Qualitative Reasoning for the Dining Philosophers [⋆]
— *Extended Abstract* —

Stefan Hallerstede[1] and Thai Son Hoang[2]

[1] Institut für Informatik
Heinrich-Heine-Universität Düsseldorf
halstefa@cs.uni-duesseldorf.de
[2] Department of Computer Science
Swiss Federal Institute of Technology Zürich (ETH Zürich)
htson@inf.ethz.ch

Abstract. We continue our investigation of qualitative probabilistic reasoning in Event-B. In the past we have applied it protocol verification, in particular, the Firewire protocol. There is still some way to go to achieve a practical method for qualitative probabilistic reasoning, especially concerning with refinement. We describe here our attempt for a probabilistic solution to the dining philosophers problem, in order to move further towards such a method.
Keywords: Event-B, probability, qualitative reasoning, refinement.

1 Overview

Our motivation here is construct a proof for the probabilistic solution for the Dining Philosophers problem. The proof from McIver and Morgan [5] uses both fairness assumption and probabilistic arguments. We attempt here to reason using only qualitative reasoning, hence the fairness assumption is replaced by a probabilistic reasoning. Moreover, we want to construct a practical method for reasoning about this kind of system which should be simple to use. We now give an overview of some important properties of the Event-B method that we are going to use, in particular about different technique for proving convergent of events.

1.1 The Event-B Modelling Method

A development in Event-B [2] is a set of formal models. The models are built from expressions in a mathematical language, which are stored in a repository. 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*. *Machines* specify behavioural properties of Event-B models. Machines may contain *variables*, *invariants*, *theorems*, *events*, and *variants*. Variables v define the state of a machine. They are constrained by invariants $I(v)$. Possible state changes are described by events. Each event is composed of a *guard* $G(t, v)$ (the conjunction of one or more predicates) and

[⋆] Part of this research was carried out within the European Commission ICT project 214158 DEPLOY (http://www.deploy-project.eu/index.html).

an *action* $S(t,v)$, where t are the *parameters* of the event. The guard states the necessary condition under which an event may occur, and the action describes how the state variables evolve when the event occurs. An event can be represented by the term "**any** t **where** $G(t,v)$ **then** $S(t,v)$ **end**". 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 *initialisation*.

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

$$x \ := \ E(t,v) \tag{1}$$

$$x \ :\in \ E(t,v) \tag{2}$$

$$x \ :| \ Q(t,v,x') \quad , \tag{3}$$

where x is a variable contained in v, $E(t,v)$ is an expression, and $Q(t,v,x')$ is a predicate. Assignments of the form (1) are *deterministic*, whereas the other two forms are *nondeterministic*. In (2), x (which must be a single variable) is assigned an element of a set. In (3), Q is a "before-after predicate", which relates the values x (before the action) and x' (afterwards). (3) is the most general form of assignment and nondeterministically selects an after-state x' satisfying Q and assigns it to x. Variables other than x are unchanged by the above assignments. There is also a side condition on the action of an event: the variables on the left-hand side of the assignments contained in the action must be disjoint.

Proof obligations serve to verify certain properties of machines. Formal definitions of all proof obligations are given in [1]. For a machine, we must prove *invariant preservation* and *feasibility* of events. *Invariant preservation* states that invariants hold whenever variables change their values. *Feasibility* state that the action of an event must be feasible whenever the event is enable.

Machine refinement provides a means to introduce details about the dynamic properties of a model [2]. For more details on the theory of refinement, we refer to the Action System formalism [3], which has inspired the development of Event-B.

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

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

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

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

Somewhat simplified, we say that ec refines ea if the guard of ec is stronger than the guard of ea (*guard strengthening*), and the gluing invariant $J(v,w)$ establishes a simulation of ec by ea (*simulation*). Proving guard strengthening just amounts to proving an implication. For simulation, under the assumption of the invariants and of the concrete guard $H(u,w)$ we must show that it is possible to choose a value for the abstract parameter t such that the abstract guard holds and the gluing invariant $J(v,w)$ is re-established. The possible values for the abstract parameter are given as *witness* in ec with the keyword **with**. 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.

