Probabilistic Invariants for Probabilistic Machines

Thai Son Hoang

Department of Computer Science
Swiss Federal Institute of Technology Zürich (ETH Zürich)

(Joint work with Zhendong Jin, Ken Robinson, Annabelle McIver and Carroll Morgan)

Formal Method Club, 16th November 2005, Manchester

Outline

1. Motivation
   - Extension to the B-Method
   - Background
2. Our Results/Contribution
   - Library Example
   - The **expectations** Clause
   - Standard and Probabilistic Invariant: the Difference

Outline

1. Motivation
   - Extension to the B-Method
   - Background
2. Our Results/Contribution
   - Library Example
   - The **expectations** Clause
   - Standard and Probabilistic Invariant: the Difference

Extending the B-Method

- To extend the scope of the *B-Method* (*B*) for probabilistic machines;
- To introduce the probabilistic choice substitution;
- To introduce the concept of probabilistic invariant (here called *expectation*);
- To establish the corresponding *probabilistic Abstract Machine Notation* (*pAMN*) for the new constructs;
- To establish the proof rules for the new constructs;

Extending the B-Method

- To extend the scope of the *B-Method* (*B*) for probabilistic machines;
- To introduce the probabilistic choice substitution;
- To introduce the concept of probabilistic invariant (here called *expectation*);
- To establish the corresponding *probabilistic Abstract Machine Notation* (*pAMN*) for the new constructs;
- To establish the proof rules for the new constructs;

Extending the B-Method

- To extend the scope of the *B-Method* (*B*) for probabilistic machines;
- To introduce the probabilistic choice substitution;
- To introduce the concept of probabilistic invariant (here called *expectation*);
- To establish the corresponding *probabilistic Abstract Machine Notation (pAMN)* for the new constructs;
- To establish the proof rules for the new constructs;

Extending the B-Method

- To extend the scope of the *B-Method* (*B*) for probabilistic machines;
- To introduce the probabilistic choice substitution;
- To introduce the concept of probabilistic invariant (here called *expectation*);
- To establish the corresponding *probabilistic Abstract Machine Notation (pAMN)* for the new constructs;
- To establish the proof rules for the new constructs;

Extending the B-Method

- To extend the scope of the *B-Method* (*B*) for probabilistic machines;
- To introduce the probabilistic choice substitution;
- To introduce the concept of probabilistic invariant (here called *expectation*);
- To establish the corresponding *probabilistic Abstract Machine Notation (pAMN)* for the new constructs;
- To establish the proof rules for the new constructs;

Outline

1. Motivation
   - Extension to the B-Method
   - Background

2. Our Results/Contribution
   - Library Example
   - The **expectations** Clause
   - Standard and Probabilistic Invariant: the Difference

The B-Method

- *Abstract machines*.
- *Variables*, e.g. x, y.
- *Invariant*, e.g. $x \in \mathbb{N} \ \wedge \ y \in \mathbb{N} \ \wedge \ x \leq y$.
- *Operations*, e.g.

  IncX $\widehat{=}$

  pre $x < y$ **then**

  $x := x + 1$

  end
- *Maintaining* the invariant.

The B-Method

- *Abstract machines*.
- *Variables*, e.g. x, y.
- *Invariant*, e.g. $x \in \mathbb{N} \ \wedge \ y \in \mathbb{N} \ \wedge \ x \leq y$.
- *Operations*, e.g.

  IncX $\widehat{=}$

  pre $x < y$ **then**

  $x := x + 1$

  end
- *Maintaining* the invariant.

The B-Method

- *Abstract machines*.
- *Variables*, e.g. x, y.
- *Invariant*, e.g. $x \in \mathbb{N} \ \wedge \ y \in \mathbb{N} \ \wedge \ x \leq y$.
- *Operations*, e.g.

  IncX $\widehat{=}$

  pre $x < y$ **then**

  $x := x + 1$

  end
- *Maintaining* the invariant.

The B-Method

- *Abstract machines*.
- *Variables*, e.g. x, y.
- *Invariant*, e.g. $x \in \mathbb{N} \ \wedge \ y \in \mathbb{N} \ \wedge \ x \leq y$.
- *Operations*, e.g.

  IncX $\widehat{=}$

  pre $x < y$ **then**

  $x := x + 1$

  end
- *Maintaining* the invariant.

