Programming the Internet of Uncertain <T>hings

James Bornholt  
Na Meng  
Todd Mytkowicz  
Kathryn S. McKinley

University of Washington  
University of Texas at Austin  
Microsoft Research  
Microsoft Research
GeoCoordinate PrevLocn = Get();
Sleep(5);
GeoCoordinate Location = Get();
GeoCoordinate PrevLocn = Get();
Sleep(5);
GeoCoordinate Location = Get();
double Dist =
    Distance(LastLocn, Location);
double Speed = Dist / 5;
GeoCoordinate PrevLocn = Get();
Sleep(5);
GeoCoordinate Location = Get();
double Dist =
    Distance(LastLocn, Location);
double Speed = Dist / 5;

if (Speed > 4)
    Alert("Keep it up!");
59 mph
sensors

machine learning
sensors

machine learning

approximate computing
Uncertain\(<T>\)

an abstraction for reasoning about noise [ASPLOS’14]

exploiting context

language constructs to make data more accurate
Uncertain\(<T>\)

an abstraction for reasoning about noise [ASPLOS’14]

exploiting context

language constructs to make data more accurate
GeoCoordinate PrevLocn = Get();
Sleep(5);
GeoCoordinate Location = Get();
double Dist =
               Distance(LastLocn, Location);
double Speed = Dist / 5;

if (Speed > 4)
   Alert("Keep it up!");
Uncertain<GeoCoordinate> PrevLocn = Get();
Sleep(5);
Uncertain<GeoCoordinate> Location = Get();
Uncertain<double> Dist =
    Distance(PrevLocn, Location);
Uncertain<double> Speed = Dist / 5;

if (Speed > 4)
    Alert("Keep it up!");
Uncertain<GeoCoordinate> PrevLocn = Get();
Sleep(5);
Uncertain<GeoCoordinate> Location = Get();
Uncertain<double> Dist =
    Distance(LastLocn, Location);
Uncertain<double> Speed = Dist / 5;

if (Speed > 4)
    Alert("Keep it up!");

86% fewer errors
Semantics

**Uncertain<T**> encapsulates probability distributions and hides statistical complexity.

- Computing over random variables
- Deciding conditionals
Computations

Represent distributions by random samples
Computations

Represent distributions by random samples
Computations

Represent distributions by random samples
Computations

Operators build a Bayesian network rather than evaluating immediately.

\[ D = \frac{A}{B} \]
Operators build a Bayesian network rather than evaluating immediately.

\[ D = \frac{A}{B} \]
Computations

Operators build a Bayesian network rather than evaluating immediately.

\[
\begin{align*}
D &= A \div B \\
E &= D - C
\end{align*}
\]
Computations

Operators build a Bayesian network rather than evaluating immediately.

\[
\begin{align*}
D &= A \div B \\
E &= D - C
\end{align*}
\]
Deciding conditionals

```plaintext
if (Speed > 4)
    Alert("Keep it up!");
```
Deciding conditionals

```javascript
if (Speed > 4)
    Alert("Keep it up!");
```
Deciding conditionals

```java
if (Speed > 4)
    Alert("Keep it up!");
```
Deciding conditionals

```python
if (Speed > 4):
    Alert("Keep it up!"瘙);
```

More likely than not that Speed > 4?
if ((Speed > 4).Pr(0.9))
Alert("Keep it up!");
Deciding conditionals

\[
\text{if } ((\text{Speed} > 4).\Pr(0.9)) \\
\text{Alert("Keep it up!");}
\]

At least 90% likely that Speed > 4?
Identifying absurd data

GeoCoordinate PrevLocn = Get();
Sleep(5);
GeoCoordinate Location = Get();
double Dist = 
    Distance(LastLocn, Location);
double Speed = Dist / 5;

if (Speed > 4) // 7 mph
    Alert("That’s crazy!");
Identifying absurd data

```plaintext
GeoCoordinate PrevLocn = Get();
Sleep(5);
GeoCoordinate Location = Get();
double Dist = Distance(LastLocn, Location);
double Speed = Dist / 5;

if (Speed > 4) // 7 mph
    Alert("That’s crazy!");
```