Moreover, it may be proved that events do not collectively diverge. In other words, the 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 w to a finite set. One then proves that each new event *strictly* decreases V.

Given an event

```
evt
  any  x  where
    G(x, v)
  then
    v :| S(x, v, v')
  end
```

ones has to prove

```
Invariants and axioms
G(x, v)
⊢
∀v'·S(x, v, v')⇒
    V(v') ⊂ V(v)
```

1.2 Qualitative Reasoning: Probabilistic Action

In our earlier work [4], we extend the Event-B with probabilistic action $v \oplus\!| \; S(v, v')$, and the notion of probabilistic (eventually) termination of events. This extension requires a slightly modification to the variant proof obligation: event *might* decrease the variant V.

Given an event

```
evt
  any  x  where
    G(x, v)
  then
    v ⊕| S(x, v, v')
  end
```

ones has to prove

```
Invariants and axioms
G(x, v)
⊢
∃v'·S(x, v, v')∧
    V(v') ⊂ V(v)
```

A great advantage of this approach is that the proof obligations still within first-order predicate logic hence we do not need to extend our proof system.

2 The Dining Philosophers

We summary the dining philosophers problem as follows:

- A number of philosophers sit at a round table.
- Between each adjacent pair of philosopher is a single fork.
- In order to eat, each philosopher need two forks on both sides.
- When hungry, a philosopher might want to pick up a fork, but this might already be taken by his neighbouring philosopher.
- There is a possibility of deadlock or livelock.

There are various solution for the problem, including some deterministic solutions, e.g. using a waiter to break symmetry. We consider here a symmetric probabilistic solution as described in [5]. The algorithm for each philosopher is summarised in Figure 1. The table describes possible state changes for a particular philosopher. The only probabilistic choice that the philosopher made is when deciding either to pick up the left fork first or the right fork first.

The proofs from [5] using the fairness assumption which prove the Pseudo loop on the left to terminate. By replacing the fairness assumption with a probabilistic one, we attempt to verify that the Pseudo loop on the right to terminate probabilistically.

```
Some philosophers are hungry;
while ”No philosopher is eating” do
  Schedule one of the philosopher fairly
end
```

```
Some philosophers are hungry;
while ”No philosopher is eating” do
  Schedule one of the philosopher probabilistically
end
```

Our initial model is as follows, where h, t and e represent the set of *hungry*, *thinking* and *eating* philosophers, respectively.

Action	Before	After
Becomes hungry	t	h
Decides	h	l $_{0.5}\oplus$ r
Picks first left fork (if left fork available)	l	L
Picks first right fork (if right fork available)	r	R
Eats by pick up right (if right fork available)	L	E
Drops left fork (if right fork not available)	L	h
Eats by pick up left (if left fork available)	R	E
Drops right fork (if left fork not available)	R	h
Thinks	E	t

t	thinking
h	hungry
l	left first
r	right first
L	hold left only
R	hold right only
E	eating
	(holding both fo

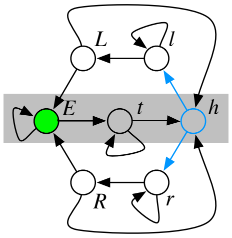


Fig. 1. Actions of a philosophers

```
variables:   $h, t, e$
```

```
invariants:
  $partition(P, h, t, e)$
```

```
init
  begin
    $h, t :| partition(P, h', t') \wedge h' \neq \varnothing$
    $e := \varnothing$
  end
```

```
eats
  any  $p$  where
    $p \in h$
  then
    $e := e \cup \{p\}$
    $h := h \setminus \{p\}$
  end
```

