Reactive Probabilistic Programming a Discussion

Albert Benveniste   Jean-Baptiste Raclet

INRIA Rennes and IRIT Toulouse

November 27, 2019

Basic issues in probabilistic paradigms

Requirements on Probabilistic Programming of React. Syst.

A use case related to safety analysis

Approaches

- Statisticians and AI people
- SW engineering style: Katoen 2017
- Reactive Programming: ProbZelus
- I have a dream

Probabilistic models

- Existing approaches (PA and variations)
- Alternative idea for the probabilistic parallel composition

Mixed (static) Systems (MS) [Benveniste et al.1995]

- Details

Mixed Markov Decision Processes (MMDP)

Link between MMDP and PA [Segala et al.]

Probabilistic Interface Theory (sketch)

Basic issues in probabilistic paradigms

Specifying a probabilistic system

- A probability distribution (Bernoulli, Gaussian,. . .)
- A dynamical system: Markov Chain, HMM, Markov Decision Process, time series,
- CPS subject to random excitation and measurement noise
- Safety analyses,. . .

Issues

- Blending probabilities and nondeterminism
- Modular specification:
  - graphical models
  - OO (like in SW eng)
  - parallel composition
- Conservative extension of reactive systems

Basic issues in probabilistic paradigms

Estimating, learning, inferring

- The parameters of a probability distribution (Bernoulli,. . .)
- The parameters of a dynamical system: Markov Chain,. . .
- An i/o-map (parametric and nonparametric — neural networks)
- The value of some unobserved signal, knowing some observations (filtering and smoothing)

Issues

- Modular specification:
  - Bayesian reasoning & Bayesian networks; Graphical models
- Blending probabilities and nondeterminism
- Estimation/learning/inference algorithms

Basic issues in probabilistic paradigms

Statistical decision and detection, classification

- decide whether $\mathbf{P} \in \mathcal{P}_1$ or $\mathbf{P} \in \mathcal{P}_2$,
  - where $\mathcal{P}_1, \mathcal{P}_2$ are two disjoint sets of proba distributions;
  - Ex: decide if the mean of a Gaussian variable is < 0 or > 0
- detect when $\mathbf{P}_t, t \in \mathbb{R}$ jumps from $\mathcal{P}_1$ to $\mathcal{P}_2$,
  - where $\mathbf{P}_t$ is the distribution of some random signal $X_t, t \in \mathbb{R}$
  - Ex: detect when the mean of a Gauss signal jumps from < 0 to > 0

Issues

- Modular specification:
  - Bayesian reasoning & Bayesian networks; Graphical models
  - Classification?
- Blending probabilities and nondeterminism
- Decision, detection, and classification algorithms

Basic issues in probabilistic paradigms

Requirements on Probabilistic Programming of React. Syst.

A use case related to safety analysis

Approaches

- Statisticians and AI people
- SW engineering style: Katoen 2017
- Reactive Programming: ProbZelus
- I have a dream

Probabilistic models

- Existing approaches (PA and variations)
- Alternative idea for the probabilistic parallel composition

Mixed (static) Systems (MS) [Benveniste et al.1995]

- Details

Mixed Markov Decision Processes (MMDP)

Link between MMDP and PA [Segala et al.]

Probabilistic Interface Theory (sketch)

Requirements on Probabilistic Programming

- Probabilistic programming: offer a high-level language for the
  - specification
  - estimation
  - decision/detection/classification

  of systems involving a mix of proba and nondeterminism

Requirements on Probabilistic Programming

- Probabilistic programming: offer a high-level language for the
  - specification
  - estimation
  - decision/detection/classification

  of systems involving a mix of proba and nondeterminism

- Supporting important nontrivial constructions:
  - Conditioning: $\pi(A \mid B) =_{\mathrm{def}} \frac{\pi(A \cap B)}{\pi(B)}$ provided that $\pi(B) > 0$
  - Modularity in specification, estimation, and decision:
    - Graphical models & Bayesian network reasoning (generalizations of Bayes rule $P(X, Y) = P(X)P(Y|X)$)
    - Parallel composition

