Discrete Choice in the Presence of Numerical Uncertainties

How often does your program make a wrong decision?

Debasmita Lohar, Eva Darulova, Sylvie Putot, Eric Goubault

EMSOFT 2018
def controller(x: Real, y: Real, z: Real): Real = {
    val res = -x*y - 2*y*z - x - z
    return res
}

- Reals are implemented in Floating point/ Fixed point data type
Reals are implemented in Floating point/ Fixed point data type

- Introduces round-off error

```scala
def controller(x: Float32, y: Float32, z: Float32): Float32 = {
  val res = -x * y - 2 * y * z - x - z
  return res
}
```
def controller(x: Float32, y: Float32, z: Float32): Float32 = {
    val res = -x*y - 2*y*z - x - z
    return res
}

Computes **sound absolute error bound** in the worst case
def controller(x: Float32, y: Float32, z: Float32): Float32 = {
    val res = -x*y - 2*y*z - x - z
    if (res <= 0.0)
        raiseAlarm()
    else
        doNothing()
}
A program can make a **wrong decision** due to **numerical uncertainties**!
Our Goal

```scala
def controller(x: Float32, y: Float32, z: Float32): Float32 = {
  val res = -x*y - 2*y*z - x - z
  if (res <= 0.0)
    raiseAlarm()
  else
    doNothing()
}
```

Compute **how often** does your program make a **wrong decision**?
Worst Case Analysis is not enough!

A program always takes the wrong path in the worst case.

Consider the probability distributions of inputs.

```scala
def controller(x: Float32, y: Float32, z: Float32): Float32 = {
  val res = -x * y - 2 * y * z - x - z
  if (res <= 0.0)
    raiseAlarm()
  else
    doNothing()
}
```
def controller(x: Float32, y: Float32, z: Float32): Float32 = {
    require (-15.0 <= x, y, z <= 15.0)

    val res = -x*y - 2*y*z - x - z
    if (res <= 0.0)
        raiseAlarm()
    else
        doNothing()
}
def controller(x: Float32, y: Float32, z: Float32): Float32 = {
    require (-15.0 <= x, y, z <= 15.0)
    val res = -x*y - 2*y*z - x - z
    if (res <= 0.0)
        raiseAlarm()
    else
        doNothing()
}
Input distributions are important!

```python
def controller(x: Float32, y: Float32, z: Float32): Float32 =
    require (-15.0 <= x, y, z <= 15.0)

    val res = -x*y - 2*y*z - x - z

    if (res <= 0.0)
        raiseAlarm()
    else
        doNothing()  
```

How often? Compute Wrong Path Probability
Contributions

Sound analysis of numerical uncertainties on decisions

Evaluation on embedded examples

Prototype implementation in Daisy

https://github.com/malyzajko/daisy/tree/probabilistic
Overview: Sound Analysis

Finite Precision
Program with
Probabilistic Inputs

Wrong Path
Probability (WPP)
Overview: Sound Analysis

Round-off Error Analysis

Finite Precision Program with Probabilistic Inputs

Wrong Path Probability (WPP)
Overview: Sound Analysis

Round-off Error Analysis

\[ e = 0.01 \]

Finite Precision Program with Probabilistic Inputs

Wrong Path Probability (WPP)
Overview: Sound Analysis

Round-off Error Analysis

Finite Precision Program with Probabilistic Inputs

Decision Threshold (T)

Wrong Path Probability (WPP)
Overview: Sound Analysis

Finite Precision Program with Probabilistic Inputs

Round-off Error Analysis

Decision Threshold (T)

if (res <= 0.0) raiseAlarm()
else doNothing()
Overview: Sound Analysis

Round-off Error Analysis

Finite Precision Program with Probabilistic Inputs

Decision Threshold (T)

T = 0.0

Critical Interval [T-e, T+e]

if (res <= 0.0) raiseAlarm()
else doNothing()

Wrong Path Probability (WPP)
Overview: Sound Analysis

Round-off Error Analysis

Finite Precision Program with Probabilistic Inputs

Decision Threshold (T)

Critical Interval [T-e, T+e]

Wrong Path Probability (WPP)

T = 0.0

\[ [0.0 - 0.01, 0.0 + 0.01] \]

e = 0.01
Overview: Sound Analysis

Round-off Error Analysis

Finite Precision Program with Probabilistic Inputs

Decision Threshold (T)

