Program Analysis (CO470/97128/97146)
Probabilistic Programs

Herbert Wiklicky

Department of Computing
Imperial College London

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

Autumn 2020
Overview

Topics we will cover in this part will include:
Overview

Topics we will cover in this part will include:

1. Language PWHILE
Overview

Topics we will cover in this part will include:

1. Language PWHILE
2. Operational Semantics
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 $\texttt{PWHILE}$
2. Operational Semantics
3. Tensor Products
4. Linear Operator Semantics
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: \textbf{while} \[ n > 1 \] \textbf{do}
3: \[ m := m \times n \]
4: \[ n := n - 1 \]
5: \textbf{end while}
6: \[ \text{stop} \]

Concrete Probabilities
Probabilistic Problem I: Guards and Conditionals

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

2: while \([n > 1]\) do

3: \([m := m \times n]\);

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

5: end while

Concrete Probabilities
Probabilistic Problem I: Guards and Conditionals

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

\[\triangleright (p_1, p_2, p_3, \ldots) \leadsto (q_1, q_2, \ldots)\]

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

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

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: [stop]

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

\[ \triangleright (1, 0, 0, \ldots) \leftarrow (\frac{1}{2}, \frac{1}{2}, \ldots) \]

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

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

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: [stop]

Concrete Probabilities

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

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

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

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

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

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

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: [stop]

Concrete Probabilities

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

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

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

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

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

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

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

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

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

Concrete Probabilities

\[ \nabla (p_1, p_2, p_3, \ldots) \sim (\frac{1}{2}, \frac{1}{2}, \ldots) \]
\[ \nabla (1, 0, 0, \ldots) \sim (\frac{1}{2}, \frac{1}{2}, \ldots) \]
\[ \nabla (1, 0, 0, \ldots) \sim (0, \frac{1}{2}, \ldots) \]
\[ \nabla (0, 1, 0, \ldots) \sim (0, \frac{1}{2}, \ldots) \]
\[ \nabla (0, 1, 0, \ldots) \sim (\frac{1}{2}, 0, \ldots) \]
\[ \nabla (1, 0, 0, \ldots) \sim (\frac{1}{2}, 0, \ldots) \]
1: $[m := 1]$; 
2: while $[n > 1]$ do 
3: $[m := m \times n]$; 
4: $[n := n - 1]$ 
5: end while 
6: $[stop]$

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

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

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

Concrete Probabilities

\[ (1, 0, 0, \ldots) \] — \( \left( \frac{1}{2}, 0, \ldots \right) \)
\[ (0, 1, 0, \ldots) \] — \( \left( \frac{1}{2}, 0, \ldots \right) \)
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: \( [\text{stop}] \)

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

Concrete Probabilities

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

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

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

Concrete 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: \[ \text{stop} \]

Abstract Probabilities
Probabilistic Problem II: Abstract Evaluation

1: \[ m \нейм; → P(m = 2k), P(m \neq 2k) \land P(n = 1), \ldots \]
2: \textbf{while} \[n \нейм\textbf{ do}
3: \[ m \нейм m \times n \];
4: \[ n \нейм n - 1 \];
5: \textbf{end while}
6: \textbf{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: \[ \text{stop} \]

\[ (p_e, p_o) \rightarrow (q_1, q_2, \ldots) \]

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: [stop]

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

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: \[ \text{stop} \]

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

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}

\[ \triangleright (p_e, p_o) \leftarrow \left( \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots \right) \]

\[ \triangleright (0, 1) \leftarrow \left( \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots \right) \]

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}

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

1: $[m := 1]$

2: while $[n > 1]$ do

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

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

5: end while

6: [stop]

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

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

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

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

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

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: \[ \textbf{stop} \]

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

Abstract Probabilities
Abstract Probabilities
Probabilistic Problem II: Abstract Evaluation

