Development of Rabin Choice Coordination in Event-B [⋆]

Emre Yilmaz and Thai Son Hoang

Deparment of Computer Science,
Swiss Federal Institute of Technology Zurich (ETH-Zurich),
CH-8092, Zurich, Switzerland
yilmaze@student.ethz.ch, htson@inf.ethz.ch

Abstract. We continue the investigation with the integration of qualitative probabilistic reasoning into Event-B further towards the direction of having a tool support. In passing by, we formalise a non-trivial algorithm, namely Rabin's choice coordination. Our correctness reasoning is a combination of termination proofs in terms of probabilistic convergence and standard invariants techniques. Moreover, we discuss how qualitative probabilistic reasoning can be maintained during refinement.
Keywords: Event-B, qualitative reasoning, probabilistic termination, tool support, Rabin's choice coordination.

1 Introduction

In some systems, termination cannot be guaranteed for certain. Instead a slightly weaker property is mostly sufficient and appropriate: termination *with probability one*. An example having such a property is when tossing a fair coin, eventually heads will come up. In other words, the coin will turn up heads with probability one. There are many applications in distributed systems of such a "coin flip" and in particular for symmetry-breaking protocols [8,10].

This kind of qualitative probability reasoning has been integrated into Event-B [5]. Beside the standard non-deterministic actions in Event-B, a new kind of actions is added, namely, *probabilistic* actions where the probability for each possible alternative being neither 0 nor 1 (i.e. "proper" [9]). Most of the time, actions of this type behave identically to the non-deterministic actions, except when reasoning about termination: they are interpreted *angelically* (as opposed to *demonic* nondeterminism). The result is a practical method for handling qualitative reasoning that generates only proof obligations in standard first-order logic of Event-B, in particular, the exact probability for different alternatives can be left unspecified.

We continue our research to realise a tool support for this extension to Event-B. In the process, we formalise a non-trivial algorithm, namely, Rabin's choice coordination [10]. The reasoning on probabilistic termination of the algorithm is non-trivial, involving a lexicographic variant which needs to be carefully formalised and mechanically proved to have adequate assurance about the correctness of the algorithm. The

⋆ Part of this research was carried out within the European Commission ICT project 214158 DEPLOY (http://www.deploy-project.eu/index.html). We thank Jean-Raymond Abrial, Andreas Fürst and Stefan Hallerstede for their comments on drafts of this paper.

case study acts as an illustration for the scalability of the approach for reasoning qualitatively in Event-B: it can be applied to more complex systems than just "coin tossing" examples.

Our development comprises several refinements and includes reasoning about both standard and probabilistic termination, and deadlock-freedom. Our approach is to first establish the model of the system without any termination arguments, then having several refinement layers dedicated to proving convergence properties of events according to a lexicographic variant. Essentially, with this style of development, our probabilistic termination arguments are preserved with refinement.

Our contribution hence is a methodology for proving almost certain-termination, with the main novelty is the restrictions on refinement and additional condition on variants so that probabilistic termination property can be establish. We use the Rabin's Choice Coordination to illustrate our approach and extend the RODIN Platform accordingly in order to support our reasoning.

The rest of the paper is structured as follows. In Section 2 we give a brief overview of the Event-B modelling method, focusing on proofs of convergence and qualitative reasoning. Section 3 is dedicated to the formalisation of Rabin's choice coordination algorithm. We present the summary of our tool support in Section 4. Finally, we draw some conclusions in Section 5.

2 Qualitative Reasoning in Event-B

Event-B [1] is a modelling method for formalising and developing systems whose components can be modeled as discrete transition systems. We will not describe in detail the semantics of Event-B here, instead just describe some of the proof obligations that are important for our development.

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* and *axioms*. Carrier sets are similar to types. Axioms constrain carrier sets and constants.