The B-Method

- *Abstract machines*.
- *Variables*, e.g. x, y.
- *Invariant*, e.g. $x \in \mathbb{N} \ \wedge \ y \in \mathbb{N} \ \wedge \ x \leq y$.
- *Operations*, e.g.

  IncX $\widehat{=}$

  pre $x < y$ **then**

  $x := x + 1$

  end
- *Maintaining* the invariant.

The Generalised Substitution Language

The semantics of *B* machine is given by the *Generalised Substitution Language* (*GSL*) where substitutions are predicate transformers.

Summary

$[x := E]\,Q$	The predicate obtained after replacing all free occurrences of x in Q by E.
$[\mathrm{skip}]\,Q$	Q.
$[S \; [] \; T]\,Q$	$[S]\,Q \wedge [T]\,Q$.
$[P \mid S]\,Q$	$P \wedge [S]\,Q$.
$[P \Longrightarrow S]\,Q$	$P \Rightarrow [S]\,Q$.
$[@x \cdot S]\,Q$	$\forall x \cdot [S]\,Q$.

The Generalised Substitution Language

The semantics of *B* machine is given by the *Generalised Substitution Language* (*GSL*) where substitutions are predicate transformers.

Summary

$[x := E]\,Q$	The predicate obtained after replacing all free occurrences of x in Q by E.
$[\mathrm{skip}]\,Q$	Q .
$[S \talloblong T]\,Q$	$[S]\,Q \wedge [T]\,Q$.
$[P \mid S]\,Q$	$P \wedge [S]\,Q$.
$[P \Longrightarrow S]\,Q$	$P \Rightarrow [S]\,Q$.
$[@x \cdot S]\,Q$	$\forall x \cdot [S]\,Q$.

The Generalised Substitution Language

The semantics of *B* machine is given by the *Generalised Substitution Language* (*GSL*) where substitutions are predicate transformers.

Summary

$[x := E]\,Q$	The predicate obtained after replacing all free occurrences of x in Q by E.
$[\mathrm{skip}]\,Q$	Q .
$[S \,[\!]\, T]\,Q$	$[S]\,Q \wedge [T]\,Q$.
$[P \mid S]\,Q$	$P \wedge [S]\,Q$.
$[P \Longrightarrow S]\,Q$	$P \Rightarrow [S]\,Q$.
$[@x \cdot S]\,Q$	$\forall x \cdot [S]\,Q$.

The Generalised Substitution Language

The semantics of *B* machine is given by the *Generalised Substitution Language* (*GSL*) where substitutions are predicate transformers.

Summary

$[x := E]\,Q$	The predicate obtained after replacing all free occurrences of x in Q by E.
$[\mathrm{skip}]\,Q$	Q .
$[S \mathbin{[\!]} T]\,Q$	$[S]\,Q \wedge [T]\,Q$.
$[P \mid S]\,Q$	$P \wedge [S]\,Q$.
$[P \Longrightarrow S]\,Q$	$P \Rightarrow [S]\,Q$.
$[@x \cdot S]\,Q$	$\forall x \cdot [S]\,Q$.

The Generalised Substitution Language

The semantics of *B* machine is given by the *Generalised Substitution Language* (*GSL*) where substitutions are predicate transformers.

Summary

$[x := E]\,Q$	The predicate obtained after replacing all free occurrences of x in Q by E.
$[\mathrm{skip}]\,Q$	Q .
$[S \,[\!]\, T]\,Q$	$[S]\,Q \wedge [T]\,Q$.
$[P \mid S]\,Q$	$P \wedge [S]\,Q$.
$[P \Longrightarrow S]\,Q$	$P \Rightarrow [S]\,Q$.
$[@x \cdot S]\,Q$	$\forall x \cdot [S]\,Q$.

The Generalised Substitution Language

The semantics of *B* machine is given by the *Generalised Substitution Language* (*GSL*) where substitutions are predicate transformers.

Summary

$[x := E]\,Q$	The predicate obtained after replacing all free occurrences of x in Q by E.
$[\mathrm{skip}]\,Q$	Q .
$[S \,[\!]\, T]\,Q$	$[S]\,Q \wedge [T]\,Q$.
$[P \mid S]\,Q$	$P \wedge [S]\,Q$.
$[P \Longrightarrow S]\,Q$	$P \Rightarrow [S]\,Q$.
$[@x \cdot S]\,Q$	$\forall x \cdot [S]\,Q$.

