Probabilistic Program Analysis

Data Flow Analysis and Regression

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

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

Classical Dataflow Analysis

The problem could be to identify at any program point the variables which are live, i.e. which may later be used in an assignment or test.

There are two phases of a classical *LV* analysis:

(i) formulation of data-flow equations as set equations (or more generally over a property lattice *L*),

(ii) finding or constructing solutions to these equations, for example, via a fixed-point construction.

Classical Dataflow Analysis

The problem could be to identify at any program point the variables which are live, i.e. which may later be used in an assignment or test.

There are two phases of a classical *LV* analysis:

(i) formulation of data-flow equations as set equations (or more generally over a property lattice *L*),

(ii) finding or constructing solutions to these equations, for example, via a fixed-point construction.

Classical Dataflow Analysis

The problem could be to identify at any program point the variables which are live, i.e. which may later be used in an assignment or test.

There are two phases of a classical *LV* analysis:

(i) formulation of data-flow equations as set equations (or more generally over a property lattice *L*),

(ii) finding or constructing solutions to these equations, for example, via a fixed-point construction.

Classical Dataflow Analysis

The problem could be to identify at any program point the variables which are live, i.e. which may later be used in an assignment or test.

There are two phases of a classical *LV* analysis:

(i) formulation of data-flow equations as set equations (or more generally over a property lattice *L*),

(ii) finding or constructing solutions to these equations, for example, via a fixed-point construction.

Example

Consider a program like:

$$[x := 1]^1;$$
$$[y := 2]^2;$$
$$[x := x + y \bmod 4]^3;$$
$$\texttt{if } [x > 2]^4 \texttt{ then } [z := x]^5 \texttt{ else } [z := y]^6 \texttt{ fi}$$

Extract statically the control flow relation – i.e. is it possible to go from lable ℓ to label ℓ'?

$$\mathit{flow} = \{(1,2),(2,3),(3,4),(4,\underline{5}),(4,6)\}$$

Nielson, Nielson, Hankin: *Principles of Program Analysis*. Springer, 99/05.

Example

Consider a program like:

$$[x := 1]^1;$$
$$[y := z]^2;$$
$$[x := x + y \text{ mod } 4]^3;$$
$$\texttt{if } [x > 2]^4 \texttt{ then } [z := x]^5 \texttt{ else } [z := y]^6 \texttt{ fi}$$

Extract statically the control flow relation – i.e. is it possible to go from lable ℓ to label ℓ'?

$$flow = \{(1,2),(2,3),(3,4),(4,\underline{5}),(4,6)\}$$

Nielson, Nielson, Hankin: *Principles of Program Analysis*. Springer, 99/05.

Example

Consider a program like:

$$[x := 1]^1;$$
$$[y := z]^2;$$
$$[x := x + y \bmod 4]^3;$$
$$\texttt{if } [x > 2]^4 \texttt{ then } [z := x]^5 \texttt{ else } [z := y]^6 \texttt{ fi}$$

Extract statically the control flow relation – i.e. is it possible to go from lable ℓ to label ℓ'?

$$\mathit{flow} = \{(1,2),(2,3),(3,4),(4,\underline{5}),(4,6)\}$$

Nielson, Nielson, Hankin: *Principles of Program Analysis*. Springer, 99/05.

Example

Consider a program like:

$$[x := 1]^1;$$
$$[y := z]^2;$$
$$[x := x + y \bmod 4]^3;$$
$$\texttt{if } [x > 2]^4 \texttt{ then } [z := x]^5 \texttt{ else } [z := y]^6 \texttt{ fi}$$

Extract statically the control flow relation – i.e. is it possible to go from lable ℓ to label ℓ'?

$$\mathit{flow} = \{(1,2),(2,3),(3,4),(4,\underline{5}),(4,6)\}$$

Nielson, Nielson, Hankin: *Principles of Program Analysis*. Springer, 99/05.

Example

Consider a program like:

$$
\begin{array}{l}
[x := 1]^1; \\
[y := z]^2; \\
[x := x + y \bmod 4]^3; \\
\texttt{if } [x > 2]^4 \texttt{ then } [z := x]^5 \texttt{ else } [z := y]^6 \texttt{ fi}
\end{array}
$$

Extract statically the control flow relation – i.e. is it possible to go from lable ℓ to label ℓ'?

$$flow = \{(1,2),(2,3),(3,4),(4,\underline{5}),(4,6)\}$$

Nielson, Nielson, Hankin: *Principles of Program Analysis*. Springer, 99/05.

(Local) Transfer Functions

$$\begin{aligned} gen_{\mathrm{LV}}([x := a]^\ell) &= FV(a) \\ gen_{\mathrm{LV}}([\mathtt{skip}]^\ell) &= \emptyset \\ gen_{\mathrm{LV}}([b]^\ell) &= FV(b) \end{aligned}$$

$$\begin{aligned} kill_{\mathrm{LV}}([x := a]^\ell) &= \{\mathtt{x}\} \\ kill_{\mathrm{LV}}([\mathtt{skip}]^\ell) &= \emptyset \\ kill_{\mathrm{LV}}([b]^\ell) &= \emptyset \end{aligned}$$

$$f_\ell^{LV} : \mathcal{P}(\mathbf{Var}_\star) \to \mathcal{P}(\mathbf{Var}_\star)$$

$$f_\ell^{LV}(X) = X \setminus kill_{\mathrm{LV}}([B]^\ell) \cup gen_{\mathrm{LV}}([B]^\ell)$$

(Local) Transfer Functions

$$\begin{array}{rcl} gen_{\mathrm{LV}}([x := a]^{\ell}) & = & FV(a) \\ gen_{\mathrm{LV}}([\mathtt{skip}]^{\ell}) & = & \emptyset \\ gen_{\mathrm{LV}}([b]^{\ell}) & = & FV(b) \end{array}$$

$$\begin{array}{rcl} kill_{\mathrm{LV}}([x := a]^{\ell}) & = & \{\mathtt{x}\} \\ kill_{\mathrm{LV}}([\mathtt{skip}]^{\ell}) & = & \emptyset \\ kill_{\mathrm{LV}}([b]^{\ell}) & = & \emptyset \end{array}$$

$$f_{\ell}^{LV} : \mathcal{P}(\mathbf{Var}_{\star}) \rightarrow \mathcal{P}(\mathbf{Var}_{\star})$$

$$f_{\ell}^{LV}(X) = X \setminus kill_{\mathrm{LV}}([B]^{\ell}) \cup gen_{\mathrm{LV}}([B]^{\ell})$$

(Local) Transfer Functions

$$
\begin{aligned}
gen_{\mathrm{LV}}([x := a]^\ell) &= FV(a) \\
gen_{\mathrm{LV}}([\mathtt{skip}]^\ell) &= \emptyset \\
gen_{\mathrm{LV}}([b]^\ell) &= FV(b)
\end{aligned}
$$

$$
\begin{aligned}
kill_{\mathrm{LV}}([x := a]^\ell) &= \{\mathtt{x}\} \\
kill_{\mathrm{LV}}([\mathtt{skip}]^\ell) &= \emptyset \\
kill_{\mathrm{LV}}([b]^\ell) &= \emptyset
\end{aligned}
$$

$$f_\ell^{LV} : \mathcal{P}(\mathbf{Var}_\star) \rightarrow \mathcal{P}(\mathbf{Var}_\star)$$

$$f_\ell^{LV}(X) = X \setminus kill_{\mathrm{LV}}([B]^\ell) \cup gen_{\mathrm{LV}}([B]^\ell)$$

(Global) Control Flow

Formulate equations based on the control flow (relations):

$$\begin{aligned} \mathsf{LV}_{entry}(\ell) &= f_\ell^{LV}(\mathsf{LV}_{exit}(\ell)) \\ \mathsf{LV}_{exit}(\ell) &= \bigcup_{(\ell,\ell')\in flow} \mathsf{LV}_{entry}(\ell') \end{aligned}$$

Monotone Framework: Generalise this setting to lattice equations by using a general property lattice L instead of $\mathcal{P}(X)$.

This also gives ways to effectively construct solutions via various lattice theoretic concepts (fixed points, worklist, etc.)

(Global) Control Flow

Formulate equations based on the control flow (relations):

$$\begin{aligned} \mathsf{LV}_{entry}(\ell) &= f_\ell^{LV}(\mathsf{LV}_{exit}(\ell)) \\ \mathsf{LV}_{exit}(\ell) &= \bigcup_{(\ell,\ell')\in flow} \mathsf{LV}_{entry}(\ell') \end{aligned}$$

Monotone Framework: Generalise this setting to lattice equations by using a general property lattice L instead of $\mathcal{P}(X)$.

This also gives ways to effectively construct solutions via various lattice theoretic concepts (fixed points, worklist, etc.)

(Global) Control Flow

Formulate equations based on the control flow (relations):

$$\begin{aligned}\mathsf{LV}_{entry}(\ell) &= f_\ell^{LV}(\mathsf{LV}_{exit}(\ell)) \\ \mathsf{LV}_{exit}(\ell) &= \bigcup_{(\ell,\ell')\in flow} \mathsf{LV}_{entry}(\ell')\end{aligned}$$

Monotone Framework: Generalise this setting to lattice equations by using a general property lattice L instead of $\mathcal{P}(X)$.

This also gives ways to effectively construct solutions via various lattice theoretic concepts (fixed points, worklist, etc.)

Example

$$[x := 1]^1;\ [y := 2]^2;\ [x := x + y \bmod 4]^3;$$
$$\texttt{if}\ [x > 2]^4\ \texttt{then}\ [z := x]^5\ \texttt{else}\ [z := y]^6\ \texttt{fi}$$

Example

$$[x := 1]^1;\ [y := 2]^2;\ [x := x + y \bmod 4]^3;$$
$$\texttt{if}\ [x > 2]^4\ \texttt{then}\ [z := x]^5\ \texttt{else}\ [z := y]^6\ \texttt{fi}$$

Control Flow:

$$\mathit{flow} = \{(1,2),(2,3),(3,4),(4,\underline{5}),(4,6)\}$$

Example

$$[x := 1]^1;\ [y := 2]^2;\ [x := x + y \bmod 4]^3;$$
$$\texttt{if } [x > 2]^4 \texttt{ then } [z := x]^5 \texttt{ else } [z := y]^6 \texttt{ fi}$$

Auxiliary Functions:

	$gen_{LV}(\ell)$	$kill_{LV}(\ell)$
1	$\emptyset$	$\{x\}$
2	$\emptyset$	$\{y\}$
3	$\{x, y\}$	$\{x\}$
4	$\{x\}$	$\emptyset$
5	$\{x\}$	$\{z\}$
6	$\{y\}$	$\{z\}$

Example

$$[x := 1]^1;\ [y := 2]^2;\ [x := x + y \bmod 4]^3;$$
$$\texttt{if}\ [x > 2]^4\ \texttt{then}\ [z := x]^5\ \texttt{else}\ [z := y]^6\ \texttt{fi}$$

Equations (over $L = \mathcal{P}(\mathbf{Var})$)

$$\begin{aligned}
LV_{entry}(1) &= LV_{exit}(1) \setminus \{x\} \\
LV_{entry}(2) &= LV_{exit}(2) \setminus \{y\} \\
LV_{entry}(3) &= LV_{exit}(3) \setminus \{x\} \cup \{x, y\} \\
LV_{entry}(4) &= LV_{exit}(4) \cup \{x\} \\
LV_{entry}(5) &= LV_{exit}(5) \setminus \{z\} \cup \{x\} \\
LV_{entry}(6) &= LV_{exit}(6) \setminus \{z\} \cup \{y\}
\end{aligned}$$

Example

$$[x := 1]^1;\ [y := 2]^2;\ [x := x + y \bmod 4]^3;$$
$$\texttt{if } [x > 2]^4 \texttt{ then } [z := x]^5 \texttt{ else } [z := y]^6 \texttt{ fi}$$

Equations (over $L = \mathcal{P}(\mathbf{Var})$)

$$\begin{aligned}
\mathsf{LV}_{exit}(1) &= \mathsf{LV}_{entry}(2) \\
\mathsf{LV}_{exit}(2) &= \mathsf{LV}_{entry}(3) \\
\mathsf{LV}_{exit}(3) &= \mathsf{LV}_{entry}(4) \\
\mathsf{LV}_{exit}(4) &= \mathsf{LV}_{entry}(5) \cup \mathsf{LV}_{entry}(6) \\
\mathsf{LV}_{exit}(5) &= \emptyset \\
\mathsf{LV}_{exit}(6) &= \emptyset
\end{aligned}$$

Example

$$[x := 1]^1;\ [y := 2]^2;\ [x := x + y \bmod 4]^3;$$
$$\texttt{if}\ [x > 2]^4\ \texttt{then}\ [z := x]^5\ \texttt{else}\ [z := y]^6\ \texttt{fi}$$

Solutions (e.g. by fixed point iteration)

$$\begin{aligned}
LV_{entry}(1) &= \emptyset & LV_{exit}(1) &= \{x\} \\
LV_{entry}(2) &= \{x\} & LV_{exit}(2) &= \{x, y\} \\
LV_{entry}(3) &= \{x, y\} & LV_{exit}(3) &= \{x, y\} \\
LV_{entry}(4) &= \{x, y\} & LV_{exit}(4) &= \{x, y\} \\
LV_{entry}(5) &= \{x\} & LV_{exit}(5) &= \emptyset \\
LV_{entry}(6) &= \{y\} & LV_{exit}(6) &= \emptyset.
\end{aligned}$$

A Probabilistic Language (Variation)

We consider a simple language with a random assignment $\rho = \{\langle r_1, p_1\rangle, \ldots \langle r_n, p_n\rangle\}$ (rather than a probabilistic choice).

$$
\begin{array}{lll}
S & ::= & \texttt{skip} \\
 & | & x \; \texttt{:=} \; e(x_1, \ldots, x_n) \\
 & | & x \; \texttt{?=} \; \rho \\
 & | & S_1 \, \texttt{;} \; S_2 \\
 & | & \texttt{if } b \texttt{ then } S_1 \texttt{ else } S_2 \texttt{ fi} \\
 & | & \texttt{while } b \texttt{ do } S \texttt{ od}
\end{array}
$$

A Probabilistic Language (Variation)

We consider a simple language with a random assignment $\rho = \{\langle r_1, p_1\rangle, \ldots \langle r_n, p_n\rangle\}$ (rather than a probabilistic choice).

$$
\begin{array}{lll}
S & ::= & [\texttt{skip}]^\ell \\
 & \mid & [x := e(x_1, \ldots, x_n)]^\ell \\
 & \mid & [x \;\texttt{?=}\; \rho]^\ell \\
 & \mid & S_1 \;\texttt{;}\; S_2 \\
 & \mid & \texttt{if } [b]^\ell \texttt{ then } S_1 \texttt{ else } S_2 \texttt{ fi} \\
 & \mid & \texttt{while } [b]^\ell \texttt{ do } S \texttt{ od}
\end{array}
$$

Probabilistic Semantics

SOS:

$$\textbf{R0} \quad \langle \texttt{stop}, s\rangle \Rightarrow_1 \langle \texttt{stop}, s\rangle$$

$$\textbf{R1} \quad \langle \texttt{skip}, s\rangle \Rightarrow_1 \langle \texttt{stop}, s\rangle$$

$$\textbf{R2} \quad \langle v := e, s\rangle \Rightarrow_1 \langle \texttt{stop}, s[v \mapsto \mathcal{E}(e)s]\rangle$$

$$\textbf{R3} \quad \langle v \ ?= \rho, s\rangle \Rightarrow_{\rho(r)} \langle \texttt{stop}, s[v \mapsto r]\rangle$$

...