Critical Interval [T-e, T+e]

Distribution Propagation

Wrong Path Probability (WPP)
Overview: Sound Analysis

Finite Precision
Program with
Probabilistic Inputs

Round-off Error
Analysis

Decision
Threshold (T)

-/+  

Critical Interval
[T-e, T+e]

Intersection

Wrong Path
Probability (WPP)

Distribution
Propagation
Round-off Error Analysis

Finite Precision Program

Round-off Error (e)

def controller(x:Float32, y:Float32, z:Float32): Float32 = {
  require (-15 <= x, y, z <= 15)
  val res = -x*y - 2*y*z - x - z
}
Round-off Error Analysis

Finite Precision Program

```
def controller(x:Float32, y:Float32, z:Float32): Float32 = {
    require (-15 <= x, y, z <= 15)
    val res = -x*y - 2*y*z - x - z
}
```

\[ e \text{ in } \text{res} = 1.73e-04 \]
Distribution Propagation

Finite Precision Program with Probabilistic Inputs

Round-off Error Analysis

Round-off Error (e)

Decision Threshold (T)

Critical Interval [T-e, T+e]

Intersection

Wrong Path Probability (WPP)
Distribution Propagation

Finite Precision Program with Probabilistic Inputs

Round-off Error Analysis

Round-off Error (e)

Symbolic Inference

OR

Static Analysis

Probabilistic Analysis

Decision Threshold (T)

-/+ 

Critical Interval [T-e, T+e]

Intersection

Wrong Path Probability (WPP)
Symbolic Inference

```
def main() {
    x := uniform(-15.0, 15.0)
    y := uniform(-15.0, 15.0)
    z := uniform(-15.0, 15.0)
    res := -x*y - 2*y*z - x - z

    //e: round-off error; T: decision threshold
    assert(!(T-e <= res && res <= T+e))
}
```

PSI considers uniform/normal independent inputs

"PSI: Exact Symbolic Inference for Probabilistic Programs", S. Misailovic, M. Vechev, and T. Gehr, CAV 2016
Symmetric Inference

Program + Critical Interval as assert

Probabilistic Symbolic Inference (PSI)

Exact Distribution

```python
def main() {
    x:= uniform(-15.0, 15.0)
    y:= uniform(-15.0, 15.0)
    z:= uniform(-15.0, 15.0)
    res:= -x*y - 2*y*z - x - z

    // e: round-off error; T: decision threshold
    assert(!((T-e <= res && res <= T+e))
}
```

Computes probability distribution of the assertion failing

"PSI: Exact Symbolic Inference for Probabilistic Programs", S. Misailovic, M. Vechev, and T. Gehr, CAV 2016
PSI computes the expression of exact WPP

\[-\frac{1}{(-1/Pi*Sqrt[Pi])*(Erf[-1/2^(1/2)]+1)/2*Sqrt[Pi]*(Erf[1/450^(1/2)*15]+1)}\ldots\]
Symbolic Inference

Program + Critical Interval as assert

Probabilistic Symbolic Inference (PSI)

Mathematica numerically evaluates the distribution

Exact Distribution

Wrong Path Probability (WPP)
def controller(x: Float32, y: Float32, z: Float32): Float32 = {
  require (-15 <= x, y, z <= 15)
  val res = -x*y - 2*y*z - x - z
  if (res <= 0.0) raiseAlarm()
  else doNothing()
}
def controller(x: Float32, y: Float32, z: Float32): Float32 = {
    require (-15 <= x, y, z <= 15)
    val res = -x*y - 2*y*z - x - z
    if (res <= 0.0) raiseAlarm()
    else doNothing()
}
def controller(x: Float32, y: Float32, z: Float32): Float32 = {
    require (-15 <= x, y, z <= 15)
    val res = -x*y - 2*y*z - x - z
    if (res <= 0.0) raiseAlarm()
    else doNothing()
}
## WPP using Symbolic Inference

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>#ops</th>
</tr>
</thead>
<tbody>
<tr>
<td>sine</td>
<td>18</td>
</tr>
<tr>
<td>sqrt</td>
<td>14</td>
</tr>
<tr>
<td>turbine1</td>
<td>14</td>
</tr>
<tr>
<td>traincar2</td>
<td>13</td>
</tr>
<tr>
<td>doppler</td>
<td>10</td>
</tr>
<tr>
<td>bspline1</td>
<td>8</td>
</tr>
<tr>
<td>rigidbody1</td>
<td>7</td>
</tr>
<tr>
<td>traincar1</td>
<td>7</td>
</tr>
<tr>
<td>bspline0</td>
<td>6</td>
</tr>
<tr>
<td>sineorder3</td>
<td>4</td>
</tr>
</tbody>
</table>

Wrong path probability for 32 bit floating-point round-off errors and uniform input distributions
## WPP using Symbolic Inference

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>#ops</th>
<th>Sym. Inf.</th>
</tr>
</thead>
<tbody>
<tr>
<td>sine</td>
<td>18</td>
<td>7.61e-7</td>
</tr>
<tr>
<td>sqrt</td>
<td>14</td>
<td>8.74e-6</td>
</tr>
<tr>
<td>turbine1</td>
<td>14</td>
<td>TO</td>
</tr>
<tr>
<td>traincar2</td>
<td>13</td>
<td>TO</td>
</tr>
<tr>
<td>doppler</td>
<td>10</td>
<td>TO</td>
</tr>
<tr>
<td>bspline1</td>
<td>8</td>
<td>2.54e-6</td>
</tr>
<tr>
<td>rigidbody1</td>
<td>7</td>
<td>TO</td>
</tr>
<tr>
<td>traincar1</td>
<td>7</td>
<td>TO</td>
</tr>
<tr>
<td>bspline0</td>
<td>6</td>
<td>1.05e-5</td>
</tr>
<tr>
<td>sineorder3</td>
<td>4</td>
<td>1.90e-6</td>
</tr>
</tbody>
</table>

Wrong path probability for 32 bit floating-point round-off errors and uniform input distributions
Distribution Propagation

Finite Precision Program with Probabilistic Inputs

Round-off Error Analysis

Round-off Error (e)

Decision Threshold (T)

-/+ decision

Critical Interval [T-e, T+e]

Intersection

Probabilistic Analysis

Symbolic Inference

OR

Static Analysis

Wrong Path Probability (WPP)
Probabilistic Static Analysis

- Probabilistic Inputs
- Probabilistic Affine Arithmetic
- Output DSI Structure
- Intersection
- Critical Interval
- Wrong Path Probability (WPP)
Probabilistic Static Analysis

Probabilistic Inputs

Probabilistic Inputs as uncertain probabilities

Probabilistic Affine Arithmetic propagates the probabilities

Output DSI Structure

Intersection

Wrong Path Probability (WPP)

Critical Interval
Dempster Shafer Interval (DSI) Structure

Discretizes the continuous distribution into sets of intervals and weights
Dempster Shafer Interval (DSI) Structure

Discretizes the continuous distribution into sets of intervals and weights

\[
\{\langle -1.0, -0.5 \rangle, 0.1\}, \langle -0.5, 0.25 \rangle, 0.2\} \ldots
\]

- Number of discretization is fixed
- Divides the input range

Focal Element
Dempster Shafer Interval (DSI) Arithmetic

\[ x \rightarrow d_x = \{ < [x a_i, x b_i], x w_i > , i \in [1,n] \} \]

\[ y \rightarrow d_y = \{ < [y a_j, y b_j], y w_j > , j \in [1,m] \} \]

\[ z = x \Box y, (\Box = +, -, \times, \div) \]

\[ x, y \text{ are independent} \]

- Interval arithmetic for intervals
- Weights are multiplied
Dempster Shafer Interval (DSI) Arithmetic

\[ x \rightarrow d_x = \{ < [x a_i, x b_i], x w_i >, i \in [1, n] \} \]
\[ y \rightarrow d_y = \{ < [y a_j, y b_j], y w_j >, j \in [1, m] \} \]
\[ z = x \, \Box \, y, (\Box = +, -, \times, \div) \]

- x, y are independent
- x, y are dependent

- Interval arithmetic for intervals
- Weights are multiplied

- Interval arithmetic for intervals
- Simplex Solver to compute weights
Dempster Shafer Interval (DSI) Arithmetic

\[ x \rightarrow d_x = \{ < [xa_i, xb_i], xw_i > , i \in [1,n] \} \]

\[ y \rightarrow d_y = \{ < [ya_j, yb_j], yw_j > , j \in [1,m] \} \]

