Program Analysis (70020)
Probabilistic Programs

Herbert Wiklicky

Department of Computing
Imperial College London

herbert@doc.ic.ac.uk
h.wiklicky@imperial.ac.uk

Autumn 2022
Overview

Topics we will cover in this part will include:
Topics we will cover in this part will include:

1. Language PWILE
Topics we will cover in this part will include:

1. Language PWHILE
2. Operational Semantics
Overview

Topics we will cover in this part will include:

1. Language PWHILE
2. Operational Semantics
3. Tensor Products
Overview

Topics we will cover in this part will include:

1. Language PWHILE
2. Operational Semantics
3. Tensor Products
4. Linear Operator Semantics
Overview

Topics we will cover in this part will include:

1. Language \(\text{PWHILE}\)
2. Operational Semantics
3. Tensor Products
4. Linear Operator Semantics
5. Probabilistic Abstract Interpretation
1: \( [m := 1] \);  
2: while \( [n > 1] \) do  
3: \( [m := m \times n] \);  
4: \( [n := n - 1] \)  
5: end while  
6: [stop]  

Concrete Probabilities
Probabilistic Problem I: Guards and Conditionals

1: \[ m := 1 \] \; \triangleright \; P(m = 1), P(m = 2), \ldots \; \triangleright \; P(n = 1), \ldots

2: while \[ n > 1 \] do
3: \[ m := m \times n \]
4: \[ n := n - 1 \]
5: end while
6: [stop]

Concrete Probabilities
Probabilistic Problem I: Guards and Conditionals

1: \[ m := 1 \]
2: \( \text{while } [n > 1] \) do
3: \[ m := m \times n \]
4: \[ n := n - 1 \]
5: \( \text{end while} \)
6: \( \text{stop} \)

\( \triangleright \) \((p_1, p_2, p_3, \ldots) \leftarrow (q_1, q_2, \ldots)\)

Concrete Probabilities
Probabilistic Problem I: Guards and Conditionals

1: \[ m := 1 \] ;  
2: \textbf{while} \[ n > 1 \] \textbf{do}  
3: \[ m := m \times n \] ;  
4: \[ n := n - 1 \]  
5: \textbf{end while}  
6: \textbf{stop}  

\[ (p_1, p_2, p_3, \ldots) \sim (\frac{1}{2}, \frac{1}{2}, \ldots) \]

Concrete Probabilities
Probabilistic Problem I: Guards and Conditionals

1: \[ m := 1 \]
2: \[ \textbf{while } [n > 1] \textbf{ do} \]
3: \[ m := m \times n \]
4: \[ n := n - 1 \]
5: \[ \textbf{end while} \]
6: \[ \textbf{stop} \]

\[ \triangleright (p_1, p_2, p_3, \ldots) \rightarrow (\frac{1}{2}, \frac{1}{2}, \ldots) \]
\[ \triangleright (1, 0, 0, \ldots) \rightarrow (\frac{1}{2}, \frac{1}{2}, \ldots) \]

Concrete Probabilities
Probabilistic Problem I: Guards and Conditionals

1: \[ m := 1; \]
2: \[ \text{while } [n > 1] \ do \]
3: \[ m := m \times n; \]
4: \[ n := n - 1; \]
5: \[ \text{end while} \]
6: \[ \text{stop} \]

\[ \triangleright (p_1, p_2, p_3, \ldots) \rightarrow \left(\frac{1}{2}, \frac{1}{2}, \ldots\right) \]
\[ \triangleright (1, 0, 0, \ldots) \rightarrow \left(\frac{1}{2}, \frac{1}{2}, \ldots\right) \]

Concrete Probabilities
Probabilistic Problem I: Guards and Conditionals

1: \[ m := 1 \]
2: while \[ n > 1 \] do
3: \[ m := m \times n \]
4: \[ n := n - 1 \]
5: end while
6: [stop]

\[ (p_1, p_2, p_3, \ldots) \] — \( \left( \frac{1}{2}, \frac{1}{2}, \ldots \right) \)
\[ (1, 0, 0, \ldots) \] — \( \left( \frac{1}{2}, \frac{1}{2}, \ldots \right) \)
\[ (1, 0, 0, \ldots) \] — \( (0, \frac{1}{2}, \ldots) \)

Concrete Probabilities
Probabilistic Problem I: Guards and Conditionals

1: \[ m := 1 \]
2: \[ \textbf{while } [n > 1] \textbf{ do} \]
3: \[ m := m \times n \]
4: \[ n := n - 1 \]
5: \[ \textbf{end while} \]
6: \[ \textbf{stop} \]

\[ (p_1, p_2, p_3, \ldots) \overset{\triangleright}{=} (\frac{1}{2}, \frac{1}{2}, \ldots) \]
\[ (1, 0, 0, \ldots) \overset{\triangleright}{=} (\frac{1}{2}, \frac{1}{2}, \ldots) \]
\[ (1, 0, 0, \ldots) \overset{\triangleright}{=} (0, \frac{1}{2}, \ldots) \]
\[ (0, 1, 0, \ldots) \overset{\triangleright}{=} (0, \frac{1}{2}, \ldots) \]
\[ (1, 0, 0, \ldots) \overset{\triangleright}{=} (\frac{1}{2}, 0, \ldots) \]

Concrete Probabilities
Probabilistic Problem I: Guards and Conditionals

1: $[m := 1]^1$;
2: while $[n > 1]^2$ do
3: $[m := m \times n]^3$;
4: $[n := n - 1]^4$
5: end while
6: $[\text{stop}]^5$

$\triangleright (p_1, p_2, p_3, \ldots) \oplus (\frac{1}{2}, \frac{1}{2}, \ldots)$
$\triangleright (1, 0, 0, \ldots) \oplus (\frac{1}{2}, \frac{1}{2}, \ldots)$
$\triangleright (1, 0, 0, \ldots) \oplus (0, \frac{1}{2}, \ldots)$
$\triangleright (0, 1, 0, \ldots) \oplus (0, \frac{1}{2}, \ldots)$
$\triangleright (0, 1, 0, \ldots) \oplus (\frac{1}{2}, 0, \ldots)$
$\triangleright (1, 0, 0, \ldots) \oplus (\frac{1}{2}, 0, \ldots)$

Concrete Probabilities
Probabilistic Problem I: Guards and Conditionals

1: \( [m := 1] \)
2: \( \textbf{while} \ [n > 1] \textbf{ do} \)
3: \( [m := m \times n] \)
4: \( [n := n - 1] \)
5: \( \textbf{end while} \)
6: \( \textbf{stop} \)

Concrete Probabilities

\( \triangleright (p_1, p_2, p_3, \ldots) \rightarrow (\frac{1}{2}, \frac{1}{2}, \ldots) \)
\( \triangleright (1, 0, 0, \ldots) \rightarrow (\frac{1}{2}, \frac{1}{2}, \ldots) \)
\( \triangleright (1, 0, 0, \ldots) \rightarrow (0, \frac{1}{2}, \ldots) \)
\( \triangleright (0, \frac{1}{2}, 0, \ldots) \rightarrow (0, \frac{1}{2}, \ldots) \)
\( \triangleright (0, 1, 0, \ldots) \rightarrow (\frac{1}{2}, 0, \ldots) \)
\( \triangleright (1, 0, 0, \ldots) \rightarrow (\frac{1}{2}, 0, \ldots) \)

Perhaps better this way?
Probabilistic Problem I: Guards and Conditionals

1: \[ m := 1 \]
2: while \[ n > 1 \] do
3: \[ m := m \times n \]
4: \[ n := n - 1 \]
5: end while
6: [stop]

\[ (p_1, p_2, p_3, \ldots) \sim \left( \frac{1}{2}, \frac{1}{2}, \ldots \right) \]
\[ (1, 0, 0, \ldots) \sim \left( \frac{1}{2}, \frac{1}{2}, \ldots \right) \]
\[ (1, 0, 0, \ldots) \sim (0, \frac{1}{2}, \ldots) \]
\[ (0, 1, 0, \ldots) \sim (0, \frac{1}{2}, \ldots) \]
\[ (0, 1, 0, \ldots) \sim \left( \frac{1}{2}, 0, \ldots \right) \]
\[ (1, 0, 0, \ldots) \sim \left( \frac{1}{2}, 0, \ldots \right) \]
Probabilistic Problem I: Guards and Conditionals

1: \([m := 1]\)^1;
2: \textbf{while} \([n > 1]\)^2 \textbf{do}
3: \quad [m := m \times n]^3;
4: \quad [n := n - 1]^4
5: \textbf{end while}
6: [\textbf{stop}]^5

\[ \triangleright (p_1, p_2, p_3, \ldots) \rightarrow (\frac{1}{2}, \frac{1}{2}, \ldots) \]
\[ \triangleright (0, 1, 0, \ldots) \rightarrow (\frac{1}{2}, 0, \ldots) \]
\[ \triangleright (1, 0, 0, \ldots) \rightarrow (\frac{1}{2}, 0, \ldots) \]

Concrete Probabilities
Probabilistic Problem I: Guards and Conditionals

1: \[ m := 1 \]  
2: while \[ n > 1 \] do  
3: \[ m := m \times n \]  
4: \[ n := n - 1 \]  
5: end while  
6: \[ \text{stop} \]  

\[ \bowtie (p_1, p_2, p_3, \ldots) \sim (\frac{1}{2}, \frac{1}{2}, \ldots) \]  
\[ \bowtie (0, 1, 0, \ldots) \sim (\frac{1}{2}, 0, \ldots) \]  
\[ (1, 0, 0, \ldots) \sim (\frac{1}{2}, 0, \ldots) \]  
\[ (0, 1, 0, \ldots) \sim (\frac{1}{2}, 0, \ldots) \]  

Concrete Probabilities
1: \[ m := 1 \]
2: while \[ n > 1 \] do
3: \[ m := m \times n \]
4: \[ n := n - 1 \]
5: end while
6: \[ \text{stop} \]

Concrete Probabilities

\( (p_1, p_2, p_3, \ldots) \) — \( \left( \frac{1}{2}, \frac{1}{2}, \ldots \right) \)

\( (0, 1, 0, \ldots) \) — \( \left( \frac{1}{2}, 0, \ldots \right) \)

\( (1, 1, 0, \ldots) \) — \( (1, 0, \ldots) \)

Correct? How to justify this?
Probabilistic Problem I: Guards and Conditionals

1: \[ m := 1 \] ;
2: \textbf{while} \( n > 1 \) \textbf{do}
3: \[ m := m \times n \] ;
4: \[ n := n - 1 \] ;
5: \textbf{end while}
6: \textbf{stop}

\[ \triangledown (p_1, p_2, p_3, \ldots) \sim (\frac{1}{2}, \frac{1}{2}, \ldots) \]
\[ \triangledown (0, 1, 0, \ldots) \sim (\frac{1}{2}, 0, \ldots) \]

Concrete Probabilities

\[ \triangledown (1, 0, 0, \ldots) \sim (\frac{1}{2}, 0, \ldots) \]
\[ (0, 1, 0, \ldots) \sim (\frac{1}{2}, 0, \ldots) \]
Probabilistic Problem II: Abstract Evaluation

1: \[ m := 1 \];
2: \textbf{while} \[ n > 1 \] \textbf{do}
3: \[ m := m \times n \];
4: \[ n := n - 1 \]
5: \textbf{end while}
6: \textbf{stop}

Abstract Probabilities
Probabilistic Problem II: Abstract Evaluation

1: \[ m := 1 \] \hspace{1cm} \triangleright \ P(m = 2k), P(m \neq 2k) \rightarrow P(n = 1), \ldots \\
2: \textbf{while} [n > 1] \textbf{do} \\
3: \quad [m := m \times n] \\
4: \quad [n := n - 1] \\
5: \textbf{end while} \\
6: [\textbf{stop}] \\

Abstract Probabilities
Probabilistic Problem II: Abstract Evaluation

1: \[ m := 1 \]
2: \[ m := 1 \]
\[ \triangle (p_e, p_o) \rightarrow (q_1, q_2, \ldots) \]
3: \[ m := m \times n \]
4: \[ n := n - 1 \]
5: \[ \text{end while} \]
6: \[ \text{stop} \]

Abstract Probabilities
Probabilistic Problem II: Abstract Evaluation

1: \[ m := 1 \]
2: \[ \text{while } [n > 1] \text{ do} \]
3: \[ m := m \times n \]
4: \[ n := n - 1 \]
5: \[ \text{end while} \]
6: \[ \text{stop} \]

\( (p_e, p_o) \leftarrow (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots) \)

Abstract Probabilities
Probabilistic Problem II: Abstract Evaluation

1: \[ m := 1 \]
2: \textbf{while} \[ n > 1 \] \textbf{do}
3: \[ m := m \times n \]
4: \[ n := n - 1 \]
5: \textbf{end while}
6: \[ \text{stop} \]

\[ (p_e, p_o) \rightarrow (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots) \]
\[ (0, 1) \rightarrow (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots) \]

Abstract Probabilities
Probabilistic Problem II: Abstract Evaluation

1: \[ m := 1 \]
2: \[ \textbf{while } [n > 1] \textbf{ do} \]
3: \[ m := m \times n \]
4: \[ n := n - 1 \]
5: \[ \textbf{end while} \]
6: \[ \textbf{stop} \]

\[ (p_e, p_o) \rightarrow (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots) \]
\[ (0, 1) \rightarrow (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots) \]

Abstract Probabilities
Probabilistic Problem II: Abstract Evaluation

1: \[ m := 1 \] \[ \cdot \]
2: while \( n > 1 \) do
3: \[ m := m \times n \] \[ \cdot \]
4: \[ n := n - 1 \] \[ \cdot \]
5: end while
6: \[ \text{stop} \]

Abstract Probabilities

\( (p_e, p_o) \rightarrow (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots) \)
\( (0, 1) \rightarrow (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots) \)
\( (0, 1) \rightarrow (0, \frac{1}{3}, \frac{1}{3}, \ldots) \)
\( (0, 1) \rightarrow (\frac{1}{3}, 0, 0, \ldots) \)
Probabilistic Problem II: Abstract Evaluation

1: \[ m := 1 \];
2: \textbf{while} \ [n > 1] \textbf{do}
3: \[ m := m \times n \];
4: \[ n := n - 1 \]
5: \textbf{end while}
6: \textbf{stop}

\[\triangle (p_e, p_o) \rightleftharpoons \left( \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots \right)\]
\[\triangle (0, 1) \rightleftharpoons \left( \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots \right)\]
\[\triangle (0, 1) \rightleftharpoons (0, \frac{1}{3}, \frac{1}{3}, \ldots)\]
\[\triangle (1, 0) \rightleftharpoons (0, \frac{1}{3}, \frac{1}{3}, \ldots)\]

\[\triangle (0, 1) \rightleftharpoons \left( \frac{1}{3}, 0, 0, \ldots \right)\]
Probabilistic Problem II: Abstract Evaluation

1: $[m := 1]$;

2: while $[n > 1]$ do

3: $[m := m \times n]$;

4: $[n := n - 1]$;

5: end while

6: [stop]

Abstract Probabilities

$\triangleright (p_e, p_o) = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots)$

$\triangleright (0, 1) = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots)$

$\triangleright (0, 1) = (0, \frac{1}{3}, \frac{1}{3}, \ldots)$

$\triangleright (1, 0) = (0, \frac{1}{3}, \frac{1}{3}, \ldots)$

$\triangleright (1, 0) = (\frac{1}{3}, \frac{1}{3}, 0, \ldots)$

$\triangleright (0, 1) = (\frac{1}{3}, 0, 0, \ldots)$
Probabilistic Problem II: Abstract Evaluation

1: \[ m := 1 \]

2: while \[ n > 1 \] do

3: \[ m := m \times n \]

4: \[ n := n - 1 \]

5: end while

6: \[ \text{stop} \]

\[ \triangleright (p_e, p_o) \rightarrow (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots) \]

\[ \triangleright (1, 0) \rightarrow (\frac{1}{3}, \frac{1}{3}, 0, \ldots) \]

\[ \triangleright (0, 1) \rightarrow (\frac{1}{3}, 0, 0, \ldots) \]

Abstract Probabilities
Probabilistic Problem II: Abstract Evaluation

1: $[m := 1]$;  
2: while $[n > 1]$ do  
3: $[m := m \times n]$;  
4: $[n := n - 1]$  
5: end while  
6: $[\text{stop}]$  

$\triangleright (pe, po) \rightarrow (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots)$  
$\triangleright (1, 0) \rightarrow (\frac{1}{3}, \frac{1}{3}, 0, \ldots)$  
$\triangleright (1, 0) \rightarrow (0, \frac{1}{3}, 0, \ldots)$  
$\triangleright (0, 1) \rightarrow (\frac{1}{3}, 0, 0, \ldots)$  
$\triangleright (1, 0) \rightarrow (\frac{1}{3}, 0, 0, \ldots)$

Abstract Probabilities
Probabilistic Problem II: Abstract Evaluation

1: \( m := 1 \);  

2: \( \textbf{while} \ [n > 1] \textbf{do} \)
3: \( m := m \times n \);  
4: \( n := n - 1 \)
5: \( \textbf{end while} \)
6: \( \textbf{stop} \)

\( \triangledown \) \((p_e, p_o) \longrightarrow (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots)\)

\( \triangledown \) \((1, 0) \longrightarrow (\frac{1}{3}, \frac{1}{3}, 0, \ldots)\)

\( \triangledown \) \((1, 0) \longrightarrow (0, \frac{1}{3}, 0, \ldots)\)

\( \triangledown \) \((1, 0) \longrightarrow (0, \frac{1}{3}, 0, \ldots)\)