Requirements on Probabilistic Programming

- Probabilistic programming: offer a high-level language for the
  - specification
  - estimation
  - decision/detection/classification

  of systems involving a mix of proba and nondeterminism

- Supporting important nontrivial constructions:
  - Conditioning: $\pi(A \mid B) =_{\mathrm{def}} \frac{\pi(A \cap B)}{\pi(B)}$ provided that $\pi(B) > 0$
  - Modularity in specification, estimation, and decision:
    - Graphical models & Bayesian network reasoning (generalizations of Bayes rule $P(X, Y) = P(X)P(Y|X)$)
    - Parallel composition

- Hosting libraries of algorithms for estimation and decision

Requirements on Probabilistic Programming

- Probabilistic programming: offer a high-level language for the
  - specification
  - estimation
  - decision/detection/classification

  of systems involving a mix of proba and nondeterminism

- Supporting important nontrivial constructions:
  - Conditioning: $\pi(A \mid B) =_{\mathrm{def}} \frac{\pi(A \cap B)}{\pi(B)}$ provided that $\pi(B) > 0$
  - Modularity in specification, estimation, and decision:
    - Graphical models & Bayesian network reasoning (generalizations of Bayes rule $P(X, Y) = P(X)P(Y|X)$)
    - Parallel composition

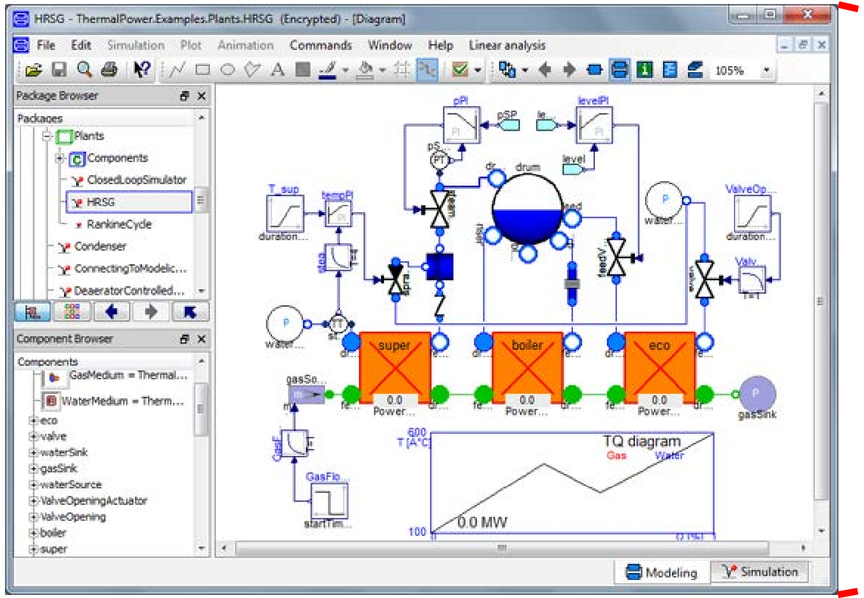
- Hosting libraries of algorithms for estimation and decision

- Providing a layered language for supporting all of this (a conservative extension of a synchronous language)

