Reasoning about Almost-Certain Convergence Properties Using Event-B [☆] ,[☆☆]

Thai Son Hoang

Institute of Information Security, ETH Zurich, Switzerland

Abstract

We propose an approach for proving that a system guarantees to establish a given property *eventually with probability one*. Using Event-B as our modelling language, our correctness reasoning is a combination of termination proofs (in terms of probabilistic convergence), deadlock-freedom and invariant techniques. We illustrate the approach by formalising some non-trivial algorithms, including the *duelling cowboys*, *Herman's probabilistic self-stabilization* and *Rabin's choice coordination*. We extend the supporting Rodin Platform (Rodin) of Event-B to generate appropriate proof obligations for our reasoning, then subsequently (automatically/interactively) discharge the obligations using the built-in provers of Rodin.

Keywords: Event-B, formal modelling, probabilistic termination, almost-certain convergence, tool support, Herman's probabilistic self-stabilization, Rabin's choice coordination.

1. Introduction

Reasoning about the correctness of probabilistic systems is a non-trivial challenge. Paper-and-pencil proofs are error-prone, whereas formal proofs are often too complicated to be manageable. In spite of these difficulties, probability is still used in many systems, since it provides much better efficiency over non-probabilistic implementations [3, 4, 5, 6, 7].

[☆] This is an extended version of our earlier work [1].
[☆☆] A more detailed version of this paper is available as a technical report [2].
Email address: `htson@inf.ethz.ch` (Thai Son Hoang)

In some probabilistic systems, we cannot be certain that a property will be eventually achieved. Instead, a slightly weaker form of assurance is more appropriate: it is established *eventually with probability one*. As an example, consider tossing a fair coin until heads come up. It cannot be certain that eventually heads will come up, however, the probability that the coin turns up heads eventually is indeed 1. This analogy has been used widely in various applications, in particular in distributed systems for symmetry-breaking protocols [4, 5, 6].

This paper presents an approach based on the Event-B modelling method [8] for formalising and reasoning about systems with qualitative probabilistic properties. Qualitative probabilistic reasoning has been introduced into Event-B [9]. A new notion of events' convergence properties is introduced: convergence with probability one. An advantage of the proposal in [9] is that the entire reasoning stays within the standard Event-B logic.

Despite its simplicity, the technique described in [9] left an open question about the interaction between refinement and qualitative probability reasoning. This makes the integration of qualitative probability reasoning with a refinement-based method such as Event-B incomplete and unsatisfactory. In this paper, we show how this reasoning can be embedded into Event-B developments without too many restrictions. Furthermore, we want to lift the reasoning about events' convergence properties into proving a certain class of system properties which we call *almost-certain convergence* properties, i.e., of the form "eventually a property holds with *probability one*". Our approach combines reasoning about qualitative probability and deadlock-freedom, and is based on the experience of [10].

We extend the *Rodin Platform* (*Rodin*) [11] in order to generate the appropriate proof obligations supporting our reasoning. Using the extended platform, we formalise several algorithms, namely, the *duelling cowboys* [12, 13], *Herman's probabilistic self-stabilization* [6] and *Rabin's choice coordination* [5]. Whereas the reasoning about the first example is straightforward, it is certainly non-trivial for the latter two examples. It involves constructing a lexicographic variant which needs to be carefully formalised and *mechanically* proved to have adequate assurance of the correctness of the algorithm. The case studies illustrate the scalability of the approach for reasoning qualitatively in Event-B: it can be applied to more complex systems other than "coin tossing" examples. Within our knowledge, the work presented here provides the first tool-supported method for proving almost-certain convergence properties of discrete transition systems, including distributed algorithms.

The rest of the paper is structured as follows. In Section 2, we give an overview of the Event-B modelling method, together with convergence and qualitative probability reasoning. We present our contributions in Section 3. Section 4 is dedicated to illustrating our approach using the aforementioned examples. Section 5 discusses some related work. Finally, we draw conclusions and propose some future research directions in Section 6.

2. Event-B and Qualitative Reasoning

Event-B [8] is a modelling method for formalising and developing systems whose components can be modelled as discrete transition systems. An evolution of the (classical) B-method [14], Event-B is centred around the general notion of *events*, which is also found in other formal methods such as Action Systems [15], TLA [16] and UNITY [17]. The semantics of Event-B, based on transition systems and simulation between such systems, is described in [8]. We will not describe in detail the semantics of Event-B here. Instead we just show some proof obligations that are important for our reasoning in the later examples.

Event-B models are organised in terms of the two basic constructs: *contexts* and *machines*. Contexts specify the static part of a model whereas machines specify the dynamic part. Contexts may contain *carrier sets*, *constants*, *axioms*, and *theorems*. Carrier sets are similar to types. Axioms constrain carrier sets and constants, whereas theorems are additional properties derived from axioms. The role of a context is to isolate the parameters of a formal model (carrier sets and constants) and their properties, which are intended to hold for all instances. For simplification, we omit references to constants, carrier sets, and the properties of them in the presentation of proof obligations.

We give an overview about machines in Section 2.1, then about machine refinement in Section 2.2, and finally about event convergence and qualitative reasoning in Section 2.3.

2.1. Machines

Machines specify behavioural properties of Event-B models. Machines may contain *variables*, *invariants* (and *theorems*), *events*, and a *variant*. Variables v define the state of a machine and are constrained by invariants $I(v)$. Theorems are additional properties of v derivable from $I(v)$. Possible state changes are described by events.

Events. An event evt can be represented by the term

$$\mathsf{evt} \mathrel{\widehat{=}} \textbf{any}\ t\ \textbf{where}\ G(t,v)\ \textbf{then}\ S(t,v)\ \textbf{end} \quad ,$$

where t stands for the event's *parameters*[1], $G(t, v)$ is the *guard* (the conjunction of one or more predicates) and $S(t, v)$ is the *action*. 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. We use the short form " evt $\widehat{=}$ **when** $G(v)$ **then** $S(v)$ **end** " when the event does not have any parameters, and we write " evt $\widehat{=}$ **begin** $S(v)$ **end** " when, in addition, the event's guard equals *true*. A dedicated event without parameters and guard is used for the *initialisation* event (usually represented as init). Note that events may be annotated with their convergence status, witnesses, and the names of the events that they refine. We will say more about these annotations later.

The action of an event is composed of one or more *assignments* of the form

$$
\begin{aligned}
x &:= E(t, v)\ , \quad \text{or} && (1)\\
x &:\in E(t, v)\ , \quad \text{or} && (2)\\
x &:\mid Q(t, v, x')\ , && (3)
\end{aligned}
$$

where x are some of the variables contained in v, $E(t, v)$ is an expression, and $Q(t, v, x')$ is a predicate. Note that the variables on the left-hand side of the assignments contained in an action must be disjoint. In (1) and (2), x must be a single variable. Assignments of the form (1) are *deterministic*, whereas the other forms are *nondeterministic*. In (2), x is assigned an element of a set $E(t, v)$. (3) refers to Q which is a *before-after predicate* relating the values v (before the action) and x' (afterwards). (3) is also the most general form of assignment and nondeterministically selects an after-state x' satisfying Q and assigns it to x. Note that the before-after predicates for the other forms are as expected; namely, $x' = E(t, v)$ and $x' \in E(t, v)$, respectively. All assignments of an action $S(t, v)$ occur simultaneously, which is expressed by conjoining their before-after predicates. Hence each event corresponds to a before-after predicate $\boldsymbol{S}(t, v, v')$ established by conjoining all before-after predicates associated with each assignment and $y = y'$, where y are unchanged variables.

