About Probabilistic Event-B

Mohamed Amine Aouadhi, Benoit Delahaye, Arnaud Lanoix

▶ **To cite this version:**

Mohamed Amine Aouadhi, Benoit Delahaye, Arnaud Lanoix. About Probabilistic Event-B. [Research Report] LINA-University of Nantes. 2015. hal-01151594v1

HAL Id: hal-01151594
https://hal.science/hal-01151594v1

Submitted on 13 May 2015 (v1), last revised 15 May 2015 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

About Probabilistic Event-B

Aouadhi Mohamed Amine, Delahaye Benoît, Lanoix Arnaud

May 13, 2015

1 Introduction

We present in this document our investigation in the probabilistic version of Event-B. We will present the reasons for expressing a such formalism and the modifications of the standard proof obligations or the new proof obligations that can be generated for this Event-B version. After that, we will study the case of refinement of a probabilistic Event-B model by another probabilistic Event-B model and express the related new proof obligations.

2 Related Work

Many formal methods were developped for the design of complex software-intensive systems including the Event-B method that was proposed as an extension of the B method to model parallel, distributed and reactive systems. However, the latter systems exhibit sometimes a probabilistic behaviour and are more appropriately modelled probabilistically. Then, many works have discussed the extension of the semantics of Event-B to ensure the expression of probabilistic information in the systems models. Earlier works have also focused in the same extension on the ancestor of the eventB method: the B method. A first step allowing probabilistic programs to be written and reasonned within B was treated by Thai Son Hoang and al is described in [21]. A study about the refinement of probabilistic program in B was conducted by the same author, the work is described in [19]. The overall works about extending B with probabilistic meaning are presented in [20]. All the research works undertaken to extend Event-B of probabilistic semantics follows the earlier work in B, by transporting ideas from B to Event-B, the purpose of these works is to describe probabilistic systems in Event-B. For probabilistic systems, some problems can only be described in terms of numerical measures such reliability problems, performance problems or dependability problems. In the other side, there are many probabilistic models where the exact numerical measures are not of importance when it come to problems like modelling communication protocols or human behaviours, the most important for these problems is to guarantee the termination of these systems. This arises two kind of propositions for the probabilistic Event-B semantics: the qualitative probabilistic operator [15] and the quantitative probabilistic operator [36, 38, 37] , we describe each one below. We state that the two propositions for the probabilistic extension does not depart from the original structure of Event-B machines but they introduce the probabilistic choice as a replacement for non-deterministic choices, specifically for these works in the place of non-deterministic assignements. A non-deterministic choice in Event-B can occurs in three places:

- choice between different events,
- choice of local variables of events,
- nondeterministic assignements.

The propositions of a probabilistic Event-B reclaims also some requirements:

- the probabilistic event-B must be simple, i.e easy to understand,

- it should be useful, i.e solve a commonly encountered class of problems,
- it should permit efficient tool support.

We explicit below the two propositions of a probabilistic event-B:

2.1 Qualitative probabilistic Event-B

Hallerstede and al propose a variant of expressing probabilistic properties in Event-B [15]. In fact, the standard Event-B is based on demonic non-determinism which is not sufficient for modelling some systems that their behaviours are more appropriately modelled probabilistically. In this work, authors restrict their attention on a qualitative aspect of probability, they do not specify the exact numerical measures of probability, but they specify that a non-deterministic choice in the standard Event-B is replaced by a probabilistic choice according to some unknown positive probability. Specifically, they specify that a non-determinism assignement in an action of an event can be replaced by a probabilistic assignement. The purpose of this kind of reasonning is to provide the concept of almost-certain convergence in Event-B like the same reasonning introduced in B. The qualitative probabilistic reasonning is introduced into Event-B by means of the qualitative probabilistic operator choice introduced in B described by:

$$\boxed{\text{S} \oplus \text{T}}$$

S or T are chosen according to some positive probability, specially, in the case of assignement, it's presented as follows:

$$\boxed{\text{x:= a} \oplus \text{ b}}$$

A variable x take as a value a or b according to some positive unknown probability. The authos have made some modifications to the proof obligations related to termination, more precisely for the **VAR** proof obligation mentionning that the invariant is decreased by each new event introduced in refinement. In fact, when introducing a new event, and when the action of this event is non-probabilistic, this action must decrease the variant. Suppose we have two events `ea` and `ec`, where `ea` is the abstract event and `ec` is an event introduced during the refinement of the abstract machine, the invariant is denoted $I(v)$ and the gluing invariant is denoted $J(v, w)$.

ea = **any** t **where** $G(t, v)$ **then** $S(t, v)$
end,
ec = **any** u **where** $H(u, w)$ **then** $T(u, w)$
end

The proof obligation for progress (**VAR**) is as follows:

$$I(v)$$
$$J(v, w)$$
$$H(u, w)$$
$$\vdash$$
$$(\ \forall\ w'.\ T(u, v, w') \implies V(w') < V(w))$$

The authors specifies that a probabilistic action behaves identically to a nondeterministic action, i.e demonically and it behaves angelically when it come respect to the progress constraint. A probabilistic action follows from the angelical interpretation from a non-deterministic action and it may decrease the variant, so the proof obligation is modified as bellow:

$$I(v)$$
$$J(v, w)$$
$$H(u, w)$$
$$\vdash$$
$$(\ \exists\ w'.\ T(u, v, w') \implies V(w') < V(w))$$

The work described in [15] has been improved by E Yilmaz in [41]. In fact, the preservation of qualitative reasonning during refinement was not treated in this work, so that, E. Yilmaz and al describes how to maintain qualitative probabilistic reasonning during refinement. In addition, they propose a tool support for this kind of reasonning in Event-B: they extend the Rodin platform to support proving almost-certain ter- mination in probabilistic Event-B by the development of a dedicated plugin.

2.2 Quantitative probabilistic Event-B

Event-B offers a scalable approach ensuring that a detailed specification of a system adheres to its abstract counterpart, i.e, it guarantees that a system is functionally correct. However, it's not sufficient to guarantee that a system is functionally correct but also it must satisfies a number of non-functional critical properties. Usually such properties are defined probabilistically, we mention for example properties like reliability (the probability that a system can perform a required function under given conditions over a given period of time) and responsiveness (the probability of the system to complete execution of a requested service within a given time bound). These properties are defined like quantitative stochastic measures, to enablle this kind of reasonning in Event-B, Many works [36, 38, 37] have extended Event-B by adding a quantitative probabilistic choice operator denoted $\oplus$ |, which allows us to represent a precise probabilistic information about how a choice can be made, i.e, this choice is made with a known probability p. All the works specify that this operator is only used in the assignement, we present below it's syntax:

$$\text{x} \oplus| \ x_1@p_1; ...x_n@p_n, \text{ where } \forall \text{i} \in 1..\text{n} \ . \ 0 < p_i \leq 1 \text{ and } \sum_{i=0}^{i=n} p_i = 1$$

It assigns to x a new value x_i with a probability p_i, like the qualitative probabilistic operator, this operator is introduced to replace a non-deterministic assignement x:$\in$ { x_i,..,x_n } where x take a value between x_1 and x_n randomly.

3 Event-B Abstract machine

3.1 Event-B Basic Notations

Event-B [2] is a formal method used for the development of complex systems and presented like an evolution of the standard classical B [3], the both was developed by Jean Raymond Abrial[1]. Systems are described in Event-B by means of models. Formally, an Event-B model is expressed by M=(X, $Init$, Inv, $Evts$) where:

- X: the set of variables of our model,
- $Init$: the initialization event,
- Inv: the invariant of our model,
- $Evts$: the set of events of our model.

An event in an Event-B model can be expressed in one of the forms presented below:

1. ea = **any** t **where** $G(t, v)$ **then** $S(t, v)$ **end** : t represents a parameter of the event, $G(t, v)$ the guard and $S(t, v)$ the action. This is the general form of an event in Event-B, it is enabled only if the guard $G(t, v)$ is fufilled.

2. ea = **when** $G(v)$ **then** $S(v)$ **end** : when an event does not have a parameter t, then the guard $G(v)$ and the action $S(v)$ are independent from the parameter.

3. ea = **begin** $S(v)$ **end** : when the event does not contain any parameter or guard. This is the simplest form of an event, the initialization event is expressed in this form.

The action of an event may contain different assignments that are executed in parallel. An assignement can be expressed in one of the following form:

[1] Jean Raymond Abrial is an independent consultant and a research visitor at many academic and research units, he is the inventor of the Z method, the B method and the Event-B method.

1. $x := E(t, v)$: a deterministic assignment, x represents some variables of our system, $E(t, v)$ some expressions.
2. $x :\in E(t, v)$: a non-deterministic assignment, x represents a variable of our system, $E(t, v)$ some expressions, x take a value non-deterministically from the set of expressions $E(t, v)$.
3. $x :| Q(t, v, x')$: a non-deterministic assignment, x represents a variable or our system, $Q(t, v, x')$ represent a predicate. In this case, x take a new value x' such that the predicate $Q(t, v, x')$ must be satisfied.

When the action of an event contain some assignments of the form 2 or 3 presented above, the action of the corresponding event is denoted by $S_{nd}(t, v)$. If all the assignments of an action are in the form 1 presented above, then the action is denoted by $S_d(t, v)$.
The effect of each assignment can be described by means of a before-after predicate (BA) that describe the relationship between the values of the variables of the system just before an assignment has occured (represented by the variable name x) and the values of the variables of our system just after the occurence of the assignment (represented by the variable name x'). A before-after predicate can be represented by:

- If the assignment is in the form 1 presented previously, then $BA(x := E(t, v)) \cong x' = E(t, v)$.
- If the assignment is in the form 2 presented previously, then $BA(x :\in E(t, v)) \cong x' \in E(t, v)$.
- If the assignment is in the form 3 presented previously, then $BA(x :| Q(t, v, x')) = Q(t, v, x')$.

3.2 Proof Obligations specific to Event-B abstract machines

In Event-B modelling, proof obligations, [2, Ch5] serve to verify certain properties of a model. Let evt be an abstract event with an invariant $inv(c, v)$ (c represent the constants of our model and v the variables). The guard of this event is $G(t, v)$ while the action is expressed by means of the before-after predicate ($v :| BA(c, v, t, v')$):

evt
any
t
where
$G(c, v, t)$
then
$v :| BA(c, v, t, v')$
end

3.2.1 Invariant preservation proof obligation rule

This proof ensures that each event in the machine preserves the different invariants. For an event evt, this proof obligation is named: "$evt/inv/INV$"
Consider the abstract event evt, the standard form of the invariant preservation proof obligation rule is presented below:

Axioms and theorems	$A(c)$
Invariants and theorems	$I(c, v)$
Guards of the event	$G(c, v, t)$
Before-after predicate of the event	$BA(c, v, t, v')$
$\vdash$	$\vdash$
Modified specific invariant	$inv(c, v')$

3.2.2 Feasability proof obligation rule: FIS

This proof aims to ensure that a non-deterministic action is feasible. Suppose that the action of the event *evt* presented above is non-deterministic. If the action name is *act* then the name of this proof obligation is "$evt/act/FIS$". This proof obligation is presented below:

Axioms and theorems Invariants and theorems Guards of the event ⊢ ∃ v'. before-after predicate	$A(c)$ $I(c, v)$ $G(c, v, t)$ ⊢ $\exists\ v'.\ BA(c, v, t, v')$

3.2.3 Deadlock freedom rule

This rule ensures that an Event-B model is deadlock free, given a model with constants c, set of axioms $A(c)$, variables v and set of invariants $inv(c, v)$, this proof obligation consists in proving that one of the guards $G_1(c, v), ..., G_n(c, v)$ of the different events is always true. We present below the form of this proof:

Axioms Invariants ⊢ Disjunction of the guards	$A(c)$ $I(c, v)$ ⊢ $G_1(c, v) \vee ... \vee G_n(c, v)$

3.3 Termination in Event-B

Termination is a crucial aspect in Event-B development, it means that a chosen set of new events introduced during refinement are enabled only a finite number of time before an event that is not marked as terminating occurs. In Event-B, the type of an event fixes its type and the proof obligations derived for it. These proof obligations are related to the concept of termination. To support proof for termination, a variant is proposed and all the proof obligations specific to termination act over this variant. There are three types of events in the standard Event-B:

- **Ordinary**: an event is ordinary if it occurs arbitrary often and does not put any restriction upon the variant.
- **Convergent**: Convergent events are considered internal and must not be executed forever, an event is said to be convergent if it must decrease the variant every time it occurs.
- **Anticipated**: Anticipated events are those which will take on their roles in later refinement steps and they might be convergent in these steps. An event is said to be anticipated if it must not decrease the variant.