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

2.1 Machines

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

Events An event can be represented by the term "**any** t **where** $G(t, v)$ **then** $S(t, v)$ **end**", 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 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 "**when** $G(v)$ **then** $S(v)$ **end**" when the event does not have any parameters, and we write "**begin** $S(v)$ **end**" when, in addition, the event's guard equals *true*. A dedicated event of the last form is used for the *initialisation* event (usually respresented as init).

The action of an event is composed of one or more *assignments* of the form "$x := E(t, v)$" or "$x :\in E(t, v)$" or "$x :| Q(t, v, x')$", 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 the action must be disjoint. The last form refers to Q which is a *before-after predicate* relating the values x (before the action) and x' (afterwards). 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 $\boldsymbol{S}(t, v, v')$ 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 have their specified properties. We describe below the proof obligation for invariant preservation. Formal definitions of all proof obligations are given in [1]. *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 $\boldsymbol{S}(t, v, v')$.

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.

2.2 Machine Refinement

Machine refinement is a mechanism for introducing details about the dynamic properties of a model [1]. For more details on the theory of refinement, we refer the reader to the Action System formalism [4], which has inspired the development of Event-B. Here we sketch some central proof obligations for machine refinement which are related to our development in Section 3.

A machine CM can 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). For simplicity, we assume that both events have the same parameters t. Let the abstract event ea and concrete event ec be as follows.

$$\mathsf{ea} \;\widehat{=}\; \textbf{any } t \textbf{ where } G(t,v) \textbf{ then } S(t,v) \textbf{ end} \quad (1)$$
$$\mathsf{ec} \;\widehat{=}\; \textbf{any } t \textbf{ where } H(t,w) \textbf{ then } T(t,w) \textbf{ end} \quad (2)$$

Somewhat simplifying, we can say that ec refines ea if the guard of ec is stronger than the guard of ea, and the gluing invariants $J(v,w)$ establish a simulation of ec by ea.

$$I(v), J(v,w), H(t,w) \;\vdash\; G(t,v) \qquad \textbf{(GRD)}$$

$$I(v), J(v,w), H(t,w), \boldsymbol{T}(t,w,w') \;\vdash\; \exists v' \cdot \boldsymbol{S}(t,v,v') \land J(v',w') \qquad \textbf{(SIM)}$$

A special case of refinement (called superposition refinement) is when v is kept in the refinement, i.e. $v \subseteq w$. In particular, if the actions are deterministic for both abstract and concrete events, and the expressions assigned to v are equivalent, the proof obligation **SIM** reduces to just proving the gluing invariants $J(v',w')$ being re-established. Our reasoning in the later sections will often use this fact. 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.

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 ae is *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 *merged* together.

2.3 Convergence and Qualitative Reasoning

At any stage, it may be proved that some set of events *do not collectively diverge* (we call them *convergent* events). In other words, these convergent events cannot take control forever and hence one of the other events eventually occurs. 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. Since the variant maps a state to a *finite* set, V induces a well-founded ordering on system states given by strict subset-inclusion of their images under V. The corresponding proof obligation is as follows.

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

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 "$<$" [1]. Later we are going to use both types of variant for our development. Note that in some cases the convergence of some events cannot be immediately shown, but only in a later refinement. In this case, their convergence is *anticipated* and we must prove that $V(v') \subseteq V(v)$, that is, these anticipated events do not enlarge the variant. The convergent attribute of an event is denoted by the keyword **status** with three possible values: *convergent*, *anticipated*, and *ordinary* (for events which are not convergent). Effectively, the use of anticipated events allows us to construct a lexicographic variant relying on the fact that standard convergent property preserved by refinement.

In some cases, termination is not definite but almost certain, i.e. the probability of termination 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 as in [5]. 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 nondeterministic action. However, it behaves angelically with respect to **VAR**: an event with probabilistic action *may* (as in contrast to *must*) decrease the variant $V(v)$. The new proof obligation rule for probabilistic events is as follows.

