Reliability in Fully Probabilistic Event-B: How to Bound the Enabling of Events

Syrine Aouadi, Arnaud Lanoix

Reliability in Fully Probabilistic Event-B: How to Bound the Enabling of Events

Syrine Aouadi[2] and Arnaud Lanoix[1]

[1] University of Nantes / LS2N UMR CNRS 6004
arnaud.lanoix@univ-nantes.fr
[2] syrine.aouadi@eleves.ec-nantes.fr

Abstract. In previous work, we have proposed a fully probabilistic version of Event-B where all the non-deterministic choices are replaced by probabilistic ones and, particularly, the events are equipped with weights that allow us to consider their enabling probability. In this work, we focus on the reliability of the system by proposing to constraint the probability of enabling an event (or a set of events) to control its importance with regard to the intended system behaviour. We add a specific upper bound which must limit the enabling probabilities of the chosen events and we consider the necessary proof obligations to check that the considered events respect the bound. At the end, we illustrate our work by presenting a case study specified in probabilistic Event-B and where bounding the enabling of some events is mandatory.
Keywords: Event-B, Probabilistic Event-B, Probabilistic properties, Reliability, Weight, Proof obligations

1 Introduction

Systems using randomized algorithms [1], probabilistic protocols [2] or failing components become more and more complex. It is then necessary to add new modeling features in order to take into account the inherent complexity of the system properties such as reliability [3], responsiveness [4,5], continuous evolution, energy consumption etc. One of these features is probabilistic reasoning, which can be used in order to introduce uncertainty in a model or to mimic randomized behavior. Probabilistic modeling formalisms have therefore been developed in the past, mainly extending automata-based formalisms [6,7]. Abstraction [8,9], refinement [10] and model-checking algorithms [11,12] have been successfully studied in this context. However, the introduction of probabilistic reasoning in proof-based modeling formalisms has been, to the best of our knowledge, quite limited [13,14,15,16,17,18,19,20]. Although translations from proof-based models to automata-based models are always possible, the use of automata-based verification techniques in the context of proof-based models is most of the time inconvenient because of possible state-space explosion introduced in the translation.

Event-B [21] is a proof-based formal method used for modeling discrete systems. It is equipped with *Rodin* [22], an open toolset for modeling and proving systems. This toolset can easily be extended, which makes Event-B a good candidate for introducing probabilistic reasoning in a proof-based modeling formalism.

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 [23], Abrial *et al.* have summarized 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 [24], 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 [25], the same authors study the refinement of qualitative probabilistic Event-B models and propose tool support inside Rodin. Other works [26,27,28] have extended this approach by refining non-deterministic assignments into *quantitative* probabilistic assignments where, unlike in [24], 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, sources of non-determinism other than assignments have been left untouched, although the authors argue that probabilistic choice between events or parameter values can be achieved by transformations of the models that embed these choices inside probabilistic assignments. While this is unarguably true, such transformations are not trivial and greatly impede the understanding of Event-B models. Moreover, these transformations would need to be included in the refinement chain when designers need it, which would certainly be counter-intuitive to engineers.

In previous works [29,30] we pursued these works by proposing a probabilistic extension of Event-B and presenting some ways of introducing probabilistic reasoning within Event-B. We have proposed some new syntactic elements for writing fully probabilistic Event-B models in the Event-B framework. The consistency of such models has been expressed, as in standard Event-B, in terms of proof obligations. In the standard Event-B setting, *convergence* is a required property for proving a refinement step as soon as new events are introduced in the model. The counterpart property in the probabilistic setting is *almost-certain convergence*, which has already been studied in [24], in the context of non-deterministic models with only probabilistic assignments. We therefore have exhibited 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 [24] concerning events and parameters, they are also less restrictive concerning probabilistic assignments. Finally, some of the previously mentioned results have been implemented in a prototype plugin for Rodin.

In the probabilistic Event-B context, the probabilistic events are equipped with weights, in order to easily consider the enabling probability of each event in any configuration. In this paper, we extend the previous work by considering a kind of probabilistic invariant that allows us to model and verify properties concerning reliability: we

add a specific upper bound to constraint the enabled probability of an event (or a set of events) in order to limit their importance with regards to the system behavior. As in the previous work, we check that the considered events respect the bound by means of new proof obligations. These new proof obligations ensure that in any configurations where the considered event can be enabled, this enabled probability is lower than the considered bound. Finally, we illustrate our work on an industrial simplified case study: the PCB manufacturing and control system [31,32]. Particularly, we show the requirements for applying the enabled bound property on a realistic used system.

