Qualitative Probabilistic Modelling in Event-B*

Stefan Hallerstede and Thai Son Hoang

ETH Zurich
Switzerland
{halstefa,htson}@inf.ethz.ch

Abstract. Event-B is a notation and method for discrete systems modelling by refinement. We introduce a small but very useful construction: qualitative probabilistic choice. It extends the expressiveness of Event-B allowing us to prove properties of systems that could not be formalised in Event-B before. We demonstrate this by means of a small example, part of a larger Event-B development that could not be fully proved before. An important feature of the introduced construction is that it does not complicate the existing Event-B notation or method, and can be explained without referring to the underlying more complicated probabilistic theory. The necessary theory [17] itself is briefly outlined in this article to justify the soundness of the proof obligations given. We also give a short account of alternative constructions that we explored, and rejected.

1 Introduction

We consider modelling of software systems and more generally of complex systems to be an important development phase. We also believe that more complex models can only be written when the method of stepwise refinement [9] is used. Formal notation is indispensable in such a modelling activity. It provides the foundation on which building models can be carried out. Simply writing a formal text is insufficient, though, to achieve a model of high quality. The only serious way to analyse a model is to reason about it, proving in a mathematically rigorous way that all required properties are satisfied.

Event-B [7] is a formalism and method for discrete systems modelling. It has been developed from the B-Method [1] using many ideas of Action Systems [8]. The semantics of an Event-B model is characterised by *proof obligations*. In fact, proof obligations have a two-fold purpose. On the one hand, they show that a model is sound with respect to some behavioural semantics. On the other hand, they serve to verify properties of the model. This goes so far that we only focus on the proof obligations and do not present a behavioural semantics at all. This approach permits us to use the same proof obligations for very different modelling domains, e.g., reactive, distributed and concurrent systems

* This research was carried out as part of the EU research project IST 511599 RODIN (Rigorous Open Development Environment for Complex Systems) http://rodin.cs.ncl.ac.uk.

[5], sequential programs [3], electronic circuits [11], or mixed designs [2], not being constrained to semantics tailored to a particular domain. Event-B is a calculus for modelling that is independent of the various models of computation.

The standard reasoning in Event-B is based on (demonic) nondeterminism which is usually sufficient for systems modelling. However, some system behaviours are more appropriately modelled probabilistically. Event-B is extensible, that is, it can be extended when more expressiveness is needed. In this article, we focus on extending Event-B with means for qualitative modelling of probability. This extension grew out of the need for "almost-certain termination" properties used in some communication protocols, e.g. [5]. We use it to demonstrate how Event-B can be extended and discuss what problems we encountered. The extension has been made so that the impact on the notation is minimal, and the resulting proof obligations are as simple as possible. We also discuss some alternatives that may appear attractive to achieve convenient notation: they would lead, however, to more complicated proof obligations. We consider this a serious drawback because we think reasoning is the main purpose of modelling.

Some probabilistic models can only be expressed in terms of numerical measures, e.g., certain reliability problems [20, Chapter 4.4], or performance problems [12]. Yet, there is also a large class of problems where the exact numerical measures are not of importance, e.g., when modelling communication protocols [15], or human behaviour [2]. When modelling these, stating exact probabilities would be over-specific: all we need is a termination property making use of a strong local fairness property associated with probabilistic choice [13]. In this article we restrict our attention to this qualitative aspect of probability.

In Event-B, simplicity and efficiency are favoured over completeness and generality [7]. Generality comes at the price of intricate reasoning and, in particular, much reduced possibilities for automated tool support [4]. The available theory [20] for probabilistic reasoning about models is very rich but associated with intricate reasoning. So, a probabilistic Event-B will have to use a simplified theory. Our requirements on probabilistic Event-B are threefold:

i. it should be *simple*, i.e., easy to understand;
ii. it should be *useful*, i.e, solve a commonly encountered class of problems;
iii. and it should permit *efficient tool support*.

Simplicity of the notation is very important because an Event-B model is understood as a means of reasoning and communication: we must not have doubts about the meaning of a model. We also require that we have good reason for the extension: if we would not know of any problem that we could solve –only or better– by means of the extended method there would be little point in extending Event-B.

Overview The paper is structured as follows. In Section 2, we give an overview of the Event-B modelling notation, along with the proof obligations that give meanings to Event-B constructs. In Section 3, we consider a probabilistic extension of Event-B for *almost-certain convergence*. In particular, Section 3.1 discusses the

necessary additions to the notation and the proof obligations in order to accommodate the extension, and in Section 3.2, we consider the rejected alternatives. An example of a communication protocol is given in Section 4 to illustrate our approach. In Section 5, we give justifications of our proof obligations. Finally, a summary and some conclusions are presented in Sections 6.

2 The Event-B Modelling Notation

Event-B [7], unlike classical B [1], does not have a fixed syntax. Instead, it is a collection of modelling elements that are stored in a repository. Still, we present the basic notation for Event-B using some syntax. We proceed like this to improve legibility and help the reader remembering the different constructs of Event-B. The syntax should be understood as a convention for presenting Event-B models in textual form rather than defining a language.

Event-B models are described in terms of the two basic constructs: *contexts* and *machines*. Contexts contain the static part of a model whereas machines contain the dynamic part. Contexts may contain *carrier sets*, *constants*, *axioms*, where carrier sets are similar to types [7]. In this article, we simply assume that there is some context and do not mention it explicitly. Machines are presented in Section 2.1, and machine refinement in Section 2.2.