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

Note that the rule that we showed here is for an abstract convergent event. For a concrete event, the corresponding proof obligation rule is similar with the exception that one can assume both abstract and gluing invariants hold.

Even though probabilistic convergent events can increase the variant $V(v)$, it is required that $V(v)$ is bounded above [5]. The upper bound B is a constant[1] and the proof obligation **BND**, which needs to be discharged for all anticipated events and convergent events (both standard and probabilistic), is as follows.

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

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

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

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

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

2.4 Our Contribution

The earlier work in [5] does not address the refinement of the probabilistic events. Whereas standard convergent argument is preserved by refinement, probabilistic convergent argument is not maintained since a "good" choice for termination could be accidentally removed. Forbidding refinement all together after proving probabilistic convergent is not an option for us, since we want to construct a lexicographic variant using refinement. As a result we restrict our refinement such that *the event and variable system must stay same after proving probabilistic convergence*. The only allowed modifications are additional invariants. Note that event splitting by having additional guards, e.g. in Section 3.2 and event merging satisfy this condition, i.e. they preserve the probabilistic convergent proofs. This is also the key aspect of our approach for proving probabilistic termination of algorithm, with some additional features as follows.

[1] In general, this could be a non-decreasing function on the state.

- To prove that eventually the algorithm establishes certain conditions, we follow the approach in [7] for reasoning about liveness properties, with the correctness argument combining appropriate proofs of event convergence (both standard and probabilistic) and deadlock freedom. More details on this approach is in Section 3.2.
- We first establish the full algorithm with several *anticipated* events, before converting them to *convergent*, either *standard* or *probabilistic* (taking into account the above restriction on the refinement). The use of anticipated events is first suggested in [5].
- Finally, with the use of anticipated events in early refinement and later converting them to either convergent where some of them probabilistically, we prove that the set of events terminates *probabilistically*. For this reason we need to prove that the combining lexicographic variant is bounded above. As a result, we require that not only the variant concerning with the probabilistic events, but *all other variants need to be bounded above* as well.

We have used the *RODIN Platform* [2] for our formal development. This is an industrial-strength 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 probabilistic convergent events and generating appropriate proof obligations. The new obligations are still in first-order logic hence we can reuse the proving support of the RODIN Platform without needing any additional extension. More detailed discussions on the tool support are in Section 4.

3 Rabin's Choice Coordination Algorithm

Rabin's choice coordination algorithm as explained in [10] is an example of the use of 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 at the end. 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 1. Our second contribution is the formalisation of the algorithm in Event-B and the proofs of the associated obligations using the RODIN Platform.

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 et. al. [9]. 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 1 Each tourist can be in one of the following locations: *inside-left*, *inside-right*, *outside-left*, *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 2 A tourist can move between the two outside locations.
ENV 3 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 cannot change his location anymore.

ENV 4 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 5 Eventually, all tourists enter the same place.

Rabin's choice coordination algorithm as described by Morgan et. al. in [9] 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.

ALG 6 Each tourist has a notepad on which he can write a number.
ALG 7 There are noticeboards at the outside-left and outside-right.

In the beginning, number 0 is written on all tourist notepads and on the two noticeboards. Initially, each tourist independently chooses 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 8 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 noticeboard.
  - 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[2] of $N + 2$. Then, he writes N' on the noticeboard 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 et. al. [9] accordingly in the next section. Note that we make an assumption about the tourist capability: he/she from an outside location can "look" inside of the same place (he still cannot see the other place, either inside or outside). A more realistic implementation as described in [9] is to have a the first tourist entering an inside location to write some special note e.g. "Here", on the noticeboard. However, this will complicate our reasoning hence we make this simplification.

[2] 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.

3.2 Formal Development

In this section, we present the formal development of Rabin's choice coordination algorithm in Event-B[3].

