Introducing Probabilistic Reasoning within Event-B

Mohamed Aouadhi, Benoît Delahaye, Arnaud Lanoix

International Journal on Software and Systems Modeling manuscript No.
(will be inserted by the editor)

Introducing Probabilistic Reasoning within Event-B

Mohamed Amine Aouadhi[1], **Benoît Delahaye**[1], **Arnaud Lanoix**[1]

University of Nantes / LS2N UMR CNRS 6004

Received: date / Revised version: date

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.

1 Introduction

As systems become more and more complex, with randomised algorithms [31], probabilistic protocols [5] 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 [15,38], continuous evolution, energy consumption etc. 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 [34,

Send offprint requests to:

32]. Abstraction [27,17], refinement [26] and model-checking algorithms [13,10] 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 [19,33,22,18,11,25,7,24]. 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 [3] is a proof-based formal method used for discrete systems modelling. It is equipped with *Rodin* [4], 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.

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 [30], 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 [20], Hallerstede *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 [40], the same authors study the refinement of qualitative probabilistic Event-B models and propose a tool support inside Rodin. Other works [35,37,36] have extended this approach by refining non-deterministic assignments into *quantitative* probabilistic assignments where, unlike in [20], 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

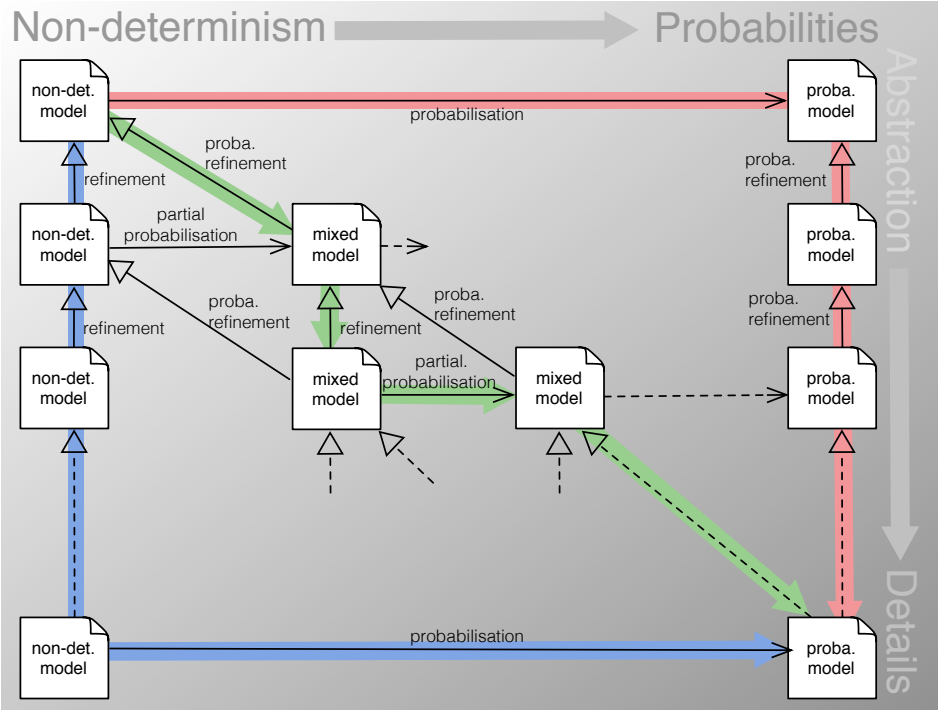

Figure 1: How to introduce probabilities within Event-B?

need to be included in the refinement chain when designers need it, which would certainly be counter-intuitive 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 Figure 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. Fig. 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, 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 steps

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 [29,22] in the context of probabilistic programs and the standard B method, and in [23,20] in the context of non-deterministic Event-B models with only probabilistic assignments. While the authors of [29,22] 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, [23,20] focus 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 [20] 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 [6], where we have first introduced a fully probabilistic extension of Event-B. Compared to [6], 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, probabilistic). This paper also provides new discussions, figures and examples illustrating in better details contributions from [6].

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 Section 3 introduces our running case-study: a simple Peer-To-Peer protocol. In Section 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 Sections 6 and 7. Finally, Section 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 $\delta : S \to [0,1]$ such that $\sum_{s \in S} \delta(s) = 1$.

Labelled Transition System [10] A labelled transition system (LTS for short) is a tuple $\mathcal{M}$=(S, $Acts$, $\to$, s_0, AP, L) where S is a set of states, $Acts$ is a set of actions, $\to \subseteq S \times Acts \times S$ is a transition relation, $s_0 \subseteq S$ is the initial state, AP is a set of atomic propositions, and $L : S \to 2^{AP}$ is a labelling function.

Probabilistic Labelled Transition System [10] 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: $S \to AP$ is a labelling function, and P: $S \times Acts \times S \to [0,1]$ 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 [10] 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 $L : S \to 2^{AP}$ is a labelling function. An action α 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 [3] 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=($\bar{v}$,$\mathsf{I}(\bar{v})$, $\mathsf{V}(\bar{v})$, Evts, Init) where $\bar{v}$= $\{\mathsf{v}_1 \ldots \mathsf{v}_n\}$ is a set of variables, $\mathsf{I}(\bar{v})$ is an invariant, $\mathsf{V}(\bar{v})$ is an (optional) variant used for proving the convergence of the model, Evts is a set of events and $\mathsf{Init} \in \mathsf{Evts}$ is the initialisation event. The invariant $\mathsf{I}(\bar{v})$ is a conjunction of predicates over the variables of the system specifying properties that must always hold.

Events An event has the following structure (see Figure on the side), where e_i is the name of the event, $\bar{t} = \{t_1 \ldots t_n\}$ 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:

$e_i \mathrel{\widehat{=}}$
any $\bar{t}$ **where**
$G_i(\bar{t},\bar{v})$
then
$S_i(\bar{t},\bar{v})$
end

- **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**: $x :\in \{E_1(\bar{t},\bar{v}) \ldots E_n(\bar{t},\bar{v})\}$ means that the variable x is assigned a new value taken from the set $\{E_1(\bar{t},\bar{v}) \ldots E_n(\bar{t},\bar{v})\}$.

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 $x' \in \{E_1(\bar{t},\bar{v}) \ldots E_n(\bar{t},\bar{v})\}$.

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 $v_1 \ldots v_i$ are the variables assigned in $S_j(\bar{t},\bar{v})$ – variables $v_{i+1} \ldots v_n$ are thus not modified – and let $Q(\bar{t},\bar{v},v_1,v'_1) \ldots Q(\bar{t},\bar{v},v_i,v'_i)$ 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:

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

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 [3]. 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:

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

Operational semantics As established in [12], the semantics of an Event-B model M=($\bar{v}$,I($\bar{v}$),V($\bar{v}$),Evts, Init) is expressed in terms of a labelled transition system (LTS) $\mathcal{M}$=(S, $Acts$, T, s_0, AP, L) where S is a set of states, each state in S being uniquely identified by its label; $Acts$ 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 $T \subseteq S \times Acts \times S$ 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 work have focused in the development of the theory of refinement [9,8,29]: 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 [21,16]). 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 [8] 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($\bar{v}$,$\bar{w}$),Evts$_c$, Init $_c$) may refine an abstract model M=($\bar{v}$,I($\bar{v}$),V($\bar{v}$),Evts, 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 ($\bar{w}$), Evts$_c$ is the set of concrete events that contains both the refined events (Evts$_c$ $\cap$ Evts$_a$) and the new events introduced by refinement (Evts$_c$ $\setminus$ Evts$_a$), and Init $_c$ is the concrete initialisation event.

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

In this paper, we do not treat the cases of event *splitting* and *merging*, 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 $\mathsf{Wi}_c(\bar{t},\bar{u},\bar{v},\bar{v}',\bar{w},\bar{w}')$ 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 $\bar{w}'$.

$\mathsf{e}_a \mathrel{\widehat{=}}$
any $\bar{t}$ **where**
$\mathsf{G}_a(\bar{t},\bar{v})$
then
$\mathsf{S}_a(\bar{t},\bar{v})$
end

e_c $\mathrel{\widehat{=}}$
refines e_{i+1}
any $\bar{u}$ **where**
G_c($\bar{u},\bar{w}$)
with
Wi_c($\bar{t},\bar{u},\bar{v},\bar{v}',\bar{w},\bar{w}'$)
then
S_c($\bar{u},\bar{w}$)
end

As the Event-B refinement process allows a more detailed description of the execution of the system, it is necessary to introduce new events ($\mathsf{Evts}_c \setminus \mathsf{Evts}_a$) 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 [3]. 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:

$$\mathsf{I}(\bar{v}) \wedge \mathsf{J}(\bar{v},\bar{w}) \wedge \mathsf{G}_c(\bar{u},\bar{w}) \wedge \mathsf{Wi}_c(\bar{t},\bar{u},\bar{v},\bar{v}',\bar{w},\bar{w}') \vdash \mathsf{G}_a(\bar{t},\bar{v}) \qquad \text{(grd/STRENGTH)}$$

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

$$\mathsf{I}(\bar{v}) \wedge \mathsf{J}(\bar{v},\bar{w}) \wedge \mathsf{G}_c(\bar{u},\bar{w}) \wedge \mathsf{S}_c(\bar{u},\bar{w},\bar{w}') \vdash \mathsf{S}_a(\bar{t},\bar{v},\bar{v}') \qquad \text{(act/SIM)}$$

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 $\mathsf{V}(\bar{v})$, called a *variant*, and show that all convergent events strictly decrease the value of this variant. As a conse-

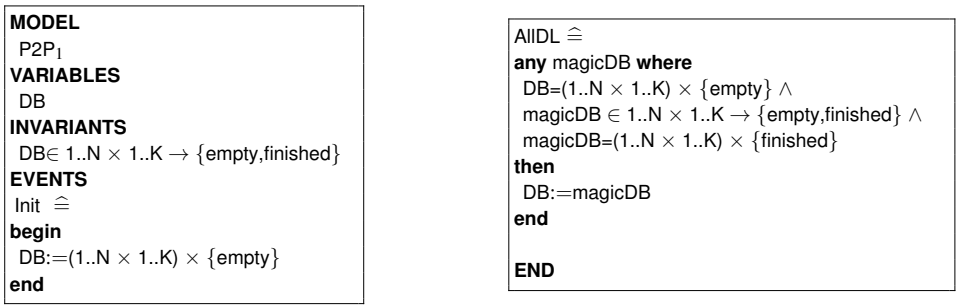

Figure 2: Initial Event-B model of the simple P2P protocol

quence, 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.