\( \triangledown \) \((0, 1) \longrightarrow (\frac{1}{3}, 0, 0, \ldots)\)

\( (1, 0) \longrightarrow (\frac{1}{3}, 0, 0, \ldots)\)

Abstract Probabilities
Probabilistic Problem II: Abstract Evaluation

1: \[ m \leftarrow 1 \]
2: while \[ n > 1 \] do
3: \[ m \leftarrow m \times n \]
4: \[ n \leftarrow n - 1 \]
5: end while
6: \[ \text{stop} \]

\[ (p_e, p_o) \rightarrow (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots) \]
\[ (1, 0) \rightarrow (\frac{1}{3}, \frac{1}{3}, 0, \ldots) \]
\[ (1, 0) \rightarrow (0, \frac{1}{3}, 0, \ldots) \]
\[ (1, 0) \rightarrow (0, \frac{1}{3}, 0, \ldots) \]
\[ (1, 0) \rightarrow (\frac{1}{3}, 0, 0, \ldots) \]
\[ (0, 1) \rightarrow (\frac{1}{3}, 0, 0, \ldots) \]
\[ (1, 0) \rightarrow (\frac{1}{3}, 0, 0, \ldots) \]

Abstract Probabilities
Probabilistic Problem II: Abstract Evaluation

1: \[ m := 1 \]
2: \[ \text{while } [n > 1] \text{ do} \]
3: \[ m := m \times n \]
4: \[ n := n - 1 \]
5: \[ \text{end while} \]
6: \[ \text{stop} \]

\[ \triangleright (p_e, p_o) \rightarrow (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots) \]
\[ \triangleright (1, 0) \rightarrow (\frac{1}{3}, 0, 0, \ldots) \]

Abstract Probabilities
Abstract Probabilities
Probabilistic Problem II: Abstract Evaluation

1: \[ m := 1 \]

2: \[ \textbf{while } [n > 1] \textbf{ do} \]

3: \[ m := m \times n \]

4: \[ n := n - 1 \]

5: \[ \textbf{end while} \]

6: \[ \text{stop} \]

\[ (p_e, p_o) \rightarrow (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots) \]

\[ (1, 0) \rightarrow (\frac{1}{3}, \frac{1}{3}, 0, \ldots) \]

\[ (0, 1) \rightarrow (0, \frac{1}{3}, \frac{1}{3}, \ldots) \]

\[ (1, 0) \rightarrow (0, \frac{1}{3}, \frac{1}{3}, \ldots) \]

Abstract Probabilities

Correct?
Probabilistic Problem II: Abstract Evaluation

1: \[ m := 1 \]
2: \[ \text{while } [n > 1] \]
3: \[ m := m \times n \]
4: \[ n := n - 1 \]
5: \[ \text{end while} \]
6: \[ \text{stop} \]

Abstract Probabilities

\[ (p_e, p_o) \rightarrow (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots) \]
\[ (1, 0) \rightarrow (\frac{1}{3}, \frac{1}{3}, 0, \ldots) \]
\[ (0, 1) \rightarrow (0, \frac{1}{3}, \frac{1}{3}, \ldots) \]
\[ (1, 0) \rightarrow (0, \frac{1}{3}, \frac{1}{3}, \ldots) \]

Abstract Probabilities

How to justify this?
Probabilistic Problem III: Relational Dependency

Given an (input) distribution \((\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots)\) for \(n\) one would expect an (output) distribution \((\frac{2}{3}, \frac{1}{3})\) for even\((m)\) and odd\((m)\).
Probabilistic Problem III: Relational Dependency

Given an (input) distribution \((\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots)\) for \(n\) one would expect an (output) distribution \((\frac{2}{3}, \frac{1}{3})\) for \(\text{even}(m)\) and \(\text{odd}(m)\).

For every pair \((m, n)\) we can write the probabilities to observe it as \(P(m = i \land n = j) = P(m = i)P(n = j)\) – assume perhaps that \(n\) does not change.
Given an (input) distribution \((\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots)\) for \(n\) one would expect an (output) distribution \((\frac{2}{3}, \frac{1}{3})\) for \(\text{even}(m)\) and \(\text{odd}(m)\).

For every pair \((m, n)\) we can write the probabilities to observe it as \(P(m = i \wedge n = j) = P(m = i)P(n = j)\) – assume perhaps that \(n\) does not change.

The available data thus suggest this probability distribution:

<table>
<thead>
<tr>
<th></th>
<th>(n = 1)</th>
<th>(n = 2)</th>
<th>(n = 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{even}(m))</td>
<td>(\frac{1}{3} \cdot \frac{2}{3})</td>
<td>(\frac{1}{3} \cdot \frac{2}{3})</td>
<td>(\frac{1}{3} \cdot \frac{2}{3})</td>
</tr>
<tr>
<td>(\text{odd}(m))</td>
<td>(\frac{1}{3} \cdot \frac{1}{3})</td>
<td>(\frac{1}{3} \cdot \frac{1}{3})</td>
<td>(\frac{1}{3} \cdot \frac{1}{3})</td>
</tr>
</tbody>
</table>
Given an (input) distribution \( (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots) \) for \( n \) one would expect an (output) distribution \( (\frac{2}{3}, \frac{1}{3}) \) for even\((m)\) and odd\((m)\).

For every pair \((m, n)\) we can write the probabilities to observe it as \( P(m = i \wedge n = j) = P(m = i)P(n = j) \) – assume perhaps that \( n \) does not change.

The available data thus suggest this probability distribution:

\[
\begin{array}{c|ccc}
  & n = 1 & n = 2 & n = 3 \\
\hline
\text{even}(m) & \frac{2}{9} & \frac{2}{9} & \frac{2}{9} \\
\text{odd}(m) & \frac{1}{9} & \frac{1}{9} & \frac{1}{9}
\end{array}
\]
Given an (input) distribution \((\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots)\) for \(n\) one would expect an (output) distribution \((\frac{2}{3}, \frac{1}{3})\) for \(\text{even}(m)\) and \(\text{odd}(m)\).

For every pair \((m, n)\) we can write the probabilities to observe it as \(P(m = i \land n = j) = P(m = i)P(n = j)\) – assume perhaps that \(n\) does not change.

In fact, we have the following joint probability distribution:

<table>
<thead>
<tr>
<th></th>
<th>(n = 1)</th>
<th>(n = 2)</th>
<th>(n = 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{even}(m))</td>
<td>0</td>
<td>(\frac{1}{3})</td>
<td>(\frac{1}{3})</td>
</tr>
<tr>
<td>(\text{odd}(m))</td>
<td>(\frac{1}{3})</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Problems in Probabilistic Program Analysis

1: \[ m := 1 \]

2: \[ \text{while } [n > 1] \text{ do} \]

3: \[ m := m \times n \]

4: \[ n := n - 1 \]

5: end while

6: [stop]

\[ (p_e, p_o) \rightarrow (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots) \]

\[ (0, 1) \rightarrow (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots) \]

\[ (0, 1) \rightarrow (0, \frac{1}{3}, \frac{1}{3}, \ldots) \]

\[ (1, 0) \rightarrow (0, \frac{1}{3}, \frac{1}{3}, \ldots) \]

\[ (0, 1) \rightarrow (\frac{1}{3}, 0, 0, \ldots) \]
Problems in Probabilistic Program Analysis

1: \[ m := 1 \]
2: while \[ n > 1 \] do
3: \[ m := m \times n \]
4: \[ n := n - 1 \]
5: end while
6: \[ \text{stop} \]

\[ (p_e, p_o) \rightarrow \left( \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots \right) \]
\[ (0, 1) \rightarrow \left( \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots \right) \]
\[ (0, 1) \rightarrow (0, \frac{1}{3}, \frac{1}{3}, \ldots) \]
\[ (1, 0) \rightarrow (0, \frac{1}{3}, \frac{1}{3}, \ldots) \]
\[ (0, 1) \rightarrow (\frac{1}{3}, 0, 0, \ldots) \]
Problems in Probabilistic Program Analysis

1: \[m := 1\]

2: \textbf{while} \[n > 1\] \textbf{do}

3: \[m := m \times n\]

4: \[n := n - 1\]

5: \textbf{end while}

6: [stop]

\[\triangleright (p_e, p_o) \rightarrow (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots)\]

\[\triangleright (0, 1) \rightarrow (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots)\]

\[\triangleright (0, 1) \rightarrow (0, \frac{1}{3}, \frac{1}{3}, \ldots)\]

\[\triangleright (1, 0) \rightarrow (0, \frac{1}{3}, \frac{1}{3}, \ldots)\]

\[\triangleright (0, 1) \rightarrow (\frac{1}{3}, 0, 0, \ldots)\]

Splitting: How to distribute information along branches?
Problems in Probabilistic Program Analysis

1: \[m := 1\]
2: \[\textbf{while} \ [n > 1] \textbf{do}\]
3: \[m := m \times n\]
4: \[n := n - 1\]
5: \textbf{end while}
6: \[\textbf{stop}\]

\[\triangleright (p_e, p_o) = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots)\]
\[\triangleright (0, 1) = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots)\]
\[\triangleright (0, 1) = (0, \frac{1}{3}, \frac{1}{3}, \ldots)\]
\[\triangleright (1, 0) = (0, \frac{1}{3}, \frac{1}{3}, \ldots)\]
\[\triangleright (0, 1) = (\frac{1}{3}, 0, 0, \ldots)\]

**Splitting:** How to distribute information along branches?

**Transforming:** How computing changes the information?
Problems in Probabilistic Program Analysis

1: $[m := 1]$; 
2: while $[n > 1]$ do 
3: $[m := m \times n]$; 
4: $[n := n - 1]$ 
5: end while 
6: [stop]

\[ (p_e, p_o) \rightarrow (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots) \]
\[ (0, 1) \rightarrow (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots) \]
\[ (0, 1) \rightarrow (0, \frac{1}{3}, \frac{1}{3}, \ldots) \]
\[ (1, 0) \rightarrow (0, \frac{1}{3}, \frac{1}{3}, \ldots) \]

Splitting: How to distribute information along branches?
Transforming: How computing changes the information?
Joining: How to combine information along branches?
Commonly, computations are understood to follow a well defined (deterministic) set of rules as to obtain a certain result.
Commonly, computations are understood to follow a well defined (deterministic) set of rules as to obtain a certain result.

There are randomised algorithms which involve an element of chance or randomness.
Commonly, computations are understood to follow a well defined (deterministic) set of rules as to obtain a certain result.

There are **randomised** algorithms which involve an element of chance or randomness.

**Las Vegas Algorithms** are randomised algorithms that always give correct results (with non-deterministic running time), e.g. QuickSort (with random pivoting).
Commonly, computations are understood to follow a well defined (deterministic) set of rules as to obtain a certain result.

There are randomised algorithms which involve an element of chance or randomness.

Las Vegas Algorithms are randomised algorithms that always give correct results (with non-deterministic running time), e.g. QuickSort (with random pivoting).

Monte Carlo Algorithms produce (with deterministic running time) an output which may be incorrect with a certain probability, e.g. Buffon’s Needle.
Pr(cross) = \frac{2}{\pi} \text{ or } \pi = \frac{2}{\Pr(\text{cross})}
The Monty Hall Problem

The game show proceeds as follows: First the contestant is invited to pick one of three doors (behind one is the prize) but the door is not yet opened.
The Monty Hall Problem

- The game show proceeds as follows: First the contestant is invited to pick one of three doors (behind one is the prize) but the door is not yet opened.
- Instead, the host – legendary Monty Hall – opens one of the other doors which is empty.
The Monty Hall Problem

- The game show proceeds as follows: First the contestant is invited to pick one of three doors (behind one is the prize) but the door is not yet opened.
- Instead, the host – legendary Monty Hall – opens one of the other doors which is empty.
- After that the contestant is given a last chance to stick with his/her door or to switch to the other closed one.
The Monty Hall Problem

- The game show proceeds as follows: First the contestant is invited to pick one of three doors (behind one is the prize) but the door is not yet opened.
- Instead, the host – legendary Monty Hall – opens one of the other doors which is empty.
- After that the contestant is given a last chance to stick with his/her door or to switch to the other closed one.
- Note that the host (knowing where the prize is) has always at least one door he can open.
Optimal Strategy: To Switch or not to Switch

\[ w_i = \text{win behind } i \quad p_i = \text{pick door } i \quad o_i = \text{Monty opens door } i \]
Certainty, Possibility, Probability

Certainty — Determinism  
Model: Definite Value  
e.g. $2 \in \mathbb{N}$
Certainty, Possibility, Probability

Certainty — Determinism
Model: Definite Value
e.g. $2 \in \mathbb{N}$

Possibility — Non-Determinism
Model: Set of Values
e.g. $\{2, 4, 6, 8, 10\} \in \mathcal{P}(\mathbb{N})$
Certainty — Determinism
Model: Definite Value
e.g. $2 \in \mathbb{N}$

Possibility — Non-Determinism
Model: Set of Values
e.g. $\{2, 4, 6, 8, 10\} \in \mathcal{P}(\mathbb{N})$

Probability — Probabilistic Non-Determinism
Model: Distribution (Measure)
e.g. $(0, 0, \frac{1}{5}, 0, \frac{1}{5}, 0, \ldots) \in \mathcal{V}(\mathbb{N})$
Given a finite set (universe) $\Omega$ (of states) we can construct the power set $\mathcal{P}(\Omega)$ of $\Omega$ easily as:

$$\mathcal{P}(\Omega) = \{ X \mid X \subseteq \Omega \}$$

Ordered by inclusion “$\subseteq$” this is the example of a lattice/order.
Structures: Power Sets

Given a finite set (universe) $\Omega$ (of states) we can construct the power set $\mathcal{P}(\Omega)$ of $\Omega$ easily as:

$$\mathcal{P}(\Omega) = \{ X \mid X \subseteq \Omega \}$$

Ordered by inclusion “$\subseteq$” this is the example of a lattice/order.

It can also be seen as the set of functions from $S$ into a two element set, thus $\mathcal{P}(\Omega) = 2^\Omega$:

$$\mathcal{P}(\Omega) = \{ \chi : \Omega \rightarrow \{0, 1\} \}$$
Structures: Power Sets

Given a finite set (universe) $\Omega$ (of states) we can construct the power set $\mathcal{P}(\Omega)$ of $\Omega$ easily as:

$$\mathcal{P}(\Omega) = \{X \mid X \subseteq \Omega\}$$

Ordered by inclusion “$\subseteq$” this is the example of a lattice/order.

It can also be seen as the set of functions from $S$ into a two element set, thus $\mathcal{P}(\Omega) = 2^\Omega$:

$$\mathcal{P}(\Omega) = \{\chi : \Omega \to \{0, 1\}\}$$

A priori, no major problems when $\Omega$ is (un)countable infinite.
Vector Spaces

Given a finite set $\Omega$ we can construct the (free) vector space $V(\Omega)$ of $\Omega$ as a tuple space (with $K$ a field like $\mathbb{R}$ or $\mathbb{C}$):

$$V(\Omega) = \{ \langle \omega, x_\omega \rangle | \omega \in \Omega, x_\omega \in K \} = \{ (x_\omega)_{\omega \in \Omega} | x_\omega \in K \}$$

As function spaces $V(\Omega)$ and $P(\Omega)$ are not so different:

$$V(\Omega) = \{ v: \Omega \to K \}$$

However, there are major topological problems when $\Omega$ is (un)countable infinite.
Structures: Vector Spaces

Vector Spaces = Abelian Additive Group + Quantities
Vector Spaces = Abelian Additive Group + Quantities

Given a finite set $\Omega$ we can construct the (free) vector space $\mathcal{V}(\Omega)$ of $\Omega$ as a tuple space (with $\mathbb{K}$ a field like $\mathbb{R}$ or $\mathbb{C}$):

$$\mathcal{V}(\Omega) = \{ \langle \omega, x_\omega \rangle \mid \omega \in \Omega, x_\omega \in \mathbb{K} \} = \{(x_\omega)_{\omega \in \Omega} \mid x_\omega \in \mathbb{K}\}$$
Structures: Vector Spaces

Vector Spaces = Abelian Additive Group + Quantities

Given a finite set $\Omega$ we can construct the (free) vector space $V(\Omega)$ of $\Omega$ as a tuple space (with $K$ a field like $\mathbb{R}$ or $\mathbb{C}$):

$$V(\Omega) = \{ \langle \omega, x_\omega \rangle \mid \omega \in \Omega, x_\omega \in K \} = \{ (x_\omega)_{\omega \in \Omega} \mid x_\omega \in K \}$$

As function spaces $V(\Omega)$ and $\mathcal{P}(\Omega)$ are not so different:

$$V(\Omega) = \{ v : \Omega \to K \}$$
Vector Spaces = Abelian Additive Group + Quantities

Given a finite set Ω we can construct the (free) vector space \( V(Ω) \) of Ω as a tuple space (with \( K \) a field like \( \mathbb{R} \) or \( \mathbb{C} \)):

\[
V(Ω) = \{ ⟨ω, x_ω⟩ | ω ∈ Ω, x_ω ∈ K \} = \{ (x_ω)_{ω∈Ω} | x_ω ∈ K \}
\]

As function spaces \( V(Ω) \) and \( P(Ω) \) are not so different:

\[
V(Ω) = \{ ν : Ω → K \}
\]

However, there are major topological problems when Ω is (un)countable infinite.
Tuple Spaces

Theorem
All finite dimensional vector spaces are isomorphic to the (finite) Cartesian product of the underlying field $\mathbb{K}^n$ (e.g. $\mathbb{R}^n$ or $\mathbb{C}^m$).