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 lin and rin, representing those at the inside-left and inside-right accordingly. Note that invariant **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.

```
variables:  $lin, rin$
```

```
invariants:
  inv0_1 :  $lin \subseteq T$
  inv0_2 :  $rin \subseteq T$
  inv0_3 :  $lin = \varnothing \lor rin = \varnothing$
```

```
init
  begin
    $lin := \varnothing$
    $rin := \varnothing$
  end
```

We have two events L_IN and R_IN to model the situation when a tourist enters the inside-left or inside-right accordingly (**ENV 3**). Moreover there are no leaving events: a tourist once inside cannot change his location (**ENV 4**).

```
L_IN
  status  $convergent$
  any  $t$  where
    $rin = \varnothing$
    $t \notin lin$
  then
    $lin := lin \cup \{t\}$
  end
```

```
R_IN
  status  $convergent$
  any  $t$  where
    $lin = \varnothing$
    $t \notin rin$
  then
    $rin := rin \cup \{t\}$
  end
```

```
variant:  $T \setminus (lin \cup rin)$
```

The two events are *convergent*, with the variant V_0 representing the set of tourists not inside the two places. Note that the variant V_0 here is bounded above by the set of tourists T which is finite.

Finally, we have one *ordinary* event, namely final. This is an *observer* event (similar to those defined in [7]) in the sense that this does not change the state of model, but to *observe certain condition* about the state of the model. The observing condition is encoded as the guard of the events: if the event is enabled, the condition is satisfied. Here we are interested in the fact that all tourists will end up in the same place. Note that according to invariant **inv0_3**, if all tourists are in one place, the other place must be empty.

$$\mathsf{final} \ \widehat{=}\ \textbf{when}\ rin = T \lor lin = T\ \textbf{then}\ \textsc{skip}\ \textbf{end}$$

Further refinements keep event final unchanged and our goal is to prove that eventually event final is enabled. At the end of the development, beside event final we have a number of events $e_1, \ldots, e_n$. We will prove that all events $e_1, \ldots, e_n$ are convergent (standard or probabilistically). We must prove that the event system containing $e_1, \ldots, e_n$ and final are deadlock-free. According to the convergence argument, all events $e_1, \ldots, e_n$ will eventually converge, i.e. these events will be disabled. Together with the deadlock-freedom argument, the only event that does not deadlock is final whose guard must be satisfied when all other events are disabled.

[3] The archive of the development can be found on-line at `http://deploy-eprints.ecs.soton.ac.uk/232/`.

Refinement 1. The Sets of Outside Tourists There are two new variables $lout$ and $rout$ representing the tourists outside the two places. Invariant **inv1_1** states that a tourist cannot be at two locations at the same time, and each tourist must be in one of the locations[4]. This corresponds to requirement **ENV 1**. Initially, some tourists decide to go to the outside-left and some tourists to the outside-right.

```
variables:   ..., $lout$, $rout$
```

```
invariants:
  inv1_1 :   $partition(T, lin, rin, lout, rout)$
```

```
init
  begin
    ...
    $lout, rout :\mid lout' = T \setminus rout'$
  end
```

There are two new events namely L_2_R and R_2_L to model the movement of a tourist between the two outside locations. This corresponds to the requirement **ENV 2**.

```
L_2_R
  status   anticipated
  any   $t$   where
    $t \in lout$
    $lin = \varnothing$
  then
    $rout, lout := rout \cup \{t\}, lout \setminus \{t\}$
  end
```

```
R_2_L
  status   anticipated
  any   $t$   where
    $t \in rout$
    $rin = \varnothing$
  then
    $lout, rout := lout \cup \{t\}, rout \setminus \{t\}$
  end
```