Naive: 30 times
Identifying absurd data

Uncertain\<GeoCoordinate>\> PrevLocn = Get();
Sleep(5);
Uncertain\<GeoCoordinate>\> Location = Get();
Uncertain\<double>\> Dist =
    Distance(LastLocn, Location);
Uncertain\<double>\> Speed = Dist / 5;

if (Speed > 4) // 7 mph
    Alert("That’s crazy!");

Naive: 30 times
Identifying absurd data

```c
Uncertain<GeoCoordinate> PrevLocn = Get();
Sleep(5);
Uncertain<GeoCoordinate> Location = Get();
Uncertain<double> Dist =
    Distance(LastLocn, Location);
Uncertain<double> Speed = Dist / 5;

if (Speed > 4) // 7 mph      Naive: 30 times
    Alert("That’s crazy!");    50%: 4 times
```
Identifying absurd data

Uncertain<GeoCoordinate> PrevLocn = Get();
Sleep(5);
Uncertain<GeoCoordinate> Location = Get();
Uncertain<double> Dist = Distance(LastLocn, Location);
Uncertain<double> Speed = Dist / 5;

if ((Speed > 4).Pr(0.9)) Alert("That’s crazy!");
Naive: 30 times 50%: 4 times
Identifying absurd data

Uncertain<GeoCoordinate> PrevLocn = Get();
Sleep(5);
Uncertain<GeoCoordinate> Location = Get();
Uncertain<double> Dist =
    Distance(LastLocn, Location);
Uncertain<double> Speed = Dist / 5;

if ((Speed > 4).Pr(0.9))
    Alert("That’s crazy!");

Naive: 30 times
50%: 4 times
90%: never
Uncertain<\(T\)>

an abstraction for reasoning about noise [ASPLOS’14]

exploiting context

language constructs to make data more accurate
Uncertain<
an abstraction for reasoning about noise

exploiting context
language constructs to make data more accurate
if (RecognizeBeard(photo))
    AddBeardToAvatar();
if (RecognizeBeard(photo))
    AddBeardToAvatar();
City userCity = ...;

if (RecognizeBeard(photo))
    AddBeardToAvatar();
City userCity = ...;

if (RecognizeBeard(photo))
    AddBeardToAvatar();

context
application-specific
domain knowledge
Static context

• How does city influence beards?
Static context

- How does city influence beards?
Static context

- How does city influence beards?
  
  ```
  Bernoulli HasBeard_City(City c) {
    if (c == "Seattle")
      return new Bernoulli(0.4);
    else
      return new Bernoulli(0.2);
  }
  ```
• How does city influence beards?

```java
Bernoulli HasBeard_City(City c) {
    if (c == "Seattle")
        return new Bernoulli(0.4);
    else
        return new Bernoulli(0.2);
}
```
• How does city influence beards?

```java
Bernoulli HasBeard_City(City c) {
    if (c == "Seattle")
        return new Bernoulli(0.4);
    else
        return new Bernoulli(0.2);
}

var Cities = Uniform(...);
```
• How does city influence beards?

```plaintext
Bernoulli HasBeard_City(City c) {
    if (c == "Seattle")
        return new Bernoulli(0.4);
    else
        return new Bernoulli(0.2);
}

var Cities = Uniform(...);
```
Static context

• How does city influence beards?

```plaintext
Bernoulli HasBeard_City(City c) {
  if (c == "Seattle")
    return new Bernoulli(0.4);
  else
    return new Bernoulli(0.2);
}

var Cities = Uniform(...);

Bernoulli HasBeard =
  HasBeard_City <| Cities;
```
Dynamic context

• Exploit knowledge about this user

```javascript
var Cities = Uniform(...);
Bernoulli HasBeard =
    HasBeard_City <| Cities;
```
Dynamic context

• Exploit knowledge about this user

```csharp
var Cities = Uniform(...);
Bernoulli HasBeard =
    HasBeard_City <| Cities;

Cities.Value = "Seattle";
```
Dynamic context

