Data-flow analyses can be re-formulated in a different scenario where correctness is guaranteed by construction.

Classically, the theory of Abstract Interpretation allows us to

- construct simplified (computable) abstract semantics
- construct approximate solutions
- obtain the correctness of the approximate solution by construction.
Notions of Approximation

In order theoretic structures we are looking for Safe Approximations

\[ s^* \sqsubseteq s \text{ or } s \sqsubseteq s^* \]

In quantitative, vector space structures we want Close Approximations

\[ \| s - s^* \| = \min_x \| s - x \| \]

Example: Function Approximation

Concrete and abstract domain are step-functions on \([a, b]\). The set of (real-valued) step-function \( T_n \) is based on the sub-division of the interval into \( n \) sub-intervals.
Close Approximations

Close vs Correct Approximations
Some problems may be have too costly solutions or be uncomputable on a concrete space (complete lattice). Find abstract descriptions on which computations are easier; then relate the concrete and abstract solutions.

**Definition**

Let $\mathcal{C} = (\mathcal{C}, \leq)$ and $\mathcal{D} = (\mathcal{D}, \sqsubseteq)$ be two partially ordered set. If there are two functions $\alpha : \mathcal{C} \rightarrow \mathcal{D}$ and $\gamma : \mathcal{D} \rightarrow \mathcal{C}$ such that for all $c \in \mathcal{C}$ and all $d \in \mathcal{D}$:

$$c \leq \mathcal{C} \gamma(d) \text{ iff } \alpha(c) \sqsubseteq d,$$

then $(\mathcal{C}, \alpha, \gamma, \mathcal{D})$ form a **Galois connection**.

**Galois Connections**

**Definition**

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

1. $\alpha \circ \gamma$ is **reductive** i.e. $\forall d \in \mathcal{D}, \alpha \circ \gamma(d) \leq_{\mathcal{D}} d$,
2. $\gamma \circ \alpha$ is **extensive** i.e. $\forall c \in \mathcal{C}, c \leq_{\mathcal{C}} \gamma \circ \alpha(c)$.

**Proposition**

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

1. $\alpha \circ \gamma \circ \alpha = \alpha$
2. $\gamma \circ \alpha \circ \gamma = \gamma$
General Construction

\[ A \xrightarrow{\alpha} A^\# \]
\[ B \xleftarrow{\gamma} B^\# \]
\[ f \downarrow \quad f^\# \downarrow \]
\[ \alpha' \quad \gamma' \]

Correct approximation:

\[ \alpha' \circ f \leq f^\# \circ \alpha. \]

Induced semantics:

\[ f^\# = \alpha \circ f \circ \gamma. \]

Probabilistic Abstraction Domains

A probabilistic domain is essentially a vector space which represents the distributions \( \text{Dist}(S) \) on the state space \( S \) of a probabilistic transition system, i.e. for finite state spaces

\[ \mathcal{V}(S) = \{ (\nu_s)_{s \in S} \mid \nu_s \in \mathbb{R} \}. \]

In the finite setting we can identify \( \mathcal{V}(S) \) with the Hilbert space \( \ell^2(S) \).

The notion of norm is essential for our treatment; we will consider normed vector spaces.
Norm and Operator Norm

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

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

with $0 \in \mathcal{V}$ the zero vector.

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

$$\|M\| = \sup_{v \in \mathcal{V}} \frac{\|M(v)\|}{\|v\|} = \sup_{\|v\|=1} \|M(v)\|.$$
Least Squares Solutions

**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 \( Ax = b \).

**Corollary**

Let \( P \) be an orthogonal projection on a finite dimensional vector space \( \mathcal{V} \). Then for any \( x \in \mathcal{V} \), \( Px \) is the unique closest vector in \( \mathcal{V} \) to \( x \) wrt the Euclidean norm.
Extraction Functions

An extraction function $\eta : C \mapsto D$ is a mapping from a set of values to their descriptions in $D$.
It is easy to show that