The guards $lin = \varnothing$ and $rin = \varnothing$ 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 requirement **ALG 8**. The two new events only modify new variables $rout$, $lout$ hence clearly refine SKIP. Moreover, invariant **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 *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 $LEFT$ and $RIGHT$ are symmetric, from now on, we present only events corresponding to $LEFT$. The refinement of event L_IN is as follows.

```
(abstract_)L_IN
  any   $t$   where
    $rin = \varnothing$
    $\underline{t \notin lin}$
  then
    $lin := lin \cup \{t\}$
  end
```

```
(concrete_)L_IN
  any   $t$   where
    $rin = \varnothing$
    $\underline{t \in lout}$
  then
    $lin, lout := lin \cup \{t\}, lout \setminus \{t\}$
  end
```

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**). The assigned expressions to old variable lin are the same in both abstract and concrete events. Moreover invariant **inv1_1** is maintained since the event merely moves a tourist from the outside-left to the inside-left.

[4] $partition(S, s_1, \ldots, s_n)$ means that subsets $s_1, \ldots, s_n$ are pairwise disjoint and their union is S.

Refinement 2. Rabin's Algorithm We introduce the two noticeboards outside the places and the tourists' notepads where they can write some number on it. Initially, number 0 is written on the noticeboards and all the notepads. This corresponds to the requirements **ALG 6** and **ALG 7**.

```
variables:  ..., $L, R, np$
```

```
invariants:
  inv2_1 :  $L \in \mathbb{N}$
  inv2_2 :  $R \in \mathbb{N}$
  inv2_3 :  $np \in T \to \mathbb{N}$
```

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

We can now specify under which condition a tourist can move from one location to another.

```
L_IN
  any  $t$  where
    ...
    $L < np(t) \lor lin \neq \varnothing$
  then
    ...
  end
```

Events modelling the movement of a tourist from an outside location to an inside location, i.e. event L_IN (and similarly R_IN), are guard-strengthened as follows. The guard $L < np(t) \lor lin \neq \varnothing$ 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-noticeboard 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 the $LEFT$ to $RIGHT$ are modelled by two events L_2_R_EQ and L_2_R_NEQ depending on if the number on the tourist notepad is equal or strictly smaller than the number on the noticeboard. Using $\overline{n}$ for the conjugate number of n, the two events are as follows.

```
L_2_R_NEQ
  refines  L_2_R
  status  $anticipated$
  any  $t$  where
    ...
    $np(t) < L$
  then
    ...
    $np(t) := L$
  end
```

```
L_2_R_EQ
  refines  L_2_R
  status  $anticipated$
  any  $t$  where
    ...
    $np(t) = L$
  then
    ...
    $L, np \;:\mid\; L' \in \{L + 2, \overline{L + 2}\} \land np' = np \lhd\!\!- \{t \mapsto L'\}$
  end
```

The actions of the above events update the tourist notepad and the noticeboard accordingly. Note that both events are refinements of the original event L_2_R, i.e. the original event is *split* into two cases. Note that these events model the movement of a tourist according to requirement **ALG 8**, with the exception that we use nondeterministic choice at the moment in L_2_R_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 the requirements except for **FUN 5**. 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 L_2_R_NEQ, L_2_R_EQ, R_2_L_NEQ and R_2_L_EQ. In this section, we describe our proof of (probabilistic) convergence of the anticipated events. We formalise the variant that has been proposed in [9]. 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: $(0, 1) \mid (2, 3) \mid (4, 5) \mid (6, 7) \mid \ldots$. 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 :** $\forall x \cdot x \in lout \Rightarrow np(x) \leq R$
- **inv3_2 :** $\forall x \cdot x \in rout \Rightarrow np(x) \leq L$
- **inv3_3 :** $\widetilde{L} - \widetilde{R} \in \{-2, 0, 2\}$
- **inv3_4 :** $\overline{L} \notin np[rout]$
- **inv3_5 :** $\overline{R} \notin np[lout]$

Refinement 3. Invariants **inv3_1–5** constraint the relationship between L and R. Below, we use the notation $\widetilde{n}$ to denote the minimum of n and its conjugate $\overline{n}$. We will not go into details about proving the preservation of these invariants, only give some brief descriptions of them. Invariant **inv3_1** states that every tourist at the outside-left carries a number no greater than the right-noticeboard. 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-noticeboard. 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 noticeboards cannot be "too far apart". Referring to the splitting of natural numbers into pairs, this invariant states that L and R must be in a same pair or in two adjacent pairs. Note that when $\widetilde{L} = \widetilde{R}$, i.e. they are in the same pair, there can be two cases, either $L = R$ or $L = \overline{R}$ (equivalently $R = \overline{L}$). 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 $L = R$. Our variant is based on this relationship.

