Moore-Penrose Pseudo-Inverse

**Definition**

Let $\mathcal{C}$ and $\mathcal{D}$ be two Hilbert spaces and $A : \mathcal{C} \to \mathcal{D}$ a bounded linear map. A bounded linear map $A^\dagger = G : \mathcal{D} \to \mathcal{C}$ is the **Moore-Penrose pseudo-inverse** of $A$ iff

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

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

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

On finite dimensional vector (Hilbert) spaces we have an inner product $\langle ., . \rangle$. This 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 operator $A$ is positive, i.e. $A \succeq 0$, if there exists an operator $B$ such that $A = B^*B$.
- An (orthogonal) projection is a self-adjoint $E$ with $EE = E$.

Projections identify (closed) sub-spaces $Y_E = \{ Ex \mid x \in \mathcal{V} \}$.

Example: Sign Domain

- $\mathbb{Z}$
- $\leq 0$
- $\geq 0$
- $0$
- $\emptyset$

Enumeration: $Sign = \{ \emptyset, 0, \geq 0, \leq 0, \mathbb{Z} \}$

Free Vector Space: $\mathcal{V}(Sign) = \{ \sum_{s \in Sign} x_s \cdot s \mid x_i \in \mathbb{R} \}$

Consider the upward closed sub-domains of \( \{\emptyset, 0, \geq 0, \leq 0, \mathbb{Z}\} \):

\[
\begin{align*}
\rho_1 &= \{\mathbb{Z}\} \\
\rho_2 &= \{\mathbb{Z}, \geq 0\} \\
\rho_3 &= \{\mathbb{Z}, 0\} \\
\rho_4 &= \{\mathbb{Z}, \emptyset\} \\
\rho_5 &= \{\mathbb{Z}, \leq 0\} \\
\rho_6 &= \{\mathbb{Z}, \geq 0, \emptyset\} \\
\rho_7 &= \{\mathbb{Z}, \geq 0, 0\} \\
\rho_8 &= \{\mathbb{Z}, 0, \emptyset\} \\
\rho_9 &= \{\mathbb{Z}, \leq 0, 0\} \\
\rho_{10} &= \{\mathbb{Z}, \leq 0, \emptyset\} \\
\rho_{11} &= \{\mathbb{Z}, \geq 0, 0, \emptyset\} \\
\rho_{12} &= \{\mathbb{Z}, \leq 0, \geq 0, 0, \emptyset\} \\
\rho_{13} &= \{\mathbb{Z}, \leq 0, 0, \emptyset\} \\
\rho_{14} &= \{\mathbb{Z}, \leq 0, \geq 0, 0, \emptyset\}
\end{align*}
\]

Identify abstract domains via upward closed operators (uco) \( \rho = \alpha \circ \gamma \) (vs downward closed operators (dco) \( \gamma \circ \alpha \)).
Example: Probabilistic Abstractions $R_n$

$$R_7 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad R_8 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_9 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad R_{10} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_{11} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad R_{12} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_{13} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad R_{14} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$
Computing Intersections/Unions

Associate to every PAI \((A, G)\) a projection (similar to 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. \(X_{\chi_A}X_B = X_{\chi_AB}\). We have

\[ \chi_{A \cap B} = \chi_A \chi_B \quad \text{and} \quad \chi_{A \cup B} = \chi_A + \chi_B - \chi_A \chi_B. \]
Non-Commutative Case

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

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

The intersection of projections is given by:

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

Israel, Greville: *Generalized Inverses, Theory and Applications*, Springer 03

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 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.
Ortholattices I

Non-distributive analogs of Boolean algebras.

**Definition (Ortholattice I)**

An ortholattice \((L, \sqsubseteq, \perp, 0, 1)\) is a lattice \((L, \sqsubseteq)\) with universal bounds 0 and 1, i.e.

1. \((L, \sqsubseteq)\) is a partial order (i.e. \(\sqsubseteq\) is reflexive, antisymmetric, and transitive),
2. all pairs of elements \(a, b \in L\) have a least upper bound (sup) denoted by \(a \sqcup b\), and a greatest lower bound (inf) denoted by \(a \sqcap b\),
3. \(0 \sqsubseteq a\) and \(a \sqsubseteq 1\) for all \(a \in L\).

... 

Ortholattices II

**Definition (Ortholattice II)**

... and a unary *complementation* operation \(a \mapsto a^\perp\) satisfying:

1. \(a \sqcap a^\perp = 0\) and \(a \sqcup a^\perp = 1\) for all \(a \in L\),
2. \((a \sqcap b)^\perp = a^\perp \sqcup b^\perp\) and \((a \sqcup b)^\perp = a^\perp \sqcap b^\perp\) for all \(a, b \in L\),
3. \((a^\perp)^\perp = a\) for all \(a \in L\).

The set \(P(\mathcal{H})\) of closed-range projections on a Hilbert space \(\mathcal{H}\) is a non-distributive ortholattice

\[
\left\langle P(\mathcal{H}), \sqsubseteq, \sqcup, \sqcap, \perp, 1, 0 \right\rangle
\]
Commutativity and Distributivity

In general, $\sqcap$ and $\sqcup$ in an ortholattice are not distributive, ie.

$$(a \sqcap b) \sqcup (a \sqcap c) \not\subseteq a \sqcap (b \sqcup c)$$

$$a \sqcup (b \sqcap c) \not\subseteq (a \sqcup b) \sqcap (a \sqcup c)$$

Two elements $a$ and $b$ in an ortholattice commute, denoted by $[a, b] = 0$, iff

$$a = (a \sqcap b) \sqcup (a \sqcap b^\perp)$$

An ortholattice is called an orthomodular lattice if $[a, b] = 0$ implies $[b, a] = 0$.

Example: Projections $P_n = R_n R_n^\dagger$

$$P_1 = \begin{pmatrix}
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1
\end{pmatrix},
\quad P_2 = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

$$P_3 = \begin{pmatrix}
1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0
\end{pmatrix},
\quad P_4 = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

$$P_5 = \begin{pmatrix}
1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0
\end{pmatrix},
\quad P_6 = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{pmatrix}$$
Example: Projections $P_n = R_n R_n^+$

$P_7 = \begin{pmatrix} 1 & 1/2 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1/2 & 0 & 1/2 \end{pmatrix}$, $P_8 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 3/3 & 3/3 & 3/3 \\ 0 & 0 & 3/3 & 3/3 & 3/3 \end{pmatrix}$

$P_9 = \begin{pmatrix} 1 & 1/2 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1/2 & 1/2 \end{pmatrix}$, $P_{10} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 3/3 & 3/3 & 3/3 & 3/3 \\ 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1/2 & 1/2 \end{pmatrix}$

$P_{11} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1/2 & 0 & 1/2 \end{pmatrix}$, $P_{12} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$

$P_{13} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1/2 & 1/2 \end{pmatrix}$, $P_{14} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$
Example: The Lattice $uco(Sign)$

Example: The Lattice $\mathcal{P}(\nu(Sign))$
Example: Combining Projections

\[
P_7 \cap P_8 = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix} \cap \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{pmatrix} = \\
= \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{pmatrix} = P_3
\]

In particular, we have \( P_7 \cap P_8 = P_7 P_8 \) as \( P_7 \) and \( P_8 \) commute, i.e. \([P_7, P_8] = P_7 P_8 - P_8 P_7 = 0\).

Example: Combining Projections

\[
P_4 \cap P_7 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix} \cap \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix} = \\
= \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{pmatrix} = P_1
\]

Using the expression \( P_4 \cap P_7 = 2P_4(P_4 + P_7)\dagger P_7 \) as \( P_4 \) and \( P_7 \) do not commute.
Example: Combining Projections

Note that the simple multiplication $P_4 P_7$ is different from $P_4 \square P_7$:

$$P_4 P_7 = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 1 \\
\end{pmatrix} \begin{pmatrix}
1 & 1 & 0 & 0 & 0 \\
\frac{1}{2} \frac{1}{2} 0 0 0 \\
0 & 1 & 2 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 \\
\end{pmatrix} = \begin{pmatrix}
1 & 1 & 0 & 0 & 0 \\
\frac{1}{2} \frac{1}{2} 0 0 0 \\
0 & 1 & 2 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 \\
\end{pmatrix} \neq P_4 \square P_7$$

Precision Measures

**Definition**

Given two vector (Hilbert) spaces $C$ and $D$ and a bounded linear map $F : C \to D$, then we say that a pair of projections $P : C \to C$ and $R : D \to D$ is complete for $F$ iff

$$FP = RFP.$$

Given a pair of projections $(P, R)$ for a function $F$, we estimate the precision of the abstraction via the “difference” between $FP$ and its optimal version $RFP$.

$$\text{Prec}_F(P, R) = \|FP - RFP\|.$$
Proposition

Let $F : \mathcal{H}_1 \mapsto \mathcal{H}_2$ be a bounded linear operator between two Hilbert spaces $\mathcal{H}_1$ and $\mathcal{H}_2$, and let $P_1, P_2 \in P(\mathcal{H}_2)$ and $R \in P(\mathcal{H}_1)$.

Then we have: if $P_1 \sqsubseteq P_2$ then $\text{Prec}_F(P_1, R) \leq \text{Prec}_F(P_2, R)$.

Example: (Relative) Precisions
The collecting semantics of a program $P$ is given by:

$$T(P) = \sum p_{ij} \cdot T(\ell_i, \ell_j)$$

Local effects $T(\ell_i, \ell_j)$: Data Update + Control Step

$$T(\ell_i, \ell_j) = (N_{i1} \otimes N_{i2} \otimes \ldots \otimes N_{iv}) \otimes M_{ij}$$

**Kronecker Products**

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

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

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

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

The (algebraic) tensor product of vector spaces \( V_1, V_2, \ldots, V_n \) is given by a vector space \( \bigotimes_{i=1}^n V_i \) and a map \( p = \otimes_{i=1}^n \in \mathcal{L}(V_1, V_2, \ldots, V_n; \bigotimes_{i=1}^n V_i) \) such that if \( W \) is any vector space and \( f \in \mathcal{L}(V_1, V_2, \ldots, V_n; W) \) then there exists a unique map \( h : \bigotimes_{i=1}^n V_i \rightarrow W \) satisfying \( f = h \circ p \).

\[
\begin{array}{c}
\mathcal{V}_1 \times \mathcal{V}_2 \times \ldots \times \mathcal{V}_n \xrightarrow{f} \mathcal{W} \\
\downarrow p \quad \quad \quad \quad \quad \quad \downarrow h \\
\mathcal{V}_1 \otimes \mathcal{V}_2 \otimes \ldots \otimes \mathcal{V}_n \end{array}
\]

\( \mathcal{V}(X \times Y) = \mathcal{V}(X) \otimes \mathcal{V}(Y) \)

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:

\[ (A_1 \otimes \ldots \otimes A_n) \cdot (B_1 \otimes \ldots \otimes B_n) = (A_1 \cdot B_1 \otimes \ldots \otimes A_n \cdot B_n) \]

\[ A_1 \otimes \ldots \otimes (\alpha A_i) \otimes \ldots \otimes A_n = \alpha (A_1 \otimes \ldots \otimes A_i \otimes \ldots \otimes A_n) \]

\[ A_1 \otimes \ldots \otimes (A_i + B_i) \otimes \ldots \otimes A_n = (A_1 \otimes \ldots \otimes A_i \otimes \ldots \otimes A_n) + (A_1 \otimes \ldots \otimes B_i \otimes \ldots \otimes A_n) \]

\[ (A_1 \otimes \ldots \otimes A_i \otimes \ldots \otimes A_n)^\dagger = A_1^\dagger \otimes \ldots \otimes A_i^\dagger \otimes \ldots \otimes A_n^\dagger \]
Relational Dependency

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

Input/output behaviour: Parity of \(m\) for different values of \(n\).

- Probability that \(m = \text{even/odd}\) and \(n = 1, 2, 3\).
  - Probability that \(m\) is even/odd, and
  - Probability that \(n\) is 1, 2, 3.
- Probability that \(m\) is even/odd for \(n = 1, 2, 3\).

Dependency and Correlations

Some joint probability distributions can be expressed as tensor product of two (independent) probability distributions \(e\) and \(f\):

\[
\begin{pmatrix}
\frac{2}{9} & \frac{2}{9} & \frac{2}{9} \\
\frac{1}{9} & \frac{1}{9} & \frac{1}{9}
\end{pmatrix} = \left( \frac{1}{3}, \frac{1}{3}, \frac{1}{3} \right) \otimes \left( \frac{2}{3}, \frac{1}{3} \right)^t
\]

But there are no two vectors \(e\) and \(f\) such that for example

\[
\begin{pmatrix}
0 & \frac{1}{3} & \frac{1}{3} \\
\frac{1}{3} & 0 & 0
\end{pmatrix} = e \otimes f
\]
Consider compositional (probabilistic) abstractions of the form:

\[
S = \bigoplus_{i=1}^{\nu} S(x_i) \quad \text{with} \quad S(x_i) = \left( \bigotimes_{k=1}^{i-1} S_{-i} \right) \otimes S_i \otimes \left( \bigotimes_{k=i+1}^{\nu} S_{-i} \right)
\]

**Fully Relational**: \(S_r\) is \(S\) with \(S_i = A_i\) and \(S_{-i} = A_{-i}\)

**Weakly Relational**: \(S_w\) is \(S\) with \(S_i = A_i\) and \(S_{-i} = A_{-i}\) or \(A_f\)

**Non-Relational**: \(S_n\) is \(S\) with \(S_i = A\) and \(S_{-i} = A_f\)

With \(A_f\) forgetful and \(A_i\) and \(A_{-i}\) nontrivial abstractions.

For \(S_r\) all factors in \(\bigoplus\) are the same; we can take \(S_r = S(x_1)\).

### Examples

```plaintext
var x:[0..10]; begin x:=k; stop (k = 1, 4)
```

<table>
<thead>
<tr>
<th></th>
<th>(\emptyset)</th>
<th>(S_n)</th>
<th>(S_w)</th>
<th>(S_r)</th>
<th>(id)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\emptyset)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(S_n)</td>
<td>1.58</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(S_w)</td>
<td>1.58</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(S_r)</td>
<td>1.58</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(id)</td>
<td>2.55</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

```plaintext
var x:[0..10]; y:[0..10]; begin x:=y; stop
```

<table>
<thead>
<tr>
<th></th>
<th>(\emptyset)</th>
<th>(S_n)</th>
<th>(S_w)</th>
<th>(S_r)</th>
<th>(id)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\emptyset)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(S_n)</td>
<td>1.73</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(S_w)</td>
<td>2.24</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(S_r)</td>
<td>2.24</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(\emptyset)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

```plaintext
var x:[0..10]; y:[0..3]; begin x:=2*y; stop
```

<table>
<thead>
<tr>
<th></th>
<th>(\emptyset)</th>
<th>(S_n)</th>
<th>(S_w)</th>
<th>(S_r)</th>
<th>(id)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\emptyset)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(S_n)</td>
<td>2.41</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(S_w)</td>
<td>2.90</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(S_r)</td>
<td>2.90</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Conclusions

Some applications of PAI:

- **Approximate Process Equivalences**: The semantics of concurrent processes can be defined via \( \varepsilon \)-bisimulation.
- **Approximate Confinement**: Static analysis of security properties can be sometimes more effective if the security is guaranteed only up to some acceptable percentage threshold.
- **Probabilistic Program Transformation**: Transforming out timing leaks... probabilistically.
- ...

Further Work
LOS for Variable Probabilities

In every choice construct one must make a choice and the probabilities of all choices must sum up to one (certainty). One can’t assume (that the programmer used) normalised probabilities.

We therefore need to normalise probabilities with respect to a context of "competing" probabilities:

\[ \tilde{p} = p[p_1...p_n] = \frac{p}{p_1 + \ldots + p_n}. \]

This can be done at compile-time if all probabilities are constants, but also at runtime in the operational semantics.

