Program Analysis (70020)

Probabilistic Programs

Herbert Wiklicky

Department of Computing
Imperial College London

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

Autumn 2022

Overview

Topics we will cover in this part will include:

1. Language pWHILE
2. Operational Semantics
3. Tensor Products
4. Linear Operator Semantics
5. Probabilistic Abstract Interpretation

Probabilistic Problem I: Guards and Conditionals

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

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

Concrete Probabilities

Perhaps better this way?

Correct? How to justify this?

Probabilistic Problem II: Abstract Evaluation

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

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

Abstract Probabilities

Correct?

How to justify this?

Probabilistic Problem III: Relational Dependency

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

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

The available data thus suggest this probability distribution:

	$n = 1$	$n = 2$	$n = 3$
$even(m)$	$\frac{1}{3} \cdot \frac{2}{3}$	$\frac{1}{3} \cdot \frac{2}{3}$	$\frac{1}{3} \cdot \frac{2}{3}$
$odd(m)$	$\frac{1}{3} \cdot \frac{1}{3}$	$\frac{1}{3} \cdot \frac{1}{3}$	$\frac{1}{3} \cdot \frac{1}{3}$

Problems in Probabilistic Program Analysis

$[m := 1]^1$; $\quad \triangleright\ (p_e, p_o)$ — $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots)$

while $[n > 1]^2$ **do** $\quad \triangleright\ (0, 1)$ — $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots)$

$\quad [m := m \times n]^3$; $\quad \triangleright\ (0, 1)$ — $(0, \frac{1}{3}, \frac{1}{3}, \ldots)$

$\quad [n := n - 1]^4$ $\quad \triangleright\ (1, 0)$ — $(0, \frac{1}{3}, \frac{1}{3}, \ldots)$

end while

[**stop**]^5 $\quad \triangleright\ (0, 1)$ — $(\frac{1}{3}, 0, 0, \ldots)$

Splitting: How to distribute information along branches?
Transforming: How computing changes the information?
Joining: How to combine information along branches?

Probability and Computation

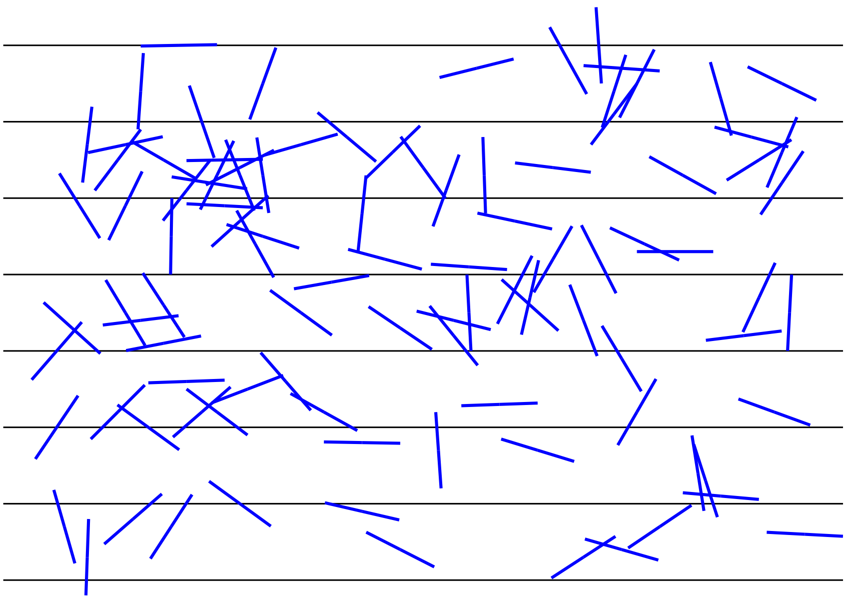
Commonly, computations are understood to follow a well defined (deterministic) set of rules as to obtain a certain result.

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

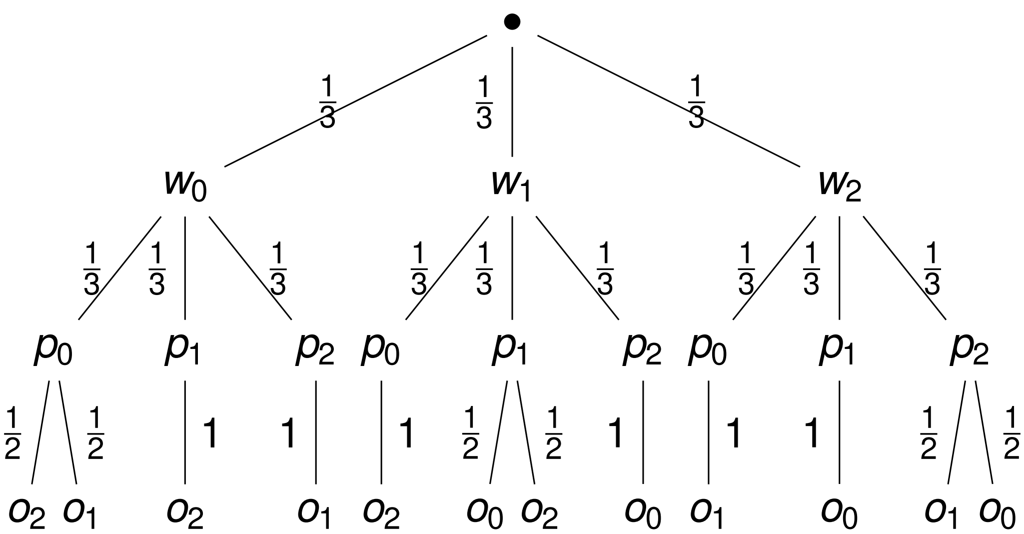


$$\mathrm{Pr}(\mathrm{cross}) = \frac{2}{\pi} \text{ or } \pi = \frac{2}{\mathrm{Pr}(\mathrm{cross})}$$

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 = win behind i p_i = pick door i o_i = 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) Ω (of states) we can construct the power set $\mathcal{P}(\Omega)$ of Ω 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 Ω is (un)countable infinite.

Structures: Vector Spaces

Vector Spaces = Abelian Additive Group + Quantities

Given a finite set Ω we can construct the (free) vector space $\mathcal{V}(\Omega)$ of Ω 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 Ω is (un)countable infinite.

Tuple Spaces

Theorem

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

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

$$\begin{aligned} x &= (x_1, x_2, x_3, \ldots, x_n) \\ y &= (y_1, y_2, y_3, \ldots, y_n) \end{aligned}$$

Algebraic Structure

$$\begin{aligned} \alpha x &= (\alpha x_1, \alpha x_2, \alpha x_3, \ldots, \alpha x_n) \\ x + y &= (x_1 + y_1, x_2 + y_2, x_3 + y_3, \ldots, x_n + y_n) \end{aligned}$$

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:

$$x \mathrel{?}= \{1, 2, 3, 4\}$$

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:

$$x \mathrel{?}= \{0, 1\}$$

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):

$$x \mathrel{?}= \{0, 1\};\ \textbf{if } (x > 0) \textbf{ then } S_1 \textbf{ else } S2 \textbf{ fi}$$

Probabilities as Ratios

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

$$\textbf{choose}\ \frac{1}{3} : S_1\ \textbf{or}\ \frac{2}{3} : S_2\ \textbf{ro}$$

is expressed equivalently as:

$$\textbf{choose}\ 1 : (x := 0)\ \textbf{or}\ 2 : (x := 1)\ \textbf{ro}$$

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

$$\textbf{choose}\ p : S_1\ \textbf{or}\ q : S_2\ \textbf{ro}$$

(by exploiting an implicit normalisation) into

$$\textbf{choose}\ \frac{p}{p+q} : S_1\ \textbf{or}\ \frac{q}{p+q} : S_2\ \textbf{ro}$$

pWhile – Concrete Syntax

The syntax of statements S is as follows:

$$\begin{array}{lcl} S & ::= & \textbf{stop} \\ & \mid & \textbf{skip} \\ & \mid & x := e \\ & \mid & x \mathrel{?=} r \\ & \mid & S_1 ;\ S_2 \\ & \mid & \textbf{choose}\ p_1 : S_1\ \textbf{or}\ p_2 : S_2\ \textbf{ro} \\ & \mid & \textbf{if}\ b\ \textbf{then}\ S_1\ \textbf{else}\ S_2\ \textbf{fi} \\ & \mid & \textbf{while}\ b\ \textbf{do}\ S\ \textbf{od} \end{array}$$

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

pWHILE – Labelled Syntax

$$
\begin{array}{rcl}
S & ::= & [\textbf{stop}]^{\ell} \\
 & | & [\textbf{skip}]^{\ell} \\
 & | & [x := e]^{\ell} \\
 & | & [x \mathrel{?\!=} r]^{\ell} \\
 & | & S_1 \, ; \; S_2 \\
 & | & \textbf{choose}^{\ell} \; p_1 : S_1 \; \textbf{or} \; p_2 : S_2 \; \textbf{ro} \\
 & | & \textbf{if} \; [b]^{\ell} \; \textbf{then} \; S_1 \; \textbf{else} \; S_2 \; \textbf{fi} \\
 & | & \textbf{while} \; [b]^{\ell} \; \textbf{do} \; S \; \textbf{od}
\end{array}
$$

Where the p_i are constants, representing choice probabilities. By r we denote a range/set, e.g. $\{-1, 0, 1\}$, from which the value of x is chosen (based on a uniform distribution).

Evaluation of Expressions [Not for Exam]

$$\sigma \ni \textbf{State} = (\textbf{Var} \rightarrow \textbf{Z} \uplus \textbf{B})$$

Evaluation $\mathcal{E}$ of expressions e in state σ:

$$
\begin{array}{rcl}
\mathcal{E}(n)\sigma & = & n \\
\mathcal{E}(x)\sigma & = & \sigma(x) \\
\mathcal{E}(a_1 \odot a_2)\sigma & = & \mathcal{E}(a_1)\sigma \odot \mathcal{E}(a_2)\sigma
\end{array}
$$

$$
\begin{array}{rcl}
\mathcal{E}(\textbf{true})\sigma & = & \textbf{tt} \\
\mathcal{E}(\textbf{false})\sigma & = & \textbf{ff} \\
\mathcal{E}(\textbf{not } b)\sigma & = & \neg\mathcal{E}(b)\sigma \\
\ldots & = & \ldots
\end{array}
$$

pWhile – SOS Semantics I [Not for Exam]

R0	$\langle \textbf{skip}, \sigma\rangle \Rightarrow_1 \langle \textbf{stop}, \sigma\rangle$
R1	$\langle \textbf{stop}, \sigma\rangle \Rightarrow_1 \langle \textbf{stop}, \sigma\rangle$
R2	$\langle \mathtt{x} := e, \sigma\rangle \Rightarrow_1 \langle \textbf{stop}, \sigma[x \mapsto \mathcal{E}(e)\sigma]\rangle$
R3'	$\langle \mathtt{x} ?= r, \sigma\rangle \Rightarrow_{\frac{1}{\vert r\vert}} \langle \textbf{stop}, \sigma[x \mapsto r_i \in r]\rangle$
R3$_1$	$\dfrac{\langle S_1, \sigma\rangle \Rightarrow_p \langle S_1', \sigma'\rangle}{\langle S_1; S_2, \sigma\rangle \Rightarrow_p \langle S_1'; S_2, \sigma'\rangle}$
R3$_2$	$\dfrac{\langle S_1, \sigma\rangle \Rightarrow_p \langle \textbf{stop}, \sigma'\rangle}{\langle S_1; S_2, \sigma\rangle \Rightarrow_p \langle S_2, \sigma'\rangle}$

pWhile – SOS Semantics II [Not for Exam]

R4$_1$	$\langle \textbf{choose } p_1 : S_1 \textbf{ or } p_2 : S_2, \sigma\rangle \Rightarrow_{p_1} \langle S_1, \sigma\rangle$	
R4$_2$	$\langle \textbf{choose } p_1 : S_1 \textbf{ or } p_2 : S_2, \sigma\rangle \Rightarrow_{p_2} \langle S_2, \sigma\rangle$	
R5$_1$	$\langle \textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2, \sigma\rangle \Rightarrow_1 \langle S_1, \sigma\rangle$	if $\mathcal{E}(b)\sigma = \textbf{tt}$
R5$_2$	$\langle \textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2, \sigma\rangle \Rightarrow_1 \langle S_2, \sigma\rangle$	if $\mathcal{E}(b)\sigma = \textbf{ff}$
R6$_1$	$\langle \textbf{while } b \textbf{ do } S, \sigma\rangle \Rightarrow_1 \langle S; \textbf{ while } b \textbf{ do } S, \sigma\rangle$	if $\mathcal{E}(b)\sigma = \textbf{tt}$
R6$_2$	$\langle \textbf{while } b \textbf{ do } S, \sigma\rangle \Rightarrow_1 \langle \textbf{stop}, \sigma\rangle$	if $\mathcal{E}(b)\sigma = \textbf{ff}$

DTMC Semantics

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

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

is the generator of a Discrete Time Markov Chain.

Transitions are implemented as

$$\mathbf{d}_n \cdot \mathbf{T} = \sum_i (\mathbf{d}_n)_i \cdot \mathbf{T}_{ij} = \mathbf{d}_{n+1}$$

where $\mathbf{d}_i$ 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 $\mathbf{x} \in \{0, 1\}$):

if $[\mathbf{x} == 0]^1$ **then**
    $[\mathbf{x} := 0]^2$;
else
    $[\mathbf{x} := 1]^3$;
end if;
$[\textbf{stop}]^4$

Example DTMC

$$\begin{array}{cc} \langle x \mapsto 0, [\mathbf{x} == 0]^1 \rangle & \ldots \\ \langle x \mapsto 0, [\mathbf{x} := 0]^2 \rangle & \ldots \\ \langle x \mapsto 0, [\mathbf{x} := 1]^3 \rangle & \ldots \\ \langle x \mapsto 0, [\mathbf{stop}]^4 \rangle & \ldots \\ \langle x \mapsto 1, [\mathbf{x} == 0]^1 \rangle & \ldots \\ \langle x \mapsto 1, [\mathbf{x} := 0]^2 \rangle & \ldots \\ \langle x \mapsto 1, [\mathbf{x} := 1]^3 \rangle & \ldots \\ \langle x \mapsto 1, [\mathbf{stop}]^4 \rangle & \ldots \end{array} \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Example Transition

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

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

This represents the (deterministic) transition step:

$$\langle x \mapsto 0, [\mathbf{x} := 1]^3 \rangle \Rightarrow_1 \langle x \mapsto 1, [\mathbf{stop}]^4 \rangle$$

Linear Operator Semantics (LOS)

The matrix representation of the SOS semantics of a pWHILE program is not 'compositional'.

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

The Space of Configurations

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

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

Assuming $v = |\mathbf{Var}|$ finite,

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

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

Thus, we can represent the space of configurations as

$$\begin{aligned} &\mathbf{Dist}(\mathbf{Value}_1 \times \ldots \times \mathbf{Value}_v \times \mathbf{Lab}) \subseteq \\ \subseteq\quad & \mathcal{V}(\mathbf{Value}_1 \times \ldots \times \mathbf{Value}_v \times \mathbf{Lab}) \\ =\quad & \mathcal{V}(\mathbf{Value}_1) \otimes \ldots \otimes \mathcal{V}(\mathbf{Value}_v) \otimes \mathcal{V}(\mathbf{Lab}). \end{aligned}$$

Tensor Product

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

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

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

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

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:

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

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

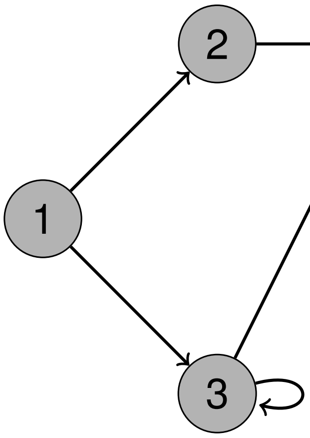
2. We have, with $v \in \mathbb{R}^m$ and $w \in \mathbb{R}^n$:

$$(\mathbf{M} \otimes \mathbf{N})(v \otimes w) = (\mathbf{M}(v)) \otimes (\mathbf{N}(w))$$
$$(\mathbf{M} \otimes \mathbf{N})(\mathbf{M}' \otimes \mathbf{N}') = (\mathbf{M}\mathbf{M}') \otimes (\mathbf{N}\mathbf{N}')$$

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

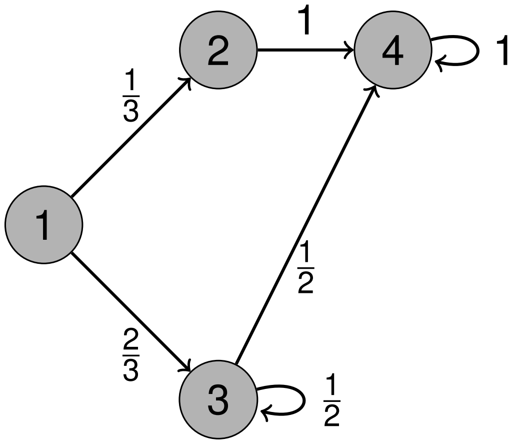
$$(\mathbf{M} \otimes \mathbf{N})^{-1} = \mathbf{M}^{-1} \otimes \mathbf{N}^{-1}$$

Transitions and Generator of DTMC (1) - Deterministic



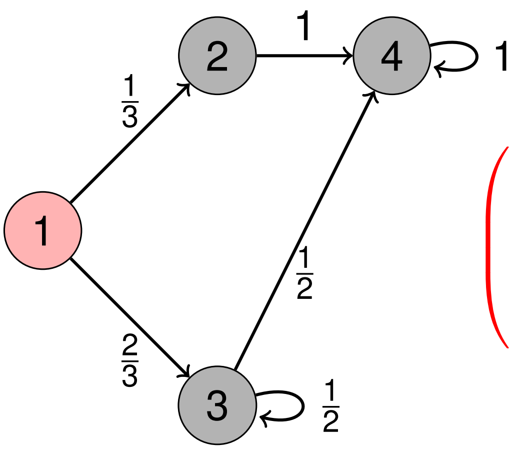
$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \mathbf{T}$$

Transitions and Generator of DTMC (2) - Probabilistic



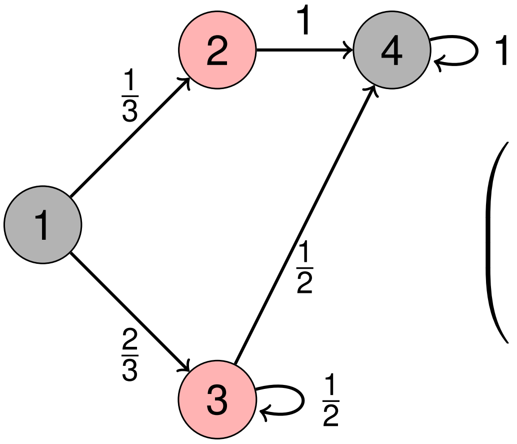
$$\begin{pmatrix} 0 & \frac{1}{3} & \frac{2}{3} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix} = \mathbf{T}$$

Transitions and Generator of DTMC (3)



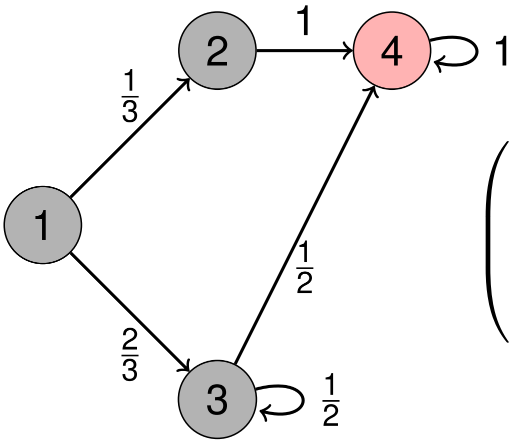
$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}^t \begin{pmatrix} 0 & \frac{1}{3} & \frac{2}{3} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Transitions and Generator of DTMC (4)



$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}^t \begin{pmatrix} 0 & \frac{1}{3} & \frac{2}{3} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{1}{3} \\ \frac{2}{3} \\ 0 \end{pmatrix}^t$$

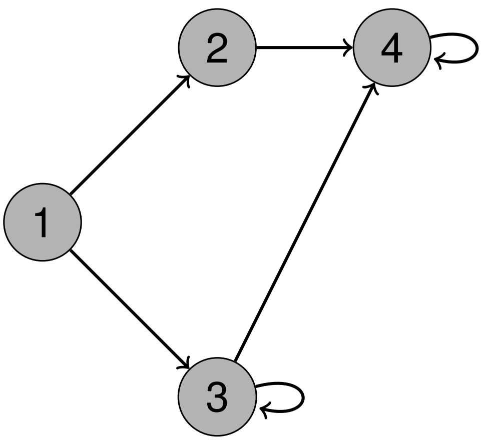
Transitions and Generator of DTMC (5)



$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}^t \begin{pmatrix} 0 & \frac{1}{3} & \frac{2}{3} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix}^\infty = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}^t$$

Combination of Steps

We can combine single steps to construct a transition graph.

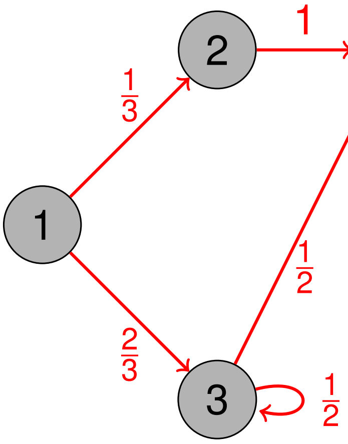


$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \mathbf{T} = \left\{ \begin{array}{ll} & \mathbf{E}(1,2) \\ + & \mathbf{E}(1,3) \\ + & \mathbf{E}(2,4) \\ + & \mathbf{E}(3,4) \\ + & \mathbf{E}(3,3) \\ + & \mathbf{E}(4,4) \end{array} \right.$$

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

Probabilistic Transitions

Constructing the matrix for probabilistic transitions:

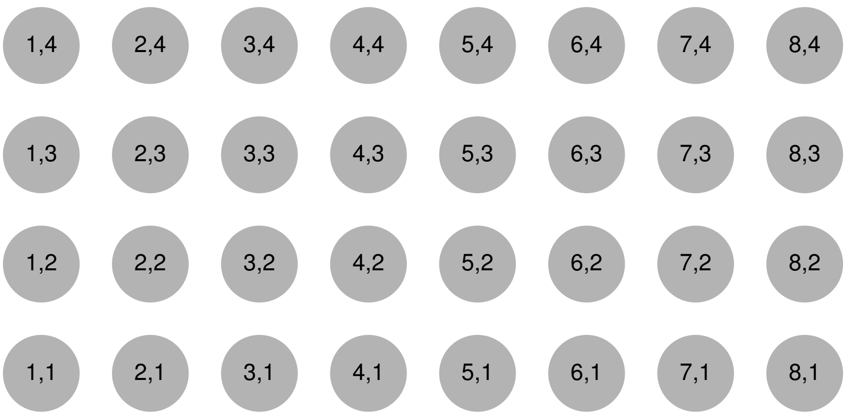


$$\begin{pmatrix} 0 & \frac{1}{3} & \frac{2}{3} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix} = \mathbf{T}$$

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

"Turtle" Graphics

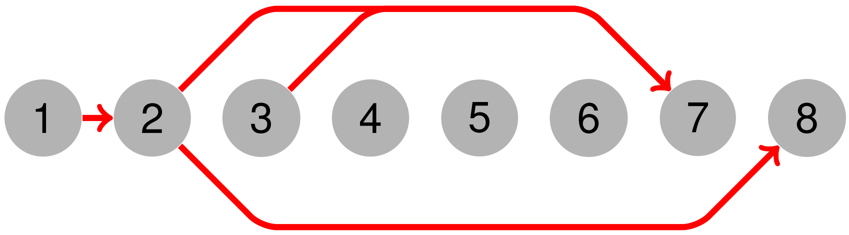
Consider a "(probabilistic) turtle graphics" with up/down and left/right moves done simultaneously and probabilistically.



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

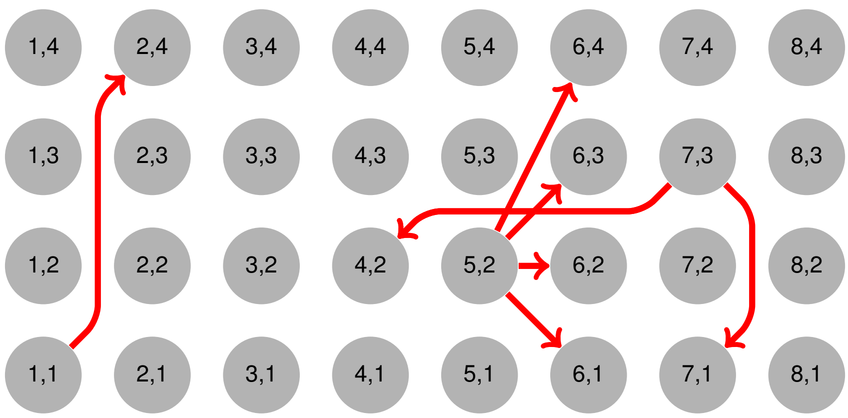
Consider only horizontal moves over eight possible positions.



Move from 1 to 2: $\mathbf{E}(1,2)$
Move from 3 to 7: $\mathbf{E}(3,7)$
Move from 2 to 7 or 8: $\mathbf{E}(2,7) + \mathbf{E}(2,8)$ or $\frac{1}{2}\mathbf{E}(2,7) + \frac{1}{2}\mathbf{E}(2,8)$

Similar representation also for vertical moves.

"Parallel" Execution: $x \in \{1,\dots,8\}$ and $y \in \{1,\dots,4\}$

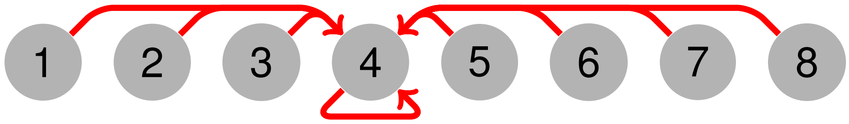


Describe the effect $\mathbf{M}$ on x and the change of y described by $\mathbf{N}$, then the combined effect on $\langle x, y\rangle$ is given by $\mathbf{M} \otimes \mathbf{N}$.

From $(1,1)$ move 1 left and 3 up: $\mathbf{E}(1,2) \otimes \mathbf{E}(1,4)$
From $(7,3)$ move $(4,2)$: $\mathbf{E}(7,4) \otimes \mathbf{E}(3,2)$
From $(7,3)$ to $(4,2)$/$(7,2)$: $\mathbf{E}(7,4) \otimes \mathbf{E}(3,2) + \mathbf{E}(7,7) \otimes \mathbf{E}(3,1)$
From $(5,2)$ move to all one right: $\mathbf{E}(5,6) \otimes (\sum_{i=1}^{4} \mathbf{E}(2,i))$

Transfer Functions (Edge Effects): Assignment

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

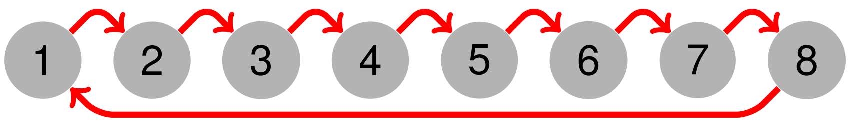


$$x \ := \ 4 \quad \text{gives} \quad \mathbf{U}(x \leftarrow 4) = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Thus, the LOS of the statement is $[\![x \ := \ 4]\!] = \mathbf{U}(x \leftarrow 4)$.

Transfer Functions (Edge Effects): Shift

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

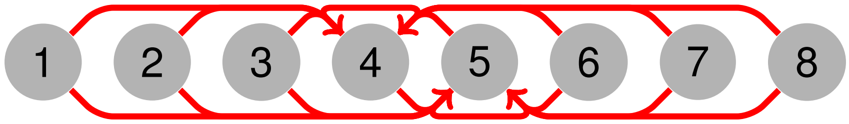


$$x \ := \ x + 1 \quad \text{gives} \quad \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

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

Transfer Functions (Edge Effects): Random Assign

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



$$x\ ?=\ \{4,5\} \quad \text{gives} \quad \begin{pmatrix} 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \end{pmatrix}$$

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

Using the Linear Operators

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

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

$$\begin{aligned} (0,1,0,0,0,0,0,0) \cdot [\![x := 4]\!] &= (0,0,0,1,0,0,0,0) \\ (0,1,0,0,0,0,0,0) \cdot [\![x? = \{4,5\}]\!] &= (0,0,0,\frac{1}{2},\frac{1}{2},0,0,0) \end{aligned}$$

but also what happens with distributions, e.g.

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

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

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

Putting Things Together

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

$$\begin{aligned}
[\![x? = \{2,4,6,8\}]\!] &= \frac{1}{4}\sum_{k=1}^{4} \mathbf{U}(x \leftarrow 2k) \otimes \mathbf{I} \\
[\![y := 3]\!] &= \mathbf{I} \otimes \mathbf{U}(y \leftarrow 3)
\end{aligned}$$

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

$$\begin{aligned}
[\![x? = \{2,4,6,8\};\ y := 3]\!] &= (\frac{1}{4}\sum_{k=1}^{4} \mathbf{U}(x \leftarrow 2k) \otimes \mathbf{I})(\mathbf{I} \otimes \mathbf{U}(y \leftarrow 3)) \\
&= \frac{1}{4}\sum_{k=1}^{4} \mathbf{U}(x \leftarrow 2k) \otimes \mathbf{U}(y \leftarrow 3)
\end{aligned}$$

"Turtle" Execution

$$\begin{aligned}
&[\![x? = \{2,4,6,8\};\ y := 3]\!] = \\
&= \frac{1}{4}\sum_{k=1}^{4} \mathbf{U}(x \leftarrow 2k) \otimes \mathbf{U}(y \leftarrow 3) \\
&= \frac{1}{4}\begin{pmatrix}
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1
\end{pmatrix} \otimes \begin{pmatrix}
0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0
\end{pmatrix}
\end{aligned}$$

Conditionals

Consider conditional jumps or statements, e.g.

$$\textbf{if } \mathit{even}(x) \textbf{ then } x := x/2 \textbf{ else } y := y+1 \textbf{ fi}$$

The branches have the following LOS:

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

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

Note: To avoid errors $a/b = \lceil a/b \rceil$ and $a+b = a+b \bmod n$.

Tests and Distribution Splitting

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

$$\mathbf{P}(\mathit{even}(x)) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \otimes \mathbf{I}$$

Its negation is represented by:

$$\mathbf{P}(\neg \mathit{even}(x)) = \mathbf{P}(\mathit{even}(x))^{\perp} = \mathbf{I} - \mathbf{P}(\mathit{even}(x)).$$

Using Tests

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

$$
\begin{aligned}
&[\![\textbf{if } \mathit{even}(x) \textbf{ then } x := x/2 \textbf{ else } y+1 \textbf{ fi}]\!] = \\
&\quad = \quad \mathbf{P}(\mathit{even}(x)) \cdot [\![x := x/2]\!] + \mathbf{P}(\mathit{even}(x))^{\perp} \cdot [\![y := y+1]\!]
\end{aligned}
$$

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

$$
\begin{aligned}
\sigma_0 \cdot \mathbf{P}(\mathit{even}(x)) &= (0,0,0,0,0,\frac{1}{2},0,0) \otimes (0,1,0,0) \\
&= \frac{1}{2} \cdot (0,0,0,0,0,1,0,0) \otimes (0,1,0,0) \\
\sigma_0 \cdot \mathbf{P}(\mathit{even}(x))^{\perp} &= (0,0,\frac{1}{2},0,0,0,0,0) \otimes (0,1,0,0) \\
&= \frac{1}{2} \cdot (0,0,1,0,0,0,0,0) \otimes (0,1,0,0)
\end{aligned}
$$

Semantics of Conditionals

Applying the semantics of both branches gives us:

$$
\begin{aligned}
&\sigma_0 \cdot \mathbf{P}(\mathit{even}(x)) \cdot [\![x := x/2]\!] = \\
&\quad = \quad (0,0,\frac{1}{2},0,0,0,0) \otimes (0,1,0,0) \\
&\sigma_0 \cdot \mathbf{P}(\mathit{even}(x))^{\perp} \cdot [\![y := y+1]\!] = \\
&\quad = \quad (0,0,\frac{1}{2},0,0,0,0,0) \otimes (0,0,1,0)
\end{aligned}
$$

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.

Probabilistic Control Flow

Consider the following labelled program:

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

Its probabilistic control flow is given by:

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

Init Label

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

Final Labels

$$
\begin{array}{rcl}
\textit{final}([\textbf{skip}]^{\ell}) & = & \{\ell\} \\
\textit{final}([\textbf{stop}]^{\ell}) & = & \{\ell\} \\
\textit{final}([\mathrm{x}{:=}e]^{\ell}) & = & \{\ell\} \\
\textit{final}([\mathrm{x}?{=}e]^{\ell}) & = & \{\ell\} \\
\textit{final}(S_1; S_2) & = & \textit{final}(S_2) \\
\textit{final}(\textbf{choose}^{\ell}\ p_1 : S_1\ \textbf{or}\ p_2 : S_2) & = & \textit{final}(S_1) \cup \textit{final}(S_2) \\
\textit{final}(\textbf{if}\ [b]^{\ell}\ \textbf{then}\ S_1\ \textbf{else}\ S_2) & = & \textit{final}(S_1) \cup \textit{final}(S_2) \\
\textit{final}(\textbf{while}\ [b]^{\ell}\ \textbf{do}\ S) & = & \{\ell\}
\end{array}
$$

Flow I — Control Transfer

The probabilistic control flow is defined by the function:

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

$$
\begin{array}{rcl}
\textit{flow}([\textbf{skip}]^{\ell}) & = & \emptyset \\
\textit{flow}([\textbf{stop}]^{\ell}) & = & \{\langle \ell, 1, \ell \rangle\} \\
\textit{flow}([\mathrm{x}{:=}e]^{\ell}) & = & \emptyset \\
\textit{flow}([\mathrm{x}?{=}e]^{\ell}) & = & \emptyset \\
\textit{flow}(S_1; S_2) & = & \textit{flow}(S_1) \cup \textit{flow}(S_2) \cup \\
 & \cup & \{(\ell, 1, \textit{init}(S_2)) \mid \ell \in \textit{final}(S_1)\}
\end{array}
$$

Flow II — Control Transfer

$$\begin{array}{rcl}
\mathit{flow}(\mathbf{choose}^{\ell}\ p_1 : S_1\ \mathbf{or}\ p_2 : S_2) & = & \mathit{flow}(S_1) \cup \mathit{flow}(S_2) \cup \\
& \cup & \{(\ell, p_1, \mathit{init}(S_1)), (\ell, p_2, \mathit{init}(S_2))\} \\
\mathit{flow}(\mathbf{if}\ [b]^{\ell}\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2) & = & \mathit{flow}(S_1) \cup \mathit{flow}(S_2) \cup \\
& \cup & \{(\ell, 1, \mathit{init}(S_1)), (\ell, 1, \mathit{init}(S_2))\} \\
\mathit{flow}(\mathbf{while}\ [b]^{\ell}\ \mathbf{do}\ S) & = & \mathit{flow}(S) \cup \\
& \cup & \{(\ell, 1, \mathit{init}(S))\} \\
& \cup & \{(\ell', 1, \ell) \mid \ell' \in \mathit{final}(S)\}
\end{array}$$

A Linear Operator Semantics (LOS) based on *flow*

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

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

where

$$\mathbf{T}(\langle \ell_i, p_{ij}, \ell_j \rangle) = \mathbf{N}_{\ell_i} \otimes \mathbf{E}(\ell_i, \ell_j),$$

With $\mathbf{N}_{\ell_1}$ the operator representing a state update (change of variable values) at the block with label ℓ_i and the second factor implementing the transfer of control from label ℓ_i to label ℓ_j.

Transfer Operators

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

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

For $[b]^\ell$ 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 $c \in \mathbf{Value}_k$ for variable x_k (with $k = 1, \ldots, v$):

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

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

$$\mathbf{P}(\sigma) = \bigotimes_{i=1}^{v} \mathbf{P}(\sigma(\mathrm{x}_i))$$

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

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

Updates

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

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

Set value of variable $\mathtt{x}_k \in \mathbf{Var}$ to constant $c \in \mathbf{Value}$:

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

Set value of variable $\mathtt{x}_k \in \mathbf{Var}$ to value given by $e = a \mid b$:

$$\mathbf{U}(\mathtt{x}_k \leftarrow e) = \sum_c \mathbf{P}(e = c)\mathbf{U}(\mathtt{x}_k \leftarrow c)$$

An Example

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

$$\begin{aligned} \mathbf{T}(S) &= \mathbf{P}(\mathtt{x} = 0) \otimes \mathbf{E}(1,2) + \\ &+ \mathbf{P}(\mathtt{x} \neq 0) \otimes \mathbf{E}(1,3) + \\ &+ \mathbf{U}(x \leftarrow 0) \otimes \mathbf{E}(2,4) + \\ &+ \mathbf{U}(x \leftarrow 1) \otimes \mathbf{E}(3,4) + \\ &+ \mathbf{I} \otimes \mathbf{E}(4,4) \end{aligned}$$

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

An Example

$$\mathbf{T}(S) = \left(\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\right) + \left(\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\right) + \left(\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\right) + \left(\begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}\right) + \left(\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\right)$$

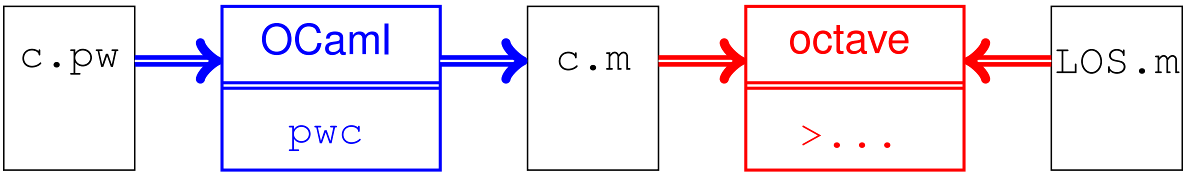
LOS and DTMC

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

$$\begin{array}{cc} \langle x \mapsto 0, [\mathbf{x} == 0]^1 \rangle & \ldots \\ \langle x \mapsto 0, [\mathbf{x}{:=}0]^2 \rangle & \ldots \\ \langle x \mapsto 0, [\mathbf{x}{:=}1]^3 \rangle & \ldots \\ \langle x \mapsto 0, [\mathbf{stop}]^4 \rangle & \ldots \\ \langle x \mapsto 1, [\mathbf{x} == 0]^1 \rangle & \ldots \\ \langle x \mapsto 1, [\mathbf{x}{:=}0]^2 \rangle & \ldots \\ \langle x \mapsto 1, [\mathbf{x}{:=}1]^3 \rangle & \ldots \\ \langle x \mapsto 1, [\mathbf{stop}]^4 \rangle & \ldots \end{array} \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Research Tool: A pWHILE Compiler `pwc`

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.

Factorial

Consider the program *F* for calculating the factorial of *n*:

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

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

Control Flow and LOS for F

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

$$\begin{aligned}\mathbf{T}(F) = \; & \mathbf{U}(m \leftarrow 1) \otimes \mathbf{E}(1,2) + \\ & \mathbf{P}((n > 1)) \otimes \mathbf{E}(2,3) + \\ & \mathbf{U}(m \leftarrow (m * n)) \otimes \mathbf{E}(3,4) + \\ & \mathbf{U}(n \leftarrow (n - 1)) \otimes \mathbf{E}(4,2) + \\ & \mathbf{P}((n <= 1)) \otimes \mathbf{E}(2,5) + \\ & \mathbf{I} \otimes \mathbf{E}(5,5)\end{aligned}$$

Introducing PAI

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

n	$\dim(\mathbf{T}(F))$
2	45×45
3	140×140
4	625×625
5	3630×3630
6	25235×25235
7	201640×201640
8	1814445×1814445
9	18144050×18144050

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

Galois Connections

Definition

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

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

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

Proposition

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

$$(i)\ \alpha \circ \gamma \circ \alpha = \alpha \quad \textit{and} \quad (ii)\ \gamma \circ \alpha \circ \gamma = \gamma$$

General Construction

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

$$\begin{tikzcd}
\mathcal{A} \arrow[r, shift left, "\alpha"] \arrow[d, "f"'] & \mathcal{A}^\# \arrow[l, shift left, "\gamma"] \arrow[d, "f^\#"] \\
\mathcal{B} \arrow[r, shift left, "\alpha'"] & \mathcal{B}^\# \arrow[l, shift left, "\gamma'"]
\end{tikzcd}$$

Correct approximation:

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

Induced semantics:

$$f^\# = \alpha' \circ f \circ \gamma.$$

Probabilistic Abstraction Domains

A probabilistic domain is essentially a vector space which represents the distributions $\mathbf{Dist}(\mathbf{State}) \subseteq \mathcal{V}(\mathbf{State})$ on the state space **State** of a probabilistic transition system, i.e. for finite state spaces

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

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

The notion of norm (distance) is essential for our treatment; we will consider normed vector spaces.

Norm and Distance [Not for Exam]

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

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

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

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

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

Moore-Penrose Generalised Inverse

Definition

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

(i) $\mathbf{A} \circ \mathbf{G} = \mathbf{P}_A$,

(ii) $\mathbf{G} \circ \mathbf{A} = \mathbf{P}_G$,

where $\mathbf{P}_A$ and $\mathbf{P}_G$ denote orthogonal projections onto the ranges of $\mathbf{A}$ and $\mathbf{G}$.

(Orthogonal) Projections – Idempotents

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

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

It also allows us to define an adjoint via:

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

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

Least Squares Solutions

Corollary

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

Definition

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

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

Theorem

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

Vector Space Lifting

Free vector space construction on a set S:

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

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

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

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

Support Set: $\mathbf{supp} : \mathcal{V}(\mathcal{C}) \to \mathcal{P}(\mathcal{C})$

$$\mathbf{supp}(\vec{x}) = \{c_i \mid \langle c_i, p_i \rangle \in \vec{x} \text{ and } p_i \neq 0\}$$

Relation with Classical Abstractions

Lemma

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

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

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

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

Proposition

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

Examples of Lifted Abstractions

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

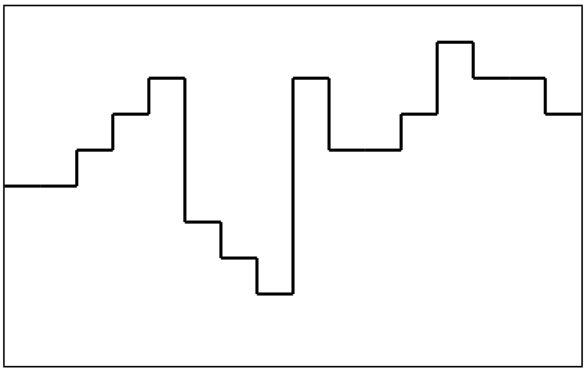
$$\mathbf{A}_p = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 0 & 1 \end{pmatrix} \qquad \mathbf{A}_p^{\dagger} = \begin{pmatrix} \frac{2}{n} & 0 & \frac{2}{n} & 0 & \ldots & 0 \\ 0 & \frac{2}{n} & 0 & \frac{2}{n} & \ldots & \frac{2}{n} \end{pmatrix}$$

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

$$\mathbf{A}_s = \begin{pmatrix} 1 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 1 \end{pmatrix} \qquad \mathbf{A}_s^{\dagger} = \begin{pmatrix} \frac{1}{n} & \ldots & \frac{1}{n} & 0 & 0 & \ldots & 0 \\ 0 & \ldots & 0 & 1 & 0 & \ldots & 0 \\ 0 & \ldots & 0 & 0 & \frac{1}{n} & \ldots & \frac{1}{n} \end{pmatrix}$$

Example: Function Approximation (ctd.)

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



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

$$\begin{pmatrix} 5 & 5 & 6 & 7 & 8 & 4 & 3 & 2 & 8 & 6 & 6 & 7 & 9 & 8 & 8 & 7 \end{pmatrix}$$

Example: Abstraction Matrices

$$\mathbf{A}_8 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{G}_8 = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

Approximation Estimates

Compute the *least square error* as

$$\|f - f\mathbf{AG}\|.$$

$$\begin{aligned}
\|f - f\mathbf{A}_8\mathbf{G}_8\| &= 3.5355 \\
\|f - f\mathbf{A}_4\mathbf{G}_4\| &= 5.3151 \\
\|f - f\mathbf{A}_2\mathbf{G}_2\| &= 5.9896 \\
\|f - f\mathbf{A}_1\mathbf{G}_1\| &= 7.6444
\end{aligned}$$

Tensor Product Properties

The tensor product of n linear operators $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_n$ is associative (but in general not commutative) and has e.g. the following properties:

1. $(\mathbf{A}_1 \otimes \ldots \otimes \mathbf{A}_n) \cdot (\mathbf{B}_1 \otimes \ldots \otimes \mathbf{B}_n) =$
   $= \mathbf{A}_1 \cdot \mathbf{B}_1 \otimes \ldots \otimes \mathbf{A}_n \cdot \mathbf{B}_n$
2. $\mathbf{A}_1 \otimes \ldots \otimes (\alpha\mathbf{A}_i) \otimes \ldots \otimes \mathbf{A}_n =$
   $= \alpha(\mathbf{A}_1 \otimes \ldots \otimes \mathbf{A}_i \otimes \ldots \otimes \mathbf{A}_n)$
3. $\mathbf{A}_1 \otimes \ldots \otimes (\mathbf{A}_i + \mathbf{B}_i) \otimes \ldots \otimes \mathbf{A}_n =$
   $= (\mathbf{A}_1 \otimes \ldots \otimes \mathbf{A}_i \otimes \ldots \otimes \mathbf{A}_n) + (\mathbf{A}_1 \otimes \ldots \otimes \mathbf{B}_i \otimes \ldots \otimes \mathbf{A}_n)$
4. $(\mathbf{A}_1 \otimes \ldots \otimes \mathbf{A}_i \otimes \ldots \otimes \mathbf{A}_n)^\dagger =$
   $= \mathbf{A}_1^\dagger \otimes \ldots \otimes \mathbf{A}_i^\dagger \otimes \ldots \otimes \mathbf{A}_n^\dagger$

Abstract Semantics

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

$$(\mathbf{A}_1 \otimes \mathbf{A}_2 \otimes \ldots \otimes \mathbf{A}_n)^\dagger = \mathbf{A}_1^\dagger \otimes \mathbf{A}_2^\dagger \otimes \ldots \otimes \mathbf{A}_n^\dagger$$

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

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

with local abstraction of individual variables:

$$\mathbf{T}^\#(\ell_i, \ell_j) = (\mathbf{A}_1^\dagger \mathbf{N}_{i1} \mathbf{A}_1) \otimes (\mathbf{A}_2^\dagger \mathbf{N}_{i2} \mathbf{A}_2) \otimes \ldots \otimes (\mathbf{A}_v^\dagger \mathbf{N}_{iv} \mathbf{A}_v) \otimes \mathbf{M}_{ij}$$

Argument [Not for Exam]

$$\begin{aligned}
\mathbf{T}^\# &= \mathbf{A}^\dagger \mathbf{T} \mathbf{A} \\
&= \mathbf{A}^\dagger (\sum_{i,j} \mathbf{T}(i,j)) \mathbf{A} \\
&= \sum_{i,j} \mathbf{A}^\dagger \mathbf{T}(i,j) \mathbf{A} \\
&= \sum_{i,j} (\bigotimes_k \mathbf{A}_k)^\dagger \mathbf{T}(i,j) (\bigotimes_k \mathbf{A}_k) \\
&= \sum_{i,j} (\bigotimes_k \mathbf{A}_k)^\dagger (\bigotimes_k \mathbf{N}_{ik}) (\bigotimes_k \mathbf{A}_k) \\
&= \sum_{i,j} \bigotimes_k (\mathbf{A}_k^\dagger \mathbf{N}_{ik} \mathbf{A}_k)
\end{aligned}$$

Parity Analysis

Determine at each program point whether a variable is *even* or *odd*.
Parity Abstraction operator on $\mathcal{V}(\{0, \dots, n\})$ (with n even):

$$\mathbf{A}_p = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 0 & 1 \end{pmatrix} \qquad \mathbf{A}^\dagger = \begin{pmatrix} \frac{2}{n} & 0 & \frac{2}{n} & 0 & \dots & 0 \\ 0 & \frac{2}{n} & 0 & \frac{2}{n} & \dots & \frac{2}{n} \end{pmatrix}$$

Example

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

$$\begin{aligned} \mathbf{T}^\# \quad = \quad & \mathbf{U}^\#(\mathrm{m} \leftarrow i) \otimes \mathbf{E}(1,2) \\ + \quad & \mathbf{P}^\#(n > 1) \otimes \mathbf{E}(2,3) \\ + \quad & \mathbf{P}^\#(n \leq 1) \otimes \mathbf{E}(2,5) \\ + \quad & \mathbf{U}^\#(\mathrm{m} \leftarrow m \times n) \otimes \mathbf{E}(3,4) \\ + \quad & \mathbf{U}^\#(\mathrm{n} \leftarrow n - 1) \otimes \mathbf{E}(4,2) \\ + \quad & \mathbf{I}^\# \otimes \mathbf{E}(5,5) \end{aligned}$$

Abstract Semantics

$$\mathbf{U}^{\#}(m \leftarrow 1) =$$

$$= \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & & \dots & 1 \end{pmatrix}$$

Abstract Semantics

$$\mathbf{U}^{\#}(n \leftarrow n - 1) =$$

$$= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 0 & 0 & \dots & 0 \\ 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 \end{pmatrix}$$

Abstract Semantics

$$\mathbf{P}^{\#}(n > 1) =$$

$$= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 \end{pmatrix}$$

Abstract Semantics

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

$$= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 \end{pmatrix}$$

Abstract Semantics

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

$$+ \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & \ddots \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & \ddots \end{pmatrix}$$

Implementation

Implementation of concrete and abstract semantics of Factorial using **octave**. Ranges: $n \in \{1, \dots, d\}$ and $m \in \{1, \dots, d!\}$.

d	$\dim(\mathbf{T}(F))$	$\dim(\mathbf{T}^{\#}(F))$
2	45	30
3	140	40
4	625	50
5	3630	60
6	25235	70
7	201640	80
8	1814445	90
9	18144050	100

Using uniform initial distributions $\mathbf{d_0}$ for n and m.

Scalablity

The abstract probabilities for m being **even** or **odd** when we execute the abstract program for various d values are:

d	**even**	**odd**
10	0.81818	0.18182
100	0.98019	0.019802
1000	0.99800	0.0019980
10000	0.99980	0.00019998

Ortholattice of Projection Operators [Not for Exam]

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

Alternatively, order projections by inclusion of their image spaces, i.e. $\mathbf{E} \sqsubseteq \mathbf{F}$ iff $Y_\mathbf{E} \subseteq Y_\mathbf{F}$.

The orthogonal projections form a complete (ortho)lattice.

The range of the intersection $\mathbf{E} \sqcap \mathbf{F}$ is to the closure of the intersection of the image spaces of $\mathbf{E}$ and $\mathbf{F}$.

The union $\mathbf{E} \sqcup \mathbf{F}$ corresponds to the union of the images.

Computing Intersections/Unions [Not for Exam]

Associate to every Probabilistic Abstract Interpretation $(\mathbf{A}, \mathbf{G})$ a projection, similar to so-called "upper closure operators" (uco):

$$\mathbf{E} = \mathbf{AG} = \mathbf{AA}^{\dagger}.$$

A general way to construct $\mathbf{E} \sqcap \mathbf{F}$ and (by exploiting de Morgan's law) also $\mathbf{E} \sqcup \mathbf{F} = (\mathbf{E}^{\perp} \sqcap \mathbf{F}^{\perp})^{\perp}$ is via an infinite approximation sequence and has been suggested by Halmos:

$$\mathbf{E} \sqcap \mathbf{F} = \lim_{n \to \infty} (\mathbf{EFE})^{n}.$$

Commutative Case

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

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

Example

Consider a finite set Ω with a probability structure. For any (measurable) subset A of Ω define the characteristic function χ_A with $\chi_A(x) = 1$ if $x \in A$ and 0 otherwise. The characteristic functions are (commutative) projections on random variables using pointwise multiplication, i.e. $X\chi_A\chi_A = X\chi_A$. We have $\chi_{A\cap B} = \chi_A\chi_B$ and $\chi_{A\cup B} = \chi_A + \chi_B - \chi_A\chi_B$.

Non-Commutative Case [Not for Exam]

The Moore-Penrose pseudo-inverse is also useful for computing the $\mathbf{E} \sqcap \mathbf{F}$ and $\mathbf{E} \sqcup \mathbf{F}$ of general, non-commuting projections via the parallel sum

$$\mathbf{A} : \mathbf{B} = \mathbf{A}(\mathbf{A} + \mathbf{B})^{\dagger}\mathbf{B}$$

The intersection of projections is given by:

$$\mathbf{E} \sqcap \mathbf{F} = 2(\mathbf{E} : \mathbf{F}) = \mathbf{E}(\mathbf{E} + \mathbf{F})^{\dagger}\mathbf{F} + \mathbf{F}(\mathbf{E} + \mathbf{F})^{\dagger}\mathbf{E}$$

Israel, Greville: *Gereralized Inverses, Theory and Applications*, Springer 2003

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?

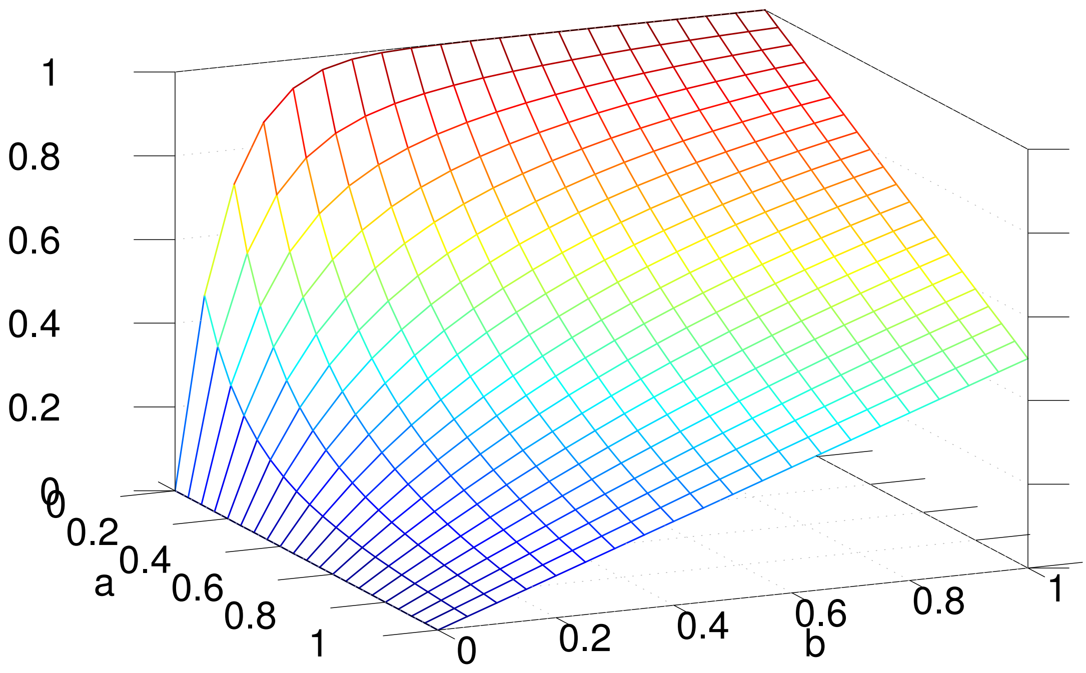
Introduced by McIver and Morgan (2005).
Discussed in detail by Gretz, Katoen, McIver (2012/14)

Example: Duelling Cowboys

```
begin
# who's first turn
choose 1:{t:=0} or 1:{t:=1} ro;
# continue until ...
c := 1;
while c == 1 do
if (t==0) then
  choose ak:{c:=0} or am:{t:=1} ro
else
  choose bk:{c:=0} or bm:{t:=0} ro
fi;
od;
stop; # terminal loop
end
```


Example: Duelling Cowboys [Not for Exam]

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



References

Alessandra Di Pierro, Chris Hankin, Herbert Wiklicky: *Probabilistic semantics and analysis*. LNCS 6154, Springer 2010.

Alessandra Di Pierro, Herbert Wiklicky: *Concurrent Constraint Programming: Towards Probabilistic Abstract Interpretation*. PPDP, ACM SIGPLAN 2000.

Adi Ben-Israel, Thomas N.E. Greville: *Generalized Inverses: Theory and Applications*. Springer 2003.

Friedrich Gretz, Joost-PieterKatoen, Annabelle McIver: *Operational versus weakest pre-expectation semantics for the probabilistic guarded command language*. Performance Evaluation, Vol. 73, 2014.

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