2.1 Machines

Machines provide behavioural properties of Event-B models. Machines may contain *variables*, *invariants*, *theorems*, *events*, and *variants*. Variables v define the state of a machine. They are constrained by invariants $I(v)$. Possible state changes are described by means of events. Each event is composed of a *guard* $G(t, v)$ and an *action* $S(t, v)$, where t are *local variables* the event may contain. The guard states the necessary condition under which an event may occur, and the action describes how the state variables evolve when the event occurs. An event can be represented by the term

$$\mathsf{any}\ t\ \mathsf{where}\ G(t,v)\ \mathsf{then}\ S(t,v)\ \mathsf{end} \quad . \tag{1}$$

The short form

$$\mathsf{when}\ G(v)\ \mathsf{then}\ S(v)\ \mathsf{end} \tag{2}$$

is used if event e does not have local variables, and the form

$$\mathsf{begin}\ S(v)\ \mathsf{end} \tag{3}$$

if in addition the guard equals *true*. A dedicated event of the form (3) is used for *initialisation*. The action of an event is composed of several *assignments* of the form

$$x\ :=\ E(t,v) \tag{4}$$

$$x\ :\in\ E(t,v) \tag{5}$$

$$x\ :|\ Q(t,v,x') \quad , \tag{6}$$

where x are some variables, $E(t, v)$ expressions, and $Q(t, v, x')$ a predicate. Assignment form (4) is *deterministic*, the other two forms are *nondeterministic*. Form (5) assigns x to an element of a set, and form (6) assigns to x a value satisfying a predicate. The effect of each assignments can also be described by a before-after predicate:

$$BA\big(x \;:=\; E(t,v)\big) \quad \widehat{=} \quad x' \;=\; E(t,v) \tag{7}$$

$$BA\big(x \;:\in\; E(t,v)\big) \quad \widehat{=} \quad x' \;\in\; E(t,v) \tag{8}$$

$$BA\big(x \;:|\; Q(t,v,x')\big) \quad \widehat{=} \quad Q(t,v,x') \quad . \tag{9}$$

A before-after predicate describes the relationship between the state just before an assignment has occurred (represented by unprimed variable names x) and the state just after the assignment has occurred (represented by primed variable names x'). All assignments of an action $S(t, v)$ occur simultaneously which is expressed by conjoining their before-after predicates, yielding a predicate $A(t, v, x')$. Variables y that do not appear on the left-hand side of an assignment of an action are not changed by the action. Formally, this is achieved by conjoining $A(t, v, x')$ with $y' = y$, yielding the before-after predicate of the action:

$$BA\big(S(t,v)\big) \quad \widehat{=} \quad A(t,v,x') \;\wedge\; y' = y \quad . \tag{10}$$

In proof obligations we represent the before-after predicate $BA\big(S(t, v)\big)$ of an action $S(t, v)$ directly by the predicate

$$\boldsymbol{S}(t,v,v') \quad .$$

Proof obligations serve to verify certain properties of a machine. All proof obligations in this article are presented in the form of sequents: "antecedent" $\vdash$ "succedent".

For each event of a machine, *feasibility* must be proved:

$$\begin{array}{l} \\ \vdash \end{array} \begin{array}{l} I(v) \\ G(t,v) \\ (\exists v' \cdot \boldsymbol{S}(t,v,v')) \quad . \end{array} \tag{11}$$

By proving feasibility, we achieve that $\boldsymbol{S}(t, v, v')$ provides an after state whenever $G(t, v)$ holds. This means that the guard indeed represents the enabling condition of the event.

Invariants are supposed to hold whenever variable values change. Obviously, this does not hold a priori for any combination of events and invariants and, thus, needs to be proved. The corresponding proof obligation is called *invariant preservation*:

$$\begin{array}{l} \\ \vdash \end{array} \begin{array}{l} I(v) \\ G(t,v) \\ \boldsymbol{S}(t,v,v') \\ I(v') \quad . \end{array} \tag{12}$$

Similar proof obligations are associated with the initialisation event of a machine. The only difference is that the invariant does not appear in the antecedent of the proof obligations (11) and (12). For brevity, we do not treat initialisation differently from ordinary events of a machine. The required modifications of the concerned proof obligations are obvious.

2.2 Machine Refinement

Machine refinement provides a means to introduce more details about the dynamic properties of a model [7]. For more on the well-known theory of refinement, we refer to the Action System formalism that has inspired the development of Event-B [8]. We present some important proof obligations for machine refinement. As mentioned before, the user of Event-B is not presented with a behavioural model but only with proof obligations. The proof obligations describe the semantics of Event-B models.

A machine CM can refine at most one other machine AM. We call AM the *abstract* machine and CM a *concrete* machine. The state of the abstract machine is related to the state of the concrete machine by a *glueing invariant* $J(v, w)$, where v are the variables of the abstract machine and w the variables of the concrete machine.

Each event ea of the abstract machine is *refined* by one or more concrete events ec. Let abstract event ea and concrete event ec be:

$$ea \quad \widehat{=} \quad \mathsf{any}\ t\ \mathsf{where}\ G(t,v)\ \mathsf{then}\ S(t,v)\ \mathsf{end} \tag{13}$$

$$ec \quad \widehat{=} \quad \mathsf{any}\ u\ \mathsf{where}\ H(u,w)\ \mathsf{then}\ T(u,w)\ \mathsf{end} \quad . \tag{14}$$

Somewhat simplified, we can say that ec refines ea if the guard of ec is stronger than the guard of ea, and the glueing invariant $J(v, w)$ establishes a simulation of ec by ea:

$$\begin{array}{l} \quad I(v) \\ \quad J(v,w) \\ \quad H(u,w) \\ \quad \boldsymbol{T}(u,w,w') \\ \vdash \ (\exists t, v' \cdot G(t,v) \ \wedge \ \boldsymbol{S}(t,v,v') \ \wedge \ J(v',w')) \quad . \end{array} \tag{15}$$

In the course of refinement, often *new events* ec are introduced into a model. New events must be proved to refine the implicit abstract event *skip* that does nothing. Moreover, it may be proved that new events do not collectively diverge by proving that a *variant* $V(w)$ is bounded below:

$$\begin{array}{l} \quad I(v) \\ \quad J(v,w) \\ \quad H(u,w) \\ \vdash \ V(w) \in \mathbb{N} \quad , \end{array} \tag{16}$$

and is decreased by each new event. We refer to the corresponding proof obligation as *progress*:

$$\begin{array}{l} \quad I(v) \\ \quad J(v,w) \\ \quad H(u,w) \\ \quad \boldsymbol{T}(u,w,w') \\ \vdash \quad V(w') < V(w) \quad , \end{array} \tag{17}$$

where we assume that the variant is an integer expression. It can be more elaborate [7] but this is not relevant here. We call events that satisfy (16) and (17) *convergent*.

