcal{P}$  if and only if there exists a valuation  $v$  solution of the CSP  $\mathbf{C}_{\exists\Gamma}(\mathcal{P}, \Gamma)$  such that  $v(\gamma_s)$  is equal to  $\mathbb{P}_s^{\mathcal{M}}(\diamond\Gamma)$  if  $s$  is reachable from the initial state  $s_0$  in  $\mathcal{M}$ , and is equal to 0 otherwise.

**Proof.** Let  $\mathcal{P} = (S, s_0, P, V, Y)$  be a pIMC and  $\Gamma \subseteq A$  be a state label.  $\mathbf{C}_{\exists\Gamma}$  extends the CSP  $\mathbf{C}'_{\exists\Gamma}$  that produces a MC  $\mathcal{M}$  satisfying  $\mathcal{P}$  (cf. Proposition 6) by computing the probability of reaching  $\Gamma$  in  $\mathcal{M}$ . The encoding proposed above ensures the partitioning of the state space  $S$  into three sets  $S_{=1}$ ,  $S_{=0}$ , and  $S_{?}$  that correspond to a subset of states reaching  $\Gamma$  with probability 1, a subset of states from which  $\Gamma$  cannot be reached, and the remaining states, respectively. Recall that for each state  $s \in S$  variable  $\lambda_s$  is equal to true if and only if  $s$  is reachable in  $\mathcal{M}$  and  $s$  can reach  $\Gamma$  with a non zero probability. States that are not reachable from  $s_0$  do not contribute to the overall probability of reaching  $\Gamma$ . It is therefore safe to place them in  $S_{=0}$ , although they might have a positive probability of reaching  $\Gamma$ . Thus we consider  $S_{=0} = \{s \mid \lambda_s = \text{false}\}$ ,  $S_{=1} = \{s \mid V(s) = \Gamma\}$ , and  $S_{?} = S \setminus (S_{=0} \cup S_{=1})$ . Finally constraints in  $\mathbf{C}_{\exists\Gamma}$  encode the equation system from [17] (Theorem 10.19, p. 766) according to the chosen sets  $S_{=0}$ ,  $S_{=1}$ , and  $S_{?}$ . Thus,  $\gamma_{s_0}$  equals the probability in  $\mathcal{M}$  to reach  $\Gamma$ .  $\square$

Let  $p \in [0, 1] \subseteq \mathbb{R}$  be a probability bound. Adding the constraint  $\gamma_{s_0} \sim p$  to the previous  $\mathbf{C}_{\exists\Gamma}$  encoding allows one to determine if there exists a MC  $\mathcal{M} \models_{\text{pI}}^a \mathcal{P}$  such that  $\mathbb{P}^{\mathcal{M}}(\diamond\Gamma) \sim p$ . Formally, let  $\sim \in \{\leq, <, \geq, >\}$  be a comparison operator, we write  $\approx$  for its negation (e.g.,  $\not\leq$  is  $>$ ). This leads to the following theorem.

**Theorem 2.** Let  $\mathcal{P} = (S, s_0, P, V, Y)$  be a pIMC,  $\Gamma \subseteq A$  be a label,  $p \in [0, 1]$ ,  $\sim \in \{\leq, <, \geq, >\}$  be a comparison operator, and  $(X, D, C)$  be  $\mathbf{C}_{\exists\Gamma}(\mathcal{P}, \Gamma)$ :

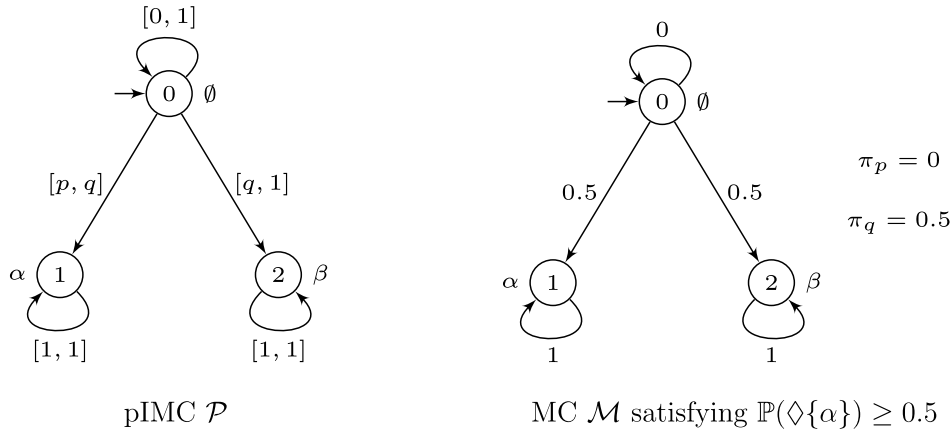
- CSP  $(X, D, C \cup (\gamma_{s_0} \sim p))$  is satisfiable if and only if there exists  $\mathcal{M} \models_{\text{pI}}^a \mathcal{P}$  such that  $\mathbb{P}^{\mathcal{M}}(\diamond\Gamma) \sim p$ .
- CSP  $(X, D, C \cup (\gamma_{s_0} \approx p))$  is unsatisfiable if and only if for all  $\mathcal{M} \models_{\text{pI}}^a \mathcal{P}$ ,  $\mathbb{P}^{\mathcal{M}}(\diamond\Gamma) \not\sim p$ .

**Proof.** Let  $\mathcal{P} = (S, s_0, P, V, Y)$  be a pIMC,  $\Gamma \subseteq A$  be a state label,  $p \in [0, 1]$ , and  $\sim \in \{\leq, <, \geq, >\}$  be a comparison operator. Recall that  $\mathbf{C}_{\exists\Gamma}(\mathcal{P}, \Gamma)$  is a CSP such that each solution corresponds to a MC  $\mathcal{M}$  satisfying  $\mathcal{P}$  where  $\gamma_{s_0}$  is equal to  $\mathbb{P}^{\mathcal{M}}(\diamond\Gamma)$ . Thus adding the constraint  $\gamma_{s_0} \sim p$  allows one to find a MC  $\mathcal{M}$  satisfying  $\mathcal{P}$  such that  $\mathbb{P}^{\mathcal{M}}(\diamond\Gamma) \sim p$ . This concludes the first item presented in the theorem. For the second item, we use Theorem 1 with Proposition 7 which ensure that if the CSP  $\mathbf{C}_{\exists\Gamma}(\mathcal{P}, \Gamma)$  to which is added the constraint  $\gamma_{s_0} \approx p$  is not satisfiable then there is no MC satisfying  $\mathcal{P}$  w.r.t.  $\models_{\text{pI}}^a$  such that  $\mathbb{P}^{\mathcal{M}}(\diamond\Gamma) \approx p$ ; thus  $\mathbb{P}^{\mathcal{M}}(\diamond\Gamma) \sim p$  for all MCs satisfying  $\mathcal{P}$  w.r.t.  $\models_{\text{pI}}^a$ .  $\square$

**Example 8.** Due to the number of constraints in  $\mathbf{C}_{\exists\Gamma}$ , we illustrate our approach on a very small pIMC  $\mathcal{P}$  given in the top left of Fig. 13. In this example, we are interested in two properties: “Does there exist an implementation such that  $\mathbb{P}(\diamond\{\alpha\}) \geq 0.5$ ?” and “Are all implementations such that  $\mathbb{P}(\diamond\{\alpha\}) \leq 0.5$ ?”. These properties are not trivial because the parameter  $q$  appears both on the right side of the interval leading to 1 and on the left side of the interval leading to 2. In the bottom of Fig. 13, we report the constraints obtained in  $\mathbf{C}_{\exists\Gamma}(\mathcal{P}, \{\alpha\})$ . Some of the most trivial constraints have been removed for readability.

We first consider the CSP obtained by adding the constraint ensuring the first quantitative reachability property:  $(QR_1) : \gamma_0 \geq 0.5$ . Solving this CSP yields a solution that allows to build a MC  $\mathcal{M}$ , given in the top right of Fig. 13, satisfying the quantitative reachability property. The valuations obtained by solving the given CSP are such that  $\pi_p = 0$ ,  $\pi_q = 0.5$ ,  $\theta_0^1 = \theta_0^2 = 0.5$ , and, more importantly,  $\gamma_0 = 0.5$ .

We now consider the CSP obtained by adding the constraint checking the second quantitative reachability property:  $(QR_2) : \gamma_0 > 0.5$ . In this case, the CSP is unsatisfiable, which proves that all implementations are such that  $\mathbb{P}(\diamond\{\alpha\}) \leq 0.5$ . This is not really surprising as the probability of going to 2 is necessarily greater or equal to the probability of going to 1.



- (1)  $\rho_0$
  - (2)  $(\neg\rho_1 \Leftrightarrow (\theta_0^1 = 0)) \wedge (\neg\rho_2 \Leftrightarrow (\theta_0^2 = 0))$
  - (3)  $(\neg\rho_1 \Leftrightarrow (\theta_0^1 = 0)) \wedge (\neg\rho_2 \Leftrightarrow (\theta_0^2 = 0)) \wedge \dots$
  - (4)  $(\rho_0 \Leftrightarrow (\theta_0^0 + \theta_0^1 + \theta_0^2 = 1)) \wedge (\rho_1 \Leftrightarrow (\theta_1^1 = 1)) \wedge (\rho_2 \Leftrightarrow (\theta_2^2 = 1))$
  - (5)  $(\rho_0 \Rightarrow (0 \leq \theta_0^0 \leq 1)) \wedge (\rho_0 \Rightarrow (\pi_p \leq \theta_0^1 \leq \pi_q)) \wedge (\rho_0 \Rightarrow (\pi_q \leq \theta_0^2 \leq 1)) \wedge \dots$
  - (6)  $\omega_0 = 1$
  - (7)  $(\omega_1 \neq 1) \wedge (\omega_2 \neq 1)$
  - (8)  $(\rho_1 \Leftrightarrow \omega_1 \neq 0) \wedge (\rho_2 \Leftrightarrow \omega_2 \neq 0) \wedge (\rho_0 \Leftrightarrow \omega_0 \neq 0)$
  - (9)  $((\omega_1 > 1) \Rightarrow (\omega_1 = \omega_0 + 1) \wedge \theta_0^1 > 0) \wedge ((\omega_2 > 1) \Rightarrow (\omega_2 = \omega_0 + 1) \wedge \theta_0^2 > 0)$
  - (10)  $((\omega_1 = 0) \Leftrightarrow (\omega_0 = 0) \vee (\theta_0^1 = 0)) \wedge ((\omega_2 = 0) \Leftrightarrow (\omega_0 = 0) \vee (\theta_0^2 = 0))$
  - (11)  $\alpha_1 = 1$
  - (12)  $(\alpha_0 \neq 1) \wedge (\alpha_2 \neq 1)$
  - (13)  $(\lambda_0 \Leftrightarrow (\rho_0 \wedge (\alpha_0 \neq 0))) \wedge (\lambda_1 \Leftrightarrow (\rho_1 \wedge (\alpha_1 \neq 0))) \wedge (\lambda_2 \Leftrightarrow (\rho_2 \wedge (\alpha_2 \neq 0)))$
  - (14)  $((\alpha_0 > 1) \Rightarrow [(\alpha_0 = \alpha_1 + 1) \wedge (\theta_0^1 > 0)] \vee [(\alpha_0 = \alpha_2 + 1) \wedge (\theta_0^2 > 0)])$   
 $\wedge ((\alpha_2 > 1) \Rightarrow \text{false})$
  - (15)  $(\alpha_0 = 0) \Leftrightarrow [(\alpha_1 = 0) \vee (\theta_0^1 = 0)] \wedge [(\alpha_2 = 0) \vee (\theta_0^2 = 0)] \wedge \dots$
  - (16)  $(\neg\lambda_0 \Rightarrow (\gamma_0 = 0)) \wedge (\neg\lambda_1 \Rightarrow (\gamma_1 = 0)) \wedge (\neg\lambda_2 \Rightarrow (\gamma_2 = 0))$
  - (17)  $\lambda_1 \Rightarrow (\gamma_1 = 1)$
  - (18)  $\lambda_0 \Rightarrow (\gamma_0 = \gamma_1 \cdot \theta_0^1 + \gamma_2 \cdot \theta_0^2 + \gamma_0 \cdot \theta_0^0) \wedge \dots$
- (QR<sub>1</sub>)  $\gamma_0 \geq 0.5$
- (QR<sub>2</sub>)  $\gamma_0 > 0.5$

Fig. 13. Constraint encodings for quantitative reachability.