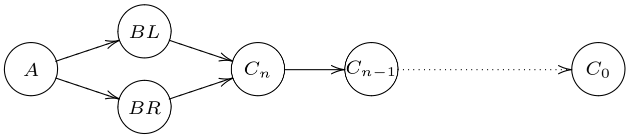
```
thinks
  any  $p$  where
    $p \in e$
  then
    $t := t \cup \{p\}$
    $e := e \setminus \{p\}$
  end
```

```
getsHungry
  any  $p$  where
    $p \in t$
  then
    $h := h \cup \{p\}$
    $t := t \setminus \{p\}$
  end
```

Our idea for developing the algorithm is as follows. Using refinement, we introduce the details of the approach with different set of philosophers, e.g. holding forks, picking forks. As a result, more events are introduced into the development. Our termination is established by the following arguments:

- Prove that events other than eats are (probabilistic) convergent.
- System is deadlock-free.

We formalise the lexicographic variant as presented in [5] in our Event-B development. The decrement of the variant can be split into different phases depending on the state of the system, where phase A is when no philosophers holding forks; phase BL (resp. BR) is when there are some philosophers holding left forks (resp. right forks); and phase C_i is when there are some philosophers holding left forks and some philosophers hold right forks. In particular, the convergent proof in phase BL and BR using probabilistic termination argument as mentioned earlier in Section 1.2.



We focus now on the probabilistic convergent argument in phases C_i. In particular, we have the events of the following forms:

```
chooseLeft
  any  $p$  where
    $p \in l$
    ...
  then
    ...
  end
```

```
dropLeft
  any  $p$  where
    $p \in L$
    ...
  then
    ...
  end
```

...

The difficulty here is that the probabilistic choice is associated with the parameter p for the philosophers. In particular, our reasoning must take into account all the actions that a philosopher can do. For our probabilistic termination argument, we have to prove the following: There exists a philosopher such that he can always act, and any action that he made decreases the variant. At the moment, our proof obligations cannot express the above condition, hence we need to extend the proof obligation rule.

```
evt_i
  any  $t$  where
    $G_i(t, v)$
  then
    $v :| Q_i(t, v, v')$
  end
```

```
variant:  $V$
```

```
witness:  $W(t, v)$
```

- Sketch of probabilistic termination witness for t, say $W(t, v)$.
- Sketch of the proof obligations.
  1. Existent of witness: $I(v) \Rightarrow (\exists t \cdot W(t, v))$.
  2. Given the witness, at least one probabilistic event is enable.
     $I(v) \wedge W(t, v) \Rightarrow G_1(t, v) \vee \ldots \vee G_n(t, v)$
  3. For any probabilistic event evt_i, it decreases the variant V: $I(v) \wedge W(t, v) \wedge G_i(t, v) \wedge Q_i(t, v, v') \Rightarrow V(v') \subset V(v)$

3 Conclusions

We sketch here an extension to our qualitative reasoning for proving probabilistic termination of an algorithm for the dining philosophers problem. Our proposed extension includes a new interpretation for probabilistic choice between events' parameters and new proof obligations which are practical for having tool support.

Moreover, for future work, because of the fact that refinement can reduce nondeterminism, qualitative termination is not necessary preserved through refinement in general. We need to have additional proof obligation(s) for preserving qualitative termination, however, any approach should be simple and usable.

References

1. J-R. Abrial. *Modeling in Event-B: System and Software Design*. CUP, 2009. To appear.
2. J-R. Abrial and S. Hallerstede. Refinement, Decomposition and Instantiation of Discrete Models: Application to Event-B. *Fundamentae Informatica*, 2006.
3. R-J Back. Refinement Calculus II: Parallel and Reactive Programs. In *Stepwise Refinement of Distributed Systems*, 1989.
4. S. Hallerstede and T.S. Hoang. Qualitative probabilistic modelling in event-b. In J. Davies and J. Gibbons, editors, *IFM*, volume 4591 of *LNCS*, pages 293–312. Springer, 2007.
5. A. McIver and C. Morgan. *Abstraction, Refinement and Proof for Probabilistic Systems*. Springer, 2005.