3 Qualitative Probabilistic Event-B

The purpose of qualitative probabilistic reasoning is to provide the concept of *almost-certain convergence* [13,17][1]. Similarly to [13,17] qualitative probabilistic reasoning is introduced into Event-B by means of the *qualitative probabilistic choice*[2]:

$$S \oplus T \quad ,$$

where S or T are chosen with some positive probability (see Section 5). The probabilistic extension should not depart from the existing structure of Event-B machines. Hence, we only consider introducing probabilistic choice in places where we already have nondeterministic choice. In Event-B nondeterministic choice appears in three places:

i. choice among different events;
ii. choice of local variables of events;
iii. nondeterministic assignments.

In each of these, we could also use probabilistic choice. We present our favoured solution based on iii. in Section 3.1, and discuss the alternatives based on i. and ii. in Section 3.2.

3.1 Almost Certain Convergence in Event-B

In this section, we introduce step by step the proof obligations for almost-certain convergence in Event-B. Although we treat probability on the level of assignments, we actually do not allow to mix probabilistic assignments and nondeterministic assignments in the same event. This saves us from having to define the meaning of their simultaneous joint effect. Hence, we say the *action* of an event is either *probabilistic* or *nondeterministic*. Still, for better readability, we

[1] The authors of [13,17] use the term "almost-certain termination".
[2] We do not use the "abstract probabilistic choice" to avoid clashes with other refinement terminology, e.g., in expression like "concrete abstract probabilistic choice".

introduce some notation for qualitative probabilistic assignments corresponding to (6):

$$x \ \oplus| \ Q(t, v, x') \quad . \tag{18}$$

With respect to invariant preservation a probabilistic action behaves identically to a nondeterministic action, i.e., demonically (see Section 5). However, it behaves angelically with respect to progress. We can rephrase the progress proof obligation (17) as follows:

$$\begin{array}{l} I(v) \\ J(v, w) \\ H(u, w) \\ \vdash \\ (\forall w' \cdot \boldsymbol{T}(u, w, w') \ \Rightarrow \ V(w') < V(w)) \end{array} \quad ,$$

i.e. the action *must* decrease the variant $V(w)$. The corresponding proof obligation for a new event with a probabilistic action follows from the angelic interpretation of the action. This means it *may* decrease the variant $V(w)$:

$$\begin{array}{l} I(v) \\ J(v, w) \\ H(u, w) \\ \vdash \\ (\exists w' \cdot \boldsymbol{T}(u, w, w') \ \wedge \ V(w') < V(w)) \end{array} \quad . \tag{19}$$

Note, that proof obligation (19) subsumes feasibility (11).

For convergence of an event, (16) and (17) are sufficient. For almost-certain convergence of an event, on the other hand, the corresponding proof obligations (16) and (19) are not sufficient. An upper bound $U(w)$ is required that dominates the variant $V(w)$:

$$\begin{array}{l} I(v) \\ J(v, w) \\ H(u, w) \\ \vdash \\ V(w) \leq U(w) \end{array} \quad , \tag{20}$$

for all new events.

Figure 1 shows the evolution of the variant $V(w)$ and the upper bound $U(w)$ in a concrete machine for a new nondeterministic event nd and a new probabilistic event pr: event nd *must* decrease the variant $V(w)$ whereas pr *may* decrease it. However, the possible variation of $V(w)$ by event pr is limited below by the constant 0 –proved by means of (16)– and above by $U(w)$. The upper bound $U(w)$ itself is bound below by 0 as a consequence of (16) and (20). Given that $U(w)$ is constant or, at least, not increasing, this is sufficient for almost-certain convergence of nd and pr. For all new events of the concrete machine we have to prove:

$$\begin{array}{l} I(v) \\ J(v, w) \\ H(u, w) \\ \boldsymbol{T}(u, w, w') \\ \vdash \\ U(w') \leq U(w) \end{array} \quad , \tag{21}$$

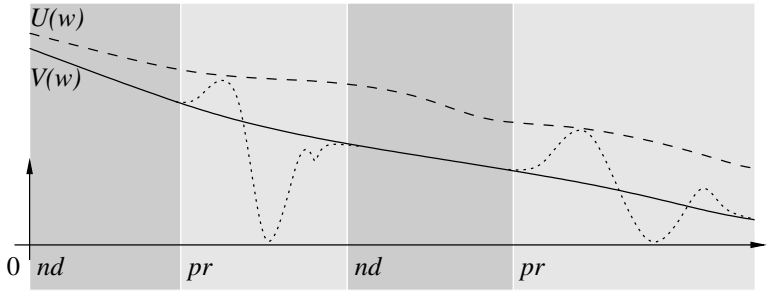


Fig. 1: Almost-certain convergence

Note, that proof obligation (21) is based on the demonic interpretation of the actions of all new events, i.e. all new events *must* not increase the upper bound. Hence, the following fact makes the difference to "certain" convergence: new events with probabilistic actions *may* decrease the variant but *must* not increase the upper bound.

The infimum probability associated with the probabilistic action $\boldsymbol{T}(u, w, w')$ must be greater than zero [17]. Using qualitative probabilistic assignment (18), we can only achieve this by requiring finiteness of the possible choices for w' of the probabilistic action $\boldsymbol{T}(u, w, w')$:

$$\begin{array}{l} I(v) \\ J(v, w) \\ H(u, w) \\ \vdash \ \mathrm{finite}(\{w' \mid \boldsymbol{T}(u, w, w')\}) \quad . \end{array} \tag{22}$$

Events with probabilistic actions that satisfy (19) to (22) are called *almost-certainly convergent*. Note, that almost-certain convergence also imposes a proof obligation (21) on new nondeterministic events, and that if we have new events with nondeterministic actions and new events with probabilistic actions we prove their joined almost-certain convergence.