Proof Obligations. Event-B defines *proof obligations*, which must be proved to show that machines meet their specified properties. We describe below the proof

[1] When referring to variables v and parameters t, we usually allow for multiple variables and parameters, i.e., they may be "vectors". When we later write expressions like $x := E(t, v)$ we mean that if x contains $n > 0$ variables, then E must also be a vector of expressions, one for each of the n variables.

obligations for invariant preservation, feasibility and deadlock-freedom. Formal definitions of all proof obligations are given in [8].

Invariant preservation states that invariants are maintained whenever variables change their values. Obviously, this does not hold a priori for any combination of events and invariants and therefore must be proved. For each event, we must prove that the invariants I are *re-established* after the event is carried out. More precisely, under the assumption of the invariants I and the event's guard G, we must prove that the invariants still hold in any possible state after the event's execution given by the before-after predicate $\boldsymbol{S}(t, v, v')$. The proof obligation is as follows:

$$I(v), G(t, v), \boldsymbol{S}(t, v, v') \vdash I(v') \, . \tag{INV}$$

Similar proof obligations are associated with a machine's initialisation event. The only difference is that there is no assumption that the invariants hold. For brevity, we do not treat initialisation differently from ordinary machine events. The required modifications of the associated proof obligations are straightforward. Note that in practice, we prove the preservation of each invariant separately.

Feasibility states that the action of an event is always feasible whenever the event is enabled. In other words, there is always a possible after value for the variables, satisfying the before-after predicate. In practice, we prove feasibility for individual assignment of the action of an event. For deterministic assignments, feasibility holds trivially. The feasibility proof obligation generated for a non-deterministic assignment of the form $x :\mid Q(t, v, x')$ is as follows:

$$I(v), G(t, v) \vdash \exists x' \cdot Q(t, v, x') \, . \tag{FIS}$$

Deadlock-freedom states that there are always some enabled events during the execution of the system. The proof obligation is as follows (where $G_i(t_i, v)$ are guards of the events of the system):

$$I(v) \vdash (\exists t_1 \cdot G_1(t_1, v)) \lor \dots \lor (\exists t_n \cdot G_n(t_n, v)) \, . \tag{DLK}$$

2.2. Machine Refinement

Machine refinement is a mechanism for introducing details about the dynamic properties of a model [8]. For more details on the theory of refinement, we refer the reader to the Action System formalism [15], which has inspired the development of Event-B. Here we sketch some central proof obligations for machine refinement which are related to our examples in Section 4.

When proving that a machine **CM** refines another machine **AM**, we refer to **AM** as the *abstract* machine and **CM** as the *concrete* machine. The states of the abstract machine are related to the states of the concrete machine by *gluing invariants* $J(v, w)$, where v are the variables of the abstract machine and w are the variables of the concrete machine. Typically, the gluing invariants are declared as invariants of **CM** and also contain the local concrete invariants constraining only w.

Each event ea of the abstract machine is *refined* by a concrete event ec (later we will relax this one-to-one constraint). Let the abstract event ea and concrete event ec be as follows:

$$\mathsf{ea} \mathrel{\widehat{=}} \textbf{any } t \textbf{ where } G(t, v) \textbf{ then } S(t, v) \textbf{ end} \tag{4}$$

$$\mathsf{ec} \mathrel{\widehat{=}} \textbf{any } u \textbf{ where } H(u, w) \textbf{ then } T(u, w) \textbf{ end} \quad . \tag{5}$$

Somewhat simplifying, we can say that ec refines ea if the guard of ec is stronger than the guard of ea (*guard strengthening*), and the gluing invariants $J(v, w)$ establish a simulation of ec by ea (*simulation*). This condition is captured by the following proof obligation:

$$I(v), J(v, w), H(u, w), \boldsymbol{T}(u, w, w') \vdash \exists t, v' \cdot G(t, v) \land \boldsymbol{S}(t, v, v') \land J(v', w') \ . \quad (\mathsf{REF})$$

In order to simplify and split the above proof obligation, Event-B introduces the notion of "witnesses" for the abstract parameters t and the after value of the abstract variables v'. The witnesses for t and v' are in the form of predicates $W_1(t, u, v, w)$ and $W_2(v', u, w, w')$, respectively. The witnesses are required to be *feasible*. The refinement proof obligation (REF) is replaced by three different proof obligations as follows:

$$I(v), J(v, w), H(u, w), W_1(t, u, v, w) \vdash G(t, v) \ , \quad (\mathsf{GRD})$$

$$I(v), J(v, w), H(t, w), \boldsymbol{T}(t, w, w'), W_1(t, u, v, w), W_2(v', u, w, w') \vdash \boldsymbol{S}(t, v, v') \ , \quad (\mathsf{SIM})$$

$$I(v), J(v, w), H(t, w), \boldsymbol{T}(t, w, w'), W_1(t, u, v, w), W_2(v', u, w, w') \vdash J(v', w') \ . \quad (\mathsf{INV\_REF})$$

In the case where t or v are retained in the concrete machine, the corresponding witnesses can be omitted. The witnesses are denoted by the keyword **with**.

The action of the concrete event is required to be feasible. The corresponding proof obligation FIS is similar to the one presented for the abstract machine, with the exception that both abstract invariants $I(v)$ and gluing invariants $J(v, w)$ can be assumed.

A special case of refinement (called superposition refinement) is when v are kept in the refinement, i.e., $v \subseteq w$. In particular, if the action of an abstract event is retained in the concrete event, the proof obligation SIM is trivial, hence we only need to consider INV_REF for proving that the gluing invariants are re-established.

Our reasoning in the later sections will often use this fact. With respect to FIS, we only need to prove the feasibility of additional assignments in the concrete event.

In the course of refinement, *new events* are often introduced into a model. New events must be proved to refine the implicit abstract event SKIP, which does nothing, i.e., does not modify abstract variable v.

The one-to-one correspondence between the abstract and concrete events can be relaxed. When an abstract event ea is refined by more than one concrete event ec, we say that the abstract event ea is *split* and prove that each concrete ec is a valid refinement of the abstract event. Conversely, several abstract events ea can be refined by one concrete ec. We say that these abstract events are *merged* together. A requirement for merging events is that the abstract events must have identical actions. We need to prove that the guard of the concrete event is stronger than the disjunction of the guards of the abstract events.

2.3. *Convergence and Qualitative Reasoning*

At any stage, it may be stated that a set of events *does not collectively diverge*; we then call these events *convergent* events. In other words, convergent events cannot take control forever and hence allow other events to occur. To prove this, one gives a *variant* V, which maps a state to a *finite set*. One then proves that each convergent event *strictly decreases* V, w.r.t the strict-subset order $\subset$. Since the variant maps a state to a finite set, $(V, \subset)$ induces a well-founded ordering on system states. The corresponding proof obligation is as follows:

$$I(v), G(t,v), \boldsymbol{S}(t,v,v') \;\vdash\; V(v') \subset V(v)\ . \qquad \text{(VAR)}$$

The above proof obligation is applied when the variant is a set expression. In Event-B, a variant can also be a natural number expression with the standard "$<$"-order. Later we will use both types of variants for our development.