LOS:

$$\ldots$$

$$\mathbf{T}(\langle \ell_1, p, \ell_2\rangle) = \mathbf{U}(\mathrm{x} \leftarrow a) \otimes \mathbf{E}(\ell_1, \ell_2) \quad \text{for } [x := a]^{\ell_1}$$

$$\mathbf{T}(\langle \ell_1, p, \ell_2\rangle) = (\textstyle\sum_i \rho(r_i) \cdot \mathbf{U}(\mathrm{x} \leftarrow r_i)) \otimes \mathbf{E}(\ell_1, \ell_2) \quad \text{for } [x \ ?= \rho]^{\ell_1}$$

$$\ldots$$

Probabilistic Semantics

SOS:

$$\textbf{R0} \quad \langle \texttt{stop}, s \rangle \Rightarrow_1 \langle \texttt{stop}, s \rangle$$

$$\textbf{R1} \quad \langle \texttt{skip}, s \rangle \Rightarrow_1 \langle \texttt{stop}, s \rangle$$

$$\textbf{R2} \quad \langle v \;:=\; e, s \rangle \Rightarrow_1 \langle \texttt{stop}, s[v \mapsto \mathcal{E}(e)s] \rangle$$

$$\textbf{R3} \quad \langle v \;?\!=\; \rho, s \rangle \Rightarrow_{\rho(r)} \langle \texttt{stop}, s[v \mapsto r] \rangle$$

$\ldots$

LOS:

$$\begin{array}{rcll} & \ldots & & \\ \mathbf{T}(\langle \ell_1, p, \ell_2 \rangle) & = & \mathbf{U}(\mathtt{x} \leftarrow a) \otimes \mathbf{E}(\ell_1, \ell_2) & \text{for } [x \;:=\; a]^{\ell_1} \\ \mathbf{T}(\langle \ell_1, p, \ell_2 \rangle) & = & \left(\sum_i \rho(r_i) \cdot \mathbf{U}(\mathtt{x} \leftarrow r_i)\right) \otimes \mathbf{E}(\ell_1, \ell_2) & \text{for } [x \;?\!=\; \rho]^{\ell_1} \\ & \ldots & & \end{array}$$

(Local) Transfer Functions (extended)

$$\begin{aligned}
gen_{\mathrm{LV}}([x := a]^\ell) &= FV(a) \\
gen_{\mathrm{LV}}([x \mathrel{?{=}} \rho]^\ell) &= \emptyset \\
gen_{\mathrm{LV}}([\mathtt{skip}]^\ell) &= \emptyset \\
gen_{\mathrm{LV}}([b]^\ell) &= FV(b)
\end{aligned}$$

$$\begin{aligned}
kill_{\mathrm{LV}}([x := a]^\ell) &= \{\mathtt{x}\} \\
kill_{\mathrm{LV}}([x \mathrel{?{=}} \rho]^\ell) &= \{\mathtt{x}\} \\
kill_{\mathrm{LV}}([\mathtt{skip}]^\ell) &= \emptyset \\
kill_{\mathrm{LV}}([b]^\ell) &= \emptyset
\end{aligned}$$

$$f_\ell^{LV} : \mathcal{P}(\mathbf{Var}_\star) \to \mathcal{P}(\mathbf{Var}_\star)$$

$$f_\ell^{LV}(X) = X \setminus kill_{\mathrm{LV}}([B]^\ell) \cup gen_{\mathrm{LV}}([B]^\ell)$$

(Local) Transfer Functions (extended)

$$
\begin{aligned}
gen_{\mathrm{LV}}([x := a]^\ell) &= FV(a) \\
gen_{\mathrm{LV}}([x \mathrel{?=} \rho]^\ell) &= \emptyset \\
gen_{\mathrm{LV}}([\mathtt{skip}]^\ell) &= \emptyset \\
gen_{\mathrm{LV}}([b]^\ell) &= FV(b)
\end{aligned}
$$

$$
\begin{aligned}
kill_{\mathrm{LV}}([x := a]^\ell) &= \{\mathtt{x}\} \\
kill_{\mathrm{LV}}([x \mathrel{?=} \rho]^\ell) &= \{\mathtt{x}\} \\
kill_{\mathrm{LV}}([\mathtt{skip}]^\ell) &= \emptyset \\
kill_{\mathrm{LV}}([b]^\ell) &= \emptyset
\end{aligned}
$$

$$f_\ell^{LV} : \mathcal{P}(\mathbf{Var}_\star) \to \mathcal{P}(\mathbf{Var}_\star)$$

$$f_\ell^{LV}(X) = X \setminus kill_{\mathrm{LV}}([B]^\ell) \cup gen_{\mathrm{LV}}([B]^\ell)$$

(Local) Transfer Functions (extended)

$$
\begin{aligned}
gen_{\mathrm{LV}}([x := a]^\ell) &= FV(a) \\
gen_{\mathrm{LV}}([x \mathrel{?=} \rho]^\ell) &= \emptyset \\
gen_{\mathrm{LV}}([\mathtt{skip}]^\ell) &= \emptyset \\
gen_{\mathrm{LV}}([b]^\ell) &= FV(b)
\end{aligned}
$$

$$
\begin{aligned}
kill_{\mathrm{LV}}([x := a]^\ell) &= \{\mathtt{x}\} \\
kill_{\mathrm{LV}}([x \mathrel{?=} \rho]^\ell) &= \{\mathtt{x}\} \\
kill_{\mathrm{LV}}([\mathtt{skip}]^\ell) &= \emptyset \\
kill_{\mathrm{LV}}([b]^\ell) &= \emptyset
\end{aligned}
$$

$$f_\ell^{LV} : \mathcal{P}(\mathbf{Var}_\star) \rightarrow \mathcal{P}(\mathbf{Var}_\star)$$

$$f_\ell^{LV}(X) = X \setminus kill_{\mathrm{LV}}([B]^\ell) \cup gen_{\mathrm{LV}}([B]^\ell)$$

Probabilistic Analysis

In the classical analysis the undecidability of predicates in tests leads us to consider a conservative approach: Everything is possible, i.e. tests are treated as non-deterministic choices in the control flow.

In a probabilistic analysis we aim instead in providing good (optimal) estimates for branch(ing) probabilities when we construct the probabilistic control flow.

Probabilistic Analysis

In the classical analysis the undecidability of predicates in tests leads us to consider a conservative approach: Everything is possible, i.e. tests are treated as non-deterministic choices in the control flow.

In a probabilistic analysis we aim instead in providing good (optimal) estimates for branch(ing) probabilities when we construct the probabilistic control flow.

Example

Consider, for example, instead of

$$[x := 1]^1;$$
$$[y := 2]^2;$$
$$[x := x + y \bmod 4]^3;$$
$$\texttt{if } [x > 2]^4 \texttt{ then } [z := x]^5 \texttt{ else } [z := y]^6 \texttt{ fi}$$

a probabilistic program like:

$$[x \mathrel{?=} \{0, 1\}]^1;$$
$$[y \mathrel{?=} \{0, 1, 2, 3\}]^2;$$
$$[x := x + y \bmod 4]^3;$$
$$\texttt{if } [x > 2]^4 \texttt{ then } [z := x]^5 \texttt{ else } [z := y]^6 \texttt{ fi}$$

Example

Consider, for example, instead of

$$[x := 1]^1;$$
$$[y := 2]^2;$$
$$[x := x + y \text{ mod } 4]^3;$$
$$\texttt{if } [x > 2]^4 \texttt{ then } [z := x]^5 \texttt{ else } [z := y]^6 \texttt{ fi}$$

a probabilistic program like:

$$[x \;?{=}\; \{0, 1\}]^1;$$
$$[y \;?{=}\; \{0, 1, 2, 3\}]^2;$$
$$[x := x + y \text{ mod } 4]^3;$$
$$\texttt{if } [x > 2]^4 \texttt{ then } [z := x]^5 \texttt{ else } [z := y]^6 \texttt{ fi}$$

Probabilistic Control Flow and Equations

We can also use the classical control flow relation (as long as we do not consider a randomised `choose` statement).

However, we can't use the same equations, because:

(i) We want to express probabilities of properties not just (safe approximations) of properties.

(ii) We also need to consider relational aspects, i.e. correlations e.g. between the sign of variables.

(iii) We would like/need to estimate the branching probabilities when tests are evaluated.

(iv) We often also need probabilistic versions of the transfer functions.

Probabilistic Control Flow and Equations

We can also use the classical control flow relation (as long as we do not consider a randomised `choose` statement).

However, we can't use the same equations, because:

(i) We want to express probabilities of properties not just (safe approximations) of properties.

(ii) We also need to consider relational aspects, i.e. correlations e.g. between the sign of variables.

(iii) We would like/need to estimate the branching probabilities when tests are evaluated.

(iv) We often also need probabilistic versions of the transfer functions.

Probabilistic Control Flow and Equations

We can also use the classical control flow relation (as long as we do not consider a randomised `choose` statement).

However, we can't use the same equations, because:

(i) We want to express probabilities of properties not just (safe approximations) of properties.

(ii) We also need to consider relational aspects, i.e. correlations e.g. between the sign of variables.

(iii) We would like/need to estimate the branching probabilities when tests are evaluated.

(iv) We often also need probabilistic versions of the transfer functions.

Probabilistic Control Flow and Equations

We can also use the classical control flow relation (as long as we do not consider a randomised `choose` statement).

However, we can't use the same equations, because:

(i) We want to express probabilities of properties not just (safe approximations) of properties.

(ii) We also need to consider relational aspects, i.e. correlations e.g. between the sign of variables.

(iii) We would like/need to estimate the branching probabilities when tests are evaluated.

(iv) We often also need probabilistic versions of the transfer functions.

Probabilistic Control Flow and Equations

We can also use the classical control flow relation (as long as we do not consider a randomised `choose` statement).

However, we can't use the same equations, because:

(i) We want to express probabilities of properties not just (safe approximations) of properties.

(ii) We also need to consider relational aspects, i.e. correlations e.g. between the sign of variables.

(iii) We would like/need to estimate the branching probabilities when tests are evaluated.

(iv) We often also need probabilistic versions of the transfer functions.

Probabilistic Control Flow and Equations

We can also use the classical control flow relation (as long as we do not consider a randomised `choose` statement).

However, we can't use the same equations, because:

(i) We want to express probabilities of properties not just (safe approximations) of properties.

(ii) We also need to consider relational aspects, i.e. correlations e.g. between the sign of variables.

(iii) We would like/need to estimate the branching probabilities when tests are evaluated.

(iv) We often also need probabilistic versions of the transfer functions.

Local Transfer

When we look at the local transfer functions f_ℓ then we now need some probabilistic version of these. For example: given probability distributions describing the values of x and y, what is the probability distribution describing possible values of $x + y \bmod 4$.

Possible ways to obtain probabilistic and abstract versions $f_\ell^\#$

- Construction of a corresponding operator.
- Abstraction of the concrete semantics.
- Testing and Profiling also give us estimates.

Local Transfer

When we look at the local transfer functions f_ℓ then we now need some probabilistic version of these. For example: given probability distributions describing the values of x and y, what is the probability distribution describing possible values of $x + y$ mod 4.

Possible ways to obtain probabilistic and abstract versions $f_\ell^\#$

- Construction of a corresponding operator.
- Abstraction of the concrete semantics.
- Testing and Profiling also give us estimates.

Local Transfer

When we look at the local transfer functions f_ℓ then we now need some probabilistic version of these. For example: given probability distributions describing the values of x and y, what is the probability distribution describing possible values of $x + y$ mod 4.

Possible ways to obtain probabilistic and abstract versions $f_\ell^\#$

- Construction of a corresponding operator.
- Abstraction of the concrete semantics.
- Testing and Profiling also give us estimates.

Local Transfer

When we look at the local transfer functions f_ℓ then we now need some probabilistic version of these. For example: given probability distributions describing the values of x and y, what is the probability distribution describing possible values of $x + y$ mod 4.

Possible ways to obtain probabilistic and abstract versions $f_\ell^\#$

- Construction of a corresponding operator.
- Abstraction of the concrete semantics.
- Testing and Profiling also give us estimates.

Local Transfer

When we look at the local transfer functions f_ℓ then we now need some probabilistic version of these. For example: given probability distributions describing the values of x and y, what is the probability distribution describing possible values of $x + y$ mod 4.

Possible ways to obtain probabilistic and abstract versions $f_\ell^\#$

- Construction of a corresponding operator.
- Abstraction of the concrete semantics.
- Testing and Profiling also give us estimates.

Probabilistic Abstract Interpretation

For an abstraction $\mathbf{A} : \mathcal{V}(\textbf{State}) \rightarrow \mathcal{V}(L)$ we get for a concrete transfer operator $\mathbf{F}$ an abstract, (least-square) optimal estimate via $\mathbf{F}^{\#} = \mathbf{A}^{\dagger}\mathbf{F}\mathbf{A}$ in analogy to Abstract Interpretation.

Definition

Let $\mathcal{C}$ and $\mathcal{D}$ be two Hilbert spaces and $\mathbf{A} : \mathcal{C} \rightarrow \mathcal{D}$ a bounded linear map. A bounded linear map $\mathbf{A}^{\dagger} = \mathbf{G} : \mathcal{D} \rightarrow \mathcal{C}$ 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 $\mathbf{G}$.

Probabilistic Abstract Interpretation

For an abstraction $\mathbf{A} : \mathcal{V}(\mathbf{State}) \rightarrow \mathcal{V}(L)$ we get for a concrete transfer operator $\mathbf{F}$ an abstract, (least-square) optimal estimate via $\mathbf{F}^{\#} = \mathbf{A}^{\dagger}\mathbf{F}\mathbf{A}$ in analogy to Abstract Interpretation.

Definition

Let $\mathcal{C}$ and $\mathcal{D}$ be two Hilbert spaces and $\mathbf{A} : \mathcal{C} \rightarrow \mathcal{D}$ a bounded linear map. A bounded linear map $\mathbf{A}^{\dagger} = \mathbf{G} : \mathcal{D} \rightarrow \mathcal{C}$ 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 $\mathbf{G}$.

Branch Probabilities

Definition

Given a program S_ℓ with $init(S_\ell) = \ell$ and a probability distribution ρ on **State**, the probability $p_{\ell,\ell'}(\rho)$ that the control is flowing from ℓ to ℓ' is defined as:

$$p_{\ell,\ell'}(\rho) = \sum_s \left\{ p \cdot \rho(s) \mid \exists s' \text{ s.t. } \langle S_\ell, s \rangle \Rightarrow_p \left\langle S_{\ell'}, s' \right\rangle \right\}.$$

The branch probabilities thus also depend on an initial distribution, even for deterministic programs.

One can implement the test b as projections $\mathbf{P}(b)$ which filter out states which do not pass the test.