The *probabilistic GSL*

How *probabilistic GSL* extends *Generalised Substitution Language*

1. Adding probabilistic choice substitution $S\ {}_p\oplus\ T$.
2. Substitutions act as expectation transformers.

Expectations replace predicates

Predicates (functions from state to Boolean) are widened to *Expectations* (functions from state to non-negative real).

- For consistency with Boolean logic, we use embedded predicates, $\langle \mathit{false} \rangle = 0$, and $\langle \mathit{true} \rangle = 1$.
- Generalised version of $\Rightarrow$: the notion of “everywhere no more than”: $B_1 \Rrightarrow B_2$.
- Notationally, we have kept predicates as much as possible.

The *probabilistic GSL*

How *probabilistic GSL* extends *Generalised Substitution Language*

1. Adding probabilistic choice substitution $S\ {}_p\!\oplus\ T$.
2. Substitutions act as expectation transformers.

Expectations replace predicates

Predicates (functions from state to Boolean) are widened to *Expectations* (functions from state to non-negative real).

- For consistency with Boolean logic, we use embedded predicates, $\langle \mathit{false} \rangle = 0$, and $\langle \mathit{true} \rangle = 1$.
- Generalised version of $\Rightarrow$: the notion of "everywhere no more than": $B_1 \Rrightarrow B_2$.
- Notationally, we have kept predicates as much as possible.

The *probabilistic GSL*

How *probabilistic GSL* extends *Generalised Substitution Language*

1. Adding probabilistic choice substitution $S \;_p\oplus\; T$.
2. Substitutions act as expectation transformers.

Expectations replace predicates

Predicates (functions from state to Boolean) are widened to *Expectations* (functions from state to non-negative real).

- For consistency with Boolean logic, we use embedded predicates, $\langle \mathit{false} \rangle = 0$, and $\langle \mathit{true} \rangle = 1$.
- Generalised version of $\Rightarrow$: the notion of "everywhere no more than": $B_1 \Rrightarrow B_2$.
- Notationally, we have kept predicates as much as possible.

The *probabilistic GSL*

How *probabilistic GSL* extends *Generalised Substitution Language*

1. Adding probabilistic choice substitution $S \;_p\oplus\; T$.
2. Substitutions act as expectation transformers.

Expectations replace predicates

Predicates (functions from state to Boolean) are widened to *Expectations* (functions from state to non-negative real).

- For consistency with Boolean logic, we use embedded predicates, $\langle \mathit{false} \rangle = 0$, and $\langle \mathit{true} \rangle = 1$.
- Generalised version of $\Rightarrow$: the notion of “everywhere no more than”: $B_1 \Rrightarrow B_2$.
- Notationally, we have kept predicates as much as possible.

The *probabilistic GSL*

How *probabilistic GSL* extends *Generalised Substitution Language*

1. Adding probabilistic choice substitution $S\ {}_{p}\!\oplus\ T$.
2. Substitutions act as expectation transformers.

Expectations replace predicates

Predicates (functions from state to Boolean) are widened to *Expectations* (functions from state to non-negative real).

- For consistency with Boolean logic, we use embedded predicates, $\langle \mathit{false} \rangle = 0$, and $\langle \mathit{true} \rangle = 1$.
- Generalised version of $\Rightarrow$: the notion of "everywhere no more than": $B_1 \ \Rrightarrow \ B_2$.
- Notationally, we have kept predicates as much as possible.

The *probabilistic GSL*

How *probabilistic GSL* extends *Generalised Substitution Language*

1. Adding probabilistic choice substitution $S\ {}_{p}\oplus\ T$.
2. Substitutions act as expectation transformers.

Expectations replace predicates

Predicates (functions from state to Boolean) are widened to *Expectations* (functions from state to non-negative real).