In the case where the convergence of some events cannot be immediately shown, but only in a later refinement, their convergence is *anticipated* and we must prove that $V(v') \subseteq V(v)$, i.e., these anticipated events *do not enlarge* the variant. Proof obligation VAR is adapted accordingly. The convergence attribute of an event is denoted by the keyword **status** with three possible values: *convergent*, *anticipated*, or *ordinary* (for events which are not necessarily convergent).

Combining convergent and anticipated reasoning, we can construct a proof of convergence property using a lexicographic variant. As an example, consider two events e_1, e_2 which are proved to be convergent in two refinements: an abstract and a concrete level. At the abstract level, using variant V_1, e_1 is proved to be convergent, whereas e_2 is anticipated. At the concrete refinement, e_2 is proved to be convergent using variant V_2. What we have proved is that e_1, e_2 are convergent using a lexicographic variant $V = (V_1, V_2)$ with V_1 having a higher precedent. The correctness of constructing a lexicographic variant relies on the fact that standard convergence arguments are maintained by refinement [18].

In some cases, termination is not definite but *almost certain*, i.e., the probability of convergence is 1. An example is when flipping a coin, heads will eventually appear *with probability one*. This type of reasoning has been introduced into Event-B in [9]. According to this work, the action of an event can be either *probabilistic* or *non-deterministic* (but not both). With respect to most proof obligations, a probabilistic action is treated identically as a non-deterministic action. However, it behaves angelically with respect to VAR: an event with a probabilistic action *may* (as in contrast to *must*) decrease the variant $V(v)$. The new proof obligation rule for probabilistic events is as follows:

$$I(v), G(t,v) \;\vdash\; \exists v' \cdot \boldsymbol{S}(t,v,v') \wedge V(v') \subset V(v)\;. \qquad \text{(PRV)}$$

The above rule is for an abstract convergent event. For a concrete event, the corresponding proof obligation rule is similar, with the exception that one can assume that both abstract and gluing invariants hold. Note that proof obligation VAR can be given in the following similar form to PRV:

$$I(v), G(t,v) \;\vdash\; \forall v' \cdot \boldsymbol{S}(t,v,v') \Rightarrow V(v') \subset V(v)\;. \qquad \text{(VAR)}$$

Even though probabilistically convergent events can increase the variant $V(v)$, it is required that $V(v)$ is bounded above [9]. The upper bound B is a *finite constant*[2] and the proof obligation BND, which needs to be discharged for all anticipated events and convergent events (both standard and probabilistic), is

$$I(v), G(t,v) \;\vdash\; V(v) \subseteq B\;. \qquad \text{(BND)}$$

It is required that the concrete probability associated with the probabilistic action $\boldsymbol{S}(t,v,v')$ must be bounded away from zero (*proper*) [19]. This imposes a constraint on the probabilistic action $\boldsymbol{S}(t,v,v')$ stating that the possible alternatives for v' are finite:

$$I(v), G(t,v) \;\vdash\; \mathit{finite}(\{v' \mid \boldsymbol{S}(t,v,v')\})\;. \qquad \text{(FINACT)}$$

[2]In general, this could be a finite non-increasing function on the state.

Note that in practice we prove FINACT for individual assignments.

Since events with a probabilistic action behave almost identically to standard non-deterministic events (with the exception of convergence proof obligations), we do not introduce additional syntax to Event-B. Instead, we have an additional value for the convergence attribute of an event, namely *probabilistic* and treat such events differently when generating proof obligations.

A very important point is that in the same refinement, there could be some anticipated events, some convergent events, and some probabilistic events. However, regardless of their status, they have to use the same variant.

3. Contribution

Our contribution is an approach for proving that a system establishes a certain (state-)property eventually with probability one. We model the system in Event-B, augmented with arguments about convergence properties of events and deadlock-freeness proofs. We first introduce two important conditions for the soundness of the approach, before introducing the main result captured by Theorem 2. The conditions are related to the preservation of probabilistic convergence during refinement and probabilistic lexicographic variant.

Probabilistic convergence and standard refinement. The earlier work in [9] does not address the refinement of probabilistic events. Whereas the standard convergence argument is preserved by (standard) refinement [18, Chapter 3], the probabilistic convergence argument is not necessarily maintained. Refinement allows non-determinism to be reduced and as a result, a "good" choice leading to convergence could be accidentally removed. Consider the coin tossing example earlier (until "head" comes up), standard refinement allows us to replace a fair coin by an unfair coin that always turns up "tail", then termination will never be achieved. As a result, we have to restrict refinement of a probabilistic event such that it cannot remove any possible outcome of its action. A straightforward solution (which we will take) is to require that the concrete action syntactically contains the abstract action[3]. This restriction can be easily statically checked.

COND1 Probabilistic events can be refined only by (probabilistic) events retaining their action.

[3]Intuitively, strengthening the guard is no problem since it only constrains possible alternatives of event parameters, rather than affects the choice of variables' after-value.

Note that this condition **COND1** subsumes SIM and simplifies FIS as explained in Section 2.2.

Theorem 1. *Given an abstract event* ea *and a concrete event* ec *as follows.*

```
ea                              ec
  status probabilistic            status probabilistic
  any t where                     refines ea
    G(t, v)                       any t, u where
  then                              H(t, u, v, w)
    v :| S(t, v, v')              then
  end                               v :| S(t, v, v')
                                    w :| T(t, u, v, w, w')
                                  end
```

Assume that ea *is probabilistic convergent with variant $V(v)$, i.e., satisfying* PRV*;* ec *is feasible, i.e., satisfying* FIS*; and* ec *is a refinement of* ea*, i.e., satisfying* GRD *and* INV_REF*. Then* ec *is probabilistic convergent with variant $V(v)$.*

Proof. The corresponding proof obligation PRV is as follows:

$$I(v), J(v,w), H(t,u,v,w) \vdash \exists v', w' \cdot S(t,v,v') \wedge T(t,u,v,w,w') \wedge V(v') \subset V(v)\,. \tag{6}$$

It is easy to see that (6) can be derived from the fact that $H(t,u,v,w)$ is stronger than $G(t,v)$ (GRD of ec); ea may decrease $V(v)$ (ea satisfies PRV); and feasibility of action $w :| T(t,u,v,w,w')$ (ec satisfies FIS). □

Note that we still need to prove that the additional assignment to w is finite, i.e., satisfying FINACT.

Probabilistic lexicographic variant. Typically, we reason about convergence properties of events gradually, spread over several refinements, combining convergent, probabilistic and anticipated events, using different variants at different levels of refinement. Without loss of generality, we assume that we have a set of events $e_1, \ldots, e_n$, which we prove to be either convergent or probabilistic, using variants $V_1, \ldots, V_n$, accordingly. Altogether, the set of events terminates *probabilistically* with a lexicographic variant formed by combining individual variants at different refinements $V = (V_1, \ldots, V_n)$. A condition for the variant for probabilistic termination is that it must be bounded above, and our constructed lexicographic variant V is no different from that. As a result, we require that not only those variants in $V_1, \ldots, V_n$ concerning probabilistic events, but *all variants need to be* *bounded above*. This constraint can be relaxed for variants that are used before a probabilistic event is introduced. The reason is that only probabilistic events can increase the variant, and if the variant is not increased then it is bounded above by its initial value.

