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.
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.
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.
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:

\[
\begin{align*}
[x &:= 1]; \\
[y &:= 2]; \\
[x &:= x + y \mod 4]; \\
\text{if } [x > 2] & \text{ then } [z := x] \text{ else } [z := y] \text{ fi}
\end{align*}
\]

Extract statically the control flow relation – i.e. is it possible to go from label \( \ell \) to label \( \ell' \)?

\[
\text{flow} = \{(1, 2), (2, 3), (3, 4), (4, 5), (4, 6)\}
\]

Example

Consider a program like:

\[
\begin{align*}
x &:= 1^1; \\
y &:= z^2; \\
x &:= x + y \mod 4^3; \\
\text{if } [x > 2]^4 \text{ then } [z := x]^5 \text{ else } [z := y]^6 \text{ fi}
\end{align*}
\]

Extract statically the control flow relation – i.e. is it possible to go from label $\ell$ to label $\ell'$?

\[
\text{flow} = \{ (1, 2), (2, 3), (3, 4), (4, 5), (4, 6) \}
\]

Example

Consider a program like:

\[
\begin{align*}
  [x &:= 1]; \\
  [y &:= z]; \\
  [x &:= x + y \mod 4]; \\
  \text{if } [x > 2] \text{ then } [z := x] \text{ else } [z := y] \text{ fi}
\end{align*}
\]

Extract statically the control flow relation – i.e. is it possible to go from label $\ell$ to label $\ell'$?

\[
\text{flow} = \{(1, 2), (2, 3), (3, 4), (4, 5), (4, 6)\}
\]

Example

Consider a program like:

\[
\begin{align*}
[x &:= 1]^1; \\
[y &:= z]^2; \\
[x &:= x + y \text{ mod } 4]^3; \\
\text{if } [x > 2]^4 \text{ then } [z := x]^5 \text{ else } [z := y]^6 \text{ fi}
\end{align*}
\]

Extract statically the control flow relation – i.e. is it possible to go from label \( \ell \) to label \( \ell' \)?

\[
\text{flow} = \{ (1, 2), (2, 3), (3, 4), (4, 5), (4, 6) \}
\]

Example

Consider a program like:

\[
\begin{align*}
[x & := 1]^{1}; \\
[y & := z]^{2}; \\
[x & := x + y \mod 4]^{3}; \\
\text{if} \ [x > 2]^{4} \text{ then } [z & := x]^{5} \text{ else } [z & := y]^{6} \text{ fi}
\end{align*}
\]

Extract statically the control flow relation – i.e. is it possible to go from label \( \ell \) to label \( \ell' \)?

\[
flow = \{(1, 2), (2, 3), (3, 4), (4, 5), (4, 6)\}
\]

\( \text{gen}_{LV}([x := a]) = FV(a) \)
\( \text{gen}_{LV}([\text{skip}]) = \emptyset \)
\( \text{gen}_{LV}([b]) = FV(b) \)

\( \text{kill}_{LV}([x := a]) = \{x\} \)
\( \text{kill}_{LV}([\text{skip}]) = \emptyset \)
\( \text{kill}_{LV}([b]) = \emptyset \)

\( f^L_{LV} : \mathcal{P}(\text{Var}_*) \rightarrow \mathcal{P}(\text{Var}_*) \)
\( f^L_{LV}(X) = X \setminus \text{kill}_{LV}([B]) \cup \text{gen}_{LV}([B]) \)
(Local) Transfer Functions

\[
\begin{align*}
gen_{LV}([x := a]^{\ell}) &= FV(a) \\
gen_{LV}([\text{skip}]^{\ell}) &= \emptyset \\
gen_{LV}([b]^{\ell}) &= FV(b) \\
kill_{LV}([x := a]^{\ell}) &= \{x\} \\
kill_{LV}([\text{skip}]^{\ell}) &= \emptyset \\
kill_{LV}([b]^{\ell}) &= \emptyset
\end{align*}
\]

\[
f_{\ell}^{LV} : \mathcal{P}(\text{Var}_*) \rightarrow \mathcal{P}(\text{Var}_*)
\]

\[
f_{\ell}^{LV}(X) = X \setminus \kill_{LV}([B]^{\ell}) \cup \gen_{LV}([B]^{\ell})
\]
(Local) Transfer Functions

\[
\begin{align*}
\text{gen}_{LV}([x := a]^\ell) & = FV(a) \\
\text{gen}_{LV}([\text{skip}]^\ell) & = \emptyset \\
\text{gen}_{LV}([b]^\ell) & = FV(b) \\
\text{kill}_{LV}([x := a]^\ell) & = \{x\} \\
\text{kill}_{LV}([\text{skip}]^\ell) & = \emptyset \\
\text{kill}_{LV}([b]^\ell) & = \emptyset
\end{align*}
\]

\[
\begin{align*}
\text{f}^\ell_{LV} : \mathcal{P}(\mathbf{Var}_*) & \rightarrow \mathcal{P}(\mathbf{Var}_*) \\
\text{f}^\ell_{LV}(X) & = X \setminus \text{kill}_{LV}([B]^\ell) \cup \text{gen}_{LV}([B]^\ell)
\end{align*}
\]
Formulate equations based on the control flow (relations):

\[
\begin{align*}
LV_{entry}(\ell) &= f^L_V(LV_{exit}(\ell)) \\
LV_{exit}(\ell) &= \bigcup_{(\ell, \ell') \in \text{flow}} LV_{entry}(\ell')
\end{align*}
\]

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.)
Formulate equations based on the control flow (relations):

\[ LV_{\text{entry}}(\ell) = f_\ell^{LV}(LV_{\text{exit}}(\ell)) \]
\[ LV_{\text{exit}}(\ell) = \bigcup_{(\ell,\ell') \in \text{flow}} LV_{\text{entry}}(\ell') \]

**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.)
Formulate equations based on the control flow (relations):

\[
\begin{align*}
LV_{\text{entry}}(\ell) &= f^LV_\ell(LV_{\text{exit}}(\ell)) \\
LV_{\text{exit}}(\ell) &= \bigcup_{(\ell, \ell') \in \text{flow}} LV_{\text{entry}}(\ell')
\end{align*}
\]

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

$$[x := 1]; [y := 2]; [x := x + y \mod 4];$$
if $$[x > 2]$$ then $$[z := x]$$ else $$[z := y]$$ fi
Example


text content

Control Flow:

\[ \text{flow} = \{(1, 2), (2, 3), (3, 4), (4, 5), (4, 6)\} \]
Example

\[
[x := 1]; [y := 2]; [x := x + y \mod 4]; \\
\text{if } [x > 2] \text{ then } [z := x] \text{ else } [z := y] \text{ fi}
\]

Auxiliary Functions:

<table>
<thead>
<tr>
<th></th>
<th>\text{gen}_{LV}(\ell)</th>
<th>\text{kill}_{LV}(\ell)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>\emptyset</td>
<td>{x}</td>
</tr>
<tr>
<td>2</td>
<td>\emptyset</td>
<td>{y}</td>
</tr>
<tr>
<td>3</td>
<td>{x, y}</td>
<td>{x}</td>
</tr>
<tr>
<td>4</td>
<td>{x}</td>
<td>\emptyset</td>
</tr>
<tr>
<td>5</td>
<td>{x}</td>
<td>{z}</td>
</tr>
<tr>
<td>6</td>
<td>{y}</td>
<td>{z}</td>
</tr>
</tbody>
</table>
Example

\[ x := 1 \]; \[ y := 2 \]; \[ x := x + y \mod 4 \];
if \[ x > 2 \] then \[ z := x \] else \[ z := y \] fi

