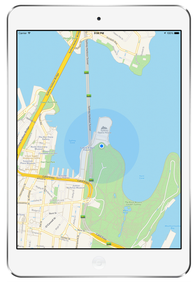
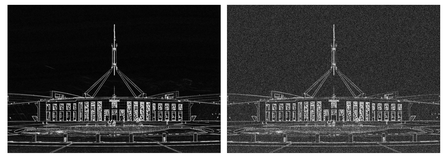
Uncertain<*T*> A First-Order Type for Uncertain Data

James Bornholt
Todd Mytkowicz
Kathryn S. McKinley

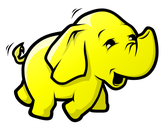
uncertain data



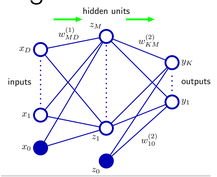
Sensors



Approximate computing



Big data



Machine learning

+ programming languages

Type pretends uncertain data is certain

```
GeoCoordinate L1 = GPS.GetLocation();
Sleep(5);  // Sleep for 5 seconds
GeoCoordinate L2 = GPS.GetLocation();

double Distance = GPS.Distance(L2, L1);
double Speed = Distance / 5;

if (Speed > 4)
    Message("Good job!");
```

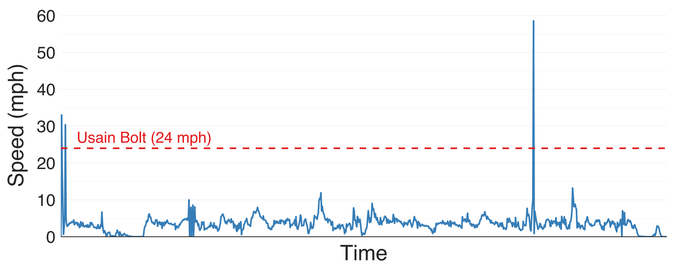
Computation compounds uncertainty

Conditionals create false positives

Translation to Uncertain<*T*> is simple

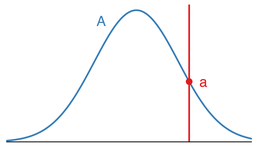
```
Uncertain<GeoCoordinate> L1 = GPS.GetLocation();
Sleep(5);  // Sleep for 5 seconds
Uncertain<GeoCoordinate> L2 = GPS.GetLocation();

Uncertain<double> Distance = GPS.Distance(L2, L1);
Uncertain<double> Speed = Distance / 5;

if (Speed > 4)
    Message("Good job!");
```

= errors!



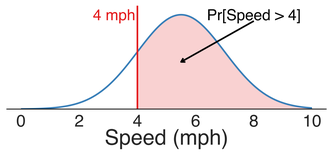
This code results in absurd speeds, such as walking at 59 mph, because the type system encourages developers to pretend the data is precise.

The Uncertain<*T*> abstraction encourages non-expert developers to explicitly reason about uncertainty.



Instances of Uncertain<*T*> are random variables, which have probability distributions.

Overloaded operators provide computations, and define a semantics for conditionals that evaluates evidence.

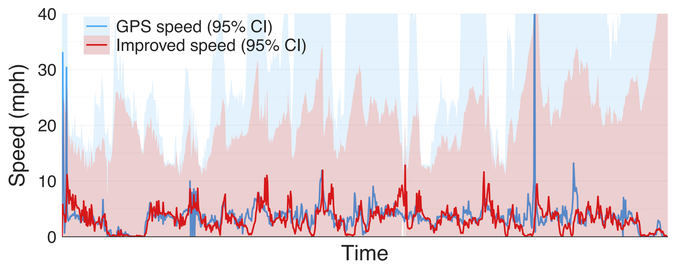



```
if (Speed > 4)
```
More likely than not that Speed > 4?

```
if ((Speed > 4).Pr(0.9)
```
At least 90% likely that Speed > 4?

Our case studies:

Improve accuracy of GPS data.

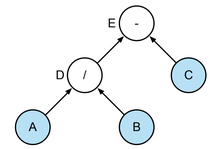


Uncertain<*T*> eases incorporating prior knowledge to improve accuracy of GPS speed estimates, and propagates error through computations to make fewer absurd decisions in conditionals.

Our implementation uses sampling functions and lazy evaluation to efficiently realize Uncertain<*T*>.

Overloaded operators create Bayesian networks, a symbolic representation that defers actual computation.

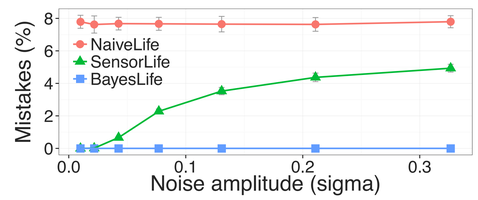
```
D = A / B
E = D - C
```



The runtime evaluates conditionals by sampling these networks, using a hypothesis test to choose which side of the branch to enter.

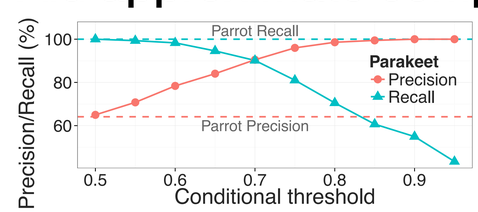
The hypothesis test is goal-oriented, drawing only as many samples as necessary to answer the specific conditional. This technique produces a principled answer to the speed-accuracy trade-off.

Mitigate digital sensor noise.



Uncertain<*T*> makes fewer mistakes in the Game of Life when noise is induced in sensing cell states (SensorLife). Incorporating prior knowledge (BayesLife) eliminates mistakes.

Aid approximate computing.



Uncertain<*T*> helps developers to control the balance between false positives and false negatives from approximate computations. Naïve approaches fix a single balance.

Australian National University

With Uncertain<*T*>, application writers don't need a PhD in statistics to use uncertain data.

Microsoft Research