COND2 All variants of a machine containing a probabilistic event must be bounded above.

Proving almost-certain convergence properties. To prove that a system eventually establishes certain conditions P with probability one, we follow the approach in [10] for reasoning about liveness properties, with the correctness argument combining appropriate proofs of event convergence (both standard and probabilistic) and deadlock freedom.

We develop the system (taking into account **COND1** and **COND2**) such that in the last refinement, we have a machine of the following shape.

1. There is a unique *ordinary* event (which will be referred to as the "observer" event) of the following form.

   $$\mathsf{obs} \mathrel{\widehat{=}} \textbf{when } P \textbf{ then } \textsc{skip} \textbf{ end}$$

   This event does not change the state of the machine, and has the guard the same as the condition of interest. When this event is enabled the condition P holds.
2. There is a set of convergent events CE.
3. There is a set of probabilistic (convergent) events PE.
4. The machine is deadlock-free.

Theorem 2. *Given a development of a system satisfying conditions **COND1** and **COND2** and having the last refinement satisfying conditions (1)–(4), eventually condition P is established by the system with probability one (almost certainly).*

Proof. From conditions (2) and (3), we conclude that with probability one, eventually all events in CE and PE are disabled. When this is the case, since the system is deadlock-free (condition (4)), the only enabled event is obs, i.e., condition P must hold at the same time (condition (1)). Hence the system guarantees that P holds eventually with probability one. □

Tool support. We have used *Rodin* [11] for our formal development. This is a supporting tool for creating and analysing Event-B models. It includes a proof-obligation generator and support for interactive and semi-automated theorem proving. We have extended the tool for specifying probabilistically convergent events and generating appropriate proof obligations. More detailed discussions on the tool support are in [18].

4. Examples

In this section, we illustrate our approach by developing three examples: the *duelling cowboys* [13, 12], *Herman's probabilistic self-stabilization* [6] and *Rabin's choice coordination* [5, 13]. The first and the second examples illustrate different orders of proving events' convergence properties. The last example shows a more complicated construction of a lexicographic variant, where the correctness proof stretches over several refinements. For each example, we present some informal requirements describing the example. In particular, we use some prefixes ASM to denote assumptions, FUN to denote functional requirements, and ALG to denote algorithmic descriptions.

4.1. The Duelling Cowboys

The duelling cowboys is a puzzle where some cowboys taking turn to shoot at each other [20]. Each cowboy has some probability of hitting the target. The original puzzle is concerned with the survival probability of each cowboy, given the individual hitting probability, and the rule of the game, e.g., the order for the cowboys' shooting. *Quantitative* analysis of the duelling cowboys puzzle can be seen in [12, 13].

Here we are concerned with the qualitative side of the puzzle, i.e., proving that eventually there is exactly a single surviving cowboy, with probability one. Our assumptions here are that there is a finite number of cowboys and each of them has a "proper" probability –bounded away from 0 and 1– of hitting his target.

ASM 1 There is a finite set of cowboys.

ASM 2 Each cowboy has some *proper* probability of hitting his opponent

FUN 3 Eventually, there is a single surviving cowboy.

While there is more than one surviving cowboy, they take turn to shoot at each other according the following rule.

ALG 4 First, a random surviving cowboy is chosen to shoot. Second, the chosen cowboy fires at a surviving cowboy other than himself.

4.1.1. Formal Development

Refinement Strategy. The development contains two machines. The initial model describes the problem including requirements ASM 1, ASM 2, ALG 4. We use the refinement to complete the proofs for event convergence properties, and subsequently the proof for FUN 3.

The Context. The context of the development[4] contains a set of cowboys (C), which is required to be finite, i.e., $\mathrm{finite}(C)$ (ASM 1).

Initial Model. Our initial model has a single variable s to model the set of alive cowboys. Initially, s is set to C, i.e., all cowboys are alive. There are two events, namely survives and shoots. The former acts as our *observer* event which is enabled when there is exactly one surviving cowboy. The latter models the case where an existing cowboy x *may probabilistically* get hit. The guard of shoots also states that there must be a surviving cowboy y other than x.

```
survives
  status ordinary
  when
    $\exists w \cdot s = \{w\}$
  then
    SKIP
  end
```

```
shoots
  status probabilistic
  any $x$ where
    $x \in s$
    $(\exists y \cdot y \in s \land x \neq y)$
  then
    $s :\in \{s, s \setminus \{x\}\}$
  end
```

Finally, we use variant $V_0 = s$ to prove that shoots probabilistically converges. The upper bound of the variant is the set of all cowboys C, which is finite (BND). Moreover, FINACT is guaranteed since we have exactly two alternatives associated with the action of shoots. Finally, shoots has some chance of decreasing V_0, in the case a surviving cowboy is hit and removed from s. The associated obligation PRV is as follows:

$$\ldots \vdash \exists s' \cdot s' \in \{s, s \setminus \{x\}\} \land s' \subset s \;,$$

[4] Available on-line at http://deploy-eprints.ecs.soton.ac.uk/333/.

which can be easily discharged by choosing the witness for s' as $s \setminus \{x\}$.

First Refinement. In this refinement, we introduce a new event for choosing the next cowboy to shoot. A variable t is used to keep the cowboy who has the turn to shoot next and a Boolean variable b is used to indicate whether the choice has been made, with invariant $b = \mathrm{TRUE} \Rightarrow t \in s$.

The observer event survives stays unchanged. We have a new convergent event chooses for selecting the next cowboy to shoot as follows.

```
chooses
 status $convergent$
 when
  $b = \mathrm{FALSE}$
  $\exists x, y \cdot x \in s \land y \in s \land x \neq y$
 then
  $b := \mathrm{TRUE}$
  $t :\in s$
 end
```

We refine the original event shoots as follows.

```
(abstract_)shoots
 status $probabilistic$
 any $x$ where
  $x \in s$
  $(\exists y \cdot y \in s \land x \neq y)$
 then
  $s :\in \{s, s \setminus \{x\}\}$
 end
```

```
(concrete_)shoots
 status $probabilistic$
 any $x$ where
  $x \in s$
  $b = \mathrm{TRUE}$
  $x \neq t$
 then
  $\underline{s :\in \{s, s \setminus \{x\}\}}$
  $b := \mathrm{FALSE}$
 end
```

Under the condition that a cowboy t has been chosen, a cowboy x (different from t) may be hit by a shot from t. Note that the concrete shoots satisfies our condition **COND1** for refining a probabilistic event, i.e., including the abstract action. Refinement proof obligation, i.e., GRD and SIM are trivially satisfied.

At this point, our model corresponds to the algorithm described in ALG 4. We prove that chooses converges using the variant $V_1 = \{b, \mathrm{TRUE}\}$, which is trivially decreased by chooses.

At this stage, we have the observer event survives, the convergent event chooses, and the probabilistic event shoots. Furthermore, we prove that our system is deadlock-free (encoded as a theorem in the model), we can conclude that with probability one, event survives is enabled, i.e., there is a single surviving cowboy (FUN 3), according to Theorem 2.

4.2. Herman's Probabilistic Self-Stabilization

As our second example, we consider a leader election protocol on a ring-shaped network. The distributed algorithm that we use is Herman's *probabilistic self-stabilisation* [6].