## 5. Prototype implementation and experiments

Our results have been implemented in a prototype tool<sup>3</sup> which generates the above CSP encodings, and CSP encodings from [12] as well. In this section, we first present our benchmark, then we evaluate our tool for the qualitative properties, and we conclude with the quantitative properties.

### 5.1. Benchmark

MCs have been used for many decades to model real-life applications. PRISM [9]<sup>4</sup> is a reference for the verification of probabilistic systems. In particular, it is able to verify properties for MCs. As said in Section 1, pIMCs correspond to abstractions of MCs. PRISM references several benchmarks based on MCs.<sup>5</sup> Note first that we only consider in this section pIMCs with linear parametric expressions. In this context all CSP encodings for verifying the qualitative properties only use linear constraints while the CSPs encodings for verifying the quantitative properties produce quadratic constraints (*i.e.*, non-linear constraints). This produces an order of magnitude between the time complexity for solving the qualitative properties vs. the quantitative properties w.r.t. our encodings. Thus, we consider two different benchmarks presented in Table 1 and 2. In both cases, pIMCs are automatically generated from the PRISM model in a text format inspired from [19].

<sup>3</sup> All resources, benchmarks, and source code are available online as a Python library at [https://github.com/anict-bart/pimc\\_pylib](https://github.com/anict-bart/pimc_pylib).

<sup>4</sup> <http://www.prismmodelchecker.org/>.

<sup>5</sup> See the category discrete-time Markov chains on the PRISM website.

**Table 1**  
Benchmarks composed of 459 pIMCs over 5 families used for verifying qualitative properties.

Set of benchmarks		#pIMCs	#nodes	#edges	#intervals			#pInBounds			#parameters
					min	avg	max	min	avg	max	
HERMAN	N=3	27	8	28	0	7	18	0	3	11	{2, 5, 10}
HERMAN	N=5	27	32	244	19	50	87	0	12	38	{2, 5, 10}
HERMAN	N=7	27	128	2 188	37	131	236	3	31	74	{5, 15, 30}
EGL	L=2; N=2	27	238	253	16	67	134	0	15	57	{2, 5, 10}
EGL	L=2; N=4	27	6910	7 165	696	1 897	3 619	55	444	1 405	{2, 5, 10}
EGL	L=4; N=2	27	494	509	47	136	276	3	32	115	{2, 5, 10}
EGL	L=4; N=4	27	15 102	15 357	1 448	4 068	7 772	156	951	3 048	{2, 5, 10}
BRP	M=3; N=16	27	886	1 155	16	64	135	1	15	45	{2, 5, 10}
BRP	M=3; N=32	27	1 766	2 307	40	128	275	3	32	129	{2, 5, 10}
BRP	M=4; N=16	27	1 095	1 443	22	80	171	0	20	62	{2, 5, 10}
BRP	M=4; N=32	27	2 183	2 883	49	164	323	3	39	139	{2, 5, 10}
CROWDS	CS=10; TR=3	27	6 563	15 143	1 466	3 036	4 598	57	235	535	{5, 15, 30}
CROWDS	CS=5; TR=3	27	1 198	2 038	190	410	652	8	31	76	{5, 15, 30}
NAND	K=1; N=10	27	7 392	11 207	497	980	1 416	109	466	1 126	{50, 100, 250}
NAND	K=1; N=5	27	930	1 371	60	121	183	9	58	159	{50, 100, 250}
NAND	K=2; N=10	27	14 322	21 567	992	1 863	2 652	197	866	2 061	{50, 100, 250}
NAND	K=2; N=5	27	1 728	2 505	114	217	329	23	101	263	{50, 100, 250}

**Table 2**  
Benchmarks composed of 4 pIMCs used for verifying quantitative properties.

Benchmarks		#nodes	#edges	#pInBounds	#parameters
NAND	K=1; N=2	104	147	82	4
NAND	K=1; N=3	252	364	200	5
NAND	K=1; N=5	930	1 371	726	7
NAND	K=1; N=10	7 392	11 207	5 698	12

For the first benchmark used for verifying qualitative properties, we constructed the pIMCs from existing MCs by randomly replacing some exact probabilities on transitions by (parametric) intervals of probabilities. Our pIMC generator takes 4 arguments: the MC transition function; the number of parameters for the generated pIMC; the ratio of the number of intervals over the number of transitions in the generated pIMC; the ratio of the number of parameters over the number of interval endpoints for the generated pIMC. The benchmarks used are presented in Table 1, where #intervals represents the number of transitions equipped with intervals and #pInBounds represents the number of parametric interval endpoints. We selected 5 applications from PRISM [9]: HERMAN – the self-stabilisation protocol of Herman from [22]; EGL – the contract signing protocol of Even, Goldreich & Lempel from [23]; BRP – the bounded retransmission protocol from [24]; CROWDS – the crowds protocol from [25]; and NAND – the nand multiplexing from [26]. Each one is instantiated by setting global constants (e.g., N for the application HERMAN, L and N for the application EGL) leading to more or less complex MCs. We used our pIMC generator to generate a heterogeneous set of benchmarks: 459 pIMCs with 8 to 15 102 states, and 28 to 21 567 transitions not reduced to [0, 0]. The pIMCs contain from 2 to 250 parameters over 0 to 7 772 intervals.

For the second benchmark used for verifying quantitative properties we extended the NAND model from [26]. The original MC NAND model has already been extended as a pMC in [7], where the authors consider a single parameter  $p$  for the probability that each of the  $N$  nand gates fails during the multiplexing. We extend this model to pIMC by considering one parameter for the probability that the initial inputs are stimulated, and we have one parameter per nand gate to represent the probability that it fails. We consider 4 pIMCs with 104 to 7 392 states, and 147 to 11 207 transitions not reduced to [0, 0]. The pIMCs contain from 4 to 12 parameters appearing over 82 to 5 698 transitions.

## 5.2. Constraint modelling

Given a pIMC in a text format our tool produces the desired CSP according to the selected encoding (i.e., one from [12],  $C_{\exists c}$ ,  $C_{\exists r}$ , or  $C_{\exists r}$ ). Recall that our benchmark only consider linear parametric expressions on transitions. The choice of the constraint programming language for implementing a CSP encoding depends on its nature (e.g., the type of the variables: integer vs. continuous, the kind of the constraints: linear vs. non-linear). Table 3 summarizes the nature of the encodings where **SotA** stands for the encoding from [12] answering the existential consistency problem. Thus, **SotA**,  $C_{\exists c}$ , and  $C_{\exists r}$  can be implemented as Mixed Integer Linear Programs (MILP) [15] and as Satisfiability Modulo Theory (SMT) programs [16] with QF\_LRA logic (Quantifier Free Linear Real-number Arithmetic). This logic deals with Boolean combinations of inequations between linear polynomials over real variables. Note that, QF\_LRA does not deal with integer variables. Indeed logics mixing integers and reals are harder than those over reals only. However, all integer variables in our encodings can be replaced

**Table 3**Characteristics of the four CSP encodings **SotA**,  $C_{\exists c}$ ,  $C_{\exists r}$ , and  $C_{\exists f}$ .

Encoding	Size of the produced CSPs	Boolean var.	Integer var.	Real-number var.	Boolean constr.	Linear constr.	Quadratic constr.
<b>SotA</b>	exponential	no	no	yes	yes	yes	no
$C_{\exists c}$	linear	yes	no	yes	yes	yes	no
$C_{\exists r}$	linear	yes	yes	yes	yes	yes	no
$C_{\exists f}$	linear	yes	yes	yes	yes	yes	yes

**Table 4**Comparison of sizes, encoding, and solving times for three approaches: (1) **SotA** encoding implemented in SMT, (2)  $C_{\exists c}$  encoding implemented in SMT, and (3)  $C_{\exists c}$  encoding implemented in MILP (times are given in seconds).

Set of benchmarks		avg(#variables)			avg(#constraints)			avg(encod. time)			avg(solv. time)		
		(1)	(2)	(3)	(1)	(2)	(3)	(1)	(2)	(3)	(1)	(2)	(3)
HERMAN	N=3	71	42	42	1258	272	238	0.10	0.07	0.07	0.01	0.01	0.01
HERMAN	N=5	1 031	282	282	51 064	2 000	1 750	1.52	0.08	0.08	0.24	0.03	0.01
HERMAN	N=7	16 402	2 333	2 333	1 293 907	16 483	14 278	50.47	0.13	0.13	5.92	0.28	0.04
EGL	L=2; N=2	462	497	497	4 609	3 917	3 658	0.21	0.08	0.08	0.02	0.04	0.01
EGL	L=2; N=4	13 786	14 081	14 081	138 596	112 349	105 178	5.66	0.44	0.44	0.54	2.15	0.36
EGL	L=4; N=2	958	1 009	1 009	9 560	8 013	7 498	0.36	0.10	0.10	0.04	0.08	0.02
EGL	L=4; N=4	30 138	30 465	30 465	301 866	243 421	228 058	13.03	0.87	0.87	1.26	11.31	0.97
BRP	MAX=3; N=16	68 995	2 047	2 047	738 580	16 063	14 902	32.29	0.12	0.12	3.54	0.21	0.06
BRP	MAX=3; N=32	OM	4 079	4 079	OM	32 047	29 734	OM	0.18	0.18	OM	0.47	0.13
BRP	MAX=4; N=16	103 105	2 544	2 544	1 114 774	19 960	18 511	46.54	0.13	0.13	5.42	0.27	0.08
BRP	MAX=4; N=32	OM	5 072	5 072	OM	39 832	36 943	OM	0.21	0.21	OM	0.63	0.17
CROWDS	CS=10; TR=3	OM	21 723	21 723	OM	165 083	149 923	OM	0.67	0.66	OM	11.48	0.79
CROWDS	CS=5; TR=3	OM	3 253	3 253	OM	25 063	23 008	OM	0.16	0.15	OM	0.39	0.09
NAND	K=1; N=10	87 506	18 732	18 732	888 733	145 108	133 768	152.06	0.56	0.56	3.72	6.21	0.79
NAND	K=1; N=5	6 277	2 434	2 434	62 987	18 098	16 594	10.26	0.12	0.12	0.24	0.25	0.07
NAND	K=2; N=10	169 786	36 022	36 022	1 722 970	279 998	258 298	298.93	1.04	1.04	7.75	31.81	2.06
NAND	K=2; N=5	11 623	4 366	4 366	117 814	33 218	30 580	19.24	0.17	0.17	0.44	0.48	0.13

by real-number variables.<sup>6</sup> Each integer variable  $x$  can be declared as a real variable whose real domain bounds are its original integer domain bounds; we also add the constraint  $x < 1 \Rightarrow x = 0$ . Since we only perform incrementation of  $x$  this preserves the same set of solutions (*i.e.*, ensures integer integrity constraints). Finally, due to the non-linear constraints in  $C_{\exists f}$ , these encodings are implemented as SMT programs [16] with the QF\_NRA logic (Quantifier Free Non linear Real-number Arithmetic). We use the same technique as for  $C_{\exists c}$  and  $C_{\exists r}$  for replacing integer variables by real-number variables. We chose the programming language Python for implementing our CSP modeller. We do not evaluate any arithmetic expression while generating CSPs, and numbers in the interval endpoints of the pIMCs are read as strings and no trivial simplification is performed while modelling. We do so to avoid any rounding of the interval endpoints when using floating point numbers.