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

axioms:
- rE_1 **:** $rE \in \mathbb{N} \times \mathbb{N} \nrightarrow \{0, 1, 2\}$
- rE_2 **:** $\forall l, r \cdot l \mapsto r \in \mathrm{dom}(rE) \Leftrightarrow \widetilde{l} - \widetilde{r} \in \{-2, 0, 2\}$
- rE_3 **:** $\forall l, r \cdot l \in \mathbb{N} \land l = r \Rightarrow rE(l \mapsto r) = 2$
- rE_4 **:** $\forall l, r \cdot l \in \mathbb{N} \land l = \overline{r} \Rightarrow rE(l \mapsto r) = 0$
- rE_5 **:** $\forall l, r \cdot l \in \mathbb{N} \land \widetilde{l} - \widetilde{r} \in \{-2, 2\} \Rightarrow rE(l \mapsto r) = 1$

variant: $rE(L \mapsto R)$

bound: 2

and define the variant V_1 as $rE(L \mapsto R)$ with upper bound of 2. We split event L_2_R_EQ into three different cases, depending on the current value of $rE(L \mapsto R)$.

```
L_2_R_EQ_0
  refines  L_2_R_EQ
  status  $convergent$
  any  $t$  where
    $t \in lout$
    $lin = \varnothing$
    $np(t) = L$
    $rE(L \mapsto R) = 0$
  then
    ...
  end
```

```
L_2_R_EQ_1
  refines  L_2_R_EQ
  status  $probabilistic$
  any  $t$  where
    $t \in lout$
    $lin = \varnothing$
    $np(t) = L$
    $rE(L \mapsto R) = 1$
  then
    ...
  end
```

```
L_2_R_EQ_2
  refines  L_2_R_EQ
  status  $convergent$
  any  $t$  where
    $t \in lout$
    $lin = \varnothing$
    $np(t) = L$
    $rE(L \mapsto R) = 2$
  then
    ...
  end
```

We prove that L_2_R_EQ_0, L_2_R_EQ_2 are *convergent*, and L_2_R_EQ_1 is *probabilistically convergent* whereas L_2_R_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 L_2_R_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 $t \in lout$ and according to invariant **inv3_5**, we have $\overline{R} \notin np[lout]$ which is a contradiction. In other words, the guard of L_2_R_EQ_0 can be used to derive $\perp$. Hence anything can be proved under the assumption of the guard of this events, including convergence proof.

For L_2_R_EQ_2, we have $rE(L \mapsto R) = 2$, i.e. $L = R$. The action will change L to either $L + 2$ or $\overline{L + 2}$, 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, hence satisfy **VAR**.

For L_2_R_NEQ, it does not change the value of L or R, hence the value of V_1 stays the same, i.e. non-increasing.

For L_2_R_EQ_1, firstly we have that the possible alternatives of the after states are finite (2 in this case) hence the event satisfies **FINACT**. Secondly, we prove that the event *may* decrease the variant V_1, i.e. satisfies **PRV**. The actual proof obligation (with some simplifications by removing unnecessary hypotheses) is as follows.

```
  $rE(L \mapsto R) = 1$
  $\forall x \cdot x \in lout \Rightarrow np(x) \leq R$
  $t \in lout$
  $np(t) = L$
$\vdash$
  $\exists L', np' \cdot L' \in \{L + 2, \overline{L + 2}\} \land np' = np \mathbin{\lhd\!\!\!-} \{t \mapsto L'\} \land rE(L' \mapsto R) < rE(L \mapsto R)$
```

