Uncertain<T> makes programs more expressive and more correct

Uncertainty bugs: Applications use estimated data, but languages use discrete types

Four steps to using Uncertain<T>, the uncertain type abstraction

1. Identifying distributions

Experts provide measurement models along with uncertain data sources to create distributions.

Uncertain<T> approximates arbitrary distributions by Monte Carlo random sampling.

2. Computing with estimates

Uncertain<T> lifts arithmetic operators over type T to act over distributions of T, which captures the effect of compounding error when making calculations.

Sample size trades speed for accuracy.

3. Asking the right questions

The semantics of conditionals are unclear when distributions are involved. There is a probability that A < B; so comparisons have type Unc<T> -> Unc<T> -> Bernoulli. But this does not preserve the total order on T if it has one.

Alternatively we can use expected values: the expected value of an Uncertain<T> is of type T, so preserves the order properties of T.

Uncertain<T> uses hypothesis tests to control approximation error.

4. Improving estimates

Distributions allow experts to apply Bayesian methods: combine evidence (observed data, E) with a hypothesis (prior knowledge, H) to obtain better estimates (a new hypothesis, H|E).

For example, we can apply a physics model to the GPS speed data from above. The model removes the absurd values and tightens the confidence interval.

Uncertain<T> makes programs more expressive and more correct

Using estimates as facts

Discrete types ignore random error. 95% of WinPhone apps ignore GPS error.

Computation compounds error

Calculations induce more uncertainty, making results even more inaccurate.

Inference asks wrong questions

Code asks boolean questions, but estimated data is probabilistic.

Location LastLoc = GetGPSLoc();
while (true) {
    Sleep(dt);
    Location Loc = GetGPSLoc();
    PlotOnMap(Loc);
    double Speed = Dist(LastLoc, Loc) / dt;
    if (Speed > 65)
        SpeedLimitWarning();
    LastLoc = Loc;
}

This app measures speeds with GPS, and reports absurd data like walking at 59 mph. Compounding error causes a wide 95% confidence interval.

Can application writers handle estimated data without requiring a PhD in statistics?