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

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

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

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 Bittorent [1]. The description of the complete case study can be found in [2]. 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∈1..N and each block b∈1..K whether the client has already downloaded the block (DB(c↦b)=finished) or not (DB(c↦bc=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. magicDB is a parameter chosen in such a way that for all client c ∈ 1..N and block b ∈ 1..K, we have magicDB(c1↦b1)=finished. The substitution DB := 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

MODEL
$P2P_2$
REFINES
$P2P_1$

VARIABLES
DB
DBin

INVARIANTS
$DBin \in 1..N \times 1..K \rightarrow \{empty,incoming,finished\} \wedge$
$\forall c \cdot (c \in 1..N \Rightarrow$
$card(\{b \mid b \in 1..K \wedge DBin(c \mapsto b)=incoming\}) \leq 1)$

VARIANT
$2 \times N*K$
$- 2 \times card(\{c \mapsto b \mid n \in 1..N \wedge b \in 1..K \wedge DBin(c \mapsto b) = finished\})$
$- card(\{c \mapsto b \mid n \in 1..N \wedge b \in 1..K \wedge DBin(c \mapsto b) = incoming\})$

EVENTS

Init $\widehat{=}$
begin
$DB := (1..N \times 1..K) \times \{empty\} \parallel$
$DBin := (1..N \times 1..K) \times \{empty\}$
end

DLFinished $\widehat{=}$
refines AllDL
when
$DB=(1..N \times 1..K) \times \{empty\} \wedge$
$DBin=(1..N \times 1..K) \times \{finished\}$
then
DB:=DBin
end

Start1DL $\widehat{=}$
any c, b **where**
$c \in 1..N \wedge b \in 1..K \wedge$
$DBin(c \mapsto b)=empty \wedge$
$card(\{k \mid k \in 1..K \wedge DBin(c \mapsto k)=incoming\})=0$
then
$DBin(c \mapsto b):=incoming$
end

Finish1DL $\widehat{=}$
any c, b **where**
$c \in 1..N \wedge b \in 1..K \wedge$
$DBin(c \mapsto b)=incoming$
then
$DBin(c \mapsto b):=finished$
end
END

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

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) $\times$ {empty}. Furthermore, we impose that each client c is not currently trying to download more than one block c, 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. 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($c \mapsto b$):=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($c \mapsto b$)=empty.

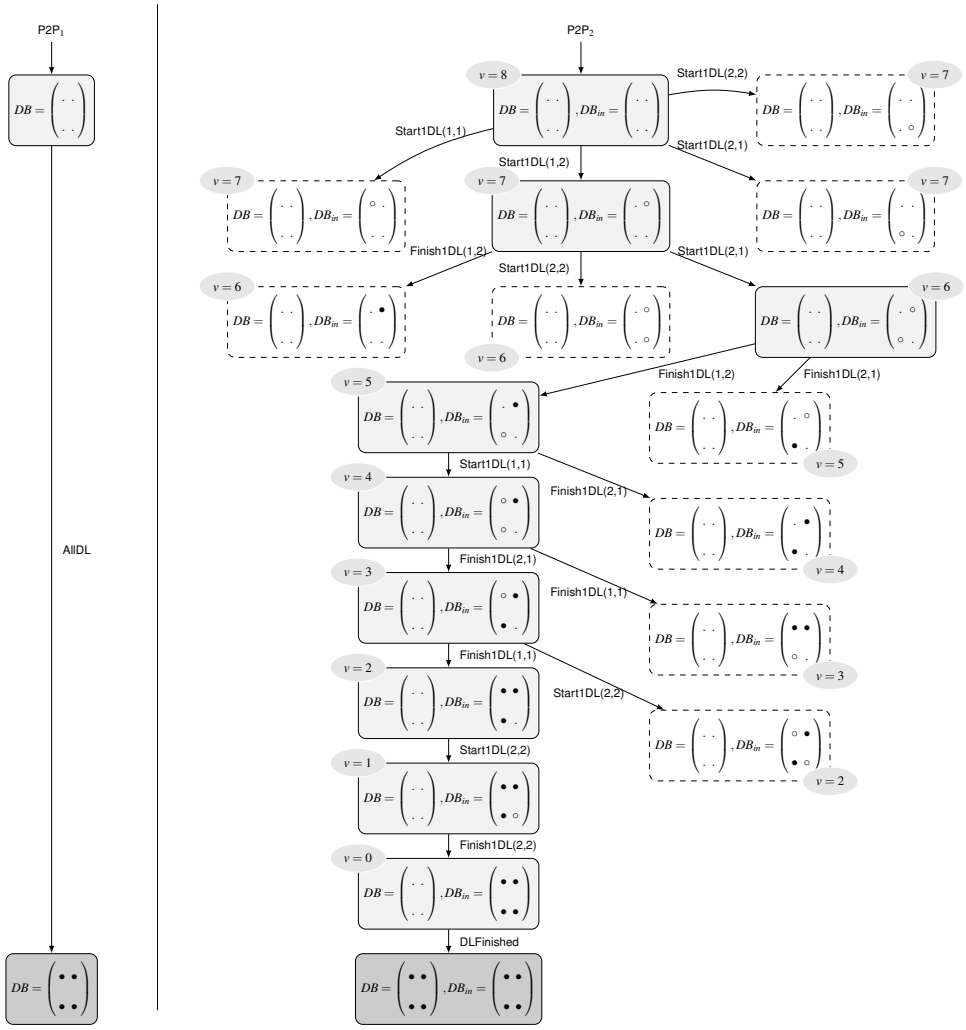

Figure 4: Extract of the transition system of the first refinement of the simple P2P protocol, with N=2 and K=2

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 (resp. 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.

Introducing failures In this refinement $\mathsf{P2P}_3$, 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 $\{\mathsf{empty}, \mathsf{incoming}\}$: 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 models the uncontrolled occurrence of failures. An extract of the operational semantics of this new model is provided in Figure 6.

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

```
FailureDL ≙
any c, b where
 c ∈ 1..N ∧ b ∈ 1..K ∧
 DBin(c↦b)=incoming
then
 DBin(c↦b):∈{empty,incoming}
end
```

Figure 5: New event introduced in the second refinement step

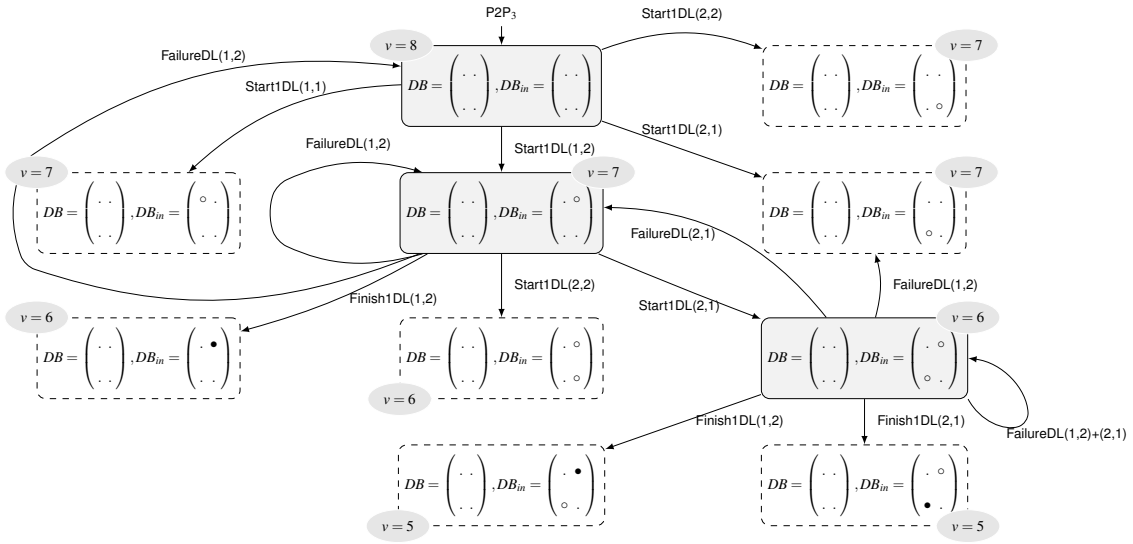

Figure 6: Extract of the transition system of the second refinement of the simple P2P protocol, with N=2 and K=2

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 $\mathsf{W}_i(\bar{v})$ of a probabilistic event e_i as an expression over the variables $\bar{v}$ 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

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

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 plug-in providing capabilities to define and use mathematical extensions to the Event-B language and the proving infrastructure [14].