The purpose of the algorithm is to elect a single node to be the leader of a directed ring-shaped network.

ASM 5 A finite set of nodes is connected in a directed ring-shaped network.

FUN 6 Eventually, a single node is elected as the leader of the network.

The algorithm for electing the leader is as follows. Initially, each node is given a single token. At any time, each node is either holding a token or not. At each step, the nodes act *synchronously* to perform the following actions.

ALG 7 Any node holding a token makes a (proper) probabilistic choice of either to *keep* its token or *pass* it on to the next node in the ring. When, a node already holding a token receives another token, the receiving token is *discarded*.

4.2.1. Formal Model

Refinement Strategy. The development contains two machines to accommodate our reasoning about event convergence properties. The initial model corresponds to requirements ASM 5 and ALG 7. The refinement completes the proof for FUN 6.

The Context. The context of the development[5] provides a finite set of nodes (N) and a constant r formalising a directed ring-shaped network on N. The formalisation of the ring-shaped network, i.e., axm0_2–axm0_4, is similar to those from [8]. More precisely, axm0_3 states that the ring does not contain any self-loop and axm0_4 formalises the fact that the ring is connected. The context corresponds to ASM 5.

[5] Available on-line at http://deploy-eprints.ecs.soton.ac.uk/334/.

```
axioms :
 axm0_1 : $\text{finite}(N)$
 axm0_2 : $r \in N \rightarrowtail\!\!\!\!\rightarrow N$
 axm0_3 : $r \cap \text{id} = \varnothing$
 axm0_4 : $\forall S \cdot S \neq \varnothing \land r[S] \subseteq S \Rightarrow N \subseteq S$
```

Initial Model. Our initial model contains three variables:

- b: a Boolean flag indicating if the protocol has finished or not;
- l: the node which has been elected as the leader of the ring when the algorithm finished; and
- t: a set of nodes holding some token.

The relationship between variables is captured by an invariant stating that when the algorithm finishes, l is the only node holding a token, i.e., $b = \text{TRUE} \Rightarrow t = \{l\}$. Initially, every node holds a token.

We model the case where a node is elected as the leader by the following event.

```
elect
 status convergent
 any $x$ where
  $b = \text{FALSE}$
  $t = \{x\}$
 then
  $b := \text{TRUE}$
  $l := x$
 end
```

Event elect specifies that when there is no leader elected and there is a single node x holding a token then x is elected as the leader of the network.

In the case where there are two distinct nodes holding some tokens, we abstractly model how the ring (more precisely the set of nodes holding some token) evolves by the following event[6].

[6] $\mathbb{P}_1(N)$ denotes the set of all non-empty subsets of N.

```
progress
 status anticipated
 when
  $\exists x, y \cdot x \in t \land y \in t \land x \neq y$
 then
  $t :\in \mathbb{P}_1(N)$
 end
```

Finally, we have an observer event final capturing the intended purpose of the algorithm, i.e., eventually the algorithm finishes. (And as a consequence of invariant inv0_2, there is a single node holding a token elected as the leader of the network.)

$$\mathsf{final} \mathrel{\widehat{=}} \textbf{when } b = \mathrm{TRUE} \textbf{ then } \textsc{skip} \textbf{ end}$$

Using the variant $V_0 = \{b, \mathrm{TRUE}\}$, we prove that elect is convergent (which is trivial). Note that progress is *anticipated*, since it does not modify b hence does not change V_0.

First refinement. In this model, we refine progress correspondingly to the algorithm described in ALG 7.

```
progress
 status probabilistic
 when
  $\exists x, y \cdot x \in t \land y \in t \land x \neq y$
 then
  $t :| \exists p \cdot p \subseteq t \land t' = (t \setminus p) \cup r[p]$
 end
```

The explanation for the action of progress is as follows. Bound variable p represents the set of nodes that are about to *pass* their tokens to the next neighbour. The status of the ring is updated by first removing the tokens of p and passing these tokens to their next neighbours represented by $r[p]$[7]. The fact that a node currently holding a token will discard any receiving token is captured by the effect of set union operator $\cup$.

We are going to reason using the notion of intervals (set of consecutive nodes) on a ring [8] which is defined inductively as follows, where $i(x)(y)$ represents the intervals from node x to node y on ring r[8].

[7] $r[p]$ is the relational image of relation r with respect to the set p.

[8] $r\sim$ denotes the inverse relation of r

axioms :

axm1_1 : $i \in N \to (N \to \mathbb{P}(N))$
axm1_2 : $\forall x, y \cdot x \in N \land x \neq y \Rightarrow i(x)(y) = i(x)(r{\sim}(y)) \cup \{y\}$
axm1_3 : $\forall x \cdot x \in N \Rightarrow i(x)(x) = \{x\}$

In order to formalise the necessary variant proving that progress probabilistically converges, we added an auxiliary variable that will not appear in the guard of the existing events.

Consider any fixed node A of the ring (i.e., a constant), and the first node after A holding a token, say B, which will be updated accordingly during the execution of the algorithm (i.e., a variable).

inv1_1 : $B \in t$
inv1_2 : $\forall n \cdot n \in i(r(A))(B) \land n \neq B \Rightarrow n \notin t$

Invariant inv1_1 states that B is holding a token. Invariant inv1_2 states that any node in the interval from $r(A)$ (the next node after A) to B excluding B does not hold any token.

In order to maintain invariant inv1_2, we update B accordingly in progress as follows. The updated value of B (denoted as B') depends on the final value of t (denoted as t'), as captured by the following before-after predicate, which is added to the action of progress.

$$\begin{array}{l}(r(A) \in t' \ \Rightarrow \ B' = r(A)) \ \land \\ (r(A) \notin t' \ \land \ B \in t' \ \Rightarrow \ B' = B) \ \land \\ (r(A) \notin t' \ \land \ B \notin t' \ \Rightarrow \ B' = r(B))\end{array}$$

The variant that we use to prove that progress converges probabilistically is the set of nodes outside the interval from $r(A)$ to B, i.e., $V_1 = N \setminus i(r(A))(B)$. Event progress has some chance of extending $i(r(A))(B)$ (hence may decrease V_1, i.e., satisfying PRV) by having A keep its token (if it is holding one) and B pass on its token as illustrated in Figure 1.

Finally, our variant V_1 is clearly bounded above by the finite set of nodes N (satisfying BND). Moreover, the possible alternatives of the action of progress is finite (it is the possible alternatives for selecting a set of nodes passing on their tokens) (hence progress satisfies FINACT).

At this stage, our first refinement has the observer event final, the convergent event elect, and the probabilistic event progress. Furthermore, the first refinement is also deadlock-free (proved as a theorem). As a result, we can apply Theorem 2 and conclude that eventually a node is elected as the leader of the network with probability one.