3.2 The Rejected Alternatives

In order to see the advantages of the approach to almost-certain convergence presented in the Section 3.1, we discuss the two alternatives: probabilistic choice among different events or probabilistic choice of local variables of events. We begin with the discussion with the latter.

It seems natural to introduce probabilistic choice at the level of local variables, say:

$$ec \quad \widehat{=} \quad \mathsf{prob\ any}\ u\ \mathsf{where}\ H(u, w)\ \mathsf{then}\ T(u, w)\ \mathsf{end}$$

However, treating probabilistic choice on this level would lead to unnecessarily complicated proof obligations while our aim is to keep them simple. In

particular, probabilistic progress proof obligations would be difficult compared to (19):

$$\begin{array}{l} \quad I(v) \\ \quad J(v,w) \\ \vdash \ (\exists u \cdot H(u,w) \ \wedge \ (\forall w' \cdot \boldsymbol{T}(u,w,w') \ \Rightarrow \ V(w') < V(w))) \quad . \end{array} \tag{23}$$

We would have to think about two quantifiers, whereas in (19) only one existential quantification needs to be discarded.

Probabilistic choice among different events has been discussed in [19]. This approach does only require little modification to the Event-B notation. It requires the introduction of additional variables to group probabilistic choices, say:

$$\begin{array}{lcl} ec_1 & \widehat{=} & \mathsf{prob}\ a\ \mathsf{any}\ u_1\ \mathsf{where}\ H_1(u_1,w)\ \mathsf{then}\ T_1(u_1,w)\ \mathsf{end} \\ ec_2 & \widehat{=} & \mathsf{prob}\ a\ \mathsf{any}\ u_2\ \mathsf{where}\ H_2(u_2,w)\ \mathsf{then}\ T_2(u_2,w)\ \mathsf{end} \quad , \end{array}$$

denoting the abstract probabilistic choice $ec_1 \oplus ec_2$. For probabilistic progress we would obtain a proof obligation with two disjuncts ($i = 1, 2$):

$$(\exists u_i \cdot H_i(u_i,w) \ \wedge \ (\forall w' \cdot \boldsymbol{T}_i(u_i,w,w') \ \Rightarrow \ V(w') < V(w)))$$

in its succedent.

More problems may appear when trying to specify more general probabilistic choices, say, between n components where n is a positive number, e.g., in the dining philosophers [20, Chapter 3]. We also need to determine the order in which probabilistic choices and nondeterministic choices are resolved: there are still nondeterministic choices among events and of local variables. Given the intricate relationship of probabilistic and nondeterministic choice this could potentially lead to models very difficult to comprehend. Then perhaps, the best would be to restrict the body of the event to being entirely deterministic. It appears that we would have to make decisions that may seem arbitrary or introduce restrictions that make the notation more complex.

3.3 Refinement

As mentioned in the introduction, we consider refinement to be crucial in the development of complex systems. As usual, it is possible to refine a nondeterministic action by a probabilistic action [18]. Concerning refinement of events with probabilistic actions, we have two major possibilities: either we permit probabilistic choice to be refined or we do not permit it.

If we consider probabilistic actions to be an "implementation" of some behaviour, it may be considered less essential to be able to refine it. However, if we would disallow refinement of events with probabilistic actions, we still need a technique that permits us to carry out refinements of machines that ultimately contain probabilistic actions. Fortunately, Event-B provides already such technique in form of *anticipated* events [6]. Anticipated events are used to construct lexicographical variants. In Event-B, an anticipated event is eventually refined

by a convergent event. Until then, we have to prove that it does *not impede progress*:

$$\begin{array}{l} I(v) \\ J(v,w) \\ H(u,w) \\ \boldsymbol{T}(u,w,w') \\ \vdash \\ V(w') \leq V(w) \quad . \end{array} \tag{24}$$

Note the similarity to (21) except that the variant is replaced by the upper bound. We could simply use an almost-certainly convergent event instead of the convergent event. This would allow us to carry out all refinements in the standard way without probabilistic consideration. Only at a very late stage, probabilistic concerns would enter the scene.

Using a second technique, called *merging* [6], we could easily extend the approach to model abstractly probabilistic choice between, say, two events – not spelling it out. Early in a development we could introduce two anticipated events. At a later stage they could be merged into one anticipated event, and subsequently this event could be refined by an event with a probabilistic action.

If we need to refine probabilistic actions, we have to take into account the angelic interpretation for probabilistic progress (19). As a consequence of this, the possible choices offered by probabilistic actions must not be reduced. This could be achieved by requiring that probabilistic actions must be functionally refined. One could consider the *variables* assigned to in a probabilistic action as probabilistic, too. Then we would require functional refinement for these variables, and we may be able to treat refinement of probabilistic variables similarly to refinement of *external* variables; see [7]. This remains to be investigated.

Which techniques are more appropriate only (more) experience will show.

4 Example: Contention Resolution in the Firewire Protocol

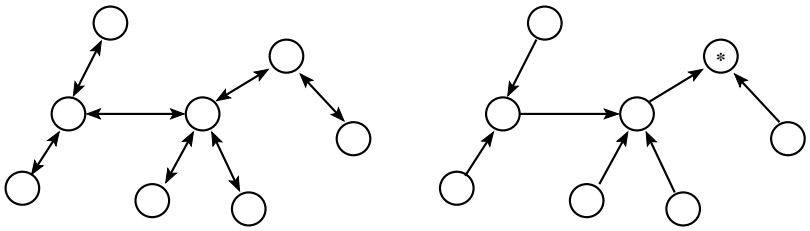
The Contention problem in the Firewire tree identify protocol [15,16] is one example of a use of probability to break the symmetry. The example has been treated in classical B [13,17]. In this section, we will look at how we can achieve a similar result in Event-B.