Each new event introduced during refinement must be either convergent or anticipated. To prove termination in the standard Event-B, we must prove that each convergent event must decrease the variant and this latter has a lower bound. If the model contains only anticipated events, we create simply a default constant variant that must not be increased by each event. Each anticipated event is refined by only an anticipated or a convergent event, the proof of termination is delayed for the stage when all the anticipated events are refined by convergent events.

3.3.1 Proof obligations specific to termination

Let m be an Event-B machine containing two events: evt_1 a convergent event and evt_2 an anticipated event.

machine
m
refines
...
sees
...
variables
v
Invariant
$I(c, v)$
theorems
...
events
evt_1, evt_2
variant
$n(c, v)$
end

evt_1
status
convergent
any
t
where
$G(c, v, t)$
then
$v{:}|\quad BA(c, v, x, v')$
end

evt_2
status
anticipated
any
x
where
$G(c, v, x)$
then
$v{:}|\quad BA(c, v, x, v')$
end

We present below the proof obligations for proving termination:

The numeric variant proof obligation rule: NAT This rule ensures that under the guards of each *convergent* or *anticipated* event, a proposed numeric variant is indeed a natural number. For a *convergent* (or *anticipated*) event *evt*, the name of this rule is "evt/NAT".

The **NAT** proof obligation is as follows:

Axioms and theorems Invariant and theorems Event guards $\vdash$ A numeric variant is a natural number	$A(c)$ $I(c, v)$ $G(c, v, x)$ $\vdash$ $n(c, v) \in \mathbb{N}$

The finite set variant proof obligation rule: FIN This rule ensures that under the guards of each convergent or anticipated event, a proposed set variant must be a finite set. For a convergent (or anticipated) event *evt*, the name of this rule is "evt/FIN". The **FIN** proof obligation is as follows:

Axioms and theorems Invariants and theorems Event guards $\vdash$ Finiteness of set variant	$A(c)$ $I(c, v)$ $G(c, v, t)$ $\vdash$ finite$(t(s, c, v))$

The variant proof obligation rule: VAR This rule ensures that each convergent event must decrease the variant, it ensures also that each anticipated event does not increase the variant. Let *evt* be an event, the name of the rule is "evt/VAR". Given the convergent event evt_1 presented above, if the variant $n(c, v)$ is numeric, then the proof obligation is as follows:

Axioms and theorems Invariants and theorems Guards of the event Before-after predicate of the event $\vdash$ Modified variant smaller than variant	$A(c)$ $I(c, v)$ $G(c, v, x)$ $BA(c, v, x, v')$ $\vdash$ $n(c, v') < n(c, v)$

If the variant $t(c, v)$ is a finite set, then the proof obligation rule is as follows:

Axioms and theorems	$A(c)$
Invariants and theorems	$I(c, v)$
Guards of the event	$G(c, v, x)$
Before-after predicate of the event	$BA(c, v, t, v')$
$\vdash$	$\vdash$
Modified variant strictly included in variant	$t(c, v') \subset t(c, v)$

Given the anticipated event evt_2, if the variant $n(c, v)$ is numeric, then the proof obligation rule is as follows:

Axioms and theorems	$A(c)$
Invariants and theorems	$I(c, v)$
Guards of the event	$G(c, v, x)$
Before-after predicate of the event	$BA(c, v, x, v')$
$\vdash$	$\vdash$
Modified variant not greater than variant	$n(c, v') \leq n(c, v)$

If the variant $t(c, v)$ is a finite set, then the proof obligation is as follows:

Axioms and theorems	$A(c)$
Invariants and theorems	$I(c, v)$
Guards of the event	$G(c, v, x)$
Before-after predicate of the event	$BA(c, v, x, v')$
$\vdash$	$\vdash$
Modified variant in or equal to variant	$t(c, v') \subseteq t(c, v)$

3.4 Refinement in Event-B

Many research works have focused in the development of the theory of refinement inducing a growing body of experience and literature around this thematic [9, 8, 27, 39, 16, 14, 13]. 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 methodolgy when the system is firstly described by an abstract specification and progressively refined by other specifications. Each abstract specification will be more detailed and new specifications can be introduced during refinement. The second concept concerns the preservation of correctness between abstract and refined specifications. The process of modelling systems in Event-B is based in the theory of refinement, Event-B is a state based methodology like Z [40], ASM [34], LOTOS [33] and Action Systems [10] where the system is described by a set of "states plus events", the state space describes the variables of the system where the events describes the dynamic part of a system (the actions performed on those variables), and then, refinement permit either a fine description of the state space and the events or the introduction of extra details to the state space (data refinement [17, 14]) or the introduction of new events. For the refinement of events in Event-B, three properties must be satisfied:

- Consistency: The effect of the concrete event corresponds to an effect that is allowed by the abstract event in a corresponding state.
- Restricted Consistency: In states where the abstract event is enabled in a corresponding state, the effect of the concrete event corresponds to an effect that is allowed by the abstract event in such a state.
- Enabledness: When events can be invoked in the abstract state, they can be invoked in the corresponding concrete state as well.

Then, starting from a system described by an abstract model, Event-B refinement allows the building of complex systems by the introduction of more details in every step, the three properties mentioned above involve that a crucial aspect in the Event-B refinement is the maintain of correctness between the

abstract and the concrete model, i.e, 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.

3.4.1 Proof obligations specific to refinement

We present below the proof obligations specific to refinement:

The guard merging proof obligation rule: MRG This proof obligation ensures that the guard of a concrete event merging two abstract events is stronger than the disjunction of the guards of the abstract events. For a merging event evt, the name of the rule is "evt/MRG". Let evt_{01} and evt_{02} be two abstract events with the same parameters and the same action and let evt be the concrete merging event:

evt_{01}
any
t
where
$G1(c, v, t)$
then
$S1$
end

evt_{02}
any
t
where
$G2(c, v, t)$
then
$S2$
end

evt
refines
evt_{01}
evt_{02}
any
y
where
$H(c, v, y)$
then
T
end

The standard proof obligation is:

Axioms and theorems Abstract invariants and theorems Concrete event guards $\vdash$ Disjunction of abstract guards	$A(c)$ $I(c, v)$ $H(c, v, y)$ $\vdash$ G1(c,v,t) $\vee$ G2(c,v,t)

Guard strengthening proof obligation rule: GRD The purpose of this rule is to ensure that the concrete guards in a concrete event are stronger than the abstract ones in the abstract event. This ensures that when a concrete event is enabled, so its corresponding abstract one is enabled too. Let an abstract event evt_0, the name of this proof obligation is: "$evt_0/grd/GRD$". Let evt the refinement of evt_0, we present below the two events:

evt_0
any
t
where
$g(c, v, t)$
then
...
end

evt
refines
evt_0
any
y
where
$H(y, c, w)$
with
...
$t : W(t, c, w, y)$
then
...
end

We notice the witness predicate $W(t, c, w, y)$ which is due to the fact that the abstract parameter t is no more used in the concrete event and is replaced by the concrete parameter y. The proof obligation rule is the following:

Axioms and theorems Abstract invariants and theorems Concrete invariants and theorems Concrete event guards Witness predicates for variables $\vdash$ Abstract event specific guard	$A(c)$ $I(c, v)$ $J(c, v, w)$ $H(y, c, w)$ $W(t, c, w, y)$ $\vdash$ $g(c, v, t)$

Simulation proof obligation The purpose of this proof is to ensure that each action in an abstract event is correctly simulated in the corresponding refinement. This ensure that when a concrete event is "executed", what it does is not contradictory with what the corresponding abstract event does. For a concrete event evt, the name of this proof obligation is "$evt/act/SIM$". Let evt_0 be an event and evt be its refinement:

evt_0 **any** x **where** $G(c, v, x)$ **then** $v :\mid BA1(c, v, x, v')$ **end**	evt **refines** evt_0 **any** y **where** $H(y, c, w)$ **with** $x : W1(x, c, w, y, w')$ $v' : W2(v', c, w, y, w')$ **then** $w :\mid BA2(c, w, y, w')$ **end**

The proof obligation is as follows:

Axioms and theorems Abstract invariants and theorems Concrete invariants and theorems Concrete event guards witness predicates for parameters witness predicates for variables Concrete before-after predicate $\vdash$ Abstract before-after predicate	$A(c)$ $I(c, v)$ $J(c, v, w)$ $H(y, c, w)$ $W1(x, c, w, y, w')$ $W2(v', c, w, y, w')$ $BA2(c, w, y, w')$ $\vdash$ $BA1(c, v, x, v')$

4 Probabilistic Event-B: Syntax

We propose a new version of the standard Event-B entitled *"Probabilistic Event-B"*. In this version, all the non-deterministic choices in an Event-B model are replaced by probabilistic choices. In the standard Event-B, a non-deterministic choice can appear in:

1. When many events are enabled in the same time, only one of these events is chosen non-deterministically to be executed.
2. When an event contains a parameter, the value taken by this parameter is chosen non-deterministically from the set of values such that for each value in this set, the guard of the event is fulfilled.
3. The last case is when the action of an event contains non-deterministic assignments.

4.1 Replacing non-deterministic choice with probabilities

In the proposed version of Event-B, we resolve the non-deterministic choice of the event to be executed in the case when many events are enabled in the same time by modifying the structure of an event in the standard Event-B such that each event will occur with a precise positive weight. In the case also when the event has a parameter t, we replace the non-deterministic choice of the value taken by the parameter by an uniform choice of this value from the subset of values in such a way that the guard of the parameter will be satisfied. Lastly, when the action of the event contains some non-deterministic assignments, we propose also a new probabilistic assignment to resolve the non-deterministic choice of the assignment to be executed. We present in what follows the different cases of non-deterministic choices and its corresponding probabilistic version.

Case 1: Annotating events with weights.
To resolve the non-deterministic choice between some enabled events, we annotate all the events by a positive weight. In the standard Event-B, the general form of an event is:

$$e_a = \textbf{any } t \textbf{ where } G(t,v) \textbf{ then } S_d(t,v) \textbf{ end}$$

We suppose that the action of the event is deterministic, it is noted by S_d. The corresponding form of this event in our new probabilistic version of Event-B will be:

$$e_a = \textbf{weight } w \textbf{ any } t \textbf{ where } G(t,v) \textbf{ then } S_d(t,v) \textbf{ end}$$

The weight of a probabilistic event must be strictly greater than 0, we have not yet specified the exact word in the syntax of the new proposition of Event-B to express the weight of an event, we mentioned above the word **weight** but it is not yet confirmed. For each probabilistic event e_a, we denote by $W(e_a)$ the weight of occurence of e_a.

Case 2: Some events contain a probabilistic choice of the value of the parameter.
We suppose that the parameter of an event take this value from a finite set T, the form of the event in the standard Event-B is:

$$e_a = \textbf{begin any } t \textbf{ where } t \in T \wedge G(t,v) \textbf{ then } S_d(t,v) \textbf{ end}.$$

The parameter t take its value non-deterministically from the set T in such a way the guard $G(t,v)$ must be fulfilled. In our new version of Event-B, the value taken by the parameter t of the event is chosen in an uniform manner from the set T. This latter must be a finite set ($\{t_1, t_2 ..., t_n\}$). Specifically, this value is chosen from a subset $T'_v = \{t_m ..., t_k\}_{1 \leq m \leq n, 1 \leq k \leq n}$ such that for each value t' of the parameter in T'_v, we have $G(t = t', v) \models true$. Then we note this subset of value of parameters by $T'_v = \{t \in T \backslash G(t,v) \models true\}$, the different values of the parameter in T'_v depends also in the valuation of the different variables of our model when the guard is evaluated. We associate a probability function P_T such that for each value $t_z \in T$, $P_T(t = t_z) = P_T(t_z) = \frac{1}{card(T)}$. It's also the case for the subset T'_v where $P_{T'_v}(t = t_z) = P_{T_{v'}}(t_z) = \frac{1}{card(T'_v)}$.

