Discrete Choice in the Presence of Numerical Uncertainties

How often does your program make a wrong decision?

MAX PLANCK INSTITUTE FOR SOFTWARE SYSTEMS

ÉCOLE POLYTECHNIQUE UNIVERSITÉ PARIS-SACLAY

Debasmita Lohar, Eva Darulova, Sylvie Putot, Eric Goubault

EMSOFT 2018

Programming with Numerical Uncertainties

```
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

Programming with Numerical Uncertainties

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

- Reals are implemented in Floating point/ Fixed point data type
- Introduces round-off error

State of the Art: Round-off Error Analysis

```
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**

Programs with Discrete Decisions

```
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()
}
```

Programs with Discrete Decisions

```
def controller(x:Float32, y:Float32, z:Float32):Float32={
   val res = -x*y - 2*y*z - x - z
   if (res <= 0.0)
     raiseAlarm()     → real valued program
   else
     doNothing()      → finite precision program
}
```

A program can make a **wrong decision** due to **numerical uncertainties**!

Our Goal

```
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**?

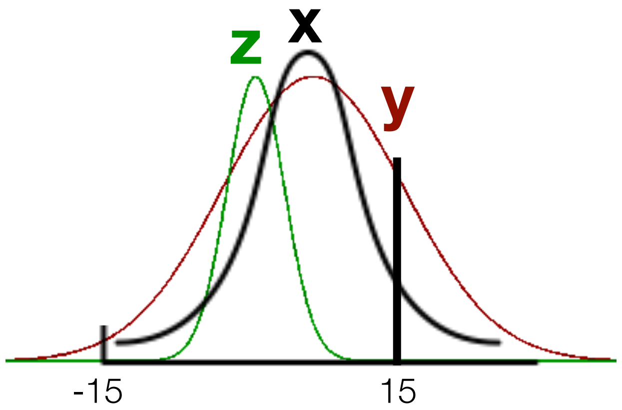
Worst Case Analysis is not enough!

```
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 **always** takes the wrong path in the **worst case**
- Consider the **probability distributions** of inputs

Input distributions are important!

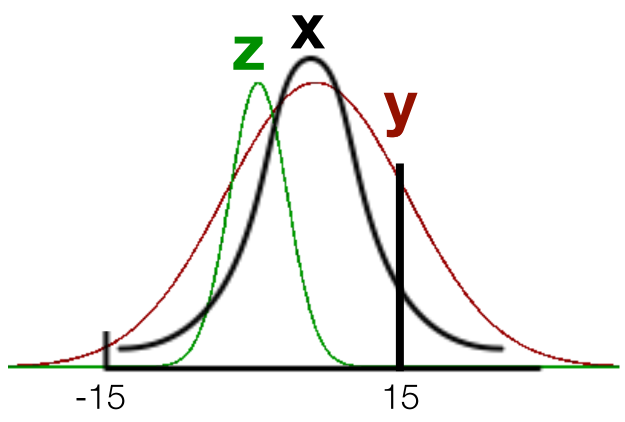
```
def controller(x:Float32, y:Float32, z:Float32):Float32={
  require (-15.0 <= x, y, z <= 15.0)
```



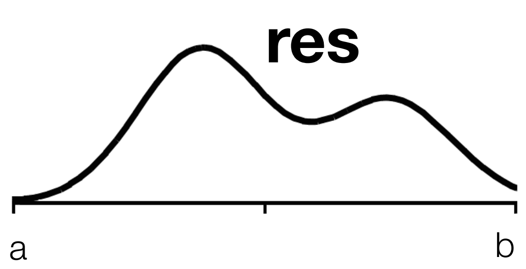

```
   val res = -x*y - 2*y*z - x - z
   if (res <= 0.0)
     raiseAlarm()
   else
     doNothing()
}
```

Input distributions are important!

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



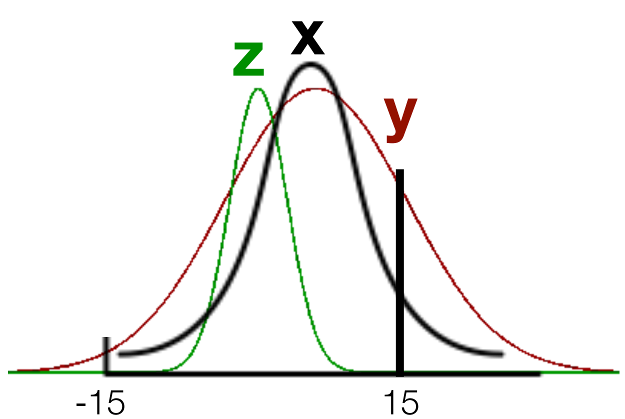

```
   val res = -x*y - 2*y*z - x - z
```



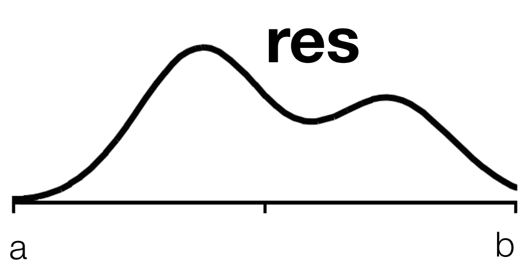

```
   if (res <= 0.0)
     raiseAlarm()
   else
     doNothing()
}
```