• Exploit knowledge about this user

```csharp
var Cities = Uniform(...);
Bernoulli HasBeard =
    HasBeard_City <| Cities;

Cities.Value = "Seattle";

Bernoulli oldHasBeard = BeardRecognizer(photo);
```
Dynamic context

- Exploit knowledge about this user
  
  ```
  var Cities = Uniform(...);
  Bernoulli HasBeard =
  HasBeard_City <| Cities;
  Cities.Value = "Seattle";
  Bernoulli oldHasBeard = BeardRecognizer(photo);
  Bernoulli newHasBeard = oldHasBeard # HasBeard;
  ```
Dynamic context

• Exploit knowledge about this user

```c
var Cities = Uniform(...);
Bernoulli HasBeard =
    HasBeard_City <| Cities;

Cities.Value = "Seattle";

Bernoulli oldHasBeard = BeardRecognizer(photo);

Bernoulli newHasBeard = oldHasBeard # HasBeard;

if (newHasBeard)
    AddBeardToAvatar();
```
Dynamic context

• Exploit knowledge about this user

```csharp
var Cities = Uniform(...);
Bernoulli HasBeard =
    HasBeard_City <| Cities;

Cities.Value = "Seattle";

Bernoulli oldHasBeard = BeardRecognizer(photo);

Bernoulli newHasBeard = oldHasBeard # HasBeard;

if (newHasBeard.Pr(0.9))
    AddBeardToAvatar();
```
Two new constructs

< |
Building probability distributions
(conditional probability)

#
Composing context and estimates
(Bayesian inference)
Implementation

• We designed a sequential likelihood reweighting algorithm to implement this abstraction

• But we’d like to compile down to probabilistic programming languages, which have better inference
Simon Says

- Stomp your feet
- Touch your nose
- Turn around
- Shake your head
Simon Says

• **Xbox Kinect gesture recognition API**

![Bar Chart]

- **F₁-score**
- **Person #1**
- **Person #2**
- **Person #3**
- **Person #4**
- **Person #5**
- **Person #6**
- **Person #7**

- **Gesture1**
- **Gesture2**
- **Gesture3**

![Graph]

<table>
<thead>
<tr>
<th>Person</th>
<th>Gesture1</th>
<th>Gesture2</th>
<th>Gesture3</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>0.85</td>
<td>0.70</td>
<td>0.90</td>
</tr>
<tr>
<td>#2</td>
<td>0.70</td>
<td>0.60</td>
<td>0.80</td>
</tr>
<tr>
<td>#3</td>
<td>0.50</td>
<td>0.45</td>
<td>0.75</td>
</tr>
<tr>
<td>#4</td>
<td>0.90</td>
<td>0.80</td>
<td>1.00</td>
</tr>
<tr>
<td>#5</td>
<td>0.80</td>
<td>0.70</td>
<td>0.95</td>
</tr>
<tr>
<td>#6</td>
<td>0.60</td>
<td>0.50</td>
<td>0.80</td>
</tr>
<tr>
<td>#7</td>
<td>0.90</td>
<td>0.80</td>
<td>1.00</td>
</tr>
</tbody>
</table>
Simon Says

Local model

• Specific to our user
• Need lots of examples to avoid noise
Simon Says

Local model

• Specific to our user
• Need lots of examples to avoid noise

Global (API) model

• Trained over many users
• Generalises well
Simon Says

Local model

• Specific to our user
• Need lots of examples to avoid noise

Global (API) model

• Trained over many users
• Generalises well

We’d like to keep both!
Simon Says

• Personalised gesture recognition model

Person

#1 #2 #3 #4 #5 #6 #7

Gesture1 Gesture2 Gesture3

F₁-score

0 0.25 0.5 0.75 1
Uncertain\langle T \rangle
an abstraction for reasoning about noise [ASPLOS’14]

exploiting context
language constructs to make data more accurate
Uncertain\(<T>\)

an abstraction for reasoning about noise [ASPLOS’14]

Thanks!

exploiting context

language constructs to make data more accurate