Experiments have been realized on a 2.4 GHz Intel Core i5 processor. Time out has been set to 10 minutes. Memory out has been set to 2 GB. Table 4 presents the average size (*i.e.*, the number of variables and the number of constraints) of the instances considered for each set of benchmarks introduced in Table 1, as well as the average encoding and solving time for the existential consistency problem using (1) SMT **SotA** encoding, (2) SMT  $C_{\exists c}$  encoding, and (3) MILP  $C_{\exists c}$  encoding. First, note that all pIMCs are successfully compiled when using our  $C_{\exists c}$  encoding while the **SotA** encoding produces out of memory errors for 4 sets of benchmarks: more than 20% of the instances (see OM cells in Table 4). We recall that the **SotA** encoding is defined inductively, and that it iterates over the power set of the states. In practice, this implies deep recursions joined with enumeration over the power set of the states. The exponential gain exposed in Section 3 is visible in terms of number of variables and constraints in Table 4, and in terms of encoding time in Fig. 14. Each dot in Fig. 14 corresponds to one instance of our benchmark. While the encoding time ranges between 0 and 1 s when using the  $C_{\exists c}$  encoding, it varies between 0 and 500 s when using the **SotA** encoding (if it does not run out of memory).

MILP formulation of logical constraints (*e.g.*, conjunction, disjunction, implication, equivalence) implies the introduction of binary variables called indicator variables [27]. Each indicator variable is associated to one or more constraints. The valuation of the indicator variable activates or deactivates its associated constraints. We tried to formulate the **SotA** encoding into MILP. Unfortunately, the nested conjunctions and disjunctions imply the introduction of a huge number of indicator variables, leading to giant instances giving bad encoding and solving time. However, since the Boolean variables in  $C_{\exists c}$  exactly correspond to indicator variables, the MILP formulation of the  $C_{\exists c}$  encoding does not introduce additional variables or constraints. The difference between  $C_{\exists c}$  in SMT and  $C_{\exists c}$  in MILP comes from the encoding of the domains of the

<sup>6</sup> Note that obtaining integer integrity constraints over real-numbers can be costly.

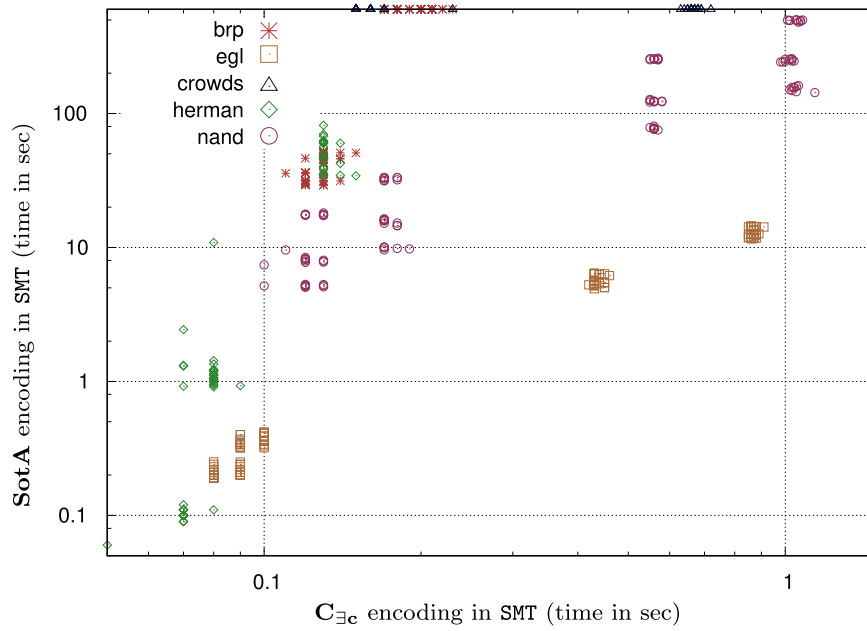


Fig. 14. Comparing encoding time for the existential consistency problem.

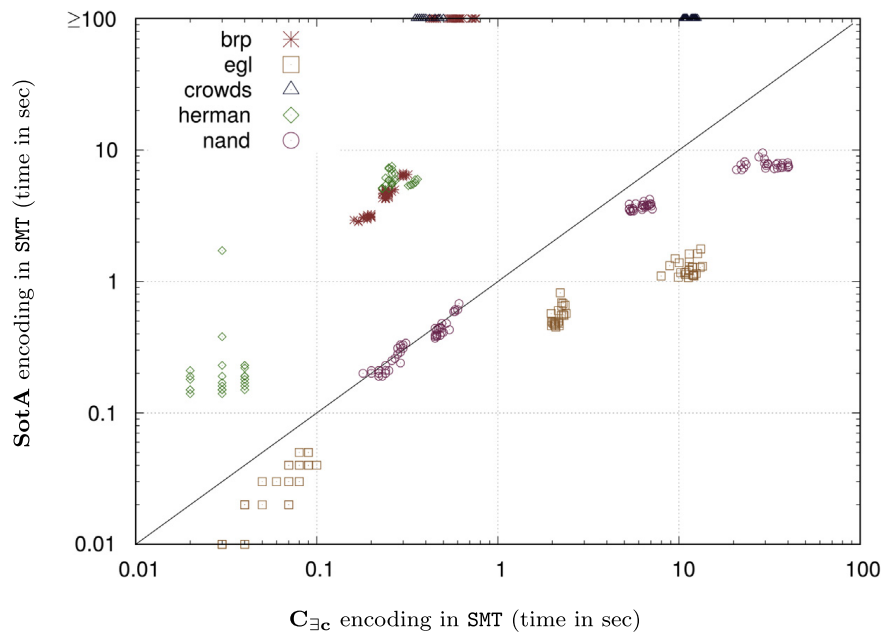


Fig. 15. Comparing solving time for the existential consistency problem.

continuous variables: in SMT, it requires the use of inequality constraints, e.g.,  $0 \leq x \leq 1$ . The encoding time is the same for SMT and MILP  $C_{\exists c}$  encoding.

### 5.3. Solving

We chose Z3 [28] in its last version (v. 4.4.2) as SMT solver. We chose CPLEX [29] in its last version (v. 12.6.3.0) as MILP solver. Both solvers have not been tuned and we use their default strategies. Experiments have been realized on a 2.4 GHz Intel Core i5 processor. Time out has been set to 10 minutes. Memory out has been set to 2 GB.

Table 4 presents the resolution time for the existential consistency problem on our first benchmark using (1) SMT **SotA** encoding, (2) SMT  $C_{\exists c}$  encoding, and (3) MILP  $C_{\exists c}$  encoding. While the **SotA** CSPs are larger than the  $C_{\exists c}$  CSPs, the solving time for the **SotA** CSPs appears to be competitive compared to the solving time for the  $C_{\exists c}$  CSPs. The scatter plot in Fig. 15 (logarithmic scale) compares solving times for the SMT **SotA** encoding and SMT  $C_{\exists c}$  encoding. However when considering the resolution time of the problem (i.e., the encoding time plus the solving time) the  $C_{\exists c}$  encoding clearly answers faster than the **SotA** encoding. Finally, the comparison between the solving time using SMT  $C_{\exists c}$  encoding and MILP  $C_{\exists c}$  encoding is illustrated in Fig. 16. It shows that both the loss of safety (passing from real numbers with Z3 SMT resolution to floating

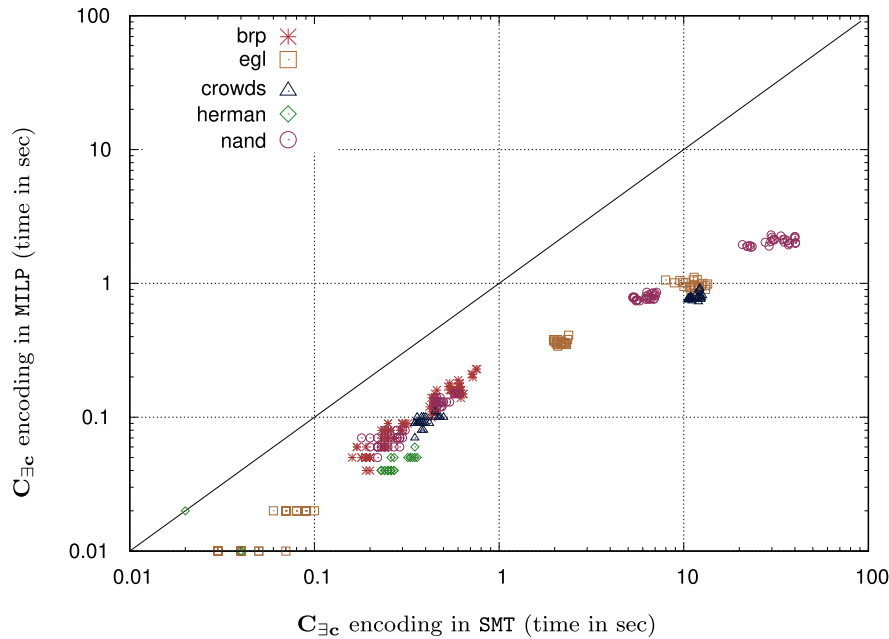


Fig. 16. Comparing solving time between SMT and MILP formulations.

Table 5

Comparison of solving times between qualitative and quantitative encodings.

Benchmark		pIMC			$C_{\exists c}$		$C_{\exists r}$			$C_{\exists f}$			
		#states	#trans.	#par.	#var.	#cstr.	time	#var.	#cstr.	time	#var.	#cstr.	time
NAND	K=1; N=2	104	147	4	255	1526	0.17 s	170	1497	0.19 s	296	2457	69.57 s
NAND	K=1; N=3	252	364	5	621	3727	0.24 s	406	3557	0.30 s	703	5828	31.69 s
NAND	K=1; N=5	930	1371	7	2308	13859	0.57 s	1378	12305	0.51 s	2404	20165	T.O.
NAND	K=1; N=10	7392	11207	12	18611	111366	9.46 s	9978	89705	13.44 s	17454	147015	T.O.

point numbers with CPLEX MILP resolution) and the fact that CPLEX is highly optimized for MILP problems whereas Z3 is a “generic” solver lead to a non negligible gain in terms of resolution time (near to an exponential gain in our benchmark). Indeed the SMT  $C_{\exists c}$  encoding requires 50 seconds to complete the solving process for all benchmarks while the MILP  $C_{\exists c}$  encoding needs less than 5 seconds for the same instances.

Table 5 summarizes the results w.r.t. our second benchmark: the pIMC sizes (in terms of states, transitions, and parameters), the CSP sizes (in terms of number of variables and constraints), and the resolution time using the Z3 solver. Note first that we perform pre-processing when verifying reachability properties: using a simple graph analysis, we eliminate some states that cannot, trivially, reach the goal states. This explains why  $C_{\exists r}$  has less variables and constraints than  $C_{\exists c}$ . Finally, note the order of magnitude between the resolution time required for solving the qualitative properties vs. the quantitative properties w.r.t. our encodings. Indeed, we did not succeed in solving pIMCs with more than 300 states and 400 transitions for quantitative properties while we verified pIMCs with more than 10 000 states and 20 000 transitions in the qualitative context.