Finite dimensional vectors can always be represented via their coordinates with respect to a given base, e.g.

$$x = (x_1, x_2, x_3, \ldots, x_n)$$
$$y = (y_1, y_2, y_3, \ldots, y_n)$$

Algebraic Structure

$$\alpha x = (\alpha x_1, \alpha x_2, \alpha x_3, \ldots, \alpha x_n)$$
$$x + y = (x_1 + y_1, x_2 + y_2, x_3 + y_3, \ldots, x_n + y_n)$$
Introducing Probability in Programs

Various ways for introducing probabilities into programs:
Various ways for introducing probabilities into programs:

**Random Assignment**  The value a variable is assigned to is chosen randomly (according to some, e.g. uniform, probability distribution) from a set:

\[ x \sim \{1, 2, 3, 4\} \]
Introducing Probability in Programs

Various ways for introducing probabilities into programs:

**Random Assignment**  The value a variable is assigned to is chosen randomly (according to some, e.g. uniform, probability distribution) from a set:

\[ x = \{1, 2, 3, 4\} \]

**Probabilistic Choice**  There is a probabilistic choice between different instructions:

\[ \text{choose } 0.5 : (x := 0) \text{ or } 0.5 : (x := 1) \text{ ro} \]
**Syntactic Sugar**

One can show that a single “coin flipping” is enough.
Syntactic Sugar

One can show that a single “coin flipping” is enough.

Random choices and assignments can be interchanged:

$$x \in \{0, 1\}$$

is equivalent to (assuming a uniform distribution):

```plaintext
choose 0.5 : (x := 0) or 0.5 : (x := 1) ro
```

Syntactic Sugar

One can show that a single “coin flipping” is enough.

Random choices and assignments can be interchanged:

\[ x \, ?= \{0, 1\} \]

is equivalent to (assuming a uniform distribution):

\[
\text{choose } 0.5 : (x := 0) \text{ or } 0.5 : (x := 1) \text{ ro}
\]

Alternatively we also have

\[
\text{choose } 0.5 : S_1 \text{ or } 0.5 : S_2 \text{ ro}
\]

is equivalent to (also with other probability distributions):

\[
x \, ?= \{0, 1\}; \text{ if } (x > 0) \text{ then } S_1 \text{ else } S_2 \text{ fi}
\]
Probabilities as Ratios

Consider integer “weights” to express relative probabilities, e.g.

\[ \text{choose } \frac{1}{3} : S_1 \text{ or } \frac{2}{3} : S_2 \]
Probabilities as Ratios

Consider integer “weights” to express relative probabilities, e.g.

\[ \text{choose } \frac{1}{3} : S_1 \text{ or } \frac{2}{3} : S_2 \text{ ro} \]

is expressed equivalently as:

\[ \text{choose } 1 : (x := 0) \text{ or } 2 : (x := 1) \text{ ro} \]
Probabilities as Ratios

Consider integer “weights” to express relative probabilities, e.g.

\[
\text{choose } \frac{1}{3} : S_1 \text{ or } \frac{2}{3} : S_2 \text{ ro}
\]

is expressed equivalently as:

\[
\text{choose } 1 : (x := 0) \text{ or } 2 : (x := 1) \text{ ro}
\]

In general, for constant "weights" \( p \) and \( q \) (int), we translate

\[
\text{choose } p : S_1 \text{ or } q : S_2 \text{ ro}
\]

(by exploiting an implicit normalisation) into

\[
\text{choose } \frac{p}{p + q} : S_1 \text{ or } \frac{q}{p + q} : S_2 \text{ ro}
\]
**PWHILE – Concrete Syntax**

The syntax of statements $S$ is as follows:

$$S ::= \text{stop}$$  
$$\quad \text{skip}$$  
$$\quad x := e$$  
$$\quad x \neq r$$  
$$\quad S_1 ; S_2$$  
$$\quad \text{choose } p_1 : S_1 \text{ or } p_2 : S_2 \text{ ro}$$  
$$\quad \text{if } b \text{ then } S_1 \text{ else } S_2 \text{ fi}$$  
$$\quad \text{while } b \text{ do } S \text{ od}$$

We also allow for boolean expressions, i.e. $e$ is an arithmetic expression $a$ or a boolean expression $b$. The `choose` statement can be generalised to more than two alternatives.
**PWHILE** – Labelled Syntax

\[
S \ ::= \ [\text{stop}]^\ell \\
[\text{skip}]^\ell \\
[\text{x := e}]^\ell \\
[\text{x := r}]^\ell \\
S_1; S_2 \\
\text{choose}^\ell p_1 : S_1 \text{ or } p_2 : S_2 \text{ ro} \\
\text{if } [b]^\ell \text{ then } S_1 \text{ else } S_2 \text{ fi} \\
\text{while } [b]^\ell \text{ do } S \text{ od}
\]

Where the \( p_i \) are constants, representing choice probabilities. By \( r \) we denote a range/set, e.g. \( \{-1, 0, 1\} \), from which the value of \( x \) is chosen (based on a uniform distribution).
Evaluation of Expressions [Not for Exam]

\[
\sigma \ni \text{State} = (\text{Var} \rightarrow \mathbb{Z} \sqcup \mathbb{B})
\]
Evaluation of Expressions [Not for Exam]

\[ \sigma \ni \text{State} = (\text{Var} \rightarrow \mathbb{Z} \cup \text{B}) \]

Evaluation \( \mathcal{E} \) of expressions \( e \) in state \( \sigma \):

\[
\begin{align*}
\mathcal{E}(n)\sigma &= n \\
\mathcal{E}(x)\sigma &= \sigma(x) \\
\mathcal{E}(a_1 \odot a_2)\sigma &= \mathcal{E}(a_1)\sigma \odot \mathcal{E}(a_2)\sigma
\end{align*}
\]
Evaluation of Expressions [Not for Exam]

\[ \sigma \ni \text{State} = (\text{Var} \rightarrow \mathbb{Z} \cup \mathbb{B}) \]

Evaluation \( \mathcal{E} \) of expressions \( \varepsilon \) in state \( \sigma \):

\[
\begin{align*}
\mathcal{E}(n) & = n \\
\mathcal{E}(x) & = \sigma(x) \\
\mathcal{E}(a_1 \odot a_2) & = \mathcal{E}(a_1) \odot \mathcal{E}(a_2)
\end{align*}
\]

\[
\begin{align*}
\mathcal{E}(\text{true}) & = \text{tt} \\
\mathcal{E}(\text{false}) & = \text{ff} \\
\mathcal{E}(\text{not } b) & = \neg \mathcal{E}(b) \\
\ldots & = \ldots
\end{align*}
\]
pWhile – SOS Semantics I [Not for Exam]

R0 \( \langle \text{skip}, \sigma \rangle \Rightarrow_1 \langle \text{stop}, \sigma \rangle \)

R1 \( \langle \text{stop}, \sigma \rangle \Rightarrow_1 \langle \text{stop}, \sigma \rangle \)

R2 \( \langle x := e, \sigma \rangle \Rightarrow_1 \langle \text{stop}, \sigma[x \mapsto \mathcal{E}(e)\sigma] \rangle \)

R3’ \( \langle x? = r, \sigma \rangle \Rightarrow_1 \langle \text{stop}, \sigma[x \mapsto \{ r_i \in r \}] \rangle \)

R3_1 \[
\frac{\langle S_1, \sigma \rangle \Rightarrow_p \langle S'_1, \sigma' \rangle}{\langle S_1; S_2, \sigma \rangle \Rightarrow_p \langle S'_1; S_2, \sigma' \rangle}
\]

R3_2 \[
\frac{\langle S_1, \sigma \rangle \Rightarrow_p \langle \text{stop}, \sigma' \rangle}{\langle S_1; S_2, \sigma \rangle \Rightarrow_p \langle S_2, \sigma' \rangle}
\]
\( R4_1 \quad \langle \text{choose } p_1 : S_1 \text{ or } p_2 : S_2, \sigma \rangle \Rightarrow p_1 \langle S_1, \sigma \rangle \)

\( R4_2 \quad \langle \text{choose } p_1 : S_1 \text{ or } p_2 : S_2, \sigma \rangle \Rightarrow p_2 \langle S_2, \sigma \rangle \)

\( R5_1 \quad \langle \text{if } b \text{ then } S_1 \text{ else } S_2, \sigma \rangle \Rightarrow_1 \langle S_1, \sigma \rangle \quad \text{if } \mathcal{E}(b)\sigma = \text{tt} \)

\( R5_2 \quad \langle \text{if } b \text{ then } S_1 \text{ else } S_2, \sigma \rangle \Rightarrow_1 \langle S_2, \sigma \rangle \quad \text{if } \mathcal{E}(b)\sigma = \text{ff} \)

\( R6_1 \quad \langle \text{while } b \text{ do } S, \sigma \rangle \Rightarrow_1 \langle S; \text{ while } b \text{ do } S, \sigma \rangle \quad \text{if } \mathcal{E}(b)\sigma = \text{tt} \)

\( R6_2 \quad \langle \text{while } b \text{ do } S, \sigma \rangle \Rightarrow_1 \langle \text{stop}, \sigma \rangle \quad \text{if } \mathcal{E}(b)\sigma = \text{ff} \)
DTMC Semantics

Given a PWHILE program, consider any enumeration of all its configurations (= pairs of statements and state) $C_1, C_2, C_3, \ldots \in \textbf{Conf}$. Then

$$(T)_{ij} = \begin{cases} p & \text{if } C_i \Rightarrow^p C_j \\ 0 & \text{otherwise} \end{cases}$$
DTMC Semantics

Given a PWHILE program, consider any enumeration of all its configurations (= pairs of statements and state) $C_1, C_2, C_3, \ldots \in \text{Conf}$. Then

$$(T)_{ij} = \begin{cases} p & \text{if } C_i = \langle S, \sigma \rangle \Rightarrow p \ C_j = \langle S', \sigma' \rangle \\ 0 & \text{otherwise} \end{cases}$$
Given a PWHILE program, consider any enumeration of all its configurations (pairs of statements and state) $C_1, C_2, C_3, \ldots \in \text{Conf}$. Then

$$(T)_{ij} = \begin{cases} 
    p & \text{if } C_i \Rightarrow_p C_j \\
    0 & \text{otherwise}
\end{cases}$$

is the generator of a Discrete Time Markov Chain.
Given a PWHILE program, consider any enumeration of all its configurations (= pairs of statements and state) $C_1, C_2, C_3, \ldots \in \textbf{Conf}$. Then

$$\left( T \right)_{ij} = \begin{cases} p & \text{if } C_i \Rightarrow_p C_j \\ 0 & \text{otherwise} \end{cases}$$

is the generator of a Discrete Time Markov Chain.

Transitions are implemented as

$$d_n \cdot T$$

where $d_i$ is the probability distribution over $\textbf{Conf}$ at the $i$th step.
DTMC Semantics

Given a PWHILE program, consider any enumeration of all its configurations (= pairs of statements and state) $C_1, C_2, C_3, \ldots \in \text{Conf}$. Then

$$ (T)_{ij} = \begin{cases} 
\rho & \text{if } C_i \Rightarrow_{\rho} C_j \\
0 & \text{otherwise}
\end{cases} $$

is the generator of a Discrete Time Markov Chain.

Transitions are implemented as

$$ d_n \cdot T = \sum_i (d_n)_i \cdot T_{ij} $$

where $d_i$ is the probability distribution over $\text{Conf}$ at the $i$th step.
DTMC Semantics

Given a PWHILE program, consider any enumeration of all its configurations (= pairs of statements and state) $C_1, C_2, C_3, \ldots \in \textbf{Conf}$. Then

$$(T)_{ij} = \begin{cases} p & \text{if } C_i \Rightarrow_p C_j \\ 0 & \text{otherwise} \end{cases}$$

is the generator of a Discrete Time Markov Chain.

Transitions are implemented as

$$d_n \cdot T = \sum_i (d_n)_i \cdot T_{ij} = d_{n+1}$$

where $d_i$ is the probability distribution over $\textbf{Conf}$ at the $i$th step.
Let us investigate the possible transitions of the following labelled program (with $x \in \{0, 1\}$):

```plaintext
if \([x == 0]^1 \) then
  \([x := 0]^2\);
else
  \([x := 1]^3\);
end if;
[stop]^4
```
Example DTMC

\[
\begin{align*}
\langle x \mapsto 0, [x == 0]^1 \rangle & \quad \ldots \quad \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \\
\langle x \mapsto 0, [x:=0]^2 \rangle & \quad \ldots \quad \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \\
\langle x \mapsto 0, [x:=1]^3 \rangle & \quad \ldots \quad \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \\
\langle x \mapsto 0, [\text{stop}]^4 \rangle & \quad \ldots \quad \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \\
\langle x \mapsto 1, [x == 0]^1 \rangle & \quad \ldots \quad \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \\
\langle x \mapsto 1, [x:=0]^2 \rangle & \quad \ldots \quad \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \\
\langle x \mapsto 1, [x:=1]^3 \rangle & \quad \ldots \quad \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \\
\langle x \mapsto 1, [\text{stop}]^4 \rangle & \quad \ldots \quad \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}
\end{align*}
\]
Example Transition

\[
\begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{pmatrix}
\]

We get:

\[
\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
\]

This represents the (deterministic) transition step:

\[\langle x \mapsto \rightarrow 0, [x] := 1 \rangle \implies 1 \langle x \mapsto \rightarrow 1, [\text{stop}] \rangle\]
Example Transition

\[
\begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
\]

We get: \((0 0 0 0 0 0 0 0 1)\).

This represents the (deterministic) transition step:

\[
\langle x \mapsto 0, [x:=1]^3 \rangle \Rightarrow_1 \langle x \mapsto 1, [\text{stop}]^4 \rangle
\]
Linear Operator Semantics (LOS)

The matrix representation of the SOS semantics of a \texttt{PWHILE} program is not `compositional`.
Linear Operator Semantics (LOS)

The matrix representation of the SOS semantics of a PWHILE program is not ‘compositional’.

In order to be able to analyse programs by analysing its parts, a more useful semantics is one resulting from the composition of different linear operators each expressing a particular operation contributing to the overall behaviour of the program.
The Space of Configurations

For a PWHILE program $S$ we can identify configurations with elements in
The Space of Configurations

For a PWHILE program $S$ we can identify configurations with elements in

$$\text{Dist}(\text{State} \times \text{Lab}) \subseteq \mathcal{V}(\text{State} \times \text{Lab}).$$
The Space of Configurations

For a PWHILE program $S$ we can identify configurations with elements in

$$\text{Dist}(\text{State} \times \text{Lab}) \subseteq \mathcal{V}(\text{State} \times \text{Lab}).$$

Assuming $\nu = |\text{Var}|$ finite,

$$\text{State} = (\mathbb{Z} + B)^\nu = \text{Value}_1 \times \text{Value}_2 \ldots \times \text{Value}_\nu$$

with $\text{Value}_i = \mathbb{Z}(= \mathbb{Z})$ or $\text{Value}_i$. 
The Space of Configurations

For a $\textbf{PWHILE}$ program $S$ we can identify configurations with elements in

$$\text{Dist}(\text{State} \times \text{Lab}) \subseteq \mathcal{V}(\text{State} \times \text{Lab}).$$

Assuming $v = \mid \text{Var} \mid$ finite,

$$\text{State} = (Z + B)^v = \text{Value}_1 \times \text{Value}_2 \ldots \times \text{Value}_v$$

with $\text{Value}_i = Z (= Z)$ or $\text{Value}_i$.

Thus, we can represent the space of configurations as

$$\text{Dist}(\text{Value}_1 \times \ldots \times \text{Value}_v \times \text{Lab}) \subseteq$$

$$\subseteq \mathcal{V}(\text{Value}_1 \times \ldots \times \text{Value}_v \times \text{Lab})$$

$$= \mathcal{V}(\text{Value}_1) \otimes \ldots \otimes \mathcal{V}(\text{Value}_v) \otimes \mathcal{V}(\text{Lab}).$$
Tensor Product

Given a $n \times m$ matrix $A$ and a $k \times l$ matrix $B$:

$$A = \begin{pmatrix} a_{11} & \ldots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \ldots & a_{nm} \end{pmatrix} \quad B = \begin{pmatrix} b_{11} & \ldots & b_{1l} \\ \vdots & \ddots & \vdots \\ b_{k1} & \ldots & b_{kl} \end{pmatrix}$$

Special cases are square matrices ($n = m$ and $k = l$) and vectors (row $n = k = 1$, column $m = l = 1$).
Tensor Product

Given a $n \times m$ matrix $A$ and a $k \times l$ matrix $B$:

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{pmatrix}, \quad B = \begin{pmatrix} b_{11} & \cdots & b_{1l} \\ \vdots & \ddots & \vdots \\ b_{k1} & \cdots & b_{kl} \end{pmatrix}$$

The tensor product $A \otimes B$ is a $nk \times ml$ matrix:

$$A \otimes B = \begin{pmatrix} a_{11}B & \cdots & a_{1m}B \\ \vdots & \ddots & \vdots \\ a_{n1}B & \cdots & a_{nm}B \end{pmatrix}$$
Tensor Product

Given a \( n \times m \) matrix \( A \) and a \( k \times l \) matrix \( B \):

\[
A = \begin{pmatrix}
  a_{11} & \cdots & a_{1m} \\
  \vdots & \ddots & \vdots \\
  a_{n1} & \cdots & a_{nm}
\end{pmatrix}
\quad B = \begin{pmatrix}
  b_{11} & \cdots & b_{1l} \\
  \vdots & \ddots & \vdots \\
  b_{k1} & \cdots & b_{kl}
\end{pmatrix}
\]

The tensor product \( A \otimes B \) is a \( nk \times ml \) matrix:

\[
A \otimes B = \begin{pmatrix}
  a_{11}B & \cdots & a_{1m}B \\
  \vdots & \ddots & \vdots \\
  a_{n1}B & \cdots & a_{nm}B
\end{pmatrix}
\]

Special cases are square matrices (\( n = m \) and \( k = l \)) and vectors (row \( n = k = 1 \), column \( m = l = 1 \)).
Tensor Product Properties

For tensor product of square matrices (linear operators):

1. The bilinearity property:
   \[(\alpha M + \alpha' M') \otimes (\beta N + \beta' N') = \alpha\beta (M \otimes N) + \alpha\beta' (M \otimes N') + \alpha' \beta (M' \otimes N) + \alpha' \beta' (M' \otimes N')\]
   \[\alpha, \alpha', \beta, \beta' \in \mathbb{R}, M, M' m \times m \text{ matrices}, N, N' n \times n \text{ matrices}.\]

2. We have, with \(v \in \mathbb{R}^m\) and \(w \in \mathbb{R}^n\):
   \[(M \otimes N)(v \otimes w) = (M(v)) \otimes (N(w))\]
   \[(M \otimes N)(M' \otimes N') = (MM') \otimes (NN')\]

3. If \(M\) and \(N\) are invertible so is \(M \otimes N\), and:
   \[(M \otimes N)^{-1} = M^{-1} \otimes N^{-1}\]
Tensor Product Properties

For tensor product of square matrices (linear operators):

1. The **bilinearity** property:

\[
(\alpha M + \alpha' M') \otimes (\beta N + \beta' N') = \\
= \alpha \beta (M \otimes N) + \alpha \beta' (M \otimes N') + \alpha' \beta (M' \otimes N) + \alpha' \beta' (M' \otimes N')
\]

\[\alpha, \alpha', \beta, \beta' \in \mathbb{R}, \ M, M' \ m \times m \ matrices \ N, N' \ n \times n \ matrices.\]
Tensor Product Properties

For tensor product of square matrices (linear operators):

1. The **bilinearity** property:

\[
(\alpha M + \alpha' M') \otimes (\beta N + \beta' N') = \\
= \alpha \beta (M \otimes N) + \alpha \beta' (M \otimes N') + \alpha' \beta (M' \otimes N) + \alpha' \beta' (M' \otimes N')
\]

\[\alpha, \alpha', \beta, \beta' \in \mathbb{R}, \ M, M' \ m \times m \text{ matrices } N, N' \ n \times n \text{ matrices.}\]

2. We have, with \(v \in \mathbb{R}^m\) and \(w \in \mathbb{R}^n\):

\[
(M \otimes N)(v \otimes w) = (M(v)) \otimes (N(w))
\]

\[
(M \otimes N)(M' \otimes N') = (MM') \otimes (NN')
\]
Tensor Product Properties

For tensor product of square matrices (linear operators):

1. The \textbf{bilinearity} property:

\[(\alpha M + \alpha' M') \otimes (\beta N + \beta' N') =\]
\[= \alpha \beta (M \otimes N) + \alpha \beta' (M \otimes N') + \alpha' \beta (M' \otimes N) + \alpha' \beta' (M' \otimes N')\]

\(\alpha, \alpha', \beta, \beta' \in \mathbb{R}, M, M' \text{ } m \times m \text{ matrices } N, N' \text{ } n \times n \text{ matrices.}\)

2. We have, with \(v \in \mathbb{R}^m\) and \(w \in \mathbb{R}^n:\)

\[(M \otimes N)(v \otimes w) = (M(v)) \otimes (N(w))\]
\[(M \otimes N)(M' \otimes N') = (MM') \otimes (NN')\]

3. If \(M\) and \(N\) are invertible so is \(M \otimes N\), and:

\[(M \otimes N)^{-1} = M^{-1} \otimes N^{-1}\]
Transitions and Generator of DTMC (1) - Deterministic

\[
\begin{bmatrix}
0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
= T_{31} / 1
\]
Transitions and Generator of DTMC (1) - Deterministic

\[
\begin{pmatrix}
0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1
\end{pmatrix} = T
\]
Transitions and Generator of DTMC (2) - Probabilistic

\[
\begin{pmatrix}
0 & \frac{1}{3} & \frac{2}{3} & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & \frac{1}{2} & \frac{1}{2} \\
0 & 0 & 0 & 1
\end{pmatrix} = T
\]
Transitions and Generator of DTMC (3)

\[
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
1 \\
0 \\
0 \\
0
\end{pmatrix}^t
\begin{pmatrix}
0 & 1/3 & 2/3 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1/2 & 1/2 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]
Transitions and Generator of DTMC (4)

\[
\begin{bmatrix}
1 \\
0 \\
0 \\
0
\end{bmatrix}
\begin{bmatrix}
0 & \frac{1}{3} & \frac{2}{3} & 0 \\
0 & 0 & \frac{1}{2} & \frac{1}{2} \\
0 & 0 & 0 & 1
\end{bmatrix}
= 
\begin{bmatrix}
0 \\
\frac{1}{3} \\
\frac{2}{3} \\
0
\end{bmatrix}
\]
Transitions and Generator of DTMC (5)

\[
\begin{pmatrix}
1 & 0 & 0 & 1 \\
0 & 1/3 & 2/3 & 0 \\
0 & 0 & 1/2 & 1/2 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
0 \\
0 \\
0 \\
1
\end{pmatrix}^t
\begin{pmatrix}
0 \\
0 \\
0 \\
1
\end{pmatrix}^\infty
\]
Combination of Steps

We can combine single steps to construct a transition graph.
Combination of Steps

We can combine single steps to construct a transition graph.

\[(E(m, n))_{ij} = \begin{cases} 
1 & \text{if } m = i \land n = j \\
0 & \text{otherwise.} 
\end{cases}\]
Combination of Steps

We can combine single steps to construct a transition graph.

\[
\begin{pmatrix}
0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 \\
\end{pmatrix} = T
\]

\[(E(m, n))_{ij} = \begin{cases} 
1 & \text{if } m = i \land n = j \\
0 & \text{otherwise.}
\end{cases}\]
Combination of Steps

We can combine single steps to construct a transition graph.

\[
\begin{pmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{pmatrix}
= \begin{cases}
E(1, 2) \\
E(m, n) \\
\end{cases}
\]

\[
(E(m, n))_{ij} = \begin{cases}
1 & \text{if } m = i \land n = j \\
0 & \text{otherwise.}
\end{cases}
\]
Combination of Steps

We can combine single steps to construct a transition graph.

\[
\begin{pmatrix}
0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{pmatrix}
= \left\{
\begin{array}{c}
E(1, 2) \\
E(1, 3)
\end{array}
\right.
\]

\[
(E(m, n))_{ij} = \begin{cases} 
1 & \text{if } m = i \land n = j \\
0 & \text{otherwise.}
\end{cases}
\]
Combination of Steps

We can combine single steps to construct a transition graph.

\[
\begin{pmatrix}
0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix}
= \left\{
\begin{aligned}
E(1, 2) \\
E(1, 3) \\
E(2, 4)
\end{aligned}
\right\
\]

\[
(E(m, n))_{ij} = \begin{cases}
1 & \text{if } m = i \land n = j \\
0 & \text{otherwise.}
\end{cases}
\]
Combination of Steps

We can combine single steps to construct a transition graph.

\[
\begin{pmatrix}
0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0
\end{pmatrix}
\]

\[
= E(1, 2) + E(1, 3) + E(2, 4) + E(3, 4)
\]

\[
(E(m, n))_{ij} = \begin{cases} 
1 & \text{if } m = i \land n = j \\
0 & \text{otherwise.}
\end{cases}
\]
Combination of Steps

We can combine single steps to construct a transition graph.

\[
\begin{pmatrix}
0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0
\end{pmatrix}
= \begin{cases}
E(1, 2) \\
E(1, 3) \\
E(2, 4) \\
E(3, 4) \\
E(3, 3)
\end{cases}
\]

\[
(E(m, n))_{ij} = \begin{cases}
1 & \text{if } m = i \land n = j \\
0 & \text{otherwise.}
\end{cases}
\]
Combination of Steps

We can combine single steps to construct a transition graph.

\[(E(m, n))_{ij} = \begin{cases} 1 & \text{if } m = i \land n = j \\ 0 & \text{otherwise.} \end{cases}\]
Probabilistic Transitions

Constructing the matrix for probabilistic transitions:

\[
\begin{pmatrix}
0 & \frac{1}{3} & \frac{2}{3} & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & \frac{1}{2} & \frac{1}{2} \\
0 & 0 & 0 & 1
\end{pmatrix} = T
\]
Probabilistic Transitions

Constructing the matrix for probabilistic transitions:

\[
\begin{pmatrix}
0 & \frac{1}{3} & \frac{2}{3} & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & \frac{1}{2} & \frac{1}{2} \\
0 & 0 & 0 & 1 \\
\end{pmatrix}
= \mathbf{T}
\]
Probabilistic Transitions

Constructing the matrix for probabilistic transitions:

\[
\begin{pmatrix}
0 & \frac{1}{3} & \frac{2}{3} & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & \frac{1}{2} & 1 \\
0 & 0 & 0 & 1 \\
\end{pmatrix} = T
\]

\[T = \frac{1}{3} E(1, 2)\]
Probabilistic Transitions

Constructing the matrix for probabilistic transitions:

\[ T = \frac{1}{3} E(1, 2) + \frac{2}{3} E(1, 3) \]

\[
\begin{pmatrix}
0 & \frac{1}{3} & \frac{2}{3} & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & \frac{1}{2} \\
0 & 0 & 0 & \frac{1}{2}
\end{pmatrix} = T
\]
Probabilistic Transitions

Constructing the matrix for probabilistic transitions:

\[ T = \frac{1}{3} E(1,2) + \frac{2}{3} E(1,3) + E(2,4) \]
Probabilistic Transitions

Constructing the matrix for probabilistic transitions:

$$
T = \frac{1}{3} E(1, 2) + \frac{2}{3} E(1, 3) + E(2, 4) + \frac{1}{2} E(3, 4)
$$

$$
\begin{pmatrix}
0 & \frac{1}{3} & \frac{2}{3} & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & \frac{1}{2} & \frac{1}{2} \\
0 & 0 & 0 & 1
\end{pmatrix}
= T
$$
Probabilistic Transitions

Constructing the matrix for probabilistic transitions:

\[
\begin{pmatrix}
0 & \frac{1}{3} & \frac{2}{3} & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & \frac{1}{2} & \frac{1}{2} \\
0 & 0 & 0 & 1
\end{pmatrix} = T
\]

\[
T = \frac{1}{3}E(1, 2) + \frac{2}{3}E(1, 3) + E(2, 4) + \frac{1}{2}E(3, 4) + \frac{1}{2}E(3, 3)
\]
Constructing the matrix for probabilistic transitions:

\[ T = \frac{1}{3} E(1, 2) + \frac{2}{3} E(1, 3) + E(2, 4) + \frac{1}{2} E(3, 4) + \frac{1}{2} E(3, 3) + E(4, 4) \]
Probabilistic Transitions

Constructing the matrix for probabilistic transitions:

\[
\begin{pmatrix}
0 & \frac{1}{3} & \frac{2}{3} & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & \frac{1}{2} & \frac{1}{2} \\
0 & 0 & 0 & 1
\end{pmatrix} = T
\]

\[
T = \frac{1}{3} E(1, 2) + \frac{2}{3} E(1, 3) + E(2, 4) + \frac{1}{2} E(3, 4) + \frac{1}{2} E(3, 3) + E(4, 4)
\]
"Turtle" Graphics

Consider a "(probabilistic) turtle graphics" with up/down and left/right moves done simultaneously and probabilistically.

The (classical) configuration space is $\{1,\ldots,8\} \times \{1,\ldots,4\}$.

To describe any probabilistic situation, i.e. joint distribution, we need $8 \times 4 = 32$ probabilities, not just $8 + 4 = 12$.

We consider $\mathbb{R}^8 \otimes \mathbb{R}^4 = \mathbb{R}^{32}$ as probabilistic configuration space rather than $\mathbb{R}^8 \oplus \mathbb{R}^4 = \mathbb{R}^{12}$, i.e. just the marginal distributions.
"Turtle" Graphics

Consider a "(probabilistic) turtle graphics" with up/down and left/right moves done simultaneously and probabilistically.

The (classical) configuration space is \( \{1, \ldots, 8\} \times \{1, \ldots, 4\} \).

To describe any probabilistic situation, i.e. joint distribution, we need \( 8 \times 4 = 32 \) probabilities, not just \( 8 + 4 = 12 \).
"Turtle" Graphics

Consider a "(probabilistic) turtle graphics" with up/down and left/right moves done simultaneously and probabilistically.

The (classical) configuration space is \( \{1, \ldots, 8\} \times \{1, \ldots, 4\} \).

To describe any probabilistic situation, i.e. joint distribution, we need \( 8 \times 4 = 32 \) probabilities, not just \( 8 + 4 = 12 \).

We consider \( \mathbb{R}^8 \otimes \mathbb{R}^4 = \mathbb{R}^{32} \) as probabilistic configuration space rather than \( \mathbb{R}^8 \oplus \mathbb{R}^4 = \mathbb{R}^{12} \), i.e. just the marginal distributions.
Moves in "Turtle" Graphics

Consider only horizontal moves over eight possible positions.
Moves in "Turtle" Graphics

Consider only horizontal moves over eight possible positions.

1 2 3 4 5 6 7 8
Moves in "Turtle" Graphics

Consider only horizontal moves over eight possible positions.

1  2  3  4  5  6  7  8
Moves in "Turtle" Graphics

Consider only horizontal moves over eight possible positions.

Move from 1 to 2: $E(1, 2)$
Moves in "Turtle" Graphics

Consider only horizontal moves over eight possible positions.

Move from 3 to 7: $E(3, 7)$
Consider only horizontal moves over eight possible positions.

Move from 2 to 7 or 8: $E(2, 7) + E(2, 8)$
Consider only horizontal moves over eight possible positions.

Move from 2 to 7 or 8: $E(2, 7) + E(2, 8)$ or $\frac{1}{2}E(2, 7) + \frac{1}{2}E(2, 8)$
Moves in "Turtle" Graphics

Consider only horizontal moves over eight possible positions.

Move from 2 to 7 or 8: $E(2, 7) + E(2, 8)$ or $\frac{1}{2}E(2, 7) + \frac{1}{2}E(2, 8)$

Similar representation also for vertical moves.
"Parallel" Execution: $x \in \{1, \ldots, 8\}$ and $y \in \{1, \ldots, 4\}$

Describe the effect $M$ on $x$ and the change of $y$ described by $N$, then the combined effect on $\langle x, y \rangle$ is given by $M \otimes N$. 
"Parallel" Execution: \( x \in \{1, \ldots, 8\} \) and \( y \in \{1, \ldots, 4\} \)

Describe the effect \( M \) on \( x \) and the change of \( y \) described by \( N \), then the combined effect on \( \langle x, y \rangle \) is given by \( M \otimes N \).

From \((1, 1)\) move 1 left and 3 up: \( E(1, 2) \otimes E(1, 4) \)
"Parallel" Execution: $x \in \{1, \ldots, 8\}$ and $y \in \{1, \ldots, 4\}$

Describe the effect $M$ on $x$ and the change of $y$ described by $N$, then the combined effect on $\langle x, y \rangle$ is given by $M \otimes N$.

From $(7, 3)$ move $(4, 2)$: $E(7, 4) \otimes E(3, 2)$
"Parallel" Execution: $x \in \{1, \ldots, 8\}$ and $y \in \{1, \ldots, 4\}$

Describe the effect $M$ on $x$ and the change of $y$ described by $N$, then the combined effect on $\langle x, y \rangle$ is given by $M \otimes N$.

From $(7, 3)$ to $(4, 2)/(7, 2)$: $E(7, 4) \otimes E(3, 2) + E(7, 7) \otimes E(3, 1)$
"Parallel" Execution: $x \in \{1, \ldots, 8\}$ and $y \in \{1, \ldots, 4\}$

Describe the effect $M$ on $x$ and the change of $y$ described by $N$, then the combined effect on $\langle x, y \rangle$ is given by $M \otimes N$.

From $(5, 2)$ move to all one right: $E(5, 6) \otimes \left( \sum_{i=1}^{4} E(2, i) \right)$
Assume $x \in 1,..,8$; How do statements change its value?
Transfer Functions (Edge Effects): Assignment

Assume $x \in 1, \ldots, 8$; How do statements change its value?

$x := 4$
Transfer Functions (Edge Effects): Assignment

Assume $x \in 1, \ldots, 8$; How do statements change its value?

$x := 4$
Transfer Functions (Edge Effects): Assignment

Assume $x \in 1, \ldots, 8$; How do statements change its value?

$x := 4$

Thus, the LOS of the statement is $\left[ x := 4 \right] = U(x ← 4)$.
Assume \( x \in 1, \ldots, 8 \); How do statements change its value?

\[ x := 4 \]
Transfer Functions (Edge Effects): Assignment

Assume $x \in 1, \ldots, 8$; How do statements change its value?

$x := 4$

Thus, the LOS of the statement is $[x := 4] = U(x \leftarrow 4)$. 
Transfer Functions (Edge Effects): Assignment

Assume $x \in 1, \ldots, 8$; How do statements change its value?

$x := 4$

Thus, the LOS of the statement is $x := 4 = U(x \leftarrow 4)$.
Transfer Functions (Edge Effects): Assignment

Assume $x \in 1, \ldots, 8$; How do statements change its value?

$x := 4$

Thus, the LOS of the statement is $[x := 4] = U(x ← 4)$. 

The diagram shows the edge effect of the assignment statement on the variables $1$ through $8$. The arrow indicates the direction of the edge effect, with the value $4$ assigned to $x$.
Transfer Functions (Edge Effects): Assignment

Assume $x \in 1, \ldots, 8$; How do statements change its value?

$x := 4$

$x := 4$ gives

$$
\begin{pmatrix}
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0
\end{pmatrix}
$$

Thus, the LOS of the statement $x := 4$ is $U(x \leftarrow 4)$. 

$x := 4$
Transfer Functions (Edge Effects): Assignment

Assume $x \in 1, \ldots, 8$; How do statements change its value?

$x := 4$
Transfer Functions (Edge Effects): Assignment

Assume $x \in 1, \ldots, 8$; How do statements change its value?

$x := 4$ gives $U(x \leftarrow 4) =$

$$
\begin{pmatrix}
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 
\end{pmatrix}
$$
Transfer Functions (Edge Effects): Assignment

Assume \( x \in 1,\ldots,8 \); How do statements change its value?

Thus, the LOS of the statement is \([x := 4] = U(x \leftarrow 4)\).
Transfer Functions (Edge Effects): Shift

Assume $x \in 1, \ldots, 8$; How do statements change its value?

$1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8$
Transfer Functions (Edge Effects): Shift

Assume $x \in 1,\ldots,8$; How do statements change its value?

$x := x + 1$
Transfer Functions (Edge Effects): Shift

Assume $x \in 1, .., 8$; How do statements change its value?

$x := x + 1$

\[
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

The LOS of the statement is $[\{ x := x + 1 \}] = U(x ← x + 1)$.

To avoid "overflow": actually $[\{ x := (x - 1 + 1 \mod 8) + 1 \}]$. 

42 / 1
Transfer Functions (Edge Effects): Shift

Assume $x \in 1, \ldots, 8$; How do statements change its value?

$x := x + 1$

The LOS of the statement is $[x := x + 1] = U(x \leftarrow x + 1)$. To avoid "overflow": actually $[x := ((x - 1) + 1 \mod 8) + 1]$. 

$x := x + 1$
Transfer Functions (Edge Effects): Shift

Assume $x \in 1, ..., 8$; How do statements change its value?

$$x := x + 1$$
Transfer Functions (Edge Effects): Shift

Assume \( x \in 1, \ldots, 8 \); How do statements change its value?

\[
x := x + 1
\]

The LOS of the statement is \([ [ x := x + 1 ] ] = U(x ← x + 1)\).

To avoid "overflow": actually \([ [ x := ((x - 1) + 1) \mod 8 + 1 ] ]\).
Transfer Functions (Edge Effects): Shift

Assume $x \in 1, .., 8$; How do statements change its value?

$x := x + 1$

$\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}$

The LOS of the statement is $[x := x + 1] = U(x ← x + 1)$.

To avoid "overflow": actually $[x := ((x - 1) + 1) \mod 8] + 1]$. 

42 / 1
Transfer Functions (Edge Effects): Shift

Assume $x \in 1, \ldots, 8$; How do statements change its value?

$x := x + 1$

The LOS of the statement is $[x := x + 1] = U(x ← x + 1)$.
Transfer Functions (Edge Effects): Shift

Assume $x \in 1, \ldots, 8$; How do statements change its value?

\[ x := x + 1 \]
Transfer Functions (Edge Effects): Shift

Assume $x \in 1, .., 8$; How do statements change its value?

$x := x + 1$

\[
\begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]

The LOS of the statement is $\left[ x := x + 1 \right] = U(x \leftarrow x + 1)$.

To avoid "overflow": actually $\left[ x := \left( (x - 1) + 1 \mod 8 \right) + 1 \right]$. 

Transfer Functions (Edge Effects): Shift

Assume $x \in 1, \ldots, 8$; How do statements change its value?

$x := x + 1$ gives

$$
\begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
$$
Assume $x \in 1, \ldots, 8$; How do statements change its value?

The LOS of the statement is $[[x := x + 1]] = U(x \leftarrow x + 1)$. To avoid “overflow”: actually $[[x := ((x - 1) + 1 \mod 8) + 1]]$. 
Transfer Functions (Edge Effects): Random Assign

Assume $x \in 1, \ldots, 8$; How do statements change its value?

1 2 3 4 5 6 7 8
Transfer Functions (Edge Effects): Random Assign

Assume $x \in 1, \ldots, 8$; How do statements change its value?

$x \in \{4, 5\}$
Transfer Functions (Edge Effects): Random Assign

Assume $x \in 1, \ldots, 8$; How do statements change its value?

$x \implies \{4, 5\}$
Transfer Functions (Edge Effects): Random Assign

Assume $x \in 1, \ldots, 8$; How do statements change its value?

\[
\begin{pmatrix}
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0
\end{pmatrix}
\]

So the LOS is $x = \{4, 5\}$
Transfer Functions (Edge Effects): Random Assign

Assume $x \in 1, \ldots, 8$; How do statements change its value?

$x \, ? \, = \, \{4, 5\}$
Transfer Functions (Edge Effects): Random Assign

Assume $x \in 1, \ldots, 8$; How do statements change its value?

$x \ ? = \ \{4, 5\}$
Transfer Functions (Edge Effects): Random Assign

Assume \( x \in 1, \ldots, 8 \); How do statements change its value?

\[ x \in \{4, 5\} \]
Transfer Functions (Edge Effects): Random Assign

Assume $x \in 1, \ldots, 8$; How do statements change its value?

$x \ ? = \ \{4, 5\}$
Assume $x \in 1, \ldots, 8$; How do statements change its value?

$x ? = \{4, 5\}$
Transfer Functions (Edge Effects): Random Assign

Assume $x \in 1, \ldots, 8$; How do statements change its value?

$x? = \{4, 5\}$
Transfer Functions (Edge Effects): Random Assign

Assume $x \in 1, \ldots, 8$; How do statements change its value?

$x = \{4, 5\}$ gives

$$
\begin{pmatrix}
0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
\end{pmatrix}
$$

So the LOS is

$$
\begin{pmatrix}
0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
\end{pmatrix}
$$
Transfer Functions (Edge Effects): Random Assign

Assume $x \in 1, \ldots, 8$; How do statements change its value?

So the LOS is $[x = \{4, 5\}] = \frac{1}{2}U(x \leftarrow 4) + \frac{1}{2}U(x \leftarrow 5)$. 
Using the Linear Operators

We have now as states probability distributions over possible values $\sigma \in \mathcal{D}(\text{Value})$ rather than classical states $s \in \text{Value}$.
Using the Linear Operators

We have now as states probability distributions over possible values $\sigma \in \mathcal{D}(\text{Value})$ rather than classical states $s \in \text{Value}$

We can compute what happens to classical states, e.g.

$$(0, 1, 0, 0, 0, 0, 0, 0) \cdot [\{x := 4\}] = (0, 0, 0, 1, 0, 0, 0, 0)$$

$$(0, 1, 0, 0, 0, 0, 0, 0) \cdot [\{x? = \{4, 5\}\}] = (0, 0, 0, \frac{1}{2}, \frac{1}{2}, 0, 0, 0)$$
Using the Linear Operators

We have now as states probability distributions over possible values $\sigma \in \mathcal{D}(\text{Value})$ rather than classical states $s \in \text{Value}$.

We can compute what happens to classical states, e.g.

\[
(0, 1, 0, 0, 0, 0, 0, 0) \cdot [x := 4] = (0, 0, 0, 1, 0, 0, 0, 0)
\]

\[
(0, 1, 0, 0, 0, 0, 0, 0) \cdot [x? = \{4, 5\}] = (0, 0, 0, \frac{1}{2}, \frac{1}{2}, 0, 0, 0)
\]

but also what happens with distributions, e.g.

\[
(0, \frac{2}{3}, 0, 0, \frac{1}{3}, 0, 0, 0) \cdot [x := x + 1] = (0, 0, \frac{2}{3}, 0, 0, \frac{1}{3}, 0, 0)
\]
Using the Linear Operators

We have now as states probability distributions over possible values \( \sigma \in D(\text{Value}) \) rather than classical states \( s \in \text{Value} \).

We can compute what happens to classical states, e.g.

\[
(0, 1, 0, 0, 0, 0, 0) \cdot [x := 4] = (0, 0, 0, 1, 0, 0, 0, 0)
\]
\[
(0, 1, 0, 0, 0, 0, 0) \cdot [x? = \{4, 5\}] = (0, 0, 0, 0, 1, 0, 0, 0)
\]

but also what happens with distributions, e.g.

\[
(0, \frac{2}{3}, 0, 0, \frac{1}{3}, 0, 0, 0) \cdot [x := x + 1] = (0, 0, \frac{2}{3}, 0, 0, \frac{1}{3}, 0, 0)
\]

and we can combine effects (to the same variable), e.g.

\[
[x? = \{4, 5\}] = \frac{1}{2} [x := 4] + \frac{1}{2} [x := 5]
\]
Putting Things Together

We can use the tensor product construction to combine the effects on different variables. For $x \in \{1..8\}$ and $y \in \{1,..4\}$

$$[x? = \{2, 4, 6, 8\}] = \frac{1}{4} \sum_{k=1}^{4} U(x \leftarrow 2k) \otimes I$$

$$[y := 3] = I \otimes U(y \leftarrow 3)$$

The execution of “$x? = \{2, 4, 6, 8\}; y := 3$” is implemented by

$$[x? = \{2, 4, 6, 8\}; y := 3] = \left( \frac{1}{4} \sum_{k=1}^{4} U(x \leftarrow 2k) \otimes I \right) (I \otimes U(y \leftarrow 3))$$

$$= \frac{1}{4} \sum_{k=1}^{4} U(x \leftarrow 2k) \otimes U(y \leftarrow 3)$$
"Turtle" Execution

\[ x? = \{2, 4, 6, 8\}; \ y := 3 \]

\[ = \frac{1}{4} \sum_{k=1}^{4} U(x \leftarrow 2k) \otimes U(y \leftarrow 3) \]

\[ = \frac{1}{4} \begin{pmatrix}
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
\end{pmatrix} \otimes \begin{pmatrix}
0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 \\
\end{pmatrix} \]
Conditionals

Consider conditional jumps or statements, e.g.

\[
\text{if } \text{even}(x) \text{ then } x := x/2 \text{ else } y := y + 1 \text{ fi}
\]
Conditionals

Consider conditional jumps or statements, e.g.

\[
\text{if } \text{even}(x) \text{ then } x := x/2 \text{ else } y := y + 1 \fi
\]

The branches have the following LOS:

\[
[x := x/2] = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
\end{pmatrix} \otimes I
\]

Note: To avoid errors, $a/b = \lceil a/b \rceil$ and $a + b = a + b \mod n$. 
Consider conditional jumps or statements, e.g.

\[
\text{if } \text{even}(x) \text{ then } x := x/2 \text{ else } y := y + 1 \text{ fi}
\]

\[
[y := y + 1] = I \otimes \begin{pmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0
\end{pmatrix}
\]
Consider conditional jumps or statements, e.g.

\[
\text{if even}(x) \ \text{then} \ x := x/2 \ \text{else} \ y := y + 1 \ \text{fi}
\]

Note: To avoid errors \( a/b = \lceil a/b \rceil \) and \( a + b = a + b \mod n \).
Tests and Distribution Splitting

We represent the filter for testing if \( x \) is even by a projection:

\[
P(\text{even}(x)) = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix} \otimes I
\]

Its negation is represented by:

\[
P(\neg\text{even}(x)) = P(\text{even}(x))^\perp = I - P(\text{even}(x)).
\]
Using Tests

The semantics of a conditional is given by applying the semantics of the branches to the filtered (probabilistic) states and to combine the results. In our example:

\[
\begin{align*}
\text{if } \text{even}(x) \text{ then } x &:= x/2 \text{ else } y + 1 \text{ fi} = \\
&= \text{P}(\text{even}(x)) \cdot \text{[} x := x/2 \text{]} + \text{P}(\text{even}(x))^\perp \cdot \text{[} y := y + 1 \text{]}
\end{align*}
\]

Given state where \( x \) has with probability \( \frac{1}{2} \) values 3 and 6, and \( y \) value 2, i.e. \( \sigma_0 = (0, 0, \frac{1}{2}, 0, 0, \frac{1}{2}, 0, 0) \otimes (0, 1, 0, 0) \) then

\[
\begin{align*}
\sigma_0 \cdot \text{P}(\text{even}(x)) &= (0, 0, 0, 0, 0, \frac{1}{2}, 0, 0) \otimes (0, 1, 0, 0) \\
&= \frac{1}{2} \cdot (0, 0, 0, 0, 0, 1, 0, 0) \otimes (0, 1, 0, 0) \\
\sigma_0 \cdot \text{P}(\text{even}(x))^\perp &= (0, 0, \frac{1}{2}, 0, 0, 0, 0, 0) \otimes (0, 1, 0, 0) \\
&= \frac{1}{2} \cdot (0, 0, 1, 0, 0, 0, 0, 0) \otimes (0, 1, 0, 0)
\end{align*}
\]
Semantics of Conditionals

Applying the semantics of both branches gives us:

\[
\sigma_0 \cdot \mathbf{P}\left(even(x)\right) \cdot \left[x := x/2\right] = \\
= \left(0, 0, \frac{1}{2}, 0, 0, 0, 0\right) \otimes \left(0, 1, 0, 0\right)
\]

\[
\sigma_0 \cdot \mathbf{P}\left(even(x)\right)^\perp \cdot \left[y := y + 1\right] = \\
= \left(0, 0, \frac{1}{2}, 0, 0, 0, 0, 0\right) \otimes \left(0, 0, 1, 0\right)
\]

The sum of both branches is now, maybe somewhat surprising:

\[
\sigma = \left(0, 0, 1, 0, 0, 0, 0, 0\right) \otimes \left(0, \frac{1}{2}, \frac{1}{2}, 0\right)
\]

Though we have started with a definitive value for \(y\) and a distribution for \(x\), the opposite is now the case.
Consider the following labelled program:

1: while \([z < 100]\) do
2: choose \(\frac{1}{3}\) : \([x:=3]\) or \(\frac{2}{3}\) : \([x:=1]\) ro
3: end while
4: [stop]
Probabilistic Control Flow

Consider the following labelled program:

1: \textbf{while} [z < 100] \^1 \textbf{do} \\
2: \textbf{choose}^{2} \frac{1}{3} : [x:=3]^{3} \textbf{or} \frac{2}{3} : [x:=1]^{4} \textbf{ro} \\
3: \textbf{end while} \\
4: [\textbf{stop}]^{5}

Its \textbf{probabilistic control flow} is given by:

\[
flow(P) = \{ \langle 1, 1, 2 \rangle, \langle 1, 1, 5 \rangle, \langle 2, \frac{1}{3}, 3 \rangle, \langle 2, \frac{2}{3}, 4 \rangle, \langle 3, 1, 1 \rangle, \langle 4, 1, 1 \rangle \}.
\]
\[\text{init}(\text{[skip]}^\ell) = \ell\]
\[\text{init}(\text{[stop]}^\ell) = \ell\]
\[\text{init}(\text{[x:=e]}^\ell) = \ell\]
\[\text{init}(\text{[x?=e]}^\ell) = \ell\]
\[\text{init}(S_1; S_2) = \text{init}(S_1)\]
\[\text{init}(\text{choose}^\ell p_1 : S_1 \text{ or } p_2 : S_2) = \ell\]
\[\text{init}(\text{if } [b]^\ell \text{ then } S_1 \text{ else } S_2) = \ell\]
\[\text{init}(\text{while } [b]^\ell \text{ do } S) = \ell\]
Final Labels

\[
\begin{align*}
\text{final}([\text{skip}]^\ell) &= \{\ell\} \\
\text{final}([\text{stop}]^\ell) &= \{\ell\} \\
\text{final}([x:=e]^\ell) &= \{\ell\} \\
\text{final}([x?=e]^\ell) &= \{\ell\} \\
\text{final}(S_1; S_2) &= \text{final}(S_2) \\
\text{final}(\text{choose}^\ell p_1 : S_1 \text{ or } p_2 : S_2) &= \text{final}(S_1) \cup \text{final}(S_2) \\
\text{final}(\text{if } [b]^\ell \text{ then } S_1 \text{ else } S_2) &= \text{final}(S_1) \cup \text{final}(S_2) \\
\text{final}(\text{while } [b]^\ell \text{ do } S) &= \{\ell\}
\end{align*}
\]
Flow I — Control Transfer

The probabilistic control flow is defined by the function:

\[ \text{flow} : \text{Stmt} \rightarrow \mathcal{P}(\text{Lab} \times [0, 1] \times \text{Lab}) \]
The probabilistic control flow is defined by the function:

\[ flow : \text{Stmt} \to \mathcal{P}(\text{Lab} \times [0, 1] \times \text{Lab}) \]

- \( flow([\text{skip}]^\ell) = \emptyset \)
- \( flow([\text{stop}]^\ell) = \{ \langle \ell, 1, \ell \rangle \} \)
- \( flow([x := e]^\ell) = \emptyset \)
- \( flow([x ?= e]^\ell) = \emptyset \)
- \( flow(S_1; S_2) = flow(S_1) \cup flow(S_2) \cup \{ (\ell, 1, \text{init}(S_2)) \mid \ell \in \text{final}(S_1) \} \)
Flow II — Control Transfer

\[
flow(\text{choose}^\ell p_1 : S_1 \text{ or } p_2 : S_2) = flow(S_1) \cup flow(S_2) \cup \\
\{(\ell, p_1, \text{init}(S_1)), (\ell, p_2, \text{init}(S_2))\}
\]

\[
flow(\text{if } [b]^\ell \text{ then } S_1 \text{ else } S_2) = flow(S_1) \cup flow(S_2) \cup \\
\{(\ell, 1, \text{init}(S_1)), (\ell, 1, \text{init}(S_2))\}
\]

\[
flow(\text{while } [b]^\ell \text{ do } S) = flow(S) \cup \\
\{(\ell, 1, \text{init}(S))\} \\
\{(\ell', 1, \ell) \mid \ell' \in \text{final}(S)\}
\]
A Linear Operator Semantics (LOS) based on \textit{flow}

Using the \textit{flow}(S) we construct a linear operator/matrix/DTMC generator in a compositional way, essentially as:

\[ T(S) = \sum_{\langle i, p_{ij}, j \rangle \in \text{flow}(S)} p_{ij} \cdot T(\langle \ell_i, p_{ij}, \ell_j \rangle), \]

where

\[ T(\langle \ell_i, p_{ij}, \ell_j \rangle) = N_{\ell_i} \otimes E(\ell_i, \ell_j), \]
Using the $flow(S)$ we construct a linear operator/matrix/DTMC generator in a compositional way, essentially as:

$$T(S) = \sum_{\langle i, p_{ij}, j \rangle \in flow(S)} p_{ij} \cdot T(\langle \ell_i, p_{ij}, \ell_j \rangle),$$

where

$$T(\langle \ell_i, p_{ij}, \ell_j \rangle) = N_{\ell_i} \otimes E(\ell_i, \ell_j),$$
A Linear Operator Semantics (LOS) based on *flow*

Using the *flow*(S) we construct a linear operator/matrix/DTMC generator in a compositional way, essentially as:

\[
T(S) = \sum_{\langle i, p_{ij}, j \rangle \in \text{flow}(S)} p_{ij} \cdot T(\langle \ell_i, p_{ij}, \ell_j \rangle),
\]

where

\[
T(\langle \ell_i, p_{ij}, \ell_j \rangle) = N_{\ell_i} \otimes E(\ell_i, \ell_j),
\]

With \( N_{\ell_1} \) the operator representing a state update (change of variable values) at the block with label \( \ell_i \) and the second factor implementing the transfer of control from label \( \ell_i \) to label \( \ell_j \).
Transfer Operators

For all the blocks in S we have transfer operators which change the state and (then/simultaneously) perform a control transfer to another block or program points:

\[
\begin{align*}
T(\langle \ell_1, p, \ell_2 \rangle) &= I \otimes E(\ell_1, \ell_2) & \text{for } [\text{skip}]^{\ell_1} \\
T(\langle \ell_1, p, \ell_2 \rangle) &= U(x \leftarrow a) \otimes E(\ell_1, \ell_2) & \text{for } [x \leftarrow a]^{\ell_1} \\
T(\langle \ell_1, p, \ell_2 \rangle) &= \sum_{i \in r} \frac{1}{|r|} U(x \leftarrow i) \otimes E(\ell_1, \ell_2) & \text{for } [x \?=? r]^{\ell_1} \\
T(\langle \ell, p, \ell_t \rangle) &= P(b = \text{true}) \otimes E(\ell, \ell_t) & \text{for } [b]^{\ell} \\
T(\langle \ell, p, \ell_f \rangle) &= P(b = \text{false}) \otimes E(\ell, \ell_f) & \text{for } [b]^{\ell} \\
T(\langle \ell, p_k, \ell_k \rangle) &= I \otimes E(\ell, \ell_k) & \text{for } [\text{choose}]^{\ell} \\
T(\langle \ell, p, \ell \rangle) &= I \otimes E(\ell, \ell) & \text{for } [\text{stop}]^{\ell}
\end{align*}
\]

For \([b]^{\ell}\) the label \(\ell_t\) denotes the label to the ‘true’ situation (e.g. \text{then} branch) and \(\ell_f\) the situation where \(b\) is ‘false’.

In the case of a \text{choose} statement the different alternatives are labeled with (initial) label \(\ell_k\).
Select a value $c \in \text{Value}_k$ for variable $x_k$ (with $k = 1, \ldots, \nu$):

$$(P(c))_{ij} = \begin{cases} 1 & \text{if } i = c = j \\ 0 & \text{otherwise.} \end{cases}$$
Tests and Filters

Select a value $c \in \text{Value}_k$ for variable $x_k$ (with $k = 1, \ldots, v$):

$$(P(c))_{ij} = \begin{cases} 1 & \text{if } i = c = j \\ 0 & \text{otherwise.} \end{cases}$$

Select a certain classical state $\sigma \in \text{State} = \text{Value}^v$:

$$P(\sigma) = \bigotimes_{i=1}^{v} P(\sigma(x_i))$$
Tests and Filters

Select a value $c \in \text{Value}_k$ for variable $x_k$ (with $k = 1, \ldots, v$):

$$(P(c))_{ij} = \begin{cases} 1 & \text{if } i = c = j \\ 0 & \text{otherwise}. \end{cases}$$

Select a certain classical state $\sigma \in \text{State} = \text{Value}^v$:

$$P(\sigma) = \prod_{i=1}^{v} P(\sigma(x_i))$$

Select states where expression $e = a \mid b$ evaluates to $c$:

$$P(e = c) = \sum_{\mathcal{E}(e)\sigma = c} P(\sigma)$$
Updates

Modify the value of variable $x_k$ to a constant $c \in \text{Value}_k$:

$$(U(c))_{ij} = \begin{cases} 1 & \text{if } j = c \\ 0 & \text{otherwise} \end{cases}.$$
Updates

Modify the value of variable $x_k$ to a constant $c \in \text{Value}_k$:

$$(U(c))_{ij} = \begin{cases} 1 & \text{if } j = c \\ 0 & \text{otherwise.} \end{cases}$$

Set value of variable $x_k \in \text{Var}$ to constant $c \in \text{Value}$:

$$U(x_k \leftarrow c) = \left( \bigotimes_{i=1}^{k-1} \mathbf{I} \right) \otimes U(c) \otimes \left( \bigotimes_{i=k+1}^{v} \mathbf{I} \right)$$
Updates

Modify the value of variable $x_k$ to a constant $c \in \text{Value}_k$:

$$(U(c))_{ij} = \begin{cases} 
1 & \text{if } j = c \\
0 & \text{otherwise.}
\end{cases}$$

Set value of variable $x_k \in \text{Var}$ to constant $c \in \text{Value}$:

$$U(x_k \leftarrow c) = \left( \bigotimes_{i=1}^{k-1} I \right) \otimes U(c) \otimes \left( \bigotimes_{i=k+1}^{v} I \right)$$

Set value of variable $x_k \in \text{Var}$ to value given by $e = a | b$:

$$U(x_k \leftarrow e) = \sum_c P(e = c)U(x_k \leftarrow c)$$
An Example

```plaintext
if [x == 0]¹ then
    [x ← 0]²;
else
    [x ← 1]³;
end if;
[stop]⁴
```
An Example

\[
\text{if } [\text{x} == 0]^{\text{1}} \text{ then} \\
\quad \text{[x }\leftarrow 0]^{\text{2}}; \\
\text{else} \\
\quad \text{[x }\leftarrow 1]^{\text{3}}; \\
\text{end if;} \\
\text{[stop]}^{\text{4}} \\
\]

\[
T(S) = P(\text{x} = 0) \otimes E(1, 2) + \\
+ P(\text{x} \neq 0) \otimes E(1, 3) + \\
+ U(\text{x} \leftarrow 0) \otimes E(2, 4) + \\
+ U(\text{x} \leftarrow 1) \otimes E(3, 4) + \\
+ I \otimes E(4, 4)
\]
An Example

\[ T(S) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \otimes \mathbf{E}(1, 2) + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \otimes \mathbf{E}(1, 3) + \]
\[ + \left( \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \otimes \mathbf{E}(2, 3) \right) + \left( \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \otimes \mathbf{E}(3, 4) \right) + \]
\[ + (I \otimes \mathbf{E}(4, 4)) \]
An Example

\[ T(S) = \left( \begin{array}{c}
1 & 0 \\
0 & 0
\end{array} \right) \otimes \left( \begin{array}{cccc}
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{array} \right) + \left( \begin{array}{c}
0 & 0 \\
0 & 1
\end{array} \right) \otimes \left( \begin{array}{cccc}
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{array} \right) + \left( \begin{array}{c}
1 & 0 \\
1 & 0
\end{array} \right) \otimes \left( \begin{array}{cccc}
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{array} \right) + \left( \begin{array}{c}
0 & 1 \\
0 & 1
\end{array} \right) \otimes \left( \begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{array} \right) + \left( \begin{array}{c}
1 & 0 \\
0 & 1
\end{array} \right) \otimes \left( \begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{array} \right) \]
LOS and DTMC

We can compare this $T(S)$ with the directly extracted operator, and indeed the two coincide.

\[
\begin{align*}
\langle x \mapsto 0, [x == 0]\rangle & \quad \ldots \quad \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \\
\langle x \mapsto 0, [x:=0]\rangle & \quad \ldots \quad \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \\
\langle x \mapsto 0, [x:=1]\rangle & \quad \ldots \quad \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \\
\langle x \mapsto 0, [\text{stop}]\rangle & \quad \ldots \quad \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \\
\langle x \mapsto 1, [x == 0]\rangle & \quad \ldots \quad \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \\
\langle x \mapsto 1, [x:=0]\rangle & \quad \ldots \quad \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \\
\langle x \mapsto 1, [x:=1]\rangle & \quad \ldots \quad \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \\
\langle x \mapsto 1, [\text{stop}]\rangle & \quad \ldots \quad \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}
\end{align*}
\]
Written in OCaml produces an *octave* file `c.m` which specify
the LOS matrices $U$, $P$, etc. for a pWhile program `c.pw`.

We can use the interactive interface of *octave* and definitions
of standard operations in `LOS.m` to analyse matrices in `c.m`.

Exploiting sparse matrix representation to handle programs
with about 3 to 5 variables, up to 10 values and program
fragments with something like 20 lines/labels.
Consider the program $F$ for calculating the factorial of $n$:

\[
\begin{align*}
\text{var} \\
m : \{0..2\}; \\
n : \{0..2\}; \\
\text{begin} \\
m := 1; \\
\text{while } (n>1) \text{ do} \\
m := m \times n; \\
n := n - 1; \\
\text{od}; \\
\text{stop}; \#\text{ looping} \\
\text{end}
\end{align*}
\]
Control Flow and LOS for $F$

\[
\text{flow}(F) = \{(1, 1, 2), (2, 1, 3), (3, 1, 4), (4, 1, 2), (2, 1, 5), (5, 1, 5)\}
\]
Control Flow and LOS for $F$

\[ \text{flow}(F) = \{(1, 1, 2), (2, 1, 3), (3, 1, 4), (4, 1, 2), (2, 1, 5), (5, 1, 5)\} \]

\[ T(F) = U(m \leftarrow 1) \otimes E(1, 2) + \]
\[ P((n > 1)) \otimes E(2, 3) + \]
\[ U(m \leftarrow (m * n)) \otimes E(3, 4) + \]
\[ U(n \leftarrow (n - 1)) \otimes E(4, 2) + \]
\[ P((n <= 1)) \otimes E(2, 5) + \]
\[ I \otimes E(5, 5) \]
Introducing PAI

The matrix $\mathbf{T}(F)$ is very big already for small $n$.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$\dim(\mathbf{T}(F))$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$45 \times 45$</td>
</tr>
<tr>
<td>3</td>
<td>$140 \times 140$</td>
</tr>
<tr>
<td>4</td>
<td>$625 \times 625$</td>
</tr>
<tr>
<td>5</td>
<td>$3630 \times 3630$</td>
</tr>
<tr>
<td>6</td>
<td>$25235 \times 25235$</td>
</tr>
<tr>
<td>7</td>
<td>$201640 \times 201640$</td>
</tr>
<tr>
<td>8</td>
<td>$1814445 \times 1814445$</td>
</tr>
<tr>
<td>9</td>
<td>$18144050 \times 18144050$</td>
</tr>
</tbody>
</table>

We will show how we can drastically reduce the dimension of the LOS by using Probabilistic Abstract Interpretation.
Galois Connections

Definition
Let $\mathcal{C} = (\mathcal{C}, \leq_{\mathcal{C}})$ and $\mathcal{D} = (\mathcal{D}, \leq_{\mathcal{D}})$ be two partially ordered sets with two order-preserving functions $\alpha : \mathcal{C} \rightarrow \mathcal{D}$ and $\gamma : \mathcal{D} \rightarrow \mathcal{C}$. Then $(\mathcal{C}, \alpha, \gamma, \mathcal{D})$ form a Galois connection iff

(i) $\alpha \circ \gamma$ is reductive i.e. $\forall d \in \mathcal{D}, \alpha \circ \gamma(d) \leq_{\mathcal{D}} d$,

(ii) $\gamma \circ \alpha$ is extensive i.e. $\forall c \in \mathcal{C}, c \leq_{\mathcal{C}} \gamma \circ \alpha(c)$.
Galois Connections

Definition
Let \( C = (\mathcal{C}, \leq_C) \) and \( D = (\mathcal{D}, \leq_D) \) be two partially ordered sets with two order-preserving functions \( \alpha : \mathcal{C} \rightarrow \mathcal{D} \) and \( \gamma : \mathcal{D} \rightarrow \mathcal{C} \). Then \((\mathcal{C}, \alpha, \gamma, \mathcal{D})\) form a Galois connection iff

(i) \( \alpha \circ \gamma \) is reductive i.e. \( \forall d \in \mathcal{D}, \alpha \circ \gamma(d) \leq_D d \),

(ii) \( \gamma \circ \alpha \) is extensive i.e. \( \forall c \in \mathcal{C}, c \leq_C \gamma \circ \alpha(c) \).

Proposition

Let \((\mathcal{C}, \alpha, \gamma, \mathcal{D})\) be a Galois connection. Then \( \alpha \) and \( \gamma \) are quasi-inverse, i.e.

\[
(i) \quad \alpha \circ \gamma \circ \alpha = \alpha \quad \text{and} \quad (ii) \quad \gamma \circ \alpha \circ \gamma = \gamma
\]
General Construction

The general construction of correct (and optimal) abstractions $f\#$ of concrete function $f$ is as follows:
General Construction

The general construction of correct (and optimal) abstractions $f#$ of concrete function $f$ is as follows:

$$
\begin{array}{ccc}
\mathcal{A} & \xrightarrow{\alpha} & \mathcal{A}^#
\\
\downarrow{f} & \mathcal{B} & \downarrow{f^#}
\\
\mathcal{B} & \leftarrow{\alpha'} & \mathcal{B}^#
\\
\downarrow{\gamma} & & \downarrow{\gamma'}
\end{array}
$$

Correct approximation:
$$
\alpha' \circ f \leq f^# \circ \alpha.
$$

Induced semantics:
$$
f^# = \alpha' \circ f \circ \gamma.
$$
The general construction of correct (and optimal) abstractions $f\#$ of concrete function $f$ is as follows:

Correct approximation:

$$\alpha' \circ f \leq \# f\# \circ \alpha.$$
The general construction of correct (and optimal) abstractions $f^\#$ of concrete function $f$ is as follows:

![Diagram](attachment:general_construction.png)

Correct approximation:

$$\alpha' \circ f \leq^\# f^\# \circ \alpha.$$

Induced semantics:

$$f^\# = \alpha' \circ f \circ \gamma.$$
A probabilistic domain is essentially a vector space which represents the distributions $\text{Dist}(\text{State}) \subseteq \mathcal{V}(\text{State})$ on the state space $\text{State}$ of a probabilistic transition system, i.e. for finite state spaces.
A probabilistic domain is essentially a vector space which represents the distributions $\text{Dist} (\text{State}) \subseteq \mathcal{V} (\text{State})$ on the state space $\text{State}$ of a probabilistic transition system, i.e. for finite state spaces

$$\mathcal{V} (\text{State}) = \{ (v_s)_{s \in \text{State}} \mid v_s \in \mathbb{R} \}.$$
A probabilistic domain is essentially a vector space which represents the distributions $\text{Dist}(\text{State}) \subseteq \mathcal{V}(\text{State})$ on the state space $\text{State}$ of a probabilistic transition system, i.e. for finite state spaces

$$\mathcal{V}(\text{State}) = \{ (v_s)_{s \in \text{State}} \mid v_s \in \mathbb{R} \}.$$ 

In the infinite setting we can identify $\mathcal{V}(\text{State})$ with the Hilbert space $\ell^2(\text{State})$. 
A probabilistic domain is essentially a vector space which represents the distributions \( \text{Dist(State)} \subseteq \mathcal{V}(\text{State}) \) on the state space \text{State} of a probabilistic transition system, i.e. for finite state spaces

\[
\mathcal{V}(\text{State}) = \{ (v_s)_{s \in \text{State}} \mid v_s \in \mathbb{R} \}.
\]

In the infinite setting we can identify \( \mathcal{V}(\text{State}) \) with the Hilbert space \( \ell^2(\text{State}) \).

The notion of norm (distance) is essential for our treatment; we will consider normed vector spaces.
A norm on a vector space $\mathcal{V}$ is a map $\| \cdot \| : \mathcal{V} \rightarrow \mathbb{R}$ such that for all $v, w \in \mathcal{V}$ and $c \in \mathbb{C}$:

1. $\| v \| \geq 0$,
2. $\| v \| = 0 \iff v = \text{o}$, where $\text{o} \in \mathcal{V}$ is the zero vector,
3. $\| cv \| = |c| \| v \|$, and
4. $\| v + w \| \leq \| v \| + \| w \|$. 

We can always use a norm to define a metric topology on a vector space via the distance function $d(v, w) = \| v - w \|$. 

Note: The structural similarities between distances and partial orders can be made precise (cf. Category Theory).
A **norm** on a vector space $V$ is a map $\| \cdot \| : V \rightarrow \mathbb{R}$ such that for all $v, w \in V$ and $c \in \mathbb{C}$:

- $\|v\| \geq 0$,
A norm on a vector space $\mathcal{V}$ is a map $\| \cdot \| : \mathcal{V} \rightarrow \mathbb{R}$ such that for all $v, w \in \mathcal{V}$ and $c \in \mathbb{C}$:

- $\|v\| \geq 0$,
- $\|cv\| = |c|\|v\|$,
- $\|v+w\| \leq \|v\|+\|w\|$,

with $o \in \mathcal{V}$ the zero vector.
A norm on a vector space $\mathcal{V}$ is a map $\|\cdot\| : \mathcal{V} \rightarrow \mathbb{R}$ such that for all $v, w \in \mathcal{V}$ and $c \in \mathbb{C}$:

- $\|v\| \geq 0$,
- $\|v\| = 0 \iff v = o$,
- $\|cv\| = |c|\|v\|$. 

We can always use a norm to define a metric topology on a vector space via the distance function $d(v, w) = \|v - w\|$. 

Note: The structural similarities between distances and partial orders can be made precise (cf. Category Theory).
A norm on a vector space $\mathcal{V}$ is a map $\| \cdot \| : \mathcal{V} \to \mathbb{R}$ such that for all $v, w \in \mathcal{V}$ and $c \in \mathbb{C}$:

- $\| v \| \geq 0$,
- $\| v \| = 0 \iff v = o$,
- $\| cv \| = |c| \| v \|$, 
- $\| v + w \| \leq \| v \| + \| w \|$, 

with $o \in \mathcal{V}$ the zero vector.
A norm on a vector space $\mathcal{V}$ is a map $\| \cdot \| : \mathcal{V} \mapsto \mathbb{R}$ such that for all $v, w \in \mathcal{V}$ and $c \in \mathbb{C}$:

- $\| v \| \geq 0$,
- $\| v \| = 0 \iff v = o$,
- $\| cv \| = |c| \| v \|$,  
- $\| v + w \| \leq \| v \| + \| w \|$, 

with $o \in \mathcal{V}$ the zero vector.
A norm on a vector space $\mathcal{V}$ is a map $\| \cdot \| : \mathcal{V} \to \mathbb{R}$ such that for all $\mathbf{v}, \mathbf{w} \in \mathcal{V}$ and $c \in \mathbb{C}$:

- $\| \mathbf{v} \| \geq 0$,
- $\| \mathbf{v} \| = 0 \iff \mathbf{v} = \mathbf{0}$,
- $\| c\mathbf{v} \| = |c| \| \mathbf{v} \|$,
- $\| \mathbf{v} + \mathbf{w} \| \leq \| \mathbf{v} \| + \| \mathbf{w} \|$,

with $\mathbf{0} \in \mathcal{V}$ the zero vector.

We can always use a norm to define a metric topology on a vector space via the distance function $d(\mathbf{v}, \mathbf{w}) = \| \mathbf{v} - \mathbf{w} \|$.
A norm on a vector space $\mathcal{V}$ is a map $\| \cdot \| : \mathcal{V} \rightarrow \mathbb{R}$ such that for all $v, w \in \mathcal{V}$ and $c \in \mathbb{C}$:

- $\|v\| \geq 0$,
- $\|v\| = 0 \iff v = o$,
- $\|cv\| = |c| \|v\|$,
- $\|v + w\| \leq \|v\| + \|w\|$,

with $o \in \mathcal{V}$ the zero vector.

We can always use a norm to define a metric topology on a vector space via the distance function $d(v, w) = \|v - w\|$.

Note: The structural similarities between distances and partial orders can be made precise (cf. Category Theory).
Moore-Penrose Generalised Inverse

Definition
Let $\mathcal{C}$ and $\mathcal{D}$ be two (finite-dimensional) vector (Hilbert) spaces and $A : \mathcal{C} \to \mathcal{D}$ a linear map. Then the linear map $A^\dagger = G : \mathcal{D} \to \mathcal{C}$ is the Moore-Penrose pseudo-inverse of $A$ iff

(i) $A \circ G = P_A$,
(ii) $G \circ A = P_G$,

where $P_A$ and $P_G$ denote orthogonal projections onto the ranges of $A$ and $G$. 
(Orthogonal) Projections – Idempotents

On finite dimensional vector (Hilbert) spaces we have an inner product $\langle ., . \rangle$. This measures some kind of similarity of vectors but also allows to define a norm:

$$ \| x \|_2 = \sqrt{\langle x, x \rangle} $$

It also allows us to define an adjoint via:

$$ \langle A(x), y \rangle = \langle x, A^*(y) \rangle $$
(Orthogonal) Projections – Idempotents

On finite dimensional vector (Hilbert) spaces we have an inner product $\langle ., . \rangle$.
This measures some kind of similarity of vectors but also allows to define a norm:

$$\|x\|_2 = \sqrt{\langle x, x \rangle}$$

It also allows us to define an adjoint via:

$$\langle A(x), y \rangle = \langle x, A^*(y) \rangle$$
On finite dimensional vector (Hilbert) spaces we have an inner product $\langle ., . \rangle$.
This measures some kind of similarity of vectors but also allows to define a norm:

$$\|x\|_2 = \sqrt{\langle x, x \rangle}$$

It also allows us to define an adjoint via:

$$\langle A(x), y \rangle = \langle x, A^*(y) \rangle$$

- An operator $A$ is self-adjoint if $A = A^*$. 
(Orthogonal) Projections – Idempotents

On finite dimensional vector (Hilbert) spaces we have an inner product $\langle \cdot, \cdot \rangle$.
This measures some kind of similarity of vectors but also allows to define a norm:
\[ \|x\|_2 = \sqrt{\langle x, x \rangle} \]

It also allows us to define an adjoint via:
\[ \langle A(x), y \rangle = \langle x, A^*(y) \rangle \]

- An operator $A$ is self-adjoint if $A = A^*$.
- An (orthogonal) projection is a self-adjoint $E$ with $EE = E$. 

Corollary

Let $P$ be a orthogonal projection on a finite dimensional vector space $V$. Then for any $x \in V$, $P(x) = xP$ is the unique closest vector in $V$ to $x$ wrt to the Euclidean norm $\| \cdot \|_2$. 
**Least Squares Solutions**

**Corollary**

Let $P$ be a orthogonal projection on a finite dimensional vector space $V$. Then for any $x \in V$, $P(x) = xP$ is the unique closest vector in $V$ to $x$ wrt to the Euclidean norm $\| \cdot \|_2$.

**Definition**

Let $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. Then $u \in \mathbb{R}^n$ is called a least squares solution to $Ax = b$ if

$$
\|Au - b\| \leq \|Av - b\|, \text{ for all } v \in \mathbb{R}^n.
$$
Least Squares Solutions

Corollary

Let $P$ be a orthogonal projection on a finite dimensional vector space $\mathcal{V}$. Then for any $x \in \mathcal{V}$, $P(x) = xP$ is the unique closest vector in $\mathcal{V}$ to $x$ wrt to the Euclidean norm $\| \cdot \|_2$.

Definition

Let $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. Then $u \in \mathbb{R}^n$ is called a least squares solution to $Ax = b$ if

$$\|Au - b\| \leq \|Av - b\|, \text{ for all } v \in \mathbb{R}^n.$$

Theorem

Let $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. Then $A^\dagger b$ is the minimal least squares solution to $Ax = b$. 

Vector Space Lifting

Free vector space construction on a set $S$:

$$\mathcal{V}(S) = \left\{ \sum x_s s \mid x_s \in \mathbb{R}, s \in S \right\}$$
Vector Space Lifting

Free vector space construction on a set \( S \):

\[
\mathcal{V}(S) = \{ \sum x_s s \mid x_s \in \mathbb{R}, s \in S \}
\]

An obvious way to lift an extraction function to a linear map between vector spaces is to construct the free vector spaces on \( \mathcal{C} \) and \( \mathcal{D} \) and define:

\[
\vec{\alpha}(p_1 \cdot \vec{c}_1 + p_2 \cdot \vec{c}_2 + \ldots) = p_i \cdot \alpha(c_1) + p_2 \cdot \alpha(c_2) + \ldots
\]

Support Set:

\[
\text{supp}(\vec{x}) = \{ c_i \mid \langle c_i, p_i \rangle \in \vec{x} \text{ and } p_i \neq 0 \}
\]
Vector Space Lifting

Free vector space construction on a set $S$:

$$\mathcal{V}(S) = \left\{ \sum x_s s \mid x_s \in \mathbb{R}, s \in S \right\}$$

An obvious way to lift an extraction function to a linear map between vector spaces is to construct the free vector spaces on $\mathcal{C}$ and $\mathcal{D}$ and define:

**Vector Space lifting:** $\bar{\alpha} : \mathcal{V}(\mathcal{C}) \rightarrow \mathcal{V}(\mathcal{D})$

$$\bar{\alpha}(p_1 \cdot \bar{c}_1 + p_2 \cdot \bar{c}_2 + \ldots) = p_i \cdot \alpha(c_1) + p_2 \cdot \alpha(c_2) \ldots$$
Vector Space Lifting

Free vector space construction on a set $S$:

$$\mathcal{V}(S) = \left\{ \sum x_s s \mid x_s \in \mathbb{R}, s \in S \right\}$$

An obvious way to lift an extraction function to a linear map between vector spaces is to construct the free vector spaces on $C$ and $D$ and define:

**Vector Space lifting**: $\vec{\alpha} : \mathcal{V}(C) \rightarrow \mathcal{V}(D)$

$$\vec{\alpha}(p_1 \cdot \vec{c}_1 + p_2 \cdot \vec{c}_2 + \ldots) = p_i \cdot \alpha(c_1) + p_2 \cdot \alpha(c_2) \ldots$$

**Support Set**: $\text{supp} : \mathcal{V}(C) \rightarrow \mathcal{P}(C)$

$$\text{supp}(\vec{x}) = \left\{ c_i \mid \langle c_i, p_i \rangle \in \vec{x} \text{ and } p_i \neq 0 \right\}$$
Relation with Classical Abstractions

Lemma
Let $\alpha$ be a probabilistic abstraction function and let $\gamma$ be its Moore-Penrose pseudo-inverse.

Then $\gamma \circ \alpha$ is extensive with respect to the inclusion on the support sets of vectors in $V(C)$, i.e. $\forall \tilde{x} \in V(C)$,

$$\text{supp}(\tilde{x}) \subseteq \text{supp}(\gamma \circ \alpha(\tilde{x})).$$
Lemma

Let \( \tilde{\alpha} \) be a probabilistic abstraction function and let \( \tilde{\gamma} \) be its Moore-Penrose pseudo-inverse.

Then \( \tilde{\gamma} \circ \tilde{\alpha} \) is extensive with respect to the inclusion on the support sets of vectors in \( \mathcal{V}(C) \), i.e. \( \forall \tilde{x} \in \mathcal{V}(C) \),

\[
\text{supp}(\tilde{x}) \subseteq \text{supp}(\tilde{\gamma} \circ \tilde{\alpha}(\tilde{x})).
\]

Analogously we can show that \( \tilde{\alpha} \circ \tilde{\gamma} \) is reductive. Therefore,
Relation with Classical Abstractions

Lemma
Let \( \vec{\alpha} \) be a probabilistic abstraction function and let \( \vec{\gamma} \) be its Moore-Penrose pseudo-inverse.

Then \( \vec{\gamma} \circ \vec{\alpha} \) is extensive with respect to the inclusion on the support sets of vectors in \( \mathcal{V}(\mathcal{C}) \), i.e. \( \forall \vec{x} \in \mathcal{V}(\mathcal{C}) \),

\[
\text{supp}(\vec{x}) \subseteq \text{supp}(\vec{\gamma} \circ \vec{\alpha}(\vec{x})).
\]

Analogously we can show that \( \vec{\alpha} \circ \vec{\gamma} \) is reductive. Therefore,

Proposition
\((\vec{\alpha}, \vec{\gamma})\) form a Galois connection wrt the support sets of \( \mathcal{V}(\mathcal{C}) \) and \( \mathcal{V}(\mathcal{D}) \), ordered by inclusion.
Examples of Lifted Abstractions

Parity Abstraction operator on $\mathcal{V}(\{1, \ldots, n\})$ (with $n$ even):

$$A_p = \begin{pmatrix}
1 & 0 \\
0 & 1 \\
1 & 0 \\
0 & 1 \\
\vdots & \vdots \\
0 & 1
\end{pmatrix}$$
Examples of Lifted Abstractions

Parity Abstraction operator on $\mathcal{V} \{1, \ldots, n\}$ (with $n$ even):

\[
A_p = \begin{pmatrix}
1 & 0 \\
0 & 1 \\
1 & 0 \\
0 & 1 \\
\vdots & \vdots \\
0 & 1 \\
\end{pmatrix}, \quad A_p^\dagger = \begin{pmatrix}
\frac{2}{n} & 0 & \frac{2}{n} & 0 & \cdots & 0 \\
0 & \frac{2}{n} & 0 & \frac{2}{n} & \cdots & \frac{2}{n} \\
\end{pmatrix}
\]
Examples of Lifted Abstractions

Sign Abstraction operator on $\mathcal{N}(-n, \ldots, 0, \ldots, n)$:

$$A_s = \begin{pmatrix}
1 & 0 & 0 \\
\vdots & \vdots & \vdots \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\vdots & \vdots & \vdots \\
0 & 0 & 1
\end{pmatrix}$$
Examples of Lifted Abstractions

Sign Abstraction operator on $\mathcal{V}(-n, \ldots, 0, \ldots, n)$:

$$A_s = \begin{pmatrix}
1 & 0 & 0 \\
\vdots & \vdots & \vdots \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\vdots & \vdots & \vdots \\
0 & 0 & 1
\end{pmatrix}$$

$$A_s^\dagger = \begin{pmatrix}
\frac{1}{n} & \cdots & \frac{1}{n} & 0 & 0 & \cdots & 0 \\
0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\
0 & \cdots & 0 & 0 & \frac{1}{n} & \cdots & \frac{1}{n}
\end{pmatrix}$$
Example: Function Approximation (ctd.)

Concrete and abstract domain are step-functions on $[a, b]$. 

\[(5 \ 5 \ 6 \ 7 \ 8 \ 4 \ 3 \ 2 \ 8 \ 6 \ 6 \ 7 \ 9 \ 8 \ 8 \ 7)\]
Concrete and abstract domain are step-functions on \([a, b]\). The set of (real-valued) step-function \(\mathcal{T}_n\) is based on the sub-division of the interval into \(n\) sub-intervals.
Concrete and abstract domain are step-functions on \([a, b]\). The set of (real-valued) step-function \(\mathcal{T}_n\) is based on the sub-division of the interval into \(n\) sub-intervals.
Concrete and abstract domain are step-functions on \([a, b]\). The set of (real-valued) step-function \(\mathcal{T}_n\) is based on the sub-division of the interval into \(n\) sub-intervals.

Each step function in \(\mathcal{T}_n\) corresponds to a vector in \(\mathbb{R}^n\), e.g.
Concrete and abstract domain are step-functions on \([a, b]\). The set of (real-valued) step-function \(\mathcal{T}_n\) is based on the sub-division of the interval into \(n\) sub-intervals.