We need to track dependency

Easy

- Interval arithmetic for intervals
- Weights are multiplied

Complex

- Interval arithmetic for intervals
- Simplex Solver to compute weights
Probabilistic Affine Arithmetic

- Affine Arithmetic propagates linear relations between variables
- Dependencies are tracked using shared noise symbol

\[ \hat{x} := x_0 + \sum_{i=1}^{p} x_i \epsilon_i, \quad \epsilon_i \in [-1,1] \]

"Static Analysis of Programs with Imprecise Probabilistic Inputs", A. Adje, O. Bouissou, J. Goubault-Larrecq, E. Goubault, S. Putot, VSTTE 2013
Probabilistic Affine Arithmetic

- Affine Arithmetic propagates linear relations between variables
- Dependencies are tracked using shared noise symbol
- Uses DSI to keep the probabilities while tracking dependencies

\[ \hat{x} := x_0 + \sum_{i=1}^{p} x_i c_i, \quad c_i \in [-1,1] \]

"Static Analysis of Programs with Imprecise Probabilistic Inputs", A. Adje, O. Bouissou, J. Goubault-Larrecq, E. Goubault, S. Putot, VSTTE 2013
Probabilistic Affine Arithmetic

- Affine Arithmetic propagates linear relations between variables
- Dependencies are tracked using shared noise symbol
- Uses DSI to keep the probabilities while tracking dependencies

\[ \hat{x} := x_0 + \sum_{i=1}^{p} x_i \epsilon_i, \epsilon_i \in [-1, 1] \]

- Arithmetic operations are computed term wise

"Static Analysis of Programs with Imprecise Probabilistic Inputs", A. Adje, O. Bouissou, J. Goubault-Larrecq, E. Goubault, S. Putot, VSTTE 2013
Output DSI $= d_x : < [a_1, b_1], w_1 >, \cdots < [a_n, b_n], w_n >$

Critical Interval $= [T - e, T + e]$
Intersection

**Critical Interval**  
$[T - e, T + e]$  

- $w_6 = 0.05$  
- $w_5 = 0.35$  
- $w_4 = 0.1$  
- $w_3 = 0.2$  
- $w_2 = 0.1$  
- $w_1 = 0.2$

**Output DSI**

- $< [a, b], w >$

- $< [a_1, b_1], w_1 >$

Interval
Wrong Path Probability = 0.2 + 0.2 + 0.1 + 0.35
= 0.85
Probabilistic Static Analysis

In general incurs huge overapproximation.

Probabilistic Inputs

Arithmetic

Interval

Output DSI Structure

Intersection

Wrong Path Probability (WPP)
In general incurs huge overapproximation

```scala
def controller(x: Float32, y: Float32, z: Float32): Float32 = {
  require (-15 <= x, y, z <= 15)
  val res = -x*y - 2*y*z - x - z
  if (res <= 0.0) raiseAlarm()
  else doNothing()
}
```

WPP = 1.0
## WPP using Probabilistic Static Analysis

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>#ops</th>
<th>Sym. Inf.</th>
<th>Probabilistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>sine</td>
<td>18</td>
<td>7.61e-7</td>
<td>0.32</td>
</tr>
<tr>
<td>sqrt</td>
<td>14</td>
<td>8.74e-6</td>
<td>1.00</td>
</tr>
<tr>
<td>turbine1</td>
<td>14</td>
<td>TO</td>
<td>1.00</td>
</tr>
<tr>
<td>traincar2</td>
<td>13</td>
<td>TO</td>
<td>0.11</td>
</tr>
<tr>
<td>doppler</td>
<td>10</td>
<td>TO</td>
<td>1.00</td>
</tr>
<tr>
<td>bspline1</td>
<td>8</td>
<td>2.54e-6</td>
<td>0.96</td>
</tr>
<tr>
<td>rigidbody1</td>
<td>7</td>
<td>TO</td>
<td>1.00</td>
</tr>
<tr>
<td>traincar1</td>
<td>7</td>
<td>TO</td>
<td>0.10</td>
</tr>
<tr>
<td>bspline0</td>
<td>6</td>
<td>1.05e-5</td>
<td>1.00</td>
</tr>
<tr>
<td>sineorder3</td>
<td>4</td>
<td>1.90e-6</td>
<td>0.36</td>
</tr>
</tbody>
</table>

Wrong path probability for 32 bit floating-point round-off errors and uniform input distributions.
Interval Subdivision

Input Intervals

n subdivided Intervals

Total WPP =
\[
\rho_{total} = \sum_{i=1}^{n} Prob(s_{ijk})\rho_{ijk}
\]

subdomains
\[
s_{ijk} = x_i \times y_j \times z_k
\]

\(\rho_{ijk} \leftarrow \text{Probabilistic Analysis}\)
Interval Subdivision

Probabilistic Analysis is costly

Total WPP =

\[ \rho_{\text{total}} = \sum_{i=1}^{n} \text{Prob}(s_{ijk}) \rho_{ijk} \]

\[ \rho_{ijk} = \text{Probabilistic Analysis} \]
Pruning subdomains by reachability checks

Is the critical interval reachable?

Subdomains $s_{ijk} = x_i \times y_j \times z_k$

Not reachable

Reachable / Don't know

Input Intervals

$n$ subdivided Intervals

Total WPP = $\rho_{total} = \sum_{i=1}^{n} Prob(s_{ijk})\rho_{ijk}$

Wrong Path Probability $\rho_{ijk} = 0.0$

$\rho_{ijk} \leftarrow$ Probabilistic Analysis
Interval Subdivision + Reachability

No. of subdomain: 8000
No. of DSI discretization: 4

Total prob. =
\[ \rho_{\text{total}} = \sum_{i=1}^{n} Prob(s_{ijk})\rho_{ijk} \]

Wrong Path Probability
\[ \rho_{ijk} = 0.0 \]

Wrong Path Probability
\[ \rho_{ijk} = 1.0 \]
Interval Subdivision + Reachability

Provides good estimates

\[ \rho_{\text{total}} = \sum_{i=1}^{n} \rho_{ijk} \]

\[ \rho_{ijk} = 0.0 \]

\[ \rho_{ij} = \text{Wrong Path Probability} \]

Is the critical interval reachable?

reachable / don't know

def controller(x: Float32, y: Float32, z: Float32): Float32 = {
    require (-15 <= x, y, z <= 15)
    val res = -x*y - 2*y*z - x - z
    if (res <= 0.0) raiseAlarm()
    else doNothing()
}

WPP = 4.39e-3
Can we skip the Probabilistic Analysis?

Input Intervals → n subdivided Intervals

Total WPP = \[ \rho_{total} = \sum_{i=1}^{n} \text{Prob}(s_{ijk})\rho_{ijk} \]

Wrong Path Probability \[ \rho_{ijk} = 0.0 \]

Is the critical interval reachable?

subdomains \( s_{ijk} = x_i \times y_j \times z_k \)

- not reachable
- reachable / don't know

Probabilistic Analysis
Non-Probabilistic Analysis

Input Intervals -> n subdivided Intervals

Total WPP = \[ \rho_{total} = \sum_{i=1}^{n} \text{Prob}(s_{ijk})\rho_{ijk} \]

Wrong Path Probability
- \( \rho_{ijk} = 0.0 \) (not reachable)
- \( \rho_{ijk} = 1.0 \) (reachable / don't know)

Is the critical interval reachable?

Subdomains \( s_{ijk} = x_i \times y_j \times z_k \)
Non-Probabilistic Analysis

Works well for univariate functions

```
def controller(x:Float32, y:Float32, z:Float32): Float32 = {
    require (-15 <= x, y, z <= 15)
    val res = -x*y - 2*y*z - x - z
    if (res <= 0.0) raiseAlarm()
    else doNothing()
}
```

WPP = 2.53e-2
Non-Probabilistic Analysis

No. of subdomain: 32000

```scala
def controller(x: Float32, y: Float32, z: Float32): Float32 = {
  require (-15 <= x, y, z <= 15)
  val res = -x*y - 2*y*z - x - z
  if (res <= 0.0) raiseAlarm()
  else doNothing()
}
```