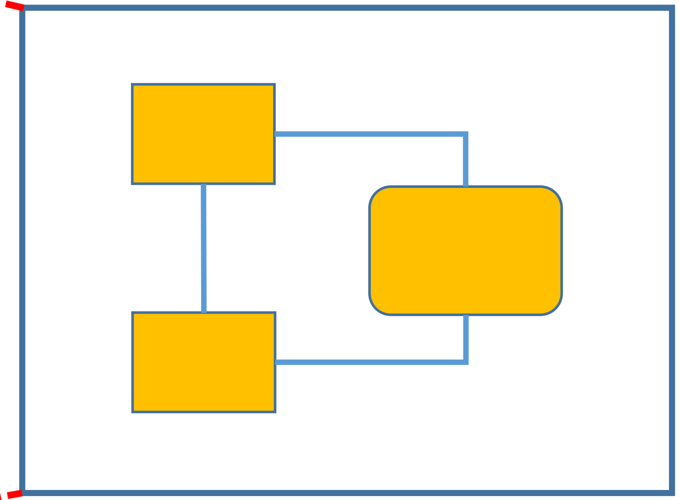
Advantages of a layered language

3 layers:

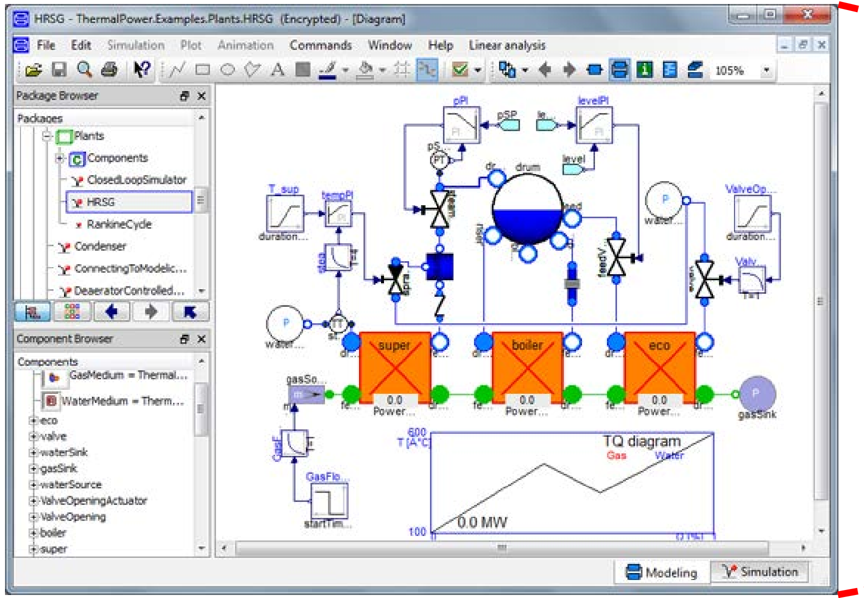
- a probabilistic system
  - semantics, equivalence, rewriting rules
- a statistical problem (probability of some property, sampling, estimating, detecting, classifying,...)
  - semantics, equivalence, rewriting rules
- algorithms for solving statistical problems
  - $\sim$ operational semantics