Each step function in \(\mathcal{T}_n\) corresponds to a vector in \(\mathbb{R}^n\), e.g.

\[
( 5 \ 5 \ 6 \ 7 \ 8 \ 4 \ 3 \ 2 \ 8 \ 6 \ 6 \ 7 \ 9 \ 8 \ 8 \ 7 )
\]
Example: Abstraction Matrices

\[
A_8 = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]
Example: Abstraction Matrices

\[
G_8 = \begin{pmatrix}
\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\
\end{pmatrix}
\]
Compute the *least square error* as

\[ \| f - f_{AG} \|. \]
Approximation Estimates

Compute the *least square error* as

$$\| f - fAG \|.$$ 

$$\| f - fA_8G_8 \| = 3.5355$$
$$\| f - fA_4G_4 \| = 5.3151$$
$$\| f - fA_2G_2 \| = 5.9896$$
$$\| f - fA_1G_1 \| = 7.6444$$
Tensor Product Properties

The tensor product of $n$ linear operators $A_1, A_2, \ldots, A_n$ is associative (but in general not commutative) and has e.g. the following properties:
Tensor Product Properties

The tensor product of \( n \) linear operators \( A_1, A_2, \ldots, A_n \) is associative (but in general not commutative) and has e.g. the following properties:

1. \((A_1 \otimes \ldots \otimes A_n) \cdot (B_1 \otimes \ldots \otimes B_n) = A_1 \cdot B_1 \otimes \ldots \otimes A_n \cdot B_n\)
Tensor Product Properties

The tensor product of \( n \) linear operators \( A_1, A_2, \ldots, A_n \) is associative (but in general not commutative) and has e.g. the following properties:

1. \( (A_1 \otimes \ldots \otimes A_n) \cdot (B_1 \otimes \ldots \otimes B_n) = A_1 \cdot B_1 \otimes \ldots \otimes A_n \cdot B_n \)

2. \( A_1 \otimes \ldots \otimes (\alpha A_j) \otimes \ldots \otimes A_n = \alpha (A_1 \otimes \ldots \otimes A_j \otimes \ldots \otimes A_n) \)
Tensor Product Properties

The tensor product of $n$ linear operators $A_1, A_2, \ldots, A_n$ is associative (but in general not commutative) and has e.g. the following properties:

1. \[(A_1 \otimes \ldots \otimes A_n) \cdot (B_1 \otimes \ldots \otimes B_n) = A_1 \cdot B_1 \otimes \ldots \otimes A_n \cdot B_n\]
2. \[A_1 \otimes \ldots \otimes (\alpha A_i) \otimes \ldots \otimes A_n = \alpha (A_1 \otimes \ldots \otimes A_i \otimes \ldots \otimes A_n)\]
3. \[A_1 \otimes \ldots \otimes (A_i + B_i) \otimes \ldots \otimes A_n = (A_1 \otimes \ldots \otimes A_i \otimes \ldots \otimes A_n) + (A_1 \otimes \ldots \otimes B_i \otimes \ldots \otimes A_n)\]
Tensor Product Properties

The tensor product of $n$ linear operators $A_1, A_2, \ldots, A_n$ is associative (but in general not commutative) and has e.g. the following properties:

1. $(A_1 \otimes \ldots \otimes A_n) \cdot (B_1 \otimes \ldots \otimes B_n) =$
   $$= A_1 \cdot B_1 \otimes \ldots \otimes A_n \cdot B_n$$

2. $A_1 \otimes \ldots \otimes (\alpha A_i) \otimes \ldots \otimes A_n =$
   $$= \alpha (A_1 \otimes \ldots \otimes A_i \otimes \ldots \otimes A_n)$$

3. $A_1 \otimes \ldots \otimes (A_i + B_i) \otimes \ldots \otimes A_n =$
   $$= (A_1 \otimes \ldots \otimes A_i \otimes \ldots \otimes A_n) + (A_1 \otimes \ldots \otimes B_i \otimes \ldots \otimes A_n)$$

4. $(A_1 \otimes \ldots \otimes A_i \otimes \ldots \otimes A_n)^\dagger =$
   $$= A_1^\dagger \otimes \ldots \otimes A_i^\dagger \otimes \ldots \otimes A_n^\dagger$$
Abstract Semantics

Moore-Penrose Pseudo-Inverse of a Tensor Product is:

\[(A_1 \otimes A_2 \otimes \ldots \otimes A_n)^\dagger = A_1^\dagger \otimes A_2^\dagger \otimes \ldots \otimes A_n^\dagger\]
Abstract Semantics

Moore-Penrose Pseudo-Inverse of a Tensor Product is:

\[(A_1 \otimes A_2 \otimes \ldots \otimes A_n)^\dagger = A_1^\dagger \otimes A_2^\dagger \otimes \ldots \otimes A_n^\dagger\]

Via linearity we can construct \(T^\#\) in the same way as \(T\), i.e

\[T^\#(P) = \sum_{\langle i, p_{ij}, j \rangle \in \mathcal{F}(P)} p_{ij} \cdot T^\#(\ell_i, \ell_j)\]

with local abstraction of individual variables:

\[T^\#(\ell_i, \ell_j) = (A_1^\dagger N_{i1} A_1) \otimes (A_2^\dagger N_{i2} A_2) \otimes \ldots \otimes (A_v^\dagger N_{iv} A_v) \otimes M_{ij}\]
\[ T^\# = A^\dagger TA \]
\[ T^\# = A^\dagger TA = A^\dagger \left( \sum_{i,j} T(i, j) \right) A \]
\[ T^\# = A^\dagger TA \]
\[ = A^\dagger \left( \sum_{i,j} T(i,j) \right) A \]
\[ = \sum_{i,j} A^\dagger T(i,j) A \]
\[ T^\# = A^\dagger TA \\
= A^\dagger \left( \sum_{i,j} T(i,j) \right) A \\
= \sum_{i,j} A^\dagger T(i,j) A \\
= \sum_{i,j} \left( \bigotimes_k A_k \right)^\dagger T(i,j) \left( \bigotimes_k A_k \right) \]
Argument [Not for Exam]

\[ T^\# = A^\dagger T A \]
\[ = A^\dagger \left( \sum_{i,j} T(i,j) \right) A \]
\[ = \sum_{i,j} A^\dagger T(i,j) A \]
\[ = \sum_{i,j} \left( \bigotimes_k A_k \right)^\dagger T(i,j) \left( \bigotimes_k A_k \right) \]
\[ = \sum_{i,j} \left( \bigotimes_k A_k \right)^\dagger \left( \bigotimes_k N_{ik} \right) \left( \bigotimes_k A_k \right) \]
Argument [Not for Exam]

\[
\begin{align*}
T^\# &= A^\dagger TA \\
&= A^\dagger \left( \sum_{i,j} T(i,j) \right) A \\
&= \sum_{i,j} A^\dagger T(i,j) A \\
&= \sum_{i,j} \left( k \bigotimes A_k \right)^\dagger T(i,j) \left( k \bigotimes A_k \right) \\
&= \sum_{i,j} \left( k \bigotimes A_k \right)^\dagger \left( k \bigotimes N_{ik} \right) \left( k \bigotimes A_k \right) \\
&= \sum_{i,j} k \bigotimes \left( A_k^\dagger N_{ik} A_k \right)
\end{align*}
\]
Parity Analysis

Determine at each program point whether a variable is *even* or *odd*.
Parity Analysis

Determine at each program point whether a variable is **even** or **odd**.

**Parity Abstraction operator on** \( \mathcal{V}([0, \ldots, n]) \) (with \( n \) even):

\[
\mathbf{A}_p = \begin{pmatrix}
1 & 0 \\
0 & 1 \\
1 & 0 \\
0 & 1 \\
\vdots & \vdots \\
0 & 1
\end{pmatrix}
\quad \mathbf{A}^\dagger = \begin{pmatrix}
\frac{2}{n} & 0 & \frac{2}{n} & 0 & \cdots & 0 \\
0 & \frac{2}{n} & 0 & \frac{2}{n} & \cdots & \frac{2}{n}
\end{pmatrix}
\]
Example

1: \[m \leftarrow i\] \[1\];
2: \textbf{while} \[n > 1\] \[2\] \textbf{do}
3: \[m \leftarrow m \times n\] \[3\];
4: \[n \leftarrow n - 1\] \[4\]
5: \textbf{end while}
6: \[\text{stop}\] \[5\]
Example

1: \[ m \leftarrow i \]
2: while \[ n > 1 \] do
3: \[ m \leftarrow m \times n \]
4: \[ n \leftarrow n - 1 \]
5: end while
6: [stop]

\[ T = U(m \leftarrow i) \otimes E(1, 2) \]
\[ + P(n > 1) \otimes E(2, 3) \]
\[ + P(n \leq 1) \otimes E(2, 5) \]
\[ + U(m \leftarrow m \times n) \otimes E(3, 4) \]
\[ + U(n \leftarrow n - 1) \otimes E(4, 2) \]
\[ + I \otimes E(5, 5) \]
Example

1: \([m \leftarrow i]^{1}\);
2: while \([n > 1]^{2}\) do
3: \([m \leftarrow m \times n]^{3}\);
4: \([n \leftarrow n - 1]^{4}\)
5: end while
6: [stop]^{5}

\[T^{\#} = U^{\#}(m \leftarrow i) \otimes E(1, 2) + P^{\#}(n > 1) \otimes E(2, 3) + P^{\#}(n \leq 1) \otimes E(2, 5) + U^{\#}(m \leftarrow m \times n) \otimes E(3, 4) + U^{\#}(n \leftarrow n - 1) \otimes E(4, 2) + I^{\#} \otimes E(5, 5)\]
Abstract Semantics

Abstraction: \( A = A_p \otimes I \), i.e. \( m \) abstract (parity) but \( n \) concrete.

\[
T^\# = U^\#(m \leftarrow 1) \otimes E(1, 2) + P^\#(n > 1) \otimes E(2, 3) + P^\#(n \leq 1) \otimes E(2, 5) + U^\#(m \leftarrow m \times n) \otimes E(3, 4) + U^\#(n \leftarrow n - 1) \otimes E(4, 2) + I^\# \otimes E(5, 5)
\]
Abstract Semantics

\[
\mathbf{U}^\#(m \leftarrow 1) = \begin{pmatrix}
0 & 1 \\
0 & 1 \\
\end{pmatrix} \times \begin{pmatrix}
1 & 0 & 0 & 0 & \ldots & 0 \\
0 & 1 & 0 & 0 & \ldots & 0 \\
0 & 0 & 1 & 0 & \ldots & 0 \\
0 & 0 & 0 & 1 & \ldots & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & 0 & \ldots & 1 \\
\end{pmatrix}
\]
Abstract Semantics

\[ U^#(n \leftarrow n - 1) = \]
\[ = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 \end{pmatrix} \]
Abstract Semantics

$P^\#(n > 1) =$

$$= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & 1 \end{pmatrix}$$
Abstract Semantics

\[ \mathbf{P}^\#(n \leq 1) = \]

\[ = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & 1 \end{pmatrix} \]
Abstract Semantics

\[ U^\#(m \leftarrow m \times n) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \end{pmatrix} \]
Implementation

Implementation of concrete and abstract semantics of Factorial using \texttt{octave}. Ranges: \( n \in \{1, \ldots, d\} \) and \( m \in \{1, \ldots, d!\} \).
Implementation of concrete and abstract semantics of Factorial using octave. Ranges: $n \in \{1, \ldots, d\}$ and $m \in \{1, \ldots, d!\}$.

<table>
<thead>
<tr>
<th>$d$</th>
<th>$\dim(T(F))$</th>
<th>$\dim(T^#(F))$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>45</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>140</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>625</td>
<td>50</td>
</tr>
<tr>
<td>5</td>
<td>3630</td>
<td>60</td>
</tr>
<tr>
<td>6</td>
<td>25235</td>
<td>70</td>
</tr>
<tr>
<td>7</td>
<td>201640</td>
<td>80</td>
</tr>
<tr>
<td>8</td>
<td>1814445</td>
<td>90</td>
</tr>
<tr>
<td>9</td>
<td>18144050</td>
<td>100</td>
</tr>
</tbody>
</table>

Using uniform initial distributions $d_0$ for $n$ and $m$. 
The abstract probabilities for $m$ being even or odd when we execute the abstract program for various $d$ values are:

<table>
<thead>
<tr>
<th>$d$</th>
<th>even</th>
<th>odd</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.81818</td>
<td>0.18182</td>
</tr>
<tr>
<td>100</td>
<td>0.98019</td>
<td>0.019802</td>
</tr>
<tr>
<td>1000</td>
<td>0.99800</td>
<td>0.00199802</td>
</tr>
<tr>
<td>10000</td>
<td>0.99980</td>
<td>0.00019998</td>
</tr>
</tbody>
</table>
Define a partial order on self-adjoint operators and projections as follows: $H \sqsubseteq K$ iff $K - H$ is positive, i.e. there exists a $B$ such that $K - H = B^*B$. 

Alternatively, order projections by inclusion of their image spaces, i.e. $E \sqsubseteq F$ iff $Y_E \subseteq Y_F$.
Ortholattice of Projection Operators [Not for Exam]

Define a partial order on self-adjoint operators and projections as follows: $H \sqsubseteq K$ iff $K - H$ is positive, i.e. there exists a $B$ such that $K - H = B^*B$.

Alternatively, order projections by inclusion of their image spaces, i.e. $E \sqsubseteq F$ iff $Y_E \subseteq Y_F$. 
Define a partial order on self-adjoint operators and projections as follows: $H \subseteq K$ iff $K - H$ is positive, i.e. there exists a $B$ such that $K - H = B^*B$.

Alternatively, order projections by inclusion of their image spaces, i.e. $E \subseteq F$ iff $Y_E \subseteq Y_F$.

The orthogonal projections form a complete (ortho)lattice.

The range of the intersection $E \cap F$ is to the closure of the intersection of the image spaces of $E$ and $F$.

The union $E \sqcup F$ corresponds to the union of the images.
Associate to every Probabilistic Abstract Interpretation \((\mathbf{A}, \mathbf{G})\) a projection, similar to so-called “upper closure operators” (uco):

\[
\mathbf{E} = \mathbf{A}\mathbf{G} = \mathbf{A}\mathbf{A}^\dagger.
\]
Associate to every Probabilistic Abstract Interpretation \((A, G)\) a projection, similar to so-called “upper closure operators” (uco):

\[ E = AG = AA^\dagger. \]

A general way to construct \(E \cap F\) and (by exploiting de Morgan’s law) also \(E \cup F = (E^\perp \cap F^\perp)^\perp\) is via an infinite approximation sequence and has been suggested by Halmos:

\[ E \cap F = \lim_{n \to \infty} (EFE)^n. \]
Commutative Case

The concrete construction of $E \sqcup F$ and $E \sqcap F$ is in general not trivial. Only for commuting projections we have:

\[ E \sqcup F = E + F - EF \quad \text{and} \quad E \sqcap F = EF. \]
Commutative Case

The concrete construction of $E \sqcup F$ and $E \sqcap F$ is in general not trivial. Only for \textit{commuting projections} we have:

$$E \sqcup F = E + F - EF \text{ and } E \sqcap F = EF.$$ 

\textbf{Example}

Consider a finite set $\Omega$ with a probability structure. For any (measurable) subset $A$ of $\Omega$ define the characteristic function $\chi_A$ with $\chi_A(x) = 1$ if $x \in A$ and 0 otherwise.
Commutative Case

The concrete construction of $E \sqcup F$ and $E \sqcap F$ is in general not trivial. Only for commuting projections we have:

$$E \sqcup F = E + F - EF \text{ and } E \sqcap F = EF.$$ 

Example

Consider a finite set $\Omega$ with a probability structure. For any (measurable) subset $A$ of $\Omega$ define the characteristic function $\chi_A$ with $\chi_A(x) = 1$ if $x \in A$ and 0 otherwise. The characteristic functions are (commutative) projections on random variables using pointwise multiplication, i.e. $X\chi_A\chi_A = X\chi_A$. We have $\chi_{A \cap B} = \chi_A\chi_B$ and $\chi_{A \cup B} = \chi_A + \chi_B - \chi_A\chi_B$. 
The Moore-Penrose pseudo-inverse is also useful for computing the $E \cap F$ and $E \uplus F$ of general, non-commuting projections via the parallel sum

$$A : B = A(A + B)^\dagger B$$

The intersection of projections is given by:

$$E \cap F = 2(E : F) = E(E + F)^\dagger F + F(E + F)^\dagger E$$

Consider a "duel" between two cowboys:

- Cowboy $A$ – hitting probability $a$
- Cowboy $B$ – hitting probability $b$
Consider a "duel" between two cowboys:

- Cowboy $A$ – hitting probability $a$
- Cowboy $B$ – hitting probability $b$

1. Choose (non-deterministically) whether $A$ or $B$ starts.
Variable Probabilities: Duel at High Noon

Consider a "duel" between two cowboys:

- Cowboy $A$ – hitting probability $a$
- Cowboy $B$ – hitting probability $b$

1. Choose (non-deterministically) whether $A$ or $B$ starts.
2. Repeat until winner is known:
Consider a "duel" between two cowboys:

- Cowboy A – hitting probability $a$
- Cowboy B – hitting probability $b$

1. Choose (non-deterministically) whether A or B starts.
2. Repeat until winner is known:
   - If it is A’s turn he will hit/shoot B with probability $a$;
     If B is shot then A is the winner, otherwise it’s B’s turn.
Variable Probabilities: Duel at High Noon

Consider a "duel" between two cowboys:

- Cowboy $A$ – hitting probability $a$
- Cowboy $B$ – hitting probability $b$

1. Choose (non-deterministically) whether $A$ or $B$ starts.
2. Repeat until winner is known:
   - If it is $A$’s turn he will hit/shoot $B$ with probability $a$;
     If $B$ is shot then $A$ is the winner, otherwise it’s $B$’s turn.
   - If it is $B$’s turn he will hit/shoot $A$ with probability $b$;
     If $A$ is shot then $B$ is the winner, otherwise it’s $A$’s turn.
Variable Probabilities: Duel at High Noon

Consider a "duel" between two cowboys:

- Cowboy $A$ – hitting probability $a$
- Cowboy $B$ – hitting probability $b$

1. Choose (non-deterministically) whether $A$ or $B$ starts.
2. Repeat until winner is known:

   - If it is $A$’s turn he will hit/shoot $B$ with probability $a$;
     If $B$ is shot then $A$ is the winner, otherwise it’s $B$’s turn.
   - If it is $B$’s turn he will hit/shoot $A$ with probability $b$;
     If $A$ is shot then $B$ is the winner, otherwise it’s $A$’s turn.

**Question:** What is the life expectancy of $A$ or $B$?

Introduced by McIver and Morgan (2005).
Discussed in detail by Gretz, Katoen, McIver (2012/14)
Consider a "duel" between two cowboys:

- Cowboy A – hitting probability $a$
- Cowboy B – hitting probability $b$

1. Choose (non-deterministically) whether A or B starts.
2. Repeat until winner is known:
   - If it is A’s turn he will hit/shoot B with probability $a$;
     If B is shot then A is the winner, otherwise it’s B’s turn.
   - If it is B’s turn he will hit/shoot A with probability $b$;
     If A is shot then B is the winner, otherwise it’s A’s turn.

**Question:** What is the life expectancy of A or B?

**Question:** What happens if A is learning to shoot better during the duel? How can we model \textit{dynamic probabilities}?

Introduced by McIver and Morgan (2005).
Discussed in detail by Gretz, Katoen, McIver (2012/14)
Example: Duelling Cowboys

begin
# who’s first turn
choose 1:{t:=0} or 1:{t:=1} ro;
# continue until ... 
c := 1;
while c == 1 do
if (t==0) then
    choose ak:{c:=0} or am:{t:=1} ro
else
    choose bk:{c:=0} or bm:{t:=0} ro
fi;
od;
stop; # terminal loop
end
Example: Duelling Cowboys [Not for Exam]

The survival chances, i.e. winning probability, for A.
References

References


References


References


References


Herbert Wiklicky: *On Dynamical Probabilities, or: How to learn to shoot straight*. Coordinations, LNCS 9686, 2016.