Input distributions are important!

```
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

Finite Precision Program with Probabilistic Inputs

Round-off Error Analysis

Wrong Path Probability (WPP)

Overview: Sound Analysis

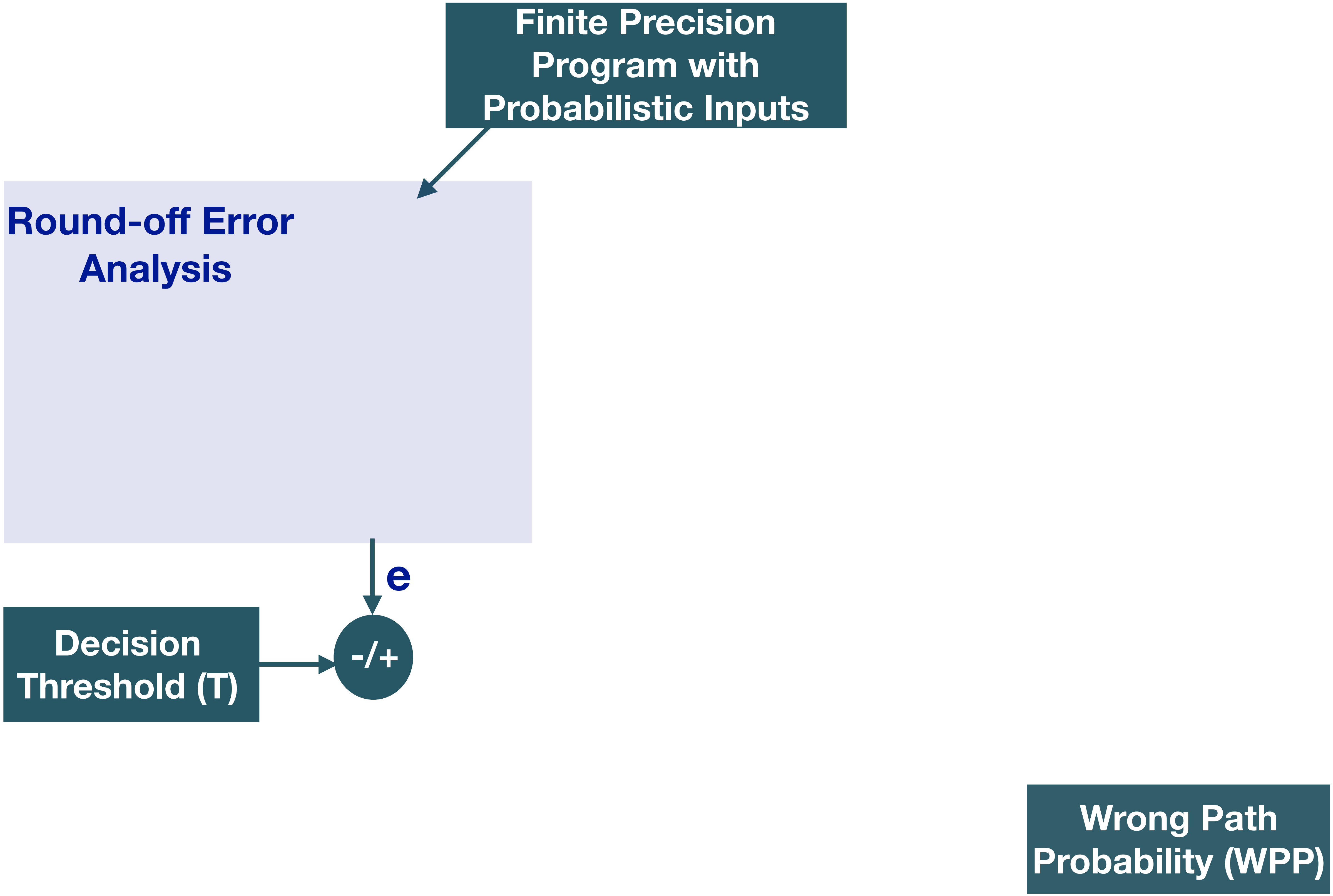
Finite Precision Program with Probabilistic Inputs

Round-off Error Analysis

$e = \mathtt{0.01}$

Wrong Path Probability (WPP)

Overview: Sound Analysis



Overview: Sound Analysis

Finite Precision Program with Probabilistic Inputs

Round-off Error Analysis

e

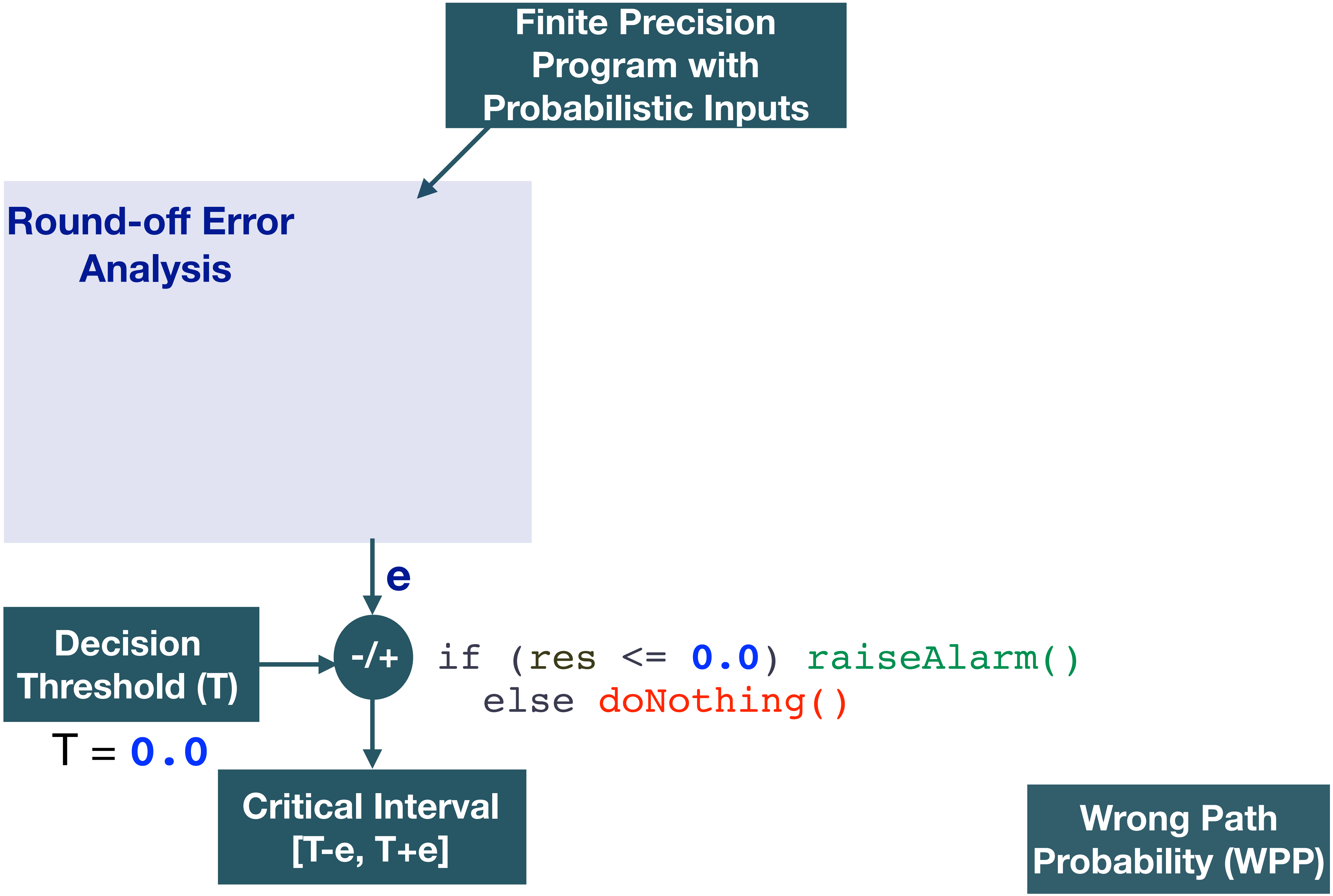
Decision Threshold (T)

-/+

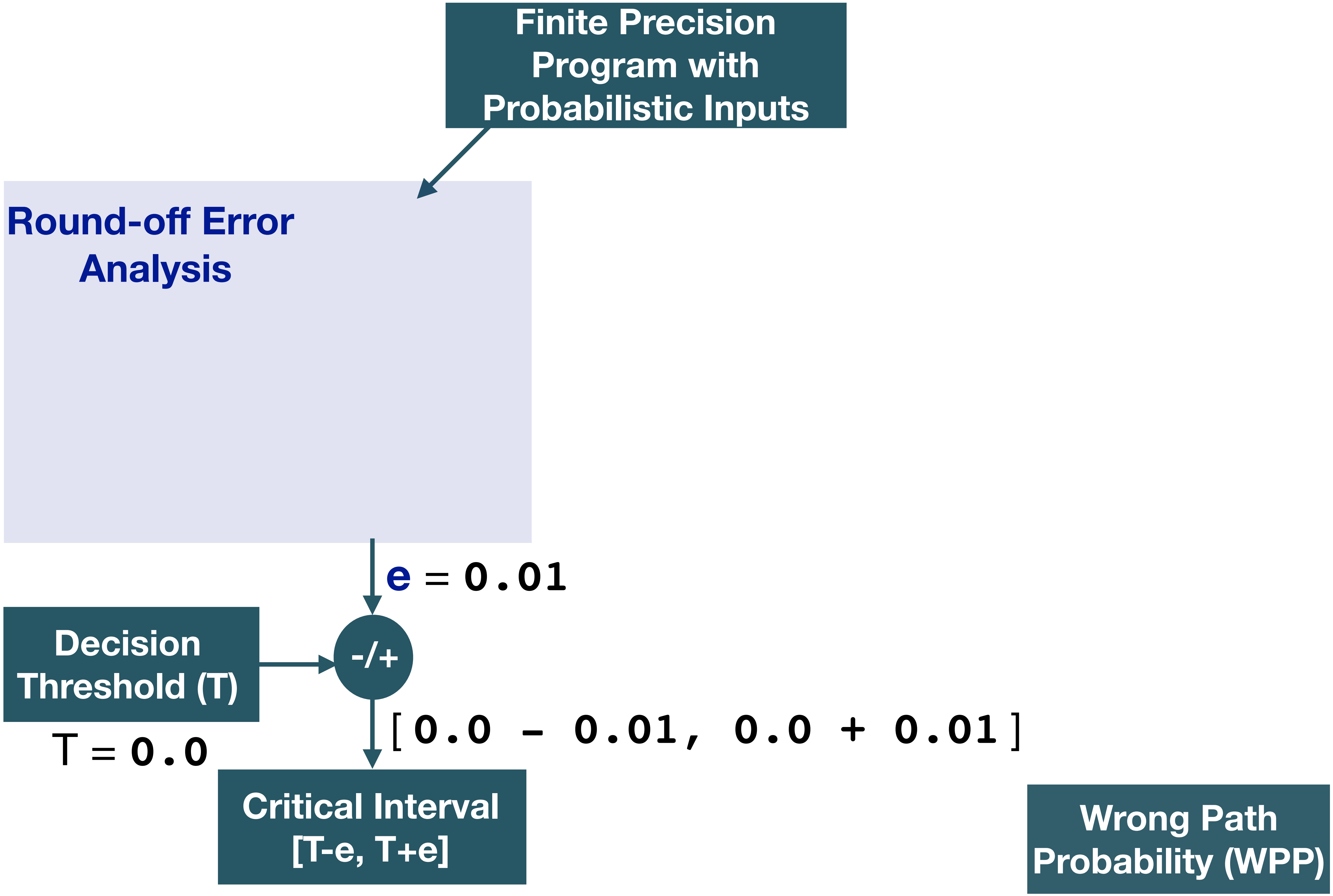
```
if (res <= 0.0) raiseAlarm()
  else doNothing()
```

Wrong Path Probability (WPP)

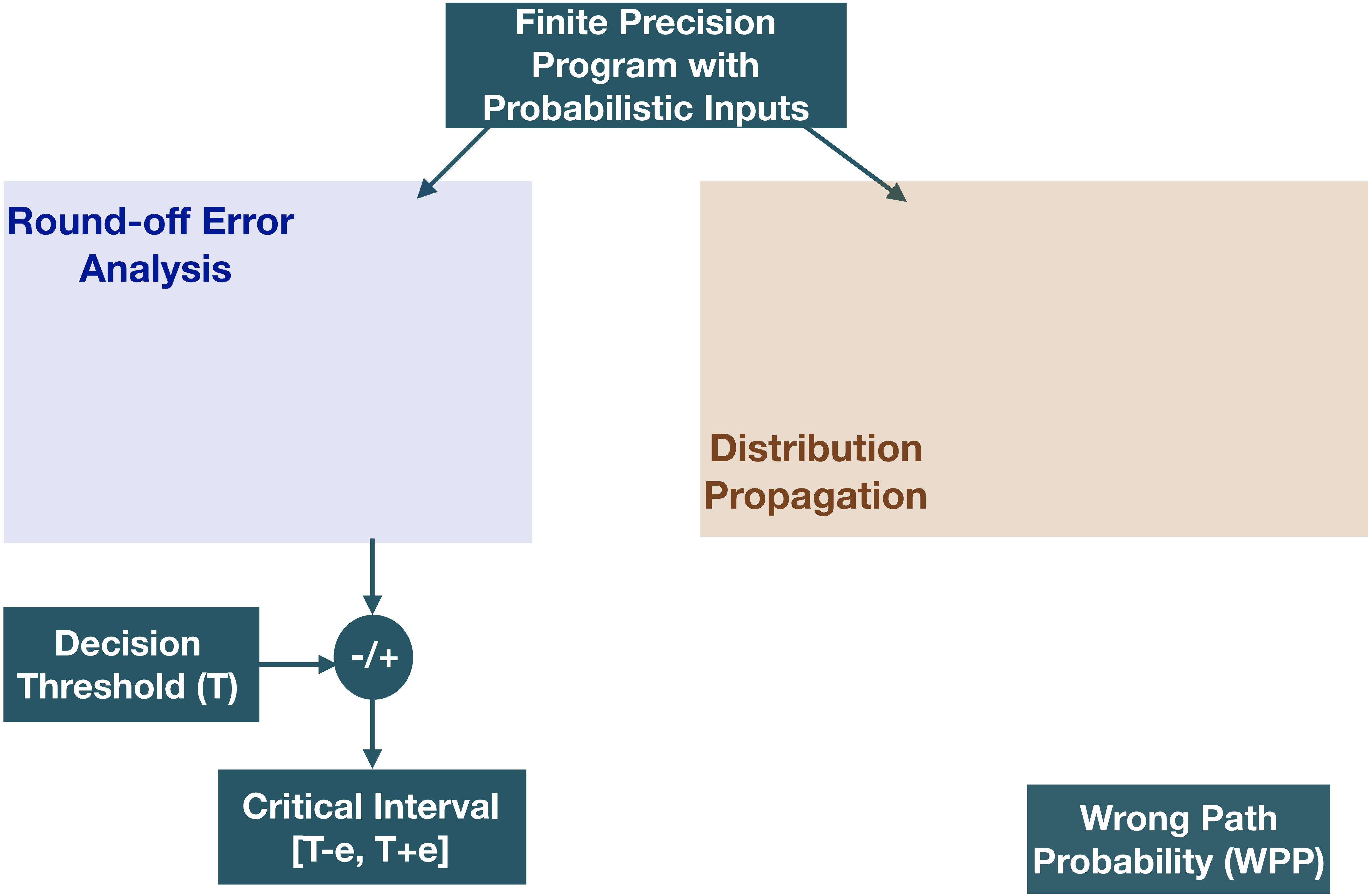
Overview: Sound Analysis



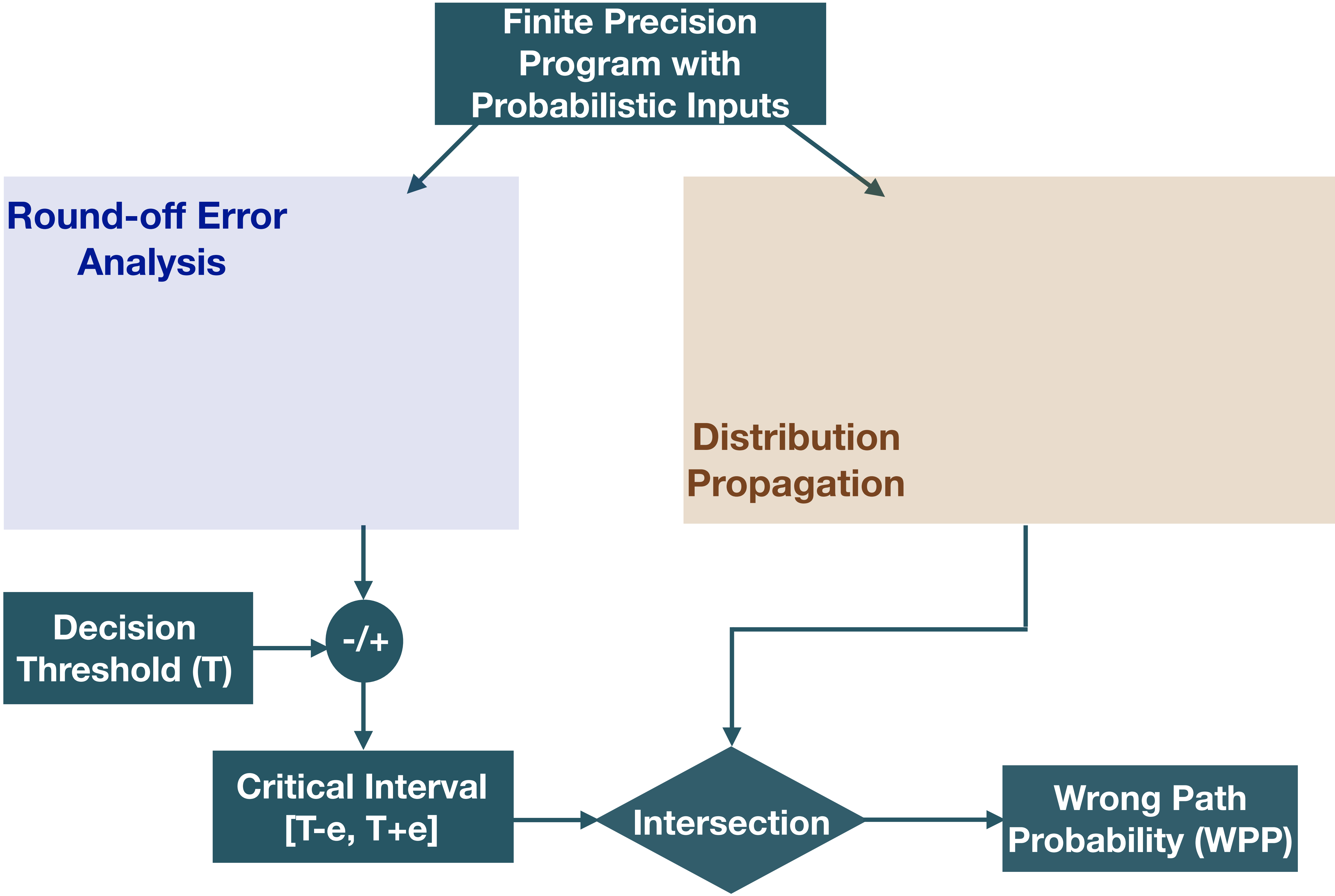
Overview: Sound Analysis



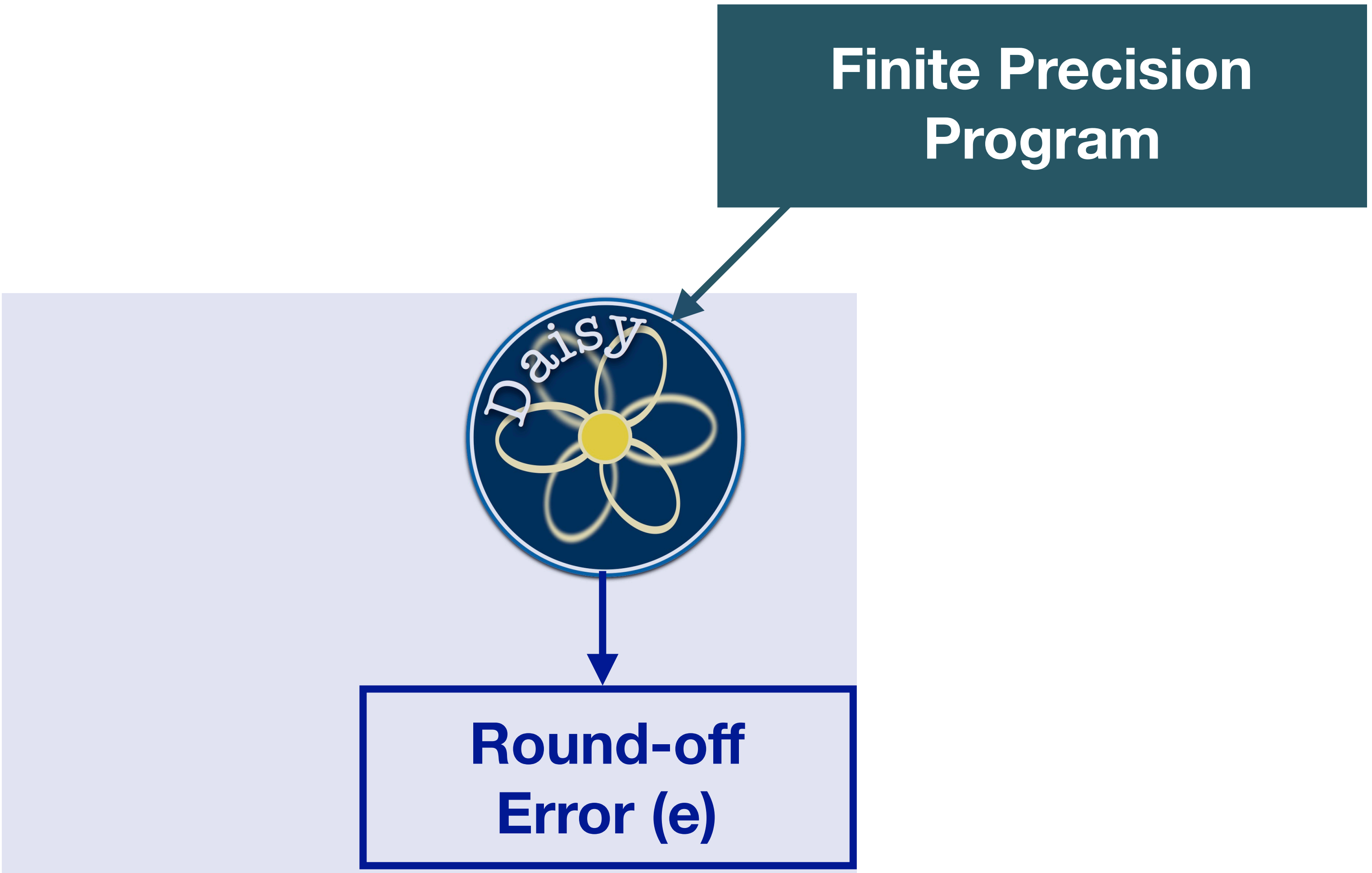
Overview: Sound Analysis



Overview: Sound Analysis



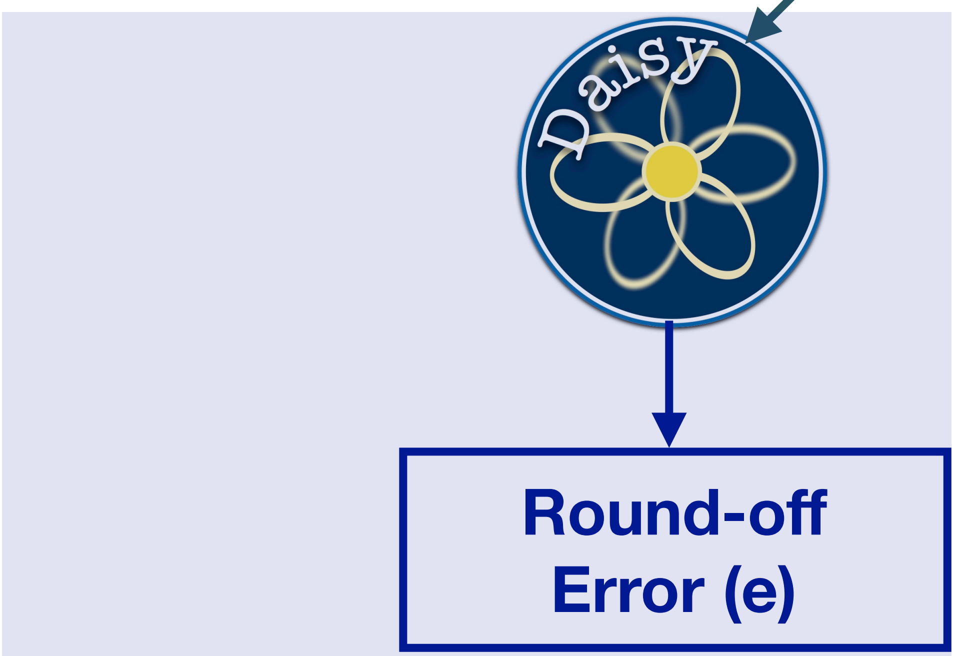
Round-off Error Analysis



"Daisy - Framework for Analysis and Optimization of Numerical Programs", E. Darulova, A. Izycheva, F. Nasir, F. Ritter, H. Becker, and R. Bastian, TACAS 2018

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
}
```

