1 Motivation

1.1 Extension to the B-Method

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;

1.2 Background

The B-Method

- Abstract machines.
- Variables, e.g. \( x, y \).
- Invariant, e.g. \( x \in \mathbb{N} \land y \in \mathbb{N} \land x \leq y \).
- Operations, e.g.

\[
\text{Inc} \overset{\Delta}{=} \begin{cases} \text{pre } x < y \text{ then } \\ x := x + 1 \\ \text{end} \end{cases}
\]

- 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

\[
\begin{align*}
[x := E]Q & \quad \text{The predicate obtained after replacing all free occurrences of } x \text{ in } Q \text{ by } E. \\
\text{skip} Q & \quad Q. \\
[S \mid T]Q & \quad [S]Q \land [T]Q. \\
P[S]Q & \quad P \land [S]Q. \\
P \Rightarrow S|Q & \quad P \Rightarrow [S]Q. \\
\exists x \cdot S|Q & \quad \forall x \cdot [S]Q. 
\end{align*}
\]
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, $\langle \text{false} \rangle = 0$, and $\langle \text{true} \rangle = 1$.
- Generalised version of $\Rightarrow$: the notion of “everywhere no more than”: $B_1 \Rightarrow B_2$.
- Notationally, we have kept predicates as much as possible.

pGSL Syntax and Semantics

Summary

\[
\begin{align*}
[x := E]B & \quad \text{The expectation obtained after replacing all free occurrences of } x \text{ in } B \text{ by } E \\
[\text{skip}]B & \quad B \\
[S_p \oplus T]B & \quad p \times [S]B + (1-p) \times [T]B \\
[S \mid T]B & \quad [S]B \min [T]B \\
[@y \cdot P \Rightarrow S]B & \quad \min (y) \cdot (P \mid [S]B)
\end{align*}
\]

Examples

Example 1

\[
\begin{align*}
[x := y \frac{1}{3} \oplus x := 2 \times y]x^2 & \\
\equiv & \quad \frac{1}{3} \times [x := y]x^2 + \frac{2}{3} \times [x := 2 \times y]x^2 & \text{probabilistic choice} \\
\equiv & \quad \frac{1}{3} \times y^2 + \frac{2}{3} \times (2 \times y)^2 & \text{simple substitutions} \\
\equiv & \quad 3 \times y^2. & \text{arithmetic}
\end{align*}
\]

Example 2

\[
\begin{align*}
[x := y \frac{1}{3} \oplus x := 2 \times y] \langle x = 2 \rangle & \\
\equiv & \quad \frac{1}{3} \times [x := y] \langle x = 2 \rangle + \frac{2}{3} \times [x := 2 \times y] \langle x = 2 \rangle & \text{probabilistic choice} \\
\equiv & \quad \frac{1}{3} \times \langle y = 2 \rangle + \frac{2}{3} \times \langle 2 \times y = 2 \rangle & \text{simple substitutions} \\
\equiv & \quad \frac{1}{3} \times \langle y = 2 \rangle + \frac{2}{3} \times \langle y = 1 \rangle. & \text{arithmetic}
\end{align*}
\]
2 Our Results/Contribution

2.1 Library Example

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 ∈ \mathbb{N} ∧ loansStarted ∈ \mathbb{N} ∧ loansEnded ∈ \mathbb{N} ∧
loansEnded ≤ loansStarted ∧
booksInLibrary + loansStarted − loansEnded = totalBooks
initialisation
booksInLibrary := totalBooks || loansStarted := 0 || loansEnded := 0

Standard Library (cont.)

operations
StartLoan =
pre booksInLibrary > 0 then
booksInLibrary := booksInLibrary − 1 ||
loansStarted := loansStarted + 1
end ;

EndLoan =
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 \triangleq
\begin{aligned}
\text{pre} & \quad \text{booksInLibrary} > 0 \quad \text{then} \\
& \quad \text{booksInLibrary} := \text{booksInLibrary} - 1 \\
\end{aligned}
end

\textbf{Problem}
The problem with this is that we have no way in B of modelling a probabilistically invoked operation.

\textbf{Solution}
An alternative, in probabilistic B, is to model operations with probabilistic effects.

\textbf{The pchoice clause}
The \texttt{pchoice} construct is the \textit{probabilistic Abstract Machine Notation} counterpart of the operator \( p \oplus \), i.e.
\[
\text{pchoice } p \text{ of } \\
\quad S \\
\text{or} \\
\quad T \\
\text{end}
\]
corresponds to \( S \ p \oplus T \).

\textbf{Probabilistic EndLoan operation}

\texttt{EndLoan} \triangleq
\begin{aligned}
\text{pre} & \quad \text{loansEnded} < \text{loansStarted} \quad \text{then} \\
& \quad \text{pchoice } pp \text{ of } \\
& \quad \text{booksLost} := \text{booksLost} + 1 \\
& \quad \text{or} \\
& \quad \text{booksInLibrary} := \text{booksInLibrary} + 1 \\
\end{aligned}
end
\begin{aligned}
\text{loansEnded} := \text{loansEnded} + 1 \\
end
\end{aligned}

\textbf{2.2 The expectations Clause}

\textbf{Reconstruct the invariants}

\textbf{Standard invariant}
\[
\text{booksInLibrary} + \text{booksLost} + \text{loansStarted} - \text{loansEnded} = \text{totalBooks}
\]

\textbf{Probabilistic invariant}

\textbf{EXPECTATIONS}
\[
0 \Rightarrow pp \times \text{loansEnded} - \text{booksLost}
\]
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:

\[ e \Rightarrow V , \]  

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:

\[ true \Rightarrow I , \]  

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 \( e \Rightarrow V \) 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 expectations: \( true \Rightarrow I \) for standard programs; and \( e \Rightarrow V \) for probabilistic programs.

Standard program:

\[
\begin{align*}
true & \Rightarrow [\text{Init}] I \\
I & \Rightarrow [\text{OpX}] I \\
I & \Rightarrow [\text{OpY}] I,
\end{align*}
\]

then we are assured that

\[ true \Rightarrow [\text{Init}; \text{Op?}; \text{Op?}; \ldots; \text{Op?}] I \]  

What do expectations guarantee?

Probabilistic program:

\[
\begin{align*}
e & \Rightarrow [\text{Init}] V \\
V & \Rightarrow [\text{OpX}] V \\
V & \Rightarrow [\text{OpY}] V,
\end{align*}
\]

then we are assured that

\[ e \Rightarrow [\text{Init}; \text{Op?}; \text{Op?}; \ldots; \text{Op?}] V \]

Proof obligations for probabilistic machines

Standard machines

- \( N1 \): The initialisation needs to establish the invariant, i.e. \([\text{Init}]I\).
- \( N2 \): The operations need to maintain the invariant, i.e. \( I \Rightarrow [\text{Op}]I \).

Probabilistic machines

- \( P1 \): The initialisation needs to establish the lower bound of the probabilistic invariant.

\[ e \Rightarrow [\text{Init}] V . \]
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

\[ V \Rightarrow [Op]V. \]

**What the invariant means**

For standard machines, the trace of the values of the expectations of the standard invariant is \( \text{true, true, \ldots, true} \), 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 \( e_0 \Rightarrow e_1 \Rightarrow \ldots \Rightarrow e_n \). 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.

**2.3 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**

\[ totalCost \leftarrow \text{StockTake} \]

\[
\begin{align*}
\text{begin} & \\
\text{totalCost} & := \text{cost} \times \text{booksLost} \ || \\
\text{booksInLibrary} & := \text{booksInLibrary} + \text{booksLost} \ || \\
\text{loansStarted} & := \text{loansStarted} - \text{loansEnded} \ || \\
\text{loansEnded} & := 0 \ || \\
\text{booksLost} & := 0
\end{align*}
\]

**StockTake operation breaks the probabilistic invariant**

For the probabilistic invariant we require \( V \Rightarrow [\text{StockTake}]V \). Consider the right-hand side of that inequality (considering the effect of variables \( \text{loansEnded} \) and \( \text{booksLost} \) only):

\[
\begin{align*}
[\text{StockTake}]V & \\
& \equiv [\text{loansEnded}, \text{booksLost} := 0, 0]V \\
& \equiv [\text{loansEnded}, \text{booksLost} := 0, 0] \\
& \quad (\text{pp} \ast \text{loansEnded} - \text{booksLost}) \\
& \equiv 0.
\end{align*}
\]

This requires us to prove

\[ \text{pp} \ast \text{loansEnded} - \text{booksLost} \Rightarrow 0, \] (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 \texttt{StockTake} whenever \( booksLost \) is large relative to \( pp \times loansEnded \), then the library managers will indeed see that system is “broken”.

Fixing \texttt{StockTake}: Capturing long-term behaviour

New \texttt{fix} variable

We introduce a new variable called \( fix \) as follows: initially, \( fix \) is given the value 0; \( fix \) is unchanged in \texttt{StartLoan} and \texttt{EndLoan} operations; and in the \texttt{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:

\[
fix := pp \times loansEnded - booksLost + fix .
\] (8)

New expectations

\[
V' \equiv pp \times loansEnded - booksLost + fix .
\] (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.

References