We use the contention problem in the Firewire protocol to demonstrate the usefulness of qualitative probabilistic modelling in a practical problem [5]. In our presentation we do not deal with the full model but focus almost-certain convergence which allows us to prove a probabilistic termination property of the Firewire protocol left open in [5].

In this section, we first give an overview of the Firewire protocol. Then we give the specification of the contention problem in Event-B. We show the failure of an attempt to use nondeterministic resolution and replace that by a probabilistic approach based on the proposal in Section 3.1.

4.1 Overview of the Firewire Protocol

Purpose A set of devices that is linked by a network of bidirectional connections. The network is an acyclic graph with devices as nodes (Figure 2a). The



(a) Initial state of network

(b) State of network after leader election (leader marked with a "∗")

Fig. 2: Abstraction of leader election protocol

protocol provides a symmetric and distributed solution finding a node that will be the leader of the network in a *finite* amount of time. All devices run the same algorithm to find the leader of the network. Figure 2b shows a possible state of the network of Figure 2a after a leader has been elected. The Firewire tree identify protocol for achieving this is described below.

Protocol Any node with only one connection can send the message "**req**" via that connection requesting the neighbouring node to be leader. Also, any node that has already received the message "**req**" via all its connections except one, can send the message "**req**" via that last remaining connection. Message sending happens distributed and nondeterministically, i.e., there is no supervisory coordination. Eventually, there will be one node that received the message "**req**" via all its connections: that node will become the leader of the network. An example of the initial state and possible final state is shown in Figure 2.

Contention At the final stage of the protocol, there are two nodes left that are linked to each other and have not yet sent the message "**req**". If both nodes try to send the message "**req**" via that (bidirectional) connection, a livelock occurs where it cannot be decided which should become the leader. Each node detects

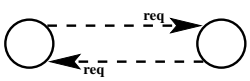


Fig. 3: Contention

the problem by receiving the message "**req**" from the node to which it has just sent the same message. We identify this as the *contention problem* illustrated in Figure 3.

Fortunately, there exists a probabilistic protocol to resolve the contention within a finite time; this is proved in Event-B by means of almost-certain convergence in Section 4.4 below. Before it is proved, we present the protocol and show that (demonic) nondeterminism is unsuitable to model the probabilistic behaviour. The protocol works as follows:

Each node independently chooses with the same non-zero probability, either to send the message after a **short** delay or after a **long** delay (the assumption for the **long** delay being that it is long enough for the message to be transferred from one node to another). Eventually, it is "almost certain" that one of them will choose to send the message after a **short** delay, while the other node will choose to send the message after a **long** delay. The message that was sent after a **short** delay will then be received before the other is sent (according to the assumption). An example for solving contention can be seen in Figure 4, where one process sends a message after a **short** delay and the other after a **long** delay.

(a) Message sent after **short** wait is received, the other message not sent

(b) State after contention resolution (leader marked with a "∗")

Fig. 4: Probabilistic contention resolution

4.2 Event-B Model of the Contention Problem

An Event-B model of the Firewire tree identify protocol has already been developed in [5]. We do not repeat the model but focus only on the contention problem that is only partially modelled in [5] leaving the termination property of the protocol unproved. In this sense, we complete the model within the Event-B framework. We take the abstract view of contention problem only presenting what is essential. We define a carrier set $WAIT$ containing the two constants: *short* and *long*.

sets: $WAIT = \{short, long\}$

Two variables x and y represent the state of the two nodes in contention: either sending the message in a *short* or *long* delay.

variables: x, y

invariants:
$x \in WAIT$
$y \in WAIT$

There is only one event which resolves the contention (in one-shot) by assigning different values to x and y. This only specifies that the problem is to be resolved but not how.

(abstract_)resolve
when
$x = y$
then
$x, y \quad :| \quad x' \in WAIT \quad \wedge \quad y' \in WAIT \quad \wedge \quad x' \neq y'$
end

4.3 Attempting Nondeterministic Contention Resolution

We attempt to achieve contention resolution by nondeterminism. We will see why it fails and see better what is gained by probabilistic reasoning. We refine the abstract model, introducing two new variables, namely u and v, in the refinement. They represent the intermediate states of the two nodes during contention resolution.

variables: x, y, u, v

invariants:
$u \in WAIT$
$v \in WAIT$

A new event draw models (nondeterministically) the effect of randomly choosing for both the two nodes either sending messages after a *long* or a *short* delay. The new event is enabled when the values of u and v are the same.

Event resolve has an additional guard $u \neq v$ (compared to the initial model of Section 4.2) indicating that two *different* delay times u and v have been successfully drawn. In this case, x and y will be assigned to u and v, respectively, and the contention is resolved.

draw
when
$u = v$
then
$u :\in WAIT$
$v :\in WAIT$
end

(concrete_)resolve
when
$u \neq v$
$x = y$
then
$x, y := u, v$
end

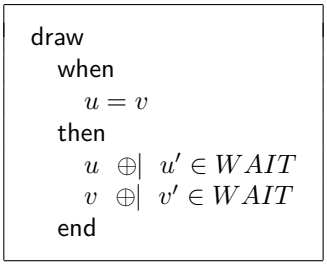
The concrete event resolve refines the abstract event resolve because the concrete event contains the guard $u \neq v$. We obtain the following proof obligation, see (15), that is trivially discharged:

$$\begin{array}{l} x' \;=\; u \\ y' \;=\; v \\ u \;\neq\; v \\ \vdash \\ x' \in WAIT \quad \wedge \quad y' \in WAIT \quad \wedge \quad x' \neq y' \quad . \end{array}$$

Failure of Demonic Nondeterminism We are left to prove that the new event draw does not take control of the system forever. However, we cannot state a variant that would satisfy the proof obligation (17). The problem is that the new event draw may behave like *skip*, doing nothing: the new event draw can be enabled always: the nondeterministic choice in event draw can always set u and v to their old values leaving draw always enabled. Using nondeterminism, we stuck and the termination property of the protocol cannot be proved.