"Daisy - Framework for Analysis and Optimization of Numerical Programs", E. Darulova, A. Izycheva, F. Nasir, F. Ritter, H. Becker, and R. Bastian, TACAS 2018

Round-off Error Analysis

Finite Precision Program

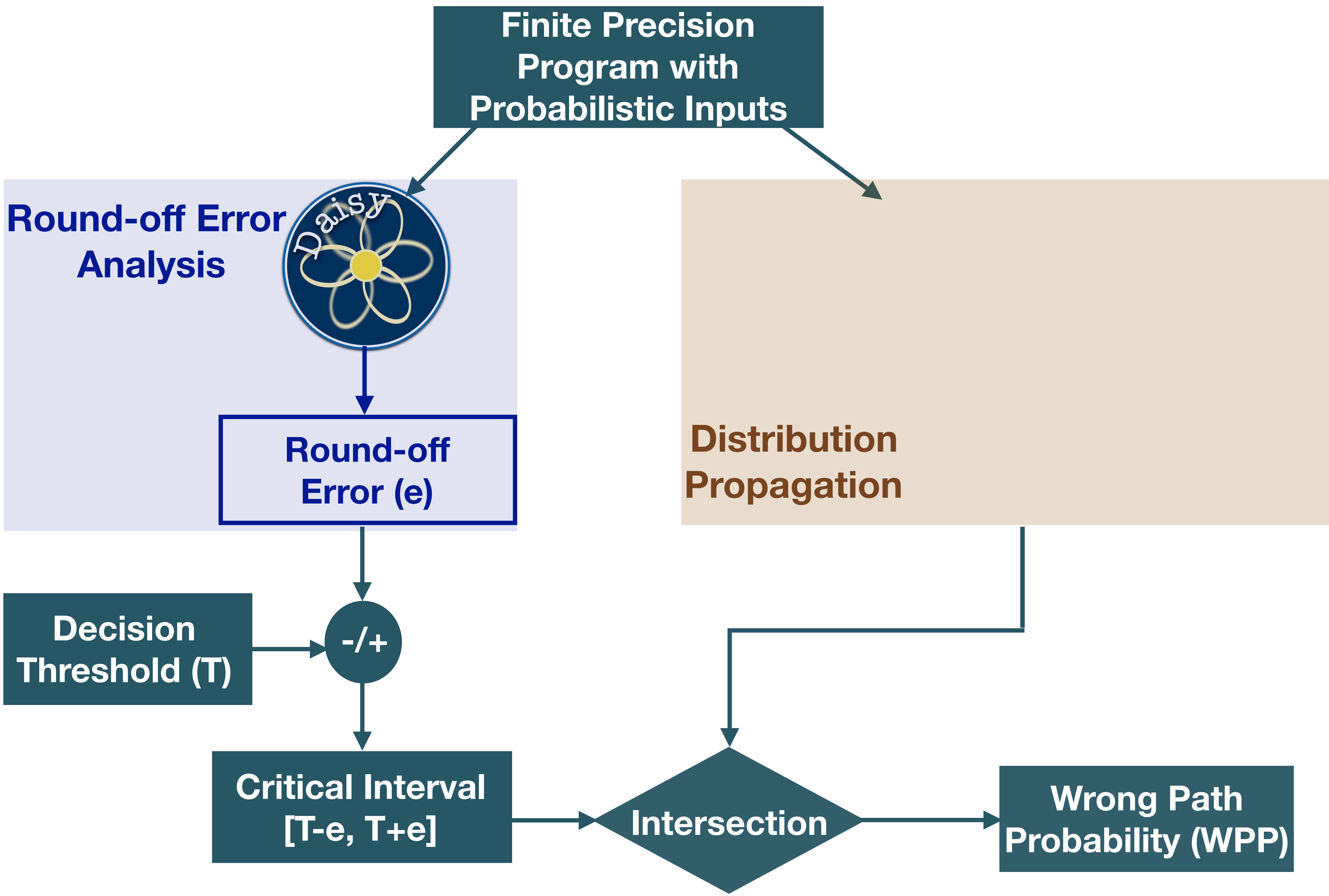
Round-off Error (e)

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

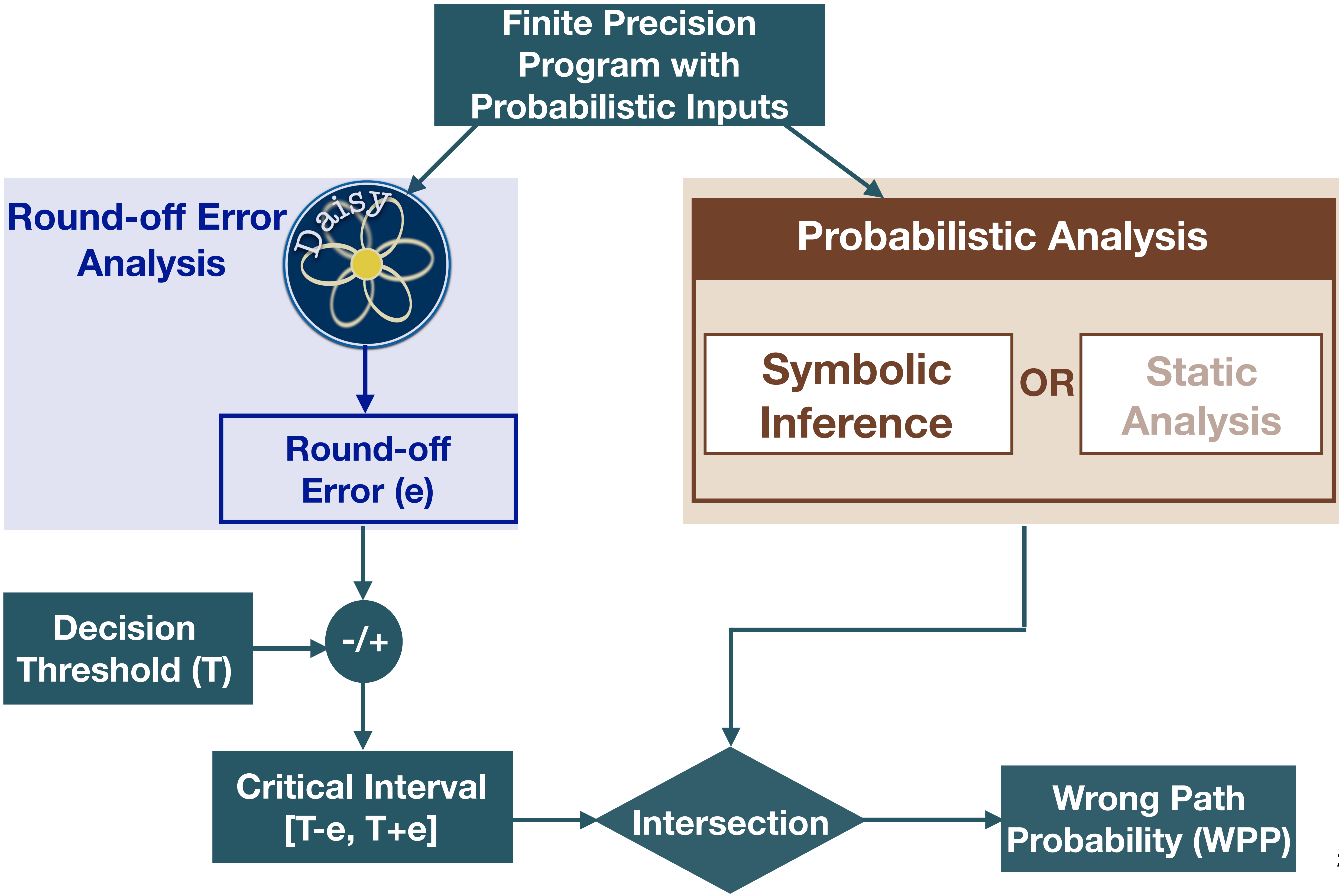
e in **res** = **1.73e-04**

"Daisy - Framework for Analysis and Optimization of Numerical Programs", E. Darulova, A. Izycheva, F. Nasir, F. Ritter, H. Becker, and R. Bastian, TACAS 2018

Distribution Propagation



Distribution Propagation



Symbolic Inference

Program + Critical Interval as assert

↓

Probabilistic Symbolic Inference (PSI)

```
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

Symbolic Inference

Program + Critical Interval as assert

↓

Probabilistic Symbolic Inference (PSI)

↓

Exact Distribution

```
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

Symbolic Inference

Program + Critical Interval as assert

↓

Probabilistic Symbolic Inference (PSI)

↓

Exact Distribution

PSI computes the expression of exact WPP

```
-1/(-1/Pi*Sqrt[Pi]*(Erf[-1/2^(1/2)]+1)/
2*Sqrt[Pi]*(Erf[1/450^(1/2)*15]+1)....
```

Symbolic Inference

Program + Critical Interval as assert

↓

Probabilistic Symbolic Inference (PSI)

↓

Exact Distribution → Wolfram Mathematica → Wrong Path Probability (WPP)

Mathematica numerically evaluates the distribution

Symbolic Inference

Program + Critical Interval as assert

Probabilistic Symbolic Inference (PSI)

Exact Distribution

Wolfram Mathematica

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()
}
```

Symbolic Inference

Program + Critical Interval as assert

Probabilistic Symbolic Inference (PSI)

Exact Distribution

Wolfram Mathematica

Wrong Path Probability (WPP)

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

Mathematica Timed Out!

Symbolic Inference

Program + Critical Interval as assert

```
def controller(x:Float32, y:Float32,
 z:Float32): Float32 = {
```

Timed out for many of the benchmarks

```
}
```

Symbolic Inference (PSI)

Exact Distribution

Wolfram Mathematica

Wrong Path Probability (WPP)

WPP using Symbolic Inference

Benchmarks	#ops
sine	18
sqrt	14
turbine1	14
traincar2	13
doppler	10
bspline1	8
rigidbody1	7
traincar1	7
bspline0	6
sineorder3	4

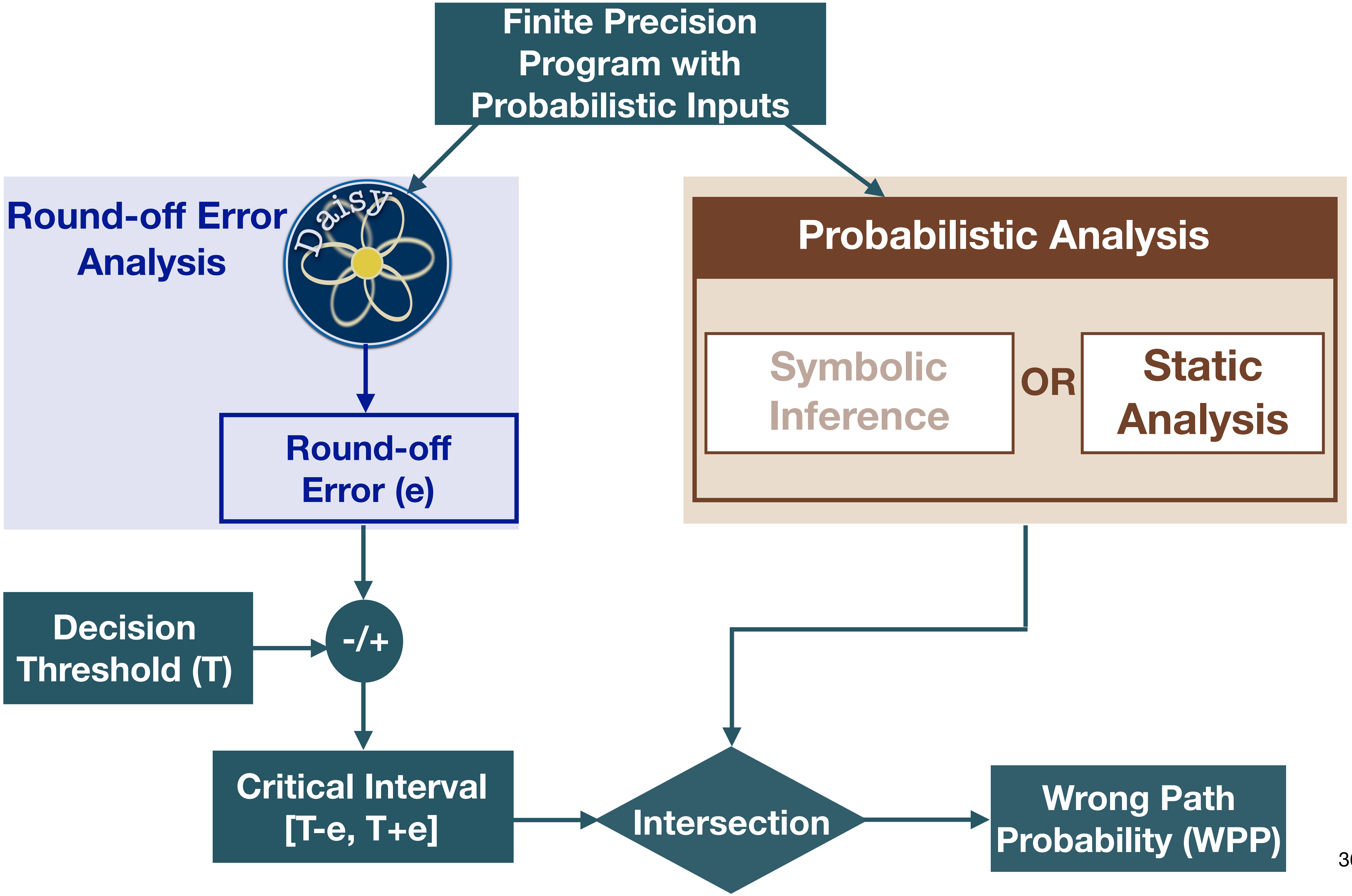
wrong path probability for 32 bit floating-point round-off errors and uniform input distributions

WPP using Symbolic Inference

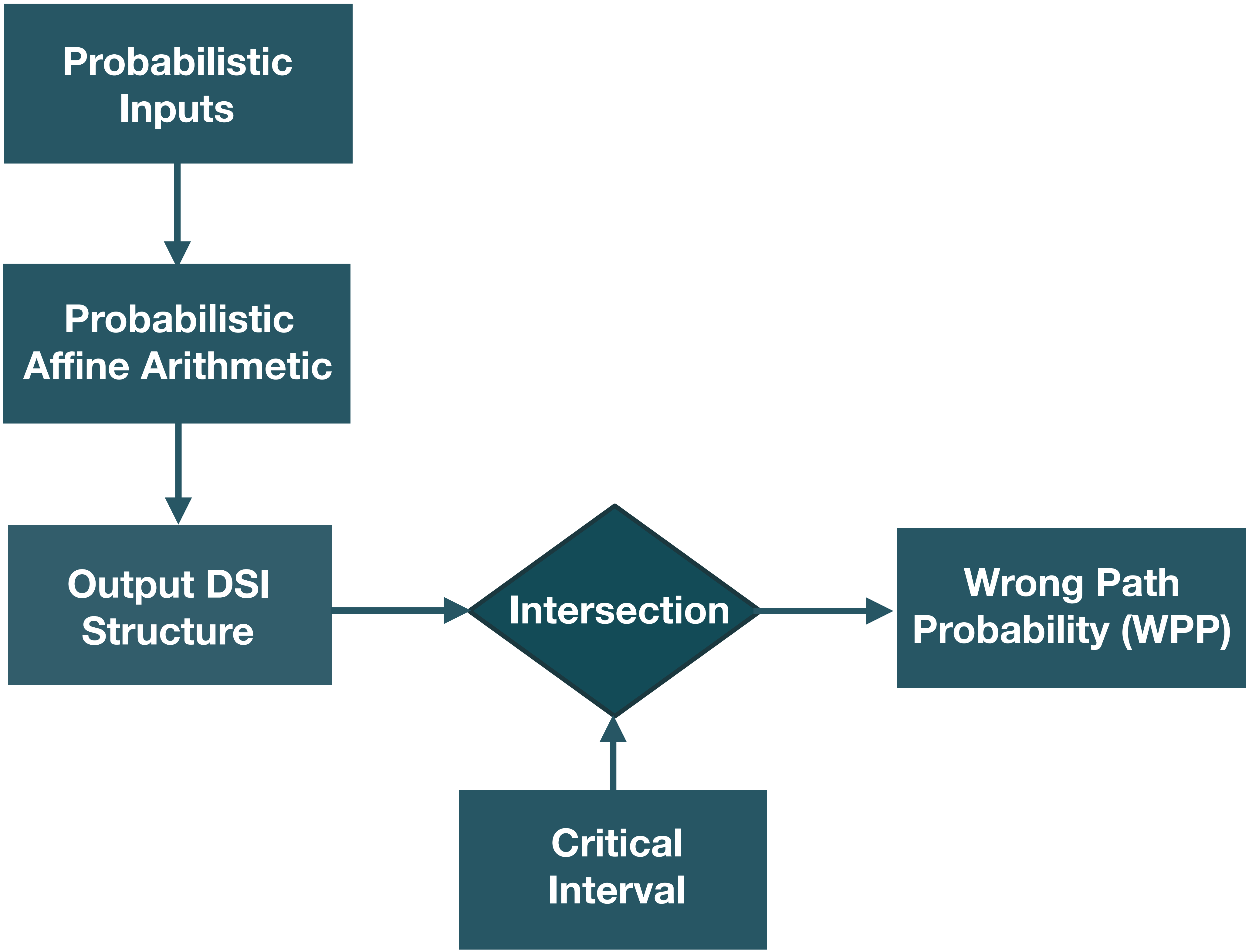
Benchmarks	#ops	**Sym. Inf.**
sine	18	7.61e-7
sqrt	14	8.74e-6
turbine1	14	**TO**
traincar2	13	**TO**
doppler	10	**TO**
bspline1	8	2.54e-6
rigidbody1	7	**TO**
traincar1	7	**TO**
bspline0	6	1.05e-5
sineorder3	4	1.90e-6

wrong path probability for 32 bit floating-point round-off errors and uniform input distributions

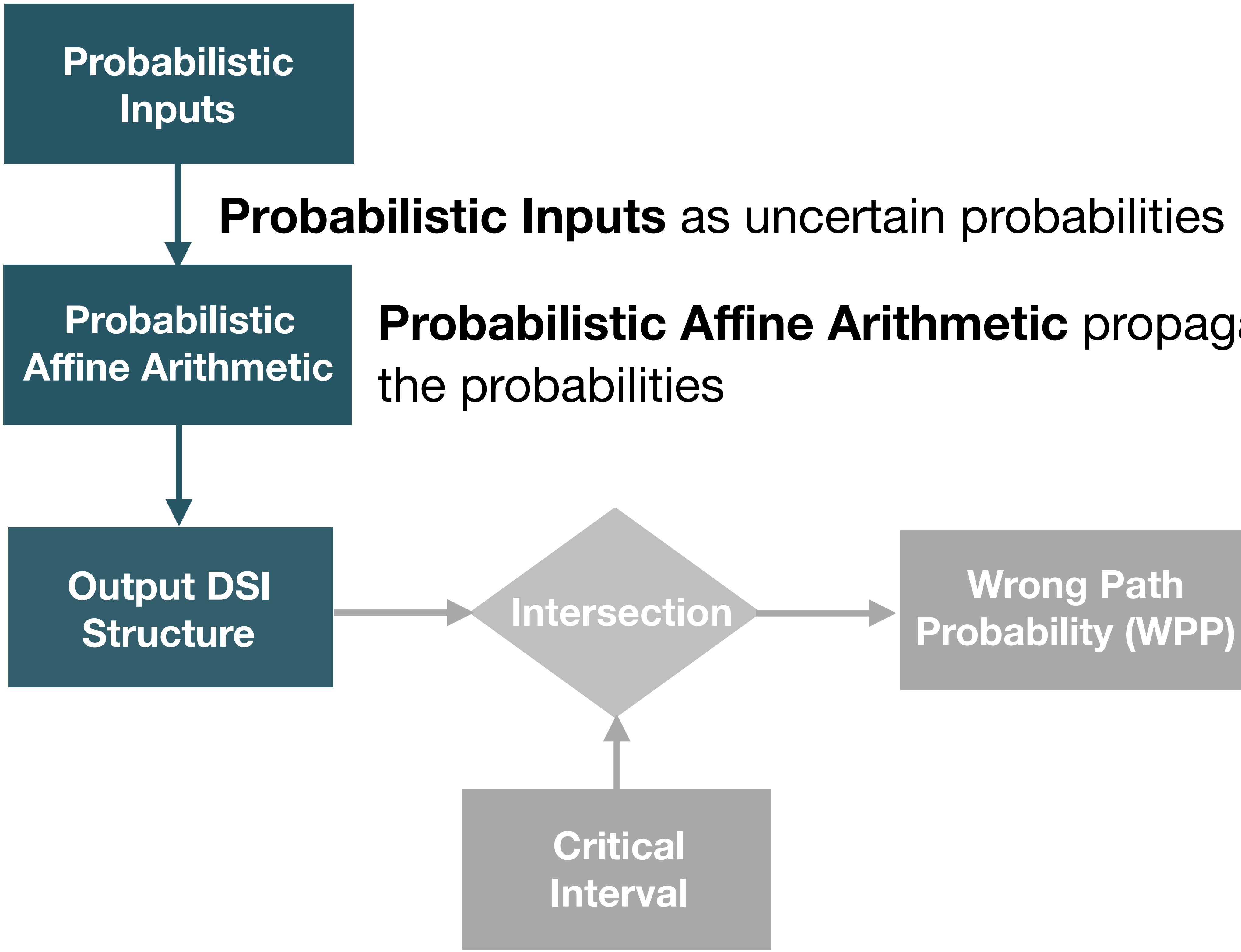
Distribution Propagation



Probabilistic Static Analysis



Probabilistic Static Analysis

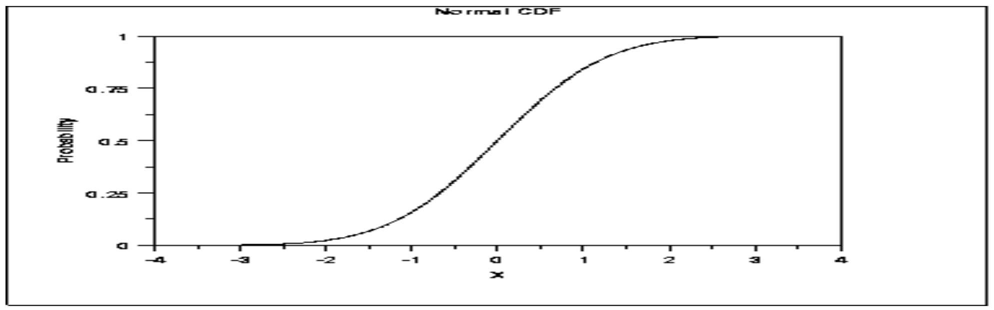


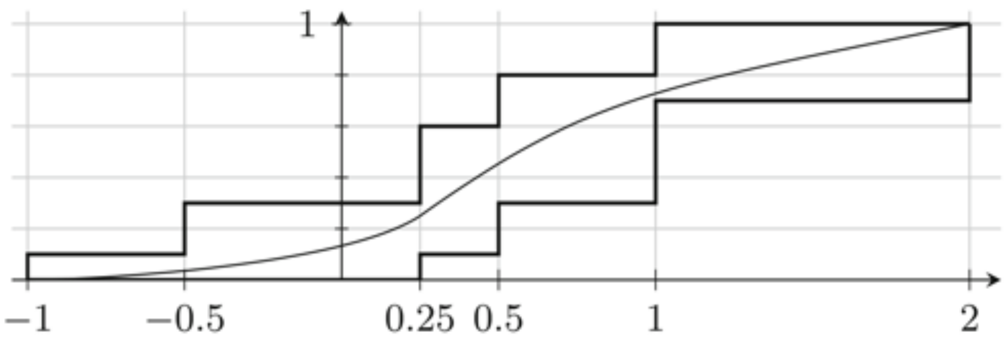
Probabilistic Inputs as uncertain probabilities

Probabilistic Affine Arithmetic propagates the probabilities

Dempster Shafer Interval (DSI) Structure

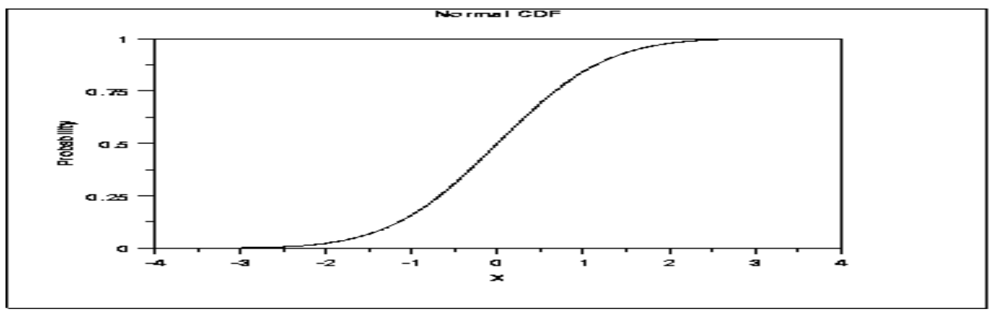
Discretizes the continuous distribution into sets of intervals and weights

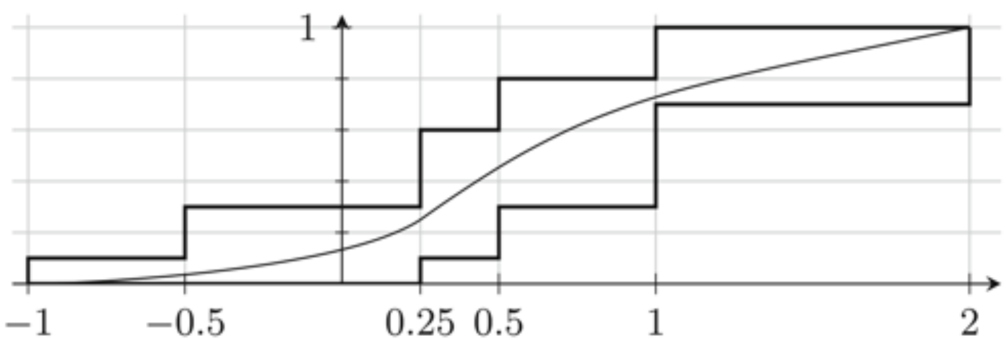




Dempster Shafer Interval (DSI) Structure

Discretizes the continuous distribution into sets of intervals and weights





- Number of discretization is fixed
- Divides the input range

```
{<[-1.0, -0.5], 0.1>, <[-0.5, 0.25], 0.2>........}
```

Focal Element

Dempster Shafer Interval (DSI) Arithmetic

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

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

$$z = x \square y, (\square = + , - , \times , \div)$$

x, y are independent

- Interval arithmetic for intervals
- Weights are multiplied

Dempster Shafer Interval (DSI) Arithmetic

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

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

$$z = x \square y, (\square = + , - , \times , \div)$$

x, y are independent

- Interval arithmetic for intervals
- Weights are multiplied

x, y are dependent

- 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

- Interval arithmetic for intervals
- Weights are multiplied

- 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

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

Noise Symbol

"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

DSI $d_x :< [a_1, b_1], w_1 > , \cdots < [a_n, b_n], w_n >$

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

Noise Symbol

"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

DSI $d_x :< [a_1, b_1], w_1 > , \cdots < [a_n, b_n], w_n >$

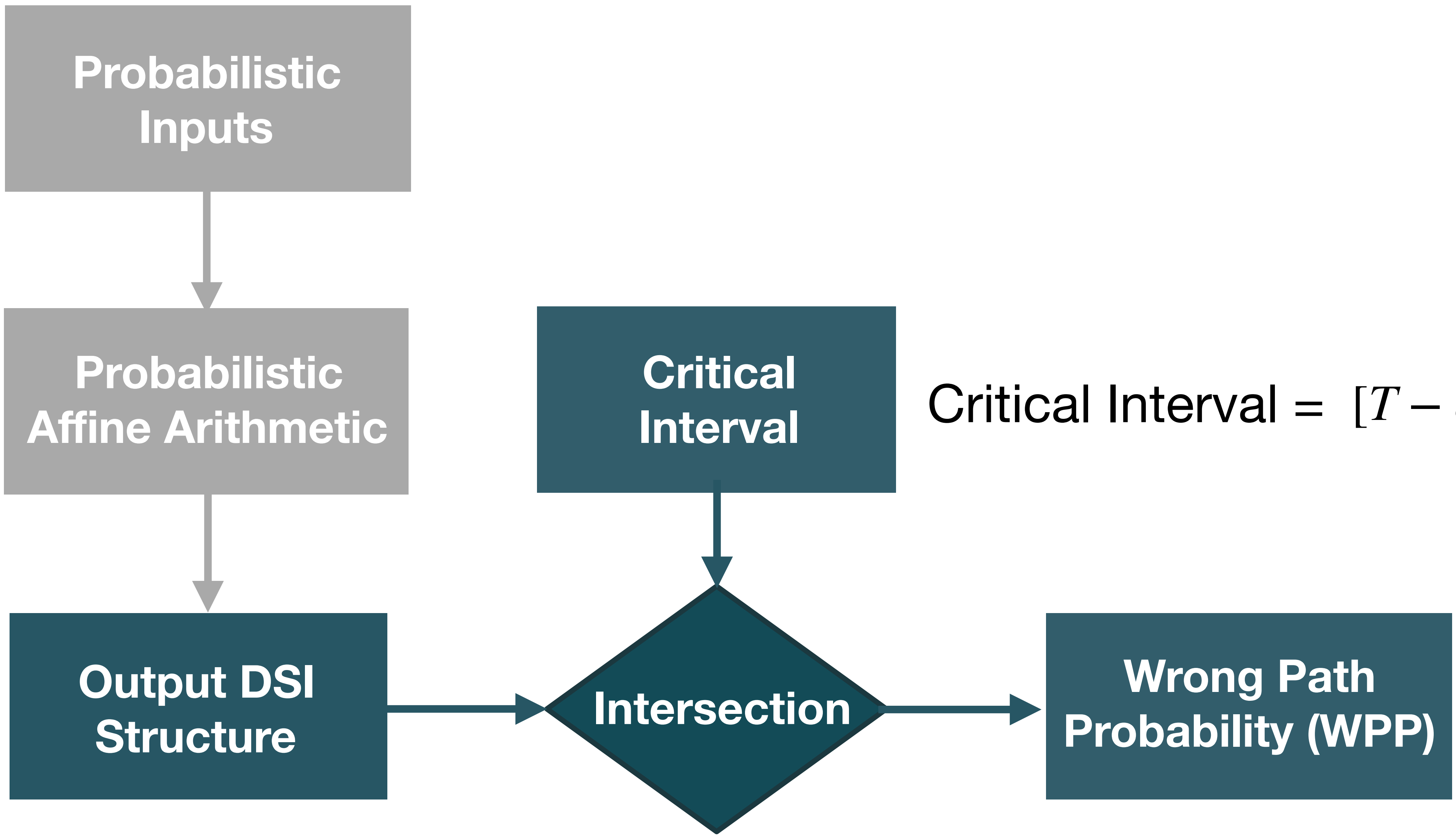
$$\hat{x} := x_0 + \sum_{i=1}^{p} x_i \epsilon_i,\ \epsilon_i \in [-1,1]$$

Noise Symbol

- 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

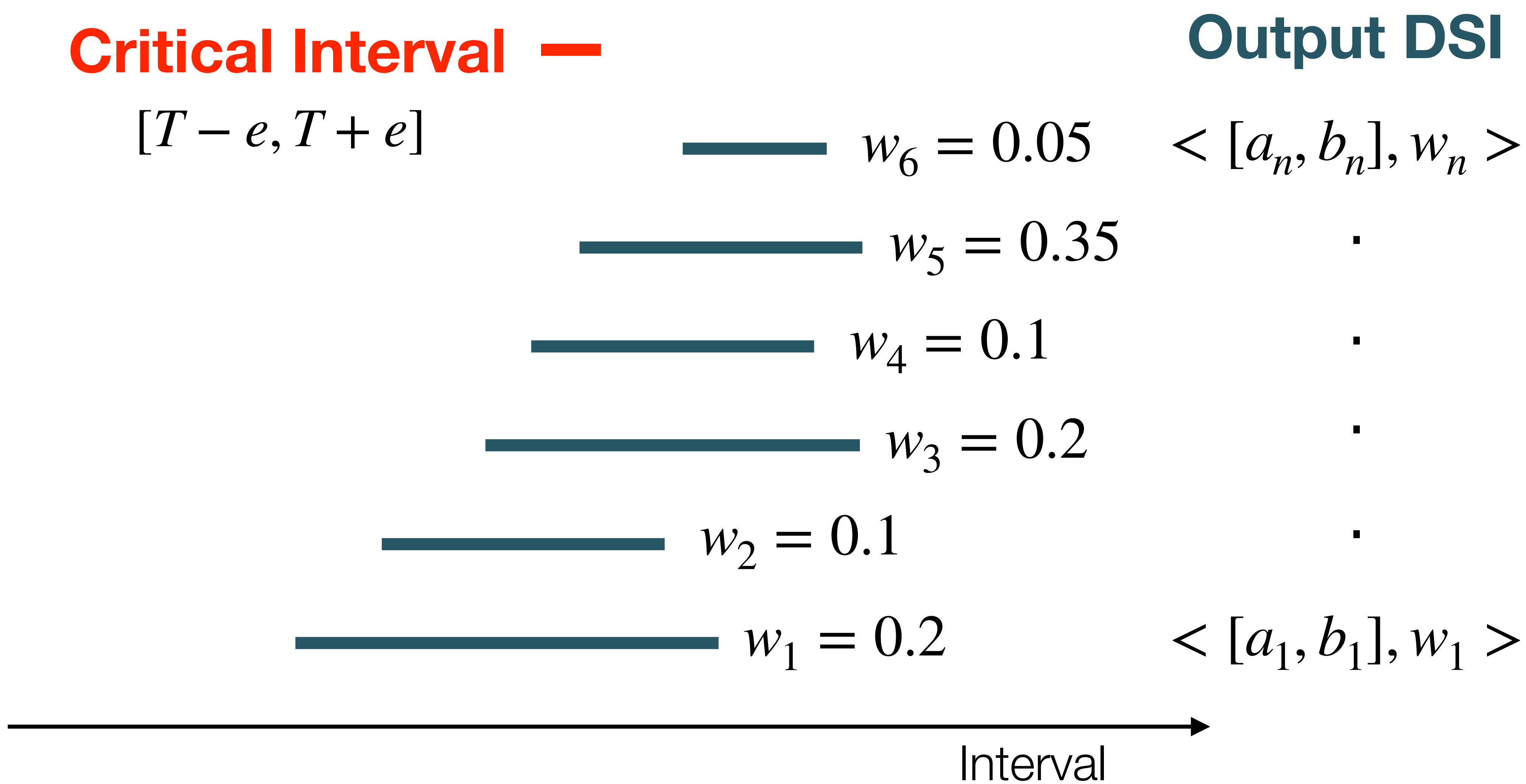
Probabilistic Static Analysis



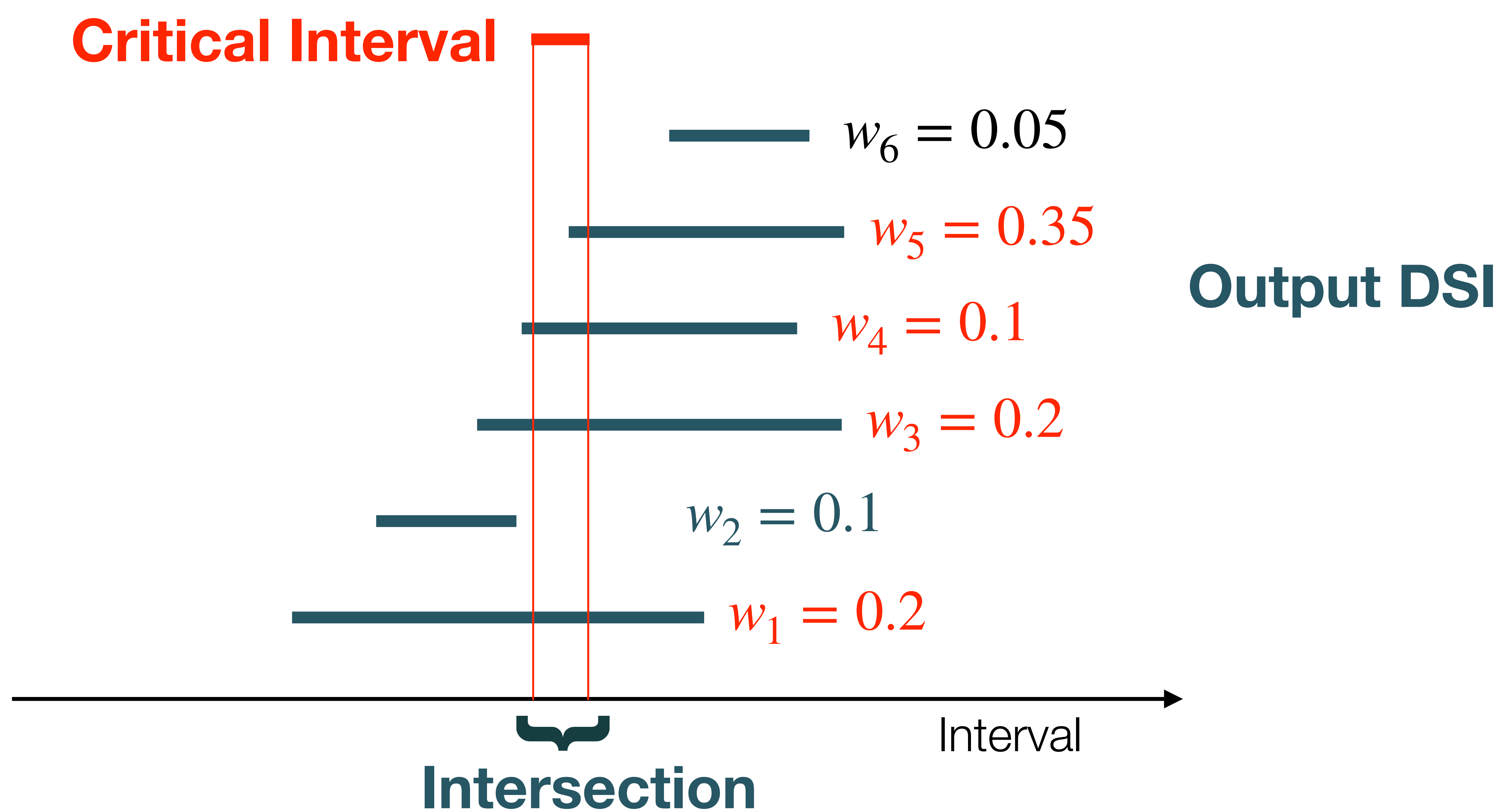
Critical Interval = $[T - e, T + e]$

Output DSI = $d_x :< [a_1, b_1], w_1 > , \cdots < [a_n, b_n], w_n >$

Intersection

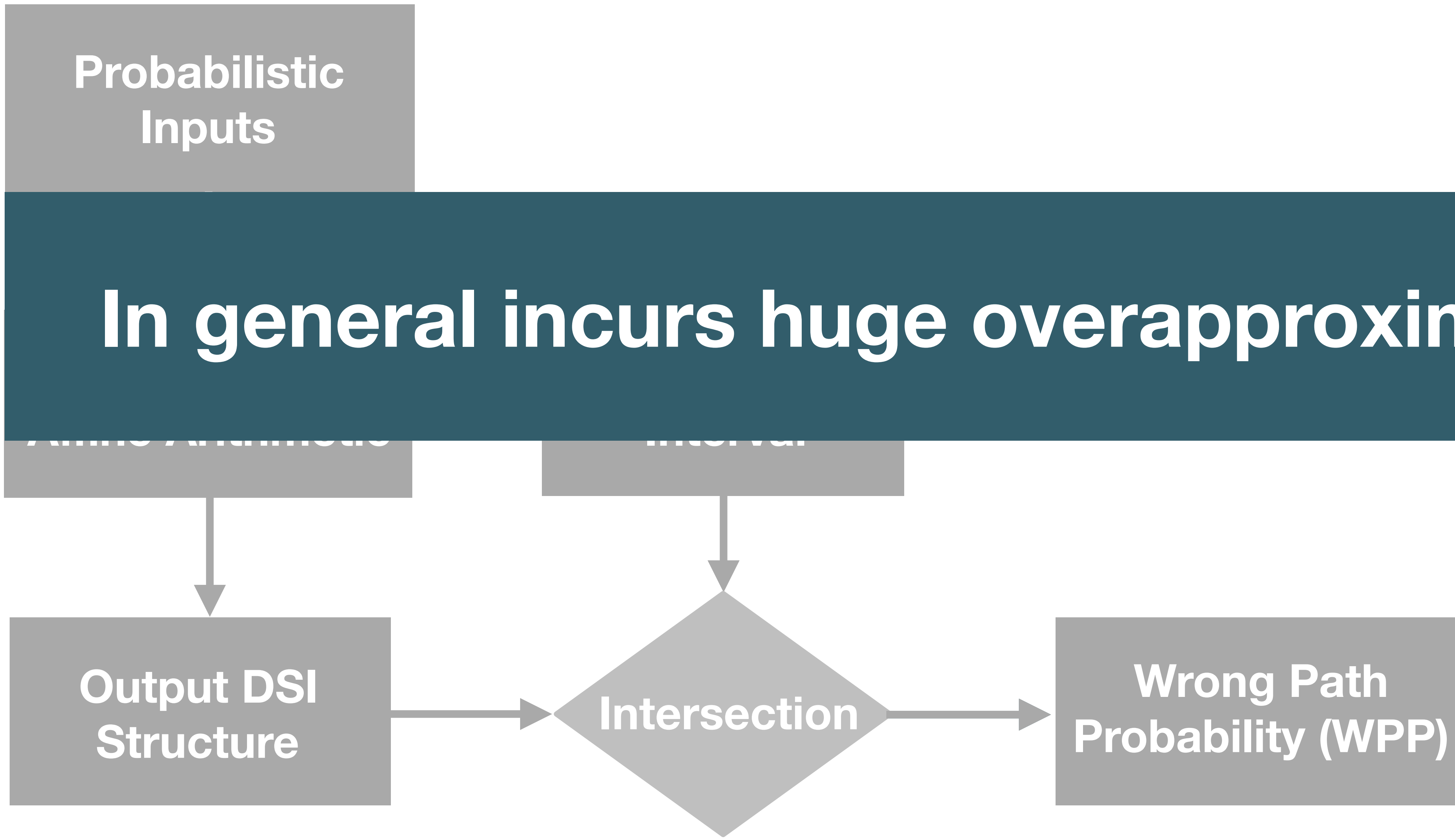


Intersection



Wrong Path Probability = `0.2 + 0.2 + 0.1 + 0.35`
`= 0.85`

Probabilistic Static Analysis



In general incurs huge overapproximation

Probabilistic Static Analysis

Probabilistic Inputs

In general incurs huge overapproximation

Output DSI Structure

```
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()
}
```

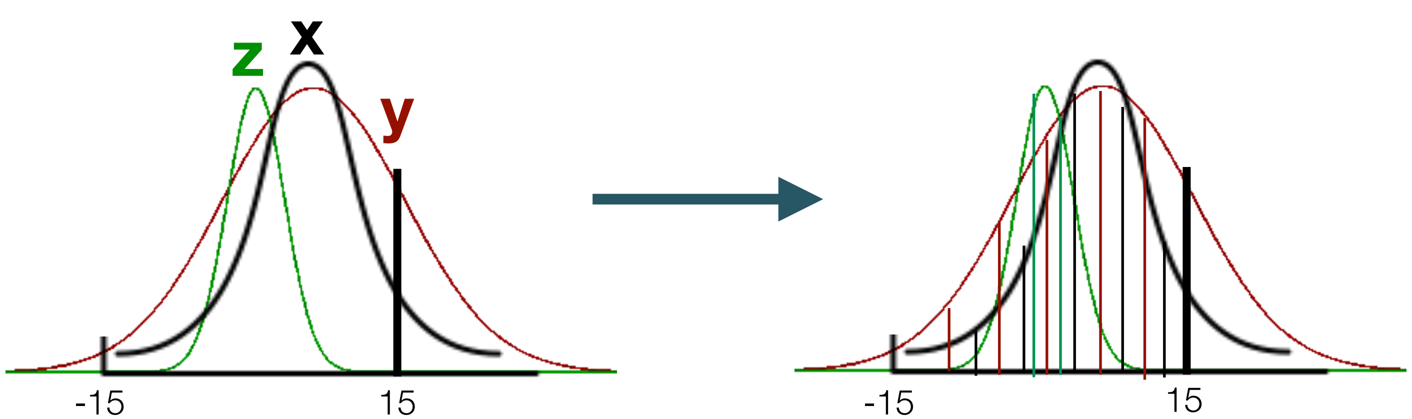
WPP = 1.0

WPP using Probabilistic Static Analysis

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

wrong path probability for 32 bit floating-point round-off errors and uniform input distributions

Interval Subdivision



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

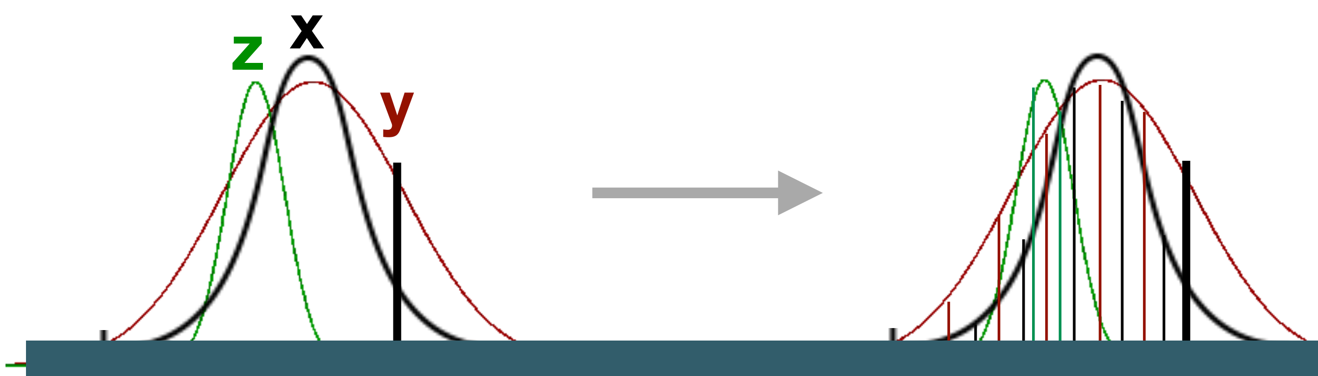
Input Intervals

n subdivided Intervals

$\rho_{ijk} \leftarrow$ **Probabilistic Analysis**

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

Interval Subdivision



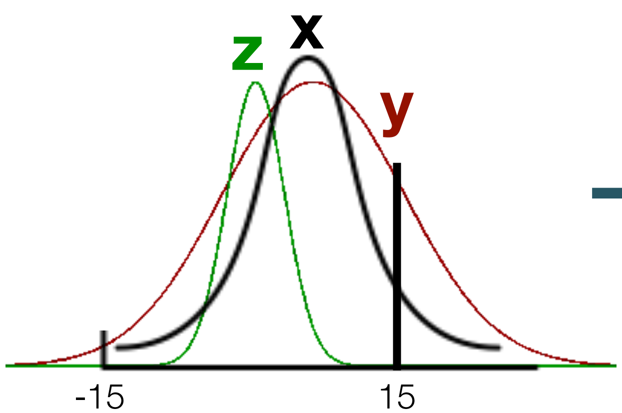
Probabilistic Analysis is costly

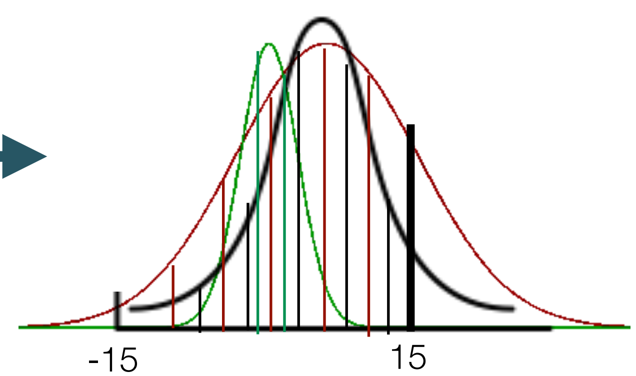
Total WPP =

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

ρ_{ijk} = Probabilistic Analysis

Pruning subdomains by reachability checks





Is the critical interval reachable?

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

1 sec

Input Intervals → **n subdivided Intervals** → Z3

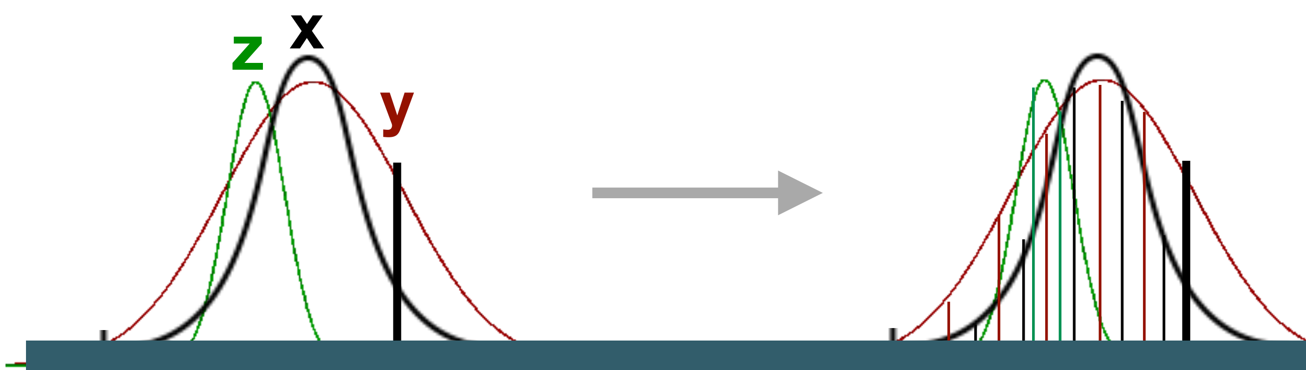
not reachable → **Wrong Path Probability** $\rho_{ijk} = 0.0$

reachable / don't know → $\rho_{ijk} \leftarrow$ **Probabilistic Analysis**

Total WPP =

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

Interval Subdivision + Reachability



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

not reachable

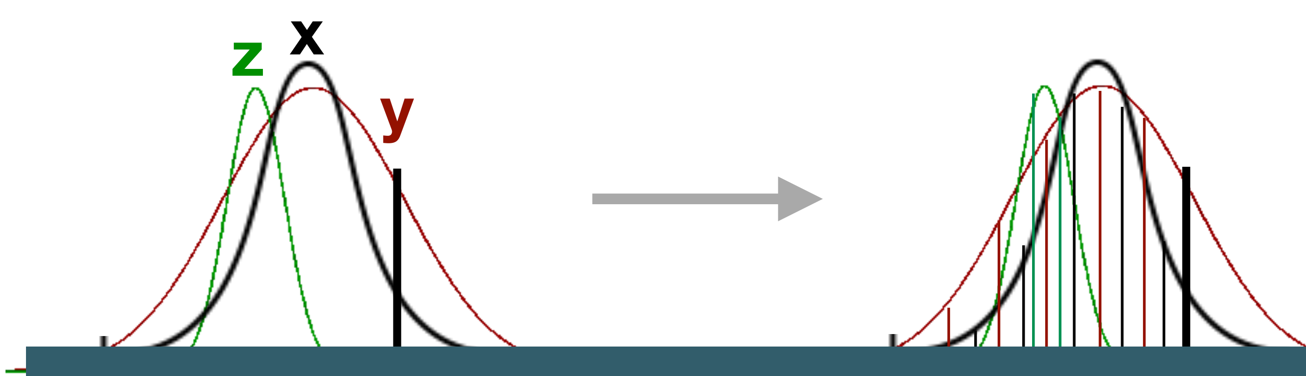
reachable / don't know

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

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

Wrong Path Probability
$$\rho_{ijk} = 1.0$$

Interval Subdivision + Reachability



Provides good estimates

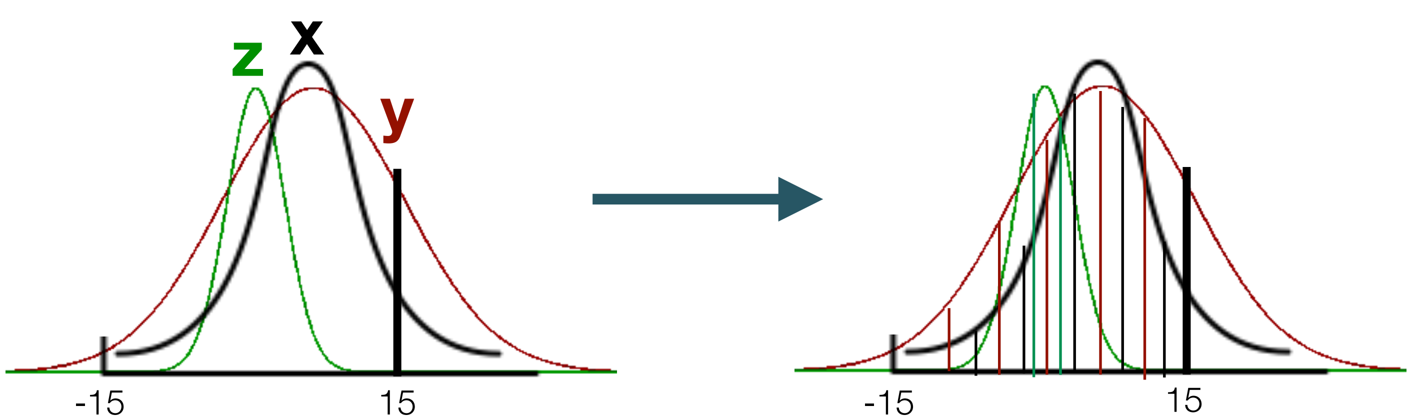
reachable /
don't know

Probabilistic Analysis

```
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?



Is the critical interval reachable?

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

1 sec

Input Intervals → **n subdivided Intervals** → Z3

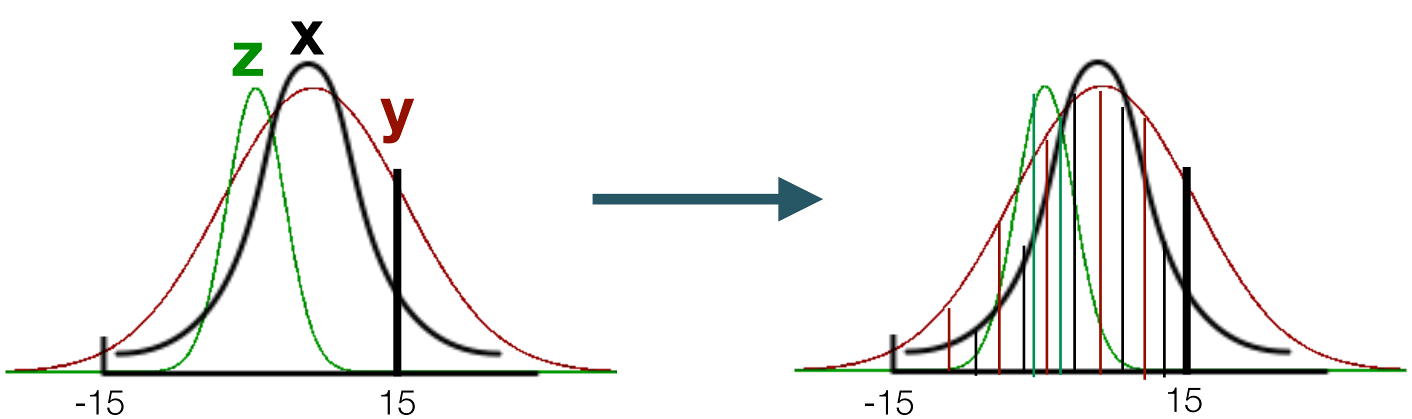
not reachable → **Wrong Path Probability** $\rho_{ijk} = 0.0$

reachable / don't know → ~~$\rho_{ijk} \leftarrow$ **Probabilistic Analysis**~~

Total WPP =

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

Non-Probabilistic Analysis



Is the critical interval reachable?

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

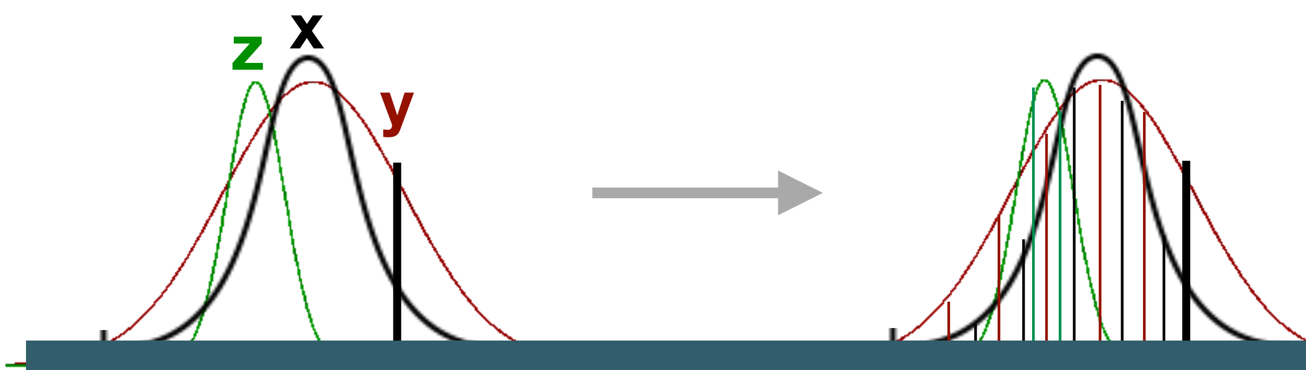
Input Intervals → **n subdivided Intervals** → Z3 (1 sec)

not reachable → **Wrong Path Probability** $\rho_{ijk} = 0.0$

reachable / don't know → **Wrong Path Probability** $\rho_{ijk} = 1.0$

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

Non-Probabilistic Analysis



Works well for univariate functions

reachable /
don't know

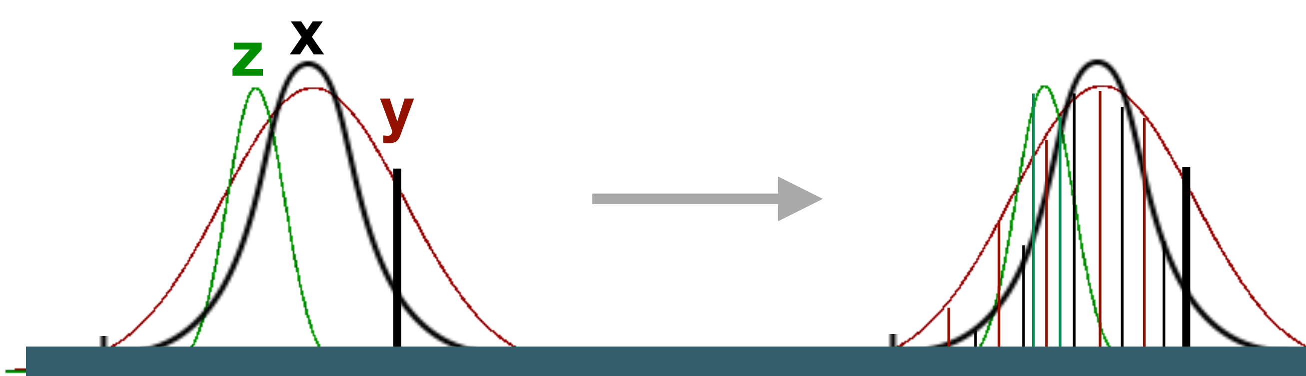
ρ_{to}

```
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()
}
```

Wrong Path Probability
$\rho_{ijk} = 1.0$

WPP = 2.53e-2

Non-Probabilistic Analysis



No. of subdomain: 32000

```
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()
}
```

reachable / don't know

Wrong Path Probability
$\rho_{ijk} = 1.0$

WPP = 2.53e-2 > 4.39e-3

WPP using Probabilistic Analysis with Subdiv

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

wrong path probability for 32 bit floating-point round-off errors and uniform input distributions

WPP using Probabilistic Analysis with Subdiv

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

wrong path probability for 32 bit floating-point round-off errors and uniform input distributions

Case Studies

Case Studies from **embedded systems** and **machine learning**

- **embedded systems**
  - filter
  - traincar
- **machine learning**
  - 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**