## 6. Conclusion and perspectives

In this paper, we have compared several Markov chain abstractions in terms of succinctness and we have shown that Parametric Interval Markov Chain is a strictly more succinct abstraction formalism than other existing formalisms such as Parametric Markov Chains and Interval Markov Chains. In addition, we have proposed constraint encodings for checking several properties over pIMC. In the context of qualitative properties such as existential consistency or consistent reachability, the size of our encodings is significantly smaller than other existing solutions. In the quantitative setting, we have compared the three semantics for IMCs and pIMCs and showed that the semantics are equivalent with respect to quantitative reachability properties. As a side effect, this result ensures that all existing tools and algorithms solving reachability problems in IMCs under the once-and-for-all semantics can safely be extended to the IMDP and at-every-step semantics with no changes. Based on this result, we have then proposed CSP encodings addressing quantitative reachability in the context of pIMCs regardless of the chosen semantics. Finally, we have developed a prototype tool that automatically generates our CSP encodings and that can be plugged to any constraint solver accepting the SMT-LIB format as input.



Our tool for pIMC verification could be extended in order to manage other, more complex, properties (e.g., supporting the LTL-language in the spirit of what Tulip [19] does). Also one could investigate a practical way of computing and representing the set of *all solutions* to the parameter synthesis problem.

## Acknowledgements

This work is partially supported by the ANR national research programs PACS (ANR-14-CE28-0002) and Coverif (ANR-15-CE25-0002), and the regional programme Atlanstic2020 funded by the French Region Pays de la Loire and European Regional Development Fund.

## References

- [1] J.A. Whittaker, M.G. Thomason, A Markov chain model for statistical software testing, *IEEE Trans. Softw. Eng.* 20 (10) (1994) 812–824.
- [2] D. Husmeier, R. Dybowski, S. Roberts, *Probabilistic Modeling in Bioinformatics and Medical Informatics*, Springer Publishing Company, Incorporated, 2010.
- [3] R. Alur, T.A. Henzinger, M.Y. Vardi, Parametric real-time reasoning, in: *Symposium on Theory of Computing*, ACM, 1993, pp. 592–601.
- [4] B. Jonsson, K.G. Larsen, Specification and refinement of probabilistic processes, in: *Symposium on Logic in Computer Science*, IEEE, 1991, pp. 266–277.
- [5] K. Chatterjee, K. Sen, T.A. Henzinger, Model-checking omega-regular properties of interval Markov chains, in: *International Conference on Foundations of Software Science and Computational Structures*, in: *Lecture Notes in Comput. Sci.*, vol. 4962, Springer, 2008, pp. 302–317.
- [6] K. Sen, M. Viswanathan, G. Agha, Model-checking Markov chains in the presence of uncertainties, in: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, in: *Lecture Notes in Comput. Sci.*, vol. 3920, Springer, Berlin, Heidelberg, 2006, pp. 394–410.
- [7] C. Dehnert, S. Junges, N. Jansen, F. Corzilius, M. Volk, H. Bruintjes, J.-P. Katoen, E. Ábrahám, Prophesy: a probabilistic parameter synthesis tool, in: *International Conference on Computer Aided Verification*, in: *Lecture Notes in Comput. Sci.*, vol. 9207, Springer, Cham, 2015, pp. 214–231.
- [8] E.M. Hahn, H. Hermans, B. Wachter, L. Zhang, PARAM: a model checker for parametric Markov models, in: *International Conference on Computer Aided Verification*, in: *Lecture Notes in Comput. Sci.*, vol. 6174, Springer, 2010, pp. 660–664.
- [9] M.Z. Kwiatkowska, G. Norman, D. Parker, PRISM 4.0: verification of probabilistic real-time systems, in: *International Conference on Computer Aided Verification*, in: *Lecture Notes in Comput. Sci.*, vol. 6806, Springer, 2011, pp. 585–591.
- [10] S. Chakraborty, J.-P. Katoen, Model checking of open interval Markov chains, in: *International Conference on Analytical and Stochastic Modelling Techniques and Applications*, in: *Lecture Notes in Comput. Sci.*, vol. 9081, Springer, Cham, 2015, pp. 30–42.
- [11] M. Benedikt, R. Lenhardt, J. Worrell, LTL model checking of interval Markov chains, in: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, in: *Lecture Notes in Comput. Sci.*, vol. 7795, Springer, 2013, pp. 32–46.
- [12] B. Delahaye, D. Lime, L. Petrucci, Parameter synthesis for parametric interval Markov chains, in: *International Conference on Verification, Model Checking, and Abstract Interpretation*, in: *Lecture Notes in Comput. Sci.*, vol. 9583, Springer, 2016, pp. 372–390.
- [13] F. Rossi, P. van Beek, T. Walsh, *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*, Elsevier Science Inc., 2006.
- [14] A. Bart, B. Delahaye, D. Lime, E. Monfroy, C. Truchet, Reachability in parametric interval Markov chains using constraints, in: *International Conference on Quantitative Evaluation of Systems*, in: *Lecture Notes in Comput. Sci.*, vol. 10503, Springer, 2017, pp. 173–189.
- [15] J.P. Vielma, Mixed integer linear programming formulation techniques, *SIAM Rev.* 57 (1) (2015) 3–57.
- [16] C.W. Barrett, R. Sebastiani, S.A. Seshia, C. Tinelli, Satisfiability modulo theories, in: *Handbook of Satisfiability*, vol. 185, 2009, pp. 825–885.
- [17] C. Baier, J.-P. Katoen, *Principles of Model Checking (Representation and Mind Series)*, The MIT Press, 2008.
- [18] G. Cantor, Über unendliche, lineare punktmannigfaltigkeiten V (On infinite, linear point-manifolds (sets)), *Math. Ann.* 21 (1883) 545–591.
- [19] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, R.M. Murray, Tulip: a software toolbox for receding horizon temporal logic planning, in: *ACM International Conference on Hybrid Systems: Computation and Control*, ACM, 2011, pp. 313–314.
- [20] A. Puggelli, W. Li, A.L. Sangiovanni-Vincentelli, S.A. Seshia, Polynomial-time verification of PCTL properties of MDPs with convex uncertainties, in: *International Conference on Computer Aided Verification*, in: *Lecture Notes in Comput. Sci.*, vol. 8044, Springer, 2013, pp. 527–542.
- [21] B. Delahaye, Consistency for parametric interval Markov chains, in: *International Workshop on Synthesis of Complex Parameters*, in: *OpenAccess Ser. Inform. (OASISs)*, vol. 44, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015, pp. 17–32.
- [22] M. Kwiatkowska, G. Norman, D. Parker, Probabilistic verification of Herman's self-stabilisation algorithm, *Form. Asp. Comput.* 24 (4) (2012) 661–670.
- [23] G. Norman, V. Shmatikov, Analysis of probabilistic contract signing, *J. Comput. Secur.* 14 (6) (2006) 561–589.
- [24] P. D'Argenio, B. Jeannot, H. Jensen, K. Larsen, Reachability analysis of probabilistic systems by successive refinements, in: *Joint International Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification*, in: *Lecture Notes in Comput. Sci.*, vol. 2165, Springer, 2001, pp. 39–56.
- [25] V. Shmatikov, Probabilistic model checking of an anonymity system, *J. Comput. Secur.* 12 (3/4) (2004) 355–377.
- [26] G. Norman, D. Parker, M. Kwiatkowska, S. Shukla, Evaluating the reliability of NAND multiplexing with PRISM, *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* 24 (10) (2005) 1629–1637.
- [27] P. Belotti, P. Bonami, M. Fischetti, A. Lodi, M. Monaci, A. Nogales-Gómez, D. Salvagnin, On handling indicator constraints in mixed integer programming, *Comput. Optim. Appl.* (2016) 1–22.
- [28] L. De Moura, N. Bjørner, Z3: an efficient SMT solver, in: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, in: *Lecture Notes in Comput. Sci.*, Springer, 2008, pp. 337–340.
- [29] IBM ILOG CPLEX Optimizer (Last 2010).

# Parametric Statistical Model Checking of UAV Flight Plan<sup>\*</sup>

Ran Bao<sup>1,2</sup>, Christian Attiogbe<sup>2[0000-0002-7815-1752]</sup>, Benoît Delahaye<sup>2[0000-0002-9104-4361]</sup>, Paulin Fournier<sup>2</sup>, and Didier Lime<sup>3</sup>

<sup>1</sup> PIXIEL GROUP, Nantes, France <https://www.pixiel-group.com/>

<sup>2</sup> Université de Nantes - LS2N UMR CNRS 6004, Nantes, France

<sup>3</sup> Centrale Nantes - LS2N UMR CNRS 6004, Nantes, France

**Abstract.** Unmanned Aerial Vehicles (UAV) are now widespread in our society and are often used in a context where they can put people at risk. Studying their reliability, in particular in the context of flight above a crowd, thus becomes a necessity. In this paper, we study the modeling and analysis of UAV in the context of their flight plan. To this purpose, we build a parametric probabilistic model of the UAV and use it, as well as a given flight plan, in order to model its trajectory. This model takes into account parameters such as potential filter or sensor (like GPS) failure as well as wind force and direction. Because of the nature and complexity of the successive obtained models, their exact verification using tools such as PRISM or PARAM is impossible. We therefore develop a new approximation method, called Parametric Statistical Model Checking, in order to compute failure probabilities. This method has been implemented in a prototype tool, which we use to resolve complex issues in a practical case study.

**Keywords:** UAV · Formal Model · Markov Chain · Parametric statistical model checking

## 1 Introduction

Unmanned Aerial Vehicles (UAV) are more and more present in our lives through entertainment or industrial activities. They can be dangerous for their environment, for instance in case of a failure when an UAV (aka a drone) is flying above a crowd. Unfortunately until today, there does not exist any kind of UAV regulation around the world. Only some recommendations are used; for instance in order to avoid accidents in case of malfunctioning, a drone should never fly above a crowd.

In this context, we are working with PIXIEL group to build a reliable UAV control system. PIXIEL group is a company expert in safety drones and public performances including UAVs. For example, PIXIEL is in particular known for developing a public performance in the French entertainment park called "Puy

---

<sup>\*</sup> Supported by PIXIEL and Association Nationale Recherche Technologie (ANRT)

du Fou” that includes both human actors and drones. The company is strongly attached to the safety of the public. Therefore, ensuring that its UAV systems are secure for humans during the performances is a priority. As for the current practices, the performances including UAVs are only allowed to occur when the weather is sunny and when the area above which the UAVs fly is unauthorized for actors and public. However, there is no certification proving that the UAVs always follow their intended flight plan.

The management of performances indeed requires to pay close attention to the drone trajectory computation as well as to the accuracy of the measurements concerning its immediate position in space and its movements. However, a rigorous study is necessary to ensure reliability of the drone control system, for instance by decreasing the risks of failure using the appropriate tuning of the drone flying parameters which impact the computation of its trajectory. Accordingly, the questions are *how to prove that the UAV failure probability is low and which parameters have to be taken into account to ensure human safety during performances including UAVs.*

High-quality aircrafts such as Hexarotors can easily avoid the majority of minor failures related to hardware because they can fly with only five motors and the probability of concurrent failure of more than two motors is in general negligible. In the same way, in case of battery failure, the UAV is able to land down on a specified area without any safety issue for the environment as long as it is situated in a safe zone where humans are not endangered. However, software failure may be a lot more problematic and complex to study. In this case, the UAV behavior might become unpredictable. One critical issue in this context is the potential inaccuracy of position estimation in drone systems, either as a result of inaccurate sensor measurements or of misinterpretation of data coming from those sensors. Besides aircraft system failure consideration, there is also a far more critical aspect to take into account: the weather environment. Therefore, a general approach to improve UAV safety is to study the impact of inaccuracy in position measurements on the resulting flight path compared to a given, fixed, flight plan while taking into account weather conditions.