- For consistency with Boolean logic, we use embedded predicates, $\langle \mathit{false} \rangle = 0$, and $\langle \mathit{true} \rangle = 1$.
- Generalised version of $\Rightarrow$: the notion of "everywhere no more than": $B_1 \Rrightarrow B_2$.
- Notationally, we have kept predicates as much as possible.

pGSL Syntax and Semantics

Summary

$[x := E]B$	The expectation obtained after replacing all free occurrences of x in B by E
$[\mathrm{skip}]B$	B
$[S\ {}_p\!\oplus\ T]B$	$p \times [S]B + (1-p) \times [T]B$
$[S \ [\!]\ T]B$	$[S]B \min [T]B$
$[@y \cdot P \implies S]B$	$\min (y) \cdot (P \mid [S]B)$

pGSL Syntax and Semantics

Summary

$[x := E]B$	The expectation obtained after replacing all free occurrences of x in B by E
$[\text{skip}]B$	B
$[S \;_p\!\oplus\; T]B$	$p \times [S]B + (1-p) \times [T]B$
$[S \; [] \; T]B$	$[S]B \text{ min } [T]B$
$[@y \cdot P \Longrightarrow S]B$	$\text{min } (y) \cdot (P \mid [S]B)$

pGSL Syntax and Semantics

Summary

$[x := E]B$	The expectation obtained after replacing all free occurrences of x in B by E
$[\text{skip}]B$	B
$[S\ {}_p\!\oplus T]B$	$p \times [S]B + (1-p) \times [T]B$
$[S \,[\!]\, T]B$	$[S]B \text{ min } [T]B$
$[@y \cdot P \Longrightarrow S]B$	$\text{min } (y) \cdot (P \mid [S]B)$

Examples

Example 1

$$\begin{array}{lll} & [x := y \;{}_{\frac{1}{3}}\oplus\; x := 2 \times y]x^2 & \\ \equiv & \frac{1}{3} \times [x := y]\, x^2 + \frac{2}{3} \times [x := 2 \times y]\, x^2 & \text{probabilistic choice} \\ \equiv & \frac{1}{3} \times y^2 + \frac{2}{3} \times (2 \times y)^2 & \text{simple subsitutions} \\ \equiv & 3 \times y^2 \,. & \text{arithmetic} \end{array}$$

Example 2

$$\begin{array}{lll} & [x := y \;{}_{\frac{1}{3}}\oplus\; x := 2 \times y]\, \langle x = 2 \rangle & \\ \equiv & & \text{probabilistic choice} \\ & \frac{1}{3} \times [x := y]\, \langle x = 2 \rangle + \frac{2}{3} \times [x := 2 \times y]\, \langle x = 2 \rangle & \\ \equiv & \frac{1}{3} \times \langle y = 2 \rangle + \frac{2}{3} \times \langle 2 \times y = 2 \rangle & \text{simple subsitutions} \\ \equiv & \frac{1}{3} \times \langle y = 2 \rangle + \frac{2}{3} \times \langle y = 1 \rangle \,. & \text{arithmetic} \end{array}$$

Examples

Example 1

$$
\begin{aligned}
& [x := y \;{}_{\frac{1}{3}}\!\oplus\; x := 2 \times y]x^2 & \\
\equiv\;& \tfrac{1}{3} \times [x := y]\, x^2 + \tfrac{2}{3} \times [x := 2 \times y]\, x^2 & \text{probabilistic choice} \\
\equiv\;& \tfrac{1}{3} \times y^2 + \tfrac{2}{3} \times (2 \times y)^2 & \text{simple subsitutions} \\
\equiv\;& 3 \times y^2 \,. & \text{arithmetic}
\end{aligned}
$$

Example 2

$$
\begin{aligned}
& [x := y \;{}_{\frac{1}{3}}\!\oplus\; x := 2 \times y]\, \langle x = 2 \rangle & \\
\equiv\;& & \text{probabilistic choice} \\
& \tfrac{1}{3} \times [x := y]\, \langle x = 2 \rangle + \tfrac{2}{3} \times [x := 2 \times y]\, \langle x = 2 \rangle & \\
\equiv\;& \tfrac{1}{3} \times \langle y = 2 \rangle + \tfrac{2}{3} \times \langle 2 \times y = 2 \rangle & \text{simple subsitutions} \\
\equiv\;& \tfrac{1}{3} \times \langle y = 2 \rangle + \tfrac{2}{3} \times \langle y = 1 \rangle \,. & \text{arithmetic}
\end{aligned}
$$

Outline

1. Motivation
   - Extension to the B-Method
   - Background
2. Our Results/Contribution
   - Library Example
   - The **expectations** Clause
   - Standard and Probabilistic Invariant: the Difference

Aims

We will take the well-known “library” example, and use that as a basis for developing a probabilistic version. Our aims are:

- To introduce and show how probabilistic invariants capture some probabilistic properties and;
- To highlight some of the unexpected and subtle issues that can arise.