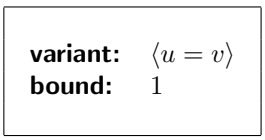
4.4 Probabilistic Contention Resolution

Probabilistic choice (18) is appropriate to model contention resolution and prove (almost-certain) termination of the protocol, thus, fully solving the problem contention. Using probabilistic choice, we can model the event draw as follows:



The meaning of the new event draw is that u and v are chosen from the set $WAIT$ probabilistically. The choices must be proper (see [17]), in other words, the probability should not be 0 or 1.

Based on the probabilistic draw, we can prove that the event draw terminates almost-certainly. According to Section 3.1, we have to show (19), (20), and (21). We take as variant the *embedded predicate* $\langle u = v \rangle$, where $\langle P \rangle$ is defined to have value 1 if P holds and 0 if P does not hold. A suitable upper bound is the constant 1.



For (21) there is nothing to prove. The proof that the variant is dominated by the bound (20) follows from the definition of the embedded predicate above:

$$\vdash \begin{array}{l} \cdots \\ \langle u = v \rangle \leq 1 \end{array} \quad .$$

Finally, one has to prove (probabilistic) progress (19). This is where nondeterminism failed: we were not able to prove progress by means of (17). We have to prove that event draw may decrease the variant $\langle u = v \rangle$:

$$\vdash \begin{array}{l} u \in WAIT \\ v \in WAIT \\ u = v \\ \exists u', v' \cdot u' \in WAIT \ \wedge \ v' \in WAIT \ \wedge \ \langle u' = v' \rangle < \langle u = v \rangle \end{array} \quad .$$

This is easy: we instantiate u' to *short* and v' to *long*, yielding for the left hand side of the inequation

$$\langle u' = v' \rangle = \langle long = short \rangle = 0$$

by definition of the embedded predicate. Also, from $u = v$, we infer for the right hand side

$$\langle u = v \rangle = 1 \quad .$$

Hence, the claim follows from $0 < 1$. Note, that the possible instantiations for u' and v' just correspond to the solutions of the contention resolution.

5 Soundness

In this section, we give justifications for the proof obligations of Section 3.1. We sketch the derivation of the proof obligations from the underlying theory. The theory is based on predicate and expectation transformers [20]. The gap left to the relational model used can be bridged by the well-known relationship between predicate transformers and before-after predicates, see e.g. [1].

The probabilistic reasoning presented in this article is based on qualitative probabilistic choice $\oplus$ (see [13, Chapter 3.2]). It is characterised by the following demonic and angelic distribution laws:

$$[\![S \ \oplus \ T]\!] \, P \quad \widehat{=} \quad [S]P \ \wedge \ [T]P \tag{25}$$

$$[\![S \ \oplus \ T]\!] \, P \quad \widehat{=} \quad [S]P \ \vee \ [T]P \quad . \tag{26}$$

The first law, called *demonic distribution*, is used when proving invariant preservation and the second, called *angelic distribution*, is used when proving almost-certain termination. The above can be easily extended to qualitative probabilistic choice with multiple branches

$$S_1 \ \oplus \ \ldots \ \oplus \ S_n \quad .$$

It is interpreted similarly to qualitative probabilistic choice: it is a probabilistic choice between substitutions $S_1, \ldots, S_n$ where the probability to execute each branch is "proper". The definition for "proper" can be found in [13, Chapter 3.2]. Note, that it is essential that the choice is between a finite number of branches. The reason for this is to get "definite" probabilistic predicate transformers (see [17, Definition 3]).

In Section 3.1, we introduce the notion of probabilistic choice $x \oplus\mid P(x, x')$, which is interpreted similarly to the qualitative multiple probabilistic choice. However, we use the choice between all possible values x' satisfying $P(x, x')$. To achieve definiteness, we require $finite(\{x' \mid P(x, x')\})$. The corresponding demonic and angelic distribution laws are:

$$[\![x \ \oplus\mid \ P(x, x')]\!] \, Q(x) \quad \widehat{=} \quad (\forall x' \cdot P(x, x') \ \Rightarrow \ Q(x')) \tag{27}$$

$$[\![x \ \oplus\mid \ P(x, x')]\!] \, Q(x) \quad \widehat{=} \quad (\exists x' \cdot P(x, x') \ \wedge \ Q(x')) \tag{28}$$

Almost-certain convergence We derive almost certain convergence for Event-B using the standard model of a generalised loop [10,20] as a basis. For ease of presentation we consider a simple Event-B machine with two new events of the form

when $G(v)$ then $S(v)$ end
when $H(v)$ then $T(v)$ end ,

where $S(v)$ is probabilistic and $T(v)$ is nondeterministic (and non-probabilistic). The loop consisting of the new events is defined by:

$$\begin{array}{lcl} loop & \widehat{=} & \mathsf{do} \\ & & \quad G(v) \Longrightarrow S(v) \\ & & [\!] \\ & & \quad H(v) \Longrightarrow T(v) \\ & & \mathsf{end} \end{array}$$

We state without proof the zero-one law for probabilistic loops (Lemma 2 in [13]) adapted to our needs:

Lemma 1. *Let $I(v)$ be the invariant of the construct. Let δ be a number strictly greater than zero. If we have that*

$$I(v) \quad \Rightarrow \quad [\![G(v) \implies S(v) \ [\!] \ H(v) \implies T(v)]\!] \, I \quad , \tag{29}$$

and

$$\delta \times \langle I \rangle \quad \Rrightarrow \quad [loop]\langle true \rangle \tag{30}$$

both hold, then in fact $\langle I \rangle \ \Rightarrow \ [loop]\langle I \rangle$.

Since $[\![\cdot]\!]$ distributes through $[\!]$, the first condition (29) can be decomposed as follows:

$$
\begin{array}{ll}
& I(v) \quad \Rightarrow \quad \llbracket G(v) \implies S(v) \quad [\!] \quad H(v) \implies T(v) \rrbracket\, I(v) \\
\Leftrightarrow & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{Distribution of } \llbracket \cdot \rrbracket \text{ through } [\!] \\
& I(v) \quad \Rightarrow \quad (\llbracket G(v) \implies S(v) \rrbracket\, I(v) \quad \wedge \quad \llbracket H(v) \implies T(v) \rrbracket\, I(v)) \\
\Leftrightarrow & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{Distribution of } \llbracket \cdot \rrbracket \text{ through } \implies \\
& I(v) \quad \Rightarrow \quad (G(v) \;\Rightarrow\; \llbracket S(v) \rrbracket\, I(v) \quad \wedge \quad H(v) \;\Rightarrow\; \llbracket T(v) \rrbracket\, I(v)) \\
\Leftrightarrow & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{Logic} \\
& (I(v) \wedge G(v) \;\Rightarrow\; \llbracket S(v) \rrbracket\, I(v)) \quad \wedge \quad (I(v) \wedge H(v) \;\Rightarrow\; \llbracket T(v) \rrbracket\, I(v)) \\
\Leftrightarrow & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad T(v) \text{ is standard} \\
& (I(v) \wedge G(v) \;\Rightarrow\; \llbracket S(v) \rrbracket\, I(v)) \quad \wedge \quad (I(v) \wedge H(v) \;\Rightarrow\; [T(v)] I(v))
\end{array}
$$

From this calculation, we can derive the standard simulation proof obligation (15) applies to events with nondeterministic and probabilistic actions. Probabilistic actions are interpreted demonically using (27). The need for definiteness stems from condition (30). With the precautions, we have taken whole body *loop* is definite.

Probabilistic Progress For the second condition (30) in Lemma 1, we introduce the notion of variant. Let $V(v)$ and $U(v)$ be two natural number expressions over the state v. It can be proved as a consequence of Lemma 5 in [13]) that condition (30) is equivalent to the following conditions (31) to (33):

$$
\begin{array}{l}
\quad I(v) \quad \wedge \quad (G(v) \vee H(v)) \\
\Rightarrow \\
\quad V(v) \leq U(v) \quad ,
\end{array}
\tag{31}
$$

$$
\begin{array}{l}
\quad I(v) \quad \wedge \quad V(v) = N \\
\Rightarrow \\
\quad \llbracket G(v) \implies S(v) \quad [\!] \quad H(v) \implies T(v) \rrbracket\, (V(v) < N) \quad ,
\end{array}
\tag{32}
$$

$$
\begin{array}{l}
\quad I(v) \quad \wedge \quad U(v) = N \\
\Rightarrow \\
\quad \llbracket G(v) \implies S(v) \quad [\!] \quad H(v) \implies T(v) \rrbracket\, (U(v) \leq N) \quad ,
\end{array}
\tag{33}
$$

where N is a logical constant.

The condition (31) can be decomposed as follows:

$$
\begin{array}{ll}
& I(v) \quad \wedge \quad (G(v) \vee H(v)) \Rightarrow V(v) \leq U(v) \\
\Leftrightarrow & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{Logic} \\
& (I(v) \;\wedge\; G(v) \quad \Rightarrow \quad V(v) \leq U(v)) \quad \wedge \quad (I(v) \;\wedge\; H(v) \quad \Rightarrow \quad V(v) \leq U(v))
\end{array}
$$

The two conjuncts in the last line correspond to proof obligation (20). It must be proved that whenever a new event, nondeterministic or probabilistic, is enabled, the variant $V(v)$ must be dominated by the upper bound $U(v)$.

Furthermore, using that $[\![\cdot]\!]$ distributes through $[\!]$, condition (32) can be decomposed as follows:

$$
\begin{array}{ll}
 & I(v) \ \wedge \ V(v) = N \\
\Rightarrow & \\
 & [\![G(v) \implies S(v)]\!] \ [\!] \ [\![H(v) \implies T(v)]\!] \, (V(v) < N) \\
 & \\
\Leftrightarrow & \qquad \text{Distribution of } [\![\cdot]\!] \text{ through } [\!] \\
 & I(v) \ \wedge \ V(v) = N \\
\Rightarrow & \\
 & ([\![G(v) \implies S(v)]\!] \, (V(v) < N) \quad \wedge \\
 & [\![H(v) \implies T(v)]\!] \, (V(v) < N)) \\
 & \\
\Leftrightarrow & \qquad \text{Distribution of } [\![\cdot]\!] \text{ through } \implies \\
 & I(v) \ \wedge \ V(v) = N \\
\Rightarrow & \\
 & (G(v) \ \Rightarrow \ [\![S(v)]\!] \, (V(v) < N) \quad \wedge \\
 & H(v) \ \Rightarrow \ [\![T(v)]\!] \, (V(v) < N)) \\
 & \\
\Leftrightarrow & \qquad \text{Logic} \\
 & (I(v) \ \wedge \ V(v) = N \ \wedge \ G(v) \ \Rightarrow \ [\![S(v)]\!] \, (V(v) < N)) \\
\wedge & \\
 & (I(v) \ \wedge \ V(v) = N \ \wedge \ H(v) \ \Rightarrow \ [\![T(v)]\!] \, (V(v) < N)) \\
 & \\
\Leftrightarrow & \qquad T(v) \text{ is standard} \\
 & (I(v) \ \wedge \ V(v) = N \ \wedge \ G(v) \ \Rightarrow \ [\![S(v)]\!] \, (V(v) < N)) \\
\wedge & \\
 & (I(v) \ \wedge \ V(v) = N \ \wedge \ H(v) \ \Rightarrow \ [T(v)](V(v) < N))
\end{array}
$$

The above reasoning yields for the event with nondeterministic action the progress proof obligation (17). For the event with probabilistic action, the action $S(v)$ is interpreted angelically, yielding the probabilistic progress proof obligation (19). The derivation of proof obligation (21) from condition (33) proceeds similarly.

6 Conclusion

The method of qualitative probabilistic reasoning in Event-B that we propose comes at very little cost of extra proof effort. The introduced concept of almost-certain convergence is easy to explain, and useful for common termination proofs based on probabilistic system behaviour. The method preserves the simplicity of Event-B proof obligations only requiring a modest extension to existing proof

obligations. Furthermore, it is not necessary to make some sort of syntactic extension. We believe that this is an important advantage. Almost-certain convergence is reduced to a problem of proof. The modelling style of Event-B is not touched. We plan to implement the extension in the RODIN platform for Event-B [4].