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

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 on the side 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

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

MODEL
P2P$_P$

VARIABLES
DB
DBin

INVARIANTS
DB$\in 1..N \times 1..K \rightarrow \{empty,finished\} \wedge$
DBin $\in 1..N \times 1..K \rightarrow \{empty,incoming,finished\} \wedge$
$\forall$ c . (c $\in$ 1..N
$\Rightarrow$ card($\{$b | b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b)=incoming$\}$) $\leq$ 1)

VARIANT
2 $\times$ N$*$K
$-$ 2 $\times$ card($\{$c$\mapsto$b |n$\in$1..N $\wedge$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b) = finished$\}$)
$-$ card($\{$c$\mapsto$b |n$\in$1..N $\wedge$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b) = incoming$\}$)

EVENTS

Init $\widehat{=}$
begin
DB:=(1..N $\times$ 1..K) $\times$ $\{$empty$\}$ ||
DBin:=(1..N $\times$ 1..K) $\times$ $\{$empty$\}$
end

DLFinished $\widehat{=}$
weight N$*$K
when
DB=(1..N $\times$ 1..K) $\times$ $\{$empty$\}$ $\wedge$
DBin=(1..N $\times$ 1..K) $\times$ $\{$finished$\}$
then
DB:=DBin
end

Start1DL $\widehat{=}$
weight
N$*$K
$-$ card($\{$c$\mapsto$b |n$\in$1..N $\wedge$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b)=incoming$\}$)
any c, b **where**
c $\in$ 1..N $\wedge$ b $\in$ 1..K $\wedge$
DBin(c$\mapsto$b)=empty $\wedge$
card($\{$k | k $\in$ 1..K $\wedge$ DBin(c$\mapsto$k)=incoming$\}$)=0
then
DBin(c$\mapsto$b):=incoming
end

Finish1DL $\widehat{=}$
weight
card($\{$c$\mapsto$b |n$\in$1..N $\wedge$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b)=finished$\}$) +1
any c, b **where**
c $\in$ 1..N $\wedge$ b $\in$ 1..K $\wedge$
DBin(c$\mapsto$b)=incoming
then
DBin(c$\mapsto$b):=finished
end

FailureDL $\widehat{=}$
weight
N$*$K
$-$ card($\{$c$\mapsto$b |n$\in$1..N $\wedge$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b)=finished$\}$)
any c, b **where**
c $\in$ 1..N $\wedge$ b $\in$ 1..K $\wedge$
DBin(c$\mapsto$b)=incoming
then
DBin(c$\mapsto$b):=empty @4/10 $\oplus$ incoming @6/10
end

END

Figure 7: Probabilistic version of the simple P2P protocol

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($\bar{v}$), PEvts, Init) where $\bar{v}$= $\{v_1 \ldots v_n\}$ is a set of variables, I($\bar{v}$) is the invariant, PEvts is a set of probabilistic events and Init is the initialisation event.*

Running Example A probabilistic version of the P2P model from Section 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.

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

$$\mathsf{I}(\bar{v}) \wedge \mathsf{G}_i(\bar{t},\bar{v}) \vdash \mathsf{W}_i(\bar{v}) \in \mathsf{NAT} \qquad \text{(event/WGHT/NAT)}$$

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.

$$\mathsf{I}(\bar{v}) \vdash \mathsf{finite}\,(\{\bar{t} \mid \mathsf{G}_i(\bar{t},\bar{v})\}) \qquad \text{(event/param/pWD)}$$

Enumerated probabilistic assignments well-definedness and feasibility In all enumerated probabilistic assignments, it is necessary to ensure that the discrete probability values $\mathsf{p}_1 \ldots \mathsf{p}_n$ 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.

$$\vdash 0 < \mathsf{p}_i \leq 1 \qquad \text{(event/assign/pWD1)}$$

2. The sum of the probability values $\mathsf{p}_1 \ldots \mathsf{p}_n$ in enumerated probabilistic assignments must be equal to 1.

$$\vdash \mathsf{p}_1 + \ldots + \mathsf{p}_n = 1 \qquad \text{(event/assign/pWD2)}$$

Feasibility of enumerated probabilistic assignments is trivial: as soon as at least one expression $\mathsf{E}_i(\bar{t},\bar{v})$ 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 $\mathsf{Q}_x(\bar{t},\bar{v},\mathsf{x}')$ of the corresponding assignment satisfied, we impose that this set must be finite.

$$\mathsf{I}(\bar{v}) \wedge \mathsf{G}_i(\bar{t},\bar{v}) \wedge \mathsf{W}_i(\bar{v})>0 \vdash \text{finite}\,(\{\mathsf{x}' \mid \mathsf{Q}_x(\bar{t},\bar{v},\mathsf{x}')\}) \qquad \text{(event/assign/pWD3)}$$

Feasibility of predicate probabilistic assignments is ensured by the standard feasibility PO [3] inherited from Event-B. It ensures that the set $\{\mathsf{x}' \mid \mathsf{Q}_x(\bar{t},\bar{v},\mathsf{x}')\}$ 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.

$$\mathsf{I}(\bar{v}) \wedge \mathsf{G}_i(\bar{t},\bar{v}) \wedge \mathbf{W}_i(\bar{v}) > \mathbf{0} \wedge \mathsf{SP}_i(\bar{t},\bar{v},\bar{v}') \vdash \mathsf{I}(\bar{v}') \qquad \text{(event/pINV)}$$

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

$$\mathsf{I}(\bar{v}) \vdash (\mathsf{G}_1(\bar{t},\bar{v}) \wedge \mathbf{W}_1(\bar{v}) > \mathbf{0}) \vee \ldots \vee (\mathsf{G}_n(\bar{t},\bar{v}) \wedge \mathbf{W}_n(\bar{v}) > \mathbf{0}) \qquad \text{(model/pDLF)}$$

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 re-write 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 $\mathsf{P2P}_P$ 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 $\mathsf{P2P}_P$ is therefore *consistent*.

4.3 Semantics

As explained in Section 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 (DTMC).

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, Init) be a fully probabilistic Event-B model and σ be a valuation of its variables. Given a variable $x \in \bar{v}$, we write $[\sigma]x$ for the value of x in σ. 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 σ. 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 θ and variable valuation σ.

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

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 ($x := E_1(\bar{t},\bar{v})@_{p_1} \oplus \ldots \oplus E_m(\bar{t},\bar{v})@_{p_m}$ $(m \geq 1)$), 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.

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

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 $e_i \in \mathsf{PEvts}$ be a probabilistic event, $x \in Var(e_i)$ be a variable, σ, σ' two valuations of the variables $\bar{v}$ and θ a valuation of the parameter values associated to the event e_i such that e_i is enabled in the context of σ and θ, and leads the system to σ'.

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

Formally,

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

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 $\mathcal{V}^{e_i}_{\theta,\sigma}(x)$ for the set of values x' that make the predicate $Q_x(\bar{t},\bar{v},x')$ true when evaluated in σ and θ.

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

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

Let e_i be a probabilistic event and let x be a variable in $Var(e_i)$, given an original valuation σ of the variables, a valuation θ of the parameters of e_i and a target valuation σ' of the variables, we write $P^{e_i}_{\sigma,\theta}(x,\sigma')$ for the probability that x is assigned the new value $[\sigma']x$ when executing e_i from the valuation σ and with parameter valuation θ. If e_i is not equipped with parameters, this is written $P^{e_i}_{\sigma}(x,\sigma')$. 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:

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

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

$$P^{e_i}_{\sigma,\theta}(x,\sigma') = \frac{1}{card(\mathcal{V}^{e_i}_{\theta,\sigma}(x))} \text{ if } [\sigma']x \in \mathcal{V}^{e_i}_{\theta,\sigma}(x) \text{ and 0 otherwise.}$$

4.3.2 DTMC operational semantics Informally, the operational semantics of a fully probabilistic Event-B model M=($\bar{v}$, I($\bar{v}$), PEvts, Init) is a Probabilistic LTS $[\![M]\!] = (S, \mathsf{Acts}, P, s_0, AP, L)$ 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 $e_i \in \mathsf{PEvts}$ be a probabilistic event, $x \in \bar{v}$ be a variable and s, s' be two states of $[\![M]\!]$ 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 θ, and ***(3)*** the overall probability that each modified variable is assigned the value given in s' under parameter valuation θ.

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

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

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 Section 4.2.1 is a DTMC.*

Proof We must prove that for each state s in $[\![M]\!]$, the sum of probabilities of the outgoing transitions from s is equal to one. Let M be a fully probabilistic Event-B model, $\bar{v} = (x_1, x_2, ..., x_n)$ the set of variables of M and $s \in S$ a state of $[\![M]\!]$. 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 $e_i \notin \mathsf{Acts}(s)$ or $\exists x \in \bar{v} \backslash \{Var(e_i)\} \mid [s]x \neq [s']x$ and otherwise:

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

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

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 $0 \leq \frac{[s]W_i(\bar{v})}{\sum_{e_j \in \mathsf{Acts}(s)} [s]W_j(\bar{v})} \leq 1$,

- (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 $0 \leq \prod_{x \in Var(e_i)} P_{s,\theta}^{e_i}(x,s') \leq 1$ for all $\theta \in T_s^{e_i}$.

Moreover, (2) is derived as follows:

By definition,

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

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

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

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

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

We now reduce the expression $(\mathbf{A})$. For all $x \in Var(e_i)$, 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 $P_{s,\theta}^{e_i}(x,s') = \frac{1}{card(\mathcal{V}_{\theta,s}^{e_i}(x))}$ 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 $x \in Var(e_i)$.

For $\bar{v} = \{x_1, \ldots, x_n\}$, we have $S_1 = \{(v'_{x_1}, \ldots, v'_{x_n}) | v'_{x_i} \in X_i$ if $x_i \in Var(e_i)$ and $v'_{x_i} = [s]x_i$ otherwise$\}$. Assuming that $Var(e_i) = \{x_1, \ldots, x_k\}$ with $k \leq n$, we can therefore re-write $(\mathbf{A})$ as follows:

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

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 $x_i \in Var(e_i)$, therefore $(\mathbf{A}) = \prod_{x_i \in Var(e_i)} [\sum_{v'_{x_i} \in X_i} F_{x_i}^{s,\theta,e_i}(v'_{x_i})] = 1$

As a consequence,

$$\sum_{s' \in S, e_i \in \mathsf{Acts}(s)} P(s, e_i, s') = \sum_{e_i \in \mathsf{Acts}(s)} \frac{[s]W_i(\vec{v})}{\sum_{e_j \in \mathsf{Acts}(s)} [s]W_j(\vec{v})} \times \sum_{\theta \in T_s^{e_i}} (P_{T_s^{e_i}}(\theta))$$

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

$$\sum_{s' \in S, e_i \in \mathsf{Acts}(s)} P(s, e_i, s') = \sum_{e_i \in \mathsf{Acts}(s)} \frac{[s]W_i(\vec{v})}{\sum_{e_j \in \mathsf{Acts}(s)} [s]W_j(\vec{v})} = 1$$

As a conclusion, $P(s,.,.)$ is indeed a probability distribution on $\mathsf{Acts}(s) \times S$ for all $s \in S$ and therefore $[\![M]\!]$ is a DTMC. □

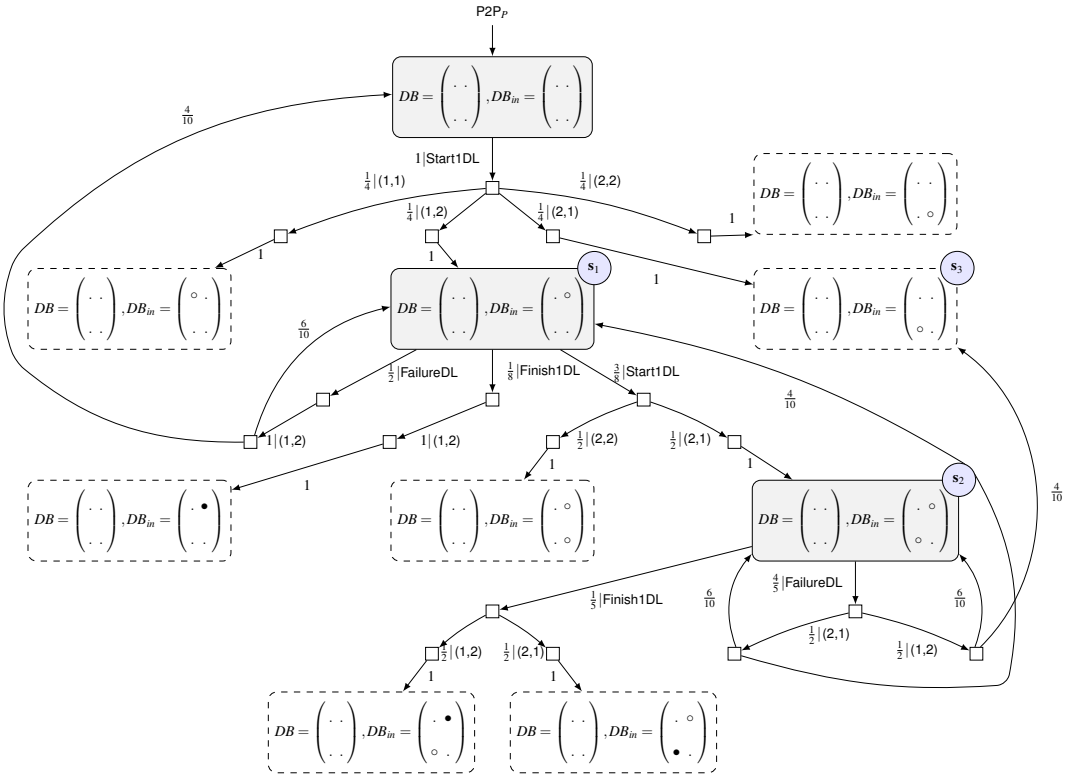

Figure 8: Extract of the Detailed construction of the DTMC of the simple P2P protocol, with N=2 and K=2

4.3.3 Running Example Fig. 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 $(\mathbf{s_1})$ on Fig. 8, three events are enabled:

- 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 $(\mathbf{s_1})$ 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 $(\mathbf{s_2})$. As a consequence, the global probability of reaching $(\mathbf{s_2})$ from $(\mathbf{s_1})$ 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 $(\mathbf{s_1})$ and $\frac{6}{10}$ of going to another state). As a consequence, the global probability of looping on $(\mathbf{s_1})$ using the event FailureDL is $\frac{1}{2} \times 1 \times \frac{6}{10} = \frac{3}{10}$.

From the state referenced as $(\mathbf{s_2})$ 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 $(\mathbf{s_1})$ is $\frac{4}{5} \times \frac{1}{2} \times \frac{4}{10} = \frac{4}{25}$,
- the global probability of reaching $(\mathbf{s_3})$ is $\frac{4}{5} \times \frac{1}{2} \times \frac{4}{10} = \frac{4}{25}$,
- the global probability of looping on $(\mathbf{s_2})$ is $\frac{4}{5} \times (\underbrace{\frac{1}{2} \times \frac{6}{10}}_{(2,1)} + \underbrace{\frac{1}{2} \times \frac{6}{10}}_{(1,2)}) = \frac{12}{25}$, 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 [35,36,37] and in Section 4 by allowing prob-

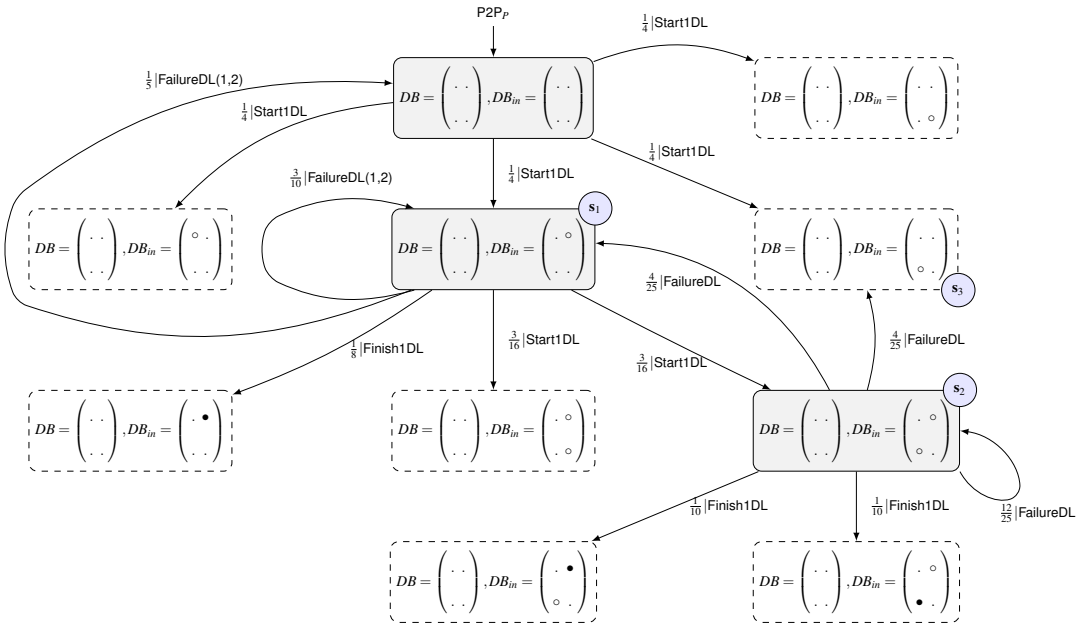

Figure 9: Extract of the DTMC of the simple P2P protocol, with N=2 and K=2

abilistic 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 Section 4. Therefore, events in a mixed Event-B model have one of the forms presented on the side.

