Proving Almost-Certain Convergence Properties Using Event-B

Thai Son Hoang
Institute of Information Security, ETH Zurich
htson@inf.ethz.ch

Abstract. We propose a methodological approach to prove that a system guarantees to establish a 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 and standard invariant techniques. We illustrate our 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 of Event-B to generate appropriate proof obligations for our reasoning, then subsequently (automatically/interactively) discharge the obligations using the built-in provers of the Rodin platform.

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 offer incomplete satisfaction, whereas formal proofs are often too complicated to be manageable. Despite of these difficulties, probability is still used in many systems, since it provides much better efficiency over non-probabilistic implementations [1].

In some probabilistic systems, a property might not be guaranteed for certain. Instead, a slightly weaker form of assurance is more appropriate: it is established with probability one. As an example, consider tossing a fair coin. 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 [1,2,3].

Qualitative probability reasoning has been introduced into Event-B [4]. A new notion of events’ convergence properties is introduced: convergence with probability one. An advantage of the proposal in [4] is that the entire reasoning stay within the standard Event-B logic.
Despite of its simplicity, the technique described in [4] 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 about 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 [5].

We extend *Rodin* in order to generate the appropriate proof obligations supporting our reasoning. Using the extended platform, we formalise several algorithms, namely, the *duelling cowboys* [6,7], *Herman’s probabilistic self-stabilization* [3] and *Rabin’s choice coordination* [2]. 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 than just “coin tossing” examples. Within our knowledge, the work presented here provides one of the first tool supported methodologies 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, including convergence and qualitative reasoning. We present our contribution in Section 3. Section 4 is dedicated to illustrate our approach using the fore-mentioned examples. We present the summary of our tool support in Section 5. Section 6 discusses some related work. Finally, we draw conclusions and propose some future research directions in Section 7.

## 2 Event-B and Qualitative Reasoning

Event-B [8] is a modelling method for formalising and developing systems whose components can be modeled as discrete transition systems.
An evolution of the (classical) B-method [9], Event-B is now centered around the general notion of events, which also found in other formal methods such as Action Systems [10], TLA [11] and UNITY [12]. 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 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 variants. 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

$$evt \triangleq \text{any } t \text{ where } G(t, v) \text{ then } S(t, v) \text{ end} \ , \quad (1)$$

where $t$ stands for the event’s parameters, $G(t, v)$ is the guard (the conjunction of one or more predicates) and $S(t, v)$ is the action. The guard

\footnote{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.}
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

$$\text{evt} \triangleq \text{when } G(v) \text{ then } S(v) \text{ end}$$

(2)

when the event does not have any parameters, and we write

$$\text{begin } S(v) \text{ end}$$

(3)

when, in addition, the event’s guard equals true. A dedicated event in the form of (3) is used for the initialisation event (usually represented as init). Note that events may be annotated to indicate whether they refine other events, witnesses, and with their convergence status. We will say more about these annotations later.

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

$$x := E(t, v)$$

(4)

or

$$x :\in E(t, v)$$

(5)

or

$$x :| Q(t, v, x')$$

(6)

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 (4) and (5), $x$ must be a single variable. Assignments of the form (4) are deterministic, whereas the other two forms are nondeterministic. In (5), $x$ is assigned an element of a set $E(t, v)$. (6) refers to $Q$ which is a before-after predicate relating the values $v$ (before the action) and $x'$ (afterwards). (6) 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 two 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 together their before-after predicates. Hence each event corresponding to a before-after predicate $S(t, v, v')$ 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 have their specified properties. We describe below the proof obligation for invariant preservation and feasibility. 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 $S(t, v, v')$. The proof obligation is as follows.

$$ I(v), G(t, v), S(t, v, v') \vdash I(v') $$  \hspace{1cm} (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, by the property of conjunctivity, we can prove the preservation of each invariant separately.

Feasibility  states that the action of an event is always feasible whenever the event is enable. In other words, there are 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 $x : | Q(t, v, x')$ is as follows.

$$ I(v), G(t, v) \vdash \exists x'. Q(t, v, x') $$  \hspace{1cm} (FIS)

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 [10], 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 refine 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.

\[
\begin{align*}
\text{ea} & \equiv \text{any } t \text{ where } G(t, v) \text{ then } S(t, v) \text{ end} \quad (7) \\
\text{ec} & \equiv \text{any } u \text{ where } H(u, w) \text{ then } T(u, w) \text{ end} \quad (8)
\end{align*}
\]

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.

\[
\begin{align*}
I(v) \\
J(v, w) \\
H(u, w) \\
T(u, w, w') \\
\vdash \exists t, v'. G(t, v) \land S(t, v, v') \land J(v', w') \quad (9)
\end{align*}
\]

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 are in the form of predicates \( W_1(t, u, v, w) \) (for \( t \)), and \( W_2(v', u, w, w') \) (for \( v' \)), which are required to be feasible. Given the witnesses, the refinement proof obligation (9) is replaced by three different proof obligations as follows.

\[
\begin{align*}
I(v), J(v, w), H(u, w), W_1(t, u, v, w) & \vdash G(t, v) \quad \text{(GRD)} \\
I(v), J(v, w), H(t, w), T(t, w, w'), W_1(t, u, v, w), W_2(v', u, w, w') & \vdash S(t, v, v') \quad \text{(SIM)}
\end{align*}
\]
\[ I(v), J(v, w), H(t, w), T(t, w, w'), W_1(t, u, v, w), W_2(v', u, w, w') \vdash J(v', w') \]

(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 \texttt{with}.

The action of the concrete event requires to be feasible. The corresponding proof obligation \texttt{FIS} is similar to the one presented for the abstract machine, with the exception that both abstract and gluing invariants 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 \texttt{SIM} is trivial, hence we only need to consider \texttt{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 \texttt{FIS}, we only need to prove the feasibility of any addition assignment in the concrete event.