We have not introduced concrete probabilities, see e.g. [20]. We believe that the qualitative approach already brings many benefits without the extra complication of numerical probabilistic reasoning. In most cases where only convergence is needed, specifying probabilities could be regarded as over-specification (at the cost of much more difficult proofs). Having said this, we do not dispute the usefulness of numerical probabilistic derivations. Note, that in that context the method we have presented in this article still applies – but some additional proof obligations would be needed [14]. We intend to work on such extensions to Event-B when we have more experience with the associated modelling in Event-B.

Note, that the formalisation of qualitative probabilistic choice we have chosen reflects closely the structure of Markov decision processes [22]. Hence, it should be possible to use some body of theory from this area with only little adaptation. In particular, our approach should be open to use techniques of performance analysis used with Markov decision processes [12].

We have discussed refinement in the context of qualitative probabilistic choice. It is not clear yet whether Event-B refinement should be extended or whether the present theory is sufficient. Future extensions concerning refinement of qualitative probabilistic choice should be defined to offer an alternative to existing techniques but not replace them. We think the Event-B technique of using anticipated events is very attractive because it allows us reason in a standard (non-probabilistic) way as much as possible.

Acknowledgement

We want to thank Jean-Raymond Abrial and Carroll Morgan for the discussions about this article, and suggestions for some improvements.

References

1. Jean-Raymond Abrial. *The B-Book: Assigning Programs to Meanings*. Cambridge University Press, 1996.
2. Jean-Raymond Abrial. Event driven system construction, 1999.
3. Jean-Raymond Abrial. Event based sequential program development: Application to constructing a pointer program. In Keijiro Araki, Stefania Gnesi, and Dino Mandrioli, editors, *FME 2003: Formal Methods*, volume 2805 of *LNCS*, pages 51–74. Springer, 2003.
4. Jean-Raymond Abrial, Michael Butler, Stefan Hallerstede, and Laurent Voisin. An open extensible tool environment for Event-B. In Z. Liu and J. He, editors, *ICFEM 2006*, volume 4260, pages 588–605. Springer, 2006.

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, Dominique Cansell, and Dominique Méry. Refinement and Reachability in EventB. In Helen Treharne, Steve King, Martin Henson, and Steve Schneider, editors, *ZB 2005*, volume 3455 of *LNCS*, pages 222–241, 2005.
7. Jean-Raymond Abrial and Stefan Hallerstede. Refinement, Decomposition and Instantiation of Discrete Models: Application to Event-B. *Fundamentae Informatica*, 2006.
8. Ralph-Johan Back. Refinement Calculus II: Parallel and Reactive Programs. In J. W. deBakker, W. P. deRoever, and G. Rozenberg, editors, *Stepwise Refinement of Distributed Systems*, volume 430 of *Lecture Notes in Computer Science*, pages 67–93, Mook, The Netherlands, May 1989. Springer-Verlag.
9. Ralph-Johan Back and Joakim von Wright. *Refinement Calculus: A Systematic Introduction*. Graduate Texts in Computer Science. Springer-Verlag, 1998.
10. Edsger W. Dijkstra. *A Discipline of Programming*. Prentice-Hall, Englewood Cliffs, NJ, 1976.
11. Stefan Hallerstede. Parallel hardware design in B. In Didier Bert, Jonathan P. Bowen, Steve King, and Marina A. Waldén, editors, *ZB*, volume 2651 of *LNCS*, pages 101–102. Springer, 2003.
12. Stefan Hallerstede and Michael J. Butler. Performance analysis of probabilistic action systems. *Formal Aspects of Computing*, 16(4):313–331, 2004.
13. Thai Son Hoang. *The Development of a Probabilistic B-Method and a Supporting Toolkit*. PhD thesis, School of Computer Science and Engineering — The University of New South Wales, July 2005.
14. Thai Son Hoang, Zhendong Jin, Ken Robinson, Annabelle McIver, and Carroll Morgan. Probabilistic Invariants for Probabilistic Machines. In Didier Bert, Jonathan Bowen, Steve King, and Marina Waldén, editors, *ZB2003: Formal Specification and Development in Z and B, Proceedings of the 3rd International Conference of B and Z Users*, volume 2651 of *LNCS*, pages 240–259, Turku, Finland, June 2003. Springer.
15. IEEE. *IEEE Standard for a High Performance Serial Bus. Std 1394-1995*, 1995.
16. IEEE. *IEEE Standard for a High Performance Serial Bus (supplement). Std 1394a-2000*, 2000.
17. Annabelle McIver, Carroll Morgan, and Thai Son Hoang. Probabilistic termination in B. In Didier Bert, Jonathan Bowen, Steve King, and Marina Waldén, editors, *ZB2003*, volume 2651 of *LNCS*, pages 216–239, Turku, Finland, June 2003. Springer.
18. Carroll Morgan. The Generalised Substitution Language Extended to Probabilistic Programs. In *Proceedings B'98: the 2nd International B Conference*, volume 1393 of *LNCS*, Montpelier, April 1998. Also available at [21, B98].
19. Carroll Morgan, Thai Son Hoang, and Jean-Raymond Abrial. The challenge of probabilistic event B - extended abstract. In Helen Treharne, Steve King, Martin C. Henson, and Steve A. Schneider, editors, *ZB 2005: Formal Specification and Development in Z and B*, volume 3455 of *LNCS*, pages 162–171. Springer, 2005.
20. Carroll Morgan and Annabelle McIver. *Abstraction, Refinement and Proof for Probabilistic Systems*. Monographs in Computer Science. Springer, 2005.
21. PSG. Probabilistic Systems Group: Collected Reports. At `http://web.comlab.ox.ac.uk/oucl/research/areas/probs/bibliography.html`.
22. Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, 1994.