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

\[
\begin{align*}
\text{LV}_{\text{entry}}(1) &= \text{LV}_{\text{exit}}(1) \setminus \{x\} \\
\text{LV}_{\text{entry}}(2) &= \text{LV}_{\text{exit}}(2) \setminus \{y\} \\
\text{LV}_{\text{entry}}(3) &= \text{LV}_{\text{exit}}(3) \setminus \{x\} \cup \{x, y\} \\
\text{LV}_{\text{entry}}(4) &= \text{LV}_{\text{exit}}(4) \cup \{x\} \\
\text{LV}_{\text{entry}}(5) &= \text{LV}_{\text{exit}}(5) \setminus \{z\} \cup \{x\} \\
\text{LV}_{\text{entry}}(6) &= \text{LV}_{\text{exit}}(6) \setminus \{z\} \cup \{y\}
\end{align*}
\]
Example

\[\begin{align*}
  & [x := 1]^1; \\
  & [y := 2]^2; \\
  & [x := x + y \mod 4]^3; \\
  & \text{if } [x > 2]^4 \text{ then } [z := x]^5 \text{ else } [z := y]^6 \text{ fi}
\end{align*}\]

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

\[
\begin{align*}
  & \text{LV}_{\text{exit}}(1) = \text{LV}_{\text{entry}}(2) \\
  & \text{LV}_{\text{exit}}(2) = \text{LV}_{\text{entry}}(3) \\
  & \text{LV}_{\text{exit}}(3) = \text{LV}_{\text{entry}}(4) \\
  & \text{LV}_{\text{exit}}(4) = \text{LV}_{\text{entry}}(5) \cup \text{LV}_{\text{entry}}(6) \\
  & \text{LV}_{\text{exit}}(5) = \emptyset \\
  & \text{LV}_{\text{exit}}(6) = \emptyset
\end{align*}
\]
Example

\[
\begin{align*}
[x & := 1]^1; [y := 2]^2; [x := x + y \mod 4]^3; \\
\text{if } [x > 2]^4 \text{ then } [z := x]^5 \text{ else } [z := y]^6 \text{ fi}
\end{align*}
\]

Solutions (e.g. by fixed point iteration)

\[
\begin{align*}
\text{LV}_{\text{entry}}(1) & = \emptyset & \text{LV}_{\text{exit}}(1) & = \{x\} \\
\text{LV}_{\text{entry}}(2) & = \{x\} & \text{LV}_{\text{exit}}(2) & = \{x, y\} \\
\text{LV}_{\text{entry}}(3) & = \{x, y\} & \text{LV}_{\text{exit}}(3) & = \{x, y\} \\
\text{LV}_{\text{entry}}(4) & = \{x, y\} & \text{LV}_{\text{exit}}(4) & = \{x, y\} \\
\text{LV}_{\text{entry}}(5) & = \{x\} & \text{LV}_{\text{exit}}(5) & = \emptyset \\
\text{LV}_{\text{entry}}(6) & = \{y\} & \text{LV}_{\text{exit}}(6) & = \emptyset.
\end{align*}
\]
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).

\[
S ::= \text{skip} \\
| \quad x := e(x_1, \ldots, x_n) \\
| \quad x := \rho \\
| \quad S_1; S_2 \\
| \quad \text{if } b \text{ then } S_1 \text{ else } S_2 \text{ fi} \\
| \quad \text{while } b \text{ do } S \text{ od}
\]
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).

\[ S ::= \begin{array}{l}
[\text{skip}]^\ell \\
[\text{x := e}(x_1, \ldots, x_n)]^\ell \\
[\text{x := } \rho]^\ell \\
S_1 ; S_2 \\
\text{if } [b]^\ell \text{ then } S_1 \text{ else } S_2 \text{ fi} \\
\text{while } [b]^\ell \text{ do } S \text{ od}
\end{array} \]
Probabilistic Semantics

SOS:

R0 \[\langle \text{stop}, s \rangle \Rightarrow _1 \langle \text{stop}, s \rangle\]

R1 \[\langle \text{skip}, s \rangle \Rightarrow _1 \langle \text{stop}, s \rangle\]

R2 \[\langle v ::= e, s \rangle \Rightarrow _1 \langle \text{stop}, s[v \mapsto E(e)] \rangle\]

R3 \[\langle v ::= \rho, s \rangle \Rightarrow \rho(r) \langle \text{stop}, s[v \mapsto r] \rangle\]

... 

LOS:

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

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

...
Probabilistic Semantics

\textbf{SOS:}

\begin{align*}
R0 & \langle \text{stop}, s \rangle \Rightarrow_1 \langle \text{stop}, s \rangle \\
R1 & \langle \text{skip}, s \rangle \Rightarrow_1 \langle \text{stop}, s \rangle \\
R2 & \langle \nu := e, s \rangle \Rightarrow_1 \langle \text{stop}, s[\nu \mapsto \mathcal{E}(e)s] \rangle \\
R3 & \langle \nu \mathbin{?=} \rho, s \rangle \Rightarrow_1 \rho(r) \langle \text{stop}, s[\nu \mapsto r] \rangle \\
\ldots
\end{align*}

\textbf{LOS:}

\begin{align*}
T(\langle \ell_1, \rho, \ell_2 \rangle) & = U(x \leftarrow a) \otimes E(\ell_1, \ell_2) & \text{for} \ [x := a]^{\ell_1} \\
T(\langle \ell_1, \rho, \ell_2 \rangle) & = \left( \sum_i \rho(r_i) \cdot U(x \leftarrow r_i) \right) \otimes E(\ell_1, \ell_2) & \text{for} \ [x \mathbin{?=} \rho]^{\ell_1} \\
\ldots
\end{align*}
(Local) Transfer Functions (extended)

\[ \text{gen}_{LV}([x := a]^\ell) = FV(a) \]
\[ \text{gen}_{LV}([x := \rho]^\ell) = \emptyset \]
\[ \text{gen}_{LV}([\text{skip}]^\ell) = \emptyset \]
\[ \text{gen}_{LV}([b]^\ell) = FV(b) \]

\[ \text{kill}_{LV}([x := a]^\ell) = \{x\} \]
\[ \text{kill}_{LV}([x := \rho]^\ell) = \{x\} \]
\[ \text{kill}_{LV}([\text{skip}]^\ell) = \emptyset \]
\[ \text{kill}_{LV}([b]^\ell) = \emptyset \]

\[ f_{LV}^\ell : \mathcal{P}(\text{Var}_*) \rightarrow \mathcal{P}(\text{Var}_*) \]

\[ f_{LV}^\ell(X) = X \setminus \text{kill}_{LV}([B]^\ell) \cup \text{gen}_{LV}([B]^\ell) \]
(Local) Transfer Functions (extended)

\[
\begin{align*}
\text{gen}_{LV}([x := a]) & = \text{FV}(a) \\
\text{gen}_{LV}([x \Leftarrow \rho]) & = \emptyset \\
\text{gen}_{LV}([\text{skip}]) & = \emptyset \\
\text{gen}_{LV}([b]) & = \text{FV}(b) \\
\text{kill}_{LV}([x := a]) & = \{x\} \\
\text{kill}_{LV}([x \Leftarrow \rho]) & = \{x\} \\
\text{kill}_{LV}([\text{skip}]) & = \emptyset \\
\text{kill}_{LV}([b]) & = \emptyset
\end{align*}
\]

\[
f^\text{LV}_\ell : \mathcal{P}(\text{Var}_*) \rightarrow \mathcal{P}(\text{Var}_*)
\]

\[
f^\text{LV}_\ell (X) = X \setminus \text{kill}_{LV}([B]) \cup \text{gen}_{LV}([B])
\]
(Local) Transfer Functions (extended)

\[ gen_{LV}([x := a]) = FV(a) \]
\[ gen_{LV}([x := \rho]) = \emptyset \]
\[ gen_{LV}([\text{skip}]) = \emptyset \]
\[ gen_{LV}([b]) = FV(b) \]

\[ kill_{LV}([x := a]) = \{x\} \]
\[ kill_{LV}([x := \rho]) = \{x\} \]
\[ kill_{LV}([\text{skip}]) = \emptyset \]
\[ kill_{LV}([b]) = \emptyset \]

\[ f^L_{LV} : \mathcal{P}(\text{Var}_\star) \to \mathcal{P}(\text{Var}_\star) \]
\[ f^L_{LV}(X) = X \setminus kill_{LV}([B]) \cup gen_{LV}([B]) \]
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.
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 \textbf{branch(ing) probabilities} when we construct the probabilistic control flow.
Example