In the course of refinement, \textit{new events} are often introduced into a model. New events must be proved to refine the implicit abstract event \texttt{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 events \( ec \), we say that the abstract event \( ea \) is \textit{split} and prove that each concrete \( ec \) is a valid refinement of the abstract event. Conversely, several abstract events \( ae \) can be refined by one concrete \( ec \). We say that these abstract events are \textit{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 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 \textit{does not collectively diverge}; we then call these events \textit{convergent} events. In other words, convergent events cannot take control forever and hence allows other events to occur. To prove this, one gives a \textit{variant} \( V \), which maps a state to a \textit{finite set}. One then proves that each convergent event \textit{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.

\[ I(v), G(t, v), S(t, v, v') \vdash V(v') \subset V(v) \]  \hspace{1cm} (VAR)

As explained above, we assume that the variant is a set expression. In Event-B, a variant can also be a natural number expression with the normal decreasing order "<" [8]. 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 \( V_1 \) has higher precedent. The correctness of constructing a lexicographic variant relies on the fact that standard convergence arguments are maintained by refinement [13].

In some cases, termination is not definite but almost certain, i.e., the termination 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 [4]. 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.

\[ I(v), G(t, v) \vdash \exists v' \cdot S(t, v, v') \land V(v') \subset V(v) \]  \hspace{1cm} (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 \textbf{VAR} can be given in the following similar form to \textbf{PRV}.

\[
I(v), G(t, v) \vdash \forall v'. S(t, v, v') \Rightarrow V(v') \subseteq V(v)
\]

\textbf{(VAR)}

Even though probabilistically convergent events can increase the variant \(V(v)\), it is required that \(V(v)\) is bounded above [4]. The upper bound \(B\) is a \textit{finite constant}\(^2\) and the proof obligation \textbf{BND}, which needs to be discharged for all anticipated events and convergent events (both standard and probabilistic), is

\[
I(v), G(t, v) \vdash V(v) \subseteq B.
\]

\textbf{(BND)}

Finally, it is required that the possible alternatives for a probabilistic action are finite.

\[
I(v), G(t, v) \vdash \text{finite}(\{v' \mid S(t, v, v')\})
\]

\textbf{(FINACT)}

Note that in practice we prove \textbf{FINACT} for individual assignment.

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 \textit{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 a methodological 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. The basic ingredients of our approach are as follows.

---

\(^2\) In general, this could be a finite non-decreasing function on the state.
Probabilistic convergence and standard refinement  The earlier work in [4] does not address the refinement of probabilistic events. Whereas the standard convergence argument is preserved by (standard) refinement [13, 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 abstract action remained unchanged as part of the concrete action\(^3\). This restriction can be easily checked syntactically, and seems to be reasonable through our example.

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

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

We give some arguments to reason that probabilistic convergence properties are maintained by standard refinement, given **COND1**. Our setting is an abstract event \(ea\) and a concrete event \(ec\) as follows.

\[
\begin{array}{l}
\text{ea} \\
\text{status probabilistic} \\
\text{any } t \text{ where } G(t, v) \\
\text{then} \\
\quad v : | S(t, v, v') \\
\text{end}
\end{array}
\]

\[
\begin{array}{l}
\text{ec} \\
\text{refines } ea \\
\text{status probabilistic} \\
\text{any } t, u \text{ where } H(t, u, v, w) \\
\text{then} \\
\quad v : | S(t, v, v') \\
\quad w : | T(t, u, v, w, w') \\
\text{end}
\end{array}
\]

In \(ec\) the abstract action is included, and there might be some additional action modifying new variables \(w\). Moreover, we assume that \(ea\)

---

\(\text{Intuitively, strengthening the guard is no problem since it only constraints possible alternatives of event parameters, rather than effects the choice of variables’ after-value.}\)
is probabilistic convergent with variant $V(v)$ –satisfying $\text{PRV}$, ec is feasible –satisfying $\text{FIS}$– and is a refinement of ea –satisfying $\text{GRD}$ and $\text{INV\_REF}$.

**Theorem 1.** Using variant $V(v)$, ec satisfies $\text{PRV}$

**Proof.** The corresponding proof obligation $\text{PRV}$ is as follows.

$$
\begin{align*}
I(v) \\
J(v, w) \\
H(t, u, v, w) \\
\vdash \\
\exists v', w' \cdot S(t, v, v') \land T(t, u, v, w, w') \land V(v') \subset V(v) \\
\end{align*}
$$

(10)

It is easy to see that (10) can be derived from the fact that $H(t, u, v, w)$ is stronger than $G(t, v)$ ($\text{GRD}$ of ec); ea may decrease $V(v)$ (ea satisfies $\text{PRV}$); and feasibility of action $w : | T(t, u, v, w, w')$ (ec satisfies $\text{FIS}$).

Note that we still need to prove that the addition assignment to $w$ is finite, i.e. satisfying $\text{FINACT}$.

In A, we give some discussions on other alternatives for maintaining probabilistic convergence arguments with refinement.

**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 level 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 [5] 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.
   \[
   \text{obs} \triangleq \text{when } P \text{ then SKIP end}
   \]
   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. A set of convergent events $CE$.
3. 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 [14] 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 Section 5.
4 Examples

In this section, we illustrate our approach by developing three examples: the duelling cowboys [7,6], Herman’s probabilistic self-stabilization [3] and Rabin’s choice coordination [2,7].

The first and the second examples illustrate different orders of proving events’ convergence properties. In the duelling cowboys example, we first convert an event to probabilistic and in the next refinement prove another event converges standardly. In Herman’s probabilistic self-stabilization example, we adopt an opposite strategy, first convert an event to convergent before prove another event to be probabilistic. The first example also illustrates how we can refine a probabilistic event. The last example shows a more complicated construction of a lexicographic variant, with the correctness proof spreads over several refinements.

4.1 The Duelling Cowboys

The duelling cowboys is a puzzle where some cowboys taking turn to shoot at each other [15]. 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 [6,7].

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 are a finite number of cowboys and each of them has a “proper” probability –bounded away from 0 and 1– of hitting his target.

<table>
<thead>
<tr>
<th>ASM 1</th>
<th>There is a finite set of cowboys.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASM 2</td>
<td>Each cowboy has some probability of hitting his opponent, bounded away from 0 and 1.</td>
</tr>
</tbody>
</table>
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.

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

Formal Development  We now present our formal development of the duelling cowboys example\(^4\).