There are many works dedicated to the UAV domain. In [20] Koppány Máthé and Lucian Buşoniu basically explain the functioning of a drone. UAV movement recognition is studied in [10]. Automatic landing on target is described in [17] and monitoring and conservation are dealt with in [11]. Some works also try to detect breakdowns and malfunctions that can impact drones. We can mention *inter alia*, the detection of communication errors in a multi-drone framework studied in [13] or the development of a basic diagnosis model for solving system issues in [9]. Our work is closer to this second category of topics. However, to the best of our knowledge, there are no existing works on the parametric study of the impact of component inaccuracy on UAV trajectory. In [5, 24] the authors study through the *secure estimation problems* how to estimate the true states of an UAV system when the measurements from sensors are corrupted, for instance by attackers. In their work, these authors reformulate the estimation problem into the error correction problem and then they use the successively observed

measurement anomalies to reconstruct the correct states of the system. While the used techniques are completely different, the objectives of avoiding bad states is similar to ours, in avoiding the states reaching bad security zones.

The purpose of our work is therefore to provide means to study the reliability of UAVs in the context of a given flight plan. In order to do that, we have to build a formal (mathematical) model which will allow us (1) to analyze the drone system and detect the most important parameters, and (2) to tune those parameters in order to reduce the system failure probability. To this intent, we thoroughly study the UAV system, formalize it and analyze it with using parametric probabilistic methods. Among the components of a drone system, we particularly focus on the *Flight Control System* (FCS), which is responsible for computing estimations of the UAV position during its flight in order to adapt its trajectory to a given predefined flight plan. We therefore build a formal model of the flight controller in terms of parametric probabilistic models that takes into account the potential inaccuracy of the position estimation. Since UAVs are particularly sensitive to the weather environment (and in particular to wind conditions), we also enhance our model in order to take into account potential wind perturbations. Since wind force can drastically vary from one point of a given flight plan to another, we also use parameters to encode the wind force and allow our model to adapt to particular weather conditions.

The contributions of this paper are:

- a method to build a parametric model of UAV systems; the parameters can then be finely tuned until reaching values that ensure defined safety thresholds;
- a parametric statistical model checking technique; this enables us to formally analyze the parametric models build for the drones. Indeed because of the complexity of the built models, tools such as PRISM [15,16] and PARAM [12] were limited for their analysis.
- an illustration of the use of our method on a complex industrial case study.

The paper is organized as follows. In Section 2 we provide the essential background to understand UAV functioning and then we build a formal model that support their behaviours. Section 3 is an introduction to parametric Markov chains and Statistical Model Checking. Implementations of the models and experimentations are presented in Section 4; finally Section 5 draws conclusions and further work.

## 2 Building a Formal Model of UAV

In this section, we present our method to build the UAV model. Recall that we are interested in studying UAV safety, i.e. studying the probability that a UAV encounters dangerous situations. These situations are of two kinds: either the UAV can stop flying and fall, or it can enter a "forbidden" zone where it endangers humans. As explained earlier, professional UAV can handle the falling risk through material redundancy. Moreover, as long as a UAV stays in a "safe"

zone, it will not endanger human even in case of falling. The aim of our model is therefore to evaluate the probability for a UAV to enter a "forbidden" zone.

We start by explaining how the zones are computed with respect to the given flight plan. We then show how the UAV software can be decomposed into components and focus on the most important ones. Finally, we detail how the formal (mathematical) models for the important components are built and present the resulting global model.

## 2.1 Safety zones

In the context of software, considerations in airborne systems and equipment certification (named DO-178C) defined five levels of safety zones, the most secure being Zone 1 and the most dangerous being Zone 5. These zones are characterized by their distance from the intended flight plan, as shown in Figure 1.

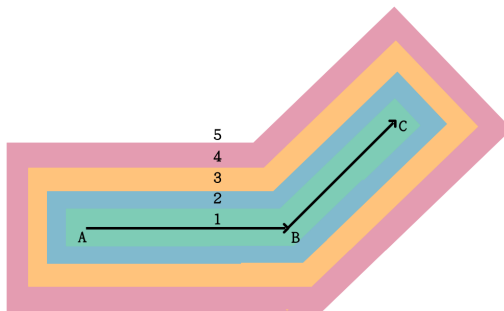


Fig. 1: Safety zones

The size of each safety zone is not definitely fixed; it can be defined for a specific requirement or for a given application. In practice the safety zones are specifically defined for a flight environment and for a given flight plan. The main principle is that no human should be present in Zones 1 to 3, while a few people can be present in Zone 4 and most people can be present in Zone 5. As a consequence, the probability that the UAV endangers humans is directly proportional to the probability that it enters Zones 4 or 5. In the following of the paper, our target will therefore be to compute this probability.

## 2.2 Drone components

We now move to the decomposition of the UAV hardware and UAV software into components and introduce the most important component in the UAV system: the flight controller (FC). The FC is responsible for collecting data from various sensors, using this data to compute the precise *position* and *attitude* of the drone and adjust the attitude in order to follow the given flight plan to the best of its ability.

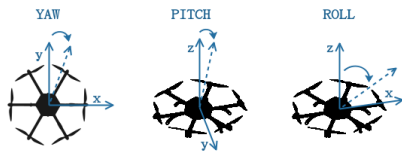


Fig. 2: Attitude coordinates

Notice the difference between position and attitude: while the position of the UAV is defined by 3-dimensional coordinates  $x$ ,  $y$  and  $z$ , its attitude is the collection of *yaw*, *pitch* and *roll* measurements for the UAV compared to the vertical (see Figure 2). The attitude allows to control the movement

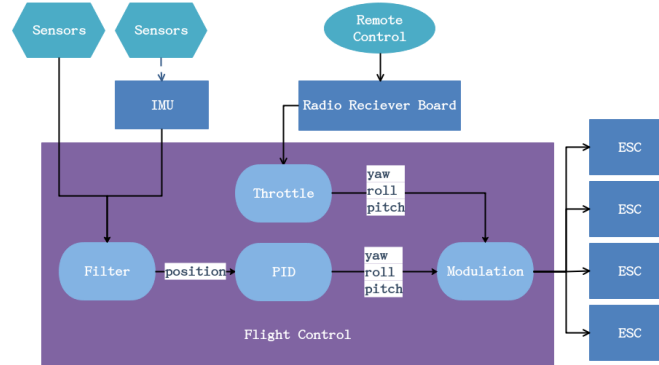


Fig. 3: Flight Control Overview

of the UAV: by controlling the speed of each motor, one can control which motor will be the highest, and hence control the direction the UAV will fly to.

**Flight Controller** As explained above, the FC is the central component in any UAV as it is responsible for collecting data from sensors and translating them to the UAV attitude. An overview of the FC of an UAV is given in Figure 3. Remark that the FC can be linked to components responsible for communicating with a remote control. While these components are necessary in order to allow a pilot to take over when the automatic flight mode of the UAV fails, we will consider in the following that this is not the case and that the UAV we study are always in automatic flight mode.

As one can see from Figure 3, the intuitive behavior of the FC is as follows. The filter uses sensors measurements in order to compute the current drone position and attitude. Since the data can be noisy and inaccurate, the filter uses complex algorithms in order to clean the noises in the measurements and compute a realistic position and attitude. Remark that in some cases, the filter can itself introduce inaccuracy in the computed position and attitude, which can be problematic. Once the estimated current position and attitude are computed, the Proportional Integral Derivative (PID) uses this information to compute the local trajectory that the drone has to follow in order to be as close as possible to its intended flight plan. This local trajectory is then transformed into a new value for the attitude of the drone. Finally, Modulation transforms this attitude into signal to the Electronic Speed Controller (ESC) which is responsible for controlling each motor's speed.

Recall that we are interested in computing the probability that a UAV enters a forbidden zone while following its flight plan. By construction, as long as the position and attitude measurements are perfect, there is no reason why the UAV should deviate from its intended trajectory, and therefore the probability that it enters a forbidden zone is null. However, as explained above, the data gathered

from sensors can be noisy and inaccuracy can sometimes be introduced through filtering. In this case, the estimated position and attitude of the UAV can be faulty, resulting in a deviation from the intended flight plan and potentially leading to a forbidden zone. It is therefore of paramount importance to study how the filters work and to take into account in our formal model the potential inaccuracy of position and attitude measurement.

**Filter** The role of the filter is to use sensors measurements in order to compute the UAV position and attitude with the highest possible precision. However, the high precision comes with a cost in terms of complexity: in order to gain precision, filters have to run complex algorithms which takes time. As a consequence, the most precise filters are also the slowest, which implies that the position can be estimated less often, which itself results in inaccuracy.

There exists a large amount of filters in UAV industry, among which one can find Extended Kalman Filter (EKF) [22], Explicit Complement Filter [8], Gradient Descent [19], Conjugate Gradient, and a more accurate but slower filter: Unscented Kalman Filter (UKF) [6], etc. Usually, researchers use EKF as a fundamental to compare to other kinds of filters and explain precision and speed differences. All filters improve their accuracy during the flight through training, in particular by recording recurrent noises and correcting them. However, this training is only valid through a single flight and is lost as soon as the UAV lands.

Since the accuracy of the estimated position and attitude is of paramount importance for computing the probability of entering a forbidden zone, and since the choice of filter has a direct impact on this measurement, we chose to implement this accuracy as a *parameter* of our model. This will be explained in more details in Section 2.4.

### 2.3 Formal Model of the UAV in its Environment

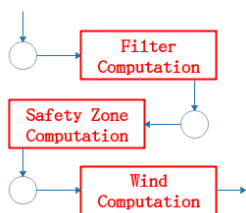


Fig. 4: A flow diagram of the formalization steps

We use a flow diagram to present our global approach for formalizing the UAV functioning (See Figure 4). After a step where the filter computation reflects the precision of position and attitude estimation, we consider the computations of the probabilities to reach the given safety zones in the next time-step; accordingly, the idea is to adapt the next attitude according to the original flight plan in order to be more secure. The last step allows to incorporate wind perturbations and compute the next UAV position.

As explained above, the filter is one of the most crucial components and its ability to estimate the UAV position precisely has a huge impact on the probability of reaching a forbidden zone. For this reason, we choose to represent the accuracy of the estimated position of the UAV (therefore including both sensor measurements and filter correction) as a parameter of our model. In the following, we show how the next

position of the UAV is computed according to the current estimated position, and how errors in the estimation can lead to the drone entering forbidden zones.

**Computation of the next position** We now explain how the next position is computed according to the estimated current position. In particular, we show that inaccuracy in the estimation can lead the UAV to entering a forbidden zone.

