Probabilistic Program Analysis

Logic and Analysis

Alessandra Di Pierro
University of Verona, Italy
alessandra.dipierro@univr.it

Herbert Wiklicky
Imperial College London, UK
herbert@doc.ic.ac.uk

Moore-Penrose Pseudo-Inverse

Definition

Let $\mathcal{C}$ and $\mathcal{D}$ be two Hilbert spaces and $\mathbf{A} : \mathcal{C} \to \mathcal{D}$ a bounded linear map. A bounded 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 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 operator $\mathbf{A}$ is positive, i.e. $\mathbf{A} \sqsupseteq 0$, if there exists an operator $\mathbf{B}$ such that $\mathbf{A} = \mathbf{B}^*\mathbf{B}$.
- An (orthogonal) projection is a self-adjoint $\mathbf{E}$ with $\mathbf{EE} = \mathbf{E}$.

Projections identify (closed) sub-spaces $Y_{\mathbf{E}} = \{\mathbf{E}x \mid x \in \mathcal{V}\}$.

(Orthogonal) Projections – Idempotents

On finite dimensional vector (Hilbert) spaces we have an inner product $\langle ., . \rangle$. This allows us to define an adjoint via:

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

- An operator $\mathbf{A}$ is self-adjoint if $\mathbf{A} = \mathbf{A}^*$.
- An operator $\mathbf{A}$ is positive, i.e. $\mathbf{A} \sqsupseteq 0$, if there exists an operator $\mathbf{B}$ such that $\mathbf{A} = \mathbf{B}^*\mathbf{B}$.
- An (orthogonal) projection is a self-adjoint $\mathbf{E}$ with $\mathbf{EE} = \mathbf{E}$.

Projections identify (closed) sub-spaces $Y_{\mathbf{E}} = \{\mathbf{E}x \mid x \in \mathcal{V}\}$.

(Orthogonal) Projections – Idempotents

On finite dimensional vector (Hilbert) spaces we have an inner product $\langle ., . \rangle$. This allows us to define an adjoint via:

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

- An operator **A** is self-adjoint if $\mathbf{A} = \mathbf{A}^*$.
- An operator **A** is positive, i.e. $\mathbf{A} \sqsupseteq 0$, if there exists an operator **B** such that $\mathbf{A} = \mathbf{B}^*\mathbf{B}$.
- An (orthogonal) projection is a self-adjoint **E** with $\mathbf{EE} = \mathbf{E}$.

Projections identify (closed) sub-spaces $Y_\mathbf{E} = \{\mathbf{E}x \mid x \in \mathcal{V}\}$.

(Orthogonal) Projections – Idempotents

On finite dimensional vector (Hilbert) spaces we have an inner product $\langle ., . \rangle$. This allows us to define an adjoint via:

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

- An operator $\mathbf{A}$ is self-adjoint if $\mathbf{A} = \mathbf{A}^*$.
- An operator $\mathbf{A}$ is positive, i.e. $\mathbf{A} \sqsupseteq 0$, if there exists an operator $\mathbf{B}$ such that $\mathbf{A} = \mathbf{B}^*\mathbf{B}$.
- An (orthogonal) projection is a self-adjoint $\mathbf{E}$ with $\mathbf{EE} = \mathbf{E}$.

Projections identify (closed) sub-spaces $Y_\mathbf{E} = \{\mathbf{E}x \mid x \in \mathcal{V}\}$.

(Orthogonal) Projections – Idempotents

On finite dimensional vector (Hilbert) spaces we have an inner product $\langle ., . \rangle$. This allows us to define an adjoint via:

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

- An operator $\mathbf{A}$ is self-adjoint if $\mathbf{A} = \mathbf{A}^*$.
- An operator $\mathbf{A}$ is positive, i.e. $\mathbf{A} \sqsupseteq 0$, if there exists an operator $\mathbf{B}$ such that $\mathbf{A} = \mathbf{B}^*\mathbf{B}$.
- An (orthogonal) projection is a self-adjoint $\mathbf{E}$ with $\mathbf{EE} = \mathbf{E}$.

Projections identify (closed) sub-spaces $Y_{\mathbf{E}} = \{\mathbf{E}x \mid x \in \mathcal{V}\}$.

Example: Sign Domain

$$\begin{tikzcd} & \bullet\ \mathbb{Z} \arrow[dl] \arrow[dr] & \\ \leq 0\ \bullet \arrow[dr] & & \bullet\ \geq 0 \arrow[dl] \\ & \bullet\ 0 \arrow[d] & \\ & \bullet\ \emptyset & \end{tikzcd}$$

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

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

Francesca Scozzari: *Domain theory in abstract interpretation: equations, completeness and logic*. PhD Thesis, Siena 1999.

Example: Sign Domain

$$\begin{tikzcd}
& \bullet\ \mathbb{Z} \arrow[dl] \arrow[dr] & \\
\leq 0\ \bullet \arrow[dr] & & \bullet\ \geq 0 \arrow[dl] \\
& \bullet\ 0 \arrow[d] & \\
& \bullet\ \emptyset &
\end{tikzcd}$$

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

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

Francesca Scozzari: *Domain theory in abstract interpretation: equations, completeness and logic.* PhD Thesis, Siena 1999.

Example: Sign Domain

$$\begin{tikzcd}
 & \bullet\ \mathbb{Z} \arrow[dl] \arrow[dr] & \\
\leq 0\ \bullet \arrow[dr] & & \bullet\ \geq 0 \arrow[dl] \\
 & \bullet\ 0 \arrow[d] & \\
 & \bullet\ \emptyset &
\end{tikzcd}$$

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

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

Francesca Scozzari: *Domain theory in abstract interpretation: equations, completeness and logic*. PhD Thesis, Siena 1999.

Example: Sign Domain

$$\begin{tikzcd}[column sep=small]
 & \bullet\ \mathbb{Z} \arrow[dl] \arrow[dr] & \\
\leq 0\ \bullet \arrow[dr] & & \bullet\ \geq 0 \arrow[dl] \\
 & \bullet\ 0 \arrow[d] & \\
 & \bullet\ \emptyset &
\end{tikzcd}$$

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

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

Francesca Scozzari: *Domain theory in abstract interpretation: equations, completeness and logic*. PhD Thesis, Siena 1999.

Example: Classical Abstractions (Domains via uco)

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

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

Identify abstract domains via upward closed operators (ucu) $\rho = \alpha \circ \gamma$ (vs downward closed operators (dco) $\gamma \circ \alpha$).

Example: Probabilistic Abstractions $\mathbf{R}_n$

$$\mathbf{R}_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \mathbf{R}_2 = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{R}_3 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \mathbf{R}_4 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{R}_5 = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \mathbf{R}_6 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Example: Probabilistic Abstractions $\mathbf{R}_n$

$$\mathbf{R}_7 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \mathbf{R}_8 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{R}_9 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \mathbf{R}_{10} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Example: Probabilistic Abstractions $\mathbf{R}_n$

$$\mathbf{R}_{11} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \mathbf{R}_{12} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{R}_{13} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \mathbf{R}_{14} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Computing Intersections/Unions

Associate to every PAI $(\mathbf{A}, \mathbf{G})$ a projection (similar to 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:

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

Computing Intersections/Unions

Associate to every PAI $(\mathbf{A}, \mathbf{G})$ a projection (similar to 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 $\chi_{A\cup B} = \chi_A + \chi_B - \chi_A\chi_B$.

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 $\chi_{A\cup B} = \chi_A + \chi_B - \chi_A\chi_B$.

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

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 03

Projection Operators

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 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.

Projection Operators

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 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.

Projection Operators

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 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.