Typically one would assume \( p_i \in \mathbb{R} \) or \( p_i \in \mathbb{Q} \). However, we can also use discrete probabilities, i.e. \( p_i \in \mathbb{Z} \).

Duel at High Noon

Consider a "duel" between two cowboys:

- Cowboy A – hitting probability \( a \)
- Cowboy B – hitting probability \( b \)

Choose (non-deterministically) whether A or B starts.

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)
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

The survival chances, i.e. winning probability, for A.
For all possible values of the variable probabilities $p_i$ compute their normalisation, compute the possible contexts.

\[
C[p_1, \ldots, p_n] = \begin{cases} 
\emptyset & \text{if } n = 0 \\
\{[p_1]\} & \text{if } n = 1 \text{ and } p_i \text{ const} \\
\{[c] \mid c \in \text{Value}(p_1)\} & \text{if } n = 1 \text{ and } p_i \text{ var} \\
\bigcup_{[i] \in C[p_1]} \{[i] \cdot C[p_2, \ldots, p_n]\} & \text{otherwise, i.e. } n > 1.
\end{cases}
\]

**Example**

Variable $x$ with $\text{Value}(x) = \{0, 1\}$ and a parameter $p = 0$ or $p = 1$ then contexts are given by:

\[
C[x, 1, p] = \{[0, 1, 0], [1, 1, 0]\} \text{ and } C[x, 1, p] = \{[0, 1, 1], [1, 1, 1]\}
\]

**Dynamic Probabilities**

For all possible values of the variable probabilities test if the current state. With $c_j \in \text{Value}(p_j)$ and $d_i \in \text{Value}(p_i)$ use:

\[
P_{c_j[d_1\ldots d_n]}^{p_i[p_1\ldots p_n]} = P(p_i = c_j) \cdot \left( \prod_{k=1,\ldots,n} P(p_k = d_k) \right)
\]

This gives the LOS Semantics for variable probabilities:

\[
\left\{ \begin{array}{lcl} 
\text{choose}^{P_i:S_1} \ldots \text{or } p_n : S_n \text{ or } \ell 
\end{array} \right\}_{\text{LOS}} = \left\{ S_i \right\}_{\text{LOS}} \cup \\
\bigcup_{i=1}^n \left\{ \sum_{c_j \in \text{Value}(p_i)} \sum_{[d_1\ldots d_n] \in C[p_1\ldots p_n]} c_j[d_1\ldots d_n] \cdot P_{c_j[d_1\ldots d_n]}^{p_i[p_1\ldots p_n]} \otimes E(\ell, \text{init}(S_i)) \right\}
\]
Learning how to shoot straight

begin
  # initialise skills of A
  akl := ak; aml := am;
  # who’s first
  choose 1:{t:=0} or 1:{t:=1} ro;
  # continue until ...
  c := 1;
  while c == 1 do
    if (t==0) then
      choose akl:{c:=0} or aml:{t:=1} ro
    else
      choose bk:{c:=0} or bm:{t:=0} ro
    fi;
    akl := @inc(akl); aml := @dec(aml);
  od;
  stop; # terminal loop
end

Back to the two Cowboys

Learning rate 0.
Learning rate 1.

Back to the two Cowboys

Learning rate 2.
Back to the two Cowboys

Learning rate 4.

 LOS for Program Synthesis

Finding the minimum length path vs minimum value of functions

As usual (for now): Take the best non-linear optimisation tool money can’t buy (leave it to "them" to make it work).
A General Approach

- Consider parameterised program $P(p_1, p_2, \ldots, p_n)$ with
  \[ \ldots \text{[choose]}^\ell p_1 : S_1 \text{ or } \ldots \text{ or } p_n : S_n \text{ ro[choose]}^\ell \lambda_1 : S_1 \text{ or } \ldots \text{ or } \lambda_n : S_n \text{ ro} \]

- Construct the parametric LOS semantics/operator, i.e.
  \[ \mathcal{P}(\lambda_1, \lambda_2, \ldots, \lambda_n) = T(\lambda_1, \lambda_2, \ldots, \lambda_n) \]

- Establish constraints on functional behaviour, e.g.
  \[ A^\dagger T(\lambda_1, \lambda_2, \ldots, \lambda_n) A = [S] \]

- Additional non-functional (performance) objectives
  \[ \min_{\lambda_1, \lambda_2, \ldots, \lambda_n} \Phi(T(\lambda_1, \lambda_2, \ldots, \lambda_n)) \]

Swapping: The XOR Trick

Consider the (probabilistic) sketch for swapping $x$ and $y$:

- [choose]$^1 \lambda_{1,1} : S_1$ or ... or $\lambda_{1,n} : S_n$ ro;
- [choose]$^2 \lambda_{2,1} : S_1$ or ... or $\lambda_{2,n} : S_n$ ro;
- [choose]$^3 \lambda_{3,1} : S_1$ or ... or $\lambda_{3,n} : S_n$ ro;

with $S_i$ one of $i = 1, \ldots, 13$ different elementary blocks:

- [skip]$^1$
- $[x := y]^2$
- $[x := z]^3$
- $[y := x]^4$
- $[y := z]^5$
- $[z := x]^6$
- $[z := y]^7$
- $[x := (x + y) \mod 2]^8$
- $[x := (x + z) \mod 2]^9$
- $[y := (y + x) \mod 2]^10$
- $[y := (y + z) \mod 2]^11$
- $[z := (z + x) \mod 2]^12$
- $[z := (z + y) \mod 2]^13$
Swapping: Parameterised LOS and Objective

Using 13 transfer functions $F_1 \ldots F_{13}$ to define

$$T(\lambda_{ij}) = \prod_{i=1}^{3} T_i(\lambda_{ij}) \quad \text{with} \quad T_i(\lambda_{ij}) = \sum_{j=1}^{13} \lambda_{ij} F_j$$

For one-bit variables $x, y$ the intended behaviour (on $\mathbb{R}^2 \otimes \mathbb{R}^2$):

$$S = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{align*}
x & \mapsto 0 & y & \mapsto 0 \\
x & \mapsto 0 & y & \mapsto 1 \\
x & \mapsto 1 & y & \mapsto 0 \\
x & \mapsto 1 & y & \mapsto 1
\end{align*}$$

Objective: $\min \Phi_{00}(\lambda_{ij}) = \|A^\dagger T(\lambda_{ij})A - S\|_2$ or $\min \Phi_{\rho\omega}(\lambda_{ij})$

which also penalises for reading or writing to $z$; using the abstraction $A = I(4) \otimes A_f(2) = \text{diag}(1,1,1,1) \otimes (1,1)^t$.

Swapping: Test Runs

Using $\text{octave}$: if we start with a swap which uses $z$, like

$$[z := x]^6; [x := y]^2; [y := z]^5$$

represented by $\lambda_{ij}$ given as:

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

For $\min \Phi_{00}$ we get no change; but with $\min \Phi_{11}$ (after 12 iterations) we get with $\text{octave}$ the optimal $\lambda_{ij}$'s:

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

This corresponds to the program:

$$[y := (y+x) \mod 2]^10; [x := (x+y) \mod 2]^8; [y := (y+x) \mod 2]^10$$
Swapping: Test Runs

For randomly chosen initial values for $\lambda_{ij}$:

$$\begin{pmatrix}
.70 & .30 & .72 & .84 & .51 & .70 & .76 & .47 & .63 & .93 & .55 & .68 \\
.74 & .22 & .37 & .70 & .67 & .13 & .93 & .69 & .30 & .88 & .03 & .52 & .80 \\
.59 & .49 & .01 & .69 & .22 & .23 & .10 & .01 & .10 & .22 & .03 & .55 & .11
\end{pmatrix}$$

For $\min \Phi_{11}$ (after 9 iterations) we get the optimal $\lambda_{ij}$'s:

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

This corresponds to the program:

$$[y := (y + x) \mod 2]^10; [x := (x + y) \mod 2]^8; [y := (y + x) \mod 2]^10$$

For $\Phi_{00}$ we may also get: $[z := x]^6; [x := y]^2; [y := z]^5$.

Some References

- Di Pierro, Wiklicky: A logico-algebraic approach to probabilistic program analysis Pre-Proceedings of LOPSTR'05.