The context of the development contains a finite set of cowboys (ASM 1).

\begin{align*}
\text{carrier sets: } & C \\
\text{axioms: } & \text{axm0.1 : finite}(C)
\end{align*}

Initial Model  Our initial model has a single variable \( s \) to model the set of current surviving cowboys. Initially, \( s \) is set to \( C \), i.e. all cowboys are alive.

\begin{align*}
\text{variables: } & s \\
\text{invariants: } & \text{inv0.1 : } s \subseteq C \\
\text{init } & \text{begin} \\
& s := C \\
\text{end}
\end{align*}

There are two additional events, namely survives and shoots. The former event acts as our observer event which is enabled when there is exactly one surviving cowboy. The latter event models the case where an

\(^4\) Available on-line at http://deploy-eprints.ecs.soton.ac.uk/333/.
existing cowboy $x$ may get hit. The guard of shoots also states that there must be a surviving cowboy $y$ other than $x$. The action *probabilistically* keeps $s$ the same or remove $x$ from $s$.

We prove that the model are deadlock-free which is encoded as a theorem in the model.

\[
\text{DLF} : (\exists w \cdot s = \{w\}) \lor (\exists x \cdot x \in s \land (\exists y \cdot y \in s \land x \neq y))
\]

In order to prove DLF, we require an additional invariant to state that there is always some surviving cowboys.

\[
\text{inv0.2} : s \neq \emptyset
\]

Invariant inv0.2 maintains trivially by survives (since it does nothing) and shoots (since there must be at least two surviving cowboys for the event to be enabled).

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. This leads to trivial BND proof obligations. Moreover, FINACT is guaranteed since we have exactly two alternatives associated with the action of shoots. Furthermore, 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,

\[
\ldots \vdash \exists s' \cdot s' \in \{s, s \setminus \{x\}\} \land s' \subset s
\]

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 if the choice has been made.

variables: $\ldots, b, t$

invariants:  
inv1.1 : $b = \text{TRUE} \Rightarrow t \in s$

The observer event survives stays unchanged. We refine shoots as follows.

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. For GRD, we have at least two surviving cowboys in $x$ and $t$. Finally inv1.1 is maintained trivially by shoots since it set $b$ to FALSE.

We have a new convergent event chooses for selecting the next cowboy to shoot. The condition is that there are no chosen cowboys and there are at least two surviving cowboys. A surviving cowboy is then chosen to have the turn for the next shot.
chooses

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

Invariant inv1 is maintained trivially by chooses. At this point, our model corresponds to the algorithm described in ALG 4.

We are going to reason that our concrete model is deadlock-free by proving that it is relatively deadlock-free with the abstract model. Since the observer event survives unchanged, it is sufficient to prove that if the abstract shoots is enable, either the concrete shoots or chooses is enable. This is encoded as a theorem (REL_DLF) in our concrete model.

$$\text{REL_DLF} : \quad (\exists x, x \in s \land (\exists y, y \in s \land x \neq y))$$
$$\Rightarrow (b = \text{FALSE} \land (\exists x, y \cdot x \in s \land y \in s \land x \neq y)) \lor$$
$$\quad (\exists x, x \neq t \land x \in s)$$

We prove that chooses converges using the variant $V_1 = \{b, \text{TRUE}\}$, which is bounded above by the (finite) set of Booleans BOOL (COND2). The variant $V_1$ is decreased by chooses: before the event, $b$ is FALSE and the variant is $\{\text{TRUE, FALSE}\}$, and after $b$ is set to TRUE, hence $V_1$ is reduced to $\{\text{TRUE}\}$.

The summary of events’ convergence status is as follows.

<table>
<thead>
<tr>
<th>event</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>survives</td>
<td>ordinary</td>
</tr>
<tr>
<td>shoots</td>
<td>probabilistic</td>
</tr>
<tr>
<td>chooses</td>
<td>convergent</td>
</tr>
</tbody>
</table>

Together with the fact that our system is (absolute) deadlock-free, 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.
Proofs Statistics  The proof statistics for the development of the duelling cowboys is in Table 1. For this relatively simple example, most of the proof obligations are automatically discharged.

<table>
<thead>
<tr>
<th>Model</th>
<th>Total</th>
<th>Auto. (%)</th>
<th>Man. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial model</td>
<td>5</td>
<td>3(60%)</td>
<td>2(40%)</td>
</tr>
<tr>
<td>1st Refinement</td>
<td>9</td>
<td>9(100%)</td>
<td>0(0%)</td>
</tr>
<tr>
<td>Total</td>
<td>14</td>
<td>12(86%)</td>
<td>2(14%)</td>
</tr>
</tbody>
</table>

Table 1. Proof statistics for the Duelling Cowboys development

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 [3].

There are a finite set of identical nodes connected in a directed ring-shaped network.

<table>
<thead>
<tr>
<th>ENV 5</th>
<th>A finite set of nodes is connected in a directed ring-shaped network.</th>
</tr>
</thead>
</table>

The purpose of the algorithm is to elect a single node to be the leader of the network.

<table>
<thead>
<tr>
<th>FUN 6</th>
<th>Eventually, a single node is elected as the leader of the network.</th>
</tr>
</thead>
</table>

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.
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.

Formal Model We now present our formal development of Herman’s algorithm in Event-B.

The context 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. \texttt{axm0.2–axm0.4}, is similar to those from \cite{8}. More precisely, \texttt{axm0.3} states that the ring does not contains any self-loop and \texttt{axm0.4} formalises the fact that the ring is connected. The context corresponds to the environment requirement ENV 5.

\begin{tabular}{|c|c|}
\hline
\textbf{carrier sets:} & \(N\) \\
\hline
\textbf{constants:} & \(r\) \\
\hline
\end{tabular}

\begin{verbatim}
axioms:
axm0_1 : finite(N)
axm0_2 : r ∈ N ↠ N
axm0_3 : r ∩ id = ∅
axm0_4 : ∀S[S, S ≠ ∅ ∧ r[S] ⊆ S ⇒ N ⊆ S]
\end{verbatim}

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.

