Probabilistic Reactive Programming

Louis Mandel, Guillaume Baudart, Avraham Shinnar, Kiran Kate, Martin Hirzel
IBM Research

Programming reactive systems with uncertainty

Reactive Systems:
Interaction with the environment

- Non terminating processes
- Can not be re-executed

Reactive Programming Languages:
DSL to program reactive systems

- Intuitive formalism
- Dedicated analysis
- Optimized code generation

***ReactiveML*: Reactive extension of OCaml**

- Synchronous model: logical time
- `||`: parallel composition
- `signal`/`emit`/`await`: communication
- `until`/`when`: control structures

***WebPPL*: Probabilistic programming**

- `sample`: draw a value from a distribution
- `factor`: penalize execution path
- `infer`: compute the distribution

References:

- Benveniste, Caspi, Edwards, Halbwachs, Le Guernic, Simone. 2003. *The Synchronous 203 Languages 12 Years Later.*
- Mandel, Pasteur, Pouzet, 2015. *ReactiveML 10 years later.*
- Goodman, Stuhlmüller, 2014. *The Design and Implementation of Probabilistic Programming Languages*
- Ritchie, Stuhlmüller, Goodman, 2016. *C3: Lightweight Incrementalized MCMC for Probabilistic Programs using Continuations and Callsite Caching*

Probabilistic Reactive Models

Extend ReactiveML with probabilistic constructs

Possible Applications

- Online time series prediction
- Agent based systems
- Infrastructure self-tuning

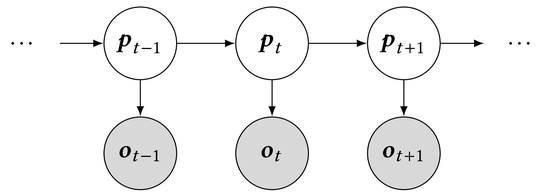


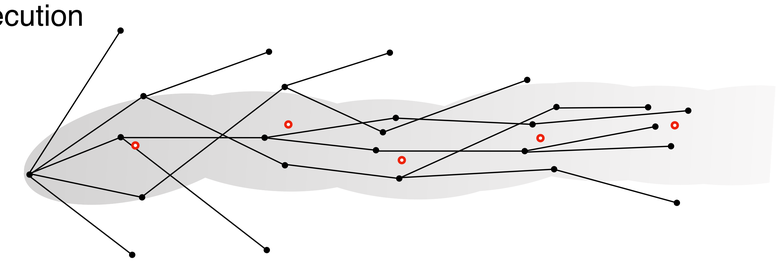
Fig: Graphical Model of the HMM

Example: Hidden Markov Model

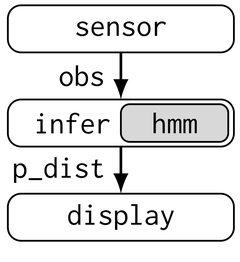
```
let rec process hmm obs p_prev =
  await obs([o_t]) in        (* Input from the environment *)
  let p_t = sample (sph_gaussian p_prev speed) in
  factor (score (sph_gaussian p_t noise) o_t);
  propose p_t;               (* Publish a probabilistic value *)
  run hmm obs p_t
```

Inference: Particles filter on execution paths

- Non terminating function
- `propose` during execution
- No rollback



Hybrid Application

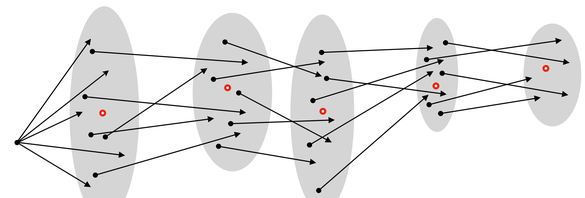



```
let process main =
  signal obs in
  signal p_dist in
  run sensor obs ||
  infer ~output:p_dist (hmm obs [0.;0.]) ||
  run display p_dist
```

Alternative: Instantaneous Model

- Explicit inference at each step
- Sample from the previous distribution

```
let rec process hmm' obs p_dist last_dist =
  await obs(o_t) in
  let dist =
    infer (fun () ->
      let p_prev = sample (last_dist) in
      let p_t = sample (sph_gaussian p_prev speed) in
      factor (score (sph_gaussian p_t noise) o_t);
      p_t)
  in
  emit p_dist dist;
  run hmm' obs p_dist dist
```



Challenges

- Mixing deterministic and probabilistic parts
  - Probabilistic: observe external inputs
  - Deterministic: access inferred distributions
- Handle state in reactive control structures
  - Parallel composition
  - Internal communication
- Inference on non-terminating functions
  - Sequential Monte-Carlo
- Real-time vs. non real-time applications
  - Relax the non-rollback constraint