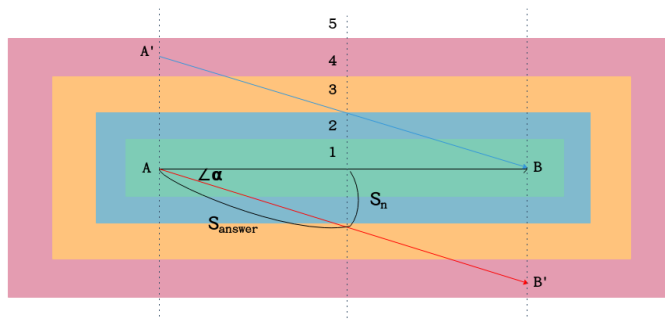


Fig. 5: Issue on drone location and misleading positions

For the sake of simplicity, we assume here that the UAV moves in 2 dimensions only and that inaccuracy only occurs on one of them. Figure 5 illustrates the situation. Assume that the intended flight plan consists in going from point  $A$  to point  $B$ . Assume also that the current position of the UAV is exactly on  $A$  but that the estimated position (taking into account sensors and filter inaccuracy) is on  $A'$ . As a consequence, the PID will try to correct the current deviation by changing the angle of the UAV in order to lead it back to  $B$ . However, since the UAV is really on  $A$ , the correction will instead lead the UAV to a position  $B'$ , in the forbidden zone. Fortunately, the position estimation takes place several times between  $A$  and  $B$ , according to the filter frequency  $f$ . Therefore, a new position will be estimated before reaching  $B'$ , hopefully with a better accuracy, which will allow the PID to again correct the trajectory. We should also take into account that the speed of the UAV is also computed according to the flight plan, which precises the remaining time and distance before the next checkpoint. We now show how we can compute the safety zone where the UAV ends before the position is estimated again. In Figure 5, this zone is represented by the distance  $S_n$ .

Let  $S_{answer}$  be the distance that the UAV covers before a new estimation of the position. Let  $V$  be the velocity of the UAV, which is computed by the PID in order to reach  $B$  on time, i.e. in precisely  $T$  time units. We therefore have  $V = A'B/T$ , and  $S_{answer} = V/f = (A'B)/(T * f)$ . Finally,  $AA'/A'B = S_n/S_{answer}$ , and therefore

$$S_n = \frac{AA'}{T * f}.$$



Remark that the resulting deviation is directly proportional to  $AA'/f$ , hence the necessity to take into account the trade-off between accuracy and filter speed in order to optimize the probability of never entering any dangerous zone.

Taking into account wind perturbations follows a similar computation than the one presented above. This allows us to incorporate wind parameters as well in our model.

## 2.4 Resulting Global Model

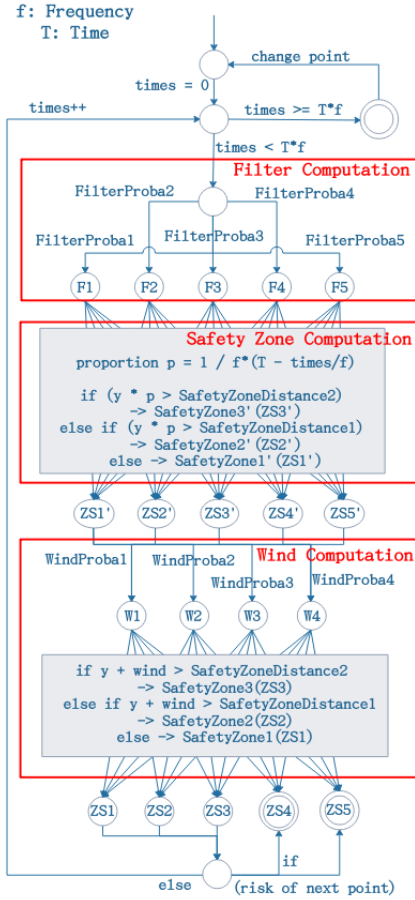


Fig. 6: Global behaviour of the FCS

safety zone to which this position belongs. When the wind is not taken into account, the result of this computation is enough to decide whether the model should pursue its execution. When the wind is taken into account, another step follows, depicted in the box labelled **Wind Computation**, where other probabilistic parameters are used in order to decide the wind strength (we assume

The global model of the UAV flight control system is depicted in Figure 6. The purpose of this model is to represent the computations taking place in the FCS in order to adapt the UAV trajectory to the intended flight plan according to inaccurate position and attitude estimations as well as wind perturbations. In this model, the exact position of the UAV is encoded using 3-d coordinates. These coordinates are then compared to the intended flight plan in order to decide to which safety zone they belong. As soon as the UAV reaches one of the forbidden zones (4 or 5), the computation stops.

The model uses several probabilistic parameters. Parameters  $FilterProba1$ ,  $FilterProba2$ ,  $FilterProba3$ ,  $FilterProba4$  and  $FilterProba5$  represent the accuracy of the position and attitude estimation by both the filter and the sensors. The resulting probabilistic choice depicted in the box labelled **Filter Computation** therefore dictates the distance between the exact and estimated position of the UAV. This choice is followed by a computation in the box labelled **Safety Zone Computation** that computes the exact coordinates of the next position of the drone and allows to decide the

that the direction is constant) and a new position taking into account these perturbations is computed. Finally, the zone to which this last position belongs is computed and, depending on whether this zone is safe, the model goes on to another position estimation.

Remark that the filter frequency and the position and distance of checkpoints in the flight plan are given as inputs to the model. The position of checkpoints in the flight plan allows to compute the required UAV speed, while the frequency of the filter allows to fix the number of position estimations that will happen in a given flight plan (i.e. the number of loops the model goes through, at most).

### 3 Parametric Statistical Model Checking

As explained above, we have developed a parametric probabilistic model in order to represent the behaviour of our UAV according to a given flight plan. We now introduce the necessary theory to formally compute the probabilities of a given UAV entering a forbidden zone in the context of its flight plan. We start by recalling a classical verification technique called Statistical Model Checking (SMC), then introduce the modeling formalism we use: parametric Markov Chains (pMCs) and finally show how SMC can be adapted to this formalism.

#### 3.1 Standard Statistical Model Checking

Recall that a Markov Chain (MC) is a purely probabilistic model  $\mathcal{M} = (S, s_0, P)$ , where  $S$  is a set of states,  $s_0 \in S$  is the initial state, and  $P : S \times S \rightarrow [0, 1]$  is a probabilistic transition function that, given a pair of states  $(s_1, s_2)$ , yields the probability of moving from  $s_1$  to  $s_2$ .

Given a MC  $\mathcal{M}$ , one can define a probability measure on the infinite executions of  $\mathcal{M}$  using a standard construction based on the  $\sigma$ -algebra of cylinders.

A run of a MC is a sequence of states  $s_0, s_1 \dots$  such that for all  $i$ ,  $P(s_i, s_{i+1}) > 0$ . Given a finite run  $\rho = s_0 s_1 \dots s_l$ , its length, written  $|\rho|$  represents the number of transitions it goes through (including repetitions). Here  $|\rho| = l$ . We write  $\Gamma_{\mathcal{M}}(l)$  (or simply  $\Gamma(l)$  when  $\mathcal{M}$  is clear from the context) for the set of all finite runs of length  $l$ , and  $\Gamma_{\mathcal{M}}$  for all finite runs i.e.  $\Gamma_{\mathcal{M}} = \cup_{l \in \mathbb{N}} \Gamma_{\mathcal{M}}(l)$ . As usual we define the probability measure, written  $\mathbb{P}_{\mathcal{M}}$  on runs based on the sigma-algebra of cylinders (see e.g. [2]). This gives us that for any finite run  $\rho = s_0 s_1 \dots s_n$ ,  $\mathbb{P}_{\mathcal{M}}(\rho) = \prod_{i=1}^n P(s_{i-1}, s_i)$ . In the rest of the section, we only consider *finite* runs. Given a reward function  $r : \Gamma(l) \rightarrow \mathbb{R}$ , we write  $\mathbb{E}_{\mathcal{M}}^l(r)$  for the expected value of  $r$  on the runs of length  $l$  of a given MC  $\mathcal{M}$ .

Statistical Model Checking [23] is an approximation technique that allows to compute an estimation of the probability that a purely probabilistic systems satisfies a given property<sup>4</sup>. In particular, the Monte Carlo technique uses samples of the runs of length  $l$ ,  $\Gamma(l)$ , of a given Markov chain  $\mathcal{M}$  in order to estimate the

<sup>4</sup> Particular SMC techniques also allow to estimate the satisfaction of qualitative properties [18].

probability that  $\mathcal{M}$  satisfies a given bounded linear property. It can also be used for approximating the expected value of a given reward function  $r$  on the runs  $\Gamma(l)$  of  $\mathcal{M}$ . In order to provide some intuition, we briefly recall how standard Monte Carlo analysis works in the context of statistical model checking of MC. In this context, a set of  $n$  samples of the runs of the MC. These runs are generated at random using the probability distribution define through the Markov chain. Each of these samples is evaluated, yielding a reward value according to the reward function  $r$ . According to the law of large numbers (see e.g. [21]), the mean value of the samples provides a good estimator for the expected value of the reward function  $r$  on the runs of the given MC. Moreover, the central limit theorem provides a confidence interval that only depends on the number of samples (provided this number is large enough).

### 3.2 Parametric Markov Chains(pMC)

Markov Chains are inadequate in the context of drone flight plan analysis. Indeed, the models we develop in this context are subject to uncertainties that we model using parameters, such as precision of the position and attitude estimations and wind strength. The resulting models are therefore not purely probabilistic since they contain parameters. As a consequence, we need to use a more expressive type of model that allows to take into account probabilistic parameters, such as Parametric Markov Chains (see e.g. [1]).

A pMC is a tuple  $\mathcal{M} = (S, s_0, P, \mathbb{X})$  such that  $S$  is a finite set of states,  $s_0 \in S$  is the initial state,  $\mathbb{X}$  is a finite set of parameters, and  $P : S \times S \rightarrow Poly(\mathbb{X})$  is a parametric transition probability function, expressed as a polynomial on  $\mathbb{X}$ . A parameter valuation is a function  $v : \mathbb{X} \rightarrow [0, 1]$  that assigns values to parameters. A parameter valuation  $v$  is valid w.r.t. a given pMC  $\mathcal{M}$  if, when replacing parameters with their assigned values, the resulting object is a MC (i.e. the outgoing probabilities of all states sum up to 1). If  $v$  is a valid parameter valuation with respect to  $\mathcal{M}$ , the resulting Markov chain is written  $\mathcal{M}^v$ .

Given a pMC  $\mathcal{M}$ , a run  $\rho$  of  $\mathcal{M}$  is a sequence of states  $s_0 s_1 \dots$  such that for all  $i \geq 0$ ,  $P(s_i, s_{i+1}) \neq 0$  (i.e. the probability is either a strictly positive real constant or a function of the parameters). As for MCs, we write  $\Gamma_{\mathcal{M}}(l)$  for the set of all finite runs of length  $l$  and  $\Gamma_{\mathcal{M}}$  for the set of all finite runs.

Observe that for any valid parameter valuation  $v$ ,  $\Gamma_{\mathcal{M}^v}(l) \subseteq \Gamma_{\mathcal{M}}(l)$  since  $v$  may assign 0 to some transition probabilities.

### 3.3 Parametric SMC

As it is, standard SMC cannot be used in the context of pMC because of their parametric nature. Indeed, we cannot produce samples according to the parametric transition probabilities. Luckily, the underlying theory used in SMC can be extended in order to take into account parameters. The method we propose in the following is in line with a technique called *importance sampling* (see [21] for a description). The purpose of this technique is to sample a stochastic system using a chosen probability distribution (which is not the original distribution

present in this system) and “compensate” the results using a *likelihood ratio* in order to estimate a measure according to the original distribution. In the context of SMC, importance sampling has mainly been used in order to estimate the probability of rare events [3] and/or to reduce the number of required samples in order to obtain a given level of guarantee [14]. It has also been used in the context of parametric continuous-time Markov chains in order to estimate the value of a given objective function on the whole parameter space while using a reduced number of samples [4]. However, to the best of our knowledge, importance sampling has never been used in order to produce symbolic functions of the parameters as we do here.

