b-it Bonn-Aachen International Center for Information Technology

Hochschule Bonn-Rhein-Sieg University of Applied Sciences

Uncertain<T>: Abstraction Technique for Programming with Uncertain Data

Team:
Hammam Abdelwahab
Krishnateja Nallanukala
Manoj Kumar Murugan
Vishnu Vardhan Dadi

Customer & Supervisor:
Deebul Nair

General Idea

Rapid growth in computing powers enabled to shift programmers towards problems with uncertainty.

- Reading data from sensors.
- Approximation of calculations.

General Idea (contd..)

Existing programming languages use simple discrete types (integers, floats or booleans) to represent uncertainty in results.

This causes several bugs.

- Random errors by treating estimates as facts.
- Compounded error through computations.
- False positives and negatives by asking the wrong questions.

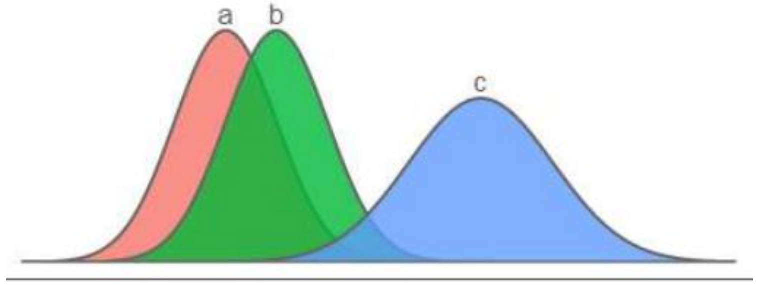
Existing approaches to handle uncertainty in calculations require experience in probabilistic programming.

Uncertain<T>: First-Order Type

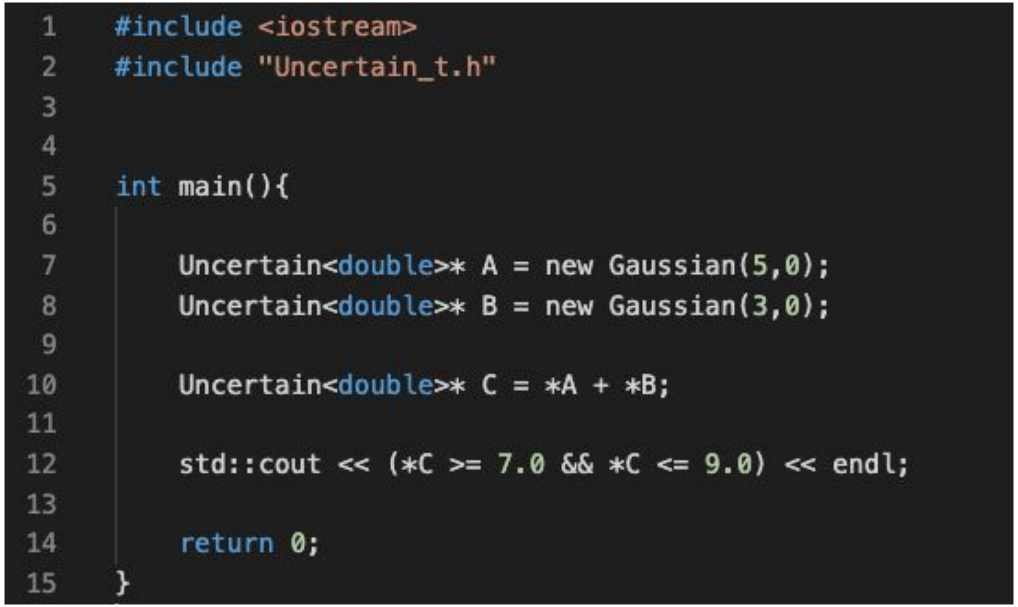
Uncertain<T> is a generic data type that enable to encapsulate and manipulate distributions.

Overloads a variety of operators to enable handling these distributions easily.

Example with adding two gaussian distributions.