Standard Library

machine *StandardLibrary* (*totalBooks*)

variables *booksInLibrary* , *loansStarted* , *loansEnded*

invariant

$booksInLibrary \in \mathbb{N} \land loansStarted \in \mathbb{N} \land loansEnded \in \mathbb{N} \land$
$loansEnded \leq loansStarted \land$
$booksInLibrary + loansStarted - loansEnded = totalBooks$

initialisation

$booksInLibrary := totalBooks \parallel loansStarted := 0 \parallel loansEnded := 0$

Standard Library

machine *StandardLibrary* (*totalBooks*)

variables *booksInLibrary* , *loansStarted* , *loansEnded*

invariant

$$booksInLibrary \in \mathbb{N} \land loansStarted \in \mathbb{N} \land loansEnded \in \mathbb{N} \land$$
$$loansEnded \leq loansStarted \land$$
$$booksInLibrary + loansStarted - loansEnded = totalBooks$$

initialisation

$$booksInLibrary := totalBooks \parallel loansStarted := 0 \parallel loansEnded := 0$$

Standard Library

machine *StandardLibrary* (*totalBooks*)

variables *booksInLibrary* , *loansStarted* , *loansEnded*

invariant

$$\mathit{booksInLibrary} \in \mathbb{N} \wedge \mathit{loansStarted} \in \mathbb{N} \wedge \mathit{loansEnded} \in \mathbb{N} \wedge$$
$$\mathit{loansEnded} \leq \mathit{loansStarted} \wedge$$
$$\mathit{booksInLibrary} + \mathit{loansStarted} - \mathit{loansEnded} = \mathit{totalBooks}$$

initialisation

$$\mathit{booksInLibrary} := \mathit{totalBooks} \parallel \mathit{loansStarted} := 0 \parallel \mathit{loansEnded} := 0$$

Standard Library

machine *StandardLibrary* (*totalBooks*)

variables *booksInLibrary* , *loansStarted* , *loansEnded*

invariant

$$booksInLibrary \in \mathbb{N} \land loansStarted \in \mathbb{N} \land loansEnded \in \mathbb{N} \land$$
$$loansEnded \leq loansStarted \land$$
$$booksInLibrary + loansStarted - loansEnded = totalBooks$$

initialisation

$$booksInLibrary := totalBooks \parallel loansStarted := 0 \parallel loansEnded := 0$$

Standard Library (cont.)

operations

StartLoan $\widehat{=}$

pre $booksInLibrary > 0$ **then**

$booksInLibrary := booksInLibrary - 1 \;\|$

$loansStarted := loansStarted + 1$

end ;

EndLoan $\widehat{=}$

pre $loansEnded < loansStarted$ **then**

$booksInLibrary := booksInLibrary + 1 \;\|$

$loansEnded := loansEnded + 1$

end

end

Probabilistic Library

Lose operation?

Arrange so that Lose is invoked, with some probability.

Lose $\widehat{=}$

pre *booksInLibrary* > 0 **then**

booksInLibrary $:=$ *booksInLibrary* $- 1$

end

Problem

The problem with this is that we have no way in *B* of modelling a probabilistically invoked operation.

Solution

An alternative, in *probabilistic B*, is to model operations with probabilistic *effects*.

Probabilistic Library

Lose operation?

Arrange so that Lose is invoked, with some probability.

Lose $\widehat{=}$

pre $\mathit{booksInLibrary} > 0$ **then**

$\mathit{booksInLibrary} := \mathit{booksInLibrary} - 1$

end

Problem

The problem with this is that we have no way in *B* of modelling a probabilistically invoked operation.

Solution

An alternative, in *probabilistic B*, is to model operations with probabilistic *effects*.

Probabilistic Library

Lose operation?

Arrange so that Lose is invoked, with some probability.

Lose $\widehat{=}$

pre *booksInLibrary* > 0 **then**

booksInLibrary $:=$ *booksInLibrary* $- 1$

end

Problem

The problem with this is that we have no way in *B* of modelling a probabilistically invoked operation.

Solution

An alternative, in *probabilistic B*, is to model operations with probabilistic *effects*.