1: \[m := 1\]
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]}

\[\begin{align*}
\triangleright (p_e, p_o) &\rightarrow (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots) \\
\triangleright (1, 0) &\rightarrow (\frac{1}{3}, \frac{1}{3}, 0, \ldots) \\
\triangleright (1, 0) &\rightarrow (0, \frac{1}{3}, 0, \ldots)
\end{align*}\]

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. \[ \text{stop} \]

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

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}

\(\triangleright (p_e, p_o) \rightarrow \left( \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots \right)\)

\(\triangleright (1, 0) \rightarrow \left( \frac{1}{3}, \frac{1}{3}, 0, \ldots \right)\)

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

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

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

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

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

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}

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

Abstract Probabilities
Probabilistic Problem II: Abstract Evaluation

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

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

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: [stop]

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

Abstract Probabilities

Correct?
Probabilistic Problem II: Abstract Evaluation

1: \[ m := 1 \] \^[1]\n2: \textbf{while} \[ n > 1 \] \^[2] \textbf{do}
3: \[ m := m \times n \] \^[3]
4: \[ n := n - 1 \] \^[4]
5: \textbf{end while}
6: \textbf{stop} \^[5]

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

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 \(\text{even}(m)\) and \(\text{odd}(m)\).
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.
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.

The available data thus suggest this probability distribution:

\[
\begin{array}{c|ccc}
   & n = 1 & n = 2 & n = 3 \\
\hline
\text{even}(m) & \frac{1}{3} \cdot \frac{2}{3} & \frac{1}{3} \cdot \frac{2}{3} & \frac{1}{3} \cdot \frac{2}{3} \\
\text{odd}(m) & \frac{1}{3} \cdot \frac{1}{3} & \frac{1}{3} \cdot \frac{1}{3} & \frac{1}{3} \cdot \frac{1}{3}
\end{array}
\]
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 \textit{even}(m) and \textit{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.

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>\textit{even}(m)</td>
<td>(\frac{2}{9})</td>
<td>(\frac{2}{9})</td>
<td>(\frac{2}{9})</td>
</tr>
<tr>
<td>\textit{odd}(m)</td>
<td>(\frac{1}{9})</td>
<td>(\frac{1}{9})</td>
<td>(\frac{1}{9})</td>
</tr>
</tbody>
</table>
Probabilistic Problem III: Relational Dependency

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

\[
\begin{array}{c|ccc}
& n = 1 & n = 2 & n = 3 \\
\hline
\text{even}(m) & 0 & \frac{1}{3} & \frac{1}{3} \\
\text{odd}(m) & \frac{1}{3} & 0 & 0 \\
\end{array}
\]
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]

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

$\triangleright (p_e, p_o) \rightarrow \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots \right)$

$\triangleright (0, 1) \rightarrow \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots \right)$

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

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

$\triangleright (0, 1) \rightarrow \left(\frac{1}{3}, 0, 0, \ldots \right)$
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}

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

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

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

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

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

\textbf{Splitting:} How to distribute information along branches?
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]

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

**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]

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

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

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

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

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

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

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

**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) = $\frac{2}{\pi}$ or $\pi = \frac{2}{\text{Pr(cross)}}$
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 \( \mathcal{P}(\Omega) = 2^\Omega \):

\[
\mathcal{P}(\Omega) = \{ \chi : \Omega \rightarrow \{0, 1\} \}
\]

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 $\mathbb{V}(\Omega)$ of $\Omega$ as a tuple space (with $K$ a field like $\mathbb{R}$ or $\mathbb{C}$):

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

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

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

However, there are major topological problems when $\Omega$ is (un)countable infinite.
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 \( K \) a field like \( \mathbb{R} \) or \( \mathbb{C} \)):

\[
\mathcal{V}(\Omega) = \left\{ \langle \omega, x_\omega \rangle \mid \omega \in \Omega, x_\omega \in K \right\} = \left\{ (x_\omega)_{\omega \in \Omega} \mid x_\omega \in K \right\}
\]
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 $P(\Omega)$ are not so different:

$$V(\Omega) = \{\nu : \Omega \rightarrow \mathbb{K}\}$$
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 $\mathbb{C}$):