Consider, for example, instead of

\[
\begin{align*}
[x & := 1] \mod 4; \\
[y & := 2] \mod 4; \\
[x & := x + y \mod 4] \mod 4; \\
\text{if } [x > 2] & \text{ then } [z := x] \text{ else } [z := y] \mod 4. \\
\end{align*}
\]

a probabilistic program like:

\[
\begin{align*}
[x & := \{0, 1\}] \mod 4; \\
[y & := \{0, 1, 2, 3\}] \mod 4; \\
[x & := x + y \mod 4] \mod 4; \\
\text{if } [x > 2] & \text{ then } [z := x] \text{ else } [z := y] \mod 4. \\
\end{align*}
\]
Consider, for example, instead of

\[
\begin{align*}
[x & := 1]^{1}; \\
y & := 2]^{2}; \\
x & := x + y \mod 4]^{3}; \\
\text{if } [x > 2]^{4} & \text{ then } [z := x]^{5} \text{ else } [z := y]^{6} \text{ fi}
\end{align*}
\]

a probabilistic program like:

\[
\begin{align*}
[x & := \{0, 1\}]^{1}; \\
y & := \{0, 1, 2, 3\}]^{2}; \\
x & := x + y \mod 4]^{3}; \\
\text{if } [x > 2]^{4} & \text{ then } [z := x]^{5} \text{ else } [z := y]^{6} \text{ fi}
\end{align*}
\]
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.
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.
We can also use the classical control flow relation (as long as we do not consider a randomised \texttt{choose} statement).

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

(i) We want to express \textit{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 \textit{branching probabilities} when tests are evaluated.

(iv) We often also need probabilistic versions of the transfer functions.
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.
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.
We can also use the classical control flow relation (as long as we do not consider a randomised \texttt{choose} statement).

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

(i) We want to express \textit{probabilities of properties} not just (safe approximations) of properties.

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

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

(iv) We often also need probabilistic versions of the \textit{transfer functions}.
Local Transfer

When we look at the local transfer functions $f_\ell$ 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 $f_\ell$ 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 $f_\ell$ 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 $f_\ell$ 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 $f_\ell$ 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.
For an abstraction $A : \mathcal{V}(\text{State}) \to \mathcal{V}(L)$ we get for a concrete transfer operator $F$ an abstract, (least-square) optimal estimate via $F^\# = A^\dagger FA$ in analogy to Abstract Interpretation.

**Definition**

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

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

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

where $P_A$ and $P_G$ denote orthogonal projections onto the ranges of $A$ and $G.$
For an abstraction $A : \mathcal{V}(\text{State}) \rightarrow \mathcal{V}(L)$ we get for a concrete transfer operator $F$ an abstract, (least-square) optimal estimate via $F^\# = A^\dagger FA$ in analogy to Abstract Interpretation.

**Definition**

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

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

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

where $P_A$ and $P_G$ denote orthogonal projections onto the ranges of $A$ and $G$. 
Branch Probabilities

**Definition**

Given a program $S_\ell$ with $\text{init}(S_\ell) = \ell$ and a probability distribution $\rho$ on $\text{State}$, the probability $p_{\ell,\ell'}(\rho)$ that the control is flowing from $\ell$ to $\ell'$ 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 $P(b)$ which filter out states which do not pass the test.
Definition

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

\[
p_{\ell,\ell'}(\rho) = \sum_{s} \{ p \cdot \rho(s) | \exists s' \text{ s.t. } \langle S_\ell, s \rangle \Rightarrow p \langle S_{\ell'}, s' \rangle \}.
\]

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

One can implement the test \( b \) as projections \( \mathbb{P}(b) \) which filter out states which do not pass the test.
Given a program $S_\ell$ with $\text{init}(S_\ell) = \ell$ and a probability distribution $\rho$ on $\text{State}$, the probability $p_{\ell,\ell'}(\rho)$ that the control is flowing from $\ell$ to $\ell'$ 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 $P(b)$ which filter out states which do not pass the test.
Consider the simple program with $x \in \{0, 1, 2\}$

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

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

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

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 $\rho P(x >= 1) = (0, p_1, p_2)$ and

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

for the else branch, with $P^\perp = I - P$:

$$p_{1,3}(\rho) = \|\rho \cdot P^\perp(x >= 1)\|_1 = p_0.$$
Tests and Branch Probabilities (Concrete)

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

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

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

\[
P(x >= 1) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad P(x >= 1)^\perp = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}
\]

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 \( \rho P(x >= 1) = (0, p_1, p_2) \) and

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

for the else branch, with \( P^\perp = I - P \):

\[
p_{1,3}(\rho) = \|\rho \cdot P^\perp(x >= 1)\|_1 = p_0.
\]
Consider the simple program with $x \in \{0, 1, 2\}$

$$\text{if } [x \geq 1] \text{ then } [x := x - 1] \text{ else } [\text{skip}] \text{ fi}$$

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

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

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 $\rho P(x \geq 1) = (0, p_1, p_2)$ and

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

for the else branch, with $P^\perp = I - P$:

$$p_{1,3}(\rho) = \| \rho \cdot P^\perp(x \geq 1) \|_1 = p_0.$$
Consider the simple program with \( x \in \{0, 1, 2\} \)

\[
\text{if } [x \geq 1] \text{ then } [x := x - 1] \text{ else } \text{[skip]} \text{ fi}
\]

Then the test \( b = (x \geq 1) \) is represented by the projection:

\[
P(x \geq 1) = \begin{pmatrix}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{pmatrix}
\quad \text{and} \quad
P(x \geq 1)^\perp = \begin{pmatrix}
1 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{pmatrix}
\]

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 \( \rho P(x \geq 1) = (0, p_1, p_2) \) and

\[
\rho_{1,2}(\rho) = \|\rho \cdot P(x \geq 1)\|_1 = p_1 + p_2,
\]

for the else branch, with \( P^\perp = I - P \):

\[
\rho_{1,3}(\rho) = \|\rho \cdot P^\perp(x \geq 1)\|_1 = p_0.
\]
If we consider abstract states $\rho^\# \in \mathcal{V}(L)$ we need abstract versions $P(b)^\#$ of $P(b)$ to compute the branch probabilities. In doing so we must guarantee that for $\rho^\# = \rho A$:

\[
\begin{align*}
\rho P(b)A & \overset{!}{=} \rho^# P^#(b) \\
\rho P(b)A & \overset{!}{=} \rho A P^#(b) \\
P(b)A & \overset{!}{=} A P^#(b)
\end{align*}
\]

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

\[
\begin{align*}
A^\dagger P(b)A & = A^\dagger A P^#(b) \\
A^\dagger P(b)A & = P^#(b)
\end{align*}
\]