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 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 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 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

$$\left\langle P(\mathcal{H}), \sqsubseteq, \sqcup, \sqcap, .^\perp, \mathbf{I}, \mathbf{0} \right\rangle$$

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

$$\left\langle P(\mathcal{H}), \sqsubseteq, \sqcup, \sqcap, .^\perp, \mathbf{I}, \mathbf{0} \right\rangle$$

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

$$\left\langle P(\mathcal{H}), \sqsubseteq, \sqcup, \sqcap, .^\perp, \mathbf{I}, \mathbf{0} \right\rangle$$

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

$$\left\langle P(\mathcal{H}), \sqsubseteq, \sqcup, \sqcap, .^\perp, \mathbf{I}, \mathbf{0} \right\rangle$$

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

$$\left\langle P(\mathcal{H}), \sqsubseteq, \sqcup, \sqcap, .^\perp, \mathbf{I}, \mathbf{0} \right\rangle$$

Commutativity and Distributivity

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

$$(a \sqcap b) \sqcup (a \sqcap c) \sqsubseteq a \sqcap (b \sqcup c)$$

$$a \sqcup (b \sqcap c) \sqsubseteq (a \sqcup b) \sqcap (a \sqcup c)$$

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 $[b, a] = 0$.

Commutativity and Distributivity

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

$$(a \sqcap b) \sqcup (a \sqcap c) \sqsubseteq a \sqcap (b \sqcup c)$$

$$a \sqcup (b \sqcap c) \sqsubseteq (a \sqcup b) \sqcap (a \sqcup c)$$

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 $[b, a] = 0$.

Commutativity and Distributivity

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

$$(a \sqcap b) \sqcup (a \sqcap c) \sqsubseteq a \sqcap (b \sqcup c)$$

$$a \sqcup (b \sqcap c) \sqsubseteq (a \sqcup b) \sqcap (a \sqcup c)$$

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 $\mathbf{P}_n = \mathbf{R}_n \mathbf{R}_n^\dagger$

$$\mathbf{P}_1 = \begin{pmatrix} \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \end{pmatrix}, \quad \mathbf{P}_2 = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix}$$

$$\mathbf{P}_3 = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{pmatrix}, \quad \mathbf{P}_4 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}$$

$$\mathbf{P}_5 = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix}, \quad \mathbf{P}_6 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix}$$

Example: Projections $\mathbf{P}_n = \mathbf{R}_n \mathbf{R}_n^\dagger$

$$\mathbf{P}_7 = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix}, \ \mathbf{P}_8 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{pmatrix}$$

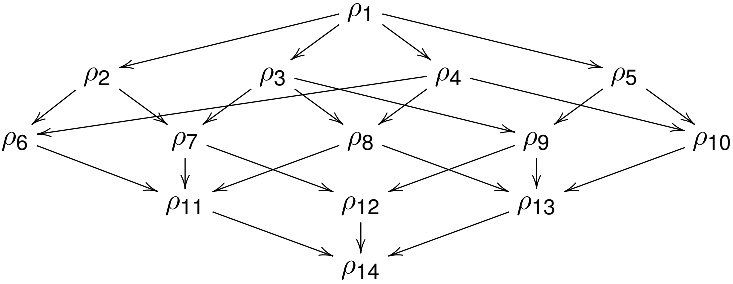
$$\mathbf{P}_9 = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix}, \ \mathbf{P}_{10} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

Example: Projections $\mathbf{P}_n = \mathbf{R}_n \mathbf{R}_n^\dagger$

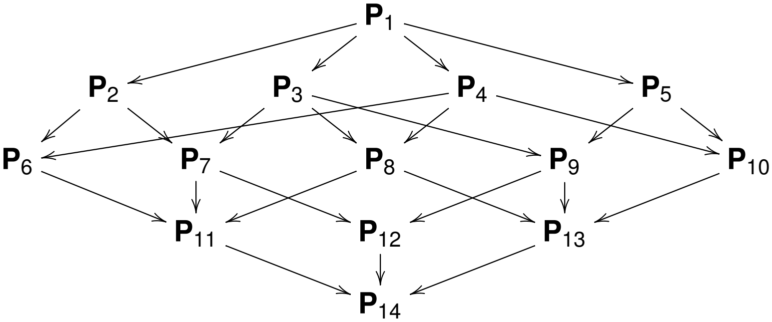
$$\mathbf{P}_{11} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix}, \mathbf{P}_{12} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{P}_{13} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix}, \mathbf{P}_{14} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Example: The Lattice *uco*(*Sign*)



Example: The Lattice $\mathcal{P}(\mathcal{V}(Sign))$



Example: Combining Projections

$$
\begin{aligned}
\mathbf{P}_7 \sqcap \mathbf{P}_8 &= \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix} \sqcap \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{pmatrix} = \\
&= \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{pmatrix} = \mathbf{P}_3
\end{aligned}
$$

In particular, we have $\mathbf{P}_7 \sqcap \mathbf{P}_8 = \mathbf{P}_7\mathbf{P}_8$ as $\mathbf{P}_7$ and $\mathbf{P}_8$ commute, i.e. $[\mathbf{P}_7, \mathbf{P}_8] = \mathbf{P}_7\mathbf{P}_8 - \mathbf{P}_8\mathbf{P}_7 = \mathbf{O}$.

Example: Combining Projections

$$
\begin{aligned}
\mathbf{P}_4 \sqcap \mathbf{P}_7 &= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix} \sqcap \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix} = \\
&= \begin{pmatrix} \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \end{pmatrix} = \mathbf{P}_1
\end{aligned}
$$

Using the expression $\mathbf{P}_4 \sqcap \mathbf{P}_7 = 2\mathbf{P}_4(\mathbf{P}_4 + \mathbf{P}_7)^\dagger \mathbf{P}_7$ as $\mathbf{P}_4$ and $\mathbf{P}_7$ do not commute.

Example: Combining Projections

Note that the simple multiplication $\mathbf{P}_4\mathbf{P}_7$ is different from $\mathbf{P}_4 \sqcap \mathbf{P}_7$:

$$
\begin{aligned}
\mathbf{P}_4\mathbf{P}_7 &= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix} \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix} = \\
&= \begin{pmatrix} \frac{1}{5} & \frac{1}{5} & 0 & 0 & 0 \\ \frac{1}{8} & \frac{1}{8} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{8} & \frac{1}{8} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{8} & \frac{1}{8} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{8} & \frac{1}{8} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix} \neq \mathbf{P}_4 \sqcap \mathbf{P}_7
\end{aligned}
$$

Precision Measures

Definition

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

$$\mathbf{FP} = \mathbf{RFP}.$$

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

$$Prec_{\mathbf{F}}(\mathbf{P}, \mathbf{R}) = \|\mathbf{FP} - \mathbf{RFP}\|.$$

Precision Measures

Definition

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

$$\mathbf{FP} = \mathbf{RFP}.$$

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

$$Prec_{\mathbf{F}}(\mathbf{P}, \mathbf{R}) = \|\mathbf{FP} - \mathbf{RFP}\|.$$

Order and Precision

Proposition

Let $\mathbf{F} : \mathcal{H}_1 \mapsto \mathcal{H}_2$ *be a bounded linear operator between two Hilbert spaces* $\mathcal{H}_1$ *and* $\mathcal{H}_2$*, and let* $\mathbf{P}_1, \mathbf{P}_2 \in P(\mathcal{H}_2)$ *and* $\mathbf{R} \in P(\mathcal{H}_1)$*.*
Then we have: if $\mathbf{P}_1 \sqsubseteq \mathbf{P}_2$ *then* $Prec_{\mathbf{F}}(\mathbf{P}_1, \mathbf{R}) \leq Prec_{\mathbf{F}}(\mathbf{P}_2, \mathbf{R})$*.*