The **pchoice** clause

The **pchoice** construct is the *probabilistic Abstract Machine Notation* counterpart of the operator ${}_p\oplus$, i.e.

pchoice p **of**
S
or
T
end

corresponds to $S \; {}_p\oplus \; T$.

Probabilistic **EndLoan** operation

EndLoan $\widehat{=}$

pre *loansEnded* $<$ *loansStarted* **then**

pchoice *pp* **of**

booksLost $:=$ *booksLost* $+ 1$

or

booksInLibrary $:=$ *booksInLibrary* $+ 1$

end $\|$

loansEnded $:=$ *loansEnded* $+ 1$

end

Outline

1. Motivation
   - Extension to the B-Method
   - Background
2. Our Results/Contribution
   - Library Example
   - The **expectations** Clause
   - Standard and Probabilistic Invariant: the Difference

Reconstruct the invariants

Standard invariant

$$\textit{booksInLibrary} + \textit{booksLost} + \textit{loansStarted} - \textit{loansEnded} = \textit{totalBooks}$$

Probabilistic invariant

EXPECTATIONS

$$0 \Rrightarrow pp \times \textit{loansEnded} - \textit{booksLost}$$

Reconstruct the invariants

Standard invariant

$$\mathit{booksInLibrary} + \mathit{booksLost} + \mathit{loansStarted} - \mathit{loansEnded} = \mathit{totalBooks}$$

Probabilistic invariant

EXPECTATIONS

$$0 \Rrightarrow \mathit{pp} \times \mathit{loansEnded} - \mathit{booksLost}$$

The **expectations** clause

Each predicate in the **expectations** clause defines a *real*-valued function from the state and the lower bound of that function. Each has the form:

$$e \Rrightarrow V \;, \tag{1}$$

where

- V is an expression over program variables,
- e is the lower bound that must be established by the initialisation.

If a standard invariant, I, was written as an expectation, we would write:

$$true \Rightarrow I \;, \tag{2}$$

but that is simply I, so nothing would appear to be achieved. We will see that there is significant difference for the probabilistic invariant. **Importantly**, although $e \Rrightarrow V$ is invariant, it is **not** used as a standard predicate invariant.

The **expectations** clause

Each predicate in the **expectations** clause defines a *real*-valued function from the state and the lower bound of that function. Each has the form:

$$e \Rrightarrow V \ , \tag{1}$$

where

- V is an expression over program variables,
- e is the lower bound that must be established by the initialisation.

If a standard invariant, I, was written as an expectation, we would write:

$$true \Rrightarrow I \ , \tag{2}$$

but that is simply I, so nothing would appear to be achieved. We will see that there is significant difference for the probabilistic invariant.

Importantly, although $e \Rrightarrow V$ is invariant, it is **not** used as a standard predicate invariant.

The **expectations** clause

Each predicate in the **expectations** clause defines a *real*-valued function from the state and the lower bound of that function. Each has the form:

$$e \Rrightarrow V \,, \tag{1}$$

where

- V is an expression over program variables,
- e is the lower bound that must be established by the initialisation.

If a standard invariant, I, was written as an expectation, we would write:

$$\mathit{true} \Rightarrow I \,, \tag{2}$$

but that is simply I, so nothing would appear to be achieved. We will see that there is significant difference for the probabilistic invariant. **Importantly**, although $e \Rrightarrow V$ is invariant, it is **not** used as a standard predicate invariant.

(Recall) Maintenance of standard invariant

We wish to interpret the conditions on initialisation and operations in the context of expections: $true \Rightarrow I$ for standard programs; and $e \Rrightarrow V$ for probabilistic programs.

Standard program:

$$\begin{aligned} true &\Rightarrow [\mathrm{Init}]\ I \\ I &\Rightarrow [\mathrm{OpX}]\ I \\ I &\Rightarrow [\mathrm{OpY}]\ I, \end{aligned} \tag{3}$$

then we are assured that

$$true \Rightarrow [\mathrm{Init}; \mathrm{Op?}; \mathrm{Op?}; \ldots; \mathrm{Op?}]\ I \tag{4}$$

(Recall) Maintenance of standard invariant

We wish to interpret the conditions on initialisation and operations in the context of expections: *true* $\Rightarrow$ I for standard programs; and $e \Rrightarrow V$ for probabilistic programs.