**Proposition**

Given an extraction function $\eta : C \mapsto D$, the quadruple $(\mathcal{P}(C), \alpha_\eta, \gamma_\eta, \mathcal{P}(D))$ is a Galois connection with $\alpha_\eta$ and $\gamma_\eta$ defined by:

$$\alpha_\eta(C') = \{ \eta(c) \mid c \in C' \} \text{ and } \gamma_\eta(D') = \{ v \mid \eta(v) \in D' \}$$

Vector Space Lifting

Free vector space construction on a set $S$:

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

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

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

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

**Support Set**: $\text{supp} : \mathcal{V}(C) \rightarrow \mathcal{P}(C)$

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

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

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

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

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

Proposition

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

Examples of Lifted Abstractions

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

$$A_p = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 0 & 1 \end{pmatrix} \quad A_p^\dagger = \begin{pmatrix} 2/n & 0 & 2/n & 0 & \cdots & 0 \\ 0 & 2/n & 0 & 2/n & \cdots & 2/n \end{pmatrix}$$

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

$$A_s = \begin{pmatrix} 1 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad A_s^\dagger = \begin{pmatrix} 1/n & \cdots & 1/n & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{pmatrix}$$
Concrete and abstract domain are step-functions on \([a, b]\).
The set of (real-valued) step-function \(T_n\) is based on the
sub-division of the interval into \(n\) sub-intervals.

Each step function in \(T_n\) corresponds to a vector in \(\mathbb{R}^n\), e.g.

\[
\begin{pmatrix}
5 & 5 & 6 & 7 & 8 & 4 & 3 & 2 & 8 & 6 & 6 & 7 & 9 & 8 & 8 & 7
\end{pmatrix}
\]
Approximation Estimates

Compute the least square error as

\[ \| f - f_{AG} \|. \]

\[ \| f - f_{A8}G_8 \| = 3.5355 \]
\[ \| f - f_{A4}G_4 \| = 5.3151 \]
\[ \| f - f_{A2}G_2 \| = 5.9896 \]
\[ \| f - f_{A1}G_1 \| = 7.6444 \]
Abstract Semantics

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

\[(A_1 \otimes A_2 \otimes \ldots \otimes A_n)^\dagger = A_1^\dagger \otimes A_2^\dagger \otimes \ldots \otimes A_n^\dagger\]

Via linearity we can construct \(T^\#\) in the same way as \(T\), i.e

\[T^\#(P) = \sum_{(i,p,j) \in F(P)} p_{ij} \cdot T^\#(\ell_i, \ell_j)\]

with local abstraction of individual variables:

\[T^\#(\ell_i, \ell_j) = (A_1^\dagger N_{i1} A_1) \otimes (A_2^\dagger N_{i2} A_2) \otimes \ldots \otimes (A_v^\dagger N_{iv} A_v) \otimes M_{ij}\]
Parity Analysis

Determine at each program point whether a variable is even or odd.

Parity Abstraction operator on $V(\{0, \ldots, n\})$ (with $n$ even):

\[
A_p = \begin{pmatrix}
1 & 0 \\
0 & 1 \\
1 & 0 \\
0 & 1 \\
\vdots & \vdots \\
0 & 1
\end{pmatrix}
\]

\[
A^\dagger = \begin{pmatrix}
\frac{2}{n} & 0 & 0 & \frac{2}{n} & \cdots & 0 \\
0 & \frac{2}{n} & 0 & \frac{2}{n} & \cdots & \frac{2}{n}
\end{pmatrix}
\]

Example

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

\[
T = U(m \leftarrow i) \otimes E(1, 2)
\]
\[
T^\# = U^\#(m \leftarrow i)
\]

\[
+ P(n > 1) \otimes E(2, 3)
\]
\[
+ P(n \leq 1) \otimes E(2, 5)
\]
\[
+ U(m \leftarrow m \times n) \otimes E(3, 4)
\]
\[
+ U(n \leftarrow n - 1) \otimes E(4, 2)
\]
\[
+ I \otimes E(5, 5)
\]
Abstract Semantics