Example: (Relative) Precisions

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}
P_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P_3	1	.75	0	.79	.75	.65	0	0	0	.65	0	0	0	0
P_4	1	.91	.79	0	.91	0	.79	0	.79	0	0	.79	0	0
P_5	1	.75	0	.79	.75	.65	0	0	0	.65	0	0	0	0
P_6	1.10	1	.87	0	1	0	.87	0	.87	0	0	.87	0	0
P_7	1.34	1	0	1.06	1	.87	0	0	0	.87	0	0	0	0
P_8	1	1	1	1	1	.82	1	0	1	.82	0	1	0	0
P_9	1.10	.82	0	.87	.82	.71	0	0	0	.71	0	0	0	0
P_{10}	1.07	.91	.87	.87	.91	.71	.87	0	.87	.71	0	.87	0	0
P_{11}	1.34	1	1	1.22	1	1	1	0	1	1	0	1	0	0
P_{12}	1.34	1	0	1.06	1	.87	0	0	0	.87	0	0	0	0
P_{13}	1.10	1	1	1.06	1	.87	1	0	1	.87	0	1	0	0
P_{14}	1.34	1	1	1.22	1	1	1	0	1	1	0	1	0	0

Linear Operator Semantics (LOS)

The collecting semantics of a program P is given by:

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

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

$$\mathbf{T}(\ell_i, \ell_j) = (\mathbf{N}_{i1} \otimes \mathbf{N}_{i2} \otimes \ldots \otimes \mathbf{N}_{iv}) \otimes \mathbf{M}_{ij}$$

Linear Operator Semantics (LOS)

The collecting semantics of a program P is given by:

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

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

$$\mathbf{T}(\ell_i, \ell_j) = (\mathbf{N}_{i1} \otimes \mathbf{N}_{i2} \otimes \ldots \otimes \mathbf{N}_{iv}) \otimes \mathbf{M}_{ij}$$

Kronecker Products

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

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

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

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

Kronecker Products

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

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

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

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

Abstract Tensor Product

The (algebraic) tensor product of vector spaces $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_n$ is given by a vector space $\bigotimes_{i=1}^n \mathcal{V}_i$ and a map $p = \otimes_{i=1}^n \in \mathcal{L}(\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_n; \bigotimes_{i=1}^n \mathcal{V}_i)$ such that if $\mathcal{W}$ is any vector space and $f \in \mathcal{L}(\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_n; \mathcal{W})$ then there exists a unique map $h : \bigotimes_{i=1}^n \mathcal{V}_i \to \mathcal{W}$ satisfying $f = h \circ p$.

$$\begin{tikzcd}
\mathcal{V}_1 \times \mathcal{V}_2 \times \ldots \times \mathcal{V}_n \arrow[r, "f"] \arrow[d, "p"'] & \mathcal{W} \\
\mathcal{V}_1 \otimes \mathcal{V}_2 \otimes \ldots \otimes \mathcal{V}_n \arrow[ur, "h"'] &
\end{tikzcd}$$

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

Abstract Tensor Product

The (algebraic) tensor product of vector spaces $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_n$ is given by a vector space $\bigotimes_{i=1}^n \mathcal{V}_i$ and a map $p = \otimes_{i=1}^n \in \mathcal{L}(\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_n; \bigotimes_{i=1}^n \mathcal{V}_i)$ such that if $\mathcal{W}$ is any vector space and $f \in \mathcal{L}(\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_n; \mathcal{W})$ then there exists a unique map $h : \bigotimes_{i=1}^n \mathcal{V}_i \to \mathcal{W}$ satisfying $f = h \circ p$.

$$\begin{tikzcd}
\mathcal{V}_1 \times \mathcal{V}_2 \times \ldots \times \mathcal{V}_n \arrow[r, "f"] \arrow[d, "p"'] & \mathcal{W} \\
\mathcal{V}_1 \otimes \mathcal{V}_2 \otimes \ldots \otimes \mathcal{V}_n \arrow[ur, "h"'] &
\end{tikzcd}$$

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

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

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

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

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

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

Relational Dependency

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

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

- Probability that $m =$ 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 $n = 1, 2, 3$.

Relational Dependency

1: $[m \leftarrow 1]^1$;
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: $[\textbf{stop}]^5$

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

- Probability that $m =$ 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 $n = 1, 2, 3$.

Relational Dependency

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

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

- Probability that $m =$ 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 $n = 1, 2, 3$.

Relational Dependency

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

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

- Probability that $m =$ 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**:

$$\begin{pmatrix} \frac{2}{9} & \frac{2}{9} & \frac{2}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}) \otimes (\frac{2}{3}, \frac{1}{3})^t$$

However, in general we can express any joint probability distribution as a linear combination of distributions.

$$\begin{pmatrix} 0 & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & 0 & 0 \end{pmatrix} = \frac{1}{3}(\mathbf{e}_1 \otimes \mathbf{f}_2) + \frac{1}{3}(\mathbf{e}_2 \otimes \mathbf{f}_1) + \frac{1}{3}(\mathbf{e}_3 \otimes \mathbf{f}_1)$$

with $\mathbf{e}_i \in \mathbb{R}^3$ and $\mathbf{f}_j \in \mathbb{R}^2$ (row and column) basis vectors

Dependency and Correlations

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

$$\begin{pmatrix} \frac{2}{9} & \frac{2}{9} & \frac{2}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}) \otimes (\frac{2}{3}, \frac{1}{3})^t$$

However, in general we can express any joint probability distribution as a linear combination of distributions.

$$\begin{pmatrix} 0 & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & 0 & 0 \end{pmatrix} = \frac{1}{3}(\mathbf{e}_1 \otimes \mathbf{f}_2) + \frac{1}{3}(\mathbf{e}_2 \otimes \mathbf{f}_1) + \frac{1}{3}(\mathbf{e}_3 \otimes \mathbf{f}_1)$$

with $\mathbf{e}_i \in \mathbb{R}^3$ and $\mathbf{f}_j \in \mathbb{R}^2$ (row and column) basis vectors

Dependency and Correlations

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

$$\begin{pmatrix} \frac{2}{9} & \frac{2}{9} & \frac{2}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}) \otimes (\frac{2}{3}, \frac{1}{3})^t$$

But there are no two vectors **e** and **f** such that for example

$$\begin{pmatrix} 0 & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & 0 & 0 \end{pmatrix} = \mathbf{e} \otimes \mathbf{f}$$

Dependency and Correlations

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

$$\begin{pmatrix} \frac{2}{9} & \frac{2}{9} & \frac{2}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}) \otimes (\frac{2}{3}, \frac{1}{3})^t$$

However, in general we can express any joint probability distribution as a linear combination of distributions.

$$\begin{pmatrix} 0 & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & 0 & 0 \end{pmatrix} = \frac{1}{3}(\mathbf{e}_1 \otimes \mathbf{f}_2) + \frac{1}{3}(\mathbf{e}_2 \otimes \mathbf{f}_1) + \frac{1}{3}(\mathbf{e}_3 \otimes \mathbf{f}_1)$$

with $\mathbf{e}_i \in \mathbb{R}^3$ and $\mathbf{f}_j \in \mathbb{R}^2$ (row and column) basis vectors

Fully, Weakly and Non-Relational Analysis

Consider compositional (probabilistic) abstractions of the form:

$$\mathbf{S} = \bigoplus_{i=1}^{v} \mathbf{S}(x_i) \quad \text{with} \quad \mathbf{S}(x_i) = (\bigotimes_{k=1}^{i-1} \mathbf{S}_{\neg i}) \otimes \mathbf{S}_i \otimes (\bigotimes_{k=i+1}^{v} \mathbf{S}_{\neg i})$$

Fully Relational: $\mathbf{S}_r$ is $\mathbf{S}$ with $\mathbf{S}_i = \mathbf{A}_i$ and $\mathbf{S}_{\neg i} = \mathbf{A}_{\neg i}$

Weakly Relational: $\mathbf{S}_w$ is $\mathbf{S}$ with $\mathbf{S}_i = \mathbf{A}_i$ and $\mathbf{S}_{\neg i} = \mathbf{A}_{\neg i}$ or $\mathbf{A}_f$

Non-Relational: $\mathbf{S}_n$ is $\mathbf{S}$ with $\mathbf{S}_i = \mathbf{A}$ and $\mathbf{S}_{\neg i} = \mathbf{A}_f$

With $\mathbf{A}_f$ forgetful and $\mathbf{A}_i$ and $\mathbf{A}_{\neg i}$ nontrivial abstractions.
For $\mathbf{S}_r$ all factors in $\bigoplus$ are the same; we can take $\mathbf{S}_r = \mathbf{S}(x_1)$.

Fully, Weakly and Non-Relational Analysis

Consider compositional (probabilistic) abstractions of the form:

$$\mathbf{S} = \bigoplus_{i=1}^{v} \mathbf{S}(x_i) \quad \text{with} \quad \mathbf{S}(x_i) = (\bigotimes_{k=1}^{i-1} \mathbf{S}_{\neg i}) \otimes \mathbf{S}_i \otimes (\bigotimes_{k=i+1}^{v} \mathbf{S}_{\neg i})$$

Fully Relational: $\mathbf{S}_r$ is $\mathbf{S}$ with $\mathbf{S}_i = \mathbf{A}_i$ and $\mathbf{S}_{\neg i} = \mathbf{A}_{\neg i}$

Weakly Relational: $\mathbf{S}_w$ is $\mathbf{S}$ with $\mathbf{S}_i = \mathbf{A}_i$ and $\mathbf{S}_{\neg i} = \mathbf{A}_{\neg i}$ or $\mathbf{A}_f$

Non-Relational: $\mathbf{S}_n$ is $\mathbf{S}$ with $\mathbf{S}_i = \mathbf{A}$ and $\mathbf{S}_{\neg i} = \mathbf{A}_f$

With $\mathbf{A}_f$ forgetful and $\mathbf{A}_i$ and $\mathbf{A}_{\neg i}$ nontrivial abstractions.
For $\mathbf{S}_r$ all factors in $\bigoplus$ are the same; we can take $\mathbf{S}_r = \mathbf{S}(x_1)$.

Fully, Weakly and Non-Relational Analysis

Consider compositional (probabilistic) abstractions of the form:

$$\mathbf{S} = \bigoplus_{i=1}^{v} \mathbf{S}(x_i) \quad \text{with} \quad \mathbf{S}(x_i) = (\bigotimes_{k=1}^{i-1} \mathbf{S}_{\neg i}) \otimes \mathbf{S}_i \otimes (\bigotimes_{k=i+1}^{v} \mathbf{S}_{\neg i})$$

Fully Relational: $\mathbf{S}_r$ is $\mathbf{S}$ with $\mathbf{S}_i = \mathbf{A}_i$ and $\mathbf{S}_{\neg i} = \mathbf{A}_{\neg i}$

Weakly Relational: $\mathbf{S}_w$ is $\mathbf{S}$ with $\mathbf{S}_i = \mathbf{A}_i$ and $\mathbf{S}_{\neg i} = \mathbf{A}_{\neg i}$ or $\mathbf{A}_f$

Non-Relational: $\mathbf{S}_n$ is $\mathbf{S}$ with $\mathbf{S}_i = \mathbf{A}$ and $\mathbf{S}_{\neg i} = \mathbf{A}_f$

With $\mathbf{A}_f$ forgetful and $\mathbf{A}_i$ and $\mathbf{A}_{\neg i}$ nontrivial abstractions.
For $\mathbf{S}_r$ all factors in $\bigoplus$ are the same; we can take $\mathbf{S}_r = \mathbf{S}(x_1)$.

Fully, Weakly and Non-Relational Analysis

Consider compositional (probabilistic) abstractions of the form:

$$\mathbf{S} = \bigoplus_{i=1}^{v} \mathbf{S}(x_i) \quad \text{with} \quad \mathbf{S}(x_i) = (\bigotimes_{k=1}^{i-1} \mathbf{S}_{\neg i}) \otimes \mathbf{S}_i \otimes (\bigotimes_{k=i+1}^{v} \mathbf{S}_{\neg i})$$

Fully Relational: $\mathbf{S}_r$ is $\mathbf{S}$ with $\mathbf{S}_i = \mathbf{A}_i$ and $\mathbf{S}_{\neg i} = \mathbf{A}_{\neg i}$

Weakly Relational: $\mathbf{S}_w$ is $\mathbf{S}$ with $\mathbf{S}_i = \mathbf{A}_i$ and $\mathbf{S}_{\neg i} = \mathbf{A}_{\neg i}$ or $\mathbf{A}_f$

Non-Relational: $\mathbf{S}_n$ is $\mathbf{S}$ with $\mathbf{S}_i = \mathbf{A}$ and $\mathbf{S}_{\neg i} = \mathbf{A}_f$

With $\mathbf{A}_f$ forgetful and $\mathbf{A}_i$ and $\mathbf{A}_{\neg i}$ nontrivial abstractions.
For $\mathbf{S}_r$ all factors in $\bigoplus$ are the same; we can take $\mathbf{S}_r = \mathbf{S}(x_1)$.

Examples

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

$\mathbf{P}\backslash\mathbf{R}$	$\emptyset$	$\mathbf{S}_n$	$\mathbf{S}_w$	$\mathbf{S}_r$	id
$\emptyset$	0	0	0	0	0
$\mathbf{S}_n$	1.58	0	0	0	0
$\mathbf{S}_w$	1.58	0	0	0	0
$\mathbf{S}_r$	1.58	0	0	0	0
id	2.55	1	1	1	0

Using cast d abstraction : $\mathbf{A}_d$ lifted $\alpha(x) = x \bmod d$

$\mathbf{S}_n$ is $\mathbf{S}$ with $\mathbf{S}_i = \mathbf{S}_4, \mathbf{S}_{\neg i} = \mathbf{A}_1$

$\mathbf{S}_w$ is $\mathbf{S}$ with $\mathbf{S}_i = \mathbf{S}_4, \mathbf{S}_{\neg i} = \mathbf{A}_2$

$\mathbf{S}_r$ is $\mathbf{S}$ with $\mathbf{S}_i = \mathbf{S}_{\neg i} = \mathbf{A}_4$

Examples

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

$\mathbf{P} \backslash \mathbf{R}$	$\emptyset$	$\mathbf{S}_n$	$\mathbf{S}_w$	$\mathbf{S}_r$	*id*
$\emptyset$	0	0	0	0	0
$\mathbf{S}_n$	1.73	0	0	0	0
$\mathbf{S}_w$	2.24	1	0	0	0
$\mathbf{S}_r$	2.24	1	1	0	0
id	3.61	3.61	3.61	3.61	0

Using cast d abstraction : $\mathbf{A}_d$ lifted $\alpha(x) = x \bmod d$

$$\mathbf{S}_n \text{ is } \mathbf{S} \quad \text{with} \quad \mathbf{S}_i = \mathbf{S}_4, \mathbf{S}_{\neg i} = \mathbf{A}_1$$

$$\mathbf{S}_w \text{ is } \mathbf{S} \quad \text{with} \quad \mathbf{S}_i = \mathbf{S}_4, \mathbf{S}_{\neg i} = \mathbf{A}_2$$

$$\mathbf{S}_r \text{ is } \mathbf{S} \quad \text{with} \quad \mathbf{S}_i = \mathbf{S}_{\neg i} = \mathbf{A}_4$$

Examples

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

$\mathbf{P} \backslash \mathbf{R}$	$\emptyset$	$\mathbf{S}_n$	$\mathbf{S}_w$	$\mathbf{S}_r$	id
$\emptyset$	0	0	0	0	0
$\mathbf{S}_n$	1.88	0.89	0.89	0.89	0
$\mathbf{S}_w$	2.14	1.52	1.29	1.29	0
$\mathbf{S}_r$	2.24	1.64	1.50	1.41	0
id	3.61	3.60	3.59	3.58	0

Using cast d abstraction : $\mathbf{A}_d$ lifted $\alpha(x) = x \bmod d$

$$\mathbf{S}_n \text{ is } \mathbf{S} \quad \text{with} \quad \mathbf{S}_i = \mathbf{S}_4, \mathbf{S}_{\neg i} = \mathbf{A}_1$$
$$\mathbf{S}_w \text{ is } \mathbf{S} \quad \text{with} \quad \mathbf{S}_i = \mathbf{S}_4, \mathbf{S}_{\neg i} = \mathbf{A}_2$$
$$\mathbf{S}_r \text{ is } \mathbf{S} \quad \text{with} \quad \mathbf{S}_i = \mathbf{S}_{\neg i} = \mathbf{A}_4$$

Examples

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

$\mathbf{P}\backslash\mathbf{R}$	$\emptyset$	$\mathbf{S}_n$	$\mathbf{S}_w$	$\mathbf{S}_r$	id
$\emptyset$	0	0	0	0	0
$\mathbf{S}_n$	1.77	0.89	0.89	0.89	0
$\mathbf{S}_w$	2.24	1.52	1.29	1.29	0
$\mathbf{S}_r$	2.24	1.64	1.50	1.41	0
id	3.61	3.60	3.59	3.58	0

Using cast d abstraction : $\mathbf{A}_d$ lifted $\alpha(x) = x \bmod d$

$\mathbf{S}_n$ is $\mathbf{S}$ with $\mathbf{S}_i = \mathbf{S}_4, \mathbf{S}_{\neg i} = \mathbf{A}_1$

$\mathbf{S}_w$ is $\mathbf{S}$ with $\mathbf{S}_i = \mathbf{S}_4, \mathbf{S}_{\neg i} = \mathbf{A}_2$

$\mathbf{S}_r$ is $\mathbf{S}$ with $\mathbf{S}_i = \mathbf{S}_{\neg i} = \mathbf{A}_4$

Further Work
Conclusions

Conclusions

Some applications of PAI:

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

Conclusions

Some applications of PAI:

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

Conclusions

Some applications of PAI:

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

Conclusions

Some applications of PAI:

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

Conclusions

Some applications of PAI:

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

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:

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

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. $p_i \in \mathbb{Z}$.

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:

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

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. $p_i \in \mathbb{Z}$.

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:

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

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. $p_i \in \mathbb{Z}$.

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:

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

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. $p_i \in \mathbb{Z}$.

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:

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

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

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)

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)

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).

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)

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)

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)

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?

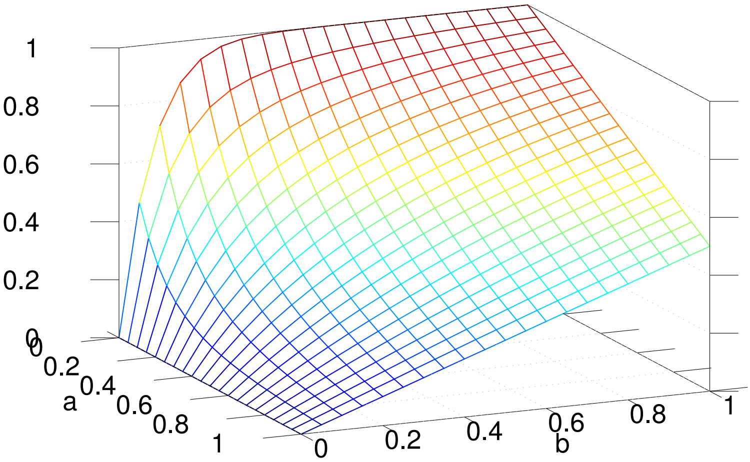
Introduced by McIver and Morgan (2005).
Discussed in detail by Gretz, Katoen, McIver (2012)

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*.



Contexts: Advance Normalisation

For all possible values of the variable probabilities p_i compute their normalisation, compute the possible contexts.

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

Example

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

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

Dynamic Probabilities

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

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

This gives the LOS Semantics for variable probabilities:

$$\{\![\textbf{choose}]^{p_1:S_1} \ldots \texttt{or}\ p_n : S_n\ \textbf{or}\ \ell\}\!\}_{LOS} = \{\!\{S_i\}\!\}_{LOS} \cup \bigcup_{i=1}^{n} \left\{ \sum_{c_j \in \textbf{Value}(p_i)} \sum_{[d_1\ldots d_n] \in \mathcal{C}[p_1\ldots p_n]} c_{j_{[d_1\ldots d_n]}} \cdot \mathbf{P}^{p_i[p_1\ldots p_n]}_{c_j[d_1\ldots d_n]} \otimes \mathbf{E}(\ell, init(S_i)) \right\}$$

Dynamic Probabilities

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

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

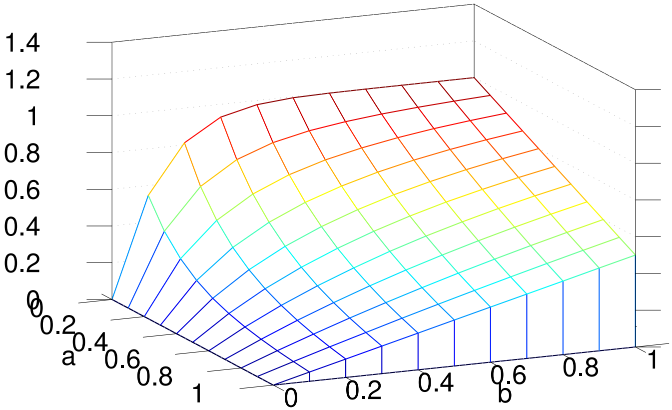
This gives the LOS Semantics for variable probabilities:

$$\{\!\{[\mathbf{choose}]^{p_1:S_1} \; \dots \texttt{or} \; p_n : S_n \; \mathbf{or} \; \ell\}\!\}_{LOS} = \{\!\{S_i\}\!\}_{LOS} \cup$$
$$\bigcup_{i=1}^{n} \left\{ \sum_{c_j \in \mathbf{Value}(p_i)} \; \sum_{[d_1 \dots d_n] \in \mathcal{C}[p_1 \dots p_n]} c_{j_{[d_1 \dots d_n]}} \cdot \mathbf{P}^{p_i[p_1 \dots p_n]}_{c_j[d_1 \dots d_n]} \otimes \mathbf{E}(\ell, init(S_i)) \right\}$$

Learning how to shoot straight

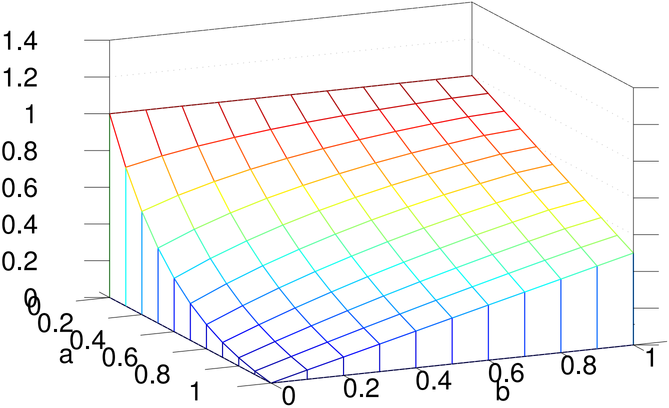
```
begin
# initialise skills of A
akl := ak; aml := am;
# who's first
choose 1:{t:=0} or 1:{t:=1} ro;
# continue until ...
c := 1;
while c == 1 do
  if (t==0) then
    choose akl:{c:=0} or aml:{t:=1} ro
  else
    choose bk:{c:=0} or bm:{t:=0} ro
  fi;
  akl := @inc(akl); aml := @dec(aml);
od;
stop;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

Back to the two Cowboys



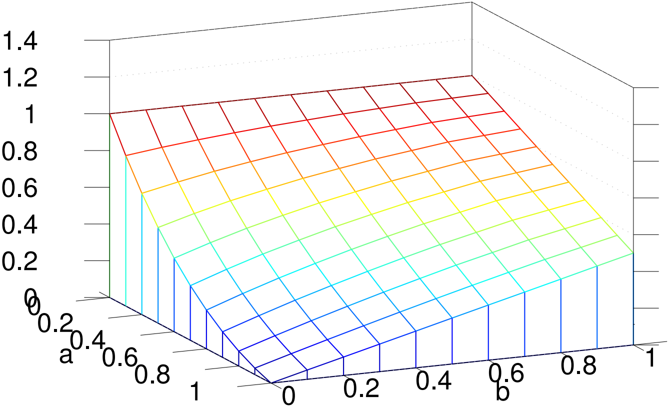
Learning rate 0.

Back to the two Cowboys



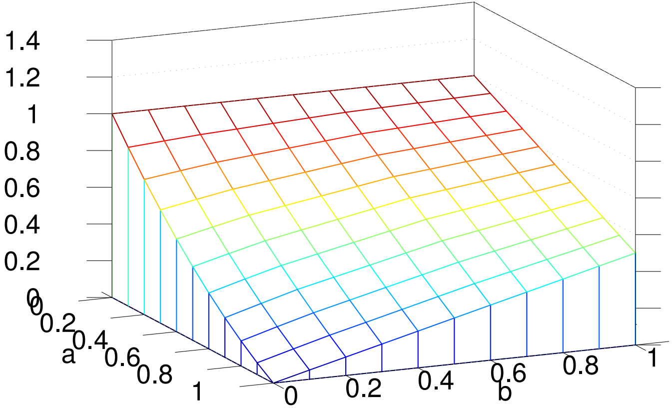
Learning rate 1.

Back to the two Cowboys



Learning rate 2.

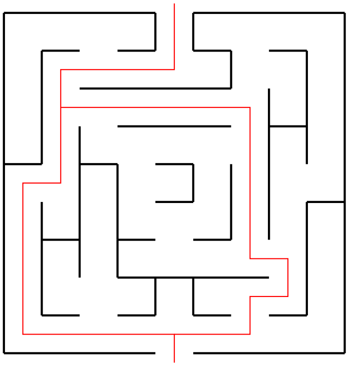
Back to the two Cowboys



Learning rate 4.

LOS for Program Synthesis

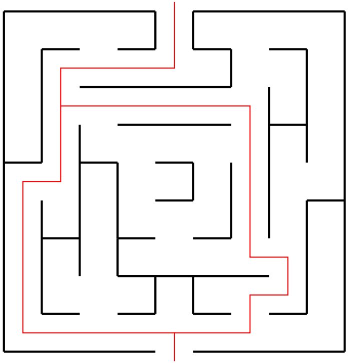
Finding the minimum length path vs minimum value of functions

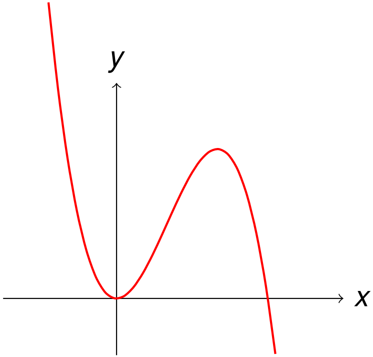


As usual (for now): Take the best non-linear optimisation tool money can't buy (leave it to "them" to make it work).

LOS for Program Synthesis

Finding the minimum length path vs minimum value of functions

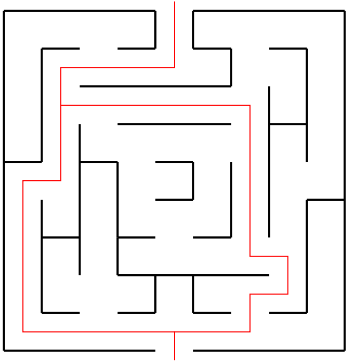


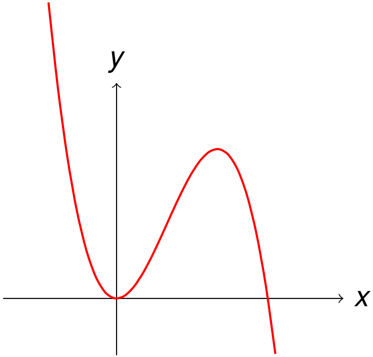


As usual (for now): Take the best non-linear optimisation tool money can't buy (leave it to "them" to make it work).

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

$$\ldots [\texttt{choose}]^\ell \; p_1 : S_1 \; \texttt{or} \; \ldots \; \texttt{or} \; p_n : S_n \; \texttt{ro}; \; \ldots$$

- Construct the parametric LOS semantics/operator, i.e.

$$[\![P(\lambda_1, \lambda_2, \ldots, \lambda_n)]\!] = \mathbf{T}(\lambda_1, \lambda_2, \ldots, \lambda_n)$$

- Establish constraints on functional behaviour, e.g.

$$\|\mathbf{A}^\dagger \mathbf{T}(\lambda_1, \lambda_2, \ldots, \lambda_n)\mathbf{A} - [\![S]\!]\| = 0$$

- Additional non-functional (performance) objectives

$$\min_{\lambda_1, \lambda_2, \ldots, \lambda_n} \Phi(\mathbf{T}(\lambda_1, \lambda_2, \ldots, \lambda_n))$$

A General Approach

- Consider parameterised program $P(\lambda_1, \lambda_2, \ldots, \lambda_n)$ with

$$\ldots [\mathtt{choose}]^{\ell}\ \lambda_1 : S_1\ \mathtt{or}\ \ldots\ \mathtt{or}\ \lambda_n : S_n\ \mathtt{ro};\ \ldots$$

- Construct the parametric LOS semantics/operator, i.e.

$$[\![P(\lambda_1, \lambda_2, \ldots, \lambda_n)]\!] = \mathbf{T}(\lambda_1, \lambda_2, \ldots, \lambda_n)$$

- Establish constraints on functional behaviour, e.g.

$$\|\mathbf{A}^{\dagger}\mathbf{T}(\lambda_1, \lambda_2, \ldots, \lambda_n)\mathbf{A} - [\![S]\!]\| = 0$$

- Additional non-functional (performance) objectives

$$\min_{\lambda_1, \lambda_2, \ldots, \lambda_n} \Phi(\mathbf{T}(\lambda_1, \lambda_2, \ldots, \lambda_n))$$

A General Approach

- Consider parameterised program $P(\lambda_1, \lambda_2, \ldots, \lambda_n)$ with

$$\ldots [\texttt{opt}]^{\ell}\ S_1\ \texttt{or}\ \ldots\ \texttt{or}\ S_n\ \texttt{top};\ \ldots$$

- Construct the parametric LOS semantics/operator, i.e.

$$[\![P(\lambda_1, \lambda_2, \ldots, \lambda_n)]\!] = \mathbf{T}(\lambda_1, \lambda_2, \ldots, \lambda_n)$$

- Establish constraints on functional behaviour, e.g.

$$\|\mathbf{A}^{\dagger}\mathbf{T}(\lambda_1, \lambda_2, \ldots, \lambda_n)\mathbf{A} - [\![S]\!]\| = 0$$

- Additional non-functional (performance) objectives

$$\min_{\lambda_1, \lambda_2, \ldots, \lambda_n} \Phi(\mathbf{T}(\lambda_1, \lambda_2, \ldots, \lambda_n))$$

A General Approach

- Consider parameterised program $P(\lambda_1, \lambda_2, \dots, \lambda_n)$ with

$$\dots [\texttt{choose}]^{\ell}\ \lambda_1 : S_1\ \texttt{or}\ \dots\ \texttt{or}\ \lambda_n : S_n\ \texttt{ro};\ \dots$$

- Construct the parametric LOS semantics/operator, i.e.

$$[\![P(\lambda_1, \lambda_2, \dots, \lambda_n)]\!] = \mathbf{T}(\lambda_1, \lambda_2, \dots, \lambda_n)$$

- Establish constraints on functional behaviour, e.g.

$$\|\mathbf{A}^{\dagger}\mathbf{T}(\lambda_1, \lambda_2, \dots, \lambda_n)\mathbf{A} - [\![S]\!]\| = 0$$

- Additional non-functional (performance) objectives

$$\min_{\lambda_1, \lambda_2, \dots, \lambda_n} \Phi(\mathbf{T}(\lambda_1, \lambda_2, \dots, \lambda_n))$$

A General Approach

- Consider parameterised program $P(\lambda_1, \lambda_2, \ldots, \lambda_n)$ with

$$\ldots [\mathtt{choose}]^\ell\ \lambda_1 : S_1\ \mathtt{or}\ \ldots\ \mathtt{or}\ \lambda_n : S_n\ \mathtt{ro};\ \ldots$$

- Construct the parametric LOS semantics/operator, i.e.

$$[\![P(\lambda_1, \lambda_2, \ldots, \lambda_n)]\!] = \mathbf{T}(\lambda_1, \lambda_2, \ldots, \lambda_n)$$

- Establish constraints on functional behaviour, e.g.

$$\|\mathbf{A}^\dagger \mathbf{T}(\lambda_1, \lambda_2, \ldots, \lambda_n)\mathbf{A} - [\![S]\!]\| = 0$$

- Additional non-functional (performance) objectives

$$\min_{\lambda_1, \lambda_2, \ldots, \lambda_n} \Phi(\mathbf{T}(\lambda_1, \lambda_2, \ldots, \lambda_n))$$

A General Approach

- Consider parameterised program $P(\lambda_1, \lambda_2, \ldots, \lambda_n)$ with

$$\ldots [\texttt{choose}]^{\ell}\ \lambda_1 : S_1\ \texttt{or}\ \ldots\ \texttt{or}\ \lambda_n : S_n\ \texttt{ro};\ \ldots$$

- Construct the parametric LOS semantics/operator, i.e.

$$[\![P(\lambda_1, \lambda_2, \ldots, \lambda_n)]\!] = \mathbf{T}(\lambda_1, \lambda_2, \ldots, \lambda_n)$$

- Establish constraints on functional behaviour, e.g.

$$\mathbf{A}^{\dagger}\mathbf{T}(\lambda_1, \lambda_2, \ldots, \lambda_n)\mathbf{A} = [\![S]\!]$$

- Additional non-functional (performance) objectives

$$\min_{\lambda_1, \lambda_2, \ldots, \lambda_n} \Phi(\mathbf{T}(\lambda_1, \lambda_2, \ldots, \lambda_n))$$

A General Approach

- Consider parameterised program $P(\lambda_1, \lambda_2, \ldots, \lambda_n)$ with

$$\ldots [\texttt{choose}]^{\ell}\ \lambda_1 : S_1\ \texttt{or}\ \ldots\ \texttt{or}\ \lambda_n : S_n\ \texttt{ro};\ \ldots$$

- Construct the parametric LOS semantics/operator, i.e.

$$[\![P(\lambda_1, \lambda_2, \ldots, \lambda_n)]\!] = \mathbf{T}(\lambda_1, \lambda_2, \ldots, \lambda_n)$$

- Establish constraints on functional behaviour, e.g.

$$\|\mathbf{A}^{\dagger}\mathbf{T}(\lambda_1, \lambda_2, \ldots, \lambda_n)\mathbf{A} - [\![S]\!]\| = 0$$

- Additional non-functional (performance) objectives

$$\min_{\lambda_1, \lambda_2, \ldots, \lambda_n} \Phi(\mathbf{T}(\lambda_1, \lambda_2, \ldots, \lambda_n))$$

Swapping: The XOR Trick

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

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

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

$$[\texttt{skip}]^1$$
$$[x := y]^2 \quad [x := z]^3$$
$$[y := x]^4 \quad [y := z]^5$$
$$[z := x]^6 \quad [z := y]^7$$
$$[x := (x + y) \bmod 2]^8 \quad [x := (x + z) \bmod 2]^9$$
$$[y := (y + x) \bmod 2]^{10} \quad [y := (y + z) \bmod 2]^{11}$$
$$[z := (z + x) \bmod 2]^{12} \quad [z := (z + y) \bmod 2]^{13}$$

Swapping: The XOR Trick

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

$$
\begin{array}{l}
[\texttt{choose}]^1\ \lambda_{1,1} : S_1\ \texttt{or}\ \ldots\ \texttt{or}\ \lambda_{1,n} : S_n\ \texttt{ro};\\
[\texttt{choose}]^2\ \lambda_{2,1} : S_1\ \texttt{or}\ \ldots\ \texttt{or}\ \lambda_{2,n} : S_n\ \texttt{ro};\\
[\texttt{choose}]^3\ \lambda_{3,1} : S_1\ \texttt{or}\ \ldots\ \texttt{or}\ \lambda_{3,n} : S_n\ \texttt{ro};
\end{array}
$$

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

$$
\begin{array}{rl}
\multicolumn{2}{c}{[\texttt{skip}]^1}\\
[x := y]^2 & [x := z]^3\\
[y := x]^4 & [y := z]^5\\
[z := x]^6 & [z := y]^7\\
[x := (x+y) \bmod 2]^8 & [x := (x+z) \bmod 2]^9\\
[y := (y+x) \bmod 2]^{10} & [y := (y+z) \bmod 2]^{11}\\
[z := (z+x) \bmod 2]^{12} & [z := (z+y) \bmod 2]^{13}
\end{array}
$$

Swapping: Parameterised LOS and Objective

Using 13 transfer functions $\mathbf{F}_1 \ldots \mathbf{F}_{13}$ to define

$$\mathbf{T}(\lambda_{ij}) = \prod_{i=1}^{3} \mathbf{T}_i(\lambda_{ij}) \quad \text{with} \quad \mathbf{T}_i(\lambda_{ij}) = \sum_{j=1}^{13} \lambda_{ij} \mathbf{F}_j$$

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

$$\mathbf{S} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \begin{array}{ll} x \mapsto 0 & y \mapsto 0 \\ x \mapsto 0 & y \mapsto 1 \\ x \mapsto 1 & y \mapsto 0 \\ x \mapsto 1 & y \mapsto 1 \end{array}$$

Objective: $\min \Phi_{00}(\lambda_{ij}) = \|\mathbf{A}^\dagger \mathbf{T}(\lambda_{ij}) \mathbf{A} - \mathbf{S}\|_2$ or $\min \Phi_{\rho\omega}(\lambda_{ij})$ which also penalises for reading or writing to z; using the abstraction $\mathbf{A} = \mathbf{I}_{(4)} \otimes \mathbf{A}_{f(2)} = \mathrm{diag}(1,1,1,1) \otimes (1,1)^t$.

Swapping: Parameterised LOS and Objective

Using 13 transfer functions $\mathbf{F}_1 \ldots \mathbf{F}_{13}$ to define

$$\mathbf{T}(\lambda_{ij}) = \prod_{i=1}^{3} \mathbf{T}_i(\lambda_{ij}) \quad \text{with} \quad \mathbf{T}_i(\lambda_{ij}) = \sum_{j=1}^{13} \lambda_{ij} \mathbf{F}_j$$

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

$$\mathbf{S} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \begin{array}{ll} x \mapsto 0 & y \mapsto 0 \\ x \mapsto 0 & y \mapsto 1 \\ x \mapsto 1 & y \mapsto 0 \\ x \mapsto 1 & y \mapsto 1 \end{array}$$

Objective: $\min \Phi_{00}(\lambda_{ij}) = \|\mathbf{A}^{\dagger}\mathbf{T}(\lambda_{ij})\mathbf{A} - \mathbf{S}\|_2$ or $\min \Phi_{\rho\omega}(\lambda_{ij})$ which also penalises for reading or writing to z; using the abstraction $\mathbf{A} = \mathbf{I}_{(4)} \otimes \mathbf{A}_{f(2)} = \mathrm{diag}(1,1,1,1) \otimes (1,1)^t$.

Swapping: Parameterised LOS and Objective

Using 13 transfer functions $\mathbf{F}_1 \ldots \mathbf{F}_{13}$ to define

$$\mathbf{T}(\lambda_{ij}) = \prod_{i=1}^{3} \mathbf{T}_i(\lambda_{ij}) \quad \text{with} \quad \mathbf{T}_i(\lambda_{ij}) = \sum_{j=1}^{13} \lambda_{ij} \mathbf{F}_j$$

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

$$\mathbf{S} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \begin{array}{ll} x \mapsto 0 & y \mapsto 0 \\ x \mapsto 0 & y \mapsto 1 \\ x \mapsto 1 & y \mapsto 0 \\ x \mapsto 1 & y \mapsto 1 \end{array}$$

Objective: $\min \Phi_{00}(\lambda_{ij}) = \|\mathbf{A}^\dagger \mathbf{T}(\lambda_{ij}) \mathbf{A} - \mathbf{S}\|_2$ or $\min \Phi_{\rho\omega}(\lambda_{ij})$ which also penalises for reading or writing to z; using the abstraction $\mathbf{A} = \mathbf{I}_{(4)} \otimes \mathbf{A}_{f(2)} = \mathrm{diag}(1,1,1,1) \otimes (1,1)^t$.

Swapping: Test Runs

Using `octave`: if we start with a swap which uses z, like

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

represented by λ_{ij} given as:

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

For $\min \Phi_{00}$ we get no change; but with $\min \Phi_{11}$ (after 12 iterations) we get with `octave` the optimal λ_{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:

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

Swapping: Test Runs

Using `octave`: if we start with a swap which uses z, like

