Probabilistic Declarative Module

Reactive Probabilistic Programming

Pedro Zuidberg Dos Martires, Sebastijan Dumančić

pedro.zuidbergdosmartires@cs.kuleuven.be, sebastijan.dumancic@cs.kuleuven.be

How to Handle Event-Driven Streams of Data within Probabilistic Programming?

Reactive Programming

An ad service provider has to decide which ads to display on a customer’s screen. Future ad placements will then depend on a customer’s actions on the website. Example code in ReactiveX (reactivex.io).

```javascript
var clickResultSets = Mouseclicks.
  .map(click => getFromDatabase(click))
  .takeUntil(mouseclicks)
  .concatAll();

clickResultSets.forEach(resultSet => updateClickResults(resultSet));
```

Asynchronous and external events at discrete points in time, such as mouse clicks, drive the execution of a program.

**Behaviors** change continuously over time and are composable first-class citizens in the reactive programming paradigm.

**Events** refer to streams of value updates to time-dependent variables (behaviors). Events occur at discrete points in time and are composable first-class citizens.

We stress the need for APIs to probabilistic programming languages!

Proof of concept implemented in the existing probabilistic programming language Distributional Clauses.

https://bitbucket.org/problog/dc_problog
https://github.com/ML-KULeuven/PyDC

Deterministic Imperative Module

```python
from pydc import DDC

# load DDC program and initialize 500 particles
ddc = DDC("weather_brussels_hmm.pl", 500)

# proceed one time step and query the state
p_hot = ddc.query("observation(activity(tinIn))=clean")

# take decision
if p_hot > 0.5: print("wear shorts!")
else: print("wear pants!")
```

Probabilistic Declarative Module

```prolog
% facts
1 city(Brussels) <- true. %true implies the city of Brussels exists in our database
2 % initial state
3 weather(C):0 ~ finite([0.6:rainy, 0.4:sunny]) <- city(C). %we initialize the weather at time step 0
4 % state model
5 temperature(C):t ~ gaussian(10,6) <- weather(C):t = rainy. %given rainy weather the mean of the temperature is 10 degrees Celsius
6 temperature(C):t ~ gaussian(24,8) <- weather(C):t = sunny. %and 24 for sunny weather
7 % transition model: here we describe how variables at time (t+1) depend on variables at time (t)
8 weather(C):t+1 ~ finite([0.7:rainy, 0.3:sunny]) <- weather(C):t = rainy.
9 weather(C):t+1 ~ finite([0.4:rainy, 0.6:sunny]) <- weather(C):t = sunny.
10 activity(tinIn):t+1 ~ finite([0.1:walk, 0.4:shop, 0.5:clean]) <- weather(brussels):t+1 = rainy.
11 activity(tinIn):t+1 ~ finite([0.6:walk, 0.3:shop, 0.1:clean]) <- weather(brussels):t+1 = sunny.
```