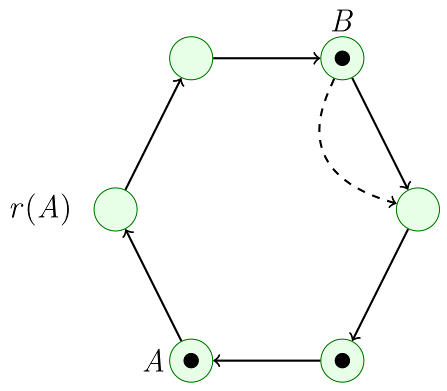

Figure 1: Increasing interval from $r(A)$ to B

4.3. Rabin's Choice Coordination Algorithm

Rabin's choice coordination algorithm as explained in [5] is an example of using probability for symmetry breaking. The choice coordination is a problem where processes $P_1, ..., P_n$ must reach a common choice out of k alternatives $A_1, ..., A_k$. It does not matter which alternative will be chosen eventually. The protocol uses k shared variables $v_1, ..., v_k$, one for each alternative. A process P_j arriving at A_i can access and modify v_i in one step *without any interruption* from other processes. The algorithm proposed by Rabin terminates with probability one.

4.3.1. Description of the Problem and Algorithm

We will look at a simplified version of the problem and the corresponding algorithm as described by Morgan and McIver [13]. Instead of n processes and k alternatives we have n tourists and 2 destinations (which we call $LEFT$ and $RIGHT$ accordingly). We also distinguish the inside and outside for each destination.

ENV 8 Each tourist can be in one of the following locations: *inside-left*, *inside-right*, *outside-left*, and *outside-right*.

Each tourist can move between the two outside locations, i.e., from outside-left to outside-right and vice versa. Furthermore, a tourist can move from the outside to the inside of the same place, e.g., from outside-left to inside-left.

ENV 9 A tourist can move between the two outside locations.

ENV 10 A tourist can move from the outside to the inside of the same place.

Other movements of the tourists are forbidden. In particular if a tourist enters an inside place, he can no longer change his location.

ENV 11 A tourist in an inside place cannot change his location

The purpose of the algorithm is to have all tourists to reach a common decision of entering the same place, without communicating directly with each other.

FUN 12 Eventually, all tourists enter the same place

Rabin's choice coordination algorithm as described by Morgan and McIver in [13] is as follows. Each tourist carries a notepad and he can write a number on it. Moreover, there are two noticeboards at the outside-left and outside-right. In the beginning, number 0 is written on all tourist notepads and on the two noticeboards.

ALG 13 Each tourist has a notepad on which he can write a number. Initially each tourist writes 0 on his notepad.

ALG 14 There are noticeboards at the outside-left and outside-right. Initially, 0 is written on both noticeboards.

Initially, each tourist independently chooses the LEFT- or RIGHT-place and goes to the outside location of that place (i.e., outside-left or outside-right). Afterwards, tourists at outside locations can asynchronously alternate between different locations according to the following algorithm.

ALG 15 An outside tourist alternates between different locations as follows.

- If there is any tourist inside, he enters this place.
- Otherwise, he compares the number n on his notepad with the number N on the notice board.
  - If $N < n$, the tourist goes inside.
  - If $N > n$, the tourist replaces n with N on his notepad and goes to the outside of other place.
  - If $N = n$, the tourist tosses a coin. If the coin comes up head, the tourist sets N' to $N + 2$. Otherwise, he sets N' to the conjugate[9] of $N + 2$. Then, he writes N' on the notice board and his notepad and goes to the outside of the other place.

We are going to formalise this version of the problem, algorithm, and proofs from Morgan and McIver [13] in the next section. Note that we make an assumption about the tourist capability: from an outside location, he can "look" inside of the same place (he still cannot see the other place, neither inside nor outside). A more realistic implementation as described in [13] is to have the first tourist entering an inside location to write some special note e.g. "Here", on the notice board. However, this will complicate our reasoning unnecessarily; hence we make this simplification.

4.3.2. Formal Development

Refinement Strategy. The development contains several levels of refinements. In the first couple of refinements, the focus is on safety properties of the algorithm. Subsequent refinements are devoted to the proof of the main liveness property FUN 12. We give some important highlights of our formal development[10]. We give a summary of the first few refinements (mostly concerning safety) as follows.

Initial Model We introduce the sets of tourists inside the LEFT and the RIGHT (li and ri) (requirement ENV 11).

First refinement We introduce the sets of tourists outside the LEFT and the RIGHT (lo and ro) (requirements ENV 8, ENV 9, ENV 10)

Second refinement We introduce the two notice boards and Rabin's algorithm (requirements ALG 13, ALG 14, ALG 15).

From Refinement 3 to Refinement 6, we prove the convergence property of different events. In Refinement 7, we prove the deadlock-freeness property, hence complete the proof of FUN 12. In the following, we focus on highlighting the proof of this liveness property.

Summary of the model at Refinement 2. Some essential information about the model at the end of the second refinement is as follows. Because of symmetry, we only show here events for when a tourist moves from the outside left to the inside left (L_IN), for when a tourist moves from left to right in two different

[9]The conjugate of a number n (denoted by $\overline{n}$) is defined to be $n + 1$ if n is even and $n - 1$ if n is odd.

[10]Available on-line at `http://deploy-eprints.ecs.soton.ac.uk/232/`.

cases (L2R_NEQ and L2R_EQ). Other events (R_IN, R2L_NEQ, and R2L_EQ) are modelled similarly. Note that events L_IN and R_IN are proved to be convergent using variant $V_0 = T \setminus (li \cup ri)$.

```
invariants :
 inv0_3 : $li = \varnothing \lor ri = \varnothing$
 inv1_1 : $\mathrm{partition}(T, li, ri, lo, ro)$
 inv2_1 : $L \in \mathbb{N}$
 inv2_2 : $R \in \mathbb{N}$
 inv2_3 : $np \in T \to \mathbb{N}$
```

```
init
 begin
  $li, ri := \varnothing, \varnothing$
  $lo, ro :\mid lo' = T \setminus ro'$
  $L, R := 0, 0$
  $np := T \times \{0\}$
 end
```

```
final
 status ordinary
 when
  $ri = T \lor li = T$
 then
  SKIP
 end
```

```
L_IN
 status convergent
 any $t$ where
  $ri = \varnothing$
  $t \in lo$
  $L < np(t) \ \lor \ li \neq \varnothing$
 then
  $li := li \cup \{t\}$
  $lo := lo \setminus \{t\}$
 end
```

```
L2R_NEQ
 status anticipated
 refines L2R
 any $t$ where
  $t \in lo$
  $li = \varnothing$
  $np(t) < L$
 then
  $ro := ro \cup \{t\}$
  $lo := lo \setminus \{t\}$
  $np(t) := L$
 end
```

```
L2R_EQ
 status anticipated
 refines L2R
 any $t$ where
  $t \in lo$
  $li = \varnothing$
  $np(t) = L$
 then
  $ro := ro \cup \{t\}$
  $lo := lo \setminus \{t\}$
  $L, np :\mid L' \in \{L + 2, \overline{L + 2}\} \land np' = np \mathbin{<\!\!+} \{t \mapsto L'\}$[11]
 end
```

The next few refinements are dedicated to proving the convergence property of events L2R_NEQ, L2R_EQ, R2L_NEQ, and R2L_EQ. We formalise the variant that

[11] $\mathbin{<\!\!+}$ denotes relational overriding.

has been proposed in [13]. The variant is a lexicographic one, with two layers: the outer layer (with higher priority) deals with the updating of L and R (3rd and 4th refinements), the inner layer (with lower priority) deals with the tourists' movements (5th and 6th refinements).

Outer layer. The outer layer of the variant relies on the relationship between L and R.