The intuition of the method we propose here is to fix the transition probabilities to an arbitrary function  $f$ , which we call *normalization function*, and to use these transition probabilities in order to produce samples of the pMC  $\mathcal{M}$ . However, instead of evaluating the obtained runs by directly using the desired reward function  $r$ , we define a new (parametric) reward function  $r'$  that takes into account the parametric transition probabilities. We show that, under any parameter valuation  $v$ , the evaluation of the mean value of  $r'$  on the set of samples is a good estimator for the expected value of the reward  $r$  on  $\mathcal{M}^v$ . The central limit theorem (see e.g. [21]) also allows to produce parametric confidence intervals, but we do not go into details here (see [7] for more details on this topic).

**Remark.** The choice of the normalization function is crucial. In particular, the results presented below require that the graph structure of the MC obtained with this normalization function is identical to the graph structure of the MC obtained using the chosen parameter valuation. This is discussed in more details in [7]. In the following, we only consider parameter valuations that assign non-zero probability to parameterized transitions. Since we use the uniform normalization function, the graph structures of the obtained MCs are indeed identical, which ensures that the results presented below hold as expected.

Let  $Pa : \Gamma_{\mathcal{M}} \rightarrow Poly(\mathbb{X})$  be a parametric reward function. For any valid valuation  $v$  and any run  $\rho \in \Gamma_{\mathcal{M}^v}$  we have  $\mathbb{P}_{\mathcal{M}^v}(\rho) = Pa(\rho)(v)$ .

Given any valid normalization function  $f$  and any run  $\rho \in \Gamma_{\mathcal{M}}$ , let parametric reward function  $r'$  be  $r'(\rho) = \frac{Pa(\rho)}{\mathbb{P}_{\mathcal{M}^f}(\rho)} r(\rho)$ .

We now prove that the expected values are equal. Let  $\rho \in \Gamma_{\mathcal{M}^f}(l)$  be a random sample of  $\mathcal{M}^f$  and let  $Y$  be the random variable defined as follows  $Y = r'(\rho)$ . The following computation shows that, under any valid parameter valuation  $v$  such that  $\mathcal{M}^f$  and  $\mathcal{M}^v$  have the same structure, we have  $\mathbb{E}(Y)(v) = \mathbb{E}_{\mathcal{M}^v}^l(r)$ .

$$\begin{aligned}
\mathbb{E}(Y)(v) &= \left( \sum_{\rho \in \Gamma_{\mathcal{M}^f}(l)} \mathbb{P}_{\mathcal{M}^f}(\rho) y(\rho) \right)(v) \\
&= \left( \sum_{\rho \in \Gamma_{\mathcal{M}^f}(l)} \mathbb{P}_{\mathcal{M}^f}(\rho) \frac{Pa(\rho)}{\mathbb{P}_{\mathcal{M}^f}(\rho)} r(\rho) \right)(v) \\
&= \sum_{\rho \in \Gamma_{\mathcal{M}^f}(l)} Pa(\rho)(v) r(\rho) \\
&= \sum_{\rho \in \Gamma_{\mathcal{M}^f}(l)} \mathbb{P}_{\mathcal{M}^f}(\rho) r(\rho) \\
&= \sum_{\rho \in \Gamma_{\mathcal{M}^v}(l)} \mathbb{P}_{\mathcal{M}^f}(\rho) r(\rho) \\
&= \mathbb{E}_{\mathcal{M}^v}^l(r)
\end{aligned}$$

Our adaptation of the Monte Carlo technique for pMC is thus to estimate the expected value of  $Y$  in order to obtain a good estimator for the expectation of  $r$ . Let  $\rho_1, \dots, \rho_n$  be a set of  $n$  runs of length  $l$  of  $\mathcal{M}^f$ . Let  $Y_i$  be the random variable with values in  $Poly(\mathbb{X})$  such that  $Y_i = r'(\rho_i)$ . Notice that the  $Y_i$  are independent copies of the random variable  $Y$ .  $Y_i$  are therefore independent and identically distributed. Let  $\gamma$  be the parametric function giving their mean value. By the results above, for all valid parameter valuation  $v$  such that  $\mathcal{M}^v$  and  $\mathcal{M}^f$  have the same structure,  $\mathbb{E}_{\mathcal{M}^v}^l(r) = \mathbb{E}(Y)(v) = \mathbb{E}(\sum_{i=1}^n Y_i/n)(v) = \gamma(v)$ . Our parametric approximation of the expected value is therefore:

$$\hat{\gamma} = \sum_{i=1}^n Y_i/n.$$

In the sequel we will use Parametric Statistical Model Checking (PSMC) to check the formal model we will implement for the UAV.

## 4 Implementation, Experimentations and Results

While our complete formal model has been introduced in Section 2 in the form of an automata, we now explain how we successively implemented and improved the model by considering different formalisms and model checking tools. At each step, we show the limitations of the related model which leads to the next step of the implementation. The different steps of the model implementations are depicted in Figure 7.

To start, a first partial version of the formal model of Section 2.3 was implemented as a PRISM model using the PRISM tool [16], without parameters.

This first version, as depicted in 7a, corresponds to a very simple UAV flight plan, going in a straight line from point  $A$  to point  $B$  in  $T$  time units. In this context, the intermediate positions are estimated  $T * f$  times, where  $f$  is the frequency of the filter. The sizes used for the five security zones are respectively 20m, 40m, 60m, 80m and 100m.

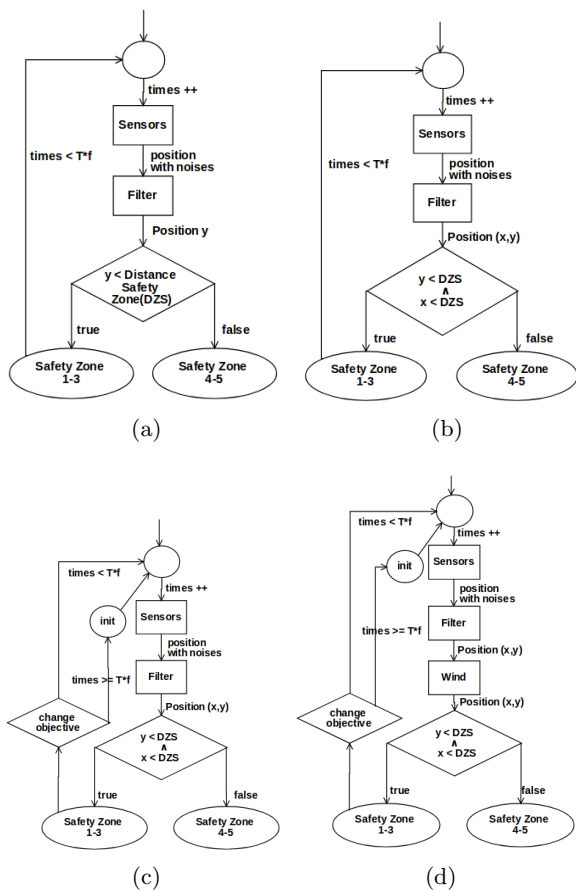


Fig. 7: Incremental development of the SMC model

using a flight controller plugged on a production line with a loop. We launched several runs of the device on the production line path and measured the outputs of the EKF filter. These measures then allowed us to compute the estimated position, which can then be compared to the exact position on the production line. We consequently obtained probabilities for the accuracy of the position estimation using an EKF filter and sensors coming from an industrial UAV. However, the major drawback of these experimentations is that they did not reflect a realistic UAV environment. In particular, since the experiment was conducted indoor using a fixed production line, the precision of some of the sensors (GPS for instance) is not representative of the precision one could obtain in a realistic flight environment. Although we were able to verify this model using PRISM, the results are not representative and can only be considered as a proof-of-concept. Since our aim is to study the same problem for

As explained in Section 2, the filter removes the noise corrupting data coming from sensors. In this first version, we only consider potential deviations along the  $y$ -axis. At each computation step, the inaccurate position given by the filter is computed using the accuracy of the filter and sensors (as a single real-valued variable), and compared to the intended position as given by the flight plan. The safety zone is deduced from the distance between the estimated position and the intended position. If the UAV enters Zones 4 or 5, the computation stops.

In this first model, the accuracy of the filter is probabilistic but not parametric, i.e. probability values have been encoded directly in the model. These values are the results of a set of experiments performed by

different accuracy probabilities, we changed the exact probability values to parameters and submitted this new model to the PRISM Model Checker. However, because of the real-valued variables used in the model and of the numerous intermediate computations, PRISM was not able to handle this model and timed-out after 2 hours of unsuccessful computations.

Facing these shortcomings with the PRISM tool, we considered the implementation of our model with the PARAM tool [12] which is a model checker for parametric discrete-time Markov chains. PARAM is efficient and allows to compute the probability of satisfying given properties as polynomials or rational functions of the parameters. As PRISM, PARAM also failed to model check our current version of the model. At this stage, since both PARAM and PRISM failed to verify our simplest model because of its complexity, we considered using a different approach based on Parametric Statistical Model Checking. For this purpose, we developed a prototype tool<sup>5</sup>. In this context, our model was expressed as a python program using real-valued variables both for the position of the UAV and for the probabilistic parameters. It appears that PSMC is particularly efficient in this context, and was able to verify our model (by performing more than 20k simulations) in less than 1 minute. We therefore chose to pursue our experimentation using this prototype tool and refined versions of our model.

In the second version of the model, depicted in Figure 7b, we allowed deviations to also occur along the  $x$ -axis. This is not problematic when considering a straight line flight plan, but could become important as soon as the flight plan is curved (as the one in Figure 1). Indeed, in this context, deviations along the  $x$ -axis (for example if the drone is "late") could result in the PID deciding to cut the trajectory, i.e. going straight to point  $C$  before reaching point  $B$ , therefore promoting a trajectory that might collide with the forbidden safety zones. Again, our tool managed to verify this model in a very short time.

For the third version of the model as depicted in Figure 7c, we add a third target point to the flight plan, which is not aligned with the first two points, i.e. like in Figure 1. In this third version, the inaccuracy of the position estimation along the  $x$ -axis also allows the UAV to be "late" and decide to cut the flight plan as explained above.