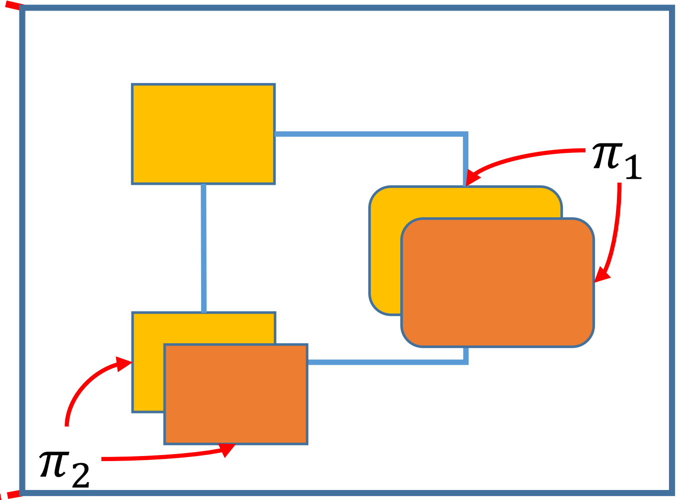
A Modelica model



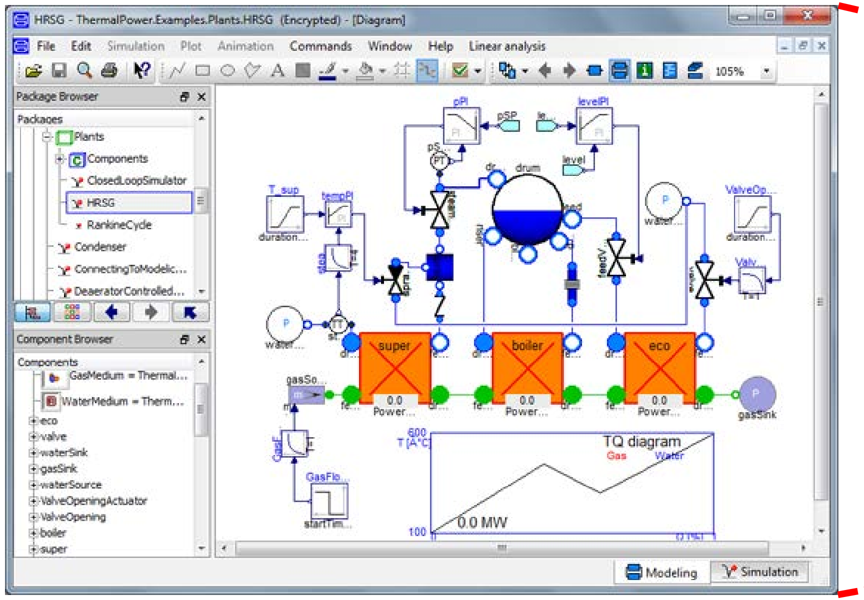
An abstraction of it
(nondeterministic relations)