Branch Probabilities

Definition

Given a program S_ℓ with $init(S_\ell) = \ell$ and a probability distribution ρ on **State**, the probability $p_{\ell,\ell'}(\rho)$ that the control is flowing from ℓ to ℓ' is defined as:

$$p_{\ell,\ell'}(\rho) = \sum_s \{p \cdot \rho(s) \mid \exists s' \text{ s.t. } \langle S_\ell, s\rangle \Rightarrow_p \langle S_{\ell'}, s'\rangle\}.$$

The branch probabilities thus also depend on an initial distribution, even for deterministic programs.

One can implement the test b as projections $\mathbf{P}(b)$ which filter out states which do not pass the test.

Branch Probabilities

Definition

Given a program S_ℓ with $init(S_\ell) = \ell$ and a probability distribution ρ on **State**, the probability $p_{\ell,\ell'}(\rho)$ that the control is flowing from ℓ to ℓ' is defined as:

$$p_{\ell,\ell'}(\rho) = \sum_s \{p \cdot \rho(s) \mid \exists s' \text{ s.t. } \langle S_\ell, s\rangle \Rightarrow_p \langle S_{\ell'}, s'\rangle\} .$$

The branch probabilities thus also depend on an initial distribution, even for deterministic programs.

One can implement the test b as projections $\mathbf{P}(b)$ which filter out states which do not pass the test.

Tests and Branch Probabilities (Concrete)

Consider the simple program with $x \in \{0, 1, 2\}$

$$\texttt{if } [x >= 1]^1 \texttt{ then } [x := x - 1]^2 \texttt{ else } [\texttt{skip}]^3 \texttt{ fi}$$

Then the test $b = (x >= 1)$ is represented by the projection:

$$\mathbf{P}(x >= 1) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{ and } \mathbf{P}(x >= 1)^{\perp} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

For $\rho = \{\langle 0, p_0 \rangle, \langle 1, p_1 \rangle, \langle 2, p_2 \rangle\} = (p_0, p_1, p_2)$ we can compute the branch(ing) probabilities as $\rho\mathbf{P}(x >= 1) = (0, p_1, p_2)$ and

$$p_{1,2}(\rho) = \|\rho \cdot \mathbf{P}(x >= 1)\|_1 = p_1 + p_2,$$

for the `else` branch, with $\mathbf{P}^{\perp} = \mathbf{I} - \mathbf{P}$:

$$p_{1,3}(\rho) = \|\rho \cdot \mathbf{P}^{\perp}(x >= 1)\|_1 = p_0.$$

Tests and Branch Probabilities (Concrete)

Consider the simple program with $x \in \{0, 1, 2\}$

$$\texttt{if } [x >= 1]^1 \texttt{ then } [x := x - 1]^2 \texttt{ else } [\texttt{skip}]^3 \texttt{ fi}$$

Then the test $b = (x >= 1)$ is represented by the projection:

$$\mathbf{P}(x >= 1) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{ and } \mathbf{P}(x >= 1)^{\perp} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

For $\rho = \{\langle 0, p_0 \rangle, \langle 1, p_1 \rangle, \langle 2, p_2 \rangle\} = (p_0, p_1, p_2)$ we can compute the branch(ing) probabilities as $\rho\mathbf{P}(x >= 1) = (0, p_1, p_2)$ and

$$p_{1,2}(\rho) = \|\rho \cdot \mathbf{P}(x >= 1)\|_1 = p_1 + p_2,$$

for the `else` branch, with $\mathbf{P}^{\perp} = \mathbf{I} - \mathbf{P}$:

$$p_{1,3}(\rho) = \|\rho \cdot \mathbf{P}^{\perp}(x >= 1)\|_1 = p_0.$$

Tests and Branch Probabilities (Concrete)

Consider the simple program with $x \in \{0, 1, 2\}$

$$\texttt{if } [x >= 1]^1 \texttt{ then } [x := x - 1]^2 \texttt{ else } [\texttt{skip}]^3 \texttt{ fi}$$

Then the test $b = (x >= 1)$ is represented by the projection:

$$\mathbf{P}(x >= 1) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{ and } \mathbf{P}(x >= 1)^\perp = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

For $\rho = \{\langle 0, p_0\rangle, \langle 1, p_1\rangle, \langle 2, p_2\rangle\} = (p_0, p_1, p_2)$ we can compute the branch(ing) probabilities as $\rho\mathbf{P}(x >= 1) = (0, p_1, p_2)$ and

$$p_{1,2}(\rho) = \|\rho \cdot \mathbf{P}(x >= 1)\|_1 = p_1 + p_2,$$

for the `else` branch, with $\mathbf{P}^\perp = \mathbf{I} - \mathbf{P}$:

$$p_{1,3}(\rho) = \|\rho \cdot \mathbf{P}^\perp(x >= 1)\|_1 = p_0.$$

Tests and Branch Probabilities (Concrete)

Consider the simple program with $x \in \{0, 1, 2\}$

$$\texttt{if } [x >= 1]^1 \texttt{ then } [x := x - 1]^2 \texttt{ else } [\texttt{skip}]^3 \texttt{ fi}$$

Then the test $b = (x >= 1)$ is represented by the projection:

$$\mathbf{P}(x >= 1) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{ and } \mathbf{P}(x >= 1)^{\perp} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

For $\rho = \{\langle 0, p_0 \rangle, \langle 1, p_1 \rangle, \langle 2, p_2 \rangle\} = (p_0, p_1, p_2)$ we can compute the branch(ing) probabilities as $\rho\mathbf{P}(x >= 1) = (0, p_1, p_2)$ and

$$p_{1,2}(\rho) = \|\rho \cdot \mathbf{P}(x >= 1)\|_1 = p_1 + p_2,$$

for the `else` branch, with $\mathbf{P}^{\perp} = \mathbf{I} - \mathbf{P}$:

$$p_{1,3}(\rho) = \|\rho \cdot \mathbf{P}^{\perp}(x >= 1)\|_1 = p_0.$$

Abstract Branch Probabilities

If we consider abstract states $\rho^{\#} \in \mathcal{V}(L)$ we need abstract versions $\mathbf{P}(b)^{\#}$ of $\mathbf{P}(b)$ to compute the branch probabilities.
In doing so we must guarantee that for $\rho^{\#} = \rho\mathbf{A}$:

$$\begin{aligned} \rho\mathbf{P}(b)\mathbf{A} &\stackrel{!}{=} \rho^{\#}\mathbf{P}^{\#}(b) \\ \rho\mathbf{P}(b)\mathbf{A} &\stackrel{!}{=} \rho\mathbf{A}\mathbf{P}^{\#}(b) \\ \mathbf{P}(b)\mathbf{A} &\stackrel{!}{=} \mathbf{A}\mathbf{P}^{\#}(b) \end{aligned}$$

Ideally, to get $\mathbf{P}^{\#}$ if we multiply the last equation from the left with $\mathbf{A}^{-1}$. However, $\mathbf{A}$ is in general not not invertible.
The optimal (least-square) estimate can be obtained via

$$\begin{aligned} \mathbf{A}^{\dagger}\mathbf{P}(b)\mathbf{A} &= \mathbf{A}^{\dagger}\mathbf{A}\mathbf{P}^{\#}(b) \\ \mathbf{A}^{\dagger}\mathbf{P}(b)\mathbf{A} &= \mathbf{P}^{\#}(b) \end{aligned}$$

We get estimates for the abstract branch probabilities.

Abstract Branch Probabilities

If we consider abstract states $\rho^\# \in \mathcal{V}(L)$ we need abstract versions $\mathbf{P}(b)^\#$ of $\mathbf{P}(b)$ to compute the branch probabilities. In doing so we must guarantee that for $\rho^\# = \rho\mathbf{A}$:

$$
\begin{aligned}
\rho\mathbf{P}(b)\mathbf{A} &\stackrel{!}{=} \rho^\#\mathbf{P}^\#(b) \\
\rho\mathbf{P}(b)\mathbf{A} &\stackrel{!}{=} \rho\mathbf{A}\mathbf{P}^\#(b) \\
\mathbf{P}(b)\mathbf{A} &\stackrel{!}{=} \mathbf{A}\mathbf{P}^\#(b)
\end{aligned}
$$

Ideally, to get $\mathbf{P}^\#$ if we multiply the last equation from the left with $\mathbf{A}^{-1}$. However, $\mathbf{A}$ is in general not not invertible. The optimal (least-square) estimate can be obtained via

$$
\begin{aligned}
\mathbf{A}^\dagger\mathbf{P}(b)\mathbf{A} &= \mathbf{A}^\dagger\mathbf{A}\mathbf{P}^\#(b) \\
\mathbf{A}^\dagger\mathbf{P}(b)\mathbf{A} &= \mathbf{P}^\#(b)
\end{aligned}
$$

We get estimates for the abstract branch probabilities.

Abstract Branch Probabilities

If we consider abstract states $\rho^\# \in \mathcal{V}(L)$ we need abstract versions $\mathbf{P}(b)^\#$ of $\mathbf{P}(b)$ to compute the branch probabilities. In doing so we must guarantee that for $\rho^\# = \rho\mathbf{A}$:

$$\begin{aligned} \rho\mathbf{P}(b)\mathbf{A} &\stackrel{!}{=} \rho^\#\mathbf{P}^\#(b) \\ \rho\mathbf{P}(b)\mathbf{A} &\stackrel{!}{=} \rho\mathbf{A}\mathbf{P}^\#(b) \\ \mathbf{P}(b)\mathbf{A} &\stackrel{!}{=} \mathbf{A}\mathbf{P}^\#(b) \end{aligned}$$

Ideally, to get $\mathbf{P}^\#$ if we multiply the last equation from the left with $\mathbf{A}^{-1}$. However, $\mathbf{A}$ is in general not not invertible.
The optimal (least-square) estimate can be obtained via

$$\begin{aligned} \mathbf{A}^\dagger\mathbf{P}(b)\mathbf{A} &= \mathbf{A}^\dagger\mathbf{A}\mathbf{P}^\#(b) \\ \mathbf{A}^\dagger\mathbf{P}(b)\mathbf{A} &= \mathbf{P}^\#(b) \end{aligned}$$

We get estimates for the abstract branch probabilities.

Abstract Branch Probabilities

If we consider abstract states $\rho^\# \in \mathcal{V}(L)$ we need abstract versions $\mathbf{P}(b)^\#$ of $\mathbf{P}(b)$ to compute the branch probabilities. In doing so we must guarantee that for $\rho^\# = \rho\mathbf{A}$:

$$\begin{aligned} \rho\mathbf{P}(b)\mathbf{A} &\stackrel{!}{=} \rho^\#\mathbf{P}^\#(b) \\ \rho\mathbf{P}(b)\mathbf{A} &\stackrel{!}{=} \rho\mathbf{A}\mathbf{P}^\#(b) \\ \mathbf{P}(b)\mathbf{A} &\stackrel{!}{=} \mathbf{A}\mathbf{P}^\#(b) \end{aligned}$$

Ideally, to get $\mathbf{P}^\#$ if we multiply the last equation from the left with $\mathbf{A}^{-1}$. However, $\mathbf{A}$ is in general not not invertible. The optimal (least-square) estimate can be obtained via

$$\begin{aligned} \mathbf{A}^\dagger\mathbf{P}(b)\mathbf{A} &= \mathbf{A}^\dagger\mathbf{A}\mathbf{P}^\#(b) \\ \mathbf{A}^\dagger\mathbf{P}(b)\mathbf{A} &= \mathbf{P}^\#(b) \end{aligned}$$

We get estimates for the abstract branch probabilities.

Abstract Branch Probabilities

If we consider abstract states $\rho^\# \in \mathcal{V}(L)$ we need abstract versions $\mathbf{P}(b)^\#$ of $\mathbf{P}(b)$ to compute the branch probabilities. In doing so we must guarantee that for $\rho^\# = \rho\mathbf{A}$:

$$\begin{aligned} \rho\mathbf{P}(b)\mathbf{A} &\stackrel{!}{=} \rho^\#\mathbf{P}^\#(b) \\ \rho\mathbf{P}(b)\mathbf{A} &\stackrel{!}{=} \rho\mathbf{A}\mathbf{P}^\#(b) \\ \mathbf{P}(b)\mathbf{A} &\stackrel{!}{=} \mathbf{A}\mathbf{P}^\#(b) \end{aligned}$$

Ideally, to get $\mathbf{P}^\#$ if we multiply the last equation from the left with $\mathbf{A}^{-1}$. However, $\mathbf{A}$ is in general not not invertible. The optimal (least-square) estimate can be obtained via

$$\begin{aligned} \mathbf{A}^\dagger\mathbf{P}(b)\mathbf{A} &= \mathbf{A}^\dagger\mathbf{A}\mathbf{P}^\#(b) \\ \mathbf{A}^\dagger\mathbf{P}(b)\mathbf{A} &= \mathbf{P}^\#(b) \end{aligned}$$

We get estimates for the abstract branch probabilities.

An Example: Prime Numbers are Odd

Consider the following program that counts the prime numbers.

$[i := 2]^1$;
while $[i < 100]^2$ do
if $[prime(i)]^3$ then $[p := p + 1]^4$
else [skip]^5 fi;
$[i := i + 1]^6$
od

Essential is the abstract branch probability for $[.]^3$:

$$\mathbf{P}(prime(i))^{\#} = \mathbf{A}_e^{\dagger}\mathbf{P}(prime(i))\mathbf{A}_e,$$

An Example: Prime Numbers are Odd

Consider the following program that counts the prime numbers.

$[i := 2]^1;$
$\texttt{while}\ [i < 100]^2\ \texttt{do}$
$\texttt{if}\ [prime(i)]^3\ \texttt{then}\ [p := p + 1]^4$
$\texttt{else}\ [\texttt{skip}]^5\ \texttt{fi};$
$[i := i + 1]^6$
$\texttt{od}$

Essential is the abstract branch probability for $[.]^3$:

$$\mathbf{P}(prime(i))^{\#} = \mathbf{A}_e^{\dagger}\mathbf{P}(prime(i))\mathbf{A}_e,$$

An Example: Abstraction

Test operators:

$$\mathbf{P}_e = (\mathbf{P}(\text{even}(n)))_{ii} = \begin{cases} 1 & \text{if } i = 2k \\ 0 & \textit{otherwise} \end{cases}$$

$$\mathbf{P}_p = (\mathbf{P}(\text{prime}(n)))_{ii} = \begin{cases} 1 & \text{if prime}(i) \\ 0 & \textit{otherwise} \end{cases}$$

Abstraction Operators:

$$(\mathbf{A}_e)_{ij} = \begin{cases} 1 & \text{if } i = 2k+1 \ \wedge \ j = 2 \\ 1 & \text{if } i = 2k \ \wedge \ j = 1 \\ 0 & \textit{otherwise} \end{cases}$$

$$(\mathbf{A}_p)_{ij} = \begin{cases} 1 & \text{if prime}(i) \ \wedge \ j = 2 \\ 1 & \text{if } \neg\text{prime}(i) \ \wedge \ j = 1 \\ 0 & \textit{otherwise} \end{cases}$$

An Example: Abstraction

Test operators:

$$\mathbf{P}_e = (\mathbf{P}(\text{even}(n)))_{ii} = \begin{cases} 1 & \text{if } i = 2k \\ 0 & \textit{otherwise} \end{cases}$$

$$\mathbf{P}_p = (\mathbf{P}(\text{prime}(n)))_{ii} = \begin{cases} 1 & \text{if prime}(i) \\ 0 & \textit{otherwise} \end{cases}$$

Abstraction Operators:

$$(\mathbf{A}_e)_{ij} = \begin{cases} 1 & \text{if } i = 2k+1 \ \wedge \ j = 2 \\ 1 & \text{if } i = 2k \ \wedge \ j = 1 \\ 0 & \textit{otherwise} \end{cases}$$

$$(\mathbf{A}_p)_{ij} = \begin{cases} 1 & \text{if prime}(i) \ \wedge \ j = 2 \\ 1 & \text{if } \neg\text{prime}(i) \ \wedge \ j = 1 \\ 0 & \textit{otherwise} \end{cases}$$

An Example: Abstract Branch Probability

For ranges $[0, \ldots, n]$ we get:

	$\mathbf{A}_e^\dagger \mathbf{P}_p \mathbf{A}_e$	$\mathbf{A}_e^\dagger \mathbf{P}_p^\perp \mathbf{A}_e$	$\mathbf{A}_p^\dagger \mathbf{P}_e \mathbf{A}_p$	$\mathbf{A}_p^\dagger \mathbf{P}_e^\perp \mathbf{A}_p$
$n = 10$	$\begin{pmatrix} 0.20 & 0.00 \\ 0.00 & 0.60 \end{pmatrix}$	$\begin{pmatrix} 0.80 & 0.00 \\ 0.00 & 0.40 \end{pmatrix}$	$\begin{pmatrix} 0.25 & 0.00 \\ 0.00 & 0.67 \end{pmatrix}$	$\begin{pmatrix} 0.75 & 0.00 \\ 0.00 & 0.33 \end{pmatrix}$
$n = 100$	$\begin{pmatrix} 0.02 & 0.00 \\ 0.00 & 0.48 \end{pmatrix}$	$\begin{pmatrix} 0.98 & 0.00 \\ 0.00 & 0.52 \end{pmatrix}$	$\begin{pmatrix} 0.04 & 0.00 \\ 0.00 & 0.65 \end{pmatrix}$	$\begin{pmatrix} 0.96 & 0.00 \\ 0.00 & 0.35 \end{pmatrix}$
$n = 1000$	$\begin{pmatrix} 0.00 & 0.00 \\ 0.00 & 0.33 \end{pmatrix}$	$\begin{pmatrix} 1.00 & 0.00 \\ 0.00 & 0.67 \end{pmatrix}$	$\begin{pmatrix} 0.01 & 0.00 \\ 0.00 & 0.60 \end{pmatrix}$	$\begin{pmatrix} 0.99 & 0.00 \\ 0.00 & 0.40 \end{pmatrix}$
$n = 10000$	$\begin{pmatrix} 0.00 & 0.00 \\ 0.00 & 0.25 \end{pmatrix}$	$\begin{pmatrix} 1.00 & 0.00 \\ 0.00 & 0.75 \end{pmatrix}$	$\begin{pmatrix} 0.00 & 0.00 \\ 0.00 & 0.57 \end{pmatrix}$	$\begin{pmatrix} 1.00 & 0.00 \\ 0.00 & 0.43 \end{pmatrix}$

The entries in the upper left corner of $\mathbf{A}_e^\dagger \mathbf{P}_p \mathbf{A}_e$ give us the chances that an even number is also a prime number, etc.

Note that the positive and negative matrices always add up to $\mathbf{I}$.

Probabilistic Dataflow Equations

Similar to classical DFA we formulate linear equations:

$$\begin{aligned} Analysis_\bullet(\ell) &= Analysis_\circ(\ell) \cdot \mathbf{F}_\ell^\# \\ Analysis_\circ(\ell) &= \begin{cases} \iota, \text{if } \ell \in E \\ \sum\{Analysis_\bullet(\ell') \cdot \mathbf{P}(\ell',\ell)^\# \mid (\ell',\ell) \in F\}, \text{else} \end{cases} \end{aligned}$$

A simpler version can be obtained by static branch prediction:

$$Analysis_\circ(\ell) = \sum\{p_{\ell',\ell} \cdot Analysis_\bullet(\ell') \mid (\ell',\ell) \in F\}$$

Abstract branch probabilities, i.e. estimates for the test operators $\mathbf{P}(\ell',\ell)^\#$, can be estimated also via a different analysis Prob, in a first phase before the actual Analysis.

Probabilistic Dataflow Equations

Similar to classical DFA we formulate linear equations:

$$
\begin{aligned}
\mathit{Analysis}_\bullet(\ell) &= \mathit{Analysis}_\circ(\ell) \cdot \mathbf{F}_\ell^\# \\
\mathit{Analysis}_\circ(\ell) &= \begin{cases} \iota, \text{if } \ell \in E \\ \sum\{\mathit{Analysis}_\bullet(\ell') \cdot \mathbf{P}(\ell', \ell)^\# \mid (\ell', \ell) \in F\}, \text{else} \end{cases}
\end{aligned}
$$

A simpler version can be obtained by static branch prediction:

$$
\mathit{Analysis}_\circ(\ell) = \sum\{p_{\ell',\ell} \cdot \mathit{Analysis}_\bullet(\ell') \mid (\ell', \ell) \in F\}
$$

Abstract branch probabilities, i.e. estimates for the test operators $\mathbf{P}(\ell', \ell)^\#$, can be estimated also via a different analysis Prob, in a first phase before the actual Analysis.

Probabilistic Dataflow Equations

Similar to classical DFA we formulate linear equations:

$$\begin{aligned} \mathit{Analysis}_\bullet(\ell) &= \mathit{Analysis}_\circ(\ell) \cdot \mathbf{F}_\ell^{\#} \\ \mathit{Analysis}_\circ(\ell) &= \begin{cases} \iota, \text{if } \ell \in E \\ \sum\{\mathit{Analysis}_\bullet(\ell') \cdot \mathbf{P}(\ell',\ell)^{\#} \mid (\ell',\ell) \in F\}, \text{else} \end{cases} \end{aligned}$$

A simpler version can be obtained by static branch prediction:

$$\mathit{Analysis}_\circ(\ell) = \sum\{p_{\ell',\ell} \cdot \mathit{Analysis}_\bullet(\ell') \mid (\ell',\ell) \in F\}$$

Abstract branch probabilities, i.e. estimates for the test operators $\mathbf{P}(\ell',\ell)^{\#}$, can be estimated also via a different analysis Prob, in a first phase before the actual Analysis.

Live Variable Analysis: Example

Coming back to our previous example and its *LV* analysis:

$$[x \;\texttt{?=}\; \{0,1\}]^1;\; [y \;\texttt{?=}\; \{0,1,2,3\}]^2;\; [x \;\texttt{:=}\; x + y \bmod 4]^3;$$
$$\texttt{if}\; [x > 2]^4 \;\texttt{then}\; [z \;\texttt{:=}\; x]^5 \;\texttt{else}\; [z \;\texttt{:=}\; y]^6 \;\texttt{fi}$$

Consider two properties *d* for ‘dead’, and *l* for ‘live’ and the space $\mathcal{V}(\{0,1\}) = \mathcal{V}(\{d,l\}) = \mathbb{R}^2$ as the property space.

$$\mathbf{L} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{K} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}.$$

We define the abstract transfers for our four blocks a

$$\mathbf{F}_\ell = \mathbf{F}_\ell^{LV} : \mathcal{V}(\{0,1\})^{\otimes|\mathbf{Var}|} \to \mathcal{V}(\{0,1\})^{\otimes|\mathbf{Var}|}$$

Live Variable Analysis: Example

Coming back to our previous example and its *LV* analysis:

$[x \texttt{ ?= } \{0,1\}]^1;\ [y \texttt{ ?= } \{0,1,2,3\}]^2;\ [x \texttt{ := } x + y \bmod 4]^3;$
$\texttt{if } [x > 2]^4 \texttt{ then } [z \texttt{ := } x]^5 \texttt{ else } [z \texttt{ := } y]^6 \texttt{ fi}$

Consider two properties *d* for 'dead', and *l* for 'live' and the space $\mathcal{V}(\{0,1\}) = \mathcal{V}(\{d,l\}) = \mathbb{R}^2$ as the property space.

$$\mathbf{L} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \text{ and } \mathbf{K} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}.$$

We define the abstract transfers for our four blocks a

$$\mathbf{F}_\ell = \mathbf{F}_\ell^{LV} : \mathcal{V}(\{0,1\})^{\otimes|\mathbf{Var}|} \to \mathcal{V}(\{0,1\})^{\otimes|\mathbf{Var}|}$$

Live Variable Analysis: Example

Coming back to our previous example and its *LV* analysis:

$$[x \texttt{ ?= } \{0,1\}]^1;\ [y \texttt{ ?= } \{0,1,2,3\}]^2;\ [x \texttt{ := } x + y \bmod 4]^3;$$
$$\texttt{if } [x > 2]^4 \texttt{ then } [z \texttt{ := } x]^5 \texttt{ else } [z \texttt{ := } y]^6 \texttt{ fi}$$

Consider two properties d for 'dead', and l for 'live' and the space $\mathcal{V}(\{0,1\}) = \mathcal{V}(\{d,l\}) = \mathbb{R}^2$ as the property space.

$$\mathbf{L} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{K} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}.$$

We define the abstract transfers for our four blocks a

$$\mathbf{F}_\ell = \mathbf{F}_\ell^{LV} : \mathcal{V}(\{0,1\})^{\otimes|\mathbf{Var}|} \to \mathcal{V}(\{0,1\})^{\otimes|\mathbf{Var}|}$$

Live Variable Analysis: Example

Coming back to our previous example and its *LV* analysis:

$[x \texttt{ ?= } \{0,1\}]^1;\ [y \texttt{ ?= } \{0,1,2,3\}]^2;\ [x \texttt{ := } x + y \bmod 4]^3;$
$\texttt{if } [x > 2]^4 \texttt{ then } [z \texttt{ := } x]^5 \texttt{ else } [z \texttt{ := } y]^6 \texttt{ fi}$

Consider two properties *d* for 'dead', and *l* for 'live' and the space $\mathcal{V}(\{0,1\}) = \mathcal{V}(\{d,l\}) = \mathbb{R}^2$ as the property space.

$$\mathbf{L} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{K} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}.$$

We define the abstract transfers for our four blocks a

$$\mathbf{F}_\ell = \mathbf{F}_\ell^{LV} : \mathcal{V}(\{0,1\})^{\otimes|\mathbf{Var}|} \to \mathcal{V}(\{0,1\})^{\otimes|\mathbf{Var}|}$$

Transfer Functions for Live Variables

For $[x := a]^\ell$ (with $\mathbf{I}$ the identity matrix)

$$\mathbf{F}_\ell = \bigotimes_{x_i \in \mathbf{Var}} \mathbf{X}_i \text{ with } \mathbf{X}_i = \begin{cases} \mathbf{L} & \text{if } x_i \in FV(a) \\ \mathbf{K} & \text{if } x_i = x \wedge x_i \notin FV(a) \\ \mathbf{I} & \text{otherwise.} \end{cases}$$

and for tests $[b]^\ell$

$$\mathbf{F}_\ell = \bigotimes_{x_i \in \mathbf{Var}} \mathbf{X}_i \text{ with } \mathbf{X}_i = \begin{cases} \mathbf{L} & \text{if } x_i \in FV(b) \\ \mathbf{I} & \text{otherwise.} \end{cases}$$

For $[\texttt{skip}]^\ell$ and $[x \texttt{ ?= } \rho]^\ell$ have $\mathbf{F}_\ell = \bigotimes_{x_i \in \mathbf{Var}} \mathbf{I}$.

Transfer Functions for Live Variables

For $[x := a]^\ell$ (with $\mathbf{I}$ the identity matrix)

$$\mathbf{F}_\ell = \bigotimes_{x_i \in \mathbf{Var}} \mathbf{X}_i \text{ with } \mathbf{X}_i = \begin{cases} \mathbf{L} & \text{if } x_i \in FV(a) \\ \mathbf{K} & \text{if } x_i = x \wedge x_i \notin FV(a) \\ \mathbf{I} & \text{otherwise.} \end{cases}$$

and for tests $[b]^\ell$

$$\mathbf{F}_\ell = \bigotimes_{x_i \in \mathbf{Var}} \mathbf{X}_i \text{ with } \mathbf{X}_i = \begin{cases} \mathbf{L} & \text{if } x_i \in FV(b) \\ \mathbf{I} & \text{otherwise.} \end{cases}$$

For $[\texttt{skip}]^\ell$ and $[x \texttt{ ?= } \rho]^\ell$ have $\mathbf{F}_\ell = \bigotimes_{x_i \in \mathbf{Var}} \mathbf{I}$.

Transfer Functions for Live Variables

For $[x := a]^\ell$ (with $\mathbf{I}$ the identity matrix)

$$\mathbf{F}_\ell = \bigotimes_{x_i \in \mathbf{Var}} \mathbf{X}_i \text{ with } \mathbf{X}_i = \begin{cases} \mathbf{L} & \text{if } x_i \in FV(a) \\ \mathbf{K} & \text{if } x_i = x \ \wedge \ x_i \notin FV(a) \\ \mathbf{I} & \text{otherwise.} \end{cases}$$

and for tests $[b]^\ell$

$$\mathbf{F}_\ell = \bigotimes_{x_i \in \mathbf{Var}} \mathbf{X}_i \text{ with } \mathbf{X}_i = \begin{cases} \mathbf{L} & \text{if } x_i \in FV(b) \\ \mathbf{I} & \text{otherwise.} \end{cases}$$

For $[\texttt{skip}]^\ell$ and $[x \texttt{ ?= } \rho]^\ell$ have $\mathbf{F}_\ell = \bigotimes_{x_i \in \mathbf{Var}} \mathbf{I}$.

Preprocessing

We present a *LV* analysis based essentially on concrete branch probabilities. That means that in the first phase of the analysis we will not abstract the values of x and y, we just ignore z all together.

If the concrete state of each variable is a value in $\{0,1,2,3\}$, then the probabilistic state is in $\mathcal{V}(\{0,1,2,3\})^{\otimes 3} = \mathbb{R}^{4^3} = \mathbb{R}^{64}$.

The abstraction we use when we compute the concrete branch probabilities is $\mathbf{A} = \mathbf{I} \otimes \mathbf{I} \otimes \mathbf{A}_f$, with $\mathbf{A}_f = (1,1,1,1)^t$ the forgetful abstraction, i.e. z is ignored. This allows us to reduce the dimensions of the probabilistic state space from 64 to just 16. Note that also $\mathbf{F}_5^{\#} = \mathbf{F}_6^{\#} = \mathbf{I}$.

Preprocessing

We present a *LV* analysis based essentially on concrete branch probabilities. That means that in the first phase of the analysis we will not abstract the values of x and y, we just ignore z all together.

If the concrete state of each variable is a value in $\{0, 1, 2, 3\}$, then the probabilistic state is in $\mathcal{V}(\{0, 1, 2, 3\})^{\otimes 3} = \mathbb{R}^{4^3} = \mathbb{R}^{64}$.

The abstraction we use when we compute the concrete branch probabilities is $\mathbf{A} = \mathbf{I} \otimes \mathbf{I} \otimes \mathbf{A}_f$, with $\mathbf{A}_f = (1, 1, 1, 1)^t$ the forgetful abstraction, i.e. z is ignored. This allows us to reduce the dimensions of the probabilistic state space from 64 to just 16. Note that also $\mathbf{F}_5^\# = \mathbf{F}_6^\# = \mathbf{I}$.