We have from $rE(L \mapsto R) = 1$ that $\widetilde{L} - \widetilde{R} \in \{-2, 2\}$. In particular, from invariant **inv3_1**, i.e. $\forall x \cdot x \in lout \Rightarrow np(x) \leq R$, and from event's guards $t \in lout$ and $np(t) = L$, we have that $L \leq R$ hence $\widetilde{L} - \widetilde{R}$ must be -2. Referring to the splitting of natural numbers into pairs, when we have $\widetilde{L} - \widetilde{R} = -2$, 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 $\overline{L + 2}$ is to have L' to be in the same pair as R, hence one of the alternative will satisfy condition $L' = \overline{R}$. For this case, $rE(L' \mapsto R) = 0 < 1 = rE(L \mapsto R)$ As a result, we have proved that L_2_R_EQ_1 *may* decrease the variant V_1.

Inner layer The variant for the inner layer is used to prove the convergence property of events L_2_R_NEQ and R_2_L_NEQ. This is done in two refinement steps.

Refinement 5. We prove that L_2_R_NEQ converges and R_2_L_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 on strictly smaller than the left-noticeboard. Event L_2_R_NEQ changes the value of a tourist notepad from *strictly less* to *equal to* L hence it decreases V_2. Event R_2_L_NEQ increase the value of a tourist notepad, hence it *cannot increase* V_2.

Refinement 6. In the second step, we prove that R_2_L_NEQ converges with a symmetric variant V_3 to be $\{t \mid np(t) < R\}$ following similar reasoning as above.

Note that both variant V_2 and V_3 are bounded above by the finite set of tourists T.

Refinement 7. Deadlock-freedom

invariants:
- **inv7_1** : $\forall x \cdot x \in lin \Rightarrow np(x) \leq R$
- **inv7_2** : $\forall x \cdot x \in rin \Rightarrow np(x) \leq L$
- **inv7_3** : $lin \neq \varnothing \Rightarrow (\exists x \cdot x \in lin \land np(x) > L)$
- **inv7_3** : $rin \neq \varnothing \Rightarrow (\exists x \cdot x \in rin \land np(x) > R)$

In this final refinement, we merge the events that have been split earlier together, i.e. L_2_R_EQ and R_2_L_EQ. Combining the convergent attribute of the sub-events, we have now that these two events are probabilistic 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 requirement **FUN 5**. Our reasoning is based on the approach in [7] and is as follows. At the last model, we have the following events: event final which is *ordinary*, events L_IN, R_IN, L_2_R_NEQ, R_2_L_NEQ which are *convergent* and events L_2_R_EQ and R_2_L_EQ which are *probabilistically convergent*. Because of the convergent proof, we ensure that together the set of convergent events (standard and probabilistic) will terminate (being disabled) with probability 1. Moreover, because of the deadlock-freedom proof, when the convergent events are disabled, event final is the only one left, and must be enabled, i.e. all tourists are in the same place.

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

Table 1. Proof statistics

Proof Statistics The statistics for our proofs is in Table 1. 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 decided to split the events L_2_R_EQ and R_2_L_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 are dealing with arithmetic reasoning related to modulo operator (as consequent of the use of conjugate number), sometimes involves doing case distinctions which known to be difficult for automated provers.

4 Tool Support

We have implemented a plug-in to the RODIN Platform [2] for supporting the generation of proof obligations for proving probabilistic termination. The summary of the work done is as follows. More details can be seen in [11].

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 obligations are generated for proving probabilistic convergence property.