$e_i \mathrel{\widehat{=}}$
any $\bar{t}$ **where**
$G_i(\bar{t},\bar{v})$
then
$S_i(\bar{t},\bar{v})$
end

or

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

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* $M=(\bar{v}, I(\bar{v}), MEvts, Init)$ *where* $\bar{v}=\{v_1 \ldots v_n\}$ *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 Section 4.2.

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

$$\mathsf{I}(\bar{v}) \ \wedge \mathsf{G}_1(\bar{t},\bar{v}) \vee \ldots \vee \mathsf{G}_i(\bar{t},\bar{v}) \vdash \neg \ ((\mathsf{G}_{i+1}(\bar{t},\bar{v}) \wedge \mathsf{W}_{i+1}(\bar{v})>0) \vee \ldots \vee (\mathsf{G}_n(\bar{t},\bar{v}) \ \wedge \mathsf{W}_n(\bar{v})>0)) \qquad \text{(mixedEB/csrt1)}$$

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:

$$\mathsf{I}(\bar{v}) \ \wedge ((\mathsf{G}_{i+1}(\bar{t},\bar{v}) \wedge \mathsf{W}_{i+1}(\bar{v})>0) \vee \ldots \vee (\mathsf{G}_n(\bar{t},\bar{v}) \ \wedge \mathsf{W}_n(\bar{v}))) \vdash \neg(\mathsf{G}_1(\bar{t},\bar{v}) \vee \ldots \vee \mathsf{G}_i(\bar{t},\bar{v})) \qquad \text{(mixedEB/csrt2)}$$

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 [3] for non-deterministic events and the dedicated POs for probabilistic events that are introduced in Section 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 [3] and in Section 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 :

$$\mathsf{I}(\bar{v}) \vdash (\mathsf{G}_1(\bar{t},\bar{v}) \vee \ldots \vee \mathsf{G}_m(\bar{t},\bar{v})) \vee (\mathsf{G}_{m+1}(\bar{t},\bar{v}) \wedge \mathbf{W}_{m+1}(\bar{v}) > \mathbf{0}) \vee \ldots \vee ((\mathsf{G}_n(\bar{t},\bar{v}) \ \wedge \mathbf{W}_n(\bar{v}) > \mathbf{0})) \qquad \text{(model/mDLF)}$$

5.4 Operational Semantics

As presented respectively in Section 2.2 and Section 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 Section 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 $\mathsf{M}=(\bar{v}, \mathsf{I}(\bar{v}), \mathsf{MEvts}, \mathsf{Init})$, is expressed by means of an MDP $[\![M]\!]=(S, Acts, P, l_{init}, AP, L)$ 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 emphasize the fact that this

new action name only appears in the MDP semantics but does not correspond to a merging of the probabilistic events.

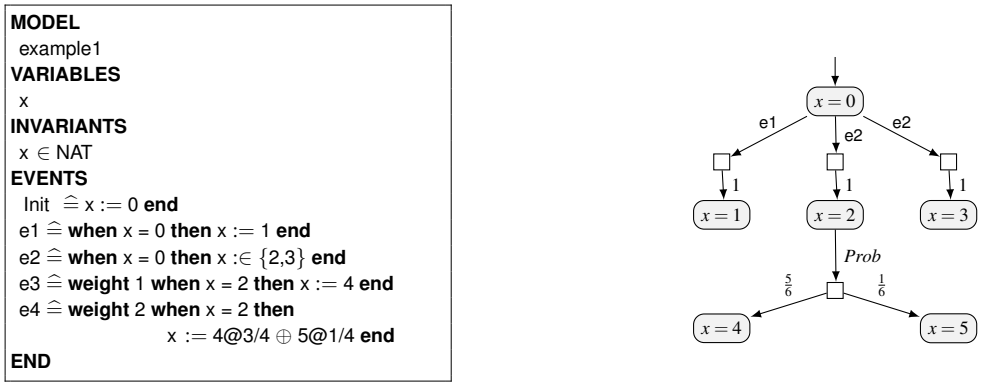

Figure 10: Partially mixed Event-B model and MDP semantics for Example 1

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.

- 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), that 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 $\frac{1}{4}$, that 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 $\underbrace{(\frac{1}{3} \times 1)}_{e3} + \underbrace{(\frac{2}{3} \times \frac{3}{4})}_{e4} = \frac{5}{6}$,
- 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 [22,20,37].

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

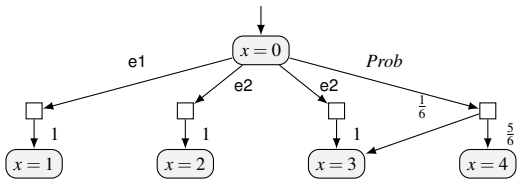

Figure 11: Fully mixed Event-B model and MDP semantics for Example 2

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 prob-

abilistic 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-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 labeled with e2, and with probability $\frac{1}{6}$ using the transition labeled 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=($\bar{v}$, I($\bar{v}$), MEvts, Init) is an MDP $[\![M]\!] = (S, Acts, T, l_{init}, AP, L)$ 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,* $\mathsf{Acts} = \mathsf{Acts}_{nd} \cup \{Prob\}$. *Finally,* $T \subseteq S \times Acts \times Dist(S)$ *is the transition relation such that* $(s, e, \delta) \in T$ *iff either*

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

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

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

□

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.

$\mathsf{I}(\bar{v}) \vdash \text{finite}(\{\bar{t} \mid \mathsf{G}(\bar{t},\bar{v})\})$	(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 $\mathsf{x} :\in \{\mathsf{E}_1(\bar{t},\bar{v}),\ldots,\mathsf{E}_i(\bar{t},\bar{v}),\ldots,\mathsf{E}_n(\bar{t},\bar{v})\}$ which appears in the concerned events of the standard Event-B model, the set of expressions assigned to the corresponding variable must be finite.

$\vdash \text{finite}(\{\mathsf{E}_1(\bar{t},\bar{v}),\ldots,\mathsf{E}_i(\bar{t},\bar{v}),\ldots,\mathsf{E}_n(\bar{t},\bar{v})\})$	(event/assign/proba)

   (b) **Predicate Assignment probabilisation.** For each predicate non-deterministic assignment $\mathsf{x} :\mid \mathsf{Q}_x(\bar{t},\bar{v},\mathsf{x}')$ which appears in the concerned events of the standard Event-B model, the set of values x' such that $\mathsf{Q}_x(\bar{t},\bar{v},\mathsf{x}')$ is true must be finite.

$\vdash \text{finite}(\{\mathsf{x}' \mid \mathsf{Q}_x(\bar{t},\bar{v},\mathsf{x}')\})$	(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 $\mathsf{W}(\bar{v})$;

   (c) Replacing each enumerated assignment by an enumerated probabilistic assignment in the form:

$$\mathsf{x} := \mathsf{E}_1(\bar{t},\bar{v})@\mathsf{p}_1 \oplus \ldots \oplus \mathsf{E}_i(\bar{t},\bar{v})@\mathsf{p}_i \oplus \ldots \oplus \mathsf{E}_n(\bar{t},\bar{v})@\mathsf{p}_n$$

   where the designer will have to precise the desired probability values $\mathsf{p}_1,\ldots,\mathsf{p}_i,\ldots,\mathsf{p}_n$.

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

$$\mathsf{x} :\oplus \mathsf{Q}_x(\bar{t},\bar{v},\mathsf{x}')$$

   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 $\mathsf{P2P}_3$ 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 decrease with the number of successfull downloads. When probabilising the event FailureDL, we need to input probability values for the transformation of the non-deterministic choice $\mathsf{DBin}(c \mapsto b) :\in \{\mathsf{empty}, \mathsf{incoming}\}$ to a probabilistic choice $\mathsf{DBin}(c \mapsto b) := \mathsf{empty}\ @4/10 \oplus \mathsf{incoming}\ @6/10$.

Finally, we obtain a fully mixed event-B model $\mathsf{P2P}_M$ 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.

MODEL
$\mathsf{P2P}_M$
PROBABILISES
$\mathsf{P2P}_3$

VARIABLES
DB
DBin
INVARIANTS
$\mathsf{DB} \in 1..N \times 1..K \rightarrow \{\mathsf{empty}, \mathsf{finished}\} \wedge$
$\mathsf{DBin} \in 1..N \times 1..K \rightarrow \{\mathsf{empty}, \mathsf{incoming}, \mathsf{finished}\} \wedge$
$\forall c \cdot (c \in 1..N \Rightarrow \mathsf{card}(\{b \mid b \in 1..K \wedge \mathsf{DBin}(c \mapsto b) = \mathsf{incoming}\}) \leq 1)$
VARIANT
$2 \times N * K - 2 \times \mathsf{card}(\{c \mapsto b \mid n \in 1..N \wedge b \in 1..K \wedge \mathsf{DBin}(c \mapsto b) = \mathsf{finished}\}) - \mathsf{card}(\{c \mapsto b \mid n \in 1..N \wedge b \in 1..K \wedge \mathsf{DBin}(c \mapsto b) = \mathsf{incoming}\})$
EVENTS
Init $\widehat{=}$
begin
$\mathsf{DB} := (1..N \times 1..K) \times \{\mathsf{empty}\} \parallel$
$\mathsf{DBin} := (1..N \times 1..K) \times \{\mathsf{empty}\}$
end