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

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

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

However, there are major topological problems when $\Omega$ 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:

- Random Assignment: The value a variable is assigned to is chosen randomly (according to some, e.g., uniform, probability distribution) from a set: $x \in \{1, 2, 3, 4\}$

- Probabilistic Choice: There is a probabilistic choice between different instructions:
  - $\text{choose} \ 0.5: (x := 0)$
  - $\text{or} \ 0.5: (x := 1)$
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:

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

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

**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:

\[ x \equiv \{0, 1\} \]

is equivalent to (assuming a uniform distribution):

\[
\textbf{choose} \ 0.5 : (x \equiv 0) \textbf{ or } 0.5 : (x \equiv 1) \textbf{ ro}
\]
Syntactic Sugar

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

Random choices and assignments can be interchanged:

\[ x \equiv \{0, 1\} \]

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):

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

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

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

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

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

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

is expressed equivalently as:

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

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

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

(by exploiting an implicit normalisation) into

```
choose \frac{p}{p+q} : S_1 \text{ or } \frac{q}{p+q} : S_2 \text{ ro}
```
The syntax of statements $S$ is as follows:

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

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

\[
S ::= [\text{stop}]^\ell \\
[\text{skip}]^\ell \\
[x := e]^\ell \\
[x \not= r]^\ell \\
S_1; S_2 \\
\text{choose}^\ell p_1 : S_1 \text{ or } p_2 : S_2 \text{ ro} \\
\text{if } [b]^\ell \text{ then } S_1 \text{ else } S_2 \text{ fi} \\
\text{while } [b]^\ell \text{ do } S \text{ od}
\]

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

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

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

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

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

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

\[
\begin{align*}
\mathcal{E}(n)\sigma &= n \\
\mathcal{E}(x)\sigma &= \sigma(x) \\
\mathcal{E}(a_1 \odot a_2)\sigma &= \mathcal{E}(a_1)\sigma \odot \mathcal{E}(a_2)\sigma \\
\mathcal{E}(\text{true})\sigma &= \text{tt} \\
\mathcal{E}(\text{false})\sigma &= \text{ff} \\
\mathcal{E}(\text{not } b)\sigma &= \neg \mathcal{E}(b)\sigma \\
\ldots &= \ldots
\end{align*}
\]
\( \text{pWhile – SOS Semantics I} \)

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

\[
\text{R3}_1 \quad \frac{\langle S_1, \sigma \rangle \Rightarrow_1 \rho \langle S_1', \sigma' \rangle}{\langle S_1 ; S_2, \sigma \rangle \Rightarrow_1 \rho \langle S_1' ; S_2, \sigma' \rangle}
\]

\[
\text{R3}_2 \quad \frac{\langle S_1, \sigma \rangle \Rightarrow_1 \rho \langle \text{stop}, \sigma' \rangle}{\langle S_1 ; S_2, \sigma \rangle \Rightarrow_1 \rho \langle S_2, \sigma' \rangle}
\]
\[ \text{pWhile} – \text{SOS Semantics II} \]

\begin{align*}
\text{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 \\
\text{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 \\
\text{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} \\
\text{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} \\
\text{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} \\
\text{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}
\end{align*}
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

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

\[
(T)_{ij} = \begin{cases} 
    p & \text{if } C_i = \langle S, \sigma \rangle \Rightarrow p \ C_j = \langle S', \sigma' \rangle \\
    0 & \text{otherwise}
\end{cases}
\]
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 \texttt{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

\[ d_n \cdot T \]

where \( d_i \) is the probability distribution over \textbf{Conf} at the \( i \)th step.
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

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

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

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

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 Conf at the \( i \)th step.
Example Program

Let us investigate the possible transitions of the following labelled program (with $x \in \{0, 1\}$):

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

\[ \langle x \mapsto 0, [x == 0]^1 \rangle \quad ... \quad \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \]

\[ \langle x \mapsto 0, [x:=0]^2 \rangle \quad ... \quad \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \]

\[ \langle x \mapsto 0, [x:=1]^3 \rangle \quad ... \quad \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \]

\[ \langle x \mapsto 0, [\text{stop}]^4 \rangle \quad ... \quad \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \]
Example Transition

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

We get:
\[
\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
\]

This represents the (deterministic) transition step:
\[
\langle x \mapsto \rightarrow 0, [x := 1] \rangle \Rightarrow 1 \langle x \mapsto \rightarrow 1, [\text{stop}] \rangle
\]
Example Transition

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

We get: \( \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \).

This represents the (deterministic) transition step:

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

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

\[ \text{Dist}(\text{Value}_1 \times \ldots \times \text{Value}_v \times \text{Lab}) \subseteq \text{V}(\text{Value}_1 \times \ldots \times \text{Value}_v \times \text{Lab}) \]

assuming $v = |\text{Var}|$ finite,

\[ \text{State} = (\mathbb{Z} + \mathbb{B})^v = \text{Value}_1 \times \ldots \times \text{Value}_v \]

Thus, we can represent the space of configurations as

\[ \text{Dist}(\text{Value}_1 \times \ldots \times \text{Value}_v \times \text{Lab}) \subseteq \text{V}(\text{Value}_1 \times \ldots \times \text{Value}_v \times \text{Lab}) = \text{V}^{\otimes} \text{Value}_1 \otimes \ldots \otimes \text{V}^{\otimes} \text{Value}_v \otimes \text{V}^{\otimes} \text{Lab} \]
The Space of Configurations

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

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

For a \texttt{PWHILE} program \textit{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,

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

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

For a 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 $\text{Value}_i$.

Thus, we can represent the space of configurations as

$$\text{Dist}(\text{Value}_1 \times \ldots \times \text{Value}_\nu \times \text{Lab}) \subseteq$$
$$\subseteq \mathcal{V}(\text{Value}_1 \times \ldots \times \text{Value}_\nu \times \text{Lab})$$
$$= \mathcal{V}(\text{Value}_1) \otimes \ldots \otimes \mathcal{V}(\text{Value}_\nu) \otimes \mathcal{V}(\text{Lab}).$$
Tensor Product

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

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

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

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

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

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

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

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

where \(\alpha, \alpha', \beta, \beta' \in \mathbb{R}\), and \(M, M' \in \mathbb{R}^{m \times m}\), \(N, N' \in \mathbb{R}^{n \times n}\).

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:

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

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

For tensor product of square matrices (linear operators):

1. The **bilinearity** property:

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

\(\alpha, \alpha', \beta, \beta' \in \mathbb{R}, M, M' \ m \times m \) matrices \(N, N' \ n \times n \) 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 **bilinearity** property:

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

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

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

\[
(\mathbf{M} \otimes \mathbf{N})(\mathbf{v} \otimes \mathbf{w}) = (\mathbf{M}(\mathbf{v})) \otimes (\mathbf{N}(\mathbf{w}))
\]

\[
(\mathbf{M} \otimes \mathbf{N})(\mathbf{M}' \otimes \mathbf{N}') = (\mathbf{M}
\mathbf{M}') \otimes (\mathbf{N}
\mathbf{N}')
\]

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

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

The transition matrix $T$ is given by:

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

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

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

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

\[
\begin{pmatrix}
1 & 1 \\
1 & 0 \\
0 & 0 \\
0 & 0
\end{pmatrix}
\begin{pmatrix}
1 & 1 & 2 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1
\end{pmatrix}^\infty
= \begin{pmatrix}
0 \\
0 \\
0 \\
1
\end{pmatrix}^t
\]
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.

$$
\begin{pmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{pmatrix}
= 
\begin{cases}
E(1, 2) \\
\end{cases}
$$

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

We can combine single steps to construct a transition graph.

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

\[
\begin{pmatrix}
E(1, 2) \\
E(1, 3)
\end{pmatrix}
\]

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

We can combine single steps to construct a transition graph.

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

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

We can combine single steps to construct a transition graph.

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

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

We can combine single steps to construct a transition graph.

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

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

We can combine single steps to construct a transition graph. 

\[
\begin{pmatrix}
0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1
\end{pmatrix}
= \sum_{i,j} E(m, n) \iff m = i \land n = j \\
1 \\
0
\]

where 

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

\[
\begin{pmatrix}
0 & \frac{1}{3} & \frac{2}{3} & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & \frac{1}{2} & \frac{1}{2} \\
0 & 0 & 0 & 1
\end{pmatrix} = \mathbf{T}
\]
Constructing the matrix for probabilistic transitions:

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

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

Constructing the matrix for probabilistic transitions:

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

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

Constructing the matrix for probabilistic transitions:

$$
egin{pmatrix}
0 & \frac{1}{3} & \frac{2}{3} & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & \frac{1}{2} & \frac{1}{2} \\
0 & 0 & 0 & 1
\end{pmatrix} = T
$$

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

Constructing the matrix for probabilistic transitions:

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

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

Constructing the matrix for probabilistic transitions:

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

Constructing the matrix for probabilistic transitions:

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

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

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.

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: $E(3, 7)$
Moves in "Turtle" Graphics

Consider only horizontal moves over eight possible positions.

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

Move from 2 to 7 or 8: $E(2, 7) + E(2, 8)$ or $\frac{1}{2}E(2, 7) + \frac{1}{2}E(2, 8)$
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 \( \mathbf{M} \) on \( x \) and the change of \( y \) described by \( \mathbf{N} \), then the combined effect on \( \langle x, y \rangle \) is given by \( \mathbf{M} \otimes \mathbf{N} \).

From \( (7, 3) \) to \( (4, 2)/(7, 2) \): \( \mathbf{E}(7, 4) \otimes \mathbf{E}(3, 2) + \mathbf{E}(7, 7) \otimes \mathbf{E}(3, 1) \)
"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: $E(5, 6) \otimes (\sum_{i=1}^{4} E(2, i))$
Transfer Functions (Edge Effects): Assignment

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

$x := 4$

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

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

$x := 4$

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

Transfer Functions (Edge Effects): Assignment

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

$x := 4$
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 $\mathbf{U}(x \leftarrow 4)$. 
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?

$x := 4$ 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 2 3 4 5 6 7 8
Transfer Functions (Edge Effects): Shift

Assume $x \in 1, .., 8$; 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 $[[x := (x - 1 + 1) \mod 8 + 1]]$. 
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 $x \in 1, \ldots, 8$; 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?

\[
x := x + 1
\]
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?

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

Assume $x \in 1, \ldots, 8$; 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 $x := x + 1$ is $U(x \leftarrow x + 1)$.

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

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

1  2  3  4  5  6  7  8

$x := x + 1$

$gives$

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

The LOS of the statement is $\begin{bmatrix} x := x + 1 \end{bmatrix} = U(x \leftarrow x + 1)$.

To avoid “overflow”: actually $\begin{bmatrix} x := ((x - 1) + 1 \mod 8) + 1 \end{bmatrix}$. 42 / 66
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 $\left[ x := x + 1 \right] = U(x \leftarrow x + 1)$.

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

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

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 $\llbracket x := x + 1 \rrbracket = U(x \leftarrow x + 1)$. To avoid “overflow”: actually $\llbracket x := ((x - 1) + 1 \text{ mod } 8) + 1 \rrbracket$. 
Transfer Functions (Edge Effects): Random Assign

Assume $x \in 1, \ldots, 8$; How do statements change its value?
Transfer Functions (Edge Effects): Random Assign

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

$$x \equiv \{4, 5\}$$
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 $x \in 1, .., 8$; How do statements change its value?

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

\[ x \leftarrow \{4, 5\} \]
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?

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

\[
\begin{pmatrix}
0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
\end{pmatrix}
\]
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 $s \in \text{Value}$. 

\begin{equation}
\begin{aligned}
(0, 1, 0, 0, 0, 0, 0, 0) \cdot \begin{bmatrix}
4
\end{bmatrix} &= (0, 0, 0, 1, 0, 0, 0, 0) \\
(0, 1, 0, 0, 0, 0, 0, 0) \cdot \begin{bmatrix}
4, 5
\end{bmatrix} &= (0, 0, 2, 0, 1, 2, 0, 0)
\end{aligned}
\end{equation}

but also what happens with distributions, e.g.

\begin{equation}
\begin{aligned}
(0, 2.3, 0, 0, 1.3, 0, 0, 0) \cdot \begin{bmatrix}
4
\end{bmatrix} &= (0, 0, 2, 0, 1, 2, 0, 0) \\
(0, 1, 0, 0, 0, 0, 0, 0) \cdot \begin{bmatrix}
4
\end{bmatrix} + (0, 0, 2, 0, 1, 2, 0, 0) &= 1 \begin{bmatrix}
4
\end{bmatrix} + 1 \begin{bmatrix}
4
\end{bmatrix}
\end{aligned}
\end{equation}
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.

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

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

but also what happens with distributions, e.g.

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

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

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

$$(0, 1, 0, 0, 0, 0, 0, 0) \cdot \begin{bmatrix} x := 4 \end{bmatrix} = (0, 0, 0, 1, 0, 0, 0, 0)$$

$$(0, 1, 0, 0, 0, 0, 0, 0) \cdot \begin{bmatrix} x ? = \{4, 5\} \end{bmatrix} = (0, 0, 0, \frac{1}{2}, \frac{1}{2}, 0, 0, 0)$$

but also what happens with distributions, e.g.

$$(0, \frac{2}{3}, 0, 0, \frac{1}{3}, 0, 0, 0) \cdot \begin{bmatrix} x := x + 1 \end{bmatrix} = (0, 0, \frac{2}{3}, 0, 0, \frac{1}{3}, 0, 0)$$

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

$$\begin{bmatrix} x ? = \{4, 5\} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} x := 4 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} x := 5 \end{bmatrix}$$
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

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

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

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

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

Consider conditional jumps or statements, e.g.

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

The branches have the following LOS:

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

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

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

\[
[y := y + 1] = I \otimes \begin{pmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0
\end{pmatrix}
\]
Conditionals

Consider conditional jumps or statements, e.g.

```plaintext
if even(x) then x := x/2 else y := y + 1 fi
```

Note: To avoid errors $a/b = \lfloor a/b \rfloor$ and $a + b = a + b \mod n$. 
Tests and Distribution Splitting

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

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

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:

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

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

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

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

Applying the semantics of both branches gives us:

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

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

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

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

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: while \([z < 100]\) do
2: \hspace{1em} \textbf{choose} \hspace{0.5em} \frac{1}{3} : [x:=3] \hspace{0.5em} \textbf{or} \hspace{0.5em} \frac{2}{3} : [x:=1] \hspace{0.5em} \textbf{ro}
3: \textbf{end while}
4: [stop]

Its probabilistic control flow is given by:

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

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

The probabilistic control flow is defined by the function:

\[
flow : \text{Stmt} \rightarrow \mathcal{P}(\text{Lab} \times [0, 1] \times \text{Lab})
\]
Flow I — Control Transfer

The probabilistic control flow is defined by the function:

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

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

\[
\begin{align*}
\text{flow(choose}^\ell \ p_1 : S_1 \ \text{or} \ p_2 : S_2) &= \text{flow}(S_1) \cup \text{flow}(S_2) \cup \\
& \cup \{(\ell, p_1, \text{init}(S_1)), (\ell, p_2, \text{init}(S_2))\} \\
\text{flow(if} [b]^{\ell} \ \text{then} \ S_1 \ \text{else} \ S_2) &= \text{flow}(S_1) \cup \text{flow}(S_2) \cup \\
& \cup \{(\ell, 1, \text{init}(S_1)), (\ell, 1, \text{init}(S_2))\} \\
\text{flow(while} [b]^{\ell} \ \text{do} \ S) &= \text{flow}(S) \cup \\
& \cup \{(\ell, 1, \text{init}(S))\} \\
& \cup \{(\ell', 1, \ell) \mid \ell' \in \text{final}(S)\}
\end{align*}
\]
A Linear Operator Semantics (LOS) based on flow

Using the \( \text{flow}(S) \) 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
A Linear Operator Semantics (LOS) based on flow

Using the \( \text{flow}(S) \) 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

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

Using the $\text{flow}(S)$ 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:

$$T(⟨ℓ_1, p, ℓ_2⟩) = I \otimes E(ℓ_1, ℓ_2) \quad \text{for } [\text{skip}]^{ℓ_1}$$
$$T(⟨ℓ_1, p, ℓ_2⟩) = U(x ← a) \otimes E(ℓ_1, ℓ_2) \quad \text{for } [x ← a]^{ℓ_1}$$
$$T(⟨ℓ_1, p, ℓ_2⟩) = \sum_{i \in r} \frac{1}{|r|} U(x ← i) \otimes E(ℓ_1, ℓ_2) \quad \text{for } [x =?= r]^{ℓ_1}$$
$$T(⟨ℓ, p, ℓ_t⟩) = P(b = true) \otimes E(ℓ, ℓ_t) \quad \text{for } [b]^ℓ$$
$$T(⟨ℓ, p, ℓ_f⟩) = P(b = false) \otimes E(ℓ, ℓ_f) \quad \text{for } [b]^ℓ$$
$$T(⟨ℓ, p_k, ℓ_k⟩) = I \otimes E(ℓ, ℓ_k) \quad \text{for } [\text{choose}]^ℓ$$
$$T(⟨ℓ, p, ℓ⟩) = I \otimes E(ℓ, ℓ) \quad \text{for } [\text{stop}]^ℓ$$

For $[b]^ℓ$ the label $ℓ_t$ denotes the label to the ‘true’ situation (e.g. then branch) and $ℓ_f$ the situation where $b$ is ‘false’.

In the case of a choose statement the different alternatives are labeled with (initial) label $ℓ_k$. 
Tests and Filters

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 $\sigma \in \text{State} = \text{Value}^v$:

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

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

$$P(e = c) = \sum_{E(e)\sigma = c} P(\sigma)$$
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 $c \in Value_k$:

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

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

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

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

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

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

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

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

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

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

\[ T(S) = \left( \begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right) \otimes \left( \begin{array}{cccc} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right) + \left( \begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{array} \right) \otimes \left( \begin{array}{cccc} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right) + \left( \begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right) \otimes \left( \begin{array}{cccc} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{array} \right) + \left( \begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right) \otimes \left( \begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{array} \right) + \left( \begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right) \otimes \left( \begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right) + \left( \begin{array}{cccc} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right) \otimes \left( \begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{array} \right) + \left( \begin{array}{cccc} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right) \otimes \left( \begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{array} \right) \]
We can compare this $\mathbf{T}(S)$ with the directly extracted operator, and indeed the two coincide.

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

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

$$flow(F) = \{(1, 1, 2), (2, 1, 3), (3, 1, 4), (4, 1, 2), (2, 1, 5), (5, 1, 5)\}$$

$$T(F) = U(m \leftarrow 1) \otimes E(1, 2) +
\quad P((n > 1)) \otimes E(2, 3) +
\quad U(m \leftarrow (m \ast n)) \otimes E(3, 4) +
\quad U(n \leftarrow (n - 1)) \otimes E(4, 2) +
\quad P((n \leq 1)) \otimes E(2, 5) +
\quad I \otimes E(5, 5)$$
Introducing PAI

The matrix $T(F)$ is very big already for small $n$.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$\dim(T(F))$</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.