Case 3: The action of the event is non-deterministic.
In the case where the action of the event is non-deterministic like presented below:

$$e_a = \textbf{any } t \textbf{ where } t \in T \wedge G(t,v) \textbf{ then } S_{nd}(t,v) \textbf{ end}$$

We denote a non-deterministic action by $S_{nd}(t,v)$, its full syntax is presented below:

$$\begin{array}{r} x_1 :\in \{E_{11}(t,v), E_{12}(t,v), ..., E_{1m_1}(t,v)\}\ (S_1(t,v)) \\ || \ x_2 :\in \{E_{21}(t,v), E_{22}(t,v), ..., E_{2m_2}(t,v)\}\ (S_2(t,v)) \\ ... \\ || \ x_k :\in \{E_{k1}(t,v), E_{k2}(t,v), ..., E_{km_k}(t,v)\}\ (S_n(t,v)) \end{array}$$

The action of the event e_a assigns to each variable x_i of our model an expression E_{ij} $(1 \leq j \leq m_i)$ that is chosen non-deterministically from a finite set of expression $\{E_{i1}(t,v), E_{i2}(t,v), ..., E_{im_1}(t,v)\}$. m_i represents the number of expressions that can be assigned to the variable x_i. We propose a new probabilistic action $S_p(t,v)$. The syntax of this action is presented below:

$$\begin{array}{r} x_1 := E_{11}(t,v) \oplus @_{p_{11}} ... E_{1m_1}(t,v) \oplus @_{p_{1m_1}} \ (S_1(t,v)) \\ || \ x_2 := E_{21}(t,v) \oplus @_{p_{21}} ... E_{2m_2}(t,v) \oplus @_{p_{2m_2}} \ (S_2(t,v)) \\ ... \\ || \ x_k := E_{k1}(t,v) \oplus @_{pk1} ... E_{km_k}(t,v) \oplus @p_{km_k} \ (S_n(t,v)) \end{array}$$

For each variable x_i of our model $(1 \leq i \leq k)$, there is a positive number (m_i) of expressions $E_{i,j}$ $(1 \leq j \leq m_i)$ that can be assigned to x_i each one with probability p_{ij} $(1 \leq j \leq m_i \wedge 0 \leq p_{ij} \leq 1)$. For each x_i, we have $\sum_{j=0}^{m_i} p_{ij} = 1$. The syntax of the new probabilistic event then will be:

$$e_a = \textbf{begin weight } w \textbf{ any } t \text{ where } t \in T \wedge G(t,v) \textbf{ then } S_p(t,v) \textbf{ end}.$$

4.2 Proof obligations related to the new proposition

We investigate in this part the modifications that can be done in the standard proof obligations, we express also some new proof obligations related to the new kind of events presented above.

In the following, we use the following notations. If the action of an event e_a is probabilistic, then we denote by $< e_a/E_{ij} >$ the fact that the action of the event e_a will assign the expression E_{ij} to the variable x_i, we denote also by $P_{x_i}(< ea/E_{ij} >)$ the probability that the expression E_{ij} will be assigned to the variable x_i. If the action of the event is deterministic, then $P_{x_i}(< ea/E_{ij} >)$ is equal to 1. We denote also by $< e_a/E_{1j_1}; E_{2j_2}; ...; E_{kj_k} >$ the fact that the action of the event e_a will assign to each of the variables of our model $(x_1, ..., x_k)$ respectively the expressions $(E_{1j_1}; E_{2j_2}; ...; E_{kj_k})$. We denote by $\gamma = \{j_1, j_2, ..., j_k\}$ the combination of expressions that will be assigned to the variables $(x_1, ..., x_k)$ such that $\gamma(i) = j_i \ \forall \ 1 \leq i \leq k$. The probability of this assignement is:

$$P(e_a/\gamma) = \prod_{i=1}^{k} P_{x_i}(< ea/E_{ij_i} >)$$

If the action of the event e_a is deterministic (S_d instead of S_p), then:

$$P(e_a/\gamma) = \prod_{i=1}^{k} P_{x_i}(< ea/E_{ij_i} >) = 1$$

For the new kind of events presented in Section 4.1, the standard proof obligations have the same form except that for the proof obligations related to termination, we add three new types of events:

- **Ordinary probabilistic**: this kind of event is like an ordinary event but it occurs with a positive weight, it does not put any restriction upon the variant.
- **Convergent probabilistic**: this kind of event is like a convergent event but likewise, it occurs with a positive weight, it must decrease the variant each time it's executed.
- **Anticipated probabilistic**: this kind of event is like an anticipated event except that it occurs with a positive weight, each time it's executed, it must not increase the variant.

The proof obligations related to termination keep the same form as standard events. We now introduce our new proof obligations.

4.2.1 The weight of a probabilistic event must be greater than 0

This proof aims to ensure that the weight of a probabilistic event must be strictly greater than 0. Let *evt* be a probabilistic event like presented below:

evt
weight
w
any
$t \in T$
where
$G(c, v, t)$
then
$S_p(c, v, t)$
end

The new proof obligation for this event is:

⊢ The weight of occurence of the event is strictly greater than 0.	⊢ w**>0**

4.2.2 Finiteness of the set of values of the parameter of an event

Another proof obligation that is derived for our probabilistic event mention that the set T of values that can be taken by the parameter must be finite:

⊢ The set of values taken by the parameter must be finite.	⊢ *finite(T)*

4.2.3 Finiteness of the set of expressions that can be assigned to a variable

The last new proof obligation concerns the probabilistic action ($S_p(t, v)$) of the probabilistic event:

$$\begin{array}{c} x_1 := E_{11}(t, v) \oplus_{p_{11}} ... \oplus_{p_{1m_1}} E_{1m_1}(t, v)\ (S_1(t, v)) \\ x_2 := E_{21}(t, v) \oplus_{p_{21}} ... \oplus_{p_{2m_2}} E_{2m_2}(t, v)\ (S_2(t, v)) \\ ... \\ x_k := E_{k1}(t, v) \oplus_{p_{k1}} ... \oplus_{p_{km_k}} E_{km_k}(t, v)\ (S_n(t, v)) \end{array}$$

Like mentionned in the previous section, each variable x_i of our model ($1 \leq i \leq k$) can take one of the expressions E_{ij} ($(1 \leq j \leq m_i)$ each one with a probability p_{ij}. This proof ensure that the sum of the probabilities of the expressions that can be assigned to each variable x_i must be equal to one:

$$\forall x_i, 1 \leq i \leq k, \sum_{j=0}^{m_i} p_{ij} = 1$$

Then for the event presented above, we have:

⊢ $\forall x_i, \sum_{j=0}^{m_i} P_{x_i}(< evt/E_{ij} >) = 1$

5 Probabilistic Event-B: Semantics

We present in this section the semantics of a probabilistic Event-B model.

5.1 Background

We give in the following some basic definitions:

Definition 1 (Probabilistic Event-B model)
A probabilistic Event-B model is a tuple M=(X, Init, Inv, Evts) where:

- $X = (x_1, x_2...., x_k)$ *: the variables of the Event-B model, these variables take their values from the set* $D = D_1 \times D_2 \times .... \times D_k$ *and must satisfy the invariant Inv.*
- *Init: the initialization event,*
- *Inv: the invariant of our model that must be satisfied by the different valuations of the variables of our model.*
- *Evts: the events of our model, each event occurs with a positive weight w. Each event occurs if its guard is enabled, we denote the guard of each event* e_a *in our model by* $grd(e_a)$*. We denote by* $\alpha(M)$ *the set of event names in M (suppose M contains two events named respectively* ea_1 *and* ea_2*, then* $\alpha(M)=\{ea_1, ea_2\}$*). The general form of an event in Evts is:*

$$e_a = \textbf{weight}\ w\ any\ t\ \textbf{where}\ t \in T \wedge G(t,v)\ \textbf{then}\ S_p(t,v)\ \textbf{end}$$

An event can be written in one of the following forms:

1. *the intialization event:*
$$e_{a_1} = \textbf{weight}\ w\ S_d(v)\ \textbf{end}$$
2. *when the event does not have a parameter and its action is deterministic:*
$$e_{a_2} = \textbf{weight}\ w\ \textbf{when}\ G(v)\ \textbf{then}\ S_d(v)\ \textbf{end}$$
3. *when the event does not have a parameter and its action is probabilistic:*
$$e_{a_3} = \textbf{weight}\ w\ \textbf{when}\ G(v)\ \textbf{then}\ S_p(v)\ \textbf{end}$$
4. *when the event has a parameter and its action is deterministic:*
$$e_{a_4} = \textbf{weight}\ w\ any\ t\ \textbf{where}\ t \in T \wedge G(t,v)\ \textbf{then}\ S_d(t,v)\ \textbf{end}$$

Definition 2 (*discrete-time* Markov chain)
We define (discrete-time) Markov chain as a tuple $\mathcal{M}$*=(S, P,* s_0*, AP, L, Acts) where:*

- *S is a countable, nonempty set of states,*
- *Acts is a set of actions ,*
- ***P**: S × Acts × S → [0,1] is the transition probability function such that for all states s:*

$$\sum_{s' \in S, a \in Acts} P(s,a,s') = 1$$

- s_0 *: the initial state,*
- *AP is a set of atomic propositions and L:* $S \rightarrow 2^{AP}$ *a labeling function.*

Definition 3 (Semantics of a Probabilistic Event-B model)
We generate from the Event-B model M presented above a discrete time markov chain $[\![M]\!]$*= (S, P,* l_{init}*, AP, L, Acts) where:*

- *S the set of states of the Markov chain, they are formed accordingly to the different valuations of the variables of the model M, the semantic interpretation of a state over X is defined by the labeling function L that associate for a given state s, for each variable $x_i \in X$, a value $v_i \in D_i$. We denote by $L(s_i, x_j)$ the current value of the variable x_j in the state s_i.*

- *l_{init} is the initial distribution such that $\sum_{s \in S} l_{init}(s) = 1$. The initial state of the Markov chain is obtained after the execution of the Init event, and $L(s_0) = \{ x_j \mapsto v_j \mid < Init > (x_j = v_j)\} \; \forall j \in 1..k$.*

- *AP represents the different assignments to the different variables of our model, these assignments must satisfy the invariant Inv of the Event-B model. $AP = \{x_i \mapsto d_j | (x_i \mapsto d_j) \models Inv\}$ $\forall x_i \in X \wedge d_j \in D_i$.*

- *Acts is the alphabet of events of the Event-B model M.*

- *P: $S \times Acts \times S \rightarrow$ [0,1] is the transition probability function such that for all state s:*

$$\sum_{s' \in S, e' \in Acts} P(s, e', s') = 1$$

. A transition in this markov chain corresponds to an occurence of an event in the Event-B model M. Given a state s of M, we denote the set of actions (events) enabled in this state by

$$Acts(s) = \{E_j / [L(s)] grd(E_j) \models true\}$$

The state destination of a transition (s, e_a, s') performed by an event e_a is labelled by:

$$Ł(s') = \{x_j \rightarrow v'_j | [L(s)](< e_a(x_j = v'_j) >)\} \forall j \in 1..k$$

If the action of the event e_a is probabilistic:

$$S_p(t, v) \simeq \begin{array}{l} x_1 := E_{11}(t, v) \oplus @_{p_{11}} ... E_{1m_1}(t, v) \; \oplus @_{p_{1m_1}} \; (S_1(t, v)) \\ x_2 := E_{21}(t, v) \oplus @_{p_{21}} ... E_{2m_2}(t, v) \; \oplus @_{p_{2m_2}} \; (S_2(t, v)) \\ ... \\ x_k := E_{k1}(t, v) \oplus @_{p_{k1}} ... E_{km_k}(t, v) \; \oplus @_{p_{km_k}} \; (S_n(t, v)) \end{array}$$

then we denote a combination of expressions that can be assigned to the variables x_i $(1 \leq i \leq k)$ by the action of the event e_a taking the system to a state s' by :

$$\gamma^s_{e_a}(s') = \{\gamma = (j_1, ..., j_k) | L(s') = \{x_i \rightarrow E_{i\gamma(i)} | [L(s)](e_a(x_i = E_{i\gamma(i)}))\}\} \forall i \in 1..k$$

From the state s, if there are many combinations of expressions that can be assigned to the variables x_i $(1 \leq i \leq k)$ taking the system to the same state s', then we denote these combinations by:

$$\beta^s_{e_a}(s') = \{\beta = (\gamma_1, ..., \gamma_t) | \forall \gamma_z \in \beta, L(s') = \{x_i \rightarrow E_{i\gamma_z(i)} | [L(s)](e_a(x_i = E_{i\gamma_z(i)}))\}\} \forall i \in 1..k$$

The probability of occurence of the event e_a in the state s taking the system to the state s' is:

$$P(s, e_a, s') = \frac{W(e_a)}{\sum_{E_j \in Acts(s)} W(E_j)} \times \sum_{t_z \in T'_v} P_{T'_v}(t_z) \times \sum_{\gamma \in \beta^s_{e_a}(s')} \prod_{i=1}^{k} P_{x_i}(e_a / E_{i\gamma(i)}(t_z, v))$$

Many cases can be derived from this formula:

1. *If the event e_a does not have a parameter t and its action is probabilistic, then the probability of the transition from s to s' is:*

$$P(s, e_a, s') = \frac{W(e_a)}{\sum_{E_j \in Acts(s)} W(E_j)} \times \sum_{\beta \in \beta^s_{e_a}(s')} \prod_{i=1}^{k} P_{x_i}(e_a / E_{i\beta(i)}(v))$$

2. *If the event e_a does not have a parameter t and its action is deterministic, then the probability of the transition from s to s' is:*

$$P(s, e_a, s') = \frac{W(e_a)}{\sum_{E_j \in Acts(s)} W(E_j)}$$

3. *If the event e_a have a parameter t and its action is deterministic, then the probability of the transition from s to s' is:*

$$P(s, e_a, s') = \frac{W(e_a)}{\sum_{E_j \in Acts(s)} W(E_j)} \times \sum_{t_z \in T'_v} P_{T'_v}(t_z)$$

Proof. [$[\![M]\!]$ is a *discrete-time* Markov chain]
We prove in this section that $[\![M]\!]$ is a discrete-time Markov chain. we must prove then that for each state s in $[\![M]\!]$, the sum of probabilities of transitions enbaled in s is equal to 1:

$$\sum_{s' \in S, e \in Acts(s)} P(s, e, s') = 1$$

Let s be a state of $[\![M]\!]$ and let e_a an event enabled in s, the action of e_a is presented below:

$$\begin{array}{l} x_1 := E_{11}(t, v) \oplus @_{p_{11}} ... E_{1m_1}(t, v) \oplus @_{p_{1m_1}} \\ x_2 := E_{21}(t, v) \oplus @_{p_{21}} ... E_{2m_2}(t, v) \oplus @_{p_{2m_2}} \\ ... \\ x_k := E_{k1}(t, v) \oplus @_{pk1} ... E_{km_k}(t, v) \oplus @_{p_{km_k}} \end{array}$$

There are $c = \prod_{i=1}^{k} m_i$ combinations of expressions that can be assigned to the variables x_i $(1 \leq i \leq k)$. Each combination take the system to a state s', many combinations can take the system to the same state s'.