Outline. The paper is structured as follows. Section 2 presents the scientific background of this paper in terms of Probabilistic Event-B. In Section 3, we focus on how to constraint the enabling of events, and the necessary proof obligations on such models. Section 4 introduces our case study: a simple PCB manufacturing and control system which illustrate the use of the previously mentioned property. Finally, Section 5 concludes the paper.

2 Preliminaries: Probabilistic Event-B

Event-B [21] is a formal method used for the development of complex discrete systems. Systems are described in Event-B by means of models. We have previously extended standard Event-B models to introduce probabilistic reasoning. 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 have proposed to replace all these non-deterministic choices with probabilistic ones.

MODEL
M
VARIABLES
$\bar{v}$
INVARIANTS
$I(\bar{v})$
VARIANT
$V(\bar{v})$
EVENTS
Init $\widehat{=} \ldots$
$e_1 \widehat{=} \ldots$
$\ldots$
$e_n \widehat{=} \ldots$
END

For the sake of simplicity, we assume in the rest of the paper that a fully probabilistic Event-B model is expressed by a tuple $M=(\bar{v}, I(\bar{v}), V(\bar{v}), PEvts, Init)$ where $\bar{v}=\{v_1 \ldots v_n\}$ is a set of variables, $I(\bar{v})$ is an invariant, $V(\bar{v})$ is an (optional) variant used for proving the (almost-certain) convergence of the model, PEvts is a set of probabilistic events and $Init \in PEvts$ is the initialization event. The invariant $I(\bar{v})$ is a conjunction of predicates over the variables of the system specifying properties that must always hold.

Probabilistic events A probabilistic event has the following structure where e_i is the name of the event, $W_i(\bar{v})$ is the weight 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. A probabilistic event is *enabled* in a given valuation of the variables (also called a configuration) if and only if *i)* there exists a parameter valuation such that its guard $G_i(\bar{t}, \bar{v})$ is fulfilled in this context and *ii)* its weight $W_i(\bar{v})$ is strictly positive.

In standard Event-B, when several events are enabled, the event to be executed is chosen non-deterministically. The weight $W_i(\bar{v})$ of the event resolves this non-deterministic choice: in configurations where several probabilistic events are enabled, the probability of enabling 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. Moreover, for the sake of expressibility, we propose to express the weight $W_i(\bar{v})$ of a probabilistic event e_i as an expression over the variables $\bar{v}$ of the fully probabilistic Event-B model. The probability of enabling a given event can therefore evolve as the system progresses.

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

For the events equipped with parameters $\bar{t}$, a valuation of the parameters is chosen such that the guard $G_i(\bar{t},\bar{v})$ of the event is satisfied. In standard Event-B, when there are several such parameter valuations, one of them is selected non-deterministically. We therefore have proposed to replace this non-deterministic choice by a uniform choice over all parameter valuations ensuring that the guard of the event is satisfied.

Probabilistic assignments The action $SP_i(\bar{t},\bar{v})$ of a probabilistic event may contain several assignments that are executed in parallel. An assignment can be expressed in one of the following forms:

- **Deterministic assignment**: $x := E(\bar{t},\bar{v})$ means that the expression $E(\bar{t},\bar{v})$ is assigned to the variable x.
- **Predicate probabilistic assignment**: $x :\oplus Q(\bar{t},\bar{v},x,x')$ means that the variable x is assigned a new value x' such that the predicate $Q(\bar{t},\bar{v},x,x')$ is satisfied. Instead of choosing non-deterministically among the values of x' such that the predicate $Q(\bar{t},\bar{v},x,x')$ is true as in standard predicate non-deterministic assignments, we propose to choose this new value using an uniform distribution.
- **Enumerated probabilistic assignment**: $x := E_1(\bar{t},\bar{v}) \mathbin{@} p_1 \oplus \ldots \oplus E_n(\bar{t},\bar{v}) \mathbin{@} p_n$ means that 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 an adequate context is present. 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 [33].

Before-after predicate and semantics The formal semantics of an assignment is described by means of a before-after predicate (BA) $Q(\bar{t},\bar{v},x,x')$, which describes the relationship between the values of the variable before (x) and after (x') the execution of an assignment.

- The BA of a deterministic assignment is $x' = E(\bar{t},\bar{v})$.
- The BA of a predicate probabilistic assignment is $Q(\bar{t},\bar{v},x,x')$.
- The BA of an enumerated probabilistic assignment is $x' \in \{E_1(\bar{t},\bar{v}) \ldots E_n(\bar{t},\bar{v})\}$.

Recall that the action $S_i(\bar{t},\bar{v})$ of a given event e_i may contain several assignments that are executed in parallel. Assume that $v_1 \ldots v_i$ are the variables assigned in $S_i(\bar{t},\bar{v})$

– 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 $S_i(\bar{t},\bar{v},\bar{v}')$ of the event action $S_i(\bar{t},\bar{v})$ is:

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

Proof obligations The consistency of a standard Event-B model is characterized by means of *proof obligations* (POs) formally defined in [21] which must be discharged. Discharging all the necessary POs allows to prove that the model is sound with respect to some underlaying behavioral semantics. The consistency of a fully probabilistic Event-B model is also characterized by means of POs to be discharged. Among all of them, the first ones are adaptation of the standard POs and the second ones are POs specific to fully probabilistic Event-B.

In the following, we recall the adaptation of the most important of the standard POs: (event/pINV) for *invariant preservation*, which states that the invariant still holds after the execution of each probabilistic event in the Event-B model M. Given an event e_i with a guard $G_i(\bar{t},\bar{v})$ and an action $S_i(\bar{t},\bar{v})$, this PO is expressed as follows:

$$I(\bar{v}) \;\wedge G_i(\bar{t},\bar{v}) \;\wedge W_i(\bar{v}) > 0 \wedge SP_i(\bar{t},\bar{v},\bar{v}') \vdash I(\bar{v}') \qquad \text{(event/INV)}$$

Then, we give some of the new POs specific to fully probabilistic Event-B:

– we impose that the expression $W_i(\bar{v})$ representing the weight of a given probabilistic event must evaluate to natural numbers.

$$I(\bar{v}) \;\wedge G_i(\bar{t},\bar{v}) \;\vdash\; W_i(\bar{v}) \in \mathsf{NAT} \qquad \text{(event/WGHT/NAT)}$$

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

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

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

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

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

$$\vdash\; p_1 + \ldots + p_n = 1 \qquad \text{(event/assign/pWD2)}$$

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

– In order to define a discrete uniform distribution over the set of values of a variable x making the predicate $Q_x(\bar{t},\bar{v},x')$ of the corresponding assignment satisfied, we impose that this set must be finite.

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

Feasibility of predicate probabilistic assignments is ensured by the standard feasibility PO [21] inherited from Event-B. It ensures that the set $\{x' \mid Q_x(\bar{t},\bar{v},x')\}$ is not empty.

3 Contribution: Limiting the Enabling of Probabilistic Events

We recall that, 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 have proposed to equip each event with a *weight* $\mathsf{W}_i(\bar{v})$ that is an expression over the variables $\bar{v}$ of the fully probabilistic Event-B model. Note that using weights instead of actual probability values is convenient as the set of enabled events evolves with the configurations of the system. Using probability values instead would require to normalize them in all configurations.

Enabling probability A probabilistic event is enabled in a given configuration if and only if there exists a parameter valuation such that its guard $\mathsf{G}_i(\bar{t},\bar{v})$ is fulfilled and its weight $\mathsf{W}_i(\bar{v})$ is strictly positive. 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 configuration. Formally, this *enabling probability* is defined as follows.

Let M=($\bar{v}$,$\mathsf{I}(\bar{v})$,$\mathsf{V}(\bar{v})$,PEvts,Init) be a fully probabilistic Event-B model. Given an event e_i in PEvts and a valuation σ of the variables $\bar{v}$ of the model, the enabling probability of e_i in σ is formally defined by

$$P(e_i,\sigma) = \frac{[\sigma]W_i(\bar{v})}{\sum_{e_j \in PEvts}\ ([\sigma]W_j(\bar{v} \mid [\sigma]W_j(\bar{v}) > 0 \wedge \exists\ \theta'.[\sigma,\theta']G_j(\bar{v},\bar{t}) = \top)} \tag{1}$$

$$\text{if } \exists\ \theta.[\sigma,\theta]G_i(\bar{v},\bar{t}) = \top \text{ and } [\sigma]W_i(\bar{v}) > 0 \tag{2}$$

$$= 0 \text{ otherwise} \tag{3}$$

where θ and θ' are possible valuations of the parameters $\bar{t}$.

The probability of enabling a given event can therefore evolve as the system progresses. Equation (1) represents the ratio of the weight of the considered event e_i against the total value of the weights of all the enabled events (including the weight of e_i), when Equation (2) is verified i.e. the event e_i is enabled. Otherwise as Equation (3) the enabling probability of e_i is equal to 0.

Enabled bound property In standard and probabilistic Event-B, the events for which we want to study their termination are annotated as **convergent**. We adopt the same principle and we annotate by **bounded** the events for which we want to limit their enabling probabilities. We also introduce a specific upper bound $\mathsf{EB}(\bar{v})$ (notice **ENABLED_BOUND** into the B model) as an expression over the variables $\bar{v}$ of the fully probabilistic Event-B model to limit the enabling probability of the **bounded** events. Note that this upper bound can evolve as the system progresses.

Considering a **bounded** event e_i, it must verify the *enabled bound property*, i.e. in all configurations in which e_i could be enabled, then its enabling probability must be lower than or equal to the value of the enabled upper bound $\mathsf{EB}(\bar{v})$ in that configuration. Formally

$$\forall\sigma.\ \exists\ \theta.[\sigma,\theta]G_i(\bar{v},\bar{t}) = \top\ \wedge [\sigma]W_i(\bar{v}) > 0 \Rightarrow\ P(e_i,\sigma) \leq [\sigma]EB(\bar{v}) \tag{4}$$

Proof obligations Checking standard or probabilistic Event-B models consists of discharging proof obligations. We then propose necessary POs to check the above mentioned enabled bound property on a fully probabilistic Event-B model.

Let M=($\bar{v}$,I($\bar{v}$),V($\bar{v}$),PEvts,Init) be a fully probabilistic Event-B model. Let e_i be a **bounded** event from PEvts= $\{\mathsf{e}_1 \ldots \mathsf{e}_i \ldots \mathsf{e}_n\}$. Let EB($\bar{v}$) be the enabled upper bound. Then, the necessary POs are defined as follows.

1. The enabled upper bound EB($\bar{v}$) must always be a rational number i.e. a positive non-zero value strictly lower than 1:

$$\mathsf{I}(\bar{v}) \vdash 0 < \mathsf{EB}(\bar{v}) < 1 \qquad \text{(eBOUND/WD)}$$

2. Each **bounded** event e_i satisfies the enabled bound property (see Equation (4)), i.e. its enabling probability is always lower than or equal to the enabled upper bound.

$$\mathsf{I}(\bar{v}) \wedge \mathsf{G}_i(\bar{t},\bar{v}) \wedge \mathsf{W}_i(\bar{v}) > 0 \vdash \frac{\mathsf{W}_i(\bar{v})}{\sum(\mathsf{e}_j).(\mathsf{W}_j(\bar{v}) \mid \mathsf{G}_j(\bar{t},\bar{v}) \wedge \mathsf{W}_j(\bar{v}) > 0)} \leq \mathsf{EB}(\bar{v}) \qquad \text{(event/WGHT/eBOUND)}$$

4 Case Study: the PCB Manufacturing and Control System

In this section, our purpose is to highlight on a concrete case study the interest of the enabled bound property. We then propose a fully probabilistic Event-B model of a simplified industrial case study: the Printed Circuit Boards (PCB) manufacturing and control system [31,32]. This case study interests electronic cards manufacturers that face the ever increasing requirement of reducing their cost and improving products quality. That requires having a fine control strategy through which we evaluate the produced electronic cards by detecting possible errors from the tests performed.

In the considered case study, we will focus on two kinds of tests: the ICT ("In Circuit Testing") tests check the presence of all the attempted components and the FCT ("Functional Testing") tests verify the functional behavior of each PCB.

Our proposed model simply abstracts the manufacturing and control process. We only identify each manufactured card by a unique identifier and we introduce two events, one representing the fair cards manufacturing, the other modeling the deficient cards manufacturing. The PCB manufacturing and control system must provide a history about the produced PCB and the error reporting.

Informal description of this system imposes two probabilistic requirements

(i) the risk of having a deficient card must decrease with the increasing number of reported errors;
(ii) having fair cards increases with correct cards production rise.

In fact, the manufacturing and control system must be a self-corrective maneuver on the PCB production line.

Event-B Context To model the static aspects of the system, we propose the Event-B context as depicted by Figure 1. Precisely,

CONTEXT
PCBctx
SETS
Error_State
CONSTANTS
Max_Cards
Max_Errors
ICT_Error
FCT_Error
ICT_FCT_Error
AXIOMS
Max_Cards $\in \mathbb{N}_1$
$\wedge$ Max_Errors $\in \mathbb{N}_1$
$\wedge$ partition (Error_State,{ICT_Error},{FCT_Error},{ICT_FCT_Error})
$\wedge$ Max_Errors $\leq$ Max_Cards
END

Fig. 1. PCB manufacturing system context

- the constant Max_Cards models the maximum number of cards that can be produced whereas the constant Max_Errors models the maximum number of errors that can be reported;
- the set Error_State represents the tree kind of errors that can be reported during the test phase:
  - the constant ICT_error designs an "In Circuit" error;
  - the constant FCT_error Designs a "Functional" error;
  - the constant ICT_FCT_error designs a double error, i.e. "In Circuit" and "Functional" errors simultaneously.

Note that Error_State is syntactically expressed as a partition between ICT_error, FCT_error and ICT_FCT_error, i.e. Error_State = {ICT_error,FCT_error,ICT_FCT_error}.

The maximum number of reported errors must be lower than or equal to the maximum number of produced cards. Only one kind of error is reported for a specific card.

Event-B model We propose to model the system's state by means of three state variables, as depicted by Figure 2:

- the set Cards represents all the produced cards;
- the partial function Errors models the history of all the cards which have reporting errors, i.e. it associates to each necessary card, the corresponding reported error;
- the variable Next_ID identifies the nextly produced card.

We then model the dynamic of the system using three (probabilistic) events.

- the event init initializes the model: regarding Cards and Errors, they are initialized to empty sets and Next_ID is initialized to any chosen integer value (10 on the illustrated specification);

MODEL
PCBsystem
SEES
PCBctx
VARIABLES
Cards
Errors
Next_ID
INVARIANTS
$\text{Cards} \subseteq \mathbb{N}_1$
$\wedge$ Errors $\in$ Cards$\nrightarrow$Error_State
$\wedge$ Next_ID $\in \mathbb{N}_1$
$\wedge$ finite (Cards)
$\wedge$ finite (Errors)
$\wedge$ card(Cards) $\leq$ Max_Cards
$\wedge$ card(Errors) $\leq$ Max_Errors
ENABLED_BOUND
(Max_Cards+card(Cards)+1) / (Max_Cards+card(Cards)+Max_Errors$-$card(Errors)+2)
EVENTS
event init $\widehat{=}$
begin
Next_ID:$=$ 10
Cards $:= \varnothing$
Error $:= \varnothing$
end
event Manufacturing_OK $\widehat{=}$
weight
Max_Cards + 1 + card(Cards)
when
card(Cards) $<$ Max_Cards $\wedge$ card(Errors) $\leq$ Max_Errors
then
Cards $:=$ Cards $\cup \{$Next_ID$\}$
Next_ID $:=$ Next_ID + 1
end
event Manufacturing_Error $\widehat{=}$
bounded
weight
Max_Errors + 1 $-$ card(Errors)
any error **where**
error $\in$ Error_State $\wedge$ card(Cards) $<$ Max_Cards $\wedge$ card(Errors) $<$ Max_Errors
then
Cards $:=$ Cards $\cup \{$Next_ID$\}$
Next_ID $:=$ Next_ID + 1
Errors $:=$ Errors $\cup \{$Next_ID $\mapsto$error$\}$
end
END

Fig. 2. PCB manufacturing system model

– the event Manufacturing_OK models the fair cards production. Cards could be produced when the maximum number of produced cards is not reached and the maximum number of errors is also not reached; The number of the newly produced card is added to Cards, and the Next_ID is incremented;
– the event Manufacturing_Error represents the production of deficient cards: The event's parameter error chooses a kind of errors among the Error_State, i.e. a "In Circuit" error, a "Functional" error or the both simultaneously. The newly produced card is also registered, the Next_ID is incremented and the reported error is added to Errors : Next_ID $\mapsto$ error.

We note that the system stops running when the allowed numbers of reported errors or total cards produced are reached.

Probabilities appear in weights associated to each events and in the uniform choice between the kind of errors in the event Manufacturing_Error. We recall that informal description of this system imposes two probabilistic requirements: the risk of having a deficient card must decrease with the increasing number of reported errors and having fair cards increases with cards production rising, due to the the manufacturing and control system must be a self-corrective maneuver on the PCB production line. In other words, the more errors are reported, the less errors will be reported, whereas the more cards are produced, the more fair care will be produced. As the events Manufacturing_OK and Manufacturing_Error will be enabled simultaneously, their respective probabilities computed from their weights translate the requirements:

– the weight of the event Manufacturing_OK increases with the number of produced cards, that corresponds to the requirement "the more cards are produced, the more fair care will be produced";
– the weight of the event Manufacturing_Error decreases with the number of reported errors, that correspond to the requirement "the more errors are reported, the less errors will be reported".

To illustrate the attempted behavior of the specified system, we give in Figure 4 a sub-part of the corresponding probabilistic transition system, with Max_Errors fixed to 2 and Max_Cards fixed to 3.

Verification We consider that the consistency of the Event-B model PCBsystem presented above is verified by discharging all the necessary consistency proof obligations. We only focus on the verification of the *enabled bound property* depicted in Section 3. we annotate by **bounded** the event Manufacturing_Error and we add an **ENABLED_BOUND**: the enabling probability of Manufacturing_Error must be always limited by the value of the **ENABLED_BOUND**. We have chosen as **ENABLED_BOUND** an expression which corresponds to the enabling probability of the event Manufacturing_OK to ensure that always errors are reported less than fair cards are produced; it is a specific case: in a more general case, any expression could be chosen with respect to the case study.

To prove that the enabled bound property is verified, we must discharge the POs (eBOUND/WD) and (event/WGHT/eBOUND). The PO (eBOUND/WD) is instantiated as follows on the Event-B model PCBsystem:

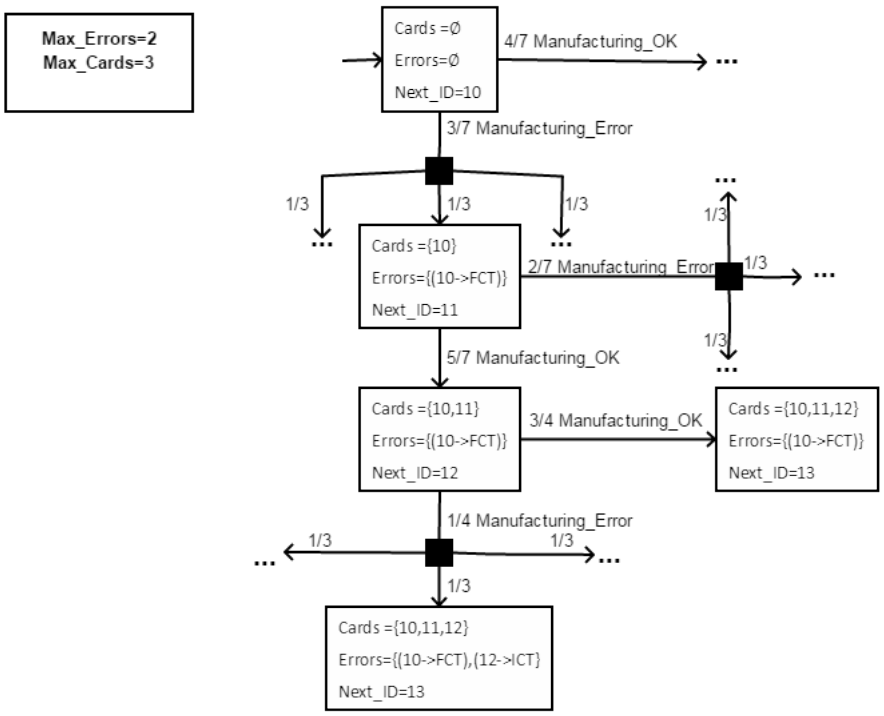

Fig. 3. Extract of the transition system of PCBsystem with Max_Errors=2 and Max_Cards=3

```
1  Max_Cards ∈ ℕ1
2  Max_Errors ∈ℕ1
3  Max_Errors  ≤ Max_Cards
4  Cards ⊆ ℕ1
5  Errors  ∈ Cards⇸Error_State
6   finite (Cards)
7   finite (Errors)
8  card(Cards) ≤ Max_Cards
9  card(Errors)  ≤ Max_Errors
10 |− − − − − − − − − − − − − − − − − − − − −
11 0 <
12 ( Max_Cards + card(Cards) +1 )
13 / ( Max_Cards + card(Cards) + Max_Errors − card(Errors) +2 )
14 < 1
```

We have to show that the goal (lines 11-14) could be established using the hypotheses (lines 1-9). It is obvious since the sum given line 12 is strictly positive when taking account of the hypotheses lines 1 and 8. The hypotheses lines 5 and 9 imply that the difference (Max_Errors + 1 − Max_Cards) is also strictly positive. Thus, numerator given

line 12 is strictly lower than denominator given line 13. So, the considered fraction is strictly lower than 1 and the PO (eBOUND/WD) is discharged.

Secondly, we instantiate the PO (event/WGHT/eBOUND) in the context of the **bounded** event Manufacturing_Error. Note that the event Manufacturing_Ok could always be triggered with Manufacturing_Error so the PO becomes as follows:

```
1  Max_Cards ∈ ℕ1
2  Max_Errors ∈ℕ1
3  Max_Errors ≤ Max_Cards
4  Cards ⊆ ℕ1
5  Errors ∈ Cards⇸Error_State
6  Next_ID ∈ ℕ1
7   finite (Cards)
8   finite (Errors)
9  card(Cards) < Max_Cards
10 card(Errors) < Max_Errors
11 |− − − − − − − − − − − − − − − − − − − − −
12 ( Max_Errors − card(Errors) +1)
13 / ( Max_Cards + card(Cards) + Max_Errors − card(Errors) +2 )
14 <
15 ( Max_Cards + card(Cards) +1 )
16 / ( Max_Cards + card(Cards) + Max_Errors − card(Errors) +2 )
```

Clearly, when we take account the hypothesis given line 3, the goal is obviously discharged.

In this example, we showed how demonstrate the necessary POs by hand, but in an industrial context, the Rodin toolset will be used and their embedded automatic provers will be in charge of discharging the POs. Discharging the POs (eBOUND/WD) and (event/WGHT/eBOUND) ensures that the enabled bound property is proved on the PCBsystem, i.e. always errors on cards are less reported than fair cards are produced.

5 Conclusion

Some properties as invariance, deadlock-freeness or convergence are natively managed in Event-B. In our probabilistic extension of Event-B, we have studied the almost certain convergence of a set of events. Moreover, a variety of research works treated the expression and verification of other probabilistic properties such as reliability or reactivity. In this paper we pursue our investigation of probabilistic properties and how to verify them using proof-based techniques. We proposed to express and check an enabled bound property where an event's probability is bounded by a fixed limit described during the requirements specification phase. This property can be used in a wide class of industrial systems, especially those where errors execution have a limit that must not be crossed. Hence, we illustrated a simplified use case of control and manufacturing of printed circuit boards where the enabled bound property was imperative to check if the likelihood of manufacturing an erroneous card can be at most equal to that of producing a correct card.

References

1. Motwani, R., Raghavan, P.: Randomized algorithms. Chapman & Hall/CRC (2010)
2. Abrial, J.R., Cansell, D., Méry, D.: A mechanically proved and incremental development of IEEE 1394 tree identify protocol. Formal aspects of computing **14**(3) (2003) 215–227
3. Villemeur, A.: Reliability, Availability, Maintainability and Safety Assessment: Assessment, Hardware, Software and Human Factors. Volume 2. Wiley (1992)
4. Chu, W.W., Sit, C.M.: Estimating task response time with contentions for real-time distributed systems. In: Real-Time Systems Symposium, 1988., Proceedings., IEEE (1988) 272–281
5. Trivedi, K.S., Ramani, S., Fricks, R.: Recent advances in modeling response-time distributions in real-time systems. Proceedings of the IEEE **91**(7) (2003) 1023–1037
6. Stoelinga, M.: An introduction to probabilistic automata. Bulletin of the EATCS **78**(176-198) (2002)
7. Puterman, M.L.: Markov decision processes: discrete stochastic dynamic programming. John Wiley & Sons (2014)
8. Katoen, J.P.: Abstraction of probabilistic systems. In: Formal Modeling and Analysis of Timed Systems: 5th International Conference, FORMATS 2007, Salzburg, Austria, October 3-5, 2007. Proceedings, Springer (2007)
9. Dehnert, C., Gebler, D., Volpato, M., Jansen, D.N.: On abstraction of probabilistic systems. In Remke, A., Stoelinga, M., eds.: Stochastic Model Checking. Rigorous Dependability Analysis Using Model Checking Techniques for Stochastic Systems: International Autumn School, ROCKS 2012, Vahrn, Italy, October 22-26, 2012, Advanced Lectures, Springer (2014)
10. Jonsson, B., Larsen, K.G.: Specification and refinement of probabilistic processes. In: Logic in Computer Science, 1991. LICS'91., IEEE (1991) 266–277
11. Bianco, A., De Alfaro, L.: Model checking of probabilistic and nondeterministic systems. In: International Conference on Foundations of Software Technology and Theoretical Computer Science, Springer (1995) 499–513
12. Baier, C., Katoen, J.P., et al.: Principles of model checking. MIT press Cambridge (2008)
13. Haghighi, H., Afshar, M.: A Z-Based formalism to specify markov chains. Computer Science and Engineering **2**(3) (2012) 24–31
14. Sere, K., Troubitsyna, E.: Probabilities in action systems. In: Proc. of the 8th Nordic Workshop on Programming Theory. (1996) 373–387
15. Hoang, T.S.: The development of a probabilistic B-method and a supporting toolkit. PhD thesis, The University of New South Wales (2005)
16. Goldreich, O.: Probabilistic proof systems. In: Modern Cryptography, Probabilistic Proofs and Pseudorandomness. Springer (1999) 39–72
17. Barthe, G., Fournet, C., Grégoire, B., Strub, P.Y., Swamy, N., Zanella-Béguelin, S.: Probabilistic relational verification for cryptographic implementations. In: ACM SIGPLAN Notices. Volume 49., ACM (2014) 193–205
18. Hurd, J., McIver, A., Morgan, C.: Probabilistic guarded commands mechanized in HOL. Electronic Notes in Theoretical Computer Science **112** (2005) 95–111
19. Audebaud, P., Paulin-Mohring, C.: Proofs of randomized algorithms in Coq. Science of Computer Programming **74**(8) (2009) 568–589
20. Hurd, J.: Formal verification of probabilistic algorithms. PhD thesis, University of Cambridge, Computer Laboratory (2003)
21. Abrial, J.R.: Modeling in Event-B: system and software engineering. Cambridge University Press (2010)

22. Abrial, J.R., Butler, M., Hallerstede, S., Hoang, T.S., Mehta, F., Voisin, L.: Rodin: an open toolset for modelling and reasoning in Event-B. International journal on software tools for technology transfer **12**(6) (2010) 447–466
23. Morgan, C., Hoang, T.S., Abrial, J.R.: The challenge of probabilistic Event B —extended abstract—. In: ZB 2005: Formal Specification and Development in Z and B, Springer (2005) 162–171
24. Hallerstede, S., Hoang, T.S.: Qualitative probabilistic modelling in Event-B. In: Integrated Formal Methods, Springer (2007) 293–312
25. Yilmaz, E.: Tool support for qualitative reasoning in Event-B. PhD thesis, Master Thesis ETH Zürich, 2010 (2010)
26. Tarasyuk, A., Troubitsyna, E., Laibinis, L.: Reliability assessment in Event-B development. NODES 09 (2009)
27. Tarasyuk, A., Troubitsyna, E., Laibinis, L.: Integrating stochastic reasoning into Event-B development. Formal Aspects of Computing **27**(1) (2015) 53–77
28. Tarasyuk, A., Troubitsyna, E., Laibinis, L.: Towards probabilistic modelling in Event-B. In: Integrated Formal Methods, Springer (2010) 275–289
29. Aouadhi, M.A., Delahaye, B., Lanoix, A.: Moving from Event-B to probabilistic Event-B. In: Proceedings of the 32th Annual ACM Symposium on Applied Computing, ACM (2017)
30. Aouadhi, M.A., Delahaye, B., Lanoix, A.: Introducing probabilistic reasoning within Event-B. Software & Systems Modeling (Oct 2017)
31. Gaiero, D., Zola, U.: ICT Vs FCT Test : case studies. (June 2014)
32. Electronics notes: PCP Inspection Techniques And Technologies. https://www.electronics-notes.com/articles/test-methods/automatic-automated-test-ate/pcb-inspection.php
33. Butler, M., Maamria, I.: Practical theory extension in Event-B. In: Theories of Programming and Formal Methods. Volume 8051 of LNCS. (2013) 67–81