Standard program:

$$\begin{aligned} \textit{true} &\Rightarrow [\text{Init}]\, I \\ I &\Rightarrow [\text{OpX}]\, I \\ I &\Rightarrow [\text{OpY}]\, I, \end{aligned} \tag{3}$$

then we are assured that

$$\textit{true} \Rightarrow [\text{Init}; \text{Op?}; \text{Op?}; \ldots; \text{Op?}]\, I \tag{4}$$

What do **expectations** guarantee?

Probabilistic program:

$$
\begin{aligned}
e &\Rrightarrow [\mathrm{Init}]\ V \\
V &\Rrightarrow [\mathrm{OpX}]\ V \\
V &\Rrightarrow [\mathrm{OpY}]\ V,
\end{aligned}
\tag{5}
$$

then we are assured that

$$e \Rrightarrow [\mathrm{Init}; \mathrm{Op?}; \mathrm{Op?}; \ldots; \mathrm{Op?}]\ V \tag{6}$$

Proof obligations for probabilistic machines

Standard machines

N1: The initialisation needs to establish the invariant, i.e. $[Init]I$.

N2: The operations need to maintain the invariant, i.e. $I \Rightarrow [Op]I$.

Probabilistic machines

P1: The initialisation needs to establish the lower bound of the probabilistic invariant.

$$e \Rrightarrow [Init]V \ .$$

P2: The operations do not decrease the expected value of the probabilistic invariant, i.e. the expected value of the invariant after the operation is at least the expected value before the operation

$$V \Rrightarrow [Op]V \ .$$

Proof obligations for probabilistic machines

Standard machines

N1: The initialisation needs to establish the invariant, i.e. $[Init]I$.

N2: The operations need to maintain the invariant, i.e. $I \Rightarrow [Op]I$.

Probabilistic machines

P1: The initialisation needs to establish the lower bound of the probabilistic invariant.

$$e \Rrightarrow [Init]V \ .$$

P2: The operations do not decrease the expected value of the probabilistic invariant, i.e. the expected value of the invariant after the operation is at least the expected value before the operation

$$V \Rrightarrow [Op]V \ .$$

What the invariant means

For standard machines, the trace of the values of the expectations of the standard invariant is *true*, *true*, ..., *true*, and is not remarkable.

For probabilistic machines, the trace of the values of the expectations of the probabilistic invariant is $e_0, e_1, \ldots, e_n$, where $e_0 \Rrightarrow e_1 \Rrightarrow \ldots \Rrightarrow e_n$. That is,

> *the trace of expectations must form a monotonically increasing chain, no matter how the nondeterminism is resolved.*

For those interested in an experimental view here is another story.

> *Over a large number of tests of the machine, carried out by an adversary, who can choose to resolve demonic choice within operations any way they wish, and who can choose to invoke operations in any order, we will observe that the average value of V is at least the stated value.*

What the invariant means

For standard machines, the trace of the values of the expectations of the standard invariant is *true*, *true*, ..., *true*, and is not remarkable.

For probabilistic machines, the trace of the values of the expectations of the probabilistic invariant is $e_0, e_1, \ldots, e_n$, where $e_0 \Rrightarrow e_1 \Rrightarrow \ldots \Rrightarrow e_n$. That is,

> *the trace of expectations must form a monotonically increasing chain, no matter how the nondeterminism is resolved.*

For those interested in an experimental view here is another story.

> *Over a large number of tests of the machine, carried out by an adversary, who can choose to resolve demonic choice within operations any way they wish, and who can choose to invoke operations in any order, we will observe that the average value of V is at least the stated value.*

What the invariant means

For standard machines, the trace of the values of the expectations of the standard invariant is *true*, *true*, $\ldots$, *true*, and is not remarkable. For probabilistic machines, the trace of the values of the expectations of the probabilistic invariant is $e_0, e_1, \ldots, e_n$, where $e_0 \Rrightarrow e_1 \Rrightarrow \ldots \Rrightarrow e_n$. That is,

> *the trace of expectations must form a monotonically increasing chain, no matter how the nondeterminism is resolved.*

For those interested in an experimental view here is another story.