Abstraction: \( A = A_p \otimes I \), i.e. \( m \) abstract (parity) but \( n \) concrete.

\[
\begin{align*}
T^# &= U^#(m \leftarrow 1) \otimes E(1, 2) \\
&\quad + P^#(n > 1) \otimes E(2, 3) \\
&\quad + P^#(n \leq 1) \otimes E(2, 5) \\
&\quad + U^#(m \leftarrow m \times n) \otimes E(3, 4) \\
&\quad + U^#(n \leftarrow n - 1) \otimes E(4, 2) \\
&\quad + I^# \otimes E(5, 5)
\end{align*}
\]

\[
U^#(m \leftarrow 1) = \begin{pmatrix} 1 & 0 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 0 & 0 & \ldots & 0 \end{pmatrix}
\]

Implementation

Implementation of concrete and abstract semantics of Factorial using \texttt{octave}. Ranges: \( n \in \{1, \ldots, d\} \) and \( m \in \{1, \ldots, d!\} \).

<table>
<thead>
<tr>
<th>( d )</th>
<th>( \dim(T(F)) )</th>
<th>( \dim(T^#(F)) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>45</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>140</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>625</td>
<td>50</td>
</tr>
<tr>
<td>5</td>
<td>3630</td>
<td>60</td>
</tr>
<tr>
<td>6</td>
<td>25235</td>
<td>70</td>
</tr>
<tr>
<td>7</td>
<td>201640</td>
<td>80</td>
</tr>
<tr>
<td>8</td>
<td>1814445</td>
<td>90</td>
</tr>
<tr>
<td>9</td>
<td>18144050</td>
<td>100</td>
</tr>
</tbody>
</table>

Using \texttt{uniform} initial distributions \( d_0 \) for \( n \) and \( m \).
Scalability

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

<table>
<thead>
<tr>
<th>$d$</th>
<th>even</th>
<th>odd</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.81818</td>
<td>0.18182</td>
</tr>
<tr>
<td>100</td>
<td>0.98019</td>
<td>0.019802</td>
</tr>
<tr>
<td>1000</td>
<td>0.99800</td>
<td>0.0019980</td>
</tr>
<tr>
<td>10000</td>
<td>0.99980</td>
<td>0.00019998</td>
</tr>
</tbody>
</table>

Live Variable Analysis

1: $[\text{skip}]^1[y \leftarrow 2 \times x]^1$
2: if $[\text{odd}(y)]^2$ then
3: $[x \leftarrow x + 1]^3$
4: else
5: $[y \leftarrow y + 1]^4$
6: fi
7: $[y \leftarrow y + 1]^5$

Classical Analysis: $LV_{entry}(2) = \{x, y\}$

Probabilistic Analysis: $LV_{entry}(2) = \{\langle x, \frac{1}{2} \rangle, \langle y, 1 \rangle\}$

$LV_{entry}(2) = LV_{entry}(2) = \{\langle y, 1 \rangle\}$
Program “Transformation”

1: \[ y \leftarrow 2 \times x \] \(^1\)
2: \[ \text{if} \ [\text{odd}(y)] \] \(^2\) \text{ then} 
3: \[ x \leftarrow x + 1 \] \(^3\)
4: \textbf{else} 
5: \[ y \leftarrow y + 1 \] \(^4\)
6: \textbf{fi} 
7: \[ y \leftarrow y + 1 \] \(^5\)

1: \[ y \leftarrow 2 \times x \] \(^1\)
2: \[ \text{choose} \] \(^2\) 
3: \[ p_\top : x \leftarrow x + 1 \] \(^3\)
4: \textbf{or} 
5: \[ p_\bot : y \leftarrow y + 1 \] \(^4\)
6: \[ y \leftarrow y + 1 \] \(^5\)

Determine branching probabilities in a first-phase analysis and utilise this information to perform the actual analysis:

\[ p_\top = A^\dagger \cdot P(b = \text{true}) \cdot A \quad \text{and} \quad p_\bot = A^\dagger \cdot P(b = \text{false}) \cdot A \]