1. The variant is always bounded above by the declared bound. (**BND**)
2. The variant might 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**)

5 Conclusion and Future Work

In conclusion, we have presented a method for reasoning about termination with probability one using refinement as an extension of the work in [5]. We have developed Rabin's choice coordination algorithm [10] in Event-B. In particular, we have formalised the lexicographical variant as presented in [9]. We extended the RODIN Platform [2] for supporting the generation of appropriate proof obligations concerning with this type of reasoning, and proved all the obligations using the proving support of the RODIN Platform [2].

The example of Rabin's choice coordination is also used in [6, Chapter 3] as an illustrative example for reasoning about almost certain termination using classical B. 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.

Using our newly developed tool support, we have modelled other examples for proving termination including *contention resolution* [5] and *duelling cowboys* [6, Chapter 6]. In the near future, we will try to integrate the reasoning about contention resolution with the development of the Firewire protocol [3] and the full k-version of Rabin's Choice Coordination algorithm [10]. In particular for the latter example, the model of the algorithm will be quite straight-forward 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 challenges will be on finding the right lexicographic variant for proving probabilistic termination of the algorithm using our tool.

We have presented our development involving several refinements, which involves reasoning about both standard and probabilistic terminations and deadlock-freedom. However, we only use superposition refinement, in particular, when dealing with convergent proofs, we merely keep the models the same, and the various refinements are there to accommodate the lexicographic variant. For this reason, i.e. having the same model through out, our reasoning about probabilistic termination is preserved. This is a very strong assumption, and it could reduce the effectiveness of using refinement. However, in general, standard refinement does not preserve this type of reasoning: a valid standard refinement can accidentally remove the choice that lead to possible termination. The argument becomes more complicated with data refinement, i.e. when one replaces some abstract variables by some new concrete variables. In order to relax the restriction on having the same model through out, additional proof obligation(s) will be needed to guarantee that our reasoning at the abstract level about probabilistic convergence remains valid at the concrete level. We regard this as possible future work.

References

1. J-R. Abrial. *Modeling in Event-B: System and Software Engineering*. Cambridge University Press, May 2010.
2. J-R. Abrial, M. Butler, S. Hallerstede, T.S. Hoang, F. Mehta, and L. Voisin. RODIN: An open toolset for modelling and reasoning in Event-B. *Internation Journal on Software Tools for Technology Transfer (STTT)*, April 2010.
3. J-R. Abrial, D. Cansell, and D. Méry. A mechanically proved and incremental development of ieee 1394 tree identify protocol. *Formal Asp. Comput.*, 14(3):215–227, 2003.

4. R-J. Back. Refinement Calculus II: Parallel and Reactive Programs. In J. W. deBakker, W. P. deRoever, and G. Rozenberg, editors, *Stepwise Refinement of Distributed Systems*, volume 430 of *LNCS*, pages 67–93, Mook, The Netherlands, May 1989. Springer-Verlag.
5. S. Hallerstede and T.S. Hoang. Qualitative Probabilistic Modelling in Event-B. In Jim David and Jeremy Gibbons, editors, *IFM 2007: Integrated Formal Methods*, volume 4591 of *LNCS*, pages 293–312, Oxford, U.K., July 2007. Springer Verlag.
6. T.S. Hoang. *The Development of a Probabilistic B-Method and a Supporting Toolkit*. PhD thesis, The University of New South Wales, July 2005.
7. T.S. Hoang, H. Kuruma, D. Basin, and J-R. Abrial. Developing topology discovery in event-b. *Sci. Comput. Program.*, 74(11-12):879–899, 2009.
8. IEEE. *IEEE Std 1394a-2000 High Performance Serial Bus – Amendment 1*, 2000.
9. C. Morgan and A. McIver. *Abstraction, Refinement and Proof for Probabilistic Systems*. Springer Verlag, 2005.
10. M. Rabin. The choice coordination problem. *Acta Informatica, 17:121-134*, 1982.
11. E. Yilmaz. Tool support for qualitative reasoning in Event-B. Master's thesis, Department of Computer Science, ETH Zurich, Switzerland, August 2010.