Overview

Topics we will cover in this part will include:

1. Language PWHILE
2. Operational Semantics
3. Tensor Products
4. Linear Operator Semantics
5. Probabilistic Abstract Interpretation
Probabilistic Problem I: Guards and Conditionals

1: \[ m := 1 \] \[ \downdownarrows (p_1, p_2, p_3, \ldots) \rightarrow \left( \frac{1}{2}, \frac{1}{2}, \ldots \right) \]
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
Perhaps better this way? Correct? How to justify this?

Probabilistic Problem II: Abstract Evaluation

1: \[ m := 1 \] \[ \downdownarrows (p_e, p_o) \rightarrow \left( \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots \right) \]
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
Correct? How to justify this?
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.

The available data thus suggest this probability distribution:

<table>
<thead>
<tr>
<th>( \text{even}(m) )</th>
<th>( n = 1 )</th>
<th>( n = 2 )</th>
<th>( n = 3 )</th>
</tr>
</thead>
<tbody>
<tr>
<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>
<td></td>
</tr>
<tr>
<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>
<td></td>
</tr>
</tbody>
</table>

Problems in Probabilistic Program Analysis

1: \([m := 1]^1;\)  \(\triangleright (p_e, p_o) \rightarrow \left( \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots \right)\)
2: while \([n > 1]^2\) do  \(\triangleright (0, 1) \rightarrow \left( \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots \right)\)
3: \([m := m \times n]^3;\)  \(\triangleright (0, 1) \rightarrow \left( 0, \frac{1}{3}, \frac{1}{3}, \ldots \right)\)
4: \([n := n - 1]^4\)  \(\triangleright (1, 0) \rightarrow \left( 0, 0, \frac{1}{3}, \ldots \right)\)
5: end while  \(\triangleright (0, 1) \rightarrow \left( 0, 0, 0, \ldots \right)\)
6: [stop]$^5$

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.

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.

---

*(Georges-Louis Leclerc, Comte de)* Buffon’s Needle

\[
\Pr(\text{cross}) = \frac{2}{\pi} \quad \text{or} \quad \pi = \frac{2}{\Pr(\text{cross})}
\]
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.
- 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. \(2 \in \mathbb{N}\)

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. \((0, 0, \frac{1}{5}, 0, \frac{1}{5}, 0, \ldots) \in \mathcal{V}(\mathbb{N})\)

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 = 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 $\mathbb{C}$):

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

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

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

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 $K^n$ (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:

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

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):

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

Alternatively we also have

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

is equivalent to (also with other probability distributions):

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

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

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

is expressed equivalently as:

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

In general, for constant "weights" \( p \) and \( q \) (\text{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:

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

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**

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

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

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

\[
\begin{align*}
R0 & \quad \langle \text{skip}, \sigma \rangle \Rightarrow_1 \langle \text{stop}, \sigma \rangle \\
R1 & \quad \langle \text{stop}, \sigma \rangle \Rightarrow_1 \langle \text{stop}, \sigma \rangle \\
R2 & \quad \langle x := e, \sigma \rangle \Rightarrow_1 \langle \text{stop}, \sigma \rangle \left[ x \mapsto \mathcal{E}(e)\sigma \right] \\
R3' & \quad \langle x \neq r, \sigma \rangle \Rightarrow_1 \langle \text{stop}, \sigma \rangle \left[ x \mapsto r_i \in r \right] \\
R3_1 & \quad \frac{\langle S_1, \sigma \rangle \Rightarrow_1 \langle S'_1, \sigma' \rangle}{\langle S_1; S_2, \sigma \rangle \Rightarrow_1 \langle S'_1; S_2, \sigma' \rangle} \\
R3_2 & \quad \frac{\langle S_1, \sigma \rangle \Rightarrow_1 \langle \text{stop}, \sigma' \rangle}{\langle S_1; S_2, \sigma \rangle \Rightarrow_1 \langle S_2, \sigma' \rangle}
\end{align*}
\]

pWhile – SOS Semantics II

\[
\begin{align*}
R4_1 & \quad \langle \text{choose } p_1 : S_1 \text{ or } p_2 : S_2, \sigma \rangle \Rightarrow_1 \langle S_1, \sigma \rangle \\
R4_2 & \quad \langle \text{choose } p_1 : S_1 \text{ or } p_2 : S_2, \sigma \rangle \Rightarrow_1 \langle S_2, \sigma \rangle \\
R5_1 & \quad \langle \text{if } b \text{ then } S_1 \text{ else } S_2, \sigma \rangle \Rightarrow_1 \langle S_1, \sigma \rangle \quad \text{if } \mathcal{E}(b)\sigma = \text{tt} \\
R5_2 & \quad \langle \text{if } b \text{ then } S_1 \text{ else } S_2, \sigma \rangle \Rightarrow_1 \langle S_2, \sigma \rangle \quad \text{if } \mathcal{E}(b)\sigma = \text{ff} \\
R6_1 & \quad \langle \text{while } b \text{ do } S, \sigma \rangle \Rightarrow_1 \langle S; \text{ while } b \text{ do } S, \sigma \rangle \quad \text{if } \mathcal{E}(b)\sigma = \text{tt} \\
R6_2 & \quad \langle \text{while } b \text{ do } S, \sigma \rangle \Rightarrow_1 \langle \text{stop}, \sigma \rangle \quad \text{if } \mathcal{E}(b)\sigma = \text{ff}
\end{align*}
\]
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

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

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 \( \text{Conf} \) at the \( i \)th step.

Example Program

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

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

\[\langle x \mapsto 0, [x := 0] \rangle \quad \ldots \quad \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \]

\[\langle x \mapsto 1, [x := 0] \rangle \quad \ldots \quad \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \]

Example Transition

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

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

This represents the (deterministic) transition step:

\[\langle x \mapsto 0, [x := 1] \rangle \Rightarrow \langle x \mapsto 1, [\text{stop}] \rangle \]
The matrix representation of the SOS semantics of a $\text{PWHILE}$ 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 $\text{PWHILE}$ 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 $v = |\text{Var}|$ 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 $B$:

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

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

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

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:

$$(\alpha M + \alpha' M') \otimes (\beta N + \beta' N') = \alpha \beta (M \otimes N) + \alpha \beta' (M \otimes N') + \alpha' \beta (M' \otimes N) + \alpha' \beta' (M' \otimes N')$$

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

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

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

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

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

Transitions and Generator of DTMC (2) - Probabilistic

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

\[
\begin{pmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{pmatrix}^t \begin{pmatrix}
0 & \frac{1}{3} & \frac{2}{3} & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & \frac{1}{2} & \frac{1}{2} \\
0 & 0 & 0 & 1
\end{pmatrix}
\]

Transitions and Generator of DTMC (4)

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

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

Combination of Steps

We can combine single steps to construct a transition graph.

\[
\begin{pmatrix}
0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 \\
\end{pmatrix} = T = \left\{ \begin{array}{l}
\text{E}(1, 2) \\
\text{E}(1, 3) \\
\text{E}(2, 4) \\
\text{E}(3, 4) \\
\text{E}(3, 3) \\
\text{E}(4, 4) \\
\end{array} \right.
\]

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

Constructing the matrix for probabilistic transitions:

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

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

"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.
Moves in "Turtle" Graphics

Consider only horizontal moves over eight possible positions.

![Diagram of eight possible positions]

Move from 1 to 2: \( E(1, 2) \)
Move from 3 to 7: \( E(3, 7) \)
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 \( y \in \{1, \ldots, 4\} \)

![Diagram of 8x4 grid with arrows]

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

From \((1, 1)\) move 1 left and 3 up: \( E(1, 2) \otimes E(1, 4) \)
From \((7, 3)\) move \((4, 2)\): \( E(7, 4) \otimes E(3, 2) \)
From \((7, 3)\) to \((4, 2)/(7, 2)\): \( E(7, 4) \otimes E(3, 2) + E(7, 7) \otimes E(3, 1) \)
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?

$x := 4$ 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}

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

Transfer Functions (Edge Effects): Shift

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

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

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

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

\( x \, ? = \{4, 5\} \) gives

So the LOS is \( \lfloor x \, ? = \{4, 5\} \rfloor = \frac{1}{2} U(x \leftarrow 4) + \frac{1}{2} U(x \leftarrow 5) \).

Using the Linear Operators

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

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

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

but also what happens with distributions, e.g.

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

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

\[
\lfloor x ? = \{4, 5\} \rfloor = \frac{1}{2} \lfloor x := 4 \rfloor + \frac{1}{2} \lfloor x := 5 \rfloor
\]
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

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

\[
= \frac{1}{4} \left( \begin{array}{cccccccc}
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
\end{array} \right) \otimes \left( \begin{array}{cccc}
0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 \\
\end{array} \right)
\]

45 / 99

46 / 99
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:
\[
[x := x/2] = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
\end{pmatrix} \otimes I
\]
\[
y := y + 1] = I \otimes \begin{pmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 \\
\end{pmatrix}
\]
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:
\[
P(\text{even}(x)) = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{pmatrix} \otimes I
\]
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:

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

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

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

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

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

Its probabilistic control flow is given by:

\[ \text{flow}(P) = \{ \langle 1, 1, 2 \rangle, \langle 1, 1, 5 \rangle, \langle 2, \frac{1}{3}, 3 \rangle, \langle 2, \frac{2}{3}, 4 \rangle, \langle 3, 1, 1 \rangle, \langle 4, 1, 1 \rangle \}. \]

\textbf{Init Label}

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

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

\[
\begin{align*}
    \text{flow}([\text{skip}]^\ell) &= \emptyset \\
    \text{flow}([\text{stop}]^\ell) &= \{(\ell, 1, \ell)\} \\
    \text{flow}([x:=e]^\ell) &= \emptyset \\
    \text{flow}([x?=e]^\ell) &= \emptyset \\
    \text{flow}(S_1; S_2) &= \text{flow}(S_1) \cup \text{flow}(S_2) \cup \\
    &\quad \cup \{(\ell, 1, \text{init}(S_2)) | \ell \in \text{final}(S_1)\}
\end{align*}
\]
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

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

With $N_{\ell_i}$ 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:

- $T((\ell_1, p, \ell_2)) = I \otimes E(\ell_1, \ell_2)$ for $[\text{skip}]^{\ell_1}$
- $T((\ell_1, p, \ell_2)) = U(x \leftarrow a) \otimes E(\ell_1, \ell_2)$ for $[x \leftarrow a]^{\ell_1}$
- $T((\ell_1, p, \ell_2)) = \sum_{i \in r} U(x \leftarrow i) \otimes E(\ell_1, \ell_2)$ for $[x \leftarrow r]^{\ell_1}$
- $T((\ell, p, \ell_t)) = P(b = \text{true}) \otimes E(\ell, \ell_t)$ for $[b]^\ell$
- $T((\ell, p, \ell_f)) = P(b = \text{false}) \otimes E(\ell, \ell_f)$ for $[b]^\ell$
- $T((\ell, p, \ell_k)) = I \otimes E(\ell, \ell_k)$ for $[\text{choose}]^\ell$
- $T((\ell, p, \ell)) = I \otimes E(\ell, \ell)$ for $[\text{stop}]^\ell$

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

In the case of a choose statement the different alternatives are labeled with (initial) label $\ell_k$.

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 $e = a | b$ evaluates to $c$:

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

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

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

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

if $[x == 0]^1$ then

$[x \leftarrow 0]^2$;

else

$[x \leftarrow 1]^3$;

end if;

[stop]$^4$

$$T(S) = P(x = 0) \otimes E(1, 2) + P(x \neq 0) \otimes E(1, 3) + U(x \leftarrow 0) \otimes E(2, 4) + U(x \leftarrow 1) \otimes E(3, 4) + I \otimes E(4, 4)$$

$$T(S) = \begin{pmatrix}
1 & 0 \\
0 & 0
\end{pmatrix} \otimes E(1, 2) + \begin{pmatrix}
0 & 0 \\
0 & 1
\end{pmatrix} \otimes E(1, 3) +$$

$$+ \begin{pmatrix}
1 & 0 \\
1 & 0
\end{pmatrix} \otimes E(2, 3) + \begin{pmatrix}
0 & 1 \\
0 & 1
\end{pmatrix} \otimes E(3, 4) +$$

$$+ (I \otimes E(4, 4))$$
An Example

\[
T(S) = \begin{pmatrix}
1 & 0 \\
0 & 0
\end{pmatrix} \otimes \begin{pmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix} + \begin{pmatrix}
0 & 0 \\
0 & 1
\end{pmatrix} \otimes \begin{pmatrix}
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix} + \begin{pmatrix}
1 & 0 \\
1 & 0
\end{pmatrix} \otimes \begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix} + \begin{pmatrix}
0 & 1 \\
0 & 1
\end{pmatrix} \otimes \begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix} + \begin{pmatrix}
1 & 0 \\
0 & 1
\end{pmatrix} \otimes \begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix}
\]

LOS and DTMC

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

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

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

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

\[
\langle x \mapsto 0, [\text{stop}]^4 \rangle \ldots \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}
\]
Research Tool: A **pWHILE** Compiler \texttt{pwc}

Written in OCaml produces an \texttt{octave} file \texttt{c.m} which specify the LOS matrices \texttt{U}, \texttt{P}, etc. for a pWhile program \texttt{c.pw}.

\begin{figure}[h]
\centering
\begin{tikzpicture}[node distance=2cm,auto,>=latex,]
    \node (c_pw) {\texttt{c.pw}};
    \node[blue, right of=c_pw] (ocaml) {OCaml} edge[blue, ->] (c_pw); \node[below of=ocaml, yshift=-1cm] (pwc) {\texttt{pwc}} edge[blue, ->] (ocaml);
    \node[red, right of=ocaml] (octave) {\texttt{octave}} edge[red, ->] (c_pw); \node[below of=octave, yshift=-1cm] (cm) {\texttt{c.m}} edge[red, ->] (octave);
    \node[green, right of=octave] (los_m) {LOS.m};
    \draw[->] (cm) -- node[above] {\texttt{c.m}} (octave);
    \end{tikzpicture}
\end{figure}

We can use the interactive interface of \texttt{octave} and definitions of standard operations in \texttt{LOS.m} to analyse matrices in \texttt{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.

### Factorial

Consider the program \texttt{F} for calculating the factorial of \texttt{n}:

```ocaml
var
  m : {0..2};
  n : {0..2};

begin
  m := 1;
  while (n>1) do
    m := m * n;
    n := n-1;
  od;
stop; # looping
end
```

63 / 99

64 / 99
Control Flow and LOS for $F$

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

\[
\begin{align*}
T(F) &= U(m \leftarrow 1) \otimes E(1, 2) + \\
&\quad P((n > 1)) \otimes E(2, 3) + \\
&\quad U(m \leftarrow (m \ast n)) \otimes E(3, 4) + \\
&\quad U(n \leftarrow (n - 1)) \otimes E(4, 2) + \\
&\quad P((n <= 1)) \otimes E(2, 5) + \\
&\quad I \otimes E(5, 5)
\end{align*}
\]

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.
Galois Connections

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

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

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

\[
\begin{align*}
(i) \quad & \alpha \circ \gamma \circ \alpha = \alpha \\
(ii) \quad & \gamma \circ \alpha \circ \gamma = \gamma
\end{align*}
\]

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

\[
\begin{array}{ccc}
A & \xrightarrow{\alpha} & A^# \\
\downarrow f & & \downarrow f^# \\
B & \xleftarrow{\alpha'} & B^#
\end{array}
\]

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

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

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.

Norm and Distance

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 \( c \in \mathbb{C} \):

- \( \| v \| \geq 0 \),
- \( \| v \| = 0 \Leftrightarrow v = 0 \),
- \( \| cv \| = |c| \| v \| \),
- \( \| v + w \| \leq \| v \| + \| w \| \),

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 $\langle \cdot, \cdot \rangle$. 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 $EE = E$. 
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 \( S \):
\[
\mathcal{V}(S) = \left\{ \sum x_s s \mid x_s \in \mathbb{R}, s \in S \right\}
\]

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: \( \bar{\alpha} : \mathcal{V}(\mathcal{C}) \rightarrow \mathcal{V}(\mathcal{D}) \)
\[
\bar{\alpha}(p_1 \cdot \bar{c}_1 + p_2 \cdot \bar{c}_2 + \ldots) = p_i \cdot \alpha(c_1) + p_2 \cdot \alpha(c_2) \ldots
\]

Support Set: \( \text{supp} : \mathcal{V}(\mathcal{C}) \rightarrow \mathcal{P}(\mathcal{C}) \)
\[
\text{supp}(\bar{x}) = \left\{ c_i \mid \langle c_i, p_i \rangle \in \bar{x} \text{ and } p_i \neq 0 \right\}
\]
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 $\mathcal{V}(\mathcal{C})$, i.e. $\forall \vec{x} \in \mathcal{V}(\mathcal{C})$,

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

Analogously we can show that $\alpha \circ \gamma$ is reductive. Therefore,

Proposition
$(\alpha, \gamma)$ 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):

$$A_p = \begin{pmatrix}
1 & 0 \\
0 & 1 \\
1 & 0 \\
0 & 1 \\
\vdots & \vdots \\
0 & 1
\end{pmatrix} \quad A_p^\dagger = \begin{pmatrix}
\frac{2}{n} & 0 & \frac{2}{n} & 0 & \cdots & 0 \\
0 & \frac{2}{n} & 0 & \frac{2}{n} & \cdots & \frac{2}{n}
\end{pmatrix}$$

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

$$A_s = \begin{pmatrix}
1 & 0 & 0 \\
\vdots & \vdots & \vdots \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\vdots & \vdots & \vdots \\
0 & 0 & 1
\end{pmatrix} \quad A_s^\dagger = \begin{pmatrix}
\frac{1}{n} & \cdots & \frac{1}{n} & 0 & 0 & \cdots & 0 \\
0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\
0 & \cdots & 0 & 0 & \frac{1}{n} & \cdots & \frac{1}{n}
\end{pmatrix}$$
Concrete and abstract domain are step-functions on \([a, b]\). The set of (real-valued) step-function \(T_n\) is based on the sub-division of the interval into \(n\) sub-intervals.