Syntax of pWhile with Pointers

\[ S :: = [\text{skip}]^\ell \\
| [\text{stop}]^\ell \\
| [p \leftarrow e]^\ell \\
| S_1 ; S_2 \\
| [\text{choose}]^\ell p_1 : S_1 \textbf{ or } p_2 : S_2 \\
| \textbf{if} [b]^\ell \textbf{ then } S_1 \textbf{ else } S_2 \\
| \textbf{while} [b]^\ell \textbf{ do } S \]

\[ p :: = \star_{x}^\ell \text{ with } x \in \text{Var} \\
| a :: = n | p | a_1 \odot a_2 \\
| l :: = \text{NIL} | p | \& p \\
| b :: = \text{true} | \text{false} | p | \neg b | b_1 \& b_2 | a_1 \approx a_2 \]
Example

\[
\begin{align*}
\text{if } & \left(\left(z_0 \mod 2 = 0\right)\right)^1 \text{ then} \quad [x ← & z_1]^2; [y ← z_2]^3 \\
\text{else} \quad & [x ← z_2]^4; [y ← z_1]^5 \\
\text{fi} \\
[\text{stop}]^6
\end{align*}
\]

[choose] \[\frac{1}{2} \colon ([x ← z_1]^2; [y ← z_2]^3) \]

or \[\frac{1}{2} \colon ([x ← z_2]^4; [y ← z_1]^5)\]

[stop]^6

Select a certain value \(c \in \text{Value} : \)

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

\[
P(2) = \begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix}
\]

Select a certain classical state \(\sigma \in \text{State}: \)

\[
P(\sigma) = \bigotimes_{i=1}^{v} P(\sigma(x_i))
\]

\[
P(\sigma(x_1, x_2, x_3, x_4)) = \begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]
Selection via Projections

Filtering out *relevant* configurations, i.e. only those which fulfill a certain condition. Use diagonal matrix $P$:

$$(P)_{ii} = \begin{cases} 
1 & \text{if condition holds for } c_i \in \text{Value} \\
0 & \text{otherwise.} 
\end{cases}$$

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

**Example**

```plaintext
if [(z_0 \mod 2 = 0)]^1 then
  [x ← &z_1]^2; [y ← &z_2]^3
else
  [x ← &z_2]^4; [y ← &z_1]^5
fi
[stop]^6
```

$$P(z_0 \mod 2 = 0) = I \times I \times \begin{pmatrix}
  0 & 0 & 0 & 0 & \ldots \\
  0 & 1 & 0 & 0 & \ldots \\
  0 & 0 & 0 & 0 & \ldots \\
  0 & 0 & 0 & 1 & \ldots \\
  \vdots & \vdots & \vdots & \vdots & \ddots 
\end{pmatrix} \times I \times I$$
Update Operators

For all initial values change to constant $c \in \text{Value}$:

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

$$U(3) = \begin{pmatrix}
0 & 0 & 1 & 0 & 0 & \ldots \\
0 & 0 & 1 & 0 & 0 & \ldots \\
0 & 0 & 1 & 0 & 0 & \ldots \\
0 & 0 & 1 & 0 & 0 & \ldots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{pmatrix}$$

Set value of variable $x_k \in \text{Var}$ to constant $c \in \text{Value}$:

$$U(x_k \leftarrow c) = \left(\begin{array}{c}
\mathbb{I}
\end{array}\right) \otimes U(c) \otimes \left(\begin{array}{c}
\mathbb{I}
\end{array}\right)$$

Set variable $x_k \in \text{Var}$ to value given by expression $e = a \mid b \mid l$:

Update for Pointers

For an assignment with a pointer on the l.h.s. we need to determine recursively the actual variable $p$ is pointing to:

$$U(*_r x_k \leftarrow e) = \sum_{x_i} P(x_k = \& x_i) U(*_{r-1} x_i \leftarrow e)$$

Note that we always get eventually to the base case, i.e. where $p$ refers to a concrete variable $x_k$ and thus the update operator $U(x_k \leftarrow e)$ from before.