> *Over a large number of tests of the machine, carried out by an adversary, who can choose to resolve demonic choice within operations any way they wish, and who can choose to invoke operations in any order, we will observe that the average value of V is at least the stated value.*

Outline

1 Motivation
- Extension to the B-Method
- Background

2 Our Results/Contribution
- Library Example
- The **expectations** Clause
- Standard and Probabilistic Invariant: the Difference

StockTake operation

There are some consequences of our use of expectations that are surprising if the difference between Boolean and probabilistic invariants is not fully appreciated.

StockTake

$$
\begin{array}{l}
\mathit{totalCost} \longleftarrow \ \textbf{StockTake} \ \widehat{=} \\
\quad \textbf{begin} \\
\qquad \mathit{totalCost} := \mathit{cost} \times \mathit{booksLost} \ \| \\
\qquad \mathit{booksInLibrary} := \mathit{booksInLibrary} + \mathit{booksLost} \ \| \\
\qquad \mathit{loansStarted} := \mathit{loansStarted} - \mathit{loansEnded} \ \| \\
\qquad \mathit{loansEnded} := 0 \ \| \\
\qquad \mathit{booksLost} := 0 \\
\quad \textbf{end}
\end{array}
$$

StockTake operation breaks the probabilistic invariant

For the probabilistic invariant we require $V \Rrightarrow [StockTake]\,V$.
Consider the right-hand side of that inequality (considering the effect of variables *loansEnded* and *booksLost* only):

$$
\begin{array}{ll}
 & [StockTake]\,V \\
\equiv & [loansEnded, booksLost := 0, 0]\,V \\
\equiv & [loansEnded, booksLost := 0, 0] \\
 & (pp * loansEnded - booksLost) \\
\equiv & 0\ .
\end{array}
$$

This requires us to prove

$$
pp * loansEnded - booksLost \quad \Rrightarrow \quad 0\ , \tag{7}
$$

which we cannot prove in this context.

What went wrong?

The problem is a consequence of us naively carrying forward from standard machines the idea that initialisation is always applicable. With standard invariants the lower bound is *true*, which is also the upper bound.

It is not normally the case with probabilistic invariants that the lower bound is the upper bound. If it were then there would be no difference between standard and probabilistic machines.

Consider the following scenario. A malevolent library administrator wishes to show that library loan system is "broken": that the rate of book loss is higher than the advertised claim of pp. If the administrator adopts a policy of running StockTake whenever $booksLost$ is large relative to $pp * loansEnded$, then the library managers will indeed see that system is "broken".

What went wrong?

The problem is a consequence of us naively carrying forward from standard machines the idea that initialisation is always applicable. With standard invariants the lower bound is *true*, which is also the upper bound.

It is not normally the case with probabilistic invariants that the lower bound is the upper bound. If it were then there would be no difference between standard and probabilistic machines.

Consider the following scenario. A malevolent library administrator wishes to show that library loan system is “broken”: that the rate of book loss is higher than the advertised claim of pp. If the administrator adopts a policy of running StockTake whenever $booksLost$ is large relative to $pp * loansEnded$, then the library managers will indeed see that system is “broken”.

Fixing **StockTake**: Capturing long-term behaviour

New *fix* variable

We introduce a new variable called *fix* as follows: initially, *fix* is given the value 0; *fix* is unchanged in StartLoan and EndLoan operations; and in the StockTake operation, we modify *fix* to maintain the information about the number of *booksLost* related to $pp \times loansEnded$, which is crucial for the expectation:

$$fix \; := \; pp \times loansEnded - booksLost + fix \; . \tag{8}$$

New **expectations**

$$V' \;\widehat{=}\; pp \times loansEnded - booksLost + fix \; . \tag{9}$$

Summary

We have extended standard *Abstract Machine Notation* (to *probabilistic Abstract Machine Notation*) and the semantics of *B*'s machine to enable the concept of a *probabilistic machine*, which supports the following *probabilistic B* constructs:

1. probabilistic invariants or expectations;
2. probabilistic choice;

- Future work
  - Probabilistic Event-B.

For further reading I

C. Morgan and A. McIver.
Abstraction, Refinement and Proof for Probabilistic Systems.
Springer-Verlag, 2004.

T.S. Hoang, Z. Jin, K. Robinson, C. Morgan and A. McIver.
Probabilistic Invariant for Probabilistic Machines.
Proceedings of the 3rd International Conference of B and Z Users, volume 2651 of *LNCS*, 2003.