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 $P(b)^#$ of $P(b)$ to compute the branch probabilities. In doing so we must guarantee that for $\rho^# = \rho A$:

$$\rho P(b)A \overset{!}{=} \rho^# P^#(b)$$
$$\rho P(b)A \overset{!}{=} \rho A P^#(b)$$
$$P(b)A \overset{!}{=} A P^#(b)$$

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

$$A^\dagger P(b)A = A^\dagger A P^#(b)$$
$$A^\dagger P(b)A = P^#(b)$$

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 $P(b)^#$ of $P(b)$ to compute the branch probabilities. In doing so we must guarantee that for $\rho^# = \rho A$:

$$\rho P(b) A \overset{!}{=} \rho^# P^#(b)$$
$$\rho P(b) A \overset{!}{=} \rho A P^#(b)$$
$$P(b) A \overset{!}{=} A P^#(b)$$

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

$$A^\dagger P(b) A = A^\dagger A P^#(b)$$
$$A^\dagger P(b) A = P^#(b)$$

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 $P(b)^\#$ of $P(b)$ to compute the branch probabilities. In doing so we must guarantee that for $\rho^\# = \rho A$:

$$\rho P(b)A \overset{!}{=} \rho^# P^#(b)$$

$$\rho P(b)A \overset{!}{=} \rho A P^#(b)$$

$$P(b)A \overset{!}{=} A P^#(b)$$

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

$$A^\dagger P(b)A = A^\dagger A P^#(b)$$

$$A^\dagger P(b)A = P^#(b)$$

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 $P(b)^#$ of $P(b)$ to compute the branch probabilities. In doing so we must guarantee that for $\rho^# = \rho A$:

$$\rho P(b)A \overset{!}{=} \rho^# P^#(b)$$
$$\rho P(b)A \overset{!}{=} \rho A P^#(b)$$
$$P(b)A \overset{!}{=} A P^#(b)$$

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

$$A^\dagger P(b)A = A^\dagger A P^#(b)$$
$$A^\dagger P(b)A = P^#(b)$$

We get estimates for the abstract branch probabilities.
An Example: Prime Numbers are Odd

Consider the following program that counts the prime numbers.

\[
\begin{align*}
  &i := 2; \\
  \text{while } [i < 100] \text{ do} \\
  &\text{if } [\text{prime}(i)] \text{ then } [p := p + 1] \\
  &\text{else } [\text{skip}] \text{ fi;} \\
  &[i := i + 1] \\
  \text{od}
\end{align*}
\]

Essential is the abstract branch probability for \([.]^3:\)

\[
P(\text{prime}(i)) \# = A_e^\dagger P(\text{prime}(i))A_e,
\]
An Example: Prime Numbers are Odd

Consider the following program that counts the prime numbers.

\[
\begin{align*}
[i := 2] &; \\
\text{while } [i < 100] &\text{ do} \\
\text{if } [\text{prime}(i)] &\text{ then } [p := p + 1] \\
\text{else } [\text{skip}] &\text{ fi;} \\
i &:= i + 1 \\
\text{od}
\end{align*}
\]

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

\[
P(\text{prime}(i))# = A_e^t P(\text{prime}(i)) A_e,
\]
An Example: Abstraction

Test operators:

\[ P_e = (P(\text{even}(n)))_{ii} = \begin{cases} 1 & \text{if } i = 2k \\ 0 & \text{otherwise} \end{cases} \]

\[ P_p = (P(\text{prime}(n)))_{ii} = \begin{cases} 1 & \text{if } \text{prime}(i) \\ 0 & \text{otherwise} \end{cases} \]

Abstraction Operators:

\[ (A_e)_{ij} = \begin{cases} 1 & \text{if } i = 2k + 1 \land j = 2 \\ 1 & \text{if } i = 2k \land j = 1 \\ 0 & \text{otherwise} \end{cases} \]

\[ (A_p)_{ij} = \begin{cases} 1 & \text{if } \text{prime}(i) \land j = 2 \\ 1 & \text{if } \neg\text{prime}(i) \land j = 1 \\ 0 & \text{otherwise} \end{cases} \]
An Example: Abstraction

Test operators:

\[ P_e = (P(\text{even}(n)))_{ii} = \begin{cases} 1 & \text{if } i = 2k \\ 0 & \text{otherwise} \end{cases} \]

\[ P_p = (P(\text{prime}(n)))_{ii} = \begin{cases} 1 & \text{if prime}(i) \\ 0 & \text{otherwise} \end{cases} \]

Abstraction Operators:

\[ (A_e)_{ij} = \begin{cases} 1 & \text{if } i = 2k + 1 \land j = 2 \\ 1 & \text{if } i = 2k \land j = 1 \\ 0 & \text{otherwise} \end{cases} \]

\[ (A_p)_{ij} = \begin{cases} 1 & \text{if prime}(i) \land j = 2 \\ 1 & \text{if } \neg\text{prime}(i) \land j = 1 \\ 0 & \text{otherwise} \end{cases} \]
An Example: Abstract Branch Probability

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

<table>
<thead>
<tr>
<th></th>
<th>(A^\dagger e P_p A_e)</th>
<th>(A^\dagger e P \perp_p A_e)</th>
<th>(A^\dagger e P e A_p)</th>
<th>(A^\dagger e P \perp e A_p)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(n = 10)</td>
<td>(\begin{pmatrix} 0.20 &amp; 0.00 \ 0.00 &amp; 0.60 \end{pmatrix})</td>
<td>(\begin{pmatrix} 0.80 &amp; 0.00 \ 0.00 &amp; 0.40 \end{pmatrix})</td>
<td>(\begin{pmatrix} 0.25 &amp; 0.00 \ 0.00 &amp; 0.67 \end{pmatrix})</td>
<td>(\begin{pmatrix} 0.75 &amp; 0.00 \ 0.00 &amp; 0.33 \end{pmatrix})</td>
</tr>
<tr>
<td>(n = 100)</td>
<td>(\begin{pmatrix} 0.02 &amp; 0.00 \ 0.00 &amp; 0.48 \end{pmatrix})</td>
<td>(\begin{pmatrix} 0.98 &amp; 0.00 \ 0.00 &amp; 0.52 \end{pmatrix})</td>
<td>(\begin{pmatrix} 0.04 &amp; 0.00 \ 0.00 &amp; 0.65 \end{pmatrix})</td>
<td>(\begin{pmatrix} 0.96 &amp; 0.00 \ 0.00 &amp; 0.35 \end{pmatrix})</td>
</tr>
<tr>
<td>(n = 1000)</td>
<td>(\begin{pmatrix} 0.00 &amp; 0.00 \ 0.00 &amp; 0.33 \end{pmatrix})</td>
<td>(\begin{pmatrix} 1.00 &amp; 0.00 \ 0.00 &amp; 0.67 \end{pmatrix})</td>
<td>(\begin{pmatrix} 0.01 &amp; 0.00 \ 0.00 &amp; 0.60 \end{pmatrix})</td>
<td>(\begin{pmatrix} 0.99 &amp; 0.00 \ 0.00 &amp; 0.40 \end{pmatrix})</td>
</tr>
<tr>
<td>(n = 10000)</td>
<td>(\begin{pmatrix} 0.00 &amp; 0.00 \ 0.00 &amp; 0.25 \end{pmatrix})</td>
<td>(\begin{pmatrix} 1.00 &amp; 0.00 \ 0.00 &amp; 0.75 \end{pmatrix})</td>
<td>(\begin{pmatrix} 0.00 &amp; 0.00 \ 0.00 &amp; 0.57 \end{pmatrix})</td>
<td>(\begin{pmatrix} 1.00 &amp; 0.00 \ 0.00 &amp; 0.43 \end{pmatrix})</td>
</tr>
</tbody>
</table>