Preprocessing

We present a *LV* analysis based essentially on concrete branch probabilities. That means that in the first phase of the analysis we will not abstract the values of x and y, we just ignore z all together.

If the concrete state of each variable is a value in $\{0, 1, 2, 3\}$, then the probabilistic state is in $\mathcal{V}(\{0, 1, 2, 3\})^{\otimes 3} = \mathbb{R}^{4^3} = \mathbb{R}^{64}$.

The abstraction we use when we compute the concrete branch probabilities is $\mathbf{A} = \mathbf{I} \otimes \mathbf{I} \otimes \mathbf{A}_f$, with $\mathbf{A}_f = (1, 1, 1, 1)^t$ the forgetful abstraction, i.e. z is ignored. This allows us to reduce the dimensions of the probabilistic state space from 64 to just 16. Note that also $\mathbf{F}_5^\# = \mathbf{F}_6^\# = \mathbf{I}$.

(Abstract) Transfer Operators

$$
\mathbf{F}_1^{\#} = \begin{pmatrix}
\frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
$$

(Abstract) Transfer Operators

$$\mathbf{F}_2^{\#} = \begin{pmatrix}
\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4}
\end{pmatrix}$$

(Abstract) Transfer Operators

$$
\mathbf{F}_3^{\#} = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0
\end{pmatrix}
$$

(Abstract) Transfer Operators

$$\mathbf{P}_4^{\#} = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

Probability Equations

The pre-processing probability analysis via equations:

$$
\begin{aligned}
\text{Prob}_{entry}(1) &= \rho \\
\text{Prob}_{entry}(2) &= \text{Prob}_{exit}(1) \\
\text{Prob}_{entry}(3) &= \text{Prob}_{exit}(2) \\
\text{Prob}_{entry}(4) &= \text{Prob}_{exit}(3) \\
\text{Prob}_{entry}(5) &= \text{Prob}_{exit}(4) \cdot \mathbf{P}_4^{\#} \\
\text{Prob}_{entry}(6) &= \text{Prob}_{exit}(4) \cdot (\mathbf{I} - \mathbf{P}_4^{\#})
\end{aligned}
$$

Probability Equations

The pre-processing probability analysis via equations:

$$
\begin{aligned}
\mathsf{Prob}_{exit}(1) &= \mathsf{Prob}_{entry}(1) \cdot \mathbf{F}_1^{\#} \\
\mathsf{Prob}_{exit}(2) &= \mathsf{Prob}_{entry}(1) \cdot \mathbf{F}_2^{\#} \\
\mathsf{Prob}_{exit}(3) &= \mathsf{Prob}_{entry}(1) \cdot \mathbf{F}_3^{\#} \\
\mathsf{Prob}_{exit}(4) &= \mathsf{Prob}_{entry}(4) \\
\mathsf{Prob}_{exit}(5) &= \mathsf{Prob}_{entry}(5) \\
\mathsf{Prob}_{exit}(6) &= \mathsf{Prob}_{entry}(6)
\end{aligned}
$$

Probability Equations

The pre-processing probability analysis via equations:

$$\begin{aligned}
\text{Prob}_{exit}(1) &= \text{Prob}_{entry}(1) \cdot \mathbf{F}_1^{\#} \\
\text{Prob}_{exit}(2) &= \text{Prob}_{entry}(1) \cdot \mathbf{F}_2^{\#} \\
\text{Prob}_{exit}(3) &= \text{Prob}_{entry}(1) \cdot \mathbf{F}_3^{\#} \\
\text{Prob}_{exit}(4) &= \text{Prob}_{entry}(4) \\
\text{Prob}_{exit}(5) &= \text{Prob}_{entry}(5) \\
\text{Prob}_{exit}(6) &= \text{Prob}_{entry}(6)
\end{aligned}$$

reduce to:

$$\begin{aligned}
\text{Prob}_{entry}(5) &= \rho \cdot \mathbf{F}_1^{\#} \cdot \mathbf{F}_2^{\#} \cdot \mathbf{F}_3^{\#} \cdot \mathbf{P}_4^{\#} \\
\text{Prob}_{entry}(6) &= \rho \cdot \mathbf{F}_1^{\#} \cdot \mathbf{F}_2^{\#} \cdot \mathbf{F}_3^{\#} \cdot \mathbf{P}_4^{\#}
\end{aligned}$$

Probability Equations

The pre-processing probability analysis via equations:

$$
\begin{aligned}
\mathrm{Prob}_{exit}(1) &= \mathrm{Prob}_{entry}(1) \cdot \mathbf{F}_1^{\#} \\
\mathrm{Prob}_{exit}(2) &= \mathrm{Prob}_{entry}(1) \cdot \mathbf{F}_2^{\#} \\
\mathrm{Prob}_{exit}(3) &= \mathrm{Prob}_{entry}(1) \cdot \mathbf{F}_3^{\#} \\
\mathrm{Prob}_{exit}(4) &= \mathrm{Prob}_{entry}(4) \\
\mathrm{Prob}_{exit}(5) &= \mathrm{Prob}_{entry}(5) \\
\mathrm{Prob}_{exit}(6) &= \mathrm{Prob}_{entry}(6)
\end{aligned}
$$

reduce to:

$$
\begin{aligned}
\mathrm{Prob}_{entry}(5) &= \rho \cdot \mathbf{F}_1^{\#} \cdot \mathbf{F}_2^{\#} \cdot \mathbf{F}_3^{\#} \cdot \mathbf{P}_4^{\#} \\
\mathrm{Prob}_{entry}(6) &= \rho \cdot \mathbf{F}_1^{\#} \cdot \mathbf{F}_2^{\#} \cdot \mathbf{F}_3^{\#} \cdot \mathbf{P}_4^{\#}
\end{aligned}
$$

We thus have for any ρ that $p_{4,5}(\rho) = \|\mathrm{Prob}_{entry}(5)\|_1 = \frac{1}{4}$ and $p_{4,6}(\rho) = \|\mathrm{Prob}_{entry}(6)\|_1 = \frac{3}{4}$.

Data Flow Equations

With this information we can formulate the actual *LV* equations:

$$
\begin{aligned}
\mathrm{LV}_{entry}(1) &= \mathrm{LV}_{exit}(1) \cdot (\mathbf{K} \otimes \mathbf{I} \otimes \mathbf{I}) \\
\mathrm{LV}_{entry}(2) &= \mathrm{LV}_{exit}(2) \cdot (\mathbf{I} \otimes \mathbf{K} \otimes \mathbf{I}) \\
\mathrm{LV}_{entry}(3) &= \mathrm{LV}_{exit}(3) \cdot (\mathbf{L} \otimes \mathbf{L} \otimes \mathbf{I}) \\
\mathrm{LV}_{entry}(4) &= \mathrm{LV}_{exit}(4) \cdot (\mathbf{L} \otimes \mathbf{I} \otimes \mathbf{I}) \\
\mathrm{LV}_{entry}(5) &= \mathrm{LV}_{exit}(5) \cdot (\mathbf{L} \otimes \mathbf{I} \otimes \mathbf{K}) \\
\mathrm{LV}_{entry}(6) &= \mathrm{LV}_{exit}(6) \cdot (\mathbf{I} \otimes \mathbf{L} \otimes \mathbf{K})
\end{aligned}
$$

Data Flow Equations

With this information we can formulate the actual *LV* equations:

$$
\begin{aligned}
\mathrm{LV}_{exit}(1) &= \mathrm{LV}_{entry}(2) \\
\mathrm{LV}_{exit}(2) &= \mathrm{LV}_{entry}(3) \\
\mathrm{LV}_{exit}(3) &= \mathrm{LV}_{entry}(4) \\
\mathrm{LV}_{exit}(4) &= p_{4,5}\mathrm{LV}_{entry}(5) + p_{4,6}\mathrm{LV}_{entry}(6) \\
\mathrm{LV}_{exit}(5) &= (1,0) \otimes (1,0) \otimes (1,0) \\
\mathrm{LV}_{exit}(6) &= (1,0) \otimes (1,0) \otimes (1,0)
\end{aligned}
$$

Example: Solution

The solution to the *LV* equations is then given by:

$$
\begin{aligned}
\mathrm{LV}_{entry}(1) &= (1,0)\otimes(1,0)\otimes(1,0)\\
\mathrm{LV}_{entry}(2) &= (0,1)\otimes(1,0)\otimes(1,0)\\
\mathrm{LV}_{entry}(3) &= 0.25\cdot(0,1)\otimes(0,1)\otimes(1,0)+\\
&+ 0.75\cdot(0,1)\otimes(0,1)\otimes(1,0)\\
&= (0,1)\otimes(0,1)\otimes(1,0)\\
\mathrm{LV}_{entry}(4) &= 0.25\cdot(0,1)\otimes(1,0)\otimes(1,0)+\\
&+ 0.75\cdot(0,1)\otimes(0,1)\otimes(1,0)\\
\mathrm{LV}_{entry}(5) &= (0,1)\otimes(1,0)\otimes(1,0)\\
\mathrm{LV}_{entry}(6) &= (1,0)\otimes(0,1)\otimes(1,0)
\end{aligned}
$$

Example: Solution

The solution to the *LV* equations is then given by:

$$
\begin{aligned}
\mathrm{LV}_{exit}(1) &= (0,1) \otimes (1,0) \otimes (1,0) \\
\mathrm{LV}_{exit}(2) &= (0,1) \otimes (0,1) \otimes (1,0) \\
\mathrm{LV}_{exit}(3) &= 0.25 \cdot (0,1) \otimes (1,0) \otimes (1,0) + \\
&+ 0.75 \cdot (0,1) \otimes (0,1) \otimes (1,0) \\
\mathrm{LV}_{exit}(4) &= 0.25 \cdot (0,1) \otimes (1,0) \otimes (1,0) + \\
&+ 0.75 \cdot (1,0) \otimes (0,1) \otimes (1,0) \\
\mathrm{LV}_{exit}(5) &= (1,0) \otimes (1,0) \otimes (1,0) \\
\mathrm{LV}_{exit}(6) &= (1,0) \otimes (1,0) \otimes (1,0)
\end{aligned}
$$

The Moore-Penrose Pseudo-Inverse

Definition

Let $\mathcal{C}$ and $\mathcal{D}$ be two finite-dimensional vector spaces and $\mathbf{A} : \mathcal{C} \to \mathcal{D}$ a linear map. Then the linear map $\mathbf{A}^\dagger = \mathbf{G} : \mathcal{D} \to \mathcal{C}$ is the Moore-Penrose pseudo-inverse of $\mathbf{A}$ iff $\mathbf{A} \circ \mathbf{G} = \mathbf{P}_A$ and $\mathbf{G} \circ \mathbf{A} = \mathbf{P}_G$, where $\mathbf{P}_A$ and $\mathbf{P}_G$ denote orthogonal projections onto the ranges of $\mathbf{A}$ and $\mathbf{G}$.

Definition

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

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

Theorem

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

The Moore-Penrose Pseudo-Inverse

Definition

Let $\mathcal{C}$ and $\mathcal{D}$ be two finite-dimensional vector spaces and $\mathbf{A} : \mathcal{C} \to \mathcal{D}$ a linear map. Then the linear map $\mathbf{A}^\dagger = \mathbf{G} : \mathcal{D} \to \mathcal{C}$ is the Moore-Penrose pseudo-inverse of $\mathbf{A}$ iff $\mathbf{A} \circ \mathbf{G} = \mathbf{P}_A$ and $\mathbf{G} \circ \mathbf{A} = \mathbf{P}_G$, where $\mathbf{P}_A$ and $\mathbf{P}_G$ denote orthogonal projections onto the ranges of $\mathbf{A}$ and $\mathbf{G}$.

Definition

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

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

Theorem

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

The Moore-Penrose Pseudo-Inverse

Definition

Let $\mathcal{C}$ and $\mathcal{D}$ be two finite-dimensional vector spaces and $\mathbf{A} : \mathcal{C} \to \mathcal{D}$ a linear map. Then the linear map $\mathbf{A}^{\dagger} = \mathbf{G} : \mathcal{D} \to \mathcal{C}$ is the Moore-Penrose pseudo-inverse of $\mathbf{A}$ iff $\mathbf{A} \circ \mathbf{G} = \mathbf{P}_A$ and $\mathbf{G} \circ \mathbf{A} = \mathbf{P}_G$, where $\mathbf{P}_A$ and $\mathbf{P}_G$ denote orthogonal projections onto the ranges of $\mathbf{A}$ and $\mathbf{G}$.

Definition

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

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

Theorem

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

Probabilistic Abstract Interpretation

Probabilistic Abstract Interpretation is based on:

- Concrete and abstract domains are linear spaces $\mathcal{C}$, $\mathcal{D}$...
- Concrete and abstract semantics are linear operators $\mathbf{T}$...

The Moore-Penrose pseudo-inverse allows us to construct the closest (i.e. least square) approximation

$$\mathbf{T}^{\#} : \mathcal{D} \rightarrow \mathcal{D} \quad \text{of a concrete semantics} \quad \mathbf{T} : \mathcal{C} \rightarrow \mathcal{C}$$

which we define via the Moore-Penrose pseudo-inverse:

$$\mathbf{T}^{\#} = \mathbf{G} \cdot \mathbf{T} \cdot \mathbf{A} = \mathbf{A}^{\dagger} \cdot \mathbf{T} \cdot \mathbf{A} = \mathbf{A} \circ \mathbf{T} \circ \mathbf{G}.$$

This gives a “smaller” DTMC via the abstracted generator $\mathbf{T}^{\#}$.

Probabilistic Abstract Interpretation

Probabilistic Abstract Interpretation is based on:

- Concrete and abstract domains are linear spaces $\mathcal{C}$, $\mathcal{D}$...
- Concrete and abstract semantics are linear operators $\mathbf{T}$...

The Moore-Penrose pseudo-inverse allows us to construct the closest (i.e. least square) approximation

$$\mathbf{T}^{\#} : \mathcal{D} \to \mathcal{D} \text{ of a concrete semantics } \mathbf{T} : \mathcal{C} \to \mathcal{C}$$

which we define via the Moore-Penrose pseudo-inverse:

$$\mathbf{T}^{\#} = \mathbf{G} \cdot \mathbf{T} \cdot \mathbf{A} = \mathbf{A}^{\dagger} \cdot \mathbf{T} \cdot \mathbf{A} = \mathbf{A} \circ \mathbf{T} \circ \mathbf{G}.$$

This gives a “smaller” DTMC via the abstracted generator $\mathbf{T}^{\#}$.

Probabilistic Abstract Interpretation

Probabilistic Abstract Interpretation is based on:

- Concrete and abstract domains are linear spaces $\mathcal{C}$, $\mathcal{D}$...
- Concrete and abstract semantics are linear operators $\mathbf{T}$...

The Moore-Penrose pseudo-inverse allows us to construct the closest (i.e. least square) approximation

$\mathbf{T}^{\#} : \mathcal{D} \rightarrow \mathcal{D}$ of a concrete semantics $\mathbf{T} : \mathcal{C} \rightarrow \mathcal{C}$

which we define via the Moore-Penrose pseudo-inverse:

$$\mathbf{T}^{\#} = \mathbf{G} \cdot \mathbf{T} \cdot \mathbf{A} = \mathbf{A}^{\dagger} \cdot \mathbf{T} \cdot \mathbf{A} = \mathbf{A} \circ \mathbf{T} \circ \mathbf{G}.$$

This gives a “smaller” DTMC via the abstracted generator $\mathbf{T}^{\#}$.