Each step function in \(T_n\) 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

\[
A_8 = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]

\[
G_8 = \begin{pmatrix}
\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]
Approximation Estimates

Compute the *least square error* as

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

\[ \| f - f_{A_8 G_8} \| = 3.5355 \]
\[ \| f - f_{A_4 G_4} \| = 5.3151 \]
\[ \| f - f_{A_2 G_2} \| = 5.9896 \]
\[ \| f - f_{A_1 G_1} \| = 7.6444 \]

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:

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:

$$(A_1 \otimes A_2 \otimes \ldots \otimes A_n)^\dagger = A_1^\dagger \otimes A_2^\dagger \otimes \ldots \otimes A_n^\dagger$$

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

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

with local abstraction of individual variables:

$$T^\#(\ell_i, \ell_j) = (A_1^\dagger N_{i1} A_1) \otimes (A_2^\dagger N_{i2} A_2) \otimes \ldots \otimes (A_v^\dagger N_{iv} A_v) \otimes M_{ij}$$

Argument

\[
T^\# = A^\dagger T A
\]

\[
= A^\dagger \left( \sum_{i,j} T(i,j) \right) A
\]

\[
= \sum_{i,j} A^\dagger T(i,j) A
\]

\[
= \sum_{i,j} (\bigotimes_k A_k)^\dagger T(i,j) (\bigotimes_k A_k)
\]

\[
= \sum_{i,j} (\bigotimes_k A_k)^\dagger (\bigotimes_k N_{ik}) (\bigotimes_k A_k)
\]

\[
= \sum_{i,j} \bigotimes_k (A_k^\dagger N_{ik} A_k)
\]
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):

$$A_p = \begin{pmatrix}
1 & 0 \\
0 & 1 \\
1 & 0 \\
0 & 1 \\
\vdots & \vdots \\
0 & 1 
\end{pmatrix} \quad A^\dagger = \begin{pmatrix}
\frac{2}{n} & 0 & \frac{2}{n} & 0 & \cdots & 0 \\
0 & \frac{2}{n} & 0 & \frac{2}{n} & \cdots & \frac{2}{n}
\end{pmatrix}$$

Example

1: $[m \leftarrow i]^1$
2: 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

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

Abstract Semantics

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

\[
\mathbf{P}^\#(n > 1) = \\
= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & 1 \end{pmatrix}
\]

Abstract Semantics

\[
\mathbf{P}^\#(n \leq 1) = \\
= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & 0 \end{pmatrix}
\]
Abstract Semantics

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

Implementation

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

\[
\begin{array}{c|cc}
\text{d} & \text{dim(}T(F)\text{)} & \text{dim(}T^\#(F)\text{)} \\
\hline
2 & 45 & 30 \\
3 & 140 & 40 \\
4 & 625 & 50 \\
5 & 3630 & 60 \\
6 & 25235 & 70 \\
7 & 201640 & 80 \\
8 & 1814445 & 90 \\
9 & 18144050 & 100 \\
\end{array}
\]

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>0.0019980</td>
</tr>
<tr>
<td>10000</td>
<td>0.99980</td>
<td>0.00019998</td>
</tr>
</tbody>
</table>

Ortholattice of Projection Operators

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 $Y_E \subseteq Y_F$.

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.
Computing Intersections/Unions

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 \(E \cup F\) and \(E \cap F\) is in general not trivial. Only for commuting projections we have:

\[
E \cup F = E + F - EF \quad \text{and} \quad E \cap F = EF.
\]

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. \(X\chi_A\chi_A = \chi_A\chi_A\). We have \(\chi_{A \cap B} = \chi_A\chi_B\) and \(\chi_{A \cup B} = \chi_A + \chi_B - \chi_A\chi_B\).
Non-Commutative Case

The Moore-Penrose pseudo-inverse is also useful for computing the $E \cap F$ and $E \cup F$ of general, non-commuting projections via the parallel sum

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

The intersection of projections is given by:

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


---

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 $B$?

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

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

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


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