The probability of a transition (s, e_a, s') is:

$$P(s, e_a, s') = \frac{W(e_a)}{\sum_{E_j \in Acts(s)} W(E_j)} \times \sum_{t_z \in T'_v} P_{T'_v}(t_z) \times \sum_{\gamma \in \beta^s_{e_a}(s')} \prod_{i=1}^{k} P_{x_i}(e_a / E_{i\gamma(i)}(t_z, v))$$

The sum of probabilities of transitions performed by the event(action) e_a is:

$$\sum_{s' \in S} P(s, e_a, s') = \sum_{s' \in S} \frac{W(e_a)}{\sum_{E_j \in Acts(s)} W(E_j)} \times \sum_{t_z \in T'_v} P_{T'_v}(t_z) \times \sum_{\gamma \in \beta^s_{e_a}(s')} \prod_{i=1}^{k} P_{x_i}(e_a / E_{i\gamma(i)}(t_z, v))$$

$$= \frac{W(e_a)}{\sum_{E_j \in Acts(s)} W(E_j)} \times \sum_{t_z \in T'_v} P_{T'_v}(t_z) \sum_{s' \in S} \sum_{\gamma \in \beta^s_{e_a}(s')} \prod_{i=1}^{k} P_{x_i}(e_a / E_{i\gamma(i)}(t_z, v))$$

$$= \frac{W(e_a)}{\sum_{E_j \in Acts(s)} W(E_j)} \times \sum_{t_z \in T'_v} P_{T'_v}(t_z) \times \prod_{i=1}^{k} \sum_{j=1}^{m_i} P_{x_i}(e_a / E_{ij})$$

or $\forall x_i, \sum_{j=1}^{m_i} P_{x_i}(e_a/E_{ij}) = 1$, then $\prod_{i=1}^{k} \sum_{j=1}^{m_i} P_{x_i}(e_a/E_{ij}) = 1$
and we have $\sum_{t_z \in T'_v} P_{T'_v(t_z)} = 1$ then:

$$\sum_{s' \in S} P(s, e_a, s') = \frac{W(e_a)}{\sum_{E_j \in Acts(s)} W(E_j)} \times \sum_{t_z \in T'_v} P_{T'_v}(t_z) \times \prod_{i=1}^{k} \sum_{j=1}^{m_i} P_{x_i}(e_a/E_{ij})$$

$$= \frac{W(e_a)}{\sum_{E_j \in Acts(s)} W(E_j)}$$

Then, the sum of probabilities of transitions enabled in the state s is:

$$\sum_{s' \in S, e \in Acts(s)} P(s, e, s') = \sum_{e \in Acts(s)} \frac{W(e)}{\sum_{e \in Acts(s)} W(e)} = \sum_{e \in Acts(s)} W(e).\frac{1}{\sum_{e \in Acts(s)} W(e)} = 1$$

and then:

$$\sum_{s' \in S, e \in Acts(s)} P(s, e, s') = 1$$

□

Example 1 (Example 1: Markov chains semantics of a probabilistic Event-B model)
We present below an example of transformation of a probabilistic Event-B model to its corresponding discrete-time Markov chain. We begin firstly by presenting the probabilistic Event-B model, after that we present the corresponding discrete-time Markov chain.

Probabilistic Event-B model.
We present below the probabilistic Event-B model, we note that the guards of the first event are independent from the parameter.

Model *example 1*
Variables x, y
Invariant $x, y \in \mathbb{N}$
Events

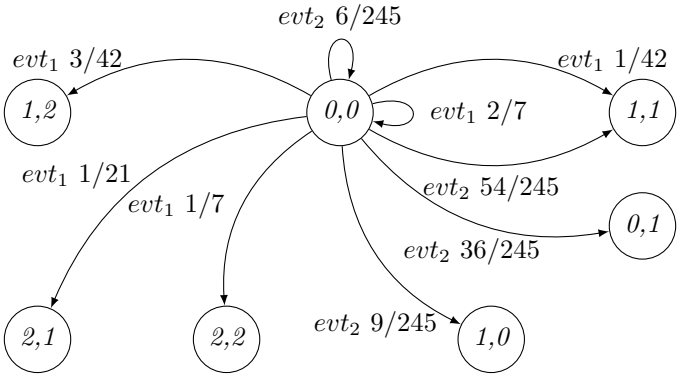
evt1 $\cong$ ***weight*** 4 ***any*** t ***where*** $(t = 0 \ @ \ \frac{1}{2}, t = 1 \ @ \ \frac{1}{2}) \wedge (x < 1 \wedge y < 1)$ ***then***

$$x := x + t \ {}_{1/3}\oplus \ x + 2t \ (S_{11})$$
$$y := y + t \ {}_{1/4}\oplus \ y + 2t \ (S_{22})$$
end

evt2 $\cong$ ***weight*** 3 ***when*** $(x < 1 \wedge y < 1)$ ***then***

$$x := x \ {}_{2/5}\oplus \ x + 1 \ (S_{21})$$
$$y := y \ {}_{1/7}\oplus \ y + 1 \ (S_{22})$$
end

Operational semantics.
We present below the corresponding discrete-time Markov chain of the Event-B model presented above:



Example 2 (Example 2: Markov chains semantics of a probabilistic Event-B model)
We present below an example of transformation of a probabilistic Event-B model to its corresponding discrete-time Markov chain. We begin firstly by presenting the probabilistic Event-B model after that we present the corresponding discrete-time Markov chain.

Probabilistic Event-B model.
We present below the probabilistic Event-B model, the only difference with the previous example is that the guards of the first event of this example depends on the value taken by the parameter of the event:

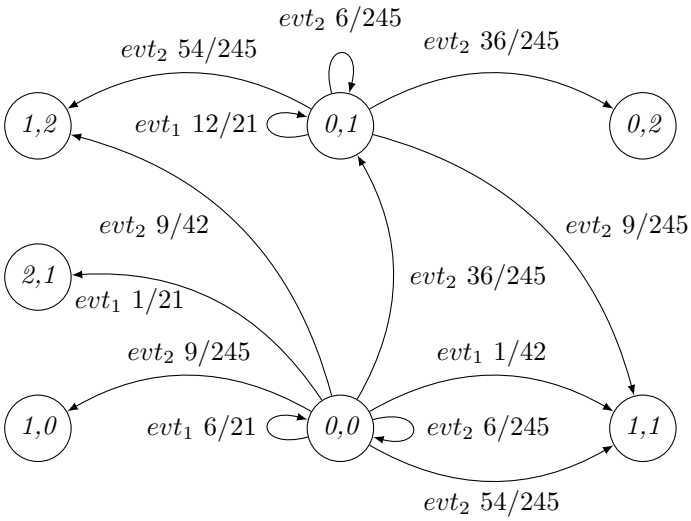
Model *example2*
Variables *x,y*
Invariant $x,y \in \mathbb{N}$
Events
evt1 $\cong$ ***weight*** *4* ***any*** t ***where*** $(t = 0 \ @ \ \frac{1}{2},\ t = 1 \ @ \ \frac{1}{2}) \wedge (x + t < 2 \wedge y + t < 2)$ ***then***

$$x := x + t \ {}_{1/3}\oplus \ x + 2t \ (S_{11})$$
$$y := y + t \ {}_{1/4}\oplus \ y + 2t \ (S_{22})$$
end

evt2 $\cong$ ***weight*** *3* ***when*** $(x < 1 \wedge y < 2)$ ***then***

$$x := x \ {}_{2/5}\oplus \ x + 1 \ (S_{21})$$
$$y := y \ {}_{1/7}\oplus \ y + 1 \ (S_{22})$$
end

Operational semantics.
We present below the corresponding discrete-time Markov chain of the Event-B model presented above, the only difference from the Markov chain derived from the Event-B model presented above is the fact that in the state $(0, 1)$*, the event evt*1 *is enabled (guard fulfilled) only if the parameter* t *take the value* 0*, so that, the subset of values for the parameter* t *in this state is the singleton* $\{0\}$ *and the parameter take this value with probability* 1*.*



6 Probabilistic Simulation

Based in the theory of probabilistic simulation and bissimulation, we have proposed a new theorem for the refinement between discrete-time Markov chains. We begin firstly by presenting some basic defintions about Probabilistic systems. After that, we present our new theorem and its corresponding proof.

6.1 Probabilistic Systems

We outline our theoreom in this section, beginning with some basic definitions about probabilistic systems.

Definition 4 (Probability Space)
A probability space is a triplet $(\Omega, \mathbb{F}, P)$ *where* Ω *is a set,* $\mathbb{F}$ *is a collection of subsets of* Ω *that is closed under complement and countable union and P is a function from* $\mathbb{F}$ *to [0,1] such that P[*Ω*]=1 and for any collection* $\{C_i\}$ *of at most countably many stepwise disjoint elements of* $\mathbb{F}$, $P[\cup_i C_i] = \sum_i P[C_i]$. *A probability space* $(\Omega, \mathbb{F}, P)$ *is discrete if* $\mathbb{F} = 2^{\Omega}$ *and for each* $C \subseteq \Omega$, $P[C] = \sum_{x \in C} P[x]$.

Definition 5 (Automata)
Formally, an automata A is composed of:

- *a set S of states,*
- *a subset* S_0 *of initial states,*
- *a set of actions Acts= {ext(A), int(A)} where ext(A) denotes the set of external actions and int(A) denotes the set of internal actions,*
- *a set of transitions* $trans(A) \subseteq S \times Acts \times S$ *that describes the different behaviours of the system describes by this automata.*

Definition 6 (Probabilistic Automata [35])
A Probabilistic Automata M is an automata where transition relations $trans(A)$ *is a subset of* $S(a) \times Probs(acts(A) \times S(A) \cup \delta)$ *where* $Probs(X)$ *is the set of discrete probability space* $(\Omega, \mathbb{F}, P)$ *where* $\Omega \subseteq X$.

A Probabilistic Automata M is simple if for each transition $trans(s, (\Omega, \mathbb{F}, P)) \in trans(M)$ there is an action $a \in acts(M)$ such that $\Omega \subseteq a \times S(A)$, in this case, a transition can be represented as $(s, a, (\Omega, \mathbb{F}, P))$ where $Probs(A) \in Probs(S(A))$ and is called a simple transition with action a and then the action is the same but it take the system to different states when the choice between this different states is done probalistically. A Probabilistic automata is fully probabilistic if it has a unique start state

and from each state, there is at most one step enabled. Thus a probabilistic automata differs from an automata in that the action and the next state of a given transition are chosen probabilistically. The symbol δ represents the situation when the system deadlocks, it's then possible that from a state s , a probabilistic automata performs some action with probability p and deadlock with probability $1-p$.

Definition 7 (Equivalence Relation [35])
Let $\mathbb{R}$ be an equivalence relation over a set X. Two probability spaces $(\Omega_1, \mathbb{F}_1, P_1)$ and $(\Omega_2, \mathbb{F}_2, P_2)$ of $Probs(X)$ are R_equivalent written $(\Omega_1, \mathbb{F}_1, P_1) \equiv_R (\Omega_2, \mathbb{F}_2, P_2)$ if for each $[x_1]_R \in \Omega_1/R$ there exists an $[x_2]_R \in \Omega_2/R$ such that $x_1\ R\ x_2$, for each $[x_2]_R \in \Omega_2/R$ there exists an $[x_1]_R \in \Omega_1/R$ such that $x_2\ R\ x_1$, and for each $[x_1]_R \in \Omega_1/R [x_2]_R \in \Omega_2/R$ such that $x_1 R x_2, \sum_{x\in\Omega_1\cap[x_1]_R} P[x] = \sum_{x\in\Omega_2\cap[x_2]_R} P[x]$. In other worlds $(\Omega_1, \mathbb{F}_1, P_1)$ and $(\Omega_2, \mathbb{F}_2, P_2)$ are R_equivalent if they assign the same probability measure to each equivalence class of $\mathbb{R}$.

Definition 8 (Weight function [35])
Let $R \subseteq X \times Y$ be a relation between two set X,Y, and let $(\Omega_1, \mathbb{F}_1, P_1)$ and $(\Omega_2, \mathbb{F}_2, P_2)$ be two probability spaces of $Probs(X)$ and $Probs(Y)$, respectively. Then $(\Omega_1, \mathbb{F}_1, P_1)$ and $(\Omega_2, \mathbb{F}_2, P_2)$ are in relation $\subseteq_R$, written $(\Omega_1, \mathbb{F}_1, P_1) \subseteq_R (\Omega_2, \mathbb{F}_2, P_2)$ if there exists a weight function $w : X \times Y \to [0, 1]$ such that:

- *for each $x \in X, \sum_{y\in Y} w(x, y) = P_1[x]$,*
- *for each $y \in Y, \sum_{x\in X} w(x, y) = P_2[y]$,*
- *for each $(x, y) \in X \times Y, if w(x, y) > 0$ then xRy.*

Simulation and Bissimulation between probabilistic automata
Simulation ($\preceq$) and bisimulation relations ($\sim$) have been widely used for the comparison of the stepwise behaviour of states in labeled transition systems [12, Ch2]. Bisimulation relation [29, 28, 31] requires that two bisimilar states exhibit an identical stepwise behaviour whereas Simulation relations [1, 18, 22, 26, 31] are preorders over the state space requiring that whenever $s \preceq s'$ (s' simulate s), state s' can emerge all the stepwise behaviour of s, the converse $s' \preceq s$ is not guaranteed and then state s' can perform some steps that cannot be matched by s. Simulation and bisimulation relations was also extended to the case of probabilistic systems and several studies has been also carried out to define probabilistic simulation and Bissimulation relations, for a more details about these works, see the research papers [23, 25, 7, 24, 11, 32, 35].

Definition 9 (Strong Probabilistic Simulation [35])
A strong probabilistic simulation between two simple probabilistic automata M_1, M_2, is a relation $\subseteq$ states(M_1) $\times$ states(M_2) such that:

- *each start state of M_1 is related to at least one start state of M_2,*
- *for each $s_1 R s_2$ and each step $s_1 \xrightarrow{a} (\Omega_1, \mathbb{F}_1, P_1) of M_1$, there exists a step $s_2 \xrightarrow{a} (\Omega_2, \mathbb{F}_2, P_2)$ of M_2 such that $(\Omega_1, \mathbb{F}_1, P_1) \subseteq_R (\Omega_2, \mathbb{F}_2, P_2)$,*
- *for each $s_1 R s_2$, if $s_2 \xrightarrow{a}$, then $s_1 \xrightarrow{a}$.*

The strong probabilistic simulation relation presented above concerns simple probabilistic automata when the probabilistic choice concerns the choice of the next state from a set of states by executing the same action a. Based on this defintion and the defintion of probabilistic bisimulation for probabilistic automata, we will define a new relation for probabilistic simulation between two discrete-time Markov chains.

Definition 10 ((*discrete-time*) Markov chain)
We define (discrete-time) Markov chain as a tuple $\mathcal{M}$=(S, P, s_0, AP, L, Acts) where:

- *S is a countable, nonempty set of states,*

- *Acts is a set of actions ,*
- ***P**: $S \times Acts \times S \rightarrow [0,1]$ is the transition probability function such that for all states s:*

$$\sum_{s' \in S, a \in Acts} P(s, a, s') = 1$$

- *s_0 : the initial state,*
- *AP is a set of atomic propositions and L: $S \rightarrow 2^{AP}$ a labeling function.*

Definition 11 (Equivalence Class)
Let S be a set of states and R an equivalence relation on S. For $s \in S$, $[s]_R$ denotes the equivalence class of state s under R, i.e, $[s]_R = \{s' \in S | (s, s') \in R\}$. Note that for $s' \in [s]_R$, we have $[s']_R = [s]_R$. The set $[s]_R$ is often referred to as the R-equivalence class of s. The quotient space of S under R, denoted by $S/R = \{[s]_R | s \in S\}$ is the set consisting of all R-equivalence classes.

Theorem 1 (Discrete-time Markov chains refinement)
We define in this section a new relation for the refinement between discrete-time Markov chains. Let M_a=(S_a, P_a, s_0, AP_a, L_a, $Acts_a$) and M_c=(S_c, P_c, t_0, AP_c, L_c, $Acts_c$) be two discrete-time Markov chains.
Given a relation L_{eq} between the set of states S_a and S_c of the two Markov chains:

$$L_{eq} \subseteq S_a \times S_c$$

For each two states s_a, s_c of M_a and M_c respectively, if the valuations of s_a and s_c agree, then $(s_a, s_c) \in L_{eq}$.
Given a relation E_{eq} between the actions of the two Markov chains

$$E_{eq} \subseteq Acts_a \times Acts_c$$

The relation E_{eq} specifies for two actions e_c and e_a of M_c and M_a respectively if e_c refine e_a and then $(e_c, e_a) \in E_{eq}$.
A relation $R_r \subseteq S_c \times S_a$ is a refinement relation between M_c and M_a if whenever $t\ R_r\ s$ where $t \in S_c$ and $s \in S_a$ we have:

1. $t\ L_{eq}\ s \iff (t, s) \in L_{eq}$
2. *there exists a function $\delta : (Acts_c \times S_c) \rightarrow ((Acts_a \times S_a) \rightarrow [0, 1])$ such that:*
   a. *for all $e_c \in Acts_c$, $t' \in S_c$ such that $P_c(t, e_c, t') > 0$, $\delta(e_c, t')$ is a distribution on $(Evts_a \times S_a)$*
   b. *$\forall\ s', e_a \in S_a \times Acts_a$ we have:*

$$\sum_{t' \in S_c, e_c \in Acts_c} P_c(t, e_c, t').\delta(e_c, t')(e_a, s') = P_a(s, e_a, s')$$

   c. *$\forall\ t', s', e_c, e_a \in S_c \times S_a \times Acts_c \times Acts_a$, if $\delta(e_c, t')(e_a, s') > 0$ then $(t', s') \in R_r$*

We say that M_c refine M_a if there exits a refinement relation R_r such that $t_0 R_r s_0$.

Definition 12 (Restatement of Strong probabilistic simulation)
We reformulate the strong probabilistic simulation relation by Segala in [35]. Given two probabilistic automata M_1 and M_2, we define two probability spaces $(\Omega_1, \mathbb{F}_1, P_1)$ on $Acts(M_1) \times states(M_1)$ and $(\Omega_2, \mathbb{F}_2, P_2)$ on $Acts(M_2) \times states(M_2)$. A strong probabilistic simulation relation between M_1, M_2 is a relation $R_s \subseteq states(M_1) \times states(M_2)$ where:

1. *each start state of M_1 is related to at least one start state of M_2,*

2. *for each* $s_1\ R_s\ s_2$ *and each step* $s_1 \to (\Omega_1, \mathbb{F}_1, P_1)$ *of* M_1*, there exists a step* $s_2 \to (\Omega_2, \mathbb{F}_2, P_2)$ *of* M_2 *such that* $(\Omega_1, \mathbb{F}_1, P_1) \subseteq_R (\Omega_2, \mathbb{F}_2, P_2)$*, i.e, there exists a weight function* w*:* $\Omega_1 \times \Omega_2 \to [0,1]$ *such that:*

   a. *for each* $x \in \Omega_1, \sum_{y \in \Omega_2} w(x,y) = P_1[x]$,
   b. *for each* $y \in \Omega_2, \sum_{x \in \Omega_1} w(x,y) = P_2[y]$,
   c. *for each* $(x,y) \in \Omega_1 \times \Omega_2$ *where* $x = (e_c, s_c)$ *and* $y = (e_a, s_a)$*, if* $w(x,y) > 0$ *then* $s_c\ R_s\ s_a$.

Proof. [Proof of equivalence of the relations R_r and R_s]
We want to prove that the discrete-time Markov chains refinement relation R_r is equivalent to the rewritten strong probabilistic simulation R_s ($R_r \iff R_s$).
$\boxed{\Rightarrow}$: we begin by proving that the relation R_r implies the relation R_s. Suppose that we have two discrete-time Markov chains M_a and M_c such that M_c refines M_a ($M_c\ R_r\ M_a$). We want to prove that M_a simulate M_c ($M_c\ R_s\ M_a$) like defined in Section 12, we prove then the different items of R_s:

1. *each start state of* M_a *is related to at least one start state of* M_c:

   We have M_c refine M_a, then there exist a refinement relation between the initial states of M_c and M_a (given an initial state t_0 of M_c and an initial state s_0 of M_a, then $t_0\ R_r\ s_0$) then each start state of M_a is related to at least one start state of M_c.

2. *for each two states* t *and* s *of* M_c *and* M_a *respectively such that* $t\ R_s\ s$*, for each step* $t \to (\Omega_1, \mathbb{F}_1, P_1)$ *of* M_c*, there exists a step* $s \to (\Omega_2, \mathbb{F}_2, P_2)$ *of* M_a *such that there exist a weight function* w*:* $\Omega_1 \times \Omega_2 \to [0,1]$ *such that:*

   a. *for each* $x \in \Omega_1\ \sum_{y \in \Omega_2} w(x,y) = P_1[x]$ *($x = (e_c, t')$ represents an action e_c enabled in the state t and take the system to a state t' where $y = (e_a, s')$ represents an action e_a enabled in the state s and take the system to a state s')*

      Let $w(x,y) = P_1[x].\delta(x)(y)$,

      $$\sum_{y \in \Omega_2} w(x,y) = \sum_{y \in \Omega_2} P_1[x].\delta(x)(y) = P_1[x].\sum_{y \in \Omega_2} \delta(x)(y)$$

      Or from 2.*a* in Section 1, we have $\delta(e_c, t')$ is a distribution on $Acts_a \times S_a$, i.e:

      $$\sum_{e \in Acts_a, s_a \in S_a} \delta(e_c, t')(e, s_a) = \sum_{y \in \Omega_2} \delta(x)(y) = 1$$

      then:

      $$\sum_{y \in \Omega_2} P_1[x].\delta(x)(y) = P_1[x].\sum_{y \in \Omega_2} \delta(x)(y) = P_1[x].1 = P_1[x]$$

   b. *for each* $y \in \Omega_2\ \sum_{x \in \Omega_1}\ w(x,y) = P_2[y]$*:*

      Let $w(x,y) = P_1[x].\delta(x)(y)$, or from 2.*b* in 1, we have that $\forall\ s'$, $e_a \in S_a \times Acts_a$:

      $$\sum_{t' \in S_c, e_c \in Acts_c} P_c(t, e_c, t').\delta(e_c, t')(e_a, s') = P_a(s, e_a, s')$$

      i.e, $\sum_{x \in \Omega_1} P_1[x].\delta(x)(y) = P_2[y]$ then:

      $$\sum_{x \in \Omega_1} w(x,y) = P_2[y]$$

   c. $\forall x, y \in \Omega_1 \times \Omega_2$ *if* $w(x,y) > 0$ *where* $x = (e_c, s_c)$ *and* $y = (e_a, s_a)$*, then* $s_c R_s s_a$

      If $w(x,y) > 0$, then $P_1[x].\delta(x)(y) > 0$ or $P_1[x]$ is strictly greater than 0, then $\delta(x)(y) > 0$,i.e, $x\ R_r\ y \implies x\ R_s\ y$.

$\boxed{\Leftarrow}$: we prove now that R_s corresponds to R_r.
Suppose that we have two discrete-time Markov chains M_a and M_c such that M_a simulate M_c (M_c R_s M_a), suppose that we have two states t and s of M_c and M_a respectively such that t R_s s, we want to prove that t R_r s, we must then prove the different items of R_r:

1. $(t, s) \in L_{eq}$
   we have t R_s $s \Longrightarrow t$ R_r s and then $(t, s) \in L_{eq}$.
2. *there exits a function* δ: $(Acts_c \times S_c) \to ((Acts_a \times S_a) \to [0, 1])$ *such that:*
   a. $\forall$ e_c, $t' \in Acts_c \times S_c$ *such that* $P_c(t, e_c, t') > 0$, $\delta(e_c, t')$ *is a distribution on* $(Acts_a \times S_a)$, *i.e,*

$$\sum_{e_a \in Acts_a, s' \in S_a} \delta(e_c, t')(e_a, s') = 1$$

   we denote by $x = (e_c, t')$ an action e_c enabled in the state t that take the system to a state t' and by $y = (e_a, s')$ an action e_a enabled in s that take the system to a state s', we want to prove then that $\sum_{y \in \Omega_2} \delta(x)(y) = 1$.

$$\text{Let } \delta(x)(y) = \begin{cases} \frac{w(x,y)}{P_1[x]} \text{ if } P_1[x] \neq 0 \\ 0 \text{ if } P_1[x] = 0 \end{cases}$$

$$\sum_{y \in \Omega_2} \delta(x)(y) = \sum_{y \in \Omega_2} \frac{w(x, y)}{P_1[x]} = \frac{1}{P_1[x]} \times \sum_{y \in \Omega_2} w(x, y)$$

   or from 2.a in 12, $\forall x \in \Omega_1, \sum_{y \in \Omega_2} w(x, y) = P_1[x]$, then:

$$\sum_{y \in \Omega_2} \delta(x)(y) = 1$$

   Then, $\delta(x)$ is a distribution on $Acts_a \times S_a$, i.e, a distribution on Ω_2.
   b. $\forall$ s', $e_a \in S_a \times Acts_a$ *we have:*

$$\sum_{t' \in T, e_c \in Acts_c} P_c(t, e_c, t').\delta(e_c, t')(e_a, s') = P_a(s, e_a, s')$$

   By analogy with R_s, this sum can be rewritten as:

$$\sum_{x \in \Omega_1} P_1[x].\delta(x)(y) = \sum_{x \in \Omega_1} P_1[x].\frac{w(x, y)}{P_1[x]} = \sum_{x \in \Omega_1} w(x, y) = P_2[y]$$

   Then we obtain:

$$\sum_{t' \in T, e_c \in Acts_c} P_c(t, e_c, t').\delta(e_c, t').(e_a, s') = P_a(s, e_a, s')$$

   c. $\forall t', s', e_c, e_a \in S_c \times S_a \times Acts_c \times Acts_a$, *if* $\delta(e_c, t')(e_a, s') > 0$ *then* $(t', s') \in R_r$
   We denote (e_c, t') by x and (e_a, s') by y. we have taken previously $\delta(x, y) = \frac{w(x,y)}{P_1[x]}$, we have $\delta(x, y) > 0$, then $\frac{w(x,y)}{P_1[x]} > 0$ or $P_1[x] > 0$, then $w(x, y) > 0$ and then $t' R_s s'$, then t' R_r s'.

□

7 Refinement of a Probabilistic Event-B model

We present here our proposition of refinement between two probabilistic Event-B models. We provide in what follows the abstract and the concrete models, after that, we present the new proof obligations related to this notion of refinement.

Abstract Probabilistic Event-B model. M_a=(X_a, $Init_a$, Inv_a, $Evts_a$) is an abstract probabilistic Event-B model where:

- $X_a = (x_1, x_2, ..., x_k)$: the variables of M_a, these variables take their values from the set $D = D_1 \times D_2 \times .... \times D_k$ and must satisfy the invariant Inv_a.
- $Init_a$: the abstract initialization event,
- Inv_a: the invariant of our model that must be satisfied by the different valuations of the variables of M_a.
- $Evts_a$: the abstract events of M_a, each event occurs with a positive weight w_a, we denote by $\alpha_a(M_a)$ the set of events names in M_a.

Concrete Probabilistic Event-B model. M_c=(X_c, $Init_c$, Inv_c, $Evts_c$) is the concrete probabilistic Event-B model where:

- $X_c = (x_1, x_2, ..., x_k, ..., x_n)$: the variables of M_c, these variables take their values from the set $D = D_1 \times D_2 \times ... \times D_k \times D_n$ and must satisfy the invariant Inv_c.
- $Init_c$: the abstract initialization event,
- Inv_c: the gluing invariant that relate the variables of M_a to the variables of M_c.
- $Evts_c$: the concrete events of M_c, we do not allow the introduction of new events in $Evts_c$ ($\alpha_c(M_c)$= αM_a). Each probabilistic event in the abstract model is refined by only one probabilistic event.

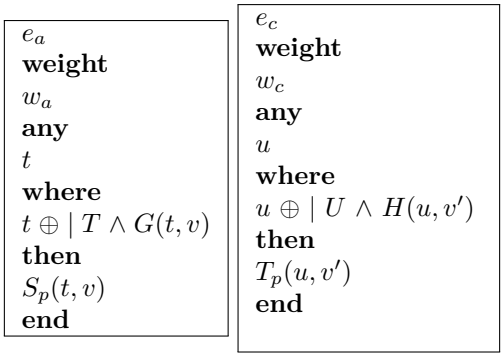
7.1 Refinement in Probabilistic Event-B

We introduce a new notion of refinement for probabilistic Event-B models. In the proposed notion, we do not allow the introduction of new events during refinement, only the introduction of new variables in the concrete model and the modification of events actions are permitted. The classical proof obligations remain applicable and unchanged for the refinement of probabilistic events. In addition, we propose some new proof obligations dedicated to this refinement. We note that the syntax of our proposition for probabilistic Event-B models is not yet complete and we have not yet define an operator for expressing the probability of an event in Event-B, so we express at the moment the new proofs using their corresponding Markov chains semantics.

7.1.1 New proof obligations related to refinement

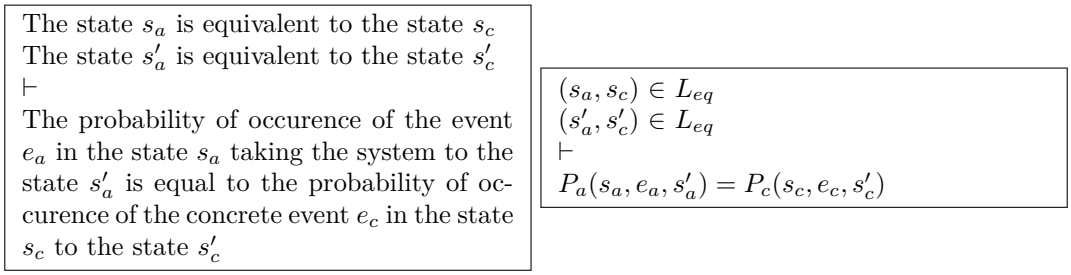
We present in this section the new proof obligations that must be verified in the case of the refinement of a probabilistic Event-B model by another probabilistic Event-B model. These proofs are expressed on the Markov chain semantics of both the concrete and the abstract models. Given two states s_a of s_c and s_c of $[\![M_a]\!]$ and $[\![M_c]\!]$ respectively. we say that s_a is equivalent to s_c only if the valuations of s_a and s_c, i.e, these valuations statisfies the gluing variant inv_c of the concrete Event-B model. We use the relation L_{eq} defined in theorem 1 to express that two states are equivalent.

Proof 1: Relation between the probability of occurence of an abstract event and that of its refined event.
This proof express the relation between the probability of occurence of an abstract event and the probability of occurence of its refined concrete event. Given an abstract probabilistic event e_a that is refined by a concrete probabilistic event e_c like presented below:

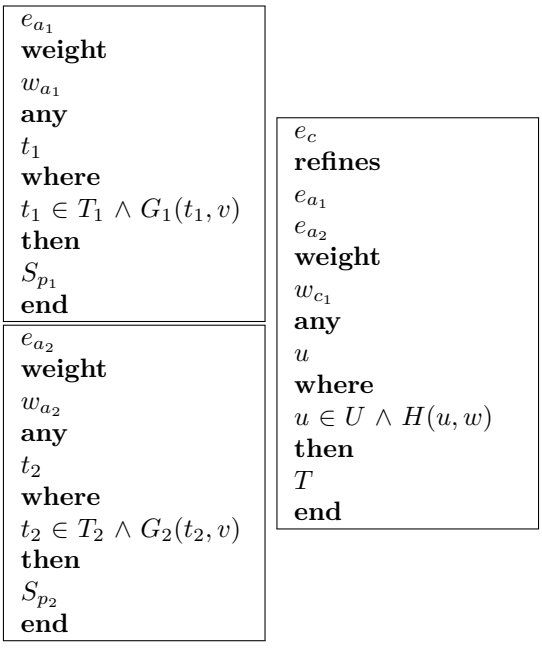


In the corresponding Markov chain semantics of M_a and M_c, suppose that the corresponding action of the event e_a is enabled in a state s_a of $[\![M_a]\!]$ and take $[\![M_a]\!]$ to a state s'_a. Suppose also that the corresponding action of the event e_c is enabled in a state s_c of $[\![M_c]\!]$ and take the system to a state s'_c such that $(s_a, s_c) \in L_{eq}$ and $(s'_a, s'_c) \in L_{eq}$. This proof obligation mention that the probability of occurence of the concrete event e_c in the state s_c taking the system to the state s'_c is equal to the probability of occurence of the abstract event e_a in the state s_a taking the system to the state s'_a:

$$P_a(s_a, e_a, s'_a) = P_c(s_c, e_c, s'_c)$$



Proof 2: Merging of two probabilistic events by a probabilistic event.
This proof is related to the case of merging of two abstract probabilistic events by a concrete probabilistic event. Suppose we have two abstract probabilistic events e_{a_1} and e_{a_2} that are merged by an event e_{c_1} like presented below:



In the corresponding Markov chain semantics of M_a and M_c, suppose that the abstract event e_{a_1} is

enabled in an abstract state s_{a_1} and the abstract event e_{a_2} is enabled in an abstract state s_{a_2} while the concrete event e_c is enabled in a concrete state s_c. We have three possibilities in this kind of refinement:

- If the state s_c is equivalent only to the state s_{a_1} but not to the state s_{a_2}, then the probability of occurence of the concrete event e_c in the state s_c taking the system to the state s'_c is equal to the probability of occurence of the abstract event e_{a_1} in the state s_{a_1} taking the system s'_{a_1}:

The state s_c is equivalent to the state s_{a_1} The state s_c is not equivalent to the state s_{a_2} $\vdash$ The probability of occurence of the event e_c in the state s_c taking the system to the state s'_c is equal to the probability of occurence of the abstract event e_{a_1} in the state s_{a_1} to the state s'_{a_1}	$(s_c,\, s_{a_1}) \in L_{eq}$ $(s_c,\, s_{a_2}) \notin L_{eq}$ $\vdash$ $P_c(s_c, e_c, s'_c) = P_a(s_{a_1}, e_{a_1}, s'_{a_1})$

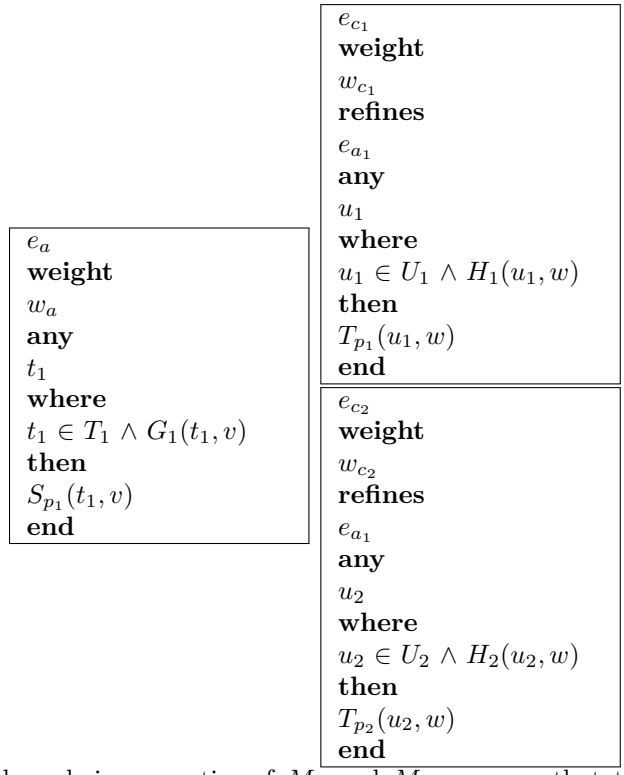
- If the state s_c is equivalent only to the state s_{a_2} but not to the state s_{a_1}, then the probability of occurence of the concrete event e_c in the state s_c taking the system to the state s'_c is equal to the probability of occurence of the abstract event e_{a_2} in the state s_{a_2} taking the system s'_{a_2}:

The state s_c is equivalent to the state s_{a_2} The state s_c is not equivalent to the state s_{a_1} $\vdash$ The probability of occurence of the event e_c in the state s_c taking the system to the state s'_c is equal to the probability of occurence of the occurence of the abstract event e_{a_2} in the state s_{a_2} to the state s'_{a_2}	$(s_c,\, s_{a_2}) \in L_{eq}$ $(s_c,\, s_{a_1}) \notin L_{eq}$ $\vdash$ $P_c(s_c, e_c, s'_c) = P_a(s_{a_2}, e_{a_2}, s'_{a_2})$

- If the state s_c is equivalent to both the states s_{a_1} and s_{a_2}, then the probability of occurence of the concrete event e_c in the state s_c taking the system to the state s'_c is equal to the sum of probabilities of occurence of the abstract events e_{a_1} in the state s_{a_1} taking the system to the state s'_{a_1} and e_{a_2} in the state s_{a_2} taking the system s'_{a_2}:

The state s_c is equivalent to the state s_{a_1} The state s_c is equivalent to the state s_{a_2} $\vdash$ The probability of occurence of the event e_c in the state s_c taking the system to the state s'_c is equal to the sum of probabilities of occurence of the abstract events e_{a_1} in the state s_{a_1} taking the system to the state s'_{a_1} and e_{a_2} in the state s_{a_2} taking the system s'_{a_2}	$(s_c,\, s_{a_1}) \in L_{eq}$ $(s_c,\, s_{a_2}) \in L_{eq}$ $\vdash$ $P_a(s_{a_1}, e_{a_1}, s'_{a_1}) + P_a(s_{a_2}, e_{a_2}, s'_{a_2}) = P_c(s_c, e_c, s'_c)$

Proof 3: Splitting of an abstract probabilistic event by two concrete probabilistic events. This proof is related to the case of splitting of a probabilistic event e_a by two others probabilistic events e_{c_1} and e_{c_2} like presented below:



In the corresponding Markov chain semantics of M_a and M_c, suppose that the abstract event e_a is enabled in an abstract state s_a and the concrete event e_{c_1} is enabled in a concrete state s_{c_1} while the concrete event e_{c_2} is enabled in a concrete state s_{c_2}. We have three possibilities in this kind of refinement:

- If the state s_a is equivalent only to the state s_{c_1} but not to the state s_{c_2}, then the probability of occurence of the concrete event e_{c_1} in the state s_{c_1} taking the system to the state s'_{c_1} is equal to the probability of occurence of the abstract event e_a in the state s_a taking the system to the state s'_a:

The state s_a is equivalent to the state s_{c_1} The state s_a is not equivalent to the state s_{c_2} $\vdash$ The probability of occurence of the event e_{c_1} in the state s_{c_1} taking the system to the state s'_{c_1} is equal to the probability of occurence of the occurence of the abstract event e_a in the state s_a to the state s'_a	$(s_{c_1},\, s_a) \in L_{eq}$ $(s_{c_2},\, s_a) \notin L_{eq}$ $\vdash$ $P_c(s_{c_1}, e_{c_1}, s'_{c_1}) = P_a(s_a, e_a, s'_a)$

- If the state s_a is equivalent only to the state s_{c_2} but not to the state s_{c_1}, then the probability of occurence of the concrete event e_{c_2} in the state s_{c_2} taking the system to the state s'_{c_2} is equal to the probability of occurence of the abstract event e_a in the state s_a taking the system s'_a:

The state s_a is equivalent to the state s_{c_2} The state s_a is not equivalent to the state s_{c_1} $\vdash$ The probability of occurence of the event e_{c_2} in the state s_{c_2} taking the system to the state s'_{c_2} is equal to the probability of occurence of the occurence of the abstract event e_a in the state s_a to the state s'_a	$(s_{c_2},\, s_a) \in L_{eq}$ $(s_{c_1},\, s_a) \notin L_{eq}$ $\vdash$ $P_c(s_{c_2}, e_{c_2}, s'_{c_2}) = P_a(s_a, e_a, s'_a)$

- If the state s_a is equivalent to both the states s_{c_1} and s_{c_2}, then the probability of occurence of the abstract event e_a in the state s_a taking the system to the state s'_a is equal to the sum of probabilities of occurence of the abstract events e_{c_1} in the state s_{c_1} taking the system to the state s'_{c_1} and and e_{c_2} in the state s_{c_2} taking the system s'_{c_2}:

The state s_a is equivalent to the state s_{c_1} The state s_a is equivalent to the state s_{c_2} $\vdash$ The probability of occurence of the abstract event e_a in the state s_a taking the system to the state s'_a is equal to the sum of probabilities of occurence of the concrete events e_{c_1} in the state s_{c_1} taking the system to the state s'_{c_1} and and e_{c_2} in the state s_{c_2} taking the system s'_{c_2}	$(s_{c_1}, s_a) \in L_{eq}$ $(s_{c_2}, s_a) \in L_{eq}$ $\vdash$ $P_a(s_a, e_a, s'_a) = P_c(s_{c_1}, e_{c_1}, s'_{c_1}) + P_c(s_{c_2}, e_{c_2}, s'_{c_2})$

7.2 Equivalence to Markov Chain refinement

We propose in this section a new theorem expressing the relation between the refinement relation between probabilistic Event-B models and the refinement relation between their corresponding Markov chains semantics. We denote the refinement relation between probabilistic Event-B models by peb_R.

Theorem 2 *The refinement relation peb_R between two probabilistic Event-B models M_a and M_c is equivalent to the refinement relation between their corresponding discrete-time Markov chains $[\![M_a]\!]$ and $[\![M_c]\!]$ ($peb_R \iff R_r$). If we have M_c peb_R M_a, then $[\![M_c]\!]$ R_r $[\![M_a]\!]$.*

Proof. Let M_a and M_c be two probabilistic Event-B models such that M_c refines M_a and let $[\![M_a]\!]$ and $[\![M_c]\!]$ the corresponding *discrete-time* Markov chains:

We must prove that the relation peb_R is equivalent to R_r ($peb_R \iff R_s$).
$\boxed{\Rightarrow}$: We begin by proving that the relation peb_R implies the relation R_r. We have $M_c peb_R M_a$, we want to prove that $[\![M_c]\!]$ R_r $[\![M_a]\!]$.

The states S_a and S_c of the two Markov chains corresponds respectively to the different valuations of the variables of M_a and M_c. The equivalence relation L_{eq} between the set of states S_a and S_c corresponds to the gluing invariant Inv_c, two states s_a and s_c are equivalent if their valuations satisfies the invariant Inv_c. The actions of each Markov chain corresponds to the names of the events in the corresponding Event-B model, then $Acts_a = \alpha_a(M_a)$ and $Acts_c = \alpha_c(M_c)$. The equivalence relation between actions E_{eq} specifies if an event refines another one, if a concrete event named e_c refine an abstract event named e_a then we have e_c E_{eq} e_a.
The relation $R_r \subseteq S_c \times S_a$ is a refinement relation between $[\![M_c]\!]$ and $[\![M_a]\!]$ if whenever $tR_r s$ where $t \in S_c$ and $s \in S_a$ we have:

1. t L_{eq} $s \iff (t, s) \in L_{eq}$
2. there exists a function $\delta : (Acts_c \times S_c) \to ((Acts_a \times S_a) \to [0, 1])$ such that:
   a. for all $e_c \in Acts_c$, $t' \in S_c$ such that $P_c(t, e_c, t') > 0$, $\delta(e_c, t')$ is a distribution on $(Evts_a \times S_a)$
   b. $\forall\ s', e_a \in S_a \times Acts_a$ we have:

$$\sum_{t' \in S_c, e_c \in Acts_c} P_c(t, e_c, t').\delta(e_c, t')(e_a, s') = P_a(s, e_a, s')$$

   c. $\forall t', s', e_c, e_a \in S_c \times S_a \times Acts_c \times Acts_a$, if $\delta(e_c, t')(e_a, s') > 0$ then $(t', s') \in R_r$

From the condition 2.b of R_r we have:
$\forall s', e_a \in S_a \times Acts_a$:

$$\sum_{t' \in S_c, e_c \in Acts_c} P_c(t, e_c, t').\delta(e_c, t')(e_a, s') = P_a(s, e_a, s')$$

this formula can be rewritten as:
$\forall s', e_a \in S_a \times Acts_a$:

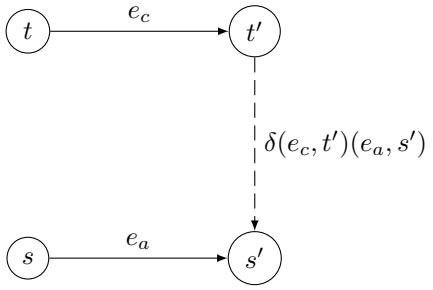
$$\sum_{t' \in \{t_1 | (t_1, s') \in L_{eq}\}, e_c \in \{e | e \in Acts_c \wedge (e, e_a) \in E_{eq}\}} P_c(t, e_c, t').\delta(e_c, t')(e_a, s') = P_a(s, e_a, s')$$

Many cases can be deduced from this condition:

Case1:

$$\begin{cases} card(\{t_1 | t_1 \in S_c \wedge (t_1, s') \in L_{eq}\}) = 1 \\ card(\{e | e \in Evts_c \wedge (e, e_a) \in E_{eq}\}) = 1 \end{cases}$$

$$\begin{cases} card(\{s' | s' \in S_a \wedge (s', t') \in L_{eq}\}) = 1 \\ card(\{e | e \in Evts_a \wedge (e, e_c) \in E_{eq}\}) = 1 \end{cases}$$



In this case, the action e_a is refined by the action e_c, i.e the corresponding event named e_a in M_a is refined by the event named e_c in M_c, the condition 2.b of the relation R_r can be rewritten as:

$$P_c(t, e_c, t').\delta(e_c, t')(e_a, s') = P_a(s, e_a, s')$$

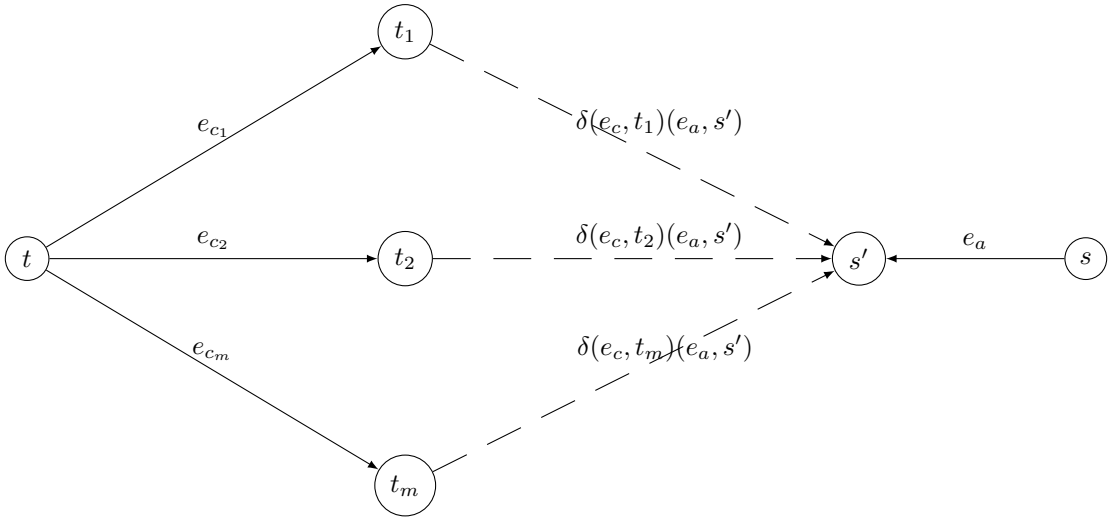
or from the condition 2.a of R_r, $\delta(e_c, t')$ is a distribution on $(Evts_a \times S_a)$, then $\delta(e_c, t')(e_a, s')= 1$, we deduce then that:

$$P_c(t, e_c, t') = P_a(s, e_a, s')$$

Case2:

$$\begin{cases} card(\{t_1 | t_1 \in S_c \wedge (t_1, s') \in L_{eq}\}) = m > 1 \\ card(\{e | e \in Evts_c \wedge (e, e_a) \in E_{eq}\}) = m > 1 \end{cases}$$

$$\begin{cases} card(\{s' | s' \in S_a \wedge (s', t') \in L_{eq}\}) = 1 \\ card(\{e | e \in Evts_a \wedge (e, e_c) \in E_{eq}\}) = 1 \end{cases}$$



In this case, the action e_a is refined by the actions e_{c_1}, e_{c_2},...,e_{c_m}, i.e, the corresponding event named e_a in M_a is splitted into the events named e_{c_1}, e_{c_2},...,e_{c_m} in M_c. The condition 2.b of R_r can be rewritten as:

$$\sum_{i=1}^{m} P_c(t, e_{c_i}, t_i).\delta(e_{c_i}, t_i)(e_a, s') = P_a(s, e_a, s')$$

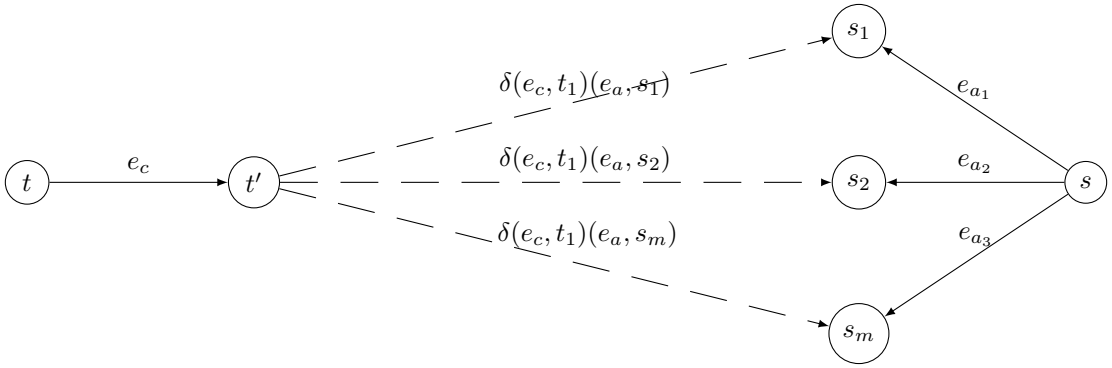
or from the condition 2.a of R_r, $\delta(e_{c_i}, t_i)$ $(1 \leq i \leq m)$ is a distribution on $(Evts_a \times S_a)$, then $\forall$ i, $\delta(e_{c_i}, t_i)(e_a, s')= 1$ and then:

$$\sum_{i=1}^{m} P_c(t, e_{c_i}, t_i) = P_a(s, e_a, s')$$

Case3:

$$\begin{cases} card(\{t_1 | t_1 \in S_c \wedge (t_1, s') \in L_{eq}\}) = 1 \\ card(\{e | e \in Evts_c \wedge (e, e_a) \in E_{eq}\}) = 1 \end{cases}$$

$$\begin{cases} card(\{s' | s' \in S_a \wedge (s', t') \in L_{eq}\}) = m > 1 \\ card(\{e | e \in Evts_a \wedge (e, e_c) \in E_{eq}\}) = m > 1 \end{cases}$$



In this case, the actions $e_{a_1}, e_{a_2}, ..., e_{a_m}$ are refined by the action e_c, i.e, the corresponding events named $e_{a_1}, e_{a_2}, ..., e_{a_m}$ in M_a are merged by the event named e_c in M_c. The condition 2.b of R_r can be rewritten as:

$$\forall i (1 \leq i \leq m), P_c(t, e_c, t').\delta(e_c, t')(e_{a_i}, s_i) = P_a(s, e_{a_i}, s_i)$$

$$\Longrightarrow$$

$$\sum_{i=1}^{m} P_c(t, e_c, t').\delta(e_c, t')(e_{a_i}, s_i) = \sum_{i=1}^{m} P_a(s, e_{a_i}, s_i)$$

$$\Longrightarrow$$

$$P_c(t, e_c, t').\sum_{i=1}^{m} \delta(e_c, t')(e_{a_i}, s_i) = \sum_{i=1}^{m} P_a(s, e_{a_i}, s_i)$$

or from the condition 2.a of R_r, we have $\delta(e_c, t')$ is a distribution on $Evts_a \times S_a$ ($\sum_{i=1}^{m} \delta(e_c, t')(e_{a_i}, s_i)$=1) then:

$$P_c(t, e_c, t') = \sum_{i=1}^{m} P_a(s, e_{a_i}, s_i)$$

Case4:

$$\begin{cases} card(\{t_1 | t_1 \in S_c \wedge (t_1, s') \in L_{eq}\}) = m > 1 \\ card(\{e | e \in Evts_c \wedge (e, e_a) \in E_{eq}\}) = m > 1 \end{cases}$$

$$\begin{cases} card(\{s' | s' \in S_a \wedge (s', t') \in L_{eq}\}) = m > 1 \\ card(\{e | e \in Evts_a \wedge (e, e_c) \in E_{eq}\}) = m > 1 \end{cases}$$

This case corresponds both to the case where some action e_{c_1} refine many actions $e_{a_1}, e_{a_2}, ..., e_{a_m}$ and one of the abstract action e_{a_1} for example refine many actions $e_{c_1}, e_{c_2}, ..., e_{c_m}$, we have then:

$$P_c(t, e_{c_i}, t_i) = \sum_{i=1}^{m} P_a(s, e_{a_i}, s_i)$$

and

$$\sum_{i=1}^{m} P_c(t, e_{c_i}, t_i) = P_a(s, e_{a_i}, s_i)$$

$\forall\ 1 \leq i \leq m$, $P_c(t, e_{c_i}, t_i) = P_a(s, e_{a_i}, s_i) = 0$ and we deduce then that it is impossible to obtain this case, an event cannot be splitted by many events and merged by one of these events at the same time. □

8 Conclusion and Future work

In this document, we have presented our proposition for introducing probabilities in Event-B. For now, we only propose a way to model purely probabilistic systems where all originally non-deterministic choices have been replaced with probabilistic choices in our new model. We also introduce a notion of refinement between two event-B models. In our proposition this notion of refinement mixes classical Event-B proof obligations with semantics-based new proof obligations. In the future, we plan on providing purely syntactical proof obligation in order to prove refinement between two Event-B models. Our last contribution is to show that this new notion of refinement between two Event-B models coincides with refinement of their Markov Chain semantics.

In this document, we have not defined a complete syntax for probabilistic Event-B models as well as the form of proof obligations specific to the introduced notion of refinement, future work will concetrate on overcoming these lacks. We present in the following some other perspectives that we would like to treat in the future:

- Providing the syntax and the semantics of Event-B models containing both probabilistic and standard events and derive the dedicated proof obligations.
- Providing the syntax and the semantics of Event-B models wehre the events are annotated by a probability range instead of a discrete probability and derive the proof obligations dedicated to refinement of these events.
- Providing a methodology for expressing and verifying some probabilistic properties expressed in several probabilistic logics(PCTL, PLTL, PCTL*, PLTL*).
- Providing a solution for the composition/Decomposition of probabilistic Event-B models.
- Providing some design patterns in Event-B for modeling probabilistic systems.
- Developping a dedicated Plugin in Rodin of the probabilistic version of Event-B.

References

[1] Martín Abadi and Leslie Lamport. The existence of refinement mappings. *Theoretical Computer Science*, 82(2):253–284, 1991.

[2] Jean-Raymond Abrial. *Modeling in Event-B: system and software engineering*. Cambridge University Press, 2010.

[3] Jean-Raymond Abrial and Jean-Raymond Abrial. *The B-Book: Assigning programs to meanings*. Cambridge University Press, 2005.

[4] Jean-Raymond Abrial, Michael Butler, Stefan Hallerstede, Thai Son Hoang, Farhad Mehta, and Laurent Voisin. Rodin: an open toolset for modelling and reasoning in event-b. *International journal on software tools for technology transfer*, 12(6):447–466, 2010.

[5] Jean-Raymond Abrial, Dominique Cansell, and Dominique Méry. A mechanically proved and incremental development of ieee 1394 tree identify protocol. *Formal aspects of computing*, 14(3):215–227, 2003.

[6] Jean-Raymond Abrial and Stefan Hallerstede. Refinement, decomposition, and instantiation of discrete models: Application to event-b. *Fundamenta Informaticae*, 77(1):1–28, 2007.

[7] Suzana Andova and Jos CM Baeten. Abstraction in probabilistic process algebra. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 204–219. Springer, 2001.

[8] Ralph-JR Back. Refinement calculus, part ii: Parallel and reactive programs. In *Stepwise Refinement of Distributed Systems Models, Formalisms, Correctness*, pages 67–93. Springer, 1990.

[9] Ralph JR Back and Joakim von Wright. Refinement calculus, part i: Sequential nondeterministic programs. In *Stepwise refinement of distributed systems models, formalisms, correctness*, pages 42–66. Springer, 1990.

[10] RJR Back and Kaisa Sere. Stepwise refinement of action systems. In *Mathematics of Program Construction*, pages 115–138. Springer, 1989.

[11] Christel Baier and Holger Hermanns. Weak bisimulation for fully probabilistic processes. In *Computer Aided Verification*, pages 119–130. Springer, 1997.

[12] Christel Baier, Joost-Pieter Katoen, et al. *Principles of model checking*, volume 26202649. MIT press Cambridge, 2008.

[13] Eerke Boiten, John Derrick, and Gerhard Schellhorn. Relational concurrent refinement part ii: Internal operations and outputs. *Formal Aspects of Computing*, 21(1-2):65–102, 2009.

[14] Willem-Paul De Roever, Kai Engelhardt, and Karl-Heinz Buth. *Data refinement: model-oriented proof methods and their comparison*, volume 47. Cambridge University Press, 1998.

[15] Stefan Hallerstede and Thai Son Hoang. Qualitative probabilistic modelling in event-b. In *Integrated Formal Methods*, pages 293–312. Springer, 2007.

[16] Jifeng He, CAR Hoare, and Jeff W Sanders. Data refinement refined resume. In *ESOP 86*, pages 187–196. Springer, 1986.

[17] Jifeng He, CAR Hoare, and Jeff W Sanders. Data refinement refined resume. In *ESOP 86*, pages 187–196. Springer, 1986.

[18] Monika Rauch Henzinger, Thomas A Henzinger, and Peter W Kopke. Computing simulations on finite and infinite graphs. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pages 453–462. IEEE, 1995.

[19] Thai Son Hoang. *The development of a probabilistic B-method and a supporting toolkit*. PhD thesis, The University of New South Wales, 2005.

[20] Thai Son Hoang. *The development of a probabilistic B-method and a supporting toolkit*. PhD thesis, The University of New South Wales, 2005.

[21] Thai Son Hoang, Zhendong Jin, Ken Robinson, Annabelle McIver, and Carroll Morgan. Probabilistic invariants for probabilistic machines. In *ZB 2003: Formal Specification and Development in Z and B*, pages 240–259. Springer, 2003.

[22] Bengt Jonsson. Simulations between specifications of distributed systems. In *CONCUR'91*, pages 346–360. Springer, 1991.

[23] Bengt Jonsson and Kim Guldstrand 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.

[24] Chi-Chang Jou and Scott A Smolka. Equivalences, congruences, and complete axiomatizations for probabilistic processes. In *CONCUR'90 Theories of Concurrency: Unification and Extension*, pages 367–383. Springer, 1990.

[25] Kim G Larsen and Arne Skou. Bisimulation through probabilistic testing (preliminary report). In *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 344–352. ACM, 1989.

[26] Nancy Lynch and Frits Vaandrager. Forward and backward simulations. *Information and Computation*, 121(2):214–233, 1995.

[27] Annabelle McIver and Charles Carroll Morgan. *Abstraction, refinement and proof for probabilistic systems*. Springer Science & Business Media, 2006.

[28] Robin Milner. *Communication and concurrency*, volume 84. Prentice hall New York etc., 1989.

[29] Robin Milner, Robin Milner, Robin Milner, and Robin Milner. *A calculus of communicating systems*, volume 92. springer-Verlag Berlin, 1980.

[30] Carroll Morgan, Thai Son Hoang, and Jean-Raymond Abrial. The challenge of probabilistic event b—extended abstract—. In *ZB 2005: Formal Specification and Development in Z and B*, pages 162–171. Springer, 2005.

[31] David Park. *Concurrency and automata on infinite sequences*. Springer, 1981.

[32] Anna Philippou, Insup Lee, and Oleg Sokolsky. Weak bisimulation for probabilistic systems. In *CONCUR 2000—Concurrency Theory*, pages 334–349. Springer, 2000.

[33] Luis Ferreira Pires and Wanderley Lopes de Souza. Step-wise refinement design example using lotos. In *FORTE*, volume 90, pages 255–262, 1990.

[34] Gerhard Schellhorn. Asm refinement and generalizations of forward simulation in data refinement: a comparison. *Theoretical Computer Science*, 336(2):403–435, 2005.

[35] Roberto Segala and Nancy Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995.

[36] Anton Tarasyuk, Elena Troubitsyna, and Linas Laibinis. Reliability assessment in event-b development. *NODES 09*, page 11, 2009.

[37] Anton Tarasyuk, Elena Troubitsyna, and Linas Laibinis. Towards probabilistic modelling in event-b. In *Proceedings of the 8th International Conference on Integrated Formal Methods*, IFM'10, pages 275–289, Berlin, Heidelberg, 2010. Springer-Verlag.

[38] Anton Tarasyuk, Elena Troubitsyna, and Linas Laibinis. Integrating stochastic reasoning into event-b development. *Formal Aspects of Computing*, 27(1):53–77, 2015.

[39] Niklaus Wirth. Program development by stepwise refinement. *Communications of the ACM*, 14(4):221–227, 1971.

[40] Jim Woodcock and Jim Davies. *Using Z: Specification, Refinement, and Proof*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

[41] Emre Yilmaz. *Tool support for qualitative reasoning in Event-B*. PhD thesis, Master Thesis ETH Zürich, 2010, 2010.