Refinement 3. In this refinement, we prove several invariants about L and R. In particular, an important property is that they cannot be too far apart, which is represented by the following relationship $\widetilde{L} - \widetilde{R} \in \{-2, 0, 2\}$, where $\widetilde{n}$ denotes the minimum of n and its conjugate $\overline{n}$. Imagining the set of natural number are split into pairs: $(0,1) \mid (2,3) \mid (4,5) \mid (6,7) \mid \ldots$, the invariant states that L and R are either in the same pair or in adjacent pairs.

Refinement 4. Note that when $\widetilde{L} = \widetilde{R}$, i.e., L and R are in the same pair, hence, either $L = R$ or $L = \overline{R}$ (equivalently $R = \overline{L}$). As a result, we can distinguish the relationship between L and R in three different cases: either $\widetilde{L} - \widetilde{R} \in \{-2, 2\}$ or $L = \overline{R}$ or $L = R$. The variant V_1 is defined as $rE(L \mapsto R)$ where

- if $L = R$ then $rE(L \mapsto R) = 2$,
- if $L = \overline{R}$ then $rE(L \mapsto R) = 0$.
- otherwise, $rE(L \mapsto R) = 1$.

The variant is clearly bounded above. We use the variant to prove the convergence property of L2R_EQ (and similarly R2L_EQ), while L2R_NEQ and R2L_NEQ are anticipated events.

To ease the burden of the proofs, we split event L2R_EQ into three different cases, depending on the current value of $rE(L \mapsto R)$.

```
L2R_EQ_0
 status $convergent$
 refines L2R_EQ
 any $t$ where
  $t \in lo$
  $li = \varnothing$
  $np(t) = L$
  $rE(L \mapsto R) = 0$
 then
  $\ldots$
 end
```

```
L2R_EQ_1
 status $probabilistic$
 refines L2R_EQ
 any $t$ where
  $t \in lo$
  $li = \varnothing$
  $np(t) = L$
  $rE(L \mapsto R) = 1$
 then
  $\ldots$
 end
```

```
L2R_EQ_2
 status $convergent$
 refines L2R_EQ
 any $t$ where
  $t \in lo$
  $li = \varnothing$
  $np(t) = L$
  $rE(L \mapsto R) = 2$
 then
  $\ldots$
 end
```

We prove that L2R_EQ_0 and L2R_EQ_2 are *convergent*, and L2R_EQ_1 is *probabilistically convergent* whereas L2R_NEQ is *anticipated* (which will be convergent with using the inner variant). The convergence attribute for the events corresponding to the $RIGHT$ are symmetric. We left out the details of the formal proofs here.

We make a remark here about splitting the events L2R_EQ and R2L_EQ into different cases. The purpose of splitting events is to separate the proofs of correctness into smaller and simpler proofs. As an alternative, we can perform *proof by cases* during proving, without splitting the events. This would reduce the number of proof obligations. However, it hides the termination argument inside the proofs and they become more complicated. Our development is more intuitive, with the correctness being easier to observe by splitting the events accordingly.

Inner layer. The variant for the inner layer is used to prove the convergence property of events L2R_NEQ and R2L_NEQ. This is done in two refinement steps.

Refinement 5. We prove that L2R_NEQ converges and R2L_NEQ is anticipated with the variant V_2 defined to be $\{t \mid np(t) < L\}$, i.e., the set of tourists carrying a number strictly smaller than on the left notice board.

Refinement 6. In the second step, we prove that R2L_NEQ converges with a symmetric variant V_3 that is $\{t \mid np(t) < R\}$.

Refinement 7 - Deadlock-freedom. In this final refinement, we merge the events that have been split earlier, i.e., L2R_EQ and R2L_EQ. Combining the convergent attribute of the sub-events, we have now that these two events are probabilistically convergent. We add a theorem to prove that our system at this point is deadlock-free. Together with the proof of convergence earlier, we can now ensure that our system satisfies the requirement FUN 12, according to Theorem 2.

5. Related Work

Our illustrative examples have also been tackled elsewhere, but mostly without machine-assisted proofs. The duelling cowboys example is extensively studied in [13] (for two cowboys), both quantitatively and qualitatively.

Regarding Herman's probabilistic self-stabilization, in the original version [6], when two tokens collide, they cancel out each other, instead of merging with each other. In Morgan and McIver's version of the algorithm [13], a node is allowed to carry more than one token. Instead of cancelling out or merging, tokens are simply accumulated. Our version of the algorithm can be seen as an abstraction of Morgan and McIver's, which allows us to prove the almost certain convergence property more conveniently. Comparing the probabilistic convergence proofs to [6, 13], we use a different variant expression which, in our opinion, is easier to formalise.

Regarding Rabin's choice coordination algorithm, we formalise the proof from Morgan and McIver [13]. The example is also used in [12, Chapter 3] as an example for reasoning about almost certain termination using classical B [14]. The main difference between the two developments is that in classical B one ends up with a sequential program which is a model of the algorithm. Our development in Event-B gives us a model of a fully distributed system. Moreover, the formalisation of lexicographic variants is suited better for Event-B since in classical B one can only have a single natural number variant. As a result, the lexicographic variant has to be encoded (unnaturally) into a natural number variant, which leads to more complicated proofs.

In the context of temporal logic, the concept of correctness with probability one is also used, and is called *P-valid* [21]. In particular, if only "simple" properties (only contain eventually ($\Diamond$), and always ($\Box$) as temporal operators) are considered, it has been showed that probabilistic choice can be replaced by strong fairness. As a result, ones can reason about P-validity without the actual concrete probabilities [22]. While their work focused on model checking probabilistic algorithms, we have showed that step-wise development of probabilistic algorithms is possible using Event-B, in which the proofs are spread amongst different levels of abstraction. Naturally, we can take the advantage of theorem proving over model checking: establishing the correctness of systems with arbitrary parameters.

Rao [23] showed a methodology for deriving properties of programs that hold deterministically or with probability one, within the context of UNITY. The key important idea of Rao is to assume that the execution of probabilistic statements is *extremely fair* (a notion that has been introduced earlier by Pnueli [24]): if a probabilistic statement is executed infinitely often, then every branch of the statement is executed infinitely often. This is similar to our angelical interpretation of the probabilistic action when it comes to reasoning about convergence properties. The purpose of [23] is eventually to develop tools and techniques for designing probabilistic algorithms. We showed in this paper that similar concepts are useful and practical using Event-B and the supporting *Rodin platform*.

6. Conclusion

We have presented a method for reasoning about almost-certain convergence properties. Our approach is an extension of the work in [9], using Event-B as the modelling method for development, combining the reasoning about deadlock-freedom and (standard/probabilistic) convergence properties of events. We illustrated our approach using three examples: the duelling cowboys [12, 13], Herman's self-stabilization [6], and Rabin's choice coordination [5]. We extended *Rodin* for supporting the generation of additional proof obligations [18], and proved all the obligations using the proof support of *Rodin*.

To our best knowledge, this is the first tool supported method for proving almost-certain properties for discrete transition systems. Using a tool assisted development method, we can have immediate feedback about our proof of correctness, e.g., in terms of modelling the problems/algorithms, and/or in terms of formalising appropriate variant. We believe that our approach can be used not only for verifying existing algorithms, but also for developing new ones.

6.1. Future Work

Currently, the probabilistic behaviour is associated with events and interpreted as for the event actions. We could benefit from having more fine-grained notion of probabilistic choice by associating probabilistic behaviour to individual assignments. An advantage is that we can have a more flexible notion of refinement when refining probabilistic events: only probabilistic assignments need to be retained, other (non-deterministic) assignment can be refined normally. We will need to adjust the proof obligation PRV accordingly.

In some other cases, it is more convenient to have probability attached to other modelling elements of the model, e.g., the guard constraining parameters of events. This requires some alternative proof obligations for reasoning about convergence properties of events and (possibly) additional constraints for refining probabilistic events. We are investigating the example of the dining cryptographers [13] along this line of extension.

Recently, we investigate proving more general liveness properties [25]. Combining the work done here with the approach in [25] would allow us to prove more classes of properties, e.g., *progress* with probability one.

Using our newly developed tool support, we have modelled other examples for proving termination including *contention resolution* [9]. In the future, we will integrate the reasoning about contention resolution with the development of the Firewire protocol [26]. Another example that we want to apply our approach to is the full k-version of Rabin's Choice Coordination algorithm [5]. In particular, for the latter example, the model of the algorithm will be straightforward with each event having an additional parameter representing a particular alternative (currently the alternative is "hard-coded" as $LEFT$ and $RIGHT$ and we have separate events for each alternative). However the challenge will be on finding the right lexicographic variant for proving probabilistic termination of the algorithm using our tool.

Another interesting future direction is to investigate support for reasoning about the expected time to converge. This requires to introduce more explicit probabilities into the model and full reasoning about *expectations* as described in [13, 12].

Acknowledgement

Part of this work is supported by the European Commission ICT project 214158 DEPLOY (http://www.deploy-project.eu/index.html). A special thank to E. Yilmaz for the work in his master thesis which inspired our work here. We thank M. Schmalz, S. Hallerstede, and A. Fürst for their critical comments on various drafts of this paper. We thank the anonymous reviewers of the AVoCS 2010 workshop (Dusseldorf, Germany) for their valuable comments for improving our paper. Finally, we would like to thank Systerel and in particular to T. Muller for the current implementation of the plug-in based on the initial work in [18].

References

[1] E. Yilmaz, T. S. Hoang, Development of Rabin's choice coordination algorithm in Event-B, Electronic Communication of the European Association of Software Science and Technology (ECEASST) 35.

[2] T. S. Hoang, Proving almost-certain convergence properties using Event-B, Tech. Rep. 768, Department of Computer Science, ETH Zurich, http://www.inf.ethz.ch/research/disstechreps/techreports (Jul. 2012).

[3] M. Rabin, Probabilistic algorithm for testing primality, Journal of Number Theory 12 (1) (1980) 128–138.

[4] IEEE, IEEE Std 1394a-2000 High Performance Serial Bus – Amendment 1 (2000).

[5] M. Rabin, The choice coordination problem, Acta Informatica 17 (1982) 121–134.

[6] T. Herman, Probabilistic self-stabilization, Information Processing Letters 35 (2) (1990) 63–67.

[7] R. Motwani, P. Raghavan, Randomized Algorithms, Cambridge University Press, 1995.

[8] J.-R. Abrial, Modeling in Event-B: System and Software Engineering, Cambridge University Press, 2010.

[9] S. Hallerstede, T. Hoang, Qualitative probabilistic modelling in Event-B, in: J. David, J. Gibbons (Eds.), IFM 2007: Integrated Formal Methods, Vol. 4591 of Lecture Notes in Computer Science, Springer Verlag, Oxford, U.K., 2007, pp. 293–312.

[10] T. Hoang, H. Kuruma, D. Basin, J.-R. Abrial, Developing topology discovery in Event-B, Science of Computer Programming 74 (11-12) (2009) 879–899.

[11] J.-R. Abrial, M. Butler, S. Hallerstede, T. Hoang, F. Mehta, L. Voisin, RODIN: An open toolset for modelling and reasoning in Event-B, Internation Journal on Software Tools for Technology Transfer (STTT) 12 (6) (2010) 447–466.

[12] T. Hoang, The development of a probabilistic B - method and a supporting toolkit, Ph.D. thesis, The University of New South Wales (Jul. 2005).

[13] C. Morgan, A. McIver, Abstraction, Refinement and Proof for Probabilistic Systems, Springer Verlag, 2005.

[14] J.-R. Abrial, The B-Book: Assigning Programs to Meanings, Cambridge University Press, 1996.

[15] R.-J. Back, Refinement calculus II: Parallel and reactive programs, in: J. W. deBakker, W. P. deRoever, G. Rozenberg (Eds.), Stepwise Refinement of Distributed Systems, Vol. 430 of Lecture Notes in Computer Science, Springer-Verlag, Mook, The Netherlands, 1989, pp. 67–93.

[16] L. Lamport, The temporal logic of actions, Transactions on Programming Languages and Systems (TOPLAS) 16 (3) (1994) 872–923.

[17] K. Chandy, J. Misra, Parallel Program Design: a Foundation, Addison-Wesley, 1989.

[18] E. Yilmaz, Tool support for qualitative reasoning in Event-B, Master's thesis, Department of Computer Science, ETH Zurich, Switzerland, `http://e-collection.ethbib.ethz.ch/view/eth:1677` (Aug. 2010).

[19] A. McIver, C. Morgan, T. S. Hoang, Probabilistic termination in B, in: D. Bert, J. Bowen, S. King, M. Waldén (Eds.), ZB2003: Proceedings of the 3rd International Conference of B and Z Users, Vol. 2651 of Lecture Notes in Computer Science, Springer-Verlag, Turku, Finland, 2003, pp. 216–239.

[20] D. M. Kilgour, S. J. Brams, The truel, Mathematics Magazine 70 (5) (1997) 315–326.

[21] S. Hart, M. Sharir, A. Pnueli, Termination of probabilistic concurrent programs, in: Symposium on Principles of Programming Languages, 1982, pp. 1–6.

[22] L. D. Zuck, A. Pnueli, Y. Kesten, Automatic verification of probabilistic free choice, in: A. Cortesi (Ed.), VMCAI 2002: Verification, Model Checking and Abstract Interpretation, Vol. 2294 of Lecture Notes in Computer Science, Springer, 2002, pp. 208–224.

[23] J. R. Rao, Reasoning about probabilistic parallel programs, ACM Transactions on Programming Languages and Systems 16 (3) (1994) 798–842.

[24] A. Pnueli, On the extremely fair treatment of probabilistic algorithms, in: D. S. Johnson, R. Fagin, M. L. Fredman, D. Harel, R. M. Karp, N. A. Lynch, C. H. Papadimitriou, R. L. Rivest, W. L. Ruzzo, J. I. Seiferas (Eds.), Symposium on the Theory of Computing, ACM, 1983, pp. 278–290.

[25] T. S. Hoang, J.-R. Abrial, Reasoning about liveness properties in Event-B, in: S. Qin, Z. Qiu (Eds.), International Conference on Formal Engineering Methods 2011, Vol. 6991 of Lecture Notes in Computer Science, Springer-Verlag, 2011, pp. 456–471.

[26] J.-R. Abrial, D. Cansell, D. Méry, A mechanically proved and incremental development of IEEE 1394 tree identify protocol, Formal Aspect of Computing 14 (3) (2003) 215–227.