For a pointer of second order with $x_2 \rightarrow x_1 \rightarrow x_0$ we get:

$$U(* * x_2 \leftarrow 4) = \sum_{x_i} P(x_2 = \& x_i) U(* x_i \leftarrow 4)$$

$$U(* x_1 \leftarrow 4) = \sum_{x_i} P(x_1 = \& x_i) U(x_i \leftarrow 4)$$

$$U(x_0 \leftarrow 4)$$
Example

\[
\text{if } ([z_0 \mod 2 = 0]) \text{ then } [x \leftarrow &z_1]^2; [y \leftarrow &z_2]^3 \text{ else } [x \leftarrow &z_2]^4; [y \leftarrow &z_1]^5 \text{ fi [stop]}^6
\]

\[
\begin{align*}
\frac{1}{2} : ([x \leftarrow &z_1]^2; [y \leftarrow &z_2]^3) & \quad \frac{1}{2} \cdot (I \otimes E(1, 2)) + \\
\frac{1}{2} : ([x \leftarrow &z_2]^4; [y \leftarrow &z_1]^5) & \quad \frac{1}{2} \cdot (I \otimes E(1, 4)) + \\
\text{or } & \quad U(x \leftarrow &z_1) \otimes E(2, 3) + \\
\text{or } & \quad U(y \leftarrow &z_2) \otimes E(3, 6) + \\
\text{or } & \quad U(x \leftarrow &z_2) \otimes E(4, 5) + \\
\text{or } & \quad I \otimes E(6, 6)
\end{align*}
\]

Abstract Branching Probabilities

The abstract tests \(P^\#\) describe the branching probabilities depending on abstract values.

For example, consider \(P(n)\) testing if a variable with values 1, \ldots, \(n\) is a prime number.
Transforming if into choose

Based on the abstract branching probabilities we can replace tests, e.g. in if's, by probabilistic choices. In a a first phase, we need to determine the probabilities of abstract values.

If we have the probabilities of $z_0$ being even or odd we can compute the probabilities of the then and else branch using $P#$. For $z_0$ being even and odd with the same probability:

```sql
if [(z_0 \mod 2 = 0)]^1 then
    [x ← &z_1]^2; [y ← &z_2]^3
else
    [x ← &z_2]^4; [y ← &z_1]^5
fi
[stop]^6
```

Where do $x$ and $y$ point to with what probabilities?

 Probabilistic Pointer Analysis

The typical result of a probabilistic pointer analysis is a so-called points-to matrix: records for every program point the probability that a pointer refers to particular (other) variable.

Consider again our standard example.

```sql
if [(z_0 \mod 2 = 0)]^1 then
    [x ← &z_1]^2; [y ← &z_2]^3
else
    [x ← &z_2]^4; [y ← &z_1]^5
fi
[stop]^6
```

Where do $x$ and $y$ point to with what probabilities?
Points-To Matrix vs Points-To Tensor

\[
\text{if } \left[ (z_0 \mod 2 = 0) \right] \text{ then }
\]
\[\begin{align*}
[x & \leftarrow & & x_1] ;
[y & \leftarrow & & x_2]
\end{align*}\]
\[
\text{else }
\]
\[\begin{align*}
[x & \leftarrow & & x_2] ;
[y & \leftarrow & & x_1]
\end{align*}\]
\[
\text{fi }
\]
\[\begin{align*}
[\text{stop}]
\end{align*}\]

Points-To Matrix

<table>
<thead>
<tr>
<th></th>
<th>$x$</th>
<th>$y$</th>
<th>$z_0$</th>
<th>$z_1$</th>
<th>$z_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
</tr>
<tr>
<td>$y$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
</tr>
</tbody>
</table>

Points-To Tensor

\[
\frac{1}{2} \cdot (0, 0, 0, 1, 0) \otimes (0, 0, 0, 0, 1) + \frac{1}{2} \cdot (0, 0, 0, 0, 1) \otimes (0, 0, 0, 1, 0)
\]