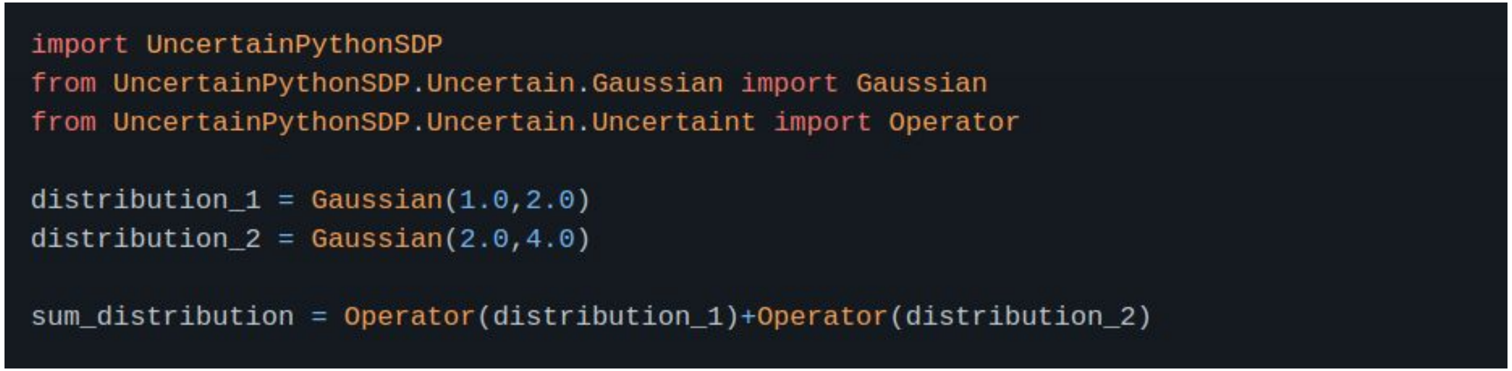
Uncertain<T>: First-Order Type



```cpp
#include <iostream>
#include "Uncertain_t.h"


int main(){

    Uncertain<double>* A = new Gaussian(5,0);
    Uncertain<double>* B = new Gaussian(3,0);

    Uncertain<double>* C = *A + *B;

    std::cout << (*C >= 7.0 && *C <= 9.0) << endl;

    return 0;
}
```

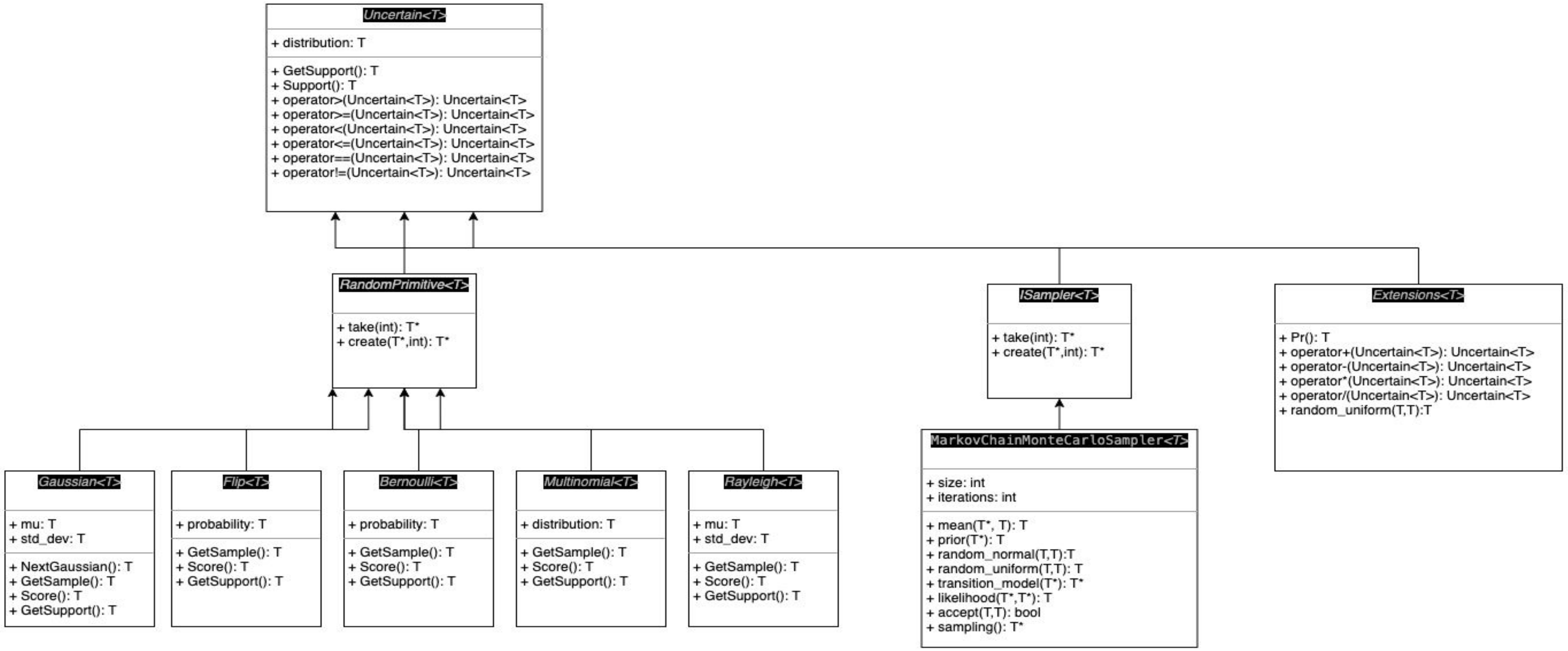
Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Uncertain<T>: First-Order Type



```python
import UncertainPythonSDP
from UncertainPythonSDP.Uncertain.Gaussian import Gaussian
from UncertainPythonSDP.Uncertain.Uncertaint import Operator

distribution_1 = Gaussian(1.0,2.0)
distribution_2 = Gaussian(2.0,4.0)

sum_distribution = Operator(distribution_1)+Operator(distribution_2)
```

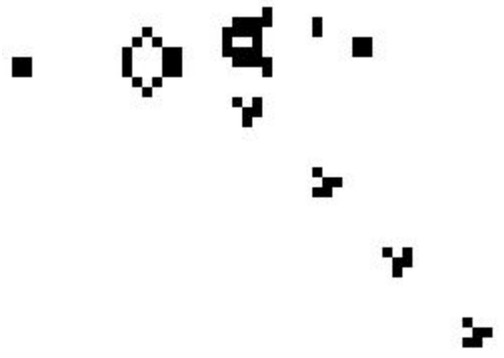
Design



Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

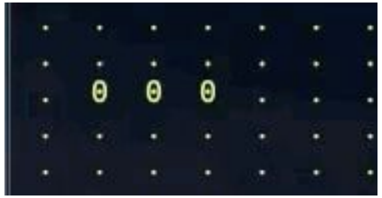
Case Study: Conway’s Game of Life

- **The game**
  - Considers world to be an infinite 2 dimensional grid.
  - The 2D grid consist of cells.
  - A cell is either alive or dead.
  - The status of a cell depends on the status of its neighbouring cell.
- **Rules**
  - A live cell with fewer than two live neighbours dies. (Underpopulation)
  - A live cell with two or three live neighbours lives.
  - A live cell with more than three live neighbours dies (Overpopulation)
  - A dead cell with exactly three live neighbours becomes alive (Reproduction)
- **Significance**
  - Generates interesting patterns


Generated patterns

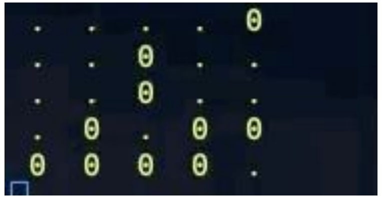
Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Case Study: Conway's Game of Life



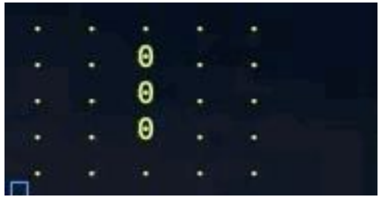
```
.  .  .  .  .  .  .
.  .  .  .  .  .  .
.  0  0  0  .  .  .
.  .  .  .  .  .  .
.  .  .  .  .  .  .
```

Ideal pattern



```
.  .  .  .  0
.  .  0  .  .
.  .  0  .  .
.  0  .  0  0
0  0  0  0  .
```

Pattern with a noisy sensor



```
.  .  .  .  .
.  .  0  .  .
.  .  0  .  .
.  .  0  .  .
.  .  .  .  .
```

Pattern with a noisy sensor and Uncertain<T> data type

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Implementation

Design decisions:

- Software development approach: Agile
- Programming paradigm : Object Oriented Programming
- Python coding convention : PEP 8
- C++ coding convention : General Best Practices

System setup:

- Ubuntu 14.0.4 and above
- Windows 7 and above
- Processor intel i3 and above (recommended)
- Python 2.7 and above
- C++ 11 and above

Dependencies:

- Python version: .2.7,3.x
- numpy version: 1.18.x
- scipy version: 1.4.x
- pytest: 6.2.2
- Catch2 for testing.

Testing

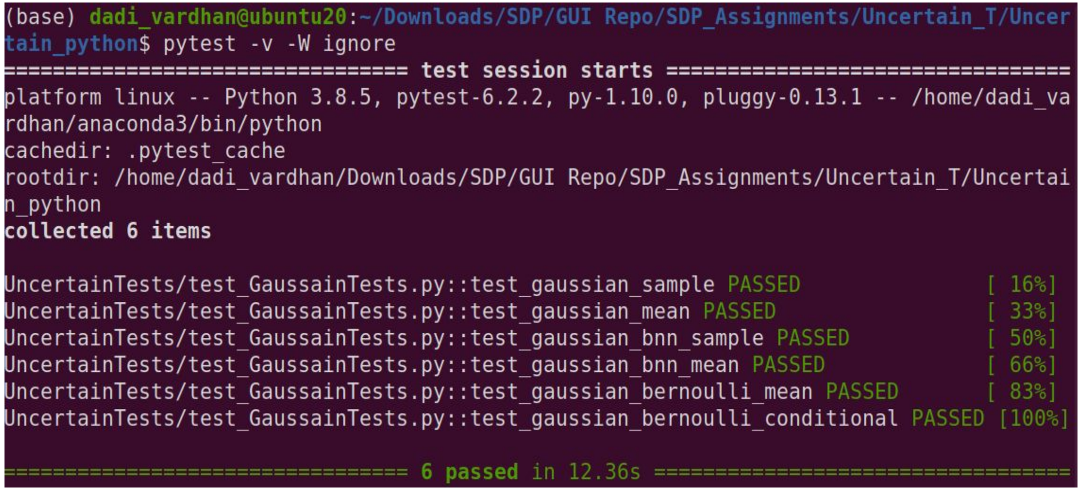
Unit tests: (Sampling from Gaussian, calculating distribution parameters, logical operations).

Continuous Integration Tests

- Github Actions

Link to repo: **Uncertain<T>_python**

Link to repo: **Uncertain<T>_cpp**



```
(base) dadi_vardhan@ubuntu20:~/Downloads/SDP/GUI Repo/SDP_Assignments/Uncertain_T/Uncer
tain_python$ pytest -v -W ignore
================================= test session starts ==================================
platform linux -- Python 3.8.5, pytest-6.2.2, py-1.10.0, pluggy-0.13.1 -- /home/dadi_va
rdhan/anaconda3/bin/python
cachedir: .pytest_cache
rootdir: /home/dadi_vardhan/Downloads/SDP/GUI Repo/SDP_Assignments/Uncertain_T/Uncertai
n_python
collected 6 items

UncertainTests/test_GaussainTests.py::test_gaussian_sample PASSED                [ 16%]
UncertainTests/test_GaussainTests.py::test_gaussian_mean PASSED                  [ 33%]
UncertainTests/test_GaussainTests.py::test_gaussian_bnn_sample PASSED            [ 50%]
UncertainTests/test_GaussainTests.py::test_gaussian_bnn_mean PASSED              [ 66%]
UncertainTests/test_GaussainTests.py::test_gaussian_bernoulli_mean PASSED        [ 83%]
UncertainTests/test_GaussainTests.py::test_gaussian_bernoulli_conditional PASSED [100%]

================================= 6 passed in 12.36s ==================================
```

Capabilities

- Ability to handle arithmetic and logical operations on several distributions.
- Ability to generate random samples given a distribution’s type and parameters.
- A generic data type that works with integers, floats and doubles.

Future Work

- Implementation of other distributions. [Python/C++]
- Proving the remaining case studies.
- Implement Uncertain<T> on high dimensional data types (e.g arrays).

Problems Faced

- Existing c# library is not well commented.
- Implementing this library requires deeper understanding of different probabilistic methods.

Lessons Learnt

- Software development cycle.
- Continuous integration and testing.
- Writing test cases.
- Version control and git.
- Software packaging and distributing.

Thank You