\footnote{Available on-line at \url{http://deploy-eprints.ecs.soton.ac.uk/334/}.}
variables: \( b, l, t \)

invariants:

\[ \text{inv0}_1 : l \in \mathbb{N} \]
\[ \text{inv0}_2 : b = \text{TRUE} \Rightarrow t = \{ l \} \]

init

begin
\( b := \text{FALSE} \)
\( l \in \mathbb{N} \)
\( t := \mathbb{N} \)
end

The relationship between variables are captured by invariant \( \text{inv0}_2 \) stating that when the algorithm finishes, \( l \) is the only node holding a token. Initially, every node holds a token.

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

 elect

\[ \text{status} \quad \text{convergent} \]
\[ \text{any} \quad x \quad \text{where} \]
\( b = \text{FALSE} \)
\( t = \{ x \} \)
\[ \text{then} \]
\( b := \text{TRUE} \)
\( l := x \)
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. It is easy to see that \( \text{inv0}_2 \) maintained by elect trivially.

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

The guard of progress states that there are at least two nodes holding token, then tokens’ allocation is updated, such that there is at least one node holding token, i.e. \( t \) becomes a non-empty set of nodes. Again, inv0.2 maintained trivially by progress since \( b \) must be FALSE before and after the invocation of progress.

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).

final
  when
    \( b = \text{TRUE} \)
  then
    SKIP
  end

We prove that the model is deadlock-free (encoded as a theorem), which requires an additional invariant stating that there are always some nodes holding a token.

\[ \text{inv0.3} : t \neq \emptyset \]

Using the variant \( V_0 = \{ b, \text{TRUE} \} \), we prove that elect is convergent (which is trivial). Note that progress is anticipated, since it does not touch \( b \) 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 : x \in t \land y \in t \land x \neq y
then
  t : | \exists p : p \subseteq t \land t' = (t \setminus p) \cup r[p]
end

The explanation for the action of progress is as follows. Bounded variable \( p \) represents the set of nodes that going to pass their tokens to the next neighbour. The status of the ring is updated by first removing the tokens of \( p \) and putting these tokens to the their next neighbours represented by \( r[p] \). The fact that two tokens can be merged into one 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 ring \([8]\) which defined inductively as follows, where \( i(x)(y) \) represents the intervals from node \( x \) to node \( y \) on ring \( r \).

constants: \( i \)

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

Axiom axm1_3 states that the interval from a node \( x \) to itself is the singleton set containing \( x \). For two distinct node \( x \) and \( y \), the interval from \( x \) to \( y \) is the union of the interval from \( x \) to the previous node of \( y \), i.e. \( r \sim(y) \), and the singleton set containing \( y \) itself (axm1_2). Additional theorems about intervals are omitted here.

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.

---

\(^6\) \( r[p] \) is the relational image of relation \( r \) with respect to the set \( p \).

\(^7\) \( r \sim \) denotes the inverse relation of \( r \).
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).

<table>
<thead>
<tr>
<th>constants:</th>
<th>$A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>axioms:</td>
<td>$\text{axm1.4}: A \in N$</td>
</tr>
<tr>
<td>variables:</td>
<td>$B$</td>
</tr>
</tbody>
</table>

Invariant $\text{inv1.1}$ states that $B$ is holding a token and $\text{inv1.2}$ states that the interval from $r(A)$ (the next node after $A$) to $B$ excluding $B^8$ does not contain any node holding a token.

In order to maintain invariant $\text{inv1.2}$, 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.

$$(r(A) \in t' \Rightarrow B' = r(A)) \land$$
$$(r(A) \notin t' \land B \in t' \Rightarrow B' = B) \land$$
$$(r(A) \notin t' \land B \notin t' \Rightarrow B' = r(B))$$

The explanation of the above before-after predicate is as follows. Consider the updated tokens’ allocation, i.e., $t'$:

1. if the node after $A$, i.e., $r(A)$ holds a token, then $B'$ will be $r(A)$;
2. otherwise, i.e. when $r(A)$ does not hold a token, and $B$ still holds a token then $B'$ is the same as $B$ (unchanged);
3. otherwise, $B'$ is the next node after $B$, i.e. $r(B)$, since the original token from $B$ has now moved to $r(B)$.

The reasoning that this before-after predicate updates $B$ correctly, i.e. maintaining $\text{inv1.1}$ and $\text{inv1.2}$, relies on properties of intervals.

$^8$ This interval is not the same as the interval from $r(A)$ to $r\sim(B)$, in particular when $A$ and $B$ are the same.
We prove these properties as theorems derived from axioms \texttt{axm1.1–axm1.3}. We omit the details of the proofs here.

The variant that we use to prove that progress converges probabilistically is the set of nodes that outside the interval from $r(A)$ to $B$, i.e. $V_1 = N \setminus i(r(A))(B)$. We informally argue below why progress may extend $i(r(A))(B)$ (hence may decrease $V_1$), i.e. satisfying \texttt{PRV}.

When progress is enabled, there are at least two nodes holding some tokens hence $B$ is different from $A^9$. As a result progress can extend the interval $i(r(A))(B)$ by updating $t'$ such that $B$ becomes $r(B)$. This can be done, by having $A$ keeps its token (if it is holding one) and $B$ passes on its token (e.g., Figure 1).

![Fig. 1. Increasing interval from $r(A)$ to $B$](image)

Finally, our variant $V_1$ is clearly bounded above by the finite set of nodes $N$ (satisfying \texttt{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 \texttt{FINACT}).

The summary of the events’ status at the first refinement is as follows.

<table>
<thead>
<tr>
<th>event</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>final</td>
<td>ordinary</td>
</tr>
<tr>
<td>elect</td>
<td>convergent</td>
</tr>
<tr>
<td>progress</td>
<td>probabilistic</td>
</tr>
</tbody>
</table>

\(^9\) If $B$ is the same as $A$, the intervals $i(r(A))(B)$ contains all nodes. And according to \texttt{inv1.2} there can be only one node holding a token which is $A$ itself.
Note also that in the first refinement, we do not modify the guard of the events (compared to its abstraction), hence the first refinement also deadlock-free (we have proved that the initial model deadlock-free). 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.

Proofs statistics Table 2 shows the proof statistics for the development of Herman’s probabilistic self-stabilization. Two thirds of the total interactive proofs are associated with proving several theorems about intervals. These theorems are used intensively for the interactive proofs in the first refinement, to reason about the probabilistic convergence property.

<table>
<thead>
<tr>
<th>Model</th>
<th>Total</th>
<th>Auto. (%)</th>
<th>Man. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interval context</td>
<td>32</td>
<td>17 (53%)</td>
<td>15 (47%)</td>
</tr>
<tr>
<td>Initial model</td>
<td>10</td>
<td>9 (90%)</td>
<td>1 (10%)</td>
</tr>
<tr>
<td>1st Refinement</td>
<td>14</td>
<td>7 (50%)</td>
<td>7 (50%)</td>
</tr>
<tr>
<td>Total</td>
<td>56</td>
<td>33 (59%)</td>
<td>23 (41%)</td>
</tr>
</tbody>
</table>

Table 2. Proof statistics for the development of Herman’s probabilistic self-stabilization algorithm

4.3 Rabin’s Choice Coordination Algorithm

Rabin’s choice coordination algorithm as explained in [2] 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.

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 [7]. 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.
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.

A tourist can move between the two outside locations.

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.

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.

Eventually, all tourists enter the same place.

Rabin’s choice coordination algorithm as described by Morgan and McIver in [7] 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, a tourist at an outside location can 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\(^{10}\) 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 [7] in the next section. Note that we make an assumption about the tourist capability: he from an outside

---

\(^{10}\) 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.
location 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 [7] 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.

**Formal Development**  In this section, we present the formal development of Rabin’s choice coordination algorithm in Event-B\(^ {11}\).

*Initial Model. The Sets of Inside Tourists*  We assume that there is a context with a *finite* carrier set \( T \) representing the set of tourists. In this initial model, we have two sets of tourists, namely \( li \) and \( ri \), representing those at the inside-left and inside-right accordingly. Note that invariant \( \text{inv0}_3 \) states that at least one of the two locations is always empty. Initially, both variables are empty sets, since all tourists are outside.

<table>
<thead>
<tr>
<th>variables: ( li, ri )</th>
</tr>
</thead>
<tbody>
<tr>
<td>invariants:</td>
</tr>
<tr>
<td>( \text{inv0}_1 : \ li \subseteq T )</td>
</tr>
<tr>
<td>( \text{inv0}_2 : \ ri \subseteq T )</td>
</tr>
<tr>
<td>( \text{inv0}_3 : \ li = \emptyset \lor ri = \emptyset )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>init</th>
</tr>
</thead>
<tbody>
<tr>
<td>begin</td>
</tr>
<tr>
<td>( li := \emptyset )</td>
</tr>
<tr>
<td>( ri := \emptyset )</td>
</tr>
<tr>
<td>end</td>
</tr>
</tbody>
</table>

We have two events \( \text{L\_IN} \) and \( \text{R\_IN} \) to model the situation when a tourist enters the inside-left or inside-right accordingly (ENV 10). Moreover there are no leaving events: a tourist once inside cannot change his location (ENV 11).

\(^{11}\) Available on-line at http://deploy-eprints.ecs.soton.ac.uk/232/.
The two events are convergent, with the variant \( V_0 = T \setminus (li \cup ri) \) representing the set of tourists not inside the two places.

Finally, we have one ordinary event, namely final. This is our observer event with its guard stating that all tourists will end up in the same inside place. Note that according to invariant inv0.3, if all tourists are in one place, the other place must be empty.

Refinement 1. The Sets of Outside Tourists  There are two new variables \( lo \) and \( ro \) representing the tourists outside the two places. Invariant inv1.1\(^ {12} \) states that a tourist cannot be at two locations at the same time, and each tourist must be in one of the locations. This corresponds to the requirement ENV 8. Initially, some tourists decide to go to the outside-left and some tourists to the outside-right.

\(^{12}\) partition\((S, s_1, \ldots, s_n)\) means that subsets \( s_1, \ldots, s_n \) are pairwise disjoint and their union is \( S \).
variables: \ldots, lo, ro

invariants:
\begin{align*}
\text{inv1.1} & : \text{partition}(T, li, ri, lo, ro)
\end{align*}

init
\begin{align*}
\text{begin} \\
\ldots \\
lo, ro : | \text{lo}' = T \setminus \text{ro}' \\
\text{end}
\end{align*}

There are two new events namely L2R and R2L to model the movement of a tourist between the two outside locations. This corresponds to the requirement ENV 9.

\begin{align*}
\text{L2R} & \quad \text{status} \quad \text{anticipated} \\
\quad \text{any} \quad t \quad \text{where} \\
& \quad t \in lo \\
& \quad li = \emptyset \\
& \quad \text{then} \\
& \quad ro := ro \cup \{t\} \\
& \quad lo := lo \setminus \{t\} \\
& \quad \text{end}
\end{align*}

\begin{align*}
\text{R2L} & \quad \text{status} \quad \text{anticipated} \\
\quad \text{any} \quad t \quad \text{where} \\
& \quad t \in ro \\
& \quad ri = \emptyset \\
& \quad \text{then} \\
& \quad lo := lo \cup \{t\} \\
& \quad ro := ro \setminus \{t\} \\
& \quad \text{end}
\end{align*}

The guards \( li = \emptyset \) and \( ri = \emptyset \) state that the tourists can only alternate between the outside locations if there is no one inside. This is a part of the algorithm described by the requirement ALG 15. The two new events only modify new variables \( ro \) and \( lo \) hence clearly refine SKIP. Moreover, invariant \text{inv1.1} is preserved since the events only change the location for one particular tourist from outside-left to outside-right and vice versa. These events are \textit{anticipated} at the moment. We will consider their convergent property in subsequent refinements.

Events L\_IN and R\_IN are refined accordingly to take into account the new variables. Since the events corresponding to \textit{LEFT} and \textit{RIGHT} are symmetric, from now on, we present only events corresponding to \textit{LEFT}. The refinement of event L\_IN is as follows.
Note that the guard strengthening proof obligation $GRD$ follows from the fact that a tourist can only be in one location at a time (invariant $inv1.1$). Since the abstract action is retained, $SIM$ is trivial. Moreover invariant $inv1.1$ is maintained since the event merely moves a tourist from the outside-left to the inside-left.

**Refinement 2. Rabin’s Algorithm** We introduce the two notice boards outside the places and the tourists’ notepads where they can write some number on it. Initially, the number 0 is written on the notice boards and all the notepads. This corresponds to the requirements $ALG 13$ and $ALG 14$.

**variables:** $\ldots, L, R, np$

**invariants:**
- $inv2.1 : L \in \mathbb{N}$
- $inv2.2 : R \in \mathbb{N}$
- $inv2.3 : np \in T \rightarrow \mathbb{N}$

```
init
begin
\ldots
L, R, np := 0, 0, T \times \{0\}
end
```

We can now specify under which condition a tourist can move from one location to another.
Events modelling the movement of a tourist from an outside location to an inside location, i.e., event $L_{\text{IN}}$ (and similarly $R_{\text{IN}}$), are guard-strengthened as follows. The guard $L < np(t) \lor li \neq \emptyset$ states that a tourist $t$ can move inside the left place only if the number on his notepad is greater than the number on the left notice board or if there is already someone at inside-left.

For events modelling the movement of a tourist between two outside locations, there are two different cases. The events corresponding to the movement of a tourist from $LEFT$ to $RIGHT$ are modelled by the two events $L2R_{\text{EQ}}$ and $L2R_{\text{NEQ}}$ depending on if the number on the tourist notepad is equal or strictly smaller than the number on the notice board. Using $\pi$ denoting the conjugate number of $n$, the two events are as follows.
The actions of the above events update the tourist notepad and the notice board accordingly. Note that both events are refinements of the original event L2R, i.e., the original event is *split* into two cases. Note that these events model the movement of a tourist according to the requirement ALG 15, with the exception that we use non-deterministic choice currently in L2R_EQ. This is an abstraction of the actual probabilistic choice (i.e., coin tossing), which we will introduce later.

Up to this refinement model, we have modelled all requirements except FUN 12. In other words, we have established the model of the problem and the algorithm. Subsequent refinements are dedicated to prove the main properties of the algorithm, i.e., eventually all tourists end up in the same place.

**Refinements 3–6. Convergence Proofs** Recall in the previous model, we have an *ordinary* event final, two *convergent* events, namely L_IN
and R_IN, and anticipated events L2R_NEQ, L2R_EQ, R2L_NEQ, and R2L_EQ. In this section, we describe our proof of (probabilistic) convergence of the anticipated events. We formalise the variant that has been proposed in [7]. The variant is a lexicographic one, with two layers: the outer layer (with higher priority) deals with the changes to L and R, the inner layer (with lower priority) deals with the tourists’ movements.

Outer layer We compare the values of L and R and notice how they can be varied. In order to understand the variant at this layer, we look at the definition of conjugate numbers. We separate the set of natural numbers into pairs: (0, 1) | (2, 3) | (4, 5) | (6, 7) | . . .. For each pair, a number is the conjugate of the other number in the pair and vice versa. The even number of each pair is also the minimum of the two. We will refer to this splitting of natural numbers later in our reasoning. We reason about the outer variant in two refinement steps.

| invariants: |
| inv3.1 : \( \forall x \cdot x \in lo \Rightarrow np(x) \leq R \) |
| inv3.2 : \( \forall x \cdot x \in ro \Rightarrow np(x) \leq L \) |
| inv3.3 : \( \tilde{L} - \tilde{R} \in \{-2, 0, 2\} \) |
| inv3.4 : \( \tilde{L} \notin np[ro] \) |
| inv3.5 : \( \tilde{R} \notin np[lo] \) |

Refinement 3. Invariants inv3.1–5 constraint the relationship between L and R. Below, we use the notation \( \tilde{n} \) to denote the minimum of n and its conjugate \( \pi \). We will not go into details about proving the preservation of these invariants, but only give some brief descriptions of them. Invariant inv3.1 states that every tourist at the outside-left carries a number not greater than the right notice board. Invariant inv3.5 states that there is no tourist at the outside-left carrying the number which is the conjugate of the number on the right notice board. The invariants related to the tourists at the outside-right, i.e. inv3.2 and inv3.4 are symmetric. Invariant inv3.3 states that the values of the two notice boards cannot be “too far apart”. Referring to the splitting of natural numbers into pairs, this invariant states that L and R must be in the same pair or in two adjacent pairs. Note that when \( \tilde{L} = \tilde{R} \), i.e. they are in the same pair, there can be two cases, either \( L = R \) or \( L = \tilde{R} \) (equivalently \( R = \tilde{L} \)). We
can distinguish the relationship between $L$ and $R$ in three different cases: either $\tilde{L} - \tilde{R} \in \{-2, 2\}$ or $L = \overline{R}$ or $L = R$. Our variant is based on this relationship.

**Refinement 4.** For the outer variant, we define the following constant function $rE$ as follows:

<table>
<thead>
<tr>
<th>axioms:</th>
<th>$rE_1$</th>
<th>$rE \in \mathbb{N} \times \mathbb{N} \mapsto {0, 1, 2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$rE_2$</td>
<td>$\forall l, r \cdot l \mapsto r \in \text{dom}(rE) \leftrightarrow \tilde{l} - \tilde{r} \in {-2, 0, 2}$</td>
<td></td>
</tr>
<tr>
<td>$rE_3$</td>
<td>$\forall l, r \cdot l \in \mathbb{N} \land l = r \Rightarrow rE(l \mapsto r) = 2$</td>
<td></td>
</tr>
<tr>
<td>$rE_4$</td>
<td>$\forall l, r \cdot l \in \mathbb{N} \land l = \tau \Rightarrow rE(l \mapsto r) = 0$</td>
<td></td>
</tr>
<tr>
<td>$rE_5$</td>
<td>$\forall l, r \cdot l \in \mathbb{N} \land \tilde{l} - \tilde{r} \in {-2, 2} \Rightarrow rE(l \mapsto r) = 1$</td>
<td></td>
</tr>
</tbody>
</table>

**variant:** $rE(L \mapsto R)$  
**bound:** 2

and define the variant $V_1$ as $rE(L \mapsto R)$ with upper bound 2. We split event $L2R_EQ$ into three different cases, depending on the current value of $rE(L \mapsto R)$. 
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. First of all, we need to prove that the variant is bounded above (BND) by the declared upper bound. This is trivial since by definition, \( rE(L \mapsto R) \leq 2 \). Next we show that each event satisfies (VAR) or (PRV) depending on their convergence attribute.

For L2R_EQ_0, this corresponds to the case that never happens, since we have \( rE(L \mapsto R) = 0 \), i.e. \( L = \overline{R} \); hence \( np(t) = \overline{R} \). However, since \( t \in lo \) and according to invariant inv3.5, we have \( \overline{R} \notin np[lo] \), which is a contradiction. In other words, the guard of L2R_EQ_0 can be used
to derive ⊥. Hence anything can be proved under the assumption of the guard of this events, including convergence proof.

For L2R_EQ_2, we have \( rE(L \mapsto R) = 2 \), i.e. \( L = R \). The action will change \( L \) to either \( L + 2 \) or \( L + 2 \), and keep \( R \) the same, hence the new value \( L' \) will be different from \( R' \), hence \( rE(L' \mapsto R') \neq 2 \), which is less than \( rE(L \mapsto R) \). As a result, the variant \( V_1 \) is decreased and hence satisfies \textsc{VAR}.

For L2R_NEQ, it does not change the value of \( L \) or \( R \). Hence the value of \( V_1 \) stays the same, i.e. is non-increasing.

For L2R_EQ_1, we first have that the possible alternatives of the after states are finite (2 in this case) and hence the event satisfies \textsc{FINACT}. Secondly, we prove that the event may decrease the variant \( V_1 \), i.e., it satisfies \textsc{PRV}. The actual proof obligation (with some simplifications by removing unnecessary hypotheses) is as follows.

\[
\begin{align*}
rE(L \mapsto R) &= 1 \\
\forall x \cdot x \in \text{lo} \Rightarrow np(x) \leq R \\
t \in \text{lo} \\
np(t) &= L \\
\vdash \exists L', np' \cdot L' \in \{L + 2, \overline{L + 2}\} \land np' = np \iff \{t \mapsto L'\} \land rE(L' \mapsto R) < rE(L \mapsto R)
\end{align*}
\]

We have from \( rE(L \mapsto R) = 1 \) that \( \overline{L} - \overline{R} \in \{-2, 2\} \). In particular, from invariant \textsc{inv3.1}, i.e. \( \forall x \cdot x \in \text{lo} \Rightarrow np(x) \leq R \), and from event’s guards \( t \in \text{lo} \) and \( np(t) = L \), we have that \( L \leq R \) and hence \( \overline{L} - \overline{R} \) must be \( -2 \). Referring to the splitting of natural numbers into pairs, when we have \( \overline{L} - \overline{R} = -2 \), it means that \( L \) is in one pair and \( R \) is in the next higher adjacent pair. For example, if \( L \) is either 2 or 3 then \( R \) is either 4 or 5. The meaning of the action assigning \( L' \) to either \( L + 2 \) or \( L + 2 \) is to have \( L' \) be in the same pair as \( R \); hence one of the alternative will satisfy condition \( L' = R \). For this case, \( rE(L' \mapsto R) = 0 < 1 = rE(L \mapsto R) \) As a result, we have proved that L2R_EQ_1 may decrease the variant \( V_1 \).

\textit{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. Event L2R_NEQ changes the value of a tourist notepad from strictly less than to equal to $L$; hence it decreases $V_2$. Event R2L_NEQ increase the value of a tourist notepad; hence it cannot increase $V_2$.

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

Note that both variant $V_2$ and $V_3$ are bounded above by the finite set of tourists $T$ (satisfying COND2).

Refinement 7. Deadlock-freedom

In this final refinement, we merge the events that have been split earlier together, 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, i.e. the disjunction of all guards always holds. In order to prove the theorem, we need the following additional invariants about the set of tourists inside the two places.

Together with the proof of convergence earlier, we can now ensure that our system satisfies the requirement FUN 12, according to Theorem 2.

Proof Statistics The statistics for our proofs are in Table 3. A large number of manual proofs are in the models for proving the outer variants and deadlock-freedom, since we need several additional supporting invariants. In particular, in order to prove obligations related to the outer variant, we split the events L2R_EQ and R2L_EQ into different cases. As a result, we have more proof obligations, which are simpler to prove. As an alternative, we can do the split while proving, i.e. to do proof by cases,
without splitting the events. This will 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. Finally, most of the manual proofs deal with arithmetic reasoning related to the modulo operator (as a consequence of the use of conjugate number), sometimes involve doing case distinctions, which are known to be difficult for automated provers.

<table>
<thead>
<tr>
<th>Model</th>
<th>Total</th>
<th>Auto. (%)</th>
<th>Man. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial model</td>
<td>6</td>
<td>6(100%)</td>
<td>0(N/A)</td>
</tr>
<tr>
<td>1st Refinement</td>
<td>8</td>
<td>7(88%)</td>
<td>1(12%)</td>
</tr>
<tr>
<td>2nd Refinement</td>
<td>19</td>
<td>15(79%)</td>
<td>4(21%)</td>
</tr>
<tr>
<td>Outer variant</td>
<td>68</td>
<td>45(66%)</td>
<td>23(34%)</td>
</tr>
<tr>
<td>Inner variant</td>
<td>7</td>
<td>4(57%)</td>
<td>3(43%)</td>
</tr>
<tr>
<td>Deadlock freedom</td>
<td>32</td>
<td>22(69%)</td>
<td>10(31%)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>140</td>
<td>99(71%)</td>
<td>41(29%)</td>
</tr>
</tbody>
</table>

Table 3. Proof statistics for the development of Rabin’s Choice Coordination Algorithm

5 Tool Support

We have implemented a plug-in to Rodin [14] for supporting the generation of proof obligations for proving probabilistic termination. The summary of the work is as follows. More details are given in [13].

**Probabilistic attribute:** An event can be marked as *probabilistic*. A probabilistic event is only treated differently from a standard event when it comes to convergence proof obligation.

**Bound element:** A new modelling element is added for declaring the upper bound.

**Static Checking:** The conditions below are checked for a model containing probabilistic events.

1. The variant $V$ (declared as usual) is either of the type integer or some set.
2. There is exactly one bound for a model where the probabilistic converge is proved. The bound element $B$ must be of the same typed as the declared variant.
3. Every probabilistic event must be refined by a probabilistic event.
4. Merging a probabilistic event and a convergent event results in a
probabilistic event.

**Proof Obligations:** Given a model, the following additional proof obli-
gations are generated for proving probabilistic convergence property.

1. The variant is always bounded above by the declared bound. (**BND**)
2. The variant may be decreased by the probabilistic events. (**PRV**)
3. The bound must be finite if it is a set. (**BFN**)
4. The bound must be well-defined. (**BWD**)

Currently, some of other proof obligations, e.g., **FINACT** are manu-
ally encoded into the model as theorems. **COND1** and **COND2** are also
manually verified by checking the models syntactically.

### 6 Related Work

Our illustrative examples also has been tackled elsewhere, but mostly
with non-mechanical proofs. The duelling cowboys is extensively studied
in [7] (for two cowboys), both quantitatively and qualitatively.

Regarding Herman’s probabilistic self-stabilization, in the original
version [3], when two tokens collide, they cancel out each other, instead
of merging with each other. In Morgan and McIver’s version of the al-
gorithm [7], a node allows to carry more than one token. Instead of can-
celling 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. Regarding the probabilistic convergence proof, we use a
different variant to [3,7] which in our opinion is easier to formalise.

Regarding Rabin’s choice coordination algorithm, we formalise the
proof from Morgan and McIver [7]. The example is also used in [6, Chap-
ter 3] as an example for reasoning about almost certain termination using
classical B [9]. 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 sin-
gle 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 [16]. 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 $P$-validity can reasoned about without the actual concrete probabilities [17]. While their work focused on model checking probabilistic algorithms, we 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 system with arbitrary parameters.

Rao [18] 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: if a probabilistic statement is executed infinitely often, then every branch of the statement is executed infinitely often. This is similar to our angelically interpretation of the probabilistic action when comes to reasoning about convergence properties. The purpose of [18] is eventually develop tools and techniques for design probabilistic algorithm. We showed in this paper that similar concepts are useful and practical using Event-B and the supporting Rodin platform.

7 Conclusion

We have presented a method for reasoning about almost-certain convergence properties. Our approach is an extension of the work in [4], using Event-B as the modelling method for development, combining with reasoning about deadlock-freedom and (standard/probabilistic) convergence properties of events. We illustrated our approach using three examples: the duelling cowboys [6,7], Herman’s self-stabilization [3], and Rabin’s choice coordination [2]. We extended Rodin [14] for supporting the generation of appropriate proof obligations concerning with this type of reasoning, and proved all the obligations using the proof support of Rodin [14].

To our best knowledge, this is the first tool supported method for proving almost-certain properties for discrete transition systems. Using
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.

7.1 Future Work

Currently, the probability is associated with event and interpreted as for
event action. We could be benefit from having more fine-grained notion
of probabilistic choice by associate probability to individual action. An
advantage is that we can have 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 case, 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) ad-
ditional constraints for refining probabilistic events. We are investigating
the example of the dining cryptographers [7] along this line of extension.

Recently, we investigate proving more general liveness properties [19].
Combining the work done here with the approach in [19] would allow us
to prove more class of properties, e.g., progress with probability one.

Using our newly developed tool support, we have modelled other ex-
amples for proving termination including contention resolution [4]. In the
future, we will integrate the reasoning about contention resolution with
the development of the Firewire protocol [20]. Another example that we
want to apply our approach is the full \( k \)-version of Rabin’s Choice Co-
ordination algorithm [2]. In particular, for the latter example, the model
of the algorithm will be straight-forward with each event having an ad-
ditional parameter representing a particular alternative (currently the al-
ternative 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.
Acknowledgement

Thai Son Hoang is supported by the European Commission ICT project 214158 DEPLOY (http://www.deploy-project.eu/index.html). We thank Matthias Schmalz and Stefan Hallerstede for their critical comments on various drafts of this paper. We thank anonymous reviewers of the AVoCS 2010 workshop (Dusseldorf, Germany) for their valuable comments for improving our paper. Finally, we would to thank Systerel and especially Thomas Muller for the current implementation of the plug-in based on the initial work in [13].

References

A Alternatives for Maintaining Probabilistic Convergence with Refinement

We discuss two alternatives for maintaining probabilistic convergence arguments.

No refinement of probabilistic events We can forbid probabilistic events to be refined any further. This is the pragmatic approach presented in [13] and in the earlier version of this paper [21]. With this restriction, we deploy an approach where the event convergence proofs are delayed until the end of the development. We first develop the system fully with all anticipated events, then converting them to either convergent/probabilistic while keeping the rest of the event unchanged. A lexicographic variant can still be built by spreading the convergence into several refinement. We reason about convergence properties at the last model in our development. The first observation here is that having no refinement of probabilistic events is too restrict (even though it should work for most examples). Moreover, reasoning about convergence properties at the last refinement level can be challenging, since all the details of the systems are presented. Some convergence argument can be observed earlier in the development and usually easier to prove then.

Generating proof obligations As an alternative for maintain probabilistic convergence properties, we could consider to generate additional proof obligations to prove that we do not remove any possibilities during refinement of the probabilistic events. The obligation will be along the line of the opposite direction of SIM proof obligation, proving that the concrete action can always simulate the abstract one. Essentially, we (almost) prove that we have refinement in the other direction from the concrete to the abstract event. This would be the most general condition for refining
a probabilistic event. However, this generating additional proof obligations that are not necessarily trivial to discharge. We do not adopt this for the moment, since this seems to be overly complicate our reasoning.

In the end, we use a slightly restricted approach, where probabilistic events can be refined as long as the abstract action remained part of the concrete event. This can be syntactically checked by the tool support.