WPP = 2.53e-2 > 4.39e-3
## WPP using Probabilistic Analysis with Subdiv

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>#ops</th>
<th>Sym. Inf.</th>
<th>Probabilistic</th>
<th>Probabilistic with Subdiv</th>
</tr>
</thead>
<tbody>
<tr>
<td>sine</td>
<td>18</td>
<td>7.61e-7</td>
<td>0.32</td>
<td>6.45e-5</td>
</tr>
<tr>
<td>sqrt</td>
<td>14</td>
<td>8.74e-6</td>
<td>1.00</td>
<td>9.38e-5</td>
</tr>
<tr>
<td>turbine1</td>
<td>14</td>
<td>TO</td>
<td>1.00</td>
<td>4.82e-2</td>
</tr>
<tr>
<td>traincar2</td>
<td>13</td>
<td>TO</td>
<td>0.11</td>
<td>9.17e-2</td>
</tr>
<tr>
<td>doppler</td>
<td>10</td>
<td>TO</td>
<td>1.00</td>
<td>2.17e-2</td>
</tr>
<tr>
<td>bspline1</td>
<td>8</td>
<td>2.54e-6</td>
<td>0.96</td>
<td>1.95e-5</td>
</tr>
<tr>
<td>rigidbody1</td>
<td>7</td>
<td>TO</td>
<td>1.00</td>
<td>7.06e-2</td>
</tr>
<tr>
<td>traincar1</td>
<td>7</td>
<td>TO</td>
<td>0.10</td>
<td>1.86e-2</td>
</tr>
<tr>
<td>bspline0</td>
<td>6</td>
<td>1.05e-5</td>
<td>1.00</td>
<td>6.06e-5</td>
</tr>
<tr>
<td>sineorder3</td>
<td>4</td>
<td>1.90e-6</td>
<td>0.36</td>
<td>1.23e-4</td>
</tr>
</tbody>
</table>

Wrong path probability for 32 bit floating-point round-off errors and uniform input distributions.
WPP using Probabilistic Analysis with Subdiv

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>#ops</th>
<th>Sym. Inf.</th>
<th>Probabilistic</th>
<th>Probabilistic with Subdiv</th>
</tr>
</thead>
<tbody>
<tr>
<td>sine</td>
<td>18</td>
<td>7.61e-7</td>
<td>0.32</td>
<td>6.45e-5</td>
</tr>
<tr>
<td>sqrt</td>
<td>14</td>
<td>8.74e-6</td>
<td>1.00</td>
<td>9.38e-5</td>
</tr>
<tr>
<td>turbine1</td>
<td>14</td>
<td>TO</td>
<td>1.00</td>
<td>4.82e-2</td>
</tr>
<tr>
<td>traincar2</td>
<td>13</td>
<td>TO</td>
<td>0.11</td>
<td>9.17e-2</td>
</tr>
<tr>
<td>doppler</td>
<td>10</td>
<td>TO</td>
<td>1.00</td>
<td>2.17e-2</td>
</tr>
<tr>
<td><strong>bspline1</strong></td>
<td>8</td>
<td>2.54e-6</td>
<td><strong>0.96</strong></td>
<td><strong>1.95e-5</strong></td>
</tr>
<tr>
<td>rigidbody1</td>
<td>7</td>
<td>TO</td>
<td>1.00</td>
<td>7.06e-2</td>
</tr>
<tr>
<td>traincar1</td>
<td>7</td>
<td>TO</td>
<td>0.10</td>
<td>1.86e-2</td>
</tr>
<tr>
<td>bspline0</td>
<td>6</td>
<td>1.05e-5</td>
<td>1.00</td>
<td>6.06e-5</td>
</tr>
<tr>
<td>sineorder3</td>
<td>4</td>
<td>1.90e-6</td>
<td>0.36</td>
<td>1.23e-4</td>
</tr>
</tbody>
</table>

wrong path probability for 32 bit floating-point round-off errors and uniform input distributions
Case Studies from **embedded systems** and **machine learning**

- filter
- traincar
- Simplified DNN
- Linear SVC

It scales for real world programs!
What else is there in the paper?

Extensive experiments on

- Fixed precision vs Floating-point precision
- Uniform vs Gaussian distributions
- Dependent vs Independent inputs

"Discrete Choice in the Presence of Numerical Uncertainties"

D. Lohar, E. Darulova, S. Putot and E. Gobault
Conclusion

- Sound Analysis of Numerical Uncertainties on Decisions
- **Probabilistic Symbolic Inference** suffers from **scalability issues**
- **Probabilistic Static Analysis** has **accuracy issues**
- **Static analysis with reachability** is **accurate** and **scalable**