The entries in the upper left corner of \(A^\dagger e P_p A_e\) 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 \(I\).
Similar to classical DFA we formulate linear equations:

\[
\text{Analysis}_\circ(\ell) = \text{Analysis}_\circ(\ell) \cdot F^\#_\ell
\]

\[
\text{Analysis}_\circ(\ell) = \begin{cases} 
i, & \text{if } \ell \in E \\ \sum \{ \text{Analysis}_\circ(\ell') \cdot P(\ell', \ell)^\# \mid (\ell', \ell) \in F \}, & \text{else}
\end{cases}
\]

A simpler version can be obtained by static branch prediction:

\[
\text{Analysis}_\circ(\ell) = \sum \{ p_{\ell', \ell} \cdot \text{Analysis}_\circ(\ell') \mid (\ell', \ell) \in F \}
\]

Abstract branch probabilities, i.e. estimates for the test operators \( P(\ell', \ell)^\# \), 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:

\[
\text{Analysis}_\circ (\ell) = \text{Analysis}_\circ (\ell) \cdot F^\#_\ell
\]

\[
\text{Analysis}_\circ (\ell) = \begin{cases} 
\iota, & \text{if } \ell \in E \\
\sum \{ \text{Analysis}_\circ (\ell') \cdot P(\ell', \ell)^\# | (\ell', \ell) \in F \}, & \text{else}
\end{cases}
\]

A simpler version can be obtained by static branch prediction:

\[
\text{Analysis}_\circ (\ell) = \sum \{ p_{\ell', \ell} \cdot \text{Analysis}_\circ (\ell') | (\ell', \ell) \in F \}
\]

Abstract branch probabilities, i.e. estimates for the test operators \( P(\ell', \ell)^\# \), 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:

\[
\text{Analysis}_\bullet(\ell) = \text{Analysis}_\circ(\ell) \cdot F^\#_\ell
\]

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

A simpler version can be obtained by static branch prediction:

\[
\text{Analysis}_\circ(\ell) = \sum \{ p_{\ell',\ell} \cdot \text{Analysis}_\bullet(\ell') \mid (\ell',\ell) \in F \}
\]

Abstract branch probabilities, i.e. estimates for the test operators \( P(\ell',\ell)^\# \), can be estimated also via a different analysis \( \text{Prob} \), in a first phase before the actual Analysis.
Coming back to our previous example and its $LV$ analysis:

\[
\begin{align*}
[x & := \{0, 1\}]^1; 
[y & := \{0, 1, 2, 3\}]^2; 
[x & := x + y \mod 4]^3; 
\text{if } [x > 2]^4 \text{ then } [z & := x]^5 \text{ else } [z & := y]^6 \text{ fi}
\end{align*}
\]

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.

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

We define the abstract transfers for our four blocks a

\[
F_\ell = F_\ell^{LV} : \mathcal{V}(\{0, 1\}) \otimes |\text{Var}| \rightarrow \mathcal{V}(\{0, 1\}) \otimes |\text{Var}|.
\]
Live Variable Analysis: Example

Coming back to our previous example and its \( LV \) analysis:

\[
\begin{align*}
[x & = \{0, 1\}]^1; \quad [y & = \{0, 1, 2, 3\}]^2; \quad [x := x + y \mod 4]^3; \\
\text{if } [x > 2]^4 \text{ then } [z := x]^5 \text{ else } [z := y]^6 \text{ fi}
\end{align*}
\]

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.

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

We define the abstract transfers for our four blocks a

\[
F_{\ell^L} = F_{\ell^L}^{LV} : \mathcal{V}(\{0, 1\})^{\bigotimes |\text{Var}|} \rightarrow \mathcal{V}(\{0, 1\})^{\bigotimes |\text{Var}|}
\]
Coming back to our previous example and its $LV$ analysis:

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

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.

\[
L = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \ \text{and} \ K = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}.
\]

We define the abstract transfers for our four blocks $a$

\[
F_{\ell} = F_{\ell}^{LV} : \mathcal{V}({0, 1}) \otimes \mathcal{Var} \rightarrow \mathcal{V}({0, 1}) \otimes \mathcal{Var}
\]
Coming back to our previous example and its $LV$ analysis:

\[
[x := \{0, 1\}]^1; [y := \{0, 1, 2, 3\}]^2; [x := x + y \mod 4]^3;
\]

\[
\text{if } [x > 2]^4 \text{ then } [z := x]^5 \text{ else } [z := y]^6 \text{ fi}
\]

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.

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

We define the abstract transfers for our four blocks as

\[
F_\ell = F_\ell^{LV} : \mathcal{V}(\{0, 1\})^\Var \rightarrow \mathcal{V}(\{0, 1\})^\Var
\]
Transfer Functions for Live Variables

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

\[F^{\ell} = \bigotimes_{x_i \in \text{Var}} X_i \text{ with } X_i = \begin{cases} L & \text{if } x_i \in \text{FV}(a) \\ K & \text{if } x_i = x \land x_i \not\in \text{FV}(a) \\ I & \text{otherwise.} \end{cases}\]

and for tests \([b]^{\ell}\)

\[F^{\ell} = \bigotimes_{x_i \in \text{Var}} X_i \text{ with } X_i = \begin{cases} L & \text{if } x_i \in \text{FV}(b) \\ I & \text{otherwise.} \end{cases}\]

For \([\text{skip}]^{\ell}\) and \([x \neq \rho]^{\ell}\) have \(F^{\ell} = \bigotimes_{x_i \in \text{Var}} I\).
Transfer Functions for Live Variables

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

$$F^{\ell} = \bigotimes_{x_i \in \text{Var}} X_i \text{ with } X_i = \begin{cases} L & \text{if } x_i \in \text{FV}(a) \\ K & \text{if } x_i = x \land x_i \not\in \text{FV}(a) \\ I & \text{otherwise.} \end{cases}$$

and for tests $[b]^{\ell}$

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

For $[\text{skip}]^{\ell}$ and $[x \not= \rho]^{\ell}$ have $F^{\ell} = \bigotimes_{x_i \in \text{Var}} I.$
Transfer Functions for Live Variables

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

\[
F^{\ell} = \bigotimes_{x_i \in \text{Var}} X_i \quad \text{with} \quad X_i = \begin{cases} 
L & \text{if } x_i \in \text{FV}(a) \\
K & \text{if } x_i = x \land x_i \notin \text{FV}(a) \\
I & \text{otherwise.}
\end{cases}
\]

and for tests \([b]^{\ell}\)

\[
F^{\ell} = \bigotimes_{x_i \in \text{Var}} X_i \quad \text{with} \quad X_i = \begin{cases} 
L & \text{if } x_i \in \text{FV}(b) \\
I & \text{otherwise.}
\end{cases}
\]

For \([\text{skip}]^{\ell}\) and \([x = \rho]^{\ell}\) have \(F^{\ell} = \bigotimes_{x_i \in \text{Var}} I\).
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 \(\mathcal{V}(\{0, 1, 2, 3\})^\otimes 3 = \mathbb{R}^{43} = \mathbb{R}^{64}\).

The abstraction we use when we compute the concrete branch probabilities is \(A = I \otimes I \otimes A_f\), with \(A_f = (1, 1, 1, 1)^t\) 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 \(F_5^\# = F_6^\# = I\).
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 $A = I \otimes I \otimes A_f$, with $A_f = (1, 1, 1, 1)^t$ 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 $F_5^# = F_6^# = I$. 