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

represented by λ_{ij} given as:

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

For $\min \Phi_{00}$ we get no change; but with $\min \Phi_{11}$ (after 12 iterations) we get with `octave` the optimal λ_{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:

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

Swapping: Test Runs

Using `octave`: if we start with a swap which uses z, like

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

represented by λ_{ij} given as:

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

For $\min \Phi_{00}$ we get no change; but with $\min \Phi_{11}$ (after 12 iterations) we get with `octave` the optimal λ_{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:

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

Swapping: Test Runs

Using `octave`: if we start with a swap which uses z, like

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

represented by λ_{ij} given as:

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

For $\min \Phi_{00}$ we get no change; but with $\min \Phi_{11}$ (after 12 iterations) we get with `octave` the optimal λ_{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:

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

Swapping: Test Runs

Using `octave`: if we start with a swap which uses z, like

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

represented by λ_{ij} given as:

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

For $\min \Phi_{00}$ we get no change; but with $\min \Phi_{11}$ (after 12 iterations) we get with `octave` the optimal λ_{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:

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

Swapping: Test Runs

For randomly chosen initial values for λ_{ij}:

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

For $\min \Phi_{11}$ (after 9 iterations) we get the optimal λ_{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:

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

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

Swapping: Test Runs

For randomly chosen initial values for λ_{ij}:

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

For $\min \Phi_{11}$ (after 9 iterations) we get the optimal λ_{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:

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

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

Swapping: Test Runs

For randomly chosen initial values for λ_{ij}:

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

For $\min \Phi_{11}$ (after 9 iterations) we get the optimal λ_{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:

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

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

Swapping: Test Runs

For randomly chosen initial values for λ_{ij}:

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

For $\min \Phi_{11}$ (after 9 iterations) we get the optimal λ_{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:

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

For Φ_{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.
- Di Pierro, Sotin, Wiklicky: *Relational analysis and precision via probabilistic abstract interpretation*. In QAPL'08 – Workshop on Quantitative Aspects of Programming Languages, ENTCS Elsevier, 2008.
- Wiklicky: *On Dynamical Probabilities, or: How to Learn to Shoot Straight*. in Proceedings of Coordination'16, LNCS 9686, Springer, 2016.
- Israel and Greville: *Gereralized Inverses – Theory and Applications*. CMS Books in Mathematics, Springer, 2003.

Some References

- Di Pierro, Wiklicky: A logico-algebraic approach to probabilistic program analysis Pre-Proceedings of LOPSTR'05.
- Di Pierro, Sotin, Wiklicky: *Relational analysis and precision via probabilistic abstract interpretation*. In QAPL'08 – Workshop on Quantitative Aspects of Programming Languages, ENTCS Elsevier, 2008.
- Wiklicky: *On Dynamical Probabilities, or: How to Learn to Shoot Straight*. in Proceedings of Coordination'16, LNCS 9686, Springer, 2016.
- Israel and Greville: *Gereralized Inverses – Theory and Applications*. CMS Books in Mathematics, Springer, 2003.

Some References

- Di Pierro, Wiklicky: A logico-algebraic approach to probabilistic program analysis Pre-Proceedings of LOPSTR'05.
- Di Pierro, Sotin, Wiklicky: *Relational analysis and precision via probabilistic abstract interpretation*. In QAPL'08 – Workshop on Quantitative Aspects of Programming Languages, ENTCS Elsevier, 2008.
- Wiklicky: *On Dynamical Probabilities, or: How to Learn to Shoot Straight*. in Proceedings of Coordination'16, LNCS 9686, Springer, 2016.
- Israel and Greville: *Gereralized Inverses – Theory and Applications*. CMS Books in Mathematics, Springer, 2003.

Some References

- Di Pierro, Wiklicky: A logico-algebraic approach to probabilistic program analysis Pre-Proceedings of LOPSTR'05.
- Di Pierro, Sotin, Wiklicky: *Relational analysis and precision via probabilistic abstract interpretation*. In QAPL'08 – Workshop on Quantitative Aspects of Programming Languages, ENTCS Elsevier, 2008.
- Wiklicky: *On Dynamical Probabilities, or: How to Learn to Shoot Straight*. in Proceedings of Coordination'16, LNCS 9686, Springer, 2016.
- Israel and Greville: *Gereralized Inverses – Theory and Applications*. CMS Books in Mathematics, Springer, 2003.