Probabilistic Abstract Interpretation

Probabilistic Abstract Interpretation is based on:

- Concrete and abstract domains are linear spaces $\mathcal{C}$, $\mathcal{D}$...
- Concrete and abstract semantics are linear operators **T**...

The Moore-Penrose pseudo-inverse allows us to construct the closest (i.e. least square) approximation

$$\mathbf{T}^{\#} : \mathcal{D} \rightarrow \mathcal{D} \text{ of a concrete semantics } \mathbf{T} : \mathcal{C} \rightarrow \mathcal{C}$$

which we define via the Moore-Penrose pseudo-inverse:

$$\mathbf{T}^{\#} = \mathbf{G} \cdot \mathbf{T} \cdot \mathbf{A} = \mathbf{A}^{\dagger} \cdot \mathbf{T} \cdot \mathbf{A} = \mathbf{A} \circ \mathbf{T} \circ \mathbf{G}.$$

This gives a “smaller” DTMC via the abstracted generator $\mathbf{T}^{\#}$.

Probabilistic Abstract Interpretation

Probabilistic Abstract Interpretation is based on:

- Concrete and abstract domains are linear spaces $\mathcal{C}$, $\mathcal{D}$...
- Concrete and abstract semantics are linear operators $\mathbf{T}$...

The Moore-Penrose pseudo-inverse allows us to construct the closest (i.e. least square) approximation

$$\mathbf{T}^{\#} : \mathcal{D} \to \mathcal{D} \text{ of a concrete semantics } \mathbf{T} : \mathcal{C} \to \mathcal{C}$$

which we define via the Moore-Penrose pseudo-inverse:

$$\mathbf{T}^{\#} = \mathbf{G} \cdot \mathbf{T} \cdot \mathbf{A} = \mathbf{A}^{\dagger} \cdot \mathbf{T} \cdot \mathbf{A} = \mathbf{A} \circ \mathbf{T} \circ \mathbf{G}.$$

This gives a “smaller” DTMC via the abstracted generator $\mathbf{T}^{\#}$.

Probabilistic Abstract Interpretation

Probabilistic Abstract Interpretation is based on:

- Concrete and abstract domains are linear spaces $\mathcal{C}$, $\mathcal{D}$...
- Concrete and abstract semantics are linear operators $\mathbf{T}$...

The Moore-Penrose pseudo-inverse allows us to construct the closest (i.e. least square) approximation

$$\mathbf{T}^{\#} : \mathcal{D} \rightarrow \mathcal{D} \text{ of a concrete semantics } \mathbf{T} : \mathcal{C} \rightarrow \mathcal{C}$$

which we define via the Moore-Penrose pseudo-inverse:

$$\mathbf{T}^{\#} = \mathbf{G} \cdot \mathbf{T} \cdot \mathbf{A} = \mathbf{A}^{\dagger} \cdot \mathbf{T} \cdot \mathbf{A} = \mathbf{A} \circ \mathbf{T} \circ \mathbf{G}.$$

This gives a “smaller” DTMC via the abstracted generator $\mathbf{T}^{\#}$.

Probabilistic Program Analysis vs Statistics

Probabilistic Program Analysis

- Probabilities are given (as values or parameters):
- Calculate properties according to these input data using the program semantics,
- i.e. deduce probabilities of properties from semantics.

Statistical Analysis

- Probabilities and initial states are not known:
- Estimate these parameters using observations of the program behaviour,
- i.e. infer execution probabilities by observing some sample runs.

Probabilistic Program Analysis vs Statistics

Probabilistic Program Analysis

- Probabilities are given (as values or parameters):
- Calculate properties according to these input data using the program semantics,
- i.e. deduce probabilities of properties from semantics.

Statistical Analysis

- Probabilities and initial states are not known:
- Estimate these parameters using observations of the program behaviour,
- i.e. infer execution probabilities by observing some sample runs.

Probabilistic Program Analysis vs Statistics

Probabilistic Program Analysis

- Probabilities are given (as values or parameters):
- Calculate properties according to these input data using the program semantics,
- i.e. deduce probabilities of properties from semantics.

Statistical Analysis

- Probabilities and initial states are not known:
- Estimate these parameters using observations of the program behaviour,
- i.e. infer execution probabilities by observing some sample runs.

Probabilistic Program Analysis vs Statistics

Probabilistic Program Analysis

- Probabilities are given (as values or parameters):
- Calculate properties according to these input data using the program semantics,
- i.e. deduce probabilities of properties from semantics.

Statistical Analysis

- Probabilities and initial states are not known:
- Estimate these parameters using observations of the program behaviour,
- i.e. infer execution probabilities by observing some sample runs.

Probabilistic Program Analysis vs Statistics

Probabilistic Program Analysis

- Probabilities are given (as values or parameters):
- Calculate properties according to these input data using the program semantics,
- i.e. deduce probabilities of properties from semantics.

Statistical Analysis

- Probabilities and initial states are not known:
- Estimate these parameters using observations of the program behaviour,
- i.e. infer execution probabilities by observing some sample runs.

Probabilistic Program Analysis vs Statistics

Probabilistic Program Analysis

- Probabilities are given (as values or parameters):
- Calculate properties according to these input data using the program semantics,
- i.e. deduce probabilities of properties from semantics.

Statistical Analysis

- Probabilities and initial states are not known:
- Estimate these parameters using observations of the program behaviour,
- i.e. infer execution probabilities by observing some sample runs.

Probabilistic Program Analysis vs Statistics

Probabilistic Program Analysis

- Probabilities are given (as values or parameters):
- Calculate properties according to these input data using the program semantics,
- i.e. deduce probabilities of properties from semantics.

Statistical Analysis

- Probabilities and initial states are not known:
- Estimate these parameters using observations of the program behaviour,
- i.e. infer execution probabilities by observing some sample runs.

Using Statistics

Infer execution probabilities by observing some sample runs.

- Identify a random vector y with some measurement results
- Identify a model by a vector of parameters β
- Construct a matrix $\mathbf{X}$ mapping models to the runs
- Use $\mathbf{X}^\dagger$ and y to find a best estimator of the model.

Theorem (Gauss-Markov)

Consider the linear model $y = \beta\mathbf{X} + \varepsilon$ with $\mathbf{X}$ of full column rank and ε (fulfilling some conditions) Then the Best Linear Unbiased Estimator *(BLUE) is given by*

$$\hat{\beta} = y\mathbf{X}^\dagger.$$

Using Statistics

Infer execution probabilities by observing some sample runs.

- Identify a random vector y with some measurement results
- Identify a model by a vector of parameters β
- Construct a matrix $\mathbf{X}$ mapping models to the runs
- Use $\mathbf{X}^{\dagger}$ and y to find a best estimator of the model.

Theorem (Gauss-Markov)

Consider the linear model $y = \beta\mathbf{X} + \varepsilon$ with $\mathbf{X}$ of full column rank and ε (fulfilling some conditions) Then the Best Linear Unbiased Estimator *(BLUE) is given by*

$$\hat{\beta} = y\mathbf{X}^{\dagger}.$$

Using Statistics

Infer execution probabilities by observing some sample runs.

- Identify a random vector y with some measurement results
- Identify a model by a vector of parameters β
- Construct a matrix $\mathbf{X}$ mapping models to the runs
- Use $\mathbf{X}^\dagger$ and y to find a best estimator of the model.

Theorem (Gauss-Markov)

Consider the linear model $y = \beta\mathbf{X} + \varepsilon$ with $\mathbf{X}$ of full column rank and ε (fulfilling some conditions) Then the Best Linear Unbiased Estimator *(BLUE) is given by*

$$\hat{\beta} = y\mathbf{X}^\dagger.$$

Using Statistics

Infer execution probabilities by observing some sample runs.

- Identify a random vector y with some measurement results
- Identify a model by a vector of parameters β
- Construct a matrix $\mathbf{X}$ mapping models to the runs
- Use $\mathbf{X}^\dagger$ and y to find a best estimator of the model.

Theorem (Gauss-Markov)

Consider the linear model $y = \beta\mathbf{X} + \varepsilon$ with $\mathbf{X}$ of full column rank and ε (fulfilling some conditions) Then the Best Linear Unbiased Estimator *(BLUE) is given by*

$$\hat{\beta} = y\mathbf{X}^\dagger.$$

Using Statistics

Infer execution probabilities by observing some sample runs.

- Identify a random vector y with some measurement results
- Identify a model by a vector of parameters β
- Construct a matrix $\mathbf{X}$ mapping models to the runs
- Use $\mathbf{X}^\dagger$ and y to find a best estimator of the model.

Theorem (Gauss-Markov)

Consider the linear model $y = \beta\mathbf{X} + \varepsilon$ with $\mathbf{X}$ of full column rank and ε (fulfilling some conditions) Then the Best Linear Unbiased Estimator *(BLUE) is given by*

$$\hat{\beta} = y\mathbf{X}^\dagger.$$

Using Statistics

Infer execution probabilities by observing some sample runs.

- Identify a random vector y with some measurement results
- Identify a model by a vector of parameters β
- Construct a matrix $\mathbf{X}$ mapping models to the runs
- Use $\mathbf{X}^\dagger$ and y to find a best estimator of the model.

Theorem (Gauss-Markov)

Consider the linear model $y = \beta\mathbf{X} + \varepsilon$ with $\mathbf{X}$ of full column rank and ε (fulfilling some conditions) Then the Best Linear Unbiased Estimator *(BLUE) is given by*

$$\hat{\beta} = y\mathbf{X}^\dagger.$$

Modular Exponentiation

```
s := 1;
i := 0;
while i<=w do
  if k[i]==1 then
     x := (s*x) mod n;
  else
     r := s;
  fi;
  s := r*r;
  i := i+1;
od;
```

P.C. Kocher: *Cryptanalysis of Diffie-Hellman, RSA, DSS, and other cryptosystems using timing attacks*, CRYPTO '95.

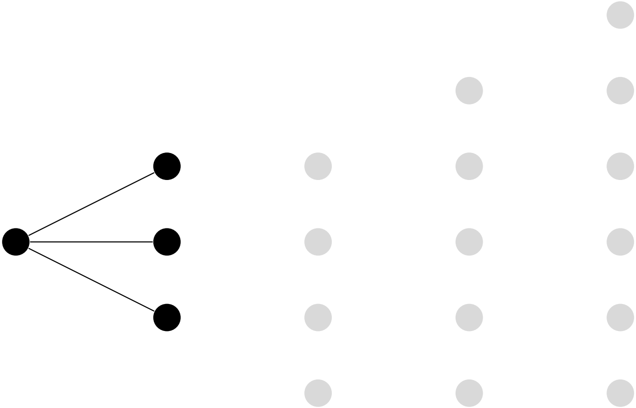
Modular Exponentiation

```
s := 1;
i := 0;
while i<=w do
  if k[i]==1 then
     x := (s*x) mod n;
  else
     r := s;
  fi;
  s := r*r;
  i := i+1;
od;
```

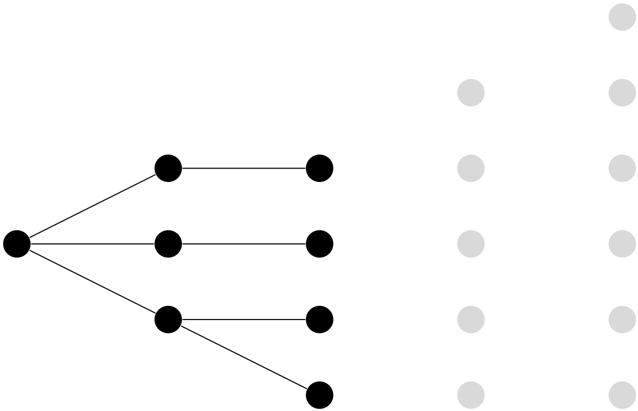
P.C. Kocher: *Cryptanalysis of Diffie-Hellman, RSA, DSS, and other cryptosystems using timing attacks*, CRYPTO '95.

Paths and Fronts

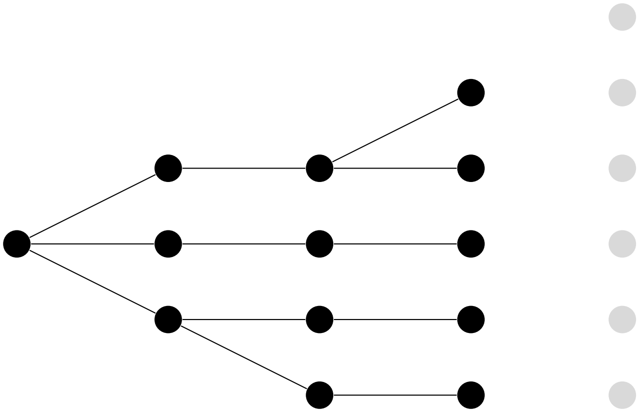
Paths and Fronts



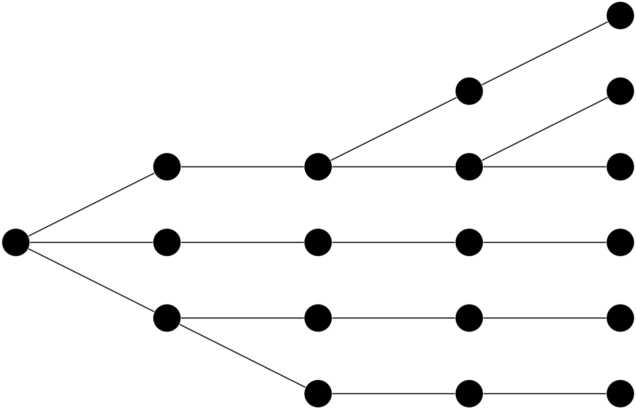
Paths and Fronts



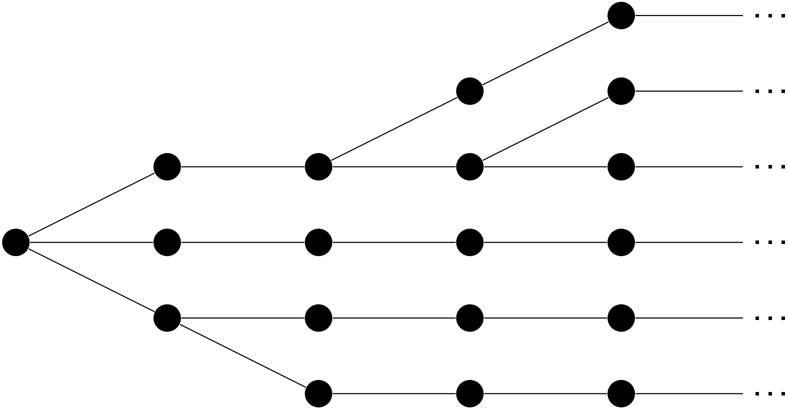
Paths and Fronts



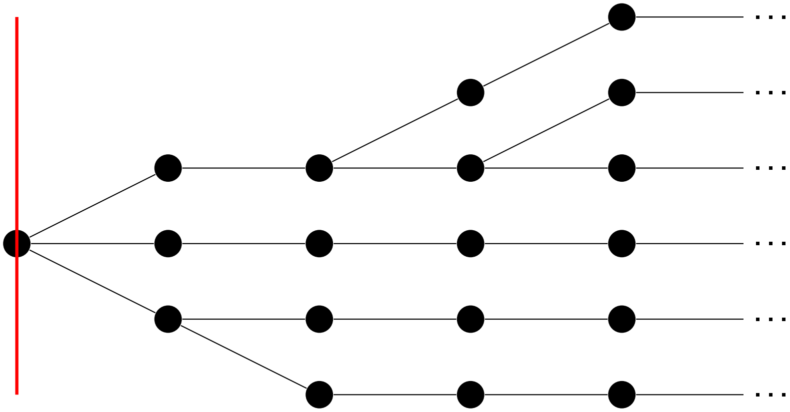
Paths and Fronts



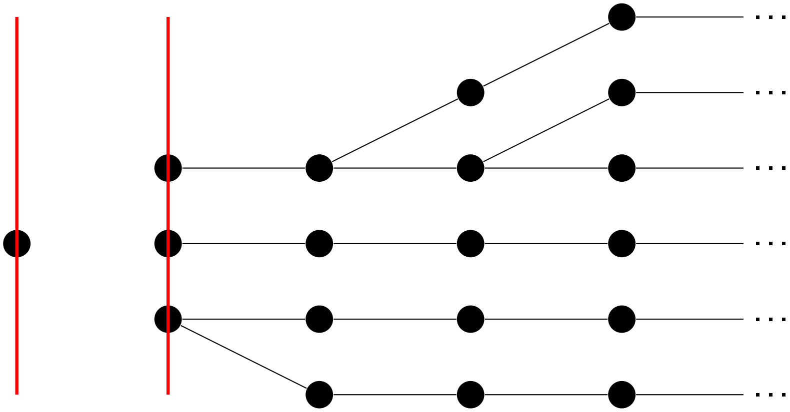
Paths and Fronts



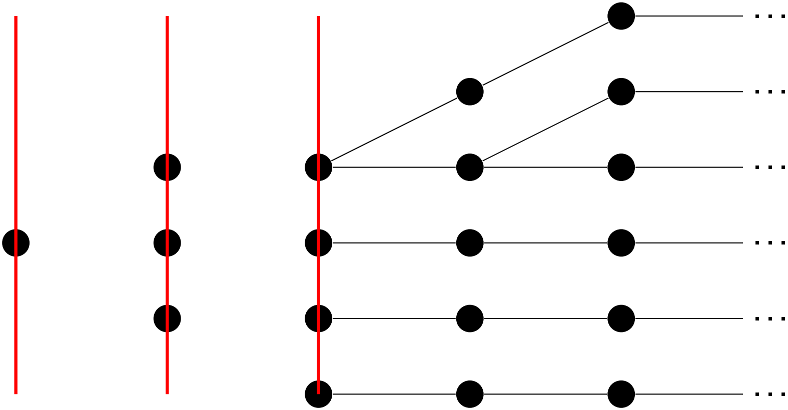
Paths and Fronts



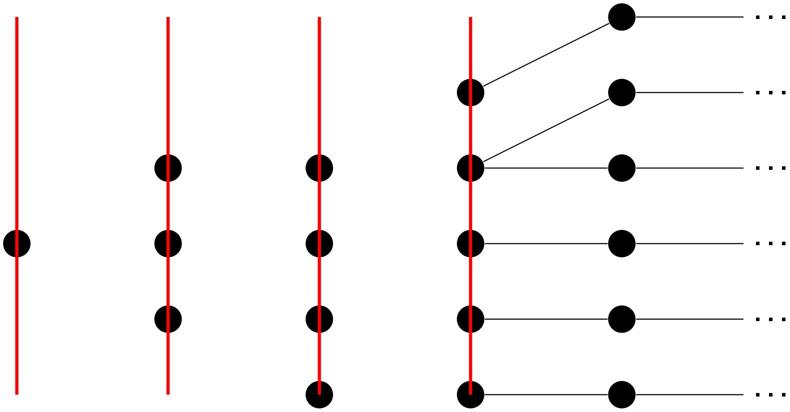
Paths and Fronts



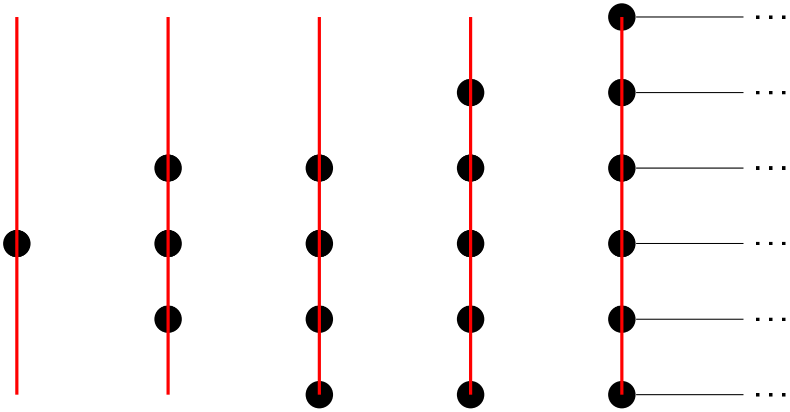
Paths and Fronts



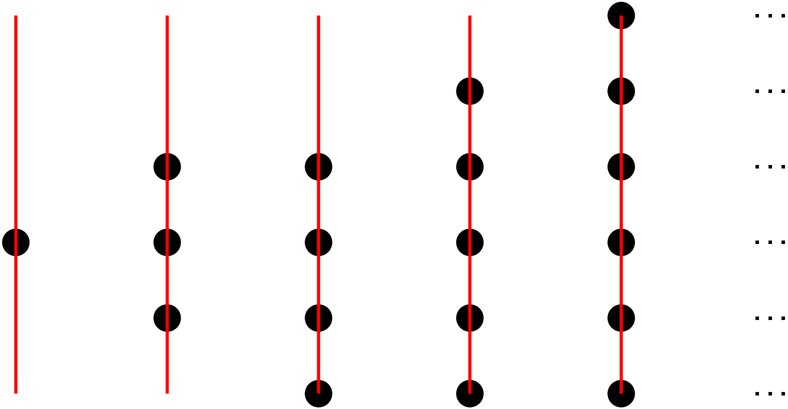
Paths and Fronts



Paths and Fronts



Paths and Fronts



Observing Traces: The DTMC

Consider the following simple DTMC with parameters p and q in the real interval $[0, 1]$:

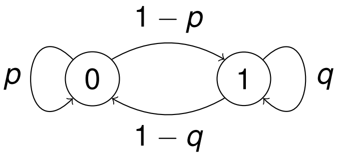
$$\mathbf{T}_{pq} = \begin{pmatrix} p & 1-p \\ 1-q & q \end{pmatrix}$$

This behaviour is essentially the one of the following program:

while (*true*) *do*
  if ($x == 1$)
    then x ?= $\{\langle 0, p\rangle, \langle 1, 1-p\rangle\}$
    else x ?= $\{\langle 0, 1-q\rangle, \langle 1, q\rangle\}$
  fi
od

Observing Traces: The DTMC

Consider the following simple DTMC with parameters p and q in the real interval $[0, 1]$:



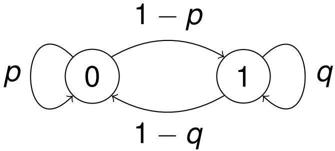
$$\mathbf{T}_{pq} = \begin{pmatrix} p & 1-p \\ 1-q & q \end{pmatrix}$$

This behaviour is essentially the one of the following program:

```
while (true) do
    if (x == 1)
        then x ?= {⟨0, p⟩, ⟨1, 1 − p⟩}
        else x ?= {⟨0, 1 − q⟩, ⟨1, q⟩}
    fi
od
```

Observing Traces: The DTMC

Consider the following simple DTMC with parameters p and q in the real interval $[0, 1]$:



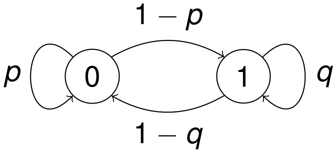
$$\mathbf{T}_{pq} = \begin{pmatrix} p & 1-p \\ 1-q & q \end{pmatrix}$$

This behaviour is essentially the one of the following program:

$$
\begin{array}{l}
\textit{while}\ \ (\textit{true})\ \ \textit{do} \\
\quad \textit{if}\ \ (x == 1) \\
\qquad \textit{then}\ \ x\ \ ?=\ \ \{\langle 0, p\rangle, \langle 1, 1-p\rangle\} \\
\qquad \textit{else}\ \ x\ \ ?=\ \ \{\langle 0, 1-q\rangle, \langle 1, q\rangle\} \\
\quad \textit{fi} \\
\textit{od}
\end{array}
$$

Observing Traces: The DTMC

Consider the following simple DTMC with parameters p and q in the real interval $[0, 1]$:



$$\mathbf{T}_{pq} = \begin{pmatrix} p & 1-p \\ 1-q & q \end{pmatrix}$$

This behaviour is essentially the one of the following program:

$$
\begin{array}{l}
\textit{while}\ \ (\textit{true})\ \ \textit{do} \\
\quad \textit{if}\ \ (x == 1) \\
\quad\quad \textit{then}\ \ x\ \ ?=\ \ \{\langle 0, p\rangle, \langle 1, 1-p\rangle\} \\
\quad\quad \textit{else}\ \ x\ \ ?=\ \ \{\langle 0, 1-q\rangle, \langle 1, q\rangle\} \\
\quad \textit{fi} \\
\textit{od}
\end{array}
$$

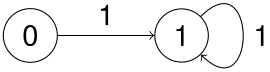
Instantiating the parameters:

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

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

Observing Traces: Possible Parameters

Instantiating the parameters:

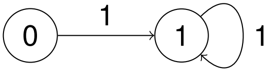


$$\mathbf{T}_{0,1} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$$

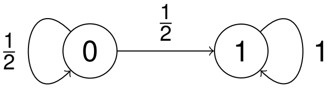
$$\mathbf{T}_{\frac{1}{2},1} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ 0 & 1 \end{pmatrix}$$

Observing Traces: Possible Parameters

Instantiating the parameters:



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



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

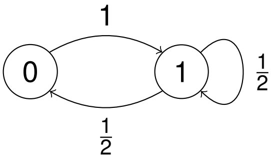
Instantiating the parameters:

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

$$\mathbf{T}_{\frac{1}{2},\frac{1}{2}} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

Observing Traces: Possible Parameters

Instantiating the parameters:

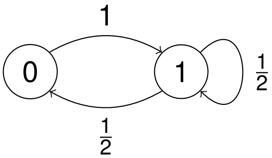


$$\mathbf{T}_{0,\frac{1}{2}} = \begin{pmatrix} 0 & 1 \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

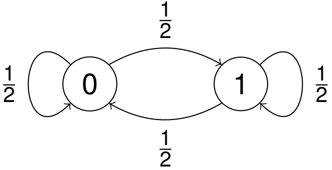
$$\mathbf{T}_{\frac{1}{2},\frac{1}{2}} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

Observing Traces: Possible Parameters

Instantiating the parameters:



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



$$\mathbf{T}_{\frac{1}{2},\frac{1}{2}} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

Identifying the Concrete Model

PAI can be used to this purpose as follows:

- Abstract domain: $\mathcal{D} = \mathcal{V}(\mathcal{M})$, with $\mathcal{M} = \{\langle s, p, q\rangle \mid s \in \{0,1\}, p, q \in [0,1]\}$
- Concrete domain: $\mathcal{C} = \mathcal{V}(\mathcal{T})$ with $\mathcal{T} = \{0,1\}^{+\infty}$ (execution traces)
- Design matrix: $\mathbf{G} : \mathcal{D} \rightarrow \mathcal{C}$ associates to each instance model the corresponding distribution on traces
- Compute the Moore-Penrose pseudo-inverse $\mathbf{G}^\dagger$ of $\mathbf{G}$ to calculate the best estimators of the parameters p and q.

Identifying the Concrete Model

PAI can be used to this purpose as follows:

- Abstract domain: $\mathcal{D} = \mathcal{V}(\mathcal{M})$, with $\mathcal{M} = \{\langle s, p, q\rangle \mid s \in \{0,1\}, p, q \in [0,1]\}$
- Concrete domain: $\mathcal{C} = \mathcal{V}(\mathcal{T})$ with $\mathcal{T} = \{0,1\}^{+\infty}$ (execution traces)
- Design matrix: $\mathbf{G} : \mathcal{D} \to \mathcal{C}$ associates to each instance model the corresponding distribution on traces
- Compute the Moore-Penrose pseudo-inverse $\mathbf{G}^\dagger$ of $\mathbf{G}$ to calculate the best estimators of the parameters p and q.

Identifying the Concrete Model

PAI can be used to this purpose as follows:

- Abstract domain: $\mathcal{D} = \mathcal{V}(\mathcal{M})$, with $\mathcal{M} = \{\langle s, p, q\rangle \mid s \in \{0, 1\}, p, q \in [0, 1]\}$
- Concrete domain: $\mathcal{C} = \mathcal{V}(\mathcal{T})$ with $\mathcal{T} = \{0, 1\}^{+\infty}$ (execution traces)
- Design matrix: $\mathbf{G} : \mathcal{D} \to \mathcal{C}$ associates to each instance model the corresponding distribution on traces
- Compute the Moore-Penrose pseudo-inverse $\mathbf{G}^\dagger$ of $\mathbf{G}$ to calculate the best estimators of the parameters p and q.

Identifying the Concrete Model

PAI can be used to this purpose as follows:

- Abstract domain: $\mathcal{D} = \mathcal{V}(\mathcal{M})$, with $\mathcal{M} = \{\langle s, p, q\rangle \mid s \in \{0,1\}, p, q \in [0,1]\}$
- Concrete domain: $\mathcal{C} = \mathcal{V}(\mathcal{T})$ with $\mathcal{T} = \{0,1\}^{+\infty}$ (execution traces)
- Design matrix: $\mathbf{G} : \mathcal{D} \to \mathcal{C}$ associates to each instance model the corresponding distribution on traces
- Compute the Moore-Penrose pseudo-inverse $\mathbf{G}^\dagger$ of $\mathbf{G}$ to calculate the best estimators of the parameters p and q.

Identifying the Concrete Model

PAI can be used to this purpose as follows:

- Abstract domain: $\mathcal{D} = \mathcal{V}(\mathcal{M})$, with $\mathcal{M} = \{\langle s, p, q\rangle \mid s \in \{0, 1\}, p, q \in [0, 1]\}$
- Concrete domain: $\mathcal{C} = \mathcal{V}(\mathcal{T})$ with $\mathcal{T} = \{0, 1\}^{+\infty}$ (execution traces)
- Design matrix: $\mathbf{G} : \mathcal{D} \to \mathcal{C}$ associates to each instance model the corresponding distribution on traces
- Compute the Moore-Penrose pseudo-inverse $\mathbf{G}^\dagger$ of $\mathbf{G}$ to calculate the best estimators of the parameters p and q.

Numerical Experiments

In order to be able to compute an analysis of the system we considered $p, q \in \{0, \frac{1}{2}, 1\}$, i.e. 9 possible semantics, with possible initial states either 0 or 1.

$$\mathcal{D} = \mathcal{V}(\{0,1\}) \otimes \mathcal{V}(\{0,\frac{1}{2},1\}) \otimes \mathcal{V}(\{0,\frac{1}{2},1\}) = \mathbb{R}^2 \otimes \mathbb{R}^3 \otimes \mathbb{R}^3 = \mathbb{R}^{18}$$

Observe traces of a certain length, e.g. traces of length $t = 3$:

$$\mathcal{C}_3 = \mathcal{V}(\{0,1\}^3) = \mathcal{V}(\{0,1\})^{\otimes 3} = (\mathbb{R}^2)^{\otimes 8} = \mathbb{R}^8$$

Actually, we simulated 10000 executions (with errors) of the system and observed traces of length $t = 10$.

$$\mathcal{C}_{10} = \mathcal{V}(\{0,1\}^{10}) = \mathcal{V}(\{0,1\})^{\otimes 10} = (\mathbb{R}^2)^{\otimes 10} = \mathbb{R}^{1024}$$

Numerical Experiments

In order to be able to compute an analysis of the system we considered $p, q \in \{0, \frac{1}{2}, 1\}$, i.e. 9 possible semantics, with possible initial states either 0 or 1.

$$\mathcal{D} = \mathcal{V}(\{0,1\}) \otimes \mathcal{V}(\{0, \frac{1}{2}, 1\}) \otimes \mathcal{V}(\{0, \frac{1}{2}, 1\}) = \mathbb{R}^2 \otimes \mathbb{R}^3 \otimes \mathbb{R}^3 = \mathbb{R}^{18}$$

Observe traces of a certain length, e.g. traces of length $t = 3$:

$$\mathcal{C}_3 = \mathcal{V}(\{0,1\}^3) = \mathcal{V}(\{0,1\})^{\otimes 3} = (\mathbb{R}^2)^{\otimes 8} = \mathbb{R}^8$$

Actually, we simulated 10000 executions (with errors) of the system and observed traces of length $t = 10$.

$$\mathcal{C}_{10} = \mathcal{V}(\{0,1\}^{10}) = \mathcal{V}(\{0,1\})^{\otimes 10} = (\mathbb{R}^2)^{\otimes 10} = \mathbb{R}^{1024}$$

Numerical Experiments

In order to be able to compute an analysis of the system we considered $p, q \in \{0, \frac{1}{2}, 1\}$, i.e. 9 possible semantics, with possible initial states either 0 or 1.

$$\mathcal{D} = \mathcal{V}(\{0,1\}) \otimes \mathcal{V}(\{0, \frac{1}{2}, 1\}) \otimes \mathcal{V}(\{0, \frac{1}{2}, 1\}) = \mathbb{R}^2 \otimes \mathbb{R}^3 \otimes \mathbb{R}^3 = \mathbb{R}^{18}$$

Observe traces of a certain length, e.g. traces of length $t = 3$:

$$\mathcal{C}_3 = \mathcal{V}(\{0,1\}^3) = \mathcal{V}(\{0,1\})^{\otimes 3} = (\mathbb{R}^2)^{\otimes 8} = \mathbb{R}^8$$

Actually, we simulated 10000 executions (with errors) of the system and observed traces of length $t = 10$.

$$\mathcal{C}_{10} = \mathcal{V}(\{0,1\}^{10}) = \mathcal{V}(\{0,1\})^{\otimes 10} = (\mathbb{R}^2)^{\otimes 10} = \mathbb{R}^{1024}$$

Numerical Experiments

In order to be able to compute an analysis of the system we considered $p, q \in \{0, \frac{1}{2}, 1\}$, i.e. 9 possible semantics, with possible initial states either 0 or 1.

$$\mathcal{D} = \mathcal{V}(\{0, 1\}) \otimes \mathcal{V}(\{0, \frac{1}{2}, 1\}) \otimes \mathcal{V}(\{0, \frac{1}{2}, 1\}) = \mathbb{R}^2 \otimes \mathbb{R}^3 \otimes \mathbb{R}^3 = \mathbb{R}^{18}$$

Observe traces of a certain length, e.g. traces of length $t = 3$:

$$\mathcal{C}_3 = \mathcal{V}(\{0, 1\}^3) = \mathcal{V}(\{0, 1\})^{\otimes 3} = (\mathbb{R}^2)^{\otimes 8} = \mathbb{R}^8$$

Actually, we simulated 10000 executions (with errors) of the system and observed traces of length $t = 10$.

$$\mathcal{C}_{10} = \mathcal{V}(\{0, 1\}^{10}) = \mathcal{V}(\{0, 1\})^{\otimes 10} = (\mathbb{R}^2)^{\otimes 10} = \mathbb{R}^{1024}$$

Numerical Experiments

In order to be able to compute an analysis of the system we considered $p, q \in \{0, \frac{1}{2}, 1\}$, i.e. 9 possible semantics, with possible initial states either 0 or 1.

$$\mathcal{D} = \mathcal{V}(\{0,1\}) \otimes \mathcal{V}(\{0, \frac{1}{2}, 1\}) \otimes \mathcal{V}(\{0, \frac{1}{2}, 1\}) = \mathbb{R}^2 \otimes \mathbb{R}^3 \otimes \mathbb{R}^3 = \mathbb{R}^{18}$$

Observe traces of a certain length, e.g. traces of length $t = 3$:

$$\mathcal{C}_3 = \mathcal{V}(\{0,1\}^3) = \mathcal{V}(\{0,1\})^{\otimes 3} = (\mathbb{R}^2)^{\otimes 8} = \mathbb{R}^8$$

Actually, we simulated 10000 executions (with errors) of the system and observed traces of length $t = 10$.

$$\mathcal{C}_{10} = \mathcal{V}(\{0,1\}^{10}) = \mathcal{V}(\{0,1\})^{\otimes 10} = (\mathbb{R}^2)^{\otimes 10} = \mathbb{R}^{1024}$$

Numerical Experiments: Parameter Space $\mathcal{D} = \mathbb{R}^9$

s	p	q
0	0	0
1	0	0
0	$\frac{1}{2}$	0
1	$\frac{1}{2}$	0
0	1	0
1	1	0
0	0	$\frac{1}{2}$
1	0	$\frac{1}{2}$
0	$\frac{1}{2}$	$\frac{1}{2}$

s	p	q
1	$\frac{1}{2}$	$\frac{1}{2}$
0	1	$\frac{1}{2}$
1	1	$\frac{1}{2}$
0	0	1
1	0	1
0	$\frac{1}{2}$	1
1	$\frac{1}{2}$	1
0	1	1
1	1	1

Experiments: Trace Space $\mathcal{C}_3 = \mathbb{R}^8$ and $\mathcal{C}_{10} = \mathbb{R}^{1024}$

trace $\mathcal{C}_3$		
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

trace $\mathcal{C}_{10}$									
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	1
0	0	0	0	0	0	1	0	1	0
0	0	0	0	0	0	1	0	1	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Experiments: Trace Space $\mathcal{C}_3 = \mathbb{R}^8$ and $\mathcal{C}_{10} = \mathbb{R}^{1024}$

trace $\mathcal{C}_3$		
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

trace $\mathcal{C}_{10}$									
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	1
0	0	0	0	0	0	1	0	1	0
0	0	0	0	0	0	1	0	1	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Experiments: Concretisation $\mathbf{G}_3$

$$
\mathbf{G}_3 = \begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
\frac{1}{4} & \frac{1}{4} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \\
\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{4} & \frac{1}{4} \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\frac{1}{4} & \frac{1}{4} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
$$

Experiments: Regression $\mathbf{G}_3^\dagger$ (Abstraction)

$$
\mathbf{G}_3^{\dagger t} = \begin{pmatrix}
0 & -\frac{2}{3} & \frac{11}{15} & -\frac{1}{15} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -\frac{1}{15} & \frac{11}{15} & -\frac{2}{3} & 0 \\
0 & \frac{4}{3} & \frac{1}{5} & -\frac{1}{5} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & -\frac{2}{3} & 0 \\
\frac{1}{3} & -\frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{11}{15} & -\frac{1}{15} & -\frac{2}{3} & 0 \\
0 & -\frac{2}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -\frac{1}{5} & \frac{1}{5} & \frac{4}{3} & 0 \\
0 & \frac{4}{3} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{4}{3} & 0 \\
\frac{1}{3} & -\frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{5} & -\frac{1}{5} & \frac{4}{3} & 0 \\
0 & -\frac{2}{3} & -\frac{1}{15} & \frac{11}{15} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{3} & \frac{1}{3} \\
0 & \frac{4}{3} & -\frac{1}{5} & \frac{1}{5} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{3} & \frac{1}{3} \\
\frac{1}{3} & -\frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{3} & \frac{1}{3}
\end{pmatrix}
$$

Numerical Experiments for $\mathcal{C}_{10}$

For the model $p = 0, q = \frac{1}{2}$ we obtained (for different noise distortions ε) by observation of the possible traces in 10000 test runs their (experimental) probability distributions y, y' etc. in $\mathbb{R}^{1024}$ (where y_i is the observed frequency of trace i) and from these estimate the (unknown) parameters via:

$$\begin{aligned}
y\mathbf{G}_{10}^{\dagger} &= (0,0,0,0,0,0,0.50,0.49,0,0.01,0,0,0,0,0,0,0,0) \\
y'\mathbf{G}_{10}^{\dagger} &= (0,0,0,0,0,0,0.49,0.50,0.01,0,0,0,0,0,0,0,0,0) \\
y''\mathbf{G}_{10}^{\dagger} &= (0,0,0,0,0,0,0.43,0.43,0.07,0.06,0,0,0,0,0,0,0,0) \\
y'''\mathbf{G}_{10}^{\dagger} &= (0,0,0.01,0,0,0,0.33,0.35,0.16,0.16,0,0,0,0,0,0,0,0)
\end{aligned}$$

The distribution y denotes the undistorted case, y' the case with $\varepsilon = 0.01$, y'' the case $\varepsilon = 0.1$, and y''' the case $\varepsilon = 0.25$.

The initial state was always chosen with probability $\frac{1}{2}$ as the state 0 or the state 1.

Numerical Experiments for $\mathcal{C}_{10}$

For the model $p = 0, q = \frac{1}{2}$ we obtained (for different noise distortions ε) by observation of the possible traces in 10000 test runs their (experimental) probability distributions y, y' etc. in $\mathbb{R}^{1024}$ (where y_i is the observed frequency of trace i) and from these estimate the (unknown) parameters via:

$$
\begin{aligned}
y\mathbf{G}_{10}^{\dagger} &= (0,0,0,0,0,0,0.50,0.49,0,0.01,0,0,0,0,0,0,0,0) \\
y'\mathbf{G}_{10}^{\dagger} &= (0,0,0,0,0,0,0.49,0.50,0.01,0,0,0,0,0,0,0,0,0) \\
y''\mathbf{G}_{10}^{\dagger} &= (0,0,0,0,0,0,0.43,0.43,0.07,0.06,0,0,0,0,0,0,0,0) \\
y'''\mathbf{G}_{10}^{\dagger} &= (0,0,0.01,0,0,0,0.33,0.35,0.16,0.16,0,0,0,0,0,0,0,0)
\end{aligned}
$$

The distribution y denotes the undistorted case, y' the case with $\varepsilon = 0.01$, y'' the case $\varepsilon = 0.1$, and y''' the case $\varepsilon = 0.25$.

The initial state was always chosen with probability $\frac{1}{2}$ as the state 0 or the state 1.

Numerical Experiments for $\mathcal{C}_{10}$

For the model $p = 0, q = \frac{1}{2}$ we obtained (for different noise distortions ε) by observation of the possible traces in 10000 test runs their (experimental) probability distributions y, y' etc. in $\mathbb{R}^{1024}$ (where y_i is the observed frequency of trace i) and from these estimate the (unknown) parameters via:

$$\begin{aligned}
y\mathbf{G}_{10}^{\dagger} &= (0,0,0,0,0,0,0.50,0.49,0,0.01,0,0,0,0,0,0,0,0) \\
y'\mathbf{G}_{10}^{\dagger} &= (0,0,0,0,0,0,0.49,0.50,0.01,0,0,0,0,0,0,0,0,0) \\
y''\mathbf{G}_{10}^{\dagger} &= (0,0,0,0,0,0,0.43,0.43,0.07,0.06,0,0,0,0,0,0,0,0) \\
y'''\mathbf{G}_{10}^{\dagger} &= (0,0,0.01,0,0,0,0.33,0.35,0.16,0.16,0,0,0,0,0,0,0,0)
\end{aligned}$$

The distribution y denotes the undistorted case, y' the case with $\varepsilon = 0.01$, y'' the case $\varepsilon = 0.1$, and y''' the case $\varepsilon = 0.25$.

The initial state was always chosen with probability $\frac{1}{2}$ as the state 0 or the state 1.

Some References

- Di Pierro, Wiklicky: *Probabilistic data flow analysis: A linear equational approach*. Proceedings of GandALF'13, EPTCS, Volume 119, 2013.
- Di Pierro, Hankin, Wiklicky: Probabilistic semantics and analysis. in Formal Methods for Quantitative Aspects of Programming Languages, LNCS 6155, Springer, 2010.
- Di Pierro, Wiklicky: *Probabilistic Abstract Intepretation: From Trace Semantics to DTMC's via Linear Regression*. LNCS 9560, Springer, 2016.
- Nielson, Nielson, Hankin: *Principles of Program Analysis*. Springer, 1999/2005.

Some References

- Di Pierro, Wiklicky: *Probabilistic data flow analysis: A linear equational approach*. Proceedings of GandALF'13, EPTCS, Volume 119, 2013.
- Di Pierro, Hankin, Wiklicky: Probabilistic semantics and analysis. in Formal Methods for Quantitative Aspects of Programming Languages, LNCS 6155, Springer, 2010.
- Di Pierro, Wiklicky: *Probabilistic Abstract Intepretation: From Trace Semantics to DTMC's via Linear Regression*. LNCS 9560, Springer, 2016.
- Nielson, Nielson, Hankin: *Principles of Program Analysis*. Springer, 1999/2005.

Some References

- Di Pierro, Wiklicky: *Probabilistic data flow analysis: A linear equational approach*. Proceedings of GandALF'13, EPTCS, Volume 119, 2013.
- Di Pierro, Hankin, Wiklicky: Probabilistic semantics and analysis. in Formal Methods for Quantitative Aspects of Programming Languages, LNCS 6155, Springer, 2010.
- Di Pierro, Wiklicky: *Probabilistic Abstract Intepretation: From Trace Semantics to DTMC's via Linear Regression*. LNCS 9560, Springer, 2016.
- Nielson, Nielson, Hankin: *Principles of Program Analysis*. Springer, 1999/2005.

Some References

- Di Pierro, Wiklicky: *Probabilistic data flow analysis: A linear equational approach*. Proceedings of GandALF'13, EPTCS, Volume 119, 2013.
- Di Pierro, Hankin, Wiklicky: Probabilistic semantics and analysis. in Formal Methods for Quantitative Aspects of Programming Languages, LNCS 6155, Springer, 2010.
- Di Pierro, Wiklicky: *Probabilistic Abstract Intepretation: From Trace Semantics to DTMC's via Linear Regression*. LNCS 9560, Springer, 2016.
- Nielson, Nielson, Hankin: *Principles of Program Analysis*. Springer, 1999/2005.