Bolzano, 22-26 August 2016 ESSLLI’16
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 $A = I \otimes I \otimes A_f$, with $A_f = (1, 1, 1, 1)^t$ 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 $F_5^\# = F_6^\# = I$. 
(Abstract) Transfer Operators

\[
F_{1}^{\#} = \begin{pmatrix}
\frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\
\frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\
0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 \\
0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} \\
0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0
\end{pmatrix}
\]
(Abstract) Transfer Operators

\[
\begin{pmatrix}
\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} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\
\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} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 \\
0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 \\
0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
\end{pmatrix}
\]
\[
F_3^\# = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{pmatrix}
\]
(Abstract) Transfer Operators

$$P_4^\# = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$
The pre-processing probability analysis via equations:

\[
\begin{align*}
\text{Prob}_{\text{entry}}(1) &= \rho \\
\text{Prob}_{\text{entry}}(2) &= \text{Prob}_{\text{exit}}(1) \\
\text{Prob}_{\text{entry}}(3) &= \text{Prob}_{\text{exit}}(2) \\
\text{Prob}_{\text{entry}}(4) &= \text{Prob}_{\text{exit}}(3) \\
\text{Prob}_{\text{entry}}(5) &= \text{Prob}_{\text{exit}}(4) \cdot \mathbf{P}_4^# \\
\text{Prob}_{\text{entry}}(6) &= \text{Prob}_{\text{exit}}(4) \cdot (\mathbf{I} - \mathbf{P}_4^#)
\end{align*}
\]
The pre-processing probability analysis via equations:

\[
\begin{align*}
Prob_{exit}(1) & = Prob_{entry}(1) \cdot F_1^# \\
Prob_{exit}(2) & = Prob_{entry}(1) \cdot F_2^# \\
Prob_{exit}(3) & = Prob_{entry}(1) \cdot F_3^# \\
Prob_{exit}(4) & = Prob_{entry}(4) \\
Prob_{exit}(5) & = Prob_{entry}(5) \\
Prob_{exit}(6) & = Prob_{entry}(6)
\end{align*}
\]
The pre-processing probability analysis via equations:

\[
\begin{align*}
\text{Prob}_{\text{exit}}(1) &= \text{Prob}_{\text{entry}}(1) \cdot F_1^# \\
\text{Prob}_{\text{exit}}(2) &= \text{Prob}_{\text{entry}}(1) \cdot F_2^# \\
\text{Prob}_{\text{exit}}(3) &= \text{Prob}_{\text{entry}}(1) \cdot F_3^# \\
\text{Prob}_{\text{exit}}(4) &= \text{Prob}_{\text{entry}}(4) \\
\text{Prob}_{\text{exit}}(5) &= \text{Prob}_{\text{entry}}(5) \\
\text{Prob}_{\text{exit}}(6) &= \text{Prob}_{\text{entry}}(6)
\end{align*}
\]

reduce to:

\[
\begin{align*}
\text{Prob}_{\text{entry}}(5) &= \rho \cdot F_1^# \cdot F_2^# \cdot F_3^# \cdot P_4^# \\
\text{Prob}_{\text{entry}}(6) &= \rho \cdot F_1^# \cdot F_2^# \cdot F_3^# \cdot P_4^#
\end{align*}
\]
The pre-processing probability analysis via equations:

\[
\begin{align*}
\text{Prob}_{\text{exit}}(1) &= \text{Prob}_{\text{entry}}(1) \cdot F_1^# \\
\text{Prob}_{\text{exit}}(2) &= \text{Prob}_{\text{entry}}(1) \cdot F_2^# \\
\text{Prob}_{\text{exit}}(3) &= \text{Prob}_{\text{entry}}(1) \cdot F_3^# \\
\text{Prob}_{\text{exit}}(4) &= \text{Prob}_{\text{entry}}(4) \\
\text{Prob}_{\text{exit}}(5) &= \text{Prob}_{\text{entry}}(5) \\
\text{Prob}_{\text{exit}}(6) &= \text{Prob}_{\text{entry}}(6)
\end{align*}
\]

reduce to:

\[
\begin{align*}
\text{Prob}_{\text{entry}}(5) &= \rho \cdot F_1^# \cdot F_2^# \cdot F_3^# \cdot P_4^# \\
\text{Prob}_{\text{entry}}(6) &= \rho \cdot F_1^# \cdot F_2^# \cdot F_3^# \cdot P_4^#
\end{align*}
\]

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


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

\[
\begin{align*}
LV_{\text{entry}}(1) &= LV_{\text{exit}}(1) \cdot (K \otimes I \otimes I) \\
LV_{\text{entry}}(2) &= LV_{\text{exit}}(2) \cdot (I \otimes K \otimes I) \\
LV_{\text{entry}}(3) &= LV_{\text{exit}}(3) \cdot (L \otimes L \otimes I) \\
LV_{\text{entry}}(4) &= LV_{\text{exit}}(4) \cdot (L \otimes I \otimes I) \\
LV_{\text{entry}}(5) &= LV_{\text{exit}}(5) \cdot (L \otimes I \otimes K) \\
LV_{\text{entry}}(6) &= LV_{\text{exit}}(6) \cdot (I \otimes L \otimes K)
\end{align*}
\]
Data Flow Equations

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

\[
\begin{align*}
LV_{exit}(1) &= LV_{entry}(2) \\
LV_{exit}(2) &= LV_{entry}(3) \\
LV_{exit}(3) &= LV_{entry}(4) \\
LV_{exit}(4) &= p_{4,5}LV_{entry}(5) + p_{4,6}LV_{entry}(6) \\
LV_{exit}(5) &= (1, 0) \otimes (1, 0) \otimes (1, 0) \\
LV_{exit}(6) &= (1, 0) \otimes (1, 0) \otimes (1, 0)
\end{align*}
\]
The solution to the $LV$ equations is then given by:

$$LV_{entry}(1) = (1, 0) \otimes (1, 0) \otimes (1, 0)$$

$$LV_{entry}(2) = (0, 1) \otimes (1, 0) \otimes (1, 0)$$

$$LV_{entry}(3) = 0.25 \cdot (0, 1) \otimes (0, 1) \otimes (1, 0) +$$
$+ 0.75 \cdot (0, 1) \otimes (0, 1) \otimes (1, 0)$$

$$LV_{entry}(3) = (0, 1) \otimes (0, 1) \otimes (1, 0)$$

$$LV_{entry}(4) = 0.25 \cdot (0, 1) \otimes (1, 0) \otimes (1, 0) +$$
$+ 0.75 \cdot (0, 1) \otimes (0, 1) \otimes (1, 0)$$

$$LV_{entry}(5) = (0, 1) \otimes (1, 0) \otimes (1, 0)$$

$$LV_{entry}(6) = (1, 0) \otimes (0, 1) \otimes (1, 0)$$
The solution to the $LV$ equations is then given by:

\[
\begin{align*}
LV_{\text{exit}}(1) &= (0, 1) \otimes (1, 0) \otimes (1, 0) \\
LV_{\text{exit}}(2) &= (0, 1) \otimes (0, 1) \otimes (1, 0) \\
LV_{\text{exit}}(3) &= 0.25 \cdot (0, 1) \otimes (1, 0) \otimes (1, 0) + 0.75 \cdot (0, 1) \otimes (0, 1) \otimes (1, 0) \\
LV_{\text{exit}}(4) &= 0.25 \cdot (0, 1) \otimes (1, 0) \otimes (1, 0) + 0.75 \cdot (1, 0) \otimes (0, 1) \otimes (1, 0) \\
LV_{\text{exit}}(5) &= (1, 0) \otimes (1, 0) \otimes (1, 0) \\
LV_{\text{exit}}(6) &= (1, 0) \otimes (1, 0) \otimes (1, 0)
\end{align*}
\]
The Moore-Penrose Pseudo-Inverse

**Definition**

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

**Definition**

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

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

**Theorem**

Let $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. Then $A^\dagger b$ is the minimal least squares solution to $Ax = b$. 
The Moore-Penrose Pseudo-Inverse

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

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

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

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

Bolzano, 22-26 August 2016 
ESSLLI'16 
Probabilistic Program Analysis 
Slide 29 of 45
The Moore-Penrose Pseudo-Inverse

Definition

Let \( C \) and \( D \) be two finite-dimensional vector spaces and \( A : C \to D \) a linear map. Then the linear map \( A^\dagger = G : D \to C \) is the Moore-Penrose pseudo-inverse of \( A \) iff \( A \circ G = P_A \) and \( G \circ A = P_G \), where \( P_A \) and \( P_G \) denote orthogonal projections onto the ranges of \( A \) and \( G \).