DLFinished $\widehat{=}$
when
$\mathsf{DB} = (1..N \times 1..K) \times \{\mathsf{empty}\} \wedge$
$\mathsf{DBin} = (1..N \times 1..K) \times \{\mathsf{finished}\}$
then
$\mathsf{DB} := \mathsf{DBin}$
end

Start1DL $\widehat{=}$
any c, b **where**
$c \in 1..N \wedge b \in 1..K \wedge$
$\mathsf{DBin}(c \mapsto b) = \mathsf{empty} \wedge$
$\mathsf{card}(\{k \mid k \in 1..K \wedge \mathsf{DBin}(c \mapsto k) = \mathsf{incoming}\}) = 0$
then
$\mathsf{DBin}(c \mapsto b) := \mathsf{incoming}$
end

Finish1DL $\widehat{=}$
weight
$\mathsf{card}(\{c \mapsto b \mid n \in 1..N \wedge b \in 1..K \wedge \mathsf{DBin}(c \mapsto b) = \mathsf{finished}\}) + 1$
any c, b **where**
$c \in 1..N \wedge b \in 1..K \wedge$
$\mathsf{DBin}(c \mapsto b) = \mathsf{incoming}$
then
$\mathsf{DBin}(c \mapsto b) := \mathsf{finished}$
end

FailureDL $\widehat{=}$
weight
$N * K - \mathsf{card}(\{c \mapsto b \mid n \in 1..N \wedge b \in 1..K \wedge \mathsf{DBin}(c \mapsto b) = \mathsf{finished}\})$
any c, b **where**
$c \in 1..N \wedge b \in 1..K \wedge$
$\mathsf{DBin}(c \mapsto b) = \mathsf{incoming}$
then
$\mathsf{DBin}(c \mapsto b) := \mathsf{empty}\ @4/10 \oplus \mathsf{incoming}\ @6/10$
end

END

Figure 12: A fully mixed Event-B version of the simple P2P protocol

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.

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 [20] and [23], 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 [20] 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 $\mathsf{V}(\bar{v})$ 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 $\mathsf{U}(\bar{v})$ which must maximise the variant $\mathsf{V}(\bar{v})$ and never increase. The proposition from [20] is refined in [23], 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 [20] and [23] 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 [20]: 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.

$$\mathsf{I}(\bar{v}) \wedge ((\mathsf{G}_i(\bar{t},\bar{v}) \wedge \mathsf{W}_i(\bar{v}) > 0) \vee \ldots \vee (\mathsf{G}_n(\bar{t},\bar{v}) \wedge \mathsf{W}_n(\bar{v}) > 0)) \vdash \qquad \text{(model/pVar)}$$

$$(\exists\, \bar{v}'.\ \mathsf{G}_i(\bar{t},\bar{v}) \wedge \mathsf{W}_i(\bar{v}){>}0 \wedge \mathsf{SP}_i(\bar{t},\bar{v},\bar{v}') \wedge \mathsf{V}(\bar{v}'){<}\mathsf{V}(\bar{v})) \vee \ldots \vee (\exists\, \bar{v}'.\ \mathsf{G}_n(\bar{t},\bar{v}) \wedge \mathsf{W}_n(\bar{v}){>}0 \wedge \mathsf{SP}_n(\bar{t},\bar{v},\bar{v}') \wedge \mathsf{V}(\bar{v}'){<}\mathsf{V}(\bar{v}))$$

As in [20], 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 [23].

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

$$\mathsf{I}(\bar{v}) \wedge \mathsf{G}_i(\bar{t},\bar{v}) \wedge \mathbf{W}_i(\bar{v}){>}\mathbf{0} \vdash \mathsf{V}(\bar{v}){\in}\mathsf{NAT} \qquad \text{(event/var/pNAT)}$$

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

$$\mathsf{I}(\bar{v}) \wedge \mathsf{G}_i(\bar{t},\bar{v}) \wedge \mathbf{W}_i(\bar{v}){>}\mathbf{0} \vdash \mathsf{V}(\bar{v}){\leq}\,\mathsf{U} \qquad \text{(event/pBOUND)}$$

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-emptyness 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.

MODEL
M1

VARIABLES
x
y

INVARIANTS
x∈ INT ∧
y∈ INT

VARIANT
x

EVENTS

Init ≙
begin
x:=1 ‖ y:=2
end

evt1 ≙
convergent
weight 1
when 0<x≤2 **then**
x:=x−1 ‖ y:=2∗y
end

evt2 ≙
convergent
weight y−1
when 0 <x ≤ 1
then
x:=x+1
end

evt3 ≙
weight 1
when x=0
then
x:=−1 ‖ y:=−1
end

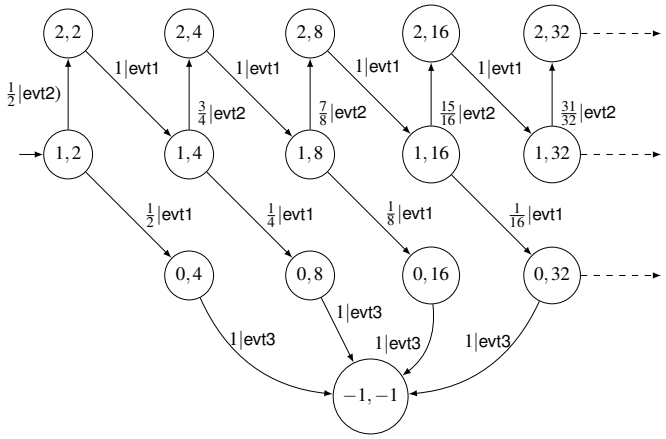

Figure 13: Probabilistic Event-B model and DTMC semantics illustrating the necessity of bounding event weights to ensure almost-certain convergence

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.

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

MODEL
M2

VARIABLES
x
y

INVARIANTS
x∈ INT ∧
y∈ INT

VARIANT
x

EVENTS

Init ≙
begin
x:=1
‖ y:=1
end

evt1 ≙
convergent
weight 1
any t **where**
t ∈ {1.. 2^y} ∧ 0<x≤1
then
x:⊕((t=1 ∧ x'−x+t=0) or
(2≤t≤2^y ∧ x'−x−1=0))
‖ y:=y+1
end

evt2 ≙
convergent
weight 1
when x=2
then
x:=x−1
end

evt3 ≙
weight 1
when x=0
then
x:=−1 ‖ y:=−1
end

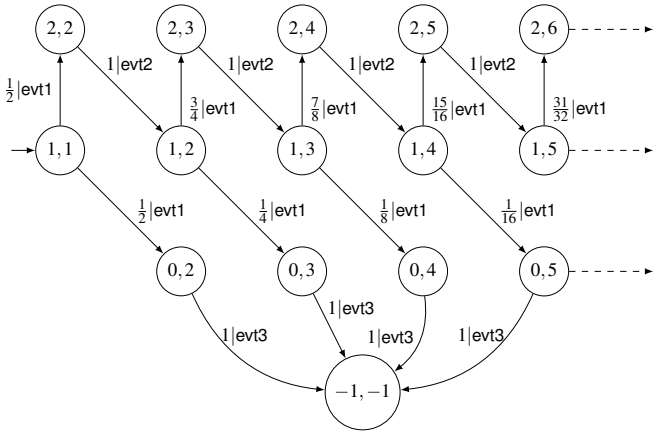

Figure 14: Probabilistic Event-B model and DTMC semantics illustrating the necessity of bounding event parameter values to ensure almost-certain convergence

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.

$$\mathsf{I}(\bar{v}) \wedge \mathsf{G}_i(\bar{t},\bar{v}) \vdash \mathsf{W}_i(\bar{v}) \leq \mathsf{BW} \qquad \text{(event/wght/BOUND)}$$

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

$$\mathsf{I}(\bar{v}) \vdash \mathsf{card}(\{\bar{t} \mid \mathsf{G}_i(\bar{t},\bar{v})\}) \leq \mathsf{BP} \qquad \text{(event/param/BOUND)}$$

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* $\mathsf{PEvts}_c \subseteq \mathsf{PEvts}$ *a set of convergent events. If M satisfies the above POs (1-5), then the set* PEvts_c *almost-certainly converges.*

Proof Let $M = (\bar{v}, I(\bar{v}), V(\bar{v}), \mathsf{Evts}, \mathsf{Init})$ be a probabilistic Event-B model. $\mathsf{Evts} = \mathsf{Evts}_{nc} \cup \mathsf{Evts}_c$ 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

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

2. event/pBOUND

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

3. event/wght/BOUND

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

4. event/param/BOUND

$$\forall e \in \mathsf{Evts}_c.I(\bar{v}) \vdash card(\{\bar{t}|G_e(\bar{t},\bar{v})\}) \leq BP$$

5. model/pVar

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

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 $[\![M]\!] = (S, s_0, AP, L, \mathsf{Acts}, P)$ be the DTMC semantics of M as introduced in Definition 2. We build the DTMC $[\![M]\!]' = (T, t_0, AP, L', \mathsf{Acts}, P')$ where

- $T \subseteq S \times (\mathsf{Acts} \cup \{\varepsilon\})$ is the set of extended states, consisting in pairs (s,a) where s is a state of $[\![M]\!]$ and a is an action (event name),
- $t_0 = (s_0, \varepsilon)$ is the initial state,
- L' is such that $L'((s,a)) = L(s)$ for all $s \in S$ and $a \in \mathsf{Acts}$, 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 $[\![M]\!]'$ from all (extended) states where convergent events are enabled.

Since $[\![M]\!]$ 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 [28], which is a sufficient condition for the "decisiveness" of infinite-state Markov Chains. Formally, given a Markov Chain $\mathcal{M} = (\mathcal{S}, \mathcal{P})$ and a target set of states $\mathcal{F} \subseteq \mathcal{S}$, 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 α. It is then shown in [28] 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 $\widetilde{\mathcal{F}}$ 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 $[\![M]\!]'$ 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 $[\![M]\!]'$ 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 $\widetilde{F}$ is empty and conclude.

We now detail each step of this proof.

(a) Consider the following notations:

- In the DTMC $[\![M]\!]'$, we partition the set of actions (event names) as follows: $\mathsf{Acts} = \mathsf{Acts}_{nc} \cup \mathsf{Acts}_{c}$, where Acts_{nc} is the set of non convergent actions and 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 $[\![M]\!]'$, we write $P(t)$ for the evaluation of P in the state s.
- Given an extended state $t = (s,a) \in T$, we write $\mathsf{Acts}(t)$ for the set of events enabled in s. Similarly, we write $\mathsf{Acts}_c(t)$ for the set of convergent events enabled in s and $\mathsf{Acts}_{nc}(t)$ 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:

$$Succ(t) = \{t' \in T | \exists e \in \mathsf{Acts}(t).P'(t,e,t') > 0\}$$

– Given a finite execution $\sigma = t_0, e_0, t_1, \ldots, t_{n-1}, e_{n-1}, t_n$ of $[\![M]\!]'$, the length of σ is written $L(\sigma)$ and is equal to the number of transitions executed in σ. In the above example case, $L(\sigma) = n$.

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

– $T_1 = \{t = (s,a) \in T \mid \exists e \in \mathsf{Evts}_c, \exists \theta \in T_s^e, G_e(s,\theta) \wedge \forall e' \in \mathsf{Evts}_{nc}, \forall \theta \in T_s^{e'}, \neg G_{e'}(s,\theta)\}$ is the set of extended states where only convergent events are enabled.

– $T_2 = \{t = (s,a) \in T \mid \exists e \in \mathsf{Evts}_c, \exists \theta \in T_s^e, G_e(s,\theta) \wedge \exists e' \in \mathsf{Evts}_{nc}, \exists \theta \in T_s^{e'}, G_{e'}(s,\theta)\}$ is the set of states where both convergent and non convergent events are enabled.

– $T_3 = \{t = (s,a) \in T \mid \forall e \in \mathsf{Evts}_c, \forall \theta \in T_s^e, \neg G_e(s,\theta)\}$ is the set of states where no convergent events are enabled.

– $T_4 = \{t = (s,a) \in T \mid a \in \mathsf{Evts}_{nc}\}$ 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 [28], we write $\widetilde{F}$ for the subset of states of T from which it is impossible to reach F. We show later that $\widetilde{F}$ 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 α.

– 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,

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

By definition of T_2, at least one convergent event is enabled in t_2, therefore we have $W^{t_2}(\mathsf{Acts}_c) > 0$. Likewise, at least one non convergent event can be enabled in t_2, thus $W^{t_2}(\mathsf{Acts}_{nc}) > 0$. Therefore $W^{t_2}(\mathsf{Acts}) > 0$. Recall from section 4.3 that the probability of a transition (t_2,e,t') where $e \in \mathsf{Acts}_{nc}(t_2)$ and $t' = (s',e) \in F_2$ is given by:

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

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:

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

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

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 α_2 be such that $(\frac{1}{\alpha_2} - 1) > BW \times card(\mathsf{Acts}_c)$. 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 $P(t_2 \models \Diamond F_2) \geq \frac{W^{s_2}(\mathsf{Acts}_{nc})}{W^{s_2}(\mathsf{Acts})}$. As a consequence, we must have:

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

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

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

As a consequence,

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

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

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

Finally, by definition of α_2, we have $W^{s_2}(\mathsf{Acts}_c) > BW \times card(\mathsf{Acts}_c)$, 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 α_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 $[\![M]\!]'$ is expressed as follows: For all $t_1 = (s_1, a) \in T_1$, $e \in \mathsf{Acts}$, and $t' = (s', a) \in T$, we have

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

Since $t_1 \in T_1$, this expression can only be non-zero if e is a convergent event. In this case, PO event/wght/BOUND ensures that $W^{s_1}(\mathsf{Acts}) \leq BW \cdot card(\mathsf{Acts}_c)$. Therefore, for all convergent events enabled in t_1, we have $\frac{W_e(s_1)}{W^{s_1}(\mathsf{Acts})} \geq \frac{1}{BW \cdot card(\mathsf{Acts}_c)}$.

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,

$$\sum_{\theta \in T^e_{s_1}} \left[P_{T^e_{s_1}}(\theta) \times \prod_{x \in Var(e)} P^e_{s_1,\theta}(x,s') \right] \geq \frac{1}{BP} \times \sum_{\theta \in T^e_{s_1}} \left[\prod_{x \in Var(e)} P^e_{s_1,\theta}(x,s') \right]$$

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

$$\sum_{\theta \in T^e_{s_1}} \left[\prod_{x \in Var(e)} P^e_{s_1,\theta}(x,s') \right] \geq \beta$$

As a consequence, there exists a minimal value $\gamma > 0$ such that $P'(t_1,e,t') \geq \gamma$ for all $t_1 \in T_1, e \in Acts_c$, 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 $\sigma = t_0,e_0,t_1,\ldots,t_{n-1},t_{n-1},t_n$ 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 $[\![M]\!]'$ is globally coarse w.r.t F. As a consequence, $\forall t \in T, P(t \models \Diamond F \vee \Diamond \widetilde{F}) = 1$.

(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 $T = T_1 \cup T_2 \cup T_3$, $\widetilde{F}$ is therefore necessarily empty.

Since $[\![M]\!]'$ is globally coarse w.r.t F and $\widetilde{F}$ 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 $[\![M]\!]'$ from all (extended) states where convergent events are enabled, which concludes our proof. □

Remark. The additionnal 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 Sec. 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 $\mathsf{P2P}_3$. Indeed, the event FailureDL fails to decrease the corresponding variant, therefore preventing convergence.

Now consider the fully probabilistic Event-B model $\mathsf{P2P}_P$ given in Fig. 7 from Sec. 4. To show that the protocol always terminates we have to show that the set of events {Start1DL, Finish1DL, FailureDL} almost-certainly converges (i.e. converges with probability 1).

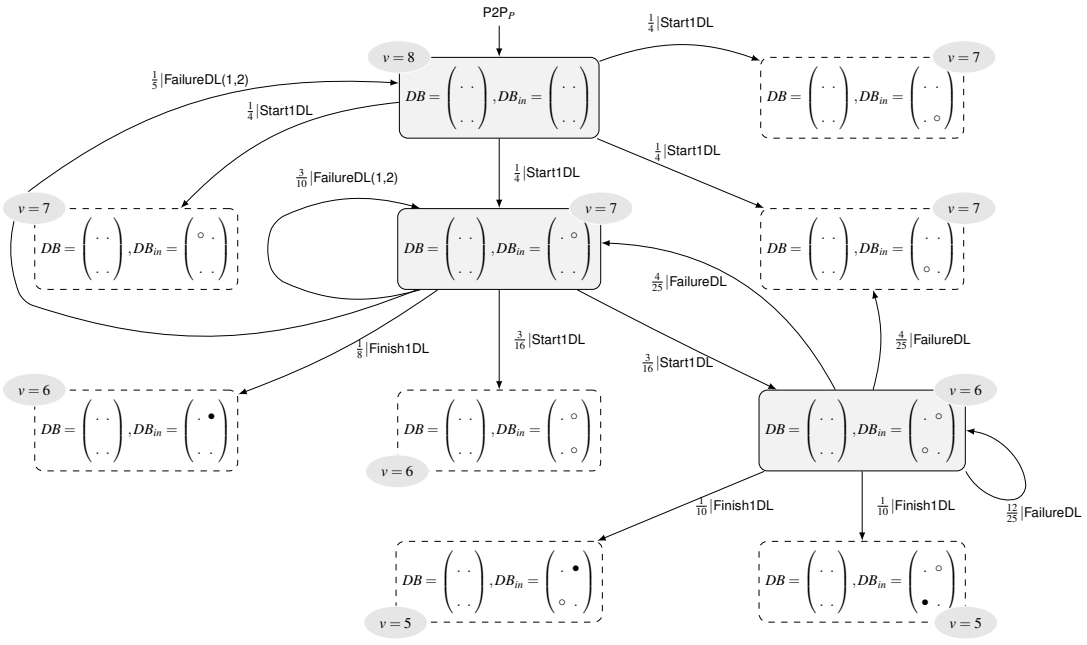

Figure 15: Extract of the DTMC of the simple P2P protocol, with N=2 and K=2

- 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 $\mathsf{P2P}_P$ satisfies all the required POs, Theorem 1 ensures that the set $\{\mathsf{Start1DL}, \mathsf{Finish1DL}, \mathsf{FailureDL}\}$ 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 $\mathsf{P2P}_P$ (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 models. 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 Section 7.1. Although the resulting semantics is an MDP instead of a Markov Chain, the almost-certain convergence results still hold.

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

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 U.

$\mathsf{I}(\bar{v}) \wedge \mathsf{G}_i(\bar{t},\bar{v}) \vdash \mathsf{V}(\bar{v}) \leq \mathsf{U}$	(event/ndpBOUND)

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

probabilistic non-convergent events MEvts_{pnc}*, standard convergent events* MEvts_{ndc}*, and standard non-convergent events* MEvts_{ndnc}*.*

If all events from $\mathsf{MEvts}_{pc} \cup \mathsf{MEvts}_{ndc}$ *satisfy the PO* (event/var/NAT) *presented in Section 2.3, all events from* MEvts_{ndc} *satisfy the POs* (event/var) *(from Section 2.3) and* (event/ndpbound) *presented above, and all events from* MEvts_{pc} *satisfy the POs* (event/wght/BOUND), (event/pBOUND) *and* (event/var/pNAT) *presented in Section 7.1, then the set of events* $\mathsf{MEvts}_{pc} \cup \mathsf{MEvts}_{ndc}$ *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 probability 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 pre-determined 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 [4] 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.

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	✓

Table 1: Plugin features

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.

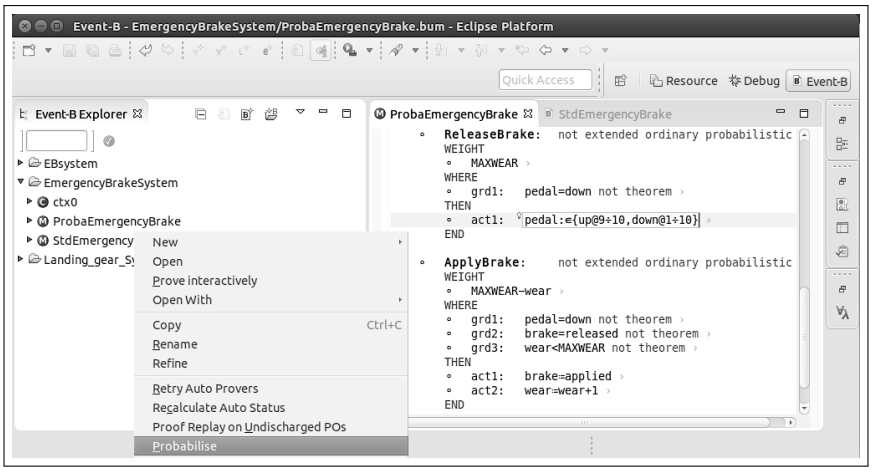

Figure 16: Probabilistic plugin to the Rodin platform

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 Tab. 1 where ✓ denotes the supported functionalities and $\sim$ the functionalities that are currently under development. As explained in Section 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 $\mathsf{W}(\bar{v}) > 0$ to the event guard $\mathsf{G}(\bar{t},\bar{v})$ of each event. We therefore obtain a new guard $\mathsf{G'}(\bar{t},\bar{v}) \;\widehat{=}\; \mathsf{G}(\bar{t},\bar{v}) \wedge \mathsf{W}(\bar{v}) > 0$, 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

1. Bittorent description. `http://www.bittorrent.com/lang/fr/`.
2. Prism modelchecker description. `http://www.prismmodelchecker.org/`.
3. J.-R. Abrial. *Modeling in Event-B: system and software engineering*. Cambridge University Press, 2010.
4. J.-R. Abrial, M. Butler, S. Hallerstede, T. S. Hoang, F. Mehta, and L. 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.
5. J.-R. Abrial, D. Cansell, and D. Méry. A mechanically proved and incremental development of ieee 1394 tree identify protocol. *Formal aspects of computing*, 14(3):215–227, 2003.

6. M. A. Aouadhi, B. Delahaye, and A. Lanoix. Moving from event-b to probabilistic event-b. In *Proceedings of the 32th Annual ACM Symposium on Applied Computing (to appear)*. ACM, 2017.
7. P. Audebaud and C. Paulin-Mohring. Proofs of randomized algorithms in coq. *Science of Computer Programming*, 74(8):568–589, 2009.
8. R.-J. Back. Refinement calculus, part ii: Parallel and reactive programs. In *Stepwise Refinement of Distributed Systems Models, Formalisms, Correctness*, pages 67–93. Springer, 1990.
9. R. J. Back and J. von Wright. Refinement calculus, part i: Sequential nondeterministic programs. In *Stepwise refinement of distributed systems models, formalisms, correctness*, pages 42–66. Springer, 1990.
10. C. Baier, J.-P. Katoen, et al. *Principles of model checking*, volume 26202649. MIT press Cambridge, 2008.
11. G. Barthe, C. Fournet, B. Grégoire, P.-Y. Strub, N. Swamy, and S. Zanella-Béguelin. Probabilistic relational verification for cryptographic implementations. In *ACM SIGPLAN Notices*, volume 49, pages 193–205. ACM, 2014.
12. D. Bert and F. Cave. Construction of finite labelled transition systems from b abstract systems. In *Integrated Formal Methods*, volume 1945 of *LNCS*, pages 235–254. Springer, 2000.
13. A. Bianco and L. De Alfaro. Model checking of probabilistic and nondeterministic systems. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 499–513. Springer, 1995.
14. M. Butler and I. Maamria. Practical theory extension in event-b. In *Theories of Programming and Formal Methods*, volume 8051 of *LNCS*, pages 67–81. 2013.
15. W. W. Chu and C.-M. Sit. Estimating task response time with contentions for real-time distributed systems. In *Real-Time Systems Symposium, 1988., Proceedings.*, pages 272–281. IEEE, 1988.
16. W.-P. De Roever, K. Engelhardt, and K.-H. Buth. *Data refinement: model-oriented proof methods and their comparison*, volume 47. Cambridge University Press, 1998.
17. C. Dehnert, D. Gebler, M. Volpato, and D. N. Jansen. *On Abstraction of Probabilistic Systems*, pages 87–116. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.
18. O. Goldreich. Probabilistic proof systems. In *Modern Cryptography, Probabilistic Proofs and Pseudorandomness*, pages 39–72. Springer, 1999.
19. H. Haghighi and M. Afshar. A z-based formalism to specify markov chains. *Computer Science and Engineering*, 2(3):24–31, 2012.
20. S. Hallerstede and T. S. Hoang. Qualitative probabilistic modelling in event-b. In *Integrated Formal Methods*, pages 293–312. Springer, 2007.
21. J. He, C. Hoare, and J. W. Sanders. Data refinement refined resume. In *ESOP 86*, pages 187–196. Springer, 1986.
22. T. S. Hoang. *The development of a probabilistic B-method and a supporting toolkit*. PhD thesis, The University of New South Wales, 2005.

23. T. S. Hoang. Reasoning about almost-certain convergence properties using event-b. *Science of Computer Programming*, 81:108–121, 2014.

24. J. Hurd. *Formal verification of probabilistic algorithms*. PhD thesis, University of Cambridge, Computer Laboratory, 2003.

25. J. Hurd, A. McIver, and C. Morgan. Probabilistic guarded commands mechanized in hol. *Electronic Notes in Theoretical Computer Science*, 112:95–111, 2005.

26. B. Jonsson and K. G. Larsen. Specification and refinement of probabilistic processes. In *Logic in Computer Science, 1991. LICS'91., Proceedings of Sixth Annual IEEE Symposium on*, pages 266–277. IEEE, 1991.

27. J.-P. Katoen. *Abstraction of Probabilistic Systems*, pages 1–3. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

28. R. Mayr, N. B. Henda, and P. A. Abdulla. Decisive markov chains. *Logical Methods in Computer Science*, 3, 2007.

29. A. McIver and C. C. Morgan. *Abstraction, refinement and proof for probabilistic systems*. Springer Science & Business Media, 2006.

30. C. Morgan, T. S. Hoang, and J.-R. 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.

31. R. Motwani and P. Raghavan. *Randomized algorithms*. Chapman & Hall/CRC, 2010.

32. M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

33. K. Sere and E. Troubitsyna. Probabilities in action systems. In *Proc. of the 8th Nordic Workshop on Programming Theory*, pages 373–387, 1996.

34. M. Stoelinga. An introduction to probabilistic automata. *Bulletin of the EATCS*, 78(176-198):2, 2002.

35. A. Tarasyuk, E. Troubitsyna, and L. Laibinis. Reliability assessment in event-b development. *NODES 09*, page 11, 2009.

36. A. Tarasyuk, E. Troubitsyna, and L. Laibinis. Towards probabilistic modelling in event-b. In *Integrated Formal Methods*, pages 275–289. Springer, 2010.

37. A. Tarasyuk, E. Troubitsyna, and L. Laibinis. Integrating stochastic reasoning into event-b development. *Formal Aspects of Computing*, 27(1):53–77, 2015.

38. K. S. Trivedi, S. Ramani, and R. Fricks. Recent advances in modeling response-time distributions in real-time systems. *Proceedings of the IEEE*, 91(7):1023–1037, 2003.

39. A. Villemeur. *Reliability, Availability, Maintainability and Safety Assessment, Assessment, Hardware, Software and Human Factors*, volume 2. Wiley, 1992.

40. E. Yilmaz. *Tool support for qualitative reasoning in Event-B*. PhD thesis, Master Thesis ETH Zürich, 2010, 2010.