Finally, the last version, as depicted in Figure 7d, takes into account wind perturbations. We assumed here that the wind direction was constant but that the wind force was again parametric. This will allow us to study the right trade-off between filter capacity and frequency *depending on the weather conditions*. This last version is the most complex we studied, and therefore took more time to verify than the previous ones. With our prototype tool, it took 190 seconds to perform the verification using 10k simulations in this context while the same amount of simulations only took 28 seconds for the previous model (without wind parameters).<sup>6</sup>

<sup>5</sup> available at <https://github.com/paulinfournier/MCpMC>

<sup>6</sup> We do not share the exact models used in our prototype tool for confidentiality reasons, but the models used in PRISM and PARAM can be found here: <https://github.com/br4444/modelPrism/tree/master>

The outputs of our prototype tool are multivariate polynomials on the parameters of our model. Given the number of parameters, the size of the model and the length of the considered simulations, these polynomials are quite complex and therefore difficult to report in this paper. As an example, below is the output polynomial representing the probability that a UAV enters Zones 4 or 5 using our last version of the model:

$$\begin{aligned}
&0.43 * ProbaFilter_3 * ProbaWind_1 + 0.16 * ProbaFilter_3 * ProbaWind_2 \\
&+ 0.17 * ProbaFilter_3 * ProbaWind_3 + 0.28 * ProbaFilter_3 * ProbaWind_4 \\
&+ 0.85 * ProbaFilter_4 * ProbaWind_1 + \dots
\end{aligned}$$

Instead of showing the resulting polynomials, we will only present the evaluation of these polynomials using realistic values for the parameters. We defined two scenarios (Scenario 1, Scenario 2) with one set of values of parameters for each scenario. For these two scenarios, ProbaF0 (resp F1, F2, F3, F4) models the probability that the estimated position is from 0 to  $2m$  (resp.  $2-4m$ ,  $4-6m$ ,  $6-8m$ ,  $8-10m$ ) from the real position. In the first (resp. second) scenario, we have set these values to  $0.15/0.3/0.4/0.1/0.05$  (resp.  $0.1/0.25/0.35/0.2/0.1$ ). According to experiments done at PIXIEL, the first scenario is more realistic than the second one. Similarly, the wind parameters correspond to the probability of having a wind force of  $0-20km/h$ ,  $20-30km/h$ ,  $30-50km/h$  and  $50-70km/h$  respectively and have been set to  $0.55/0.43/0.01/0.01$  (which corresponds to typical weather conditions in Nantes, France) for the numerical evaluation. In both scenarios, Zone 4 (resp. 5) is situated  $8m$  (resp.  $50m$ ) from the flight plan.

In Table 1, we gather the results for running the simulation for the two considered scenarios; the simulation with PSMC is performed with 10k, 20k and 50k samples. Each time, a polynomial is computed and then evaluated using the parameter values given above. In order to illustrate the stability of our results despite their statistical nature, each complete scenario was performed two times (labelled V1 and V2 in the table). The value reported in the table represents the probability of the UAV eventually reaching Zones 4 or 5 during its flight. Experiments were performed using the formal models presented in Figure 7c (without wind) and Figure 7d (including wind perturbations) on a flight plan resembling the one shown in Figure 1, with a total flight duration of  $5s$  and a filter frequency of  $1Hz$ . We considered two versions of the model from Figure 7d: 7d(np) where wind strength is directly input as a constant probability in the model (resulting in a polynomial where the only variables represent the precision of position estimation), and 7d(p) where wind strength is input as parameter variables in the model (allowing to evaluate/optimize the resulting polynomial according to any wind strength). Remark that the results in the first case are more precise because there are less variables in the polynomial, and obtained in a more efficient manner. Depending on whether we are interested in specific or generic information concerning the weather environment, we can chose to use the first of the second version. Remark that the probabilities of entering the forbidden zones are quite high. This is not surprising as Zone 4 is situated  $8m$  from the intended trajectory and the precision of position estimation can be up



Table 1: Results of the experiments

	<b>Model</b>	<b>10k</b>		<b>20k</b>		<b>50k</b>	
		<b>V1</b>	<b>V2</b>	<b>V1</b>	<b>V2</b>	<b>V1</b>	<b>V2</b>
<b>Running time</b>	7c	28s		51-54s		142-143s	
Scenario 1	7c	4.99%	5.09%	4.74%	5.10%	4.91%	4.98%
Scenario 2	7c	10.38%	10.04%	9.82%	10.05%	9.95%	9.81%
<b>Running time</b>	7d(np)	28s		53-54s		149-155s	
Scenario 1	7d(np)	5.43%	5.31%	5.61%	5.21%	5.59%	5.47%
Scenario 2	7d(np)	10.8%	10.9%	10.8%	10.8%	10.9%	10.7%
<b>Running time</b>	7d(p)	185-190s		311-314s		612-621s	
Scenario 1	7d(p)	4.95%	5.97%	5.28%	6.62%	4.16%	5.61%
Scenario 2	7d(p)	9.55%	9.87%	10.3%	11.3%	9.57%	10.7%

to 10*m*. These values have been made deliberately high for the purpose of this study but can be chosen more realistically when verifying the real model.

## 5 Conclusion and Future Work

In this paper, we have presented a formal model to study the safety of a UAV in automatic flight following a predefined flight plan. This formal model consists in a parametric Markov Chain and takes that takes into account the precision of position and attitude estimation using sensors and filters as well as potential wind perturbations. We have also proposed a new verification technique for parametric probabilistic model: parametric Statistical Model Checking. This new technique has been implemented in a prototype tool. While state of the art tools such as PRISM and PARAM have timed out on the verification of the simplest version of our formal model, our prototype tool has been able to successfully verify the most complex version in less than 12 minutes.

In the future, we plan to keep enhancing our model in order to include filter frequency to be used as a parameter in the model. Using these parameters will allow us to obtain the parametric probability to enter dangerous zones depending on both the filter frequency and the precision probabilities. Studying/optimizing this parametric probability will allow PIXIEL to work on the trade-off between frequency and precision in order to choose their components wisely depending on their intended flight plan.

## References

1. Alur, R., Henzinger, T.A., Vardi, M.Y.: Parametric real-time reasoning. In: Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of

- Computing, May 16-18, 1993, San Diego, CA, USA. pp. 592–601 (1993). <https://doi.org/10.1145/167088.167242>, <https://doi.org/10.1145/167088.167242>
2. Baier, C., Katoen, J.: Principles of model checking. MIT Press (2008)
  3. Barbot, B., Haddad, S., Picaronny, C.: Coupling and importance sampling for statistical model checking. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 331–346. Springer (2012)
  4. Bortolussi, L., Milios, D., Sanguinetti, G.: Smoothed model checking for uncertain continuous-time markov chains. *Information and Computation* **247**, 235–253 (2016)
  5. Chang, Y.H., Hu, Q., Tomlin, C.J.: Secure estimation based Kalman Filter for cyber-physical systems against sensor attacks. *Automatica* **95**, 399–412 (2018). <https://doi.org/10.1016/j.automatica.2018.06.010>, <https://doi.org/10.1016/j.automatica.2018.06.010>
  6. de Marina, H.G., Pereda, F.J., Giron-Sierra, J.M., Espinosa, F.: UAV Attitude Estimation Using Unscented Kalman Filter and TRIAD. *IEEE Transactions on Industrial Electronics* **59**(11), 4465–4474 (Nov 2012). <https://doi.org/10.1109/TIE.2011.2163913>
  7. Delahaye, B., Fournier, P., Lime, D.: Statistical model checking for parameterized models (Feb 2019), <https://hal.archives-ouvertes.fr/hal-02021064>, working paper or preprint
  8. Euston, M., Coote, P., Mahony, R., Kim, J., Hamel, T.: A complementary filter for attitude estimation of a fixed-wing UAV. In: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 340–345 (Sep 2008). <https://doi.org/10.1109/IROS.2008.4650766>
  9. Freddi, A., Longhi, S., Monteri, A.: A model-based fault diagnosis system for unmanned aerial vehicles. *IFAC Proceedings Volumes* **42**(8), 71 – 76 (2009). <https://doi.org/https://doi.org/10.3182/20090630-4-ES-2003.00012>, <http://www.sciencedirect.com/science/article/pii/S147466701635755X>, 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes
  10. Gasiór, P., Bondyra, A., Gardecki, S.: Development of Vertical Movement Controller for Multirotor UAVs. In: Szewczyk, R., Zielinski, C., Kaliczynska, M. (eds.) *Automation 2017 - Innovations in Automation, Robotics and Measurement Techniques*, Proceedings of AUTOMATION 2017, March 15-17, 2017, Warsaw, Poland. *Advances in Intelligent Systems and Computing*, vol. 550, pp. 339–348. Springer (2017). [https://doi.org/10.1007/978-3-319-54042-9\\_31](https://doi.org/10.1007/978-3-319-54042-9_31), [https://doi.org/10.1007/978-3-319-54042-9\\_31](https://doi.org/10.1007/978-3-319-54042-9_31)
  11. Gonzalez, L.F., Montes, G.A., Puig, E., Johnson, S., Mengersen, K.L., Gaston, K.J.: Unmanned aerial vehicles (uavs) and artificial intelligence revolutionizing wildlife monitoring and conservation. *Sensors* **16**(1), 97 (2016). <https://doi.org/10.3390/s16010097>, <https://doi.org/10.3390/s16010097>
  12. Hahn, E.M., Hermanns, H., Wachter, B., Zhang, L.: PARAM: A Model Checker for Parametric Markov Models. In: Touili, T., Cook, B., Jackson, P. (eds.) *Computer Aided Verification*. pp. 660–664. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
  13. Heredia, G., Caballero, F., Maza, I., Merino, L., Viguria, A., Ollero, A.: Multi-Unmanned Aerial Vehicle (UAV) Cooperative Fault Detection Employing Differential Global Positioning (DGPS), Inertial and Vision Sensors. *Sensors* **9**(9), 7566–7579 (2009). <https://doi.org/10.3390/s90907566>, <https://doi.org/10.3390/s90907566>

14. Jegourel, C., Legay, A., Sedwards, S.: Cross-entropy optimisation of importance sampling parameters for statistical model checking. In: International Conference on Computer Aided Verification. pp. 327–342. Springer (2012)
15. Kwiatkowska, M., Norman, G., Parker, D.: PRISM: Probabilistic Model Checking for Performance and Reliability Analysis. *ACM SIGMETRICS Performance Evaluation Review* **36**(4), 40–45 (2009)
16. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) Proc. 23rd International Conference on Computer Aided Verification (CAV'11). LNCS, vol. 6806, pp. 585–591. Springer (2011)
17. Kyriatsis, S., Antonopoulos, A., Chanielakis, T., Stefanakis, E., Linardos, C., Tripolitsiotis, A., Partsinevelos, P.: Towards Autonomous Modular UAV Missions: The Detection, Geo-Location and Landing Paradigm. *Sensors* **16**(11), 1844 (2016). <https://doi.org/10.3390/s16111844>, <https://doi.org/10.3390/s16111844>
18. Legay, A., Delahaye, B., Bensalem, S.: Statistical model checking: An overview. In: Proc. Runtime Verification - First International Conference, RV 2010, St. Julians, Malta, November 1-4, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6418, pp. 122–135. Springer (2010)
19. Madgwick, S.O.H.: An efficient orientation filter for inertial and inertial / magnetic sensor arrays (2010)
20. Máthé, K., Busoniu, L.: Vision and Control for UAVs: A Survey of General Methods and of Inexpensive Platforms for Infrastructure Inspection. *Sensors* **15**(7), 14887–14916 (2015). <https://doi.org/10.3390/s150714887>, <https://doi.org/10.3390/s150714887>
21. Rubinstein, R.Y., Kroese, D.P.: Simulation and the Monte Carlo method, vol. 10. John Wiley & Sons (2016)
22. Sabatelli, S., Galgani, M., Fanucci, L., Rocchi, A.: A Double-Stage Kalman Filter for Orientation Tracking With an Integrated Processor in 9-D IMU. *IEEE Transactions on Instrumentation and Measurement* **62**(3), 590–598 (March 2013). <https://doi.org/10.1109/TIM.2012.2218692>
23. Sen, K., Viswanathan, M., Agha, G.: On statistical model checking of stochastic systems. In: International Conference on Computer Aided Verification. pp. 266–280. Springer (2005)
24. Zhou, Z., Ding, J., Huang, H., Takei, R., Tomlin, C.: Efficient path planning algorithms in reach-avoid problems. *Automatica* **89**, 28–36 (2018). <https://doi.org/10.1016/j.automata.2017.11.035>, <https://doi.org/10.1016/j.automata.2017.11.035>