Definition

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

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

Theorem

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

$$\mathbf{T}^\# : \mathcal{D} \to \mathcal{D}$$

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 $C$, $D$ . . .
- Concrete and abstract semantics are linear operators $T$ . . .

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

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

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

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

This gives a “smaller” DTMC via the abstracted generator $T^\#$. 
Probabilistic Abstract Interpretation is based on:

- Concrete and abstract domains are linear spaces $C, D$...
- Concrete and abstract semantics are linear operators $T$...

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

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

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

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

This gives a “smaller” DTMC via the abstracted generator $T^\#$. 
Probabilistic Abstract Interpretation is based on:

- Concrete and abstract domains are linear spaces $C$, $D$...
- Concrete and abstract semantics are linear operators $T$...

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

$$T^\# : D \to D$$

of a concrete semantics $T : C \to C$

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

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

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

$$T^\# : \mathcal{D} \rightarrow \mathcal{D}$$

of a concrete semantics $T : \mathcal{C} \rightarrow \mathcal{C}$

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

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

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

Bolzano, 22-26 August 2016  ESSLLI’16  Probabilistic Program Analysis  Slide 30 of 45
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 $T^\#$:

$$T^\#: \mathcal{D} \to \mathcal{D}$$

of a concrete semantics $T: \mathcal{C} \to \mathcal{C}$

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

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

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

- 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

- 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

- 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

- 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

- 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 $\beta$.
- Construct a matrix $X$ mapping models to the runs.
- Use $X^\dagger$ and $y$ to find a best estimator of the model.

**Theorem (Gauss-Markov)**

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

$$\hat{\beta} = y X^\dagger.$$
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 $\beta$.
- Construct a matrix $X$ mapping models to the runs.
- Use $X^\dagger$ and $y$ to find a best estimator of the model.

**Theorem (Gauss-Markov)**

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

$$\hat{\beta} = yX^\dagger.$$
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 \( \beta \)
- Construct a matrix \( X \) mapping models to the runs
- Use \( X^\dagger \) and \( y \) to find a best estimator of the model.

---

**Theorem (Gauss-Markov)**

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

\[ \hat{\beta} = yX^\dagger. \]
Using Statistics

Infer execution probabilities by observing some sample runs.

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

**Theorem (Gauss-Markov)**

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

$$\hat{\mathbf{\beta}} = \mathbf{y}\mathbf{X}^\dagger.$$
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 $\beta$
- Construct a matrix $X$ mapping models to the runs
- Use $X^\dagger$ and $y$ to find a best estimator of the model.

**Theorem (Gauss-Markov)**

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

\[ \hat{\beta} = yX^\dagger. \]
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 \( \beta \)
- Construct a matrix \( X \) mapping models to the runs
- Use \( X^\dagger \) and \( y \) to find a best estimator of the model.

**Theorem (Gauss-Markov)**

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

\[
\hat{\beta} = yX^\dagger.
\]
Modular Exponentiation

\[
s := 1;
i := 0;
\]

while \( i \leq w \) do
    if \( k[i] = 1 \) then
        \[
x := (s \times x) \mod n;
        \]
    else
        \[
r := s;
        \]
    fi;
    \[
s := r \times r;
    \]
    \[
i := i + 1;
    \]
end while;

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

\begin{verbatim}
    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;
\end{verbatim}

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

\[ \text{Diagram showing paths and fronts.} \]
Consider the following simple DTMC with parameters $p$ and $q$ in the real interval $[0, 1]$:

\[
T_{pq} = \begin{pmatrix}
    p & 1 - p \\
    1 - q & q
\end{pmatrix}
\]

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

$$
\begin{pmatrix}
1 - p & p \\
q & 1 - q
\end{pmatrix}
$$

This behaviour is essentially the one of the following program:

```plaintext
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 $[0, 1]$:

$$T_{pq} = \begin{pmatrix} p & 1 - p \\ 1 - q & q \end{pmatrix}$$

This behaviour is essentially the one of the following program:

```plaintext
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 $[0,1]$:

\[
\begin{align*}
\text{p} & \xrightarrow{1-p} 0 \\
\text{0} & \xrightarrow{1-q} 1 \\
\text{1} & \xrightarrow{q} \text{q} \\
\end{align*}
\]

\[T_{pq} = \begin{pmatrix} p & 1-p \\ 1-q & q \end{pmatrix} \]

This behaviour is essentially the one of the following program:

\[
\text{while (true) do} \\
\text{if } (x == 1) \\
\text{then } x \text{ ?=} \{\langle 0, p \rangle, \langle 1, 1-p \rangle\} \\
\text{else } x \text{ ?=} \{\langle 0, 1 - q \rangle, \langle 1, q \rangle\} \\
\text{fi} \\
\text{od}
\]
Observing Traces: Possible Parameters

Instantiating the parameters:

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

\[ T_{\frac{1}{2},1} = \begin{pmatrix} \frac{1}{2} & 1 \\ 0 & \frac{1}{2} \end{pmatrix} \]
Observing Traces: Possible Parameters

Instantiating the parameters:

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

\[ T_{\frac{1}{2},1} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ 0 & 1 \end{pmatrix} \]
Observing Traces: Possible Parameters

Instantiating the parameters:

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

\[ T_{1/2,1} = \begin{pmatrix} 1/2 & 1/2 \\ 0 & 1 \end{pmatrix} \]
Observing Traces: Possible Parameters

Instantiating the parameters:

\[ T_{0, \frac{1}{2}} = \begin{pmatrix} 0 & 1 \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} \]

\[ T_{\frac{1}{2}, \frac{1}{2}} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} \]
Observing Traces: Possible Parameters

Instantiating the parameters:

\[
T_{0, \frac{1}{2}} = \begin{pmatrix}
0 & 1 \\
\frac{1}{2} & \frac{1}{2}
\end{pmatrix}
\]

\[
T_{\frac{1}{2}, \frac{1}{2}} = \begin{pmatrix}
\frac{1}{2} & \frac{1}{2} \\
\frac{1}{2} & \frac{1}{2}
\end{pmatrix}
\]
Observing Traces: Possible Parameters

Instantiating the parameters:

\[ T_{0, \frac{1}{2}} = \begin{pmatrix} 0 & 1 \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} \]

\[ T_{\frac{1}{2}, \frac{1}{2}} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} \]
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**: $\mathbf{G} : \mathcal{D} \rightarrow \mathcal{C}$ 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**: $D = \mathcal{V}(\mathcal{M})$, with
  $\mathcal{M} = \{\langle s, p, q \rangle \mid s \in \{0, 1\}, p, q \in [0, 1]\}$

- **Concrete domain**: $C = \mathcal{V}(\mathcal{T})$ with
  $\mathcal{T} = \{0, 1\}^{+\infty}$ (execution traces)

- **Design matrix**: $G : D \rightarrow C$ associates to each instance model the corresponding distribution on traces

- Compute the Moore-Penrose pseudo-inverse $G^\dagger$ of $G$ to calculate the best estimators of the parameters $p$ and $q$. 
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 $q$. 
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.