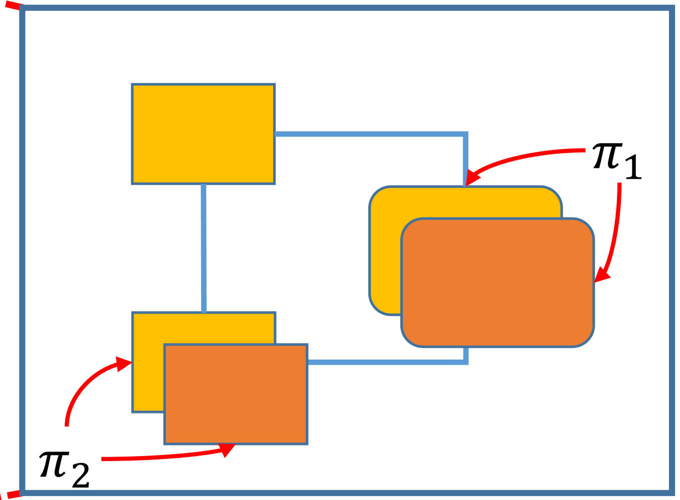
A Modelica model



A safety model
with fault injection



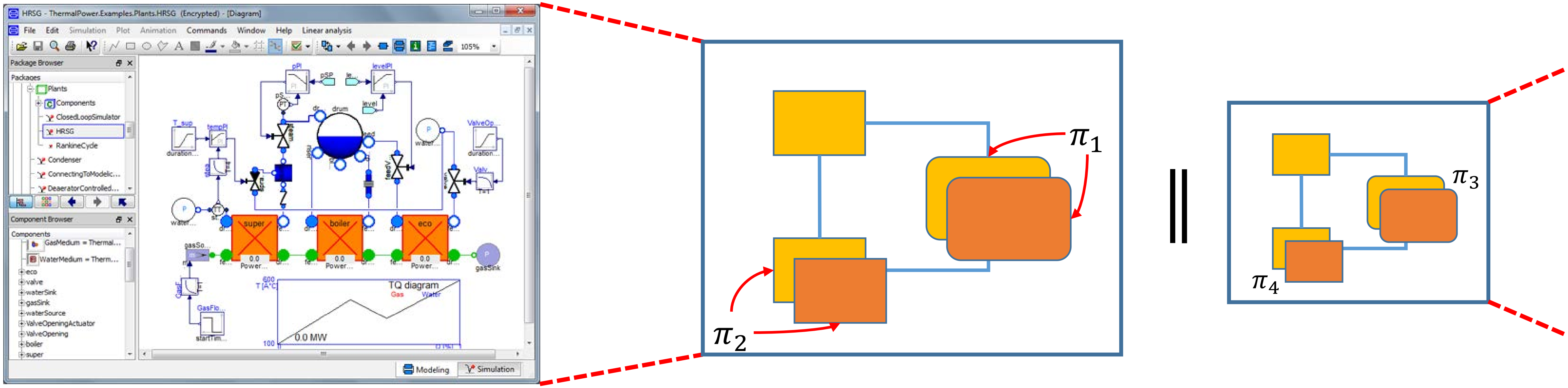
A Modelica model



A safety model with fault injection

Objectives:

- Semantic link maintained between detailed and abstract model
- Semantic link maintained with safety model



A Modelica model

A safety model with fault injection

Objectives:

- Semantic link maintained between detailed and abstract model
- Semantic link maintained with safety model
- Modularity

Existing approaches by statisticians and AI people

Pragmatic proposals by statisticians and AI people, on top of C++

- BUGS [Spiegelhalter 1994]: *a software package for Bayesian inference using Gibbs sampling. The software has been instrumental in raising awareness of Bayesian modelling among both academic and commercial communities internationally, and has enjoyed considerable success over its 20-year life span.* 2009
- Stan [Carpenter 2017]: *Stan is a probabilistic programming language for specifying statistical models. A Stan program imperatively defines a log probability function over parameters conditioned on specified data and constants. As of version 2.14.0, Stan provides full Bayesian inference for continuous-variable models through Markov chain Monte Carlo methods such as the No-U-Turn sampler, an adaptive form of Hamiltonian Monte Carlo sampling. Penalized maximum likelihood estimates are calculated using optimization methods such as the limited memory Broyden-Fletcher-Goldfarb-Shanno algorithm.*

Existing approaches [Katoen 2017]

1. skip	empty statement
2. abort	abortion
3. x := E	assignment
4. observe (G)	conditioning (value of G known)
5. prog1 ; prog2	sequential composition
6. if (G) prog1 else prog2	choice
7. prog1 [p] prog2	probabilistic choice
8. while (G) prog	iteration

- Statement 4 specifies conditional probabilities
- Statement 7 is low level and is not a parallel composition
- Support for modularity: statements 5,6,7

Existing approaches [Katoen 2017]

1. skip — empty statement
2. abort — abortion
3. x := E — assignment
4. observe (G) — conditioning (value of G known)
5. prog1 ; prog2 — sequential composition
6. if (G) prog1 else prog2 — choice
7. prog1 [p] prog2 — probabilistic choice
8. while (G) prog — iteration

- Mostly OO oriented, not for reactive systems
- No parallel composition
- Nicely layered: this is a specification language
- $\exists$ support for solving statistical problems. Not well layered.

Reactive Programming: ProbZelus [Baudart 2019]

$$d ::= \texttt{let node } f\ x = e \mid d\ d$$

$$e ::= c \mid x \mid (e, e) \mid op(e) \mid f(e) \mid \texttt{last } x \mid e \texttt{ where rec } E$$
$$\mid \texttt{present } e \rightarrow e \texttt{ else } e \mid \texttt{reset } e \texttt{ every } e$$
$$\mid \texttt{sample } e \mid \texttt{observe } e \mid \texttt{factor } e \mid \texttt{infer } e$$

$$E ::= () \mid x = e \mid \texttt{init } x = c \mid E \texttt{ and } E$$

- Zelus: a reactive (synchronous) language of streams of data
- Probabilistic extension:
  - `rec x = sample(gaussian(0->pre x, 1))` $x \sim N(pre(x), 1)$
  - `observe (gaussian(x, 1), y)` $y \sim N(x, 1)$
  - `factor(e)` $\Leftrightarrow$ `observe(Exp(1), -e)`
  - `infer(e)` estimates current proba distrib of e knowing the past

Reactive Programming: ProbZelus [Baudart 2019]

$$d ::= \texttt{let node } f \; x = e \mid d \; d$$

$$\begin{aligned} e ::= {} & c \mid x \mid (e, e) \mid op(e) \mid f(e) \mid \texttt{last}\, x \mid e \texttt{ where rec } E \\ & \mid \texttt{present } e \rightarrow e \texttt{ else } e \mid \texttt{reset } e \texttt{ every } e \\ & \mid \texttt{sample } e \mid \texttt{observe } e \mid \texttt{factor } e \mid \texttt{infer } e \end{aligned}$$

$$E ::= () \mid x = e \mid \texttt{init } x = c \mid E \texttt{ and } E$$

Some comments:

- Conservative extension of discrete time Zelus
- Interesting type system proba/nonproba
- Interesting causality analysis
- Layered????

I have a dream

```
system S1 and S2 and S3 where

system S1 (visible:u,y) =
    v = f(u)
and y = if fail then z
                else v
and fail ~ Bernoulli(10^-6)
and z ~ Gauss(v,sigma)
end

system S2 (visible:y) =
observe y
end

system S3 (visible:u) =
observe u>0
end
```

- Only the visible variables participate in interactions (define state space Q)
- Prior distributions are independent
- `observe y` specifies that the value of `y` is known; `observe u>0` states that this predicate is true
- We can first define `S1` and then indicate via `S2` where the sensors are and via `S3` the nondeterministic information we have on `u`

I have a dream

```
problem P1 in system S1 and S2 and S3 where
```

```
system S1 (visible:u,y,z,α) =
    v = f(u)
and y = if fail then z
                else v
and fail ∼ Bernoulli(α)
and z ∼ Gauss(v,sigma)
end

system S2 (visible:y) =
observe y
end

system S3 (visible:u) =
observe u>0
end
```

```
problem P1 (visible:u,z) =
estimate u,z
end
```

- `estimate` implemented by various algorithms

I have a dream

```
problem P2 in system S1 and S2 and S3 where
```

```
system S1 (visible:u,y,z,α) =
    v = f(u)
and y = if fail then z
                else v
and fail ∼ Bernoulli(α)
and z ∼ Gauss(v,sigma)
end

system S2 (visible:y) =
observe y
end

system S3 (visible:u) =
observe u>0
end
```

```
problem P2 (visible:α) =
estimate α
end
```

- `estimate` implemented by various algorithms

I have a dream

```
problem P1 and P3 in system S1 and S2 and S3 where
```

```
system S1 (visible:u,y,z,α) =
    v = f(u)
and y = if fail then z
                else v
and fail ∼ Bernoulli(α)
and z ∼ Gauss(v,sigma)
end

system S2 (visible:y) =
observe y
end

system S3 (visible:u) =
observe u>0
end
```

```
problem P1 (visible:u,z) =
estimate u,z
end

problem P3 (visible:α) =
test α > 0
end
```

- `test` implemented by various algorithms

What about semantics?

Basic issues in probabilistic paradigms

Requirements on Probabilistic Programming of React. Syst.

A use case related to safety analysis

Approaches

- Statisticians and AI people
- SW engineering style: Katoen 2017
- Reactive Programming: ProbZelus
- I have a dream

Probabilistic models

- Existing approaches (PA and variations)
- Alternative idea for the probabilistic parallel composition

Mixed (static) Systems (MS) [Benveniste et al.1995]

- