Development with Refinement in Probabilistic B — Foundation and Case Study

T.S. Hoang[1,2] Z. Jin[1] K. Robinson[1,2] C. Morgan[1]
A. McIver[3]

[1]School of Computer Science and Engineering, University of New South Wales

[2]Formal Methods Research Group, National ICT Australia

[3]Department of Computer Science, Macquarie University

4th International Conference of B and Z Users, 2005
University of Surrey, Guildford, U.K.

Outline

1. Motivation
   - Extension to Probabilistic B
   - Background

2. Our Results/Contribution
   - Probabilistic specification substitution
   - Fundamental theorem
   - Proof Obligations for Loops
   - Case Study

Outline

1. Motivation
   - Extension to Probabilistic B
   - Background
2. Our Results/Contribution
   - Probabilistic specification substitution
   - Fundamental theorem
   - Proof Obligations for Loops
   - Case Study

Extending Probabilistic B

- To extend the scope of *probabilistic B* (*pB*) to layered developments;
- Need to introduce *probabilistic specification substitution*;
- To extend *Abstract Machine Notation* (*AMN*) to express
  - probabilistic specification substitution;
  - probabilistic invariant (*expectation*) for loops.

Extending Probabilistic B

- To extend the scope of *probabilistic B* (*pB*) to layered developments;
- Need to introduce *probabilistic specification substitution*;
- To extend *Abstract Machine Notation* (*AMN*) to express
  - probabilistic specification substitution;
  - probabilistic invariant (*expectation*) for loops.

Extending Probabilistic B

- To extend the scope of *probabilistic B* (*pB*) to layered developments;
- Need to introduce *probabilistic specification substitution*;
- To extend *Abstract Machine Notation* (*AMN*) to express
  - probabilistic specification substitution;
  - probabilistic invariant (*expectation*) for loops.

Extending Probabilistic B

- To extend the scope of *probabilistic B* (*pB*) to layered developments;
- Need to introduce *probabilistic specification substitution*;
- To extend *Abstract Machine Notation* (*AMN*) to express
    - probabilistic specification substitution;
    - probabilistic invariant (*expectation*) for loops.

Extending Probabilistic B

- To extend the scope of *probabilistic B* (*pB*) to layered developments;
- Need to introduce *probabilistic specification substitution*;
- To extend *Abstract Machine Notation* (*AMN*) to express
  - probabilistic specification substitution;
  - probabilistic invariant (*expectation*) for loops.

Changing the B-Toolkit

We have adapted the *B-Toolkit* to assist the development of *pB* machines. This involves:

- new syntax;
- proof obligation generation for new constructs;
- reasoning over *real* as well as Boolean.

Changing the B-Toolkit

We have adapted the *B-Toolkit* to assist the development of *pB* machines. This involves:

- new syntax;
- proof obligation generation for new constructs;
- reasoning over *real* as well as Boolean.

Changing the B-Toolkit

We have adapted the *B-Toolkit* to assist the development of *pB* machines. This involves:

- new syntax;
- proof obligation generation for new constructs;
- reasoning over *real* as well as Boolean.

Outline

1. Motivation
   - Extension to Probabilistic B
   - Background

2. Our Results/Contribution
   - Probabilistic specification substitution
   - Fundamental theorem
   - Proof Obligations for Loops
   - Case Study

Probabilistic Generalised Substitution Language

Summary

$[x := E]exp$	The expectation obtained after replacing all free occurrences of x in exp by E
$[skip]exp$	exp
$[prog_1 \; {}_p\!\oplus \; prog_2]exp$	$p \times [prog_1]exp$ $+ \; (1-p) \times [prog_2]exp$
$prog_1 \sqsubseteq prog_2$	$[prog_1]exp \Rrightarrow [prog_2]exp$
$[prog_1 \; [] \; prog_2]exp$	$[prog_1]exp \; \min \; [prog_2]exp$
$[@y \cdot pred \Longrightarrow prog]exp$	$\min\,(y) \cdot (pred \mid [prog]exp)$

Probabilistic Generalised Substitution Language

Summary

$[x := E]exp$	The expectation obtained after replacing all free occurrences of x in exp by E
$[skip]exp$	exp
$[prog_1 \; {}_p\!\oplus \; prog_2]exp$	$p \times [prog_1]exp + (1-p) \times [prog_2]exp$
$prog_1 \sqsubseteq prog_2$	$[prog_1]exp \Rrightarrow [prog_2]exp$
$[prog_1 \; [\!] \; prog_2]exp$	$[prog_1]exp \; \min \; [prog_2]exp$
$[@y \cdot pred \implies prog]exp$	$\min(y) \cdot (pred \mid [prog]exp)$

Probabilistic Generalised Substitution Language

Summary

$[x := E]exp$	The expectation obtained after replacing all free occurrences of x in exp by E
$[skip]exp$	exp
$[prog_1 \ {}_p\!\oplus\ prog_2]exp$	$p \times [prog_1]exp + (1-p) \times [prog_2]exp$
$prog_1 \sqsubseteq prog_2$	$[prog_1]exp \Rrightarrow [prog_2]exp$
$[prog_1 \ [] \ prog_2]exp$	$[prog_1]exp \ \min\ [prog_2]exp$
$[@y \cdot pred \Longrightarrow prog]exp$	$\min\ (y) \cdot (pred \mid [prog]exp)$

Probabilistic Generalised Substitution Language

Summary

$[x := E\,]exp$	The expectation obtained after replacing all free occurrences of x in exp by E
$[skip]exp$	exp
$[prog_1\ {}_p\!\oplus\ prog_2]exp$	$\begin{array}{ccc} & p & \times & [prog_1]exp \\ + & (1-p) & \times & [prog_2]exp \end{array}$
$prog_1 \sqsubseteq prog_2$	$[prog_1]exp \Rrightarrow [prog_2]exp$
$[prog_1\ [\!]\ prog_2]exp$	$[prog_1]exp\ \min\ [prog_2]exp$
$[@y \cdot pred \implies prog\,]exp$	$\min\,(y) \cdot (pred \mid [prog\,]exp)$

How *pGSL* extends *GSL*

Expectations replace predicates

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

- For consistency with Boolean logic, we use embedded predicates, $\langle \mathit{false} \rangle = 0$, and $\langle \mathit{true} \rangle = 1$.
- Notationally, we have kept predicates as much as possible.

How *pGSL* extends *GSL*

Expectations replace predicates

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

- For consistency with Boolean logic, we use embedded predicates, $\langle \mathit{false} \rangle = 0$, and $\langle \mathit{true} \rangle = 1$.
- Notationally, we have kept predicates as much as possible.

How *pGSL* extends *GSL*

Expectations replace predicates

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

- For consistency with Boolean logic, we use embedded predicates, $\langle \mathit{false} \rangle = 0$, and $\langle \mathit{true} \rangle = 1$.
- Notationally, we have kept predicates as much as possible.

Outline

Syntax

We want to have a definition in the probabilistic world which is similar to the precondition, postcondition pair.

Standard substitution

$v : \{P\ ,\ Q\}$, where P and Q are predicates over the state, x.

- $v \subseteq x$
- Q can refer to the original state by using subscripted variables x_0.

Probabilistic substitution

$v : \{A\ ,\ B\}$, where A and B are expectations over state.
The expected value of B over the set of final distributions is at least the expected value of A over the initial distribution.

Syntax

We want to have a definition in the probabilistic world which is similar to the precondition, postcondition pair.

Standard substitution

$v : \{P\ ,\ Q\}$, where P and Q are predicates over the state, x.

- $v \subseteq x$
- Q can refer to the original state by using subscripted variables x_0.

Probabilistic substitution

$v : \{A\ ,\ B\}$, where A and B are expectations over state.
The expected value of B over the set of final distributions is at least the expected value of A over the initial distribution.

Syntax

We want to have a definition in the probabilistic world which is similar to the precondition, postcondition pair.

Standard substitution

$v : \{P\ ,\ Q\}$, where P and Q are predicates over the state, x.

- $v \subseteq x$
- Q can refer to the original state by using subscripted variables x_0.

Probabilistic substitution

$v : \{A\ ,\ B\}$, where A and B are expectations over state.
The expected value of B over the set of final distributions is at least the expected value of A over the initial distribution.

Semantics

Program

Specification $v : \{A \; , \; B\}$.

Post-expectation

Arbitrary expectation C.

Questions?

What is $[v : \{A \; , \; B\}] \, C$?

Semantics of probabilistic substitution

$$[v : \{A \; , \; B\}] \, C \quad \widehat{=} \quad A \times [x_0 := x] \left(\sqcap x \cdot \left(\frac{C}{B \times \langle w = w_0 \rangle} \right) \right)$$

(with w is the set of unchanged variables, i.e. $x - v$).
(Similar work can be seen in White[1996] and Ying[2003])

Semantics

Program

Specification $v : \{A\ ,\ B\}$.

Post-expectation

Arbitrary expectation C.

Questions?

What is $[v : \{A\ ,\ B\}]\, C$?

Semantics of probabilistic substitution

$$[v : \{A\ ,\ B\}]\, C \quad \widehat{=} \quad A \times [x_0 : = x]\left(\sqcap x \cdot \left(\frac{C}{B \times \langle w = w_0 \rangle}\right)\right)$$

(with w is the set of unchanged variables, i.e. $x - v$).
(Similar work can be seen in White[1996] and Ying[2003])

Semantics

Program

Specification $v : \{A \,,\, B\}$.

Post-expectation

Arbitrary expectation C.

Questions?

What is $[v : \{A \,,\, B\}]\, C$?

Semantics of probabilistic substitution

$$[v : \{A \,,\, B\}]\, C \quad \widehat{=} \quad A \times [x_0 := x]\left(\sqcap x \cdot \left(\frac{C}{B \times \langle w = w_0 \rangle}\right)\right)$$

(with w is the set of unchanged variables, i.e. $x - v$).
(Similar work can be seen in White[1996] and Ying[2003])

Semantics

Program

Specification $v : \{A\ ,\ B\}$.

Post-expectation

Arbitrary expectation C.

Questions?

What is $[v : \{A\ ,\ B\}]\,C$?

Semantics of probabilistic substitution

$$[v : \{A\ ,\ B\}]\,C \quad \widehat{=} \quad A \times [x_0 := x]\left(\sqcap x \cdot \left(\frac{C}{B \times \langle w = w_0 \rangle}\right)\right)$$

(with w is the set of unchanged variables, i.e. $x - v$).
(Similar work can be seen in White[1996] and Ying[2003])

Example

Program $prog_1$

$$prog_1 \mathrel{\widehat{=}} c : \left\{ \frac{1}{2} \,,\, \langle c = H \rangle \right\} .$$

Post-expectation $\langle c = H \rangle$

$$
\begin{aligned}
& [c : \{ \tfrac{1}{2} \,,\, \langle c = H \rangle \}] \; \langle c = H \rangle \\
\equiv \quad & \frac{1}{2} \quad * \quad [c_0 := c] \left(\sqcap c \cdot \left(\frac{\langle c=H \rangle}{\langle c=H \rangle} \right) \right) \\
\equiv \quad & \frac{1}{2} \quad * \quad [c_0 := c] \; 1 \\
\equiv \quad & \frac{1}{2}
\end{aligned}
$$

Example

Program $prog_1$

$$prog_1 \mathrel{\widehat{=}} c : \left\{ \frac{1}{2} \,,\, \langle c = H \rangle \right\} .$$

Post-expectation $\langle c = H \rangle$

$$\begin{array}{lllll}
 & \left[c : \left\{ \frac{1}{2} \,,\, \langle c = H \rangle \right\} \right] \langle c = H \rangle & & & \\
\equiv & \frac{1}{2} & * & [c_0 := c] \left(\sqcap c \cdot \left(\frac{\langle c=H \rangle}{\langle c=H \rangle} \right) \right) \\
\equiv & \frac{1}{2} & * & [c_0 := c] \; 1 \\
\equiv & \frac{1}{2} & &
\end{array}$$

Outline

Theorem

Standard Theorem

Assume that $prog_1 \ \widehat{=} \ v : \{P \ , \ Q\} \ .$

For any program $prog_2$, $prog_1 \sqsubseteq prog_2$ *if and only if*

$$P \implies [x_0 := x]\,[prog_2]\,Q^w \ ,$$

where $Q^w \ \widehat{=} \ Q \wedge w = w_0$.

Probabilistic Theorem

Assume that $prog_1 \ \widehat{=} \ v : \{A \ , \ B\} \ .$

For any program $prog_2$, $prog_1 \sqsubseteq prog_2$ *if and only if*

$$A \Rrightarrow [x_0 := x]\,[prog_2]\,B^w \ ,$$

where $B^w \ \widehat{=} \ B \times \langle w = w_0 \rangle$.

Theorem

Standard Theorem

Assume that $prog_1 \;\widehat{=}\; v : \{P\;,\;Q\}\;.$
For any program $prog_2$, $prog_1 \sqsubseteq prog_2$ *if and only if*

$$P \implies [x_0 := x]\,[prog_2]\,Q^w\;,$$

where $Q^w \;\widehat{=}\; Q \wedge w = w_0$.

Probabilistic Theorem

Assume that $prog_1 \;\widehat{=}\; v : \{A\;,\;B\}\;.$
For any program $prog_2$, $prog_1 \sqsubseteq prog_2$ *if and only if*

$$A \Rrightarrow [x_0 := x]\,[prog_2]\,B^w\;,$$

where $B^w \;\widehat{=}\; B \times \langle w = w_0 \rangle$.

Example

Programs $prog_1$ and $prog_2$

Consider $prog_1$ amd $prog_2$ as follows:

$$prog_1 \mathrel{\widehat{=}} c : \left\{ \frac{1}{2} \;,\; \langle c = H \rangle \right\} \quad , \quad prog_2 \mathrel{\widehat{=}} c := H \;{}_{\frac{1}{2}}\!\oplus\; c := T \, .$$

$prog_1 \sqsubseteq prog_2$?

$$\begin{aligned}
& [c_0 := c] \left[c := H \;{}_{\frac{1}{2}}\!\oplus\; c := T \right] \langle c = H \rangle \\
\equiv \quad & \frac{1}{2} \quad \times \quad [c := H] \langle c = H \rangle \\
& + \quad \left(1 - \frac{1}{2}\right) \quad \times \quad [c := T] \langle c = H \rangle \\
\equiv \quad & \frac{1}{2} \times \langle H = H \rangle \;+\; \frac{1}{2} \times \langle T = H \rangle \\
\equiv \quad & \frac{1}{2} \, .
\end{aligned}$$

Example

Programs $prog_1$ and $prog_2$

Consider $prog_1$ amd $prog_2$ as follows:

$$prog_1 \mathrel{\widehat{=}} c : \left\{ \frac{1}{2} , \langle c = H \rangle \right\} \quad , \quad prog_2 \mathrel{\widehat{=}} c : \; = H \; {}_{\frac{1}{2}}\!\oplus \; c : \; = T \, .$$

$prog_1 \sqsubseteq prog_2$?

$$\begin{array}{lccc}
 & [c_0 := c] \left[c : \; = H \; {}_{\frac{1}{2}}\!\oplus \; c : \; = T \right] \langle c = H \rangle & & \\
\equiv & \frac{1}{2} & \times & [c : \; = H] \, \langle c = H \rangle \\
 & + \quad (1 - \frac{1}{2}) & \times & [c : \; = T] \, \langle c = H \rangle \\
\equiv & \frac{1}{2} \times \langle H = H \rangle \; + \; \frac{1}{2} \times \langle T = H \rangle & & \\
\equiv & \frac{1}{2} \, . & &
\end{array}$$

Outline

Standard rules

For a standard loop, such as

$$\mathrm{loop} \;\widehat{=}\; \textsc{while}\ G\ \textsc{do}\ S\ \textsc{invariant}\ I\ \textsc{variant}\ V\ \textsc{end}\ ,$$

then $P \implies [init; loop]Q$ holds if the following are satisfied:

$S1$	P	$\implies$	$[init]I$
$S2$	$G \wedge I$	$\implies$	$[S]I$
$S3$	$\neg G \wedge I$	$\implies$	Q
$S4$	I	$\implies$	$V \in \mathbb{N}$
$S5$	$G \wedge I$	$\implies$	$[n := V][S](V < n)$

Probabilistic rules

For a probabilistic loop, such as

$$\text{loop} \;\widehat{=}\; \textsc{while}\ G\ \textsc{do}\ S\ \textsc{invariant}\ I\ \textsc{expectation}\ E\ \textsc{variant}\ V\ \textsc{end}\ .$$

then $\langle P \rangle * A \Rrightarrow [init; loop](\langle Q \rangle * B)$ holds if the following satisfies:

$P1$		$\langle P \rangle * A$	$\Rrightarrow$	$[init]\,(\langle I \rangle * E)$
	$P1a$	P	$\Longrightarrow$	$[init]\,I$
	$P1b$	$\langle P \rangle * A$	$\Rrightarrow$	$[init]E$
$P2$		$\langle G \wedge I \rangle * E$	$\Rrightarrow$	$[S]\,(\langle I \rangle * E)$
	$P2a$	$G \wedge I$	$\Longrightarrow$	$\llbracket S \rrbracket I$
	$P2b$	$\langle G \wedge I \rangle * E$	$\Rrightarrow$	$[S]\,E$
$P3$		$\langle \neg G \wedge I \rangle * E$	$\Rrightarrow$	$\langle Q \rangle * B$
	$P3a$	$\neg G \wedge I$	$\Longrightarrow$	Q
	$P3b$	$\langle \neg G \wedge I \rangle * E$	$\Rrightarrow$	B
$P4$		I	$\Longrightarrow$	$V \in \mathbb{N}$
$P5$		$G \wedge I$	$\Longrightarrow$	$[n := V]\llbracket S \rrbracket(V < n)$

The difference with the previous work is that there's a clear separation between I and E.

Outline

Description of Min-Cut algorithm

Aims

- Probabilistic fundamental theorem in practice.
- Developing probabilistic system in layers.
- Analysing some of the unexpected and subtle issues.

Two phases

The algorithm is used to find the minimum cut for a connected indirect graph:

- A cut is a set of edges such that if we remove just those edges, the graph will become disconnected;
- A minimum cut is a cut with the least number of edges.

The algorithm contains two phases: Contraction sequences and probabilistic amplification.

Description of Min-Cut algorithm

Aims

- Probabilistic fundamental theorem in practice.
- Developing probabilistic system in layers.
- Analysing some of the unexpected and subtle issues.

Two phases

The algorithm is used to find the minimum cut for a connected indirect graph:

- A cut is a set of edges such that if we remove just those edges, the graph will become disconnected;
- A minimum cut is a cut with the least number of edges.

The algorithm contains two phases: Contraction sequences and probabilistic amplification.

Contraction sequences

Description

- In a contraction step, two connected nodes are chosen randomly and merge together.
- The probability that any specific minimum cut is kept is at least

$$\frac{N-2}{N},$$

where N is the number of nodes in the current graph.

- This step is repeated until there are two nodes left, the edges connecting the last two nodes will be the cut chosen.
- Overall, the probability that the last cut is minimum cut is at least

$$p(N) = \frac{N-2}{N} \times \frac{N-3}{N-1} x \times \cdots \times \frac{2}{4} \times \frac{1}{2} = \frac{2}{N \times (N-1)}.$$

Formal development of contraction

Specification

$ans \longleftarrow$ **contraction**$(\ N\ ) \ \ \widehat{=} \ \ ans : \{\langle pre1\rangle * p(N)\ ,\ \langle ans\rangle\}$

Implementation

$ans \longleftarrow$ **contraction**$(\ N\ ) \ \widehat{=}$
VAR n **IN**
  $n := N;\ ans := TRUE;$
  WHILE $2 < n$ **DO** $ans \longleftarrow merge(n, ans);\ n := n - 1$ **END**
END

merge operation

$ans \longleftarrow$ **merge** $(\ n\ ,\ a\ ) \ \widehat{=}$
  $n \in \mathbb{N} \wedge a \in BOOL \quad | \quad ans := FALSE \ {}_{\le \frac{2}{n}}\!\oplus\ a$

Formal development of contraction

Specification

$ans \longleftarrow$ **contraction**$(\ N\ )\ \ \widehat{=}\ \ ans : \{\langle pre1\rangle * p(N)\ ,\ \langle ans\rangle\}$

Implementation

$ans \longleftarrow$ **contraction**$(\ N\ )\ \widehat{=}$
VAR n **IN**
  $n := N;\ ans := TRUE;$
  WHILE $2 < n$ **DO** $ans \longleftarrow merge(n, ans);\ n := n - 1$ **END**
END

merge operation

$ans \longleftarrow$ **merge** $(\ n\ ,\ a\ )\ \widehat{=}$
  $n \in \mathbb{N} \wedge a \in BOOL \quad | \quad ans := FALSE \ \leq_{\frac{2}{n}} \oplus\ a$

Formal development of contraction

Specification

$ans \longleftarrow \textbf{contraction}(\ N\ ) \ \ \widehat{=} \ \ ans : \{\langle pre1 \rangle * p(N)\ ,\ \langle ans \rangle\}$

Implementation

$ans \longleftarrow \textbf{contraction}(\ N\ ) \ \widehat{=}$
VAR n **IN**
  $n := N;\ ans := TRUE;$
  WHILE $2 < n$ **DO** $ans \longleftarrow merge(n, ans);\ n := n - 1$ **END**
END

merge operation

$ans \longleftarrow \textbf{merge}\ (\ n\ ,\ a\ )\ \widehat{=}$
  $n \in \mathbb{N} \wedge a \in BOOL \quad | \quad ans := FALSE \ _{\leq \frac{2}{n}}\!\oplus\ a$

Proof obligations of contraction

Here is the summary of proof obligations for the implementation generated by the modified *B-Toolkit*:

Summary

Total	Auto Prove	BTool Prove
14	11	3

Probabilistic amplification

Description

- We repeat the contraction sequences to increase the chance of finding the right minimum cut.
- Assume that we do that M times, the probability of finding the right minimum cut is at least:

$$P(N, M) = 1 - (1 - p(N))^{M} .$$

Formal development of probabilistic amplification

Specification

$ans \longleftarrow \textbf{minCut}(\ N, M\ ) \quad \widehat{=} \quad ans : \{\langle pre2\rangle * P(N, M)\ ,\ \langle ans\rangle\}$

Implementation

$ans \longleftarrow \textbf{minCut}(\ N, M\ ) \ \widehat{=}$
VAR m, a **IN**
  $m := M$; $ans := FALSE$;
  WHILE $m \neq 0$ **DO**
    $a \longleftarrow contraction(N)$;
    $ans := ans \lor a$;
    $m := m - 1$
  END
END

Formal development of probabilistic amplification

Specification

$ans \longleftarrow \textbf{minCut}(\ N, M\ ) \quad \widehat{=} \quad ans : \{\langle pre2\rangle * P(N,M)\ ,\ \langle ans\rangle\}$

Implementation

$ans \longleftarrow \textbf{minCut}(\ N, M\ ) \ \widehat{=}$
VAR m, a **IN**
  $m := M; ans := FALSE;$
  WHILE $m \neq 0$ **DO**
    $a \longleftarrow contraction(N);$
    $ans := ans \vee a;$
    $m := m - 1$
  END
END

Proof obligations of probabilistic amplification

Here is the summary of proof obligations for probabilistic amplification produced by the modified *B-Toolkit*:

Summary

Total	Auto Prove	BTool Prove
14	13	0

Problem?

There is one proof obligation that cannot be proved.

Solution

The problem observed is due to the fact that in the definition for probabilistic specification substitution, we did not specify termination.

In *B*, termination of all programs must be proved, so we should introduce terminating probabilistic specification substitution and its fundamental theorem.

Proof obligations of probabilistic amplification

Here is the summary of proof obligations for probabilistic amplification produced by the modified *B-Toolkit*:

Summary

Total	Auto Prove	BTool Prove
14	13	0

Problem?

There is one proof obligation that cannot be proved.

Solution

The problem observed is due to the fact that in the definition for probabilistic specification substitution, we did not specify termination.

In *B*, termination of all programs must be proved, so we should introduce terminating probabilistic specification substitution and its fundamental theorem.

Proof obligations of probabilistic amplification

Here is the summary of proof obligations for probabilistic amplification produced by the modified *B-Toolkit*:

Summary

Total	Auto Prove	BTool Prove
14	13	0

Problem?

There is one proof obligation that cannot be proved.

Solution

The problem observed is due to the fact that in the definition for probabilistic specification substitution, we did not specify termination.
In *B*, termination of all programs must be proved, so we should introduce terminating probabilistic specification substitution and its fundamental theorem.

Summary

- Abstractly specify and refine probabilistic system.
- Development can be separated into layers.
- Termination condition is checked when developing systems using the *B-Toolkit*.

- Future work
  - Multiple expectations.
  - Fundamental theorem for refining system with multiple expectations.

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.

- N. White.
  Probabilistic Specification and Refinement
  Master Thesis, Keble College, 1996.

- M.S. Ying.
  Reasoning about probabilistic sequential programs in a probabilistic logic.
  Acta Informatica, volume 39, 2003.