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

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

$$C_3 = \mathcal{V}(\{0, 1\}^3) = \mathcal{V}(\{0, 1\}) \otimes^3 = (\mathbb{R}^2)^{\otimes 8} = \mathbb{R}^8$$

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

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

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

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

$$C_3 = \mathcal{V}(\{0, 1\}^3) = \mathcal{V}(\{0, 1\}) \otimes^3 = (\mathbb{R}^2)^\otimes^8 = \mathbb{R}^8$$

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

$$C_{10} = \mathcal{V}(\{0, 1\}^{10}) = \mathcal{V}(\{0, 1\}) \otimes^{10} = (\mathbb{R}^2)^\otimes^{10} = \mathbb{R}^{1024}$$
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 \( t = 3 \):

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

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 $\mathcal{D} = \mathbb{R}^9$

<table>
<thead>
<tr>
<th></th>
<th>s</th>
<th>p</th>
<th>q</th>
<th></th>
<th>s</th>
<th>p</th>
<th>q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1/2</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>1</td>
<td>1/2</td>
</tr>
<tr>
<td>1</td>
<td>1/2</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>1/2</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1/2</td>
<td>0</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1/2</td>
<td>0</td>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1/2</td>
<td>0</td>
<td>1/2</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Experiments: Trace Space $C_3 = \mathbb{R}^8$ and $C_{10} = \mathbb{R}^{1024}$

<table>
<thead>
<tr>
<th>Trace $C_3$</th>
<th>Trace $C_{10}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0 0 0 0 0 0 0 0 0 1</td>
</tr>
<tr>
<td>0 1 0</td>
<td>0 0 0 0 0 0 0 0 1 0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0 0 0 0 0 0 0 1 0 1</td>
</tr>
<tr>
<td>1 0 0</td>
<td>0 0 0 0 0 0 1 0 0 0</td>
</tr>
<tr>
<td>1 0 1</td>
<td>0 0 0 0 0 1 0 0 0 0</td>
</tr>
<tr>
<td>1 1 0</td>
<td>0 0 0 0 1 0 0 0 0 0</td>
</tr>
<tr>
<td>1 1 1</td>
<td>0 0 0 1 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>$\text{trace } \mathcal{C}_3$</th>
<th>$\text{trace } \mathcal{C}_{10}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0 0 0 0 0 0 0 0 0 0 0 1 0 0</td>
</tr>
<tr>
<td>0 1 0</td>
<td>0 0 0 0 0 0 0 0 0 1 0 0 0 0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0 0 0 0 0 0 0 0 0 1 0 0 1 0</td>
</tr>
<tr>
<td>1 0 0</td>
<td>0 0 0 0 0 0 0 0 0 1 0 1 0 0</td>
</tr>
<tr>
<td>1 0 1</td>
<td>0 0 0 0 0 0 0 0 0 1 1 0 0 1</td>
</tr>
<tr>
<td>1 1 0</td>
<td>0 0 0 0 0 0 0 0 0 1 1 1 0 0</td>
</tr>
<tr>
<td>1 1 1</td>
<td>0 0 0 0 0 0 0 0 0 1 1 1 1 0</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bolzano, 22-26 August 2016 ESSLII’16

Probabilistic Program Analysis

Slide 41 of 45
Experiments: Concretisation $G_3$

$$G_3 = \begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
\frac{1}{4} & \frac{1}{4} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 \\
0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 \\
\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$
Experiments: Regression $G_{3}^{\dagger}$ (Abstraction)

$$G_{3}^{\dagger t} = \begin{pmatrix}
0 & -\frac{2}{3} & \frac{11}{15} & -\frac{1}{15} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -\frac{1}{15} & \frac{11}{15} & -\frac{2}{3} & 0 \\
0 & \frac{4}{3} & \frac{1}{5} & -\frac{1}{5} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & -\frac{2}{3} & 0 \\
\frac{1}{3} & -\frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{11}{15} & -\frac{1}{15} & -\frac{2}{3} & 0 \\
0 & -\frac{2}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -\frac{1}{5} & \frac{1}{5} & 4 \frac{3}{3} & 0 \\
\frac{4}{3} & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{3} & -\frac{1}{3} & 0 & 0 \\
\frac{1}{3} & -\frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{11}{15} & -\frac{1}{5} & 4 \frac{3}{3} & 0 \\
0 & -\frac{2}{3} & -\frac{1}{15} & \frac{11}{15} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{3} & \frac{1}{3} \\
\frac{4}{3} & -\frac{1}{5} & \frac{1}{5} & 0 & 0 & 0 & -\frac{1}{3} & \frac{1}{3} \\
0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{3} & \frac{1}{3} \\
\frac{1}{3} & -\frac{1}{3} & 0 & 0 & 0 & 0 & -\frac{1}{3} & \frac{1}{3} \\
0 & 0 & 0 & 0 & -\frac{1}{3} & \frac{1}{3} & 0 & 0 \\
\end{pmatrix}$$
Numerical Experiments for $C_{10}$

For the model $p = 0, q = \frac{1}{2}$ we obtained (for different noise distortions $\varepsilon$) 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:

$$y_{G_{10}}^\dagger = (0, 0, 0, 0, 0, 0, 0.50, 0.49, 0, 0.01, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$$

$$y'_{G_{10}}^\dagger = (0, 0, 0, 0, 0, 0, 0.49, 0.50, 0.01, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$$

$$y''_{G_{10}}^\dagger = (0, 0, 0, 0, 0, 0, 0.43, 0.43, 0.07, 0.06, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$$

$$y'''_{G_{10}}^\dagger = (0, 0, 0.01, 0, 0, 0, 0.33, 0.35, 0.16, 0.16, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$$

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 $C_{10}$

For the model $p = 0, q = \frac{1}{2}$ we obtained (for different noise distortions $\varepsilon$) 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:

\[
\begin{align*}
 yG^\dagger_{10} &= (0, 0, 0, 0, 0, 0, 0.50, 0.49, 0, 0.01, 0, 0, 0, 0, 0, 0, 0) \\
 y'G^\dagger_{10} &= (0, 0, 0, 0, 0, 0, 0.50, 0.50, 0.01, 0, 0, 0, 0, 0, 0, 0, 0) \\
 y''G^\dagger_{10} &= (0, 0, 0, 0, 0, 0, 0.43, 0.43, 0.07, 0.06, 0, 0, 0, 0, 0, 0, 0, 0) \\
 y'''G^\dagger_{10} &= (0, 0, 0, 0, 0.01, 0, 0, 0.33, 0.35, 0.16, 0.16, 0, 0, 0, 0, 0, 0, 0, 0)
\end{align*}
\]

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 $C_{10}$

For the model $p = 0, q = \frac{1}{2}$ we obtained (for different noise distortions $\varepsilon$) 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:

\[
\begin{align*}
y_{G_{10}}^\dagger &= (0, 0, 0, 0, 0, 0, 0.50, 0.49, 0, 0.01, 0, 0, 0, 0, 0, 0, 0, 0) \\
y'_{G_{10}}^\dagger &= (0, 0, 0, 0, 0, 0, 0.49, 0.50, 0.01, 0, 0, 0, 0, 0, 0, 0, 0) \\
y''_{G_{10}}^\dagger &= (0, 0, 0, 0, 0, 0, 0.43, 0.43, 0.07, 0.06, 0, 0, 0, 0, 0, 0, 0, 0) \\
y'''_{G_{10}}^\dagger &= (0, 0, 0.01, 0, 0, 0, 0.33, 0.35, 0.16, 0.16, 0, 0, 0, 0, 0, 0, 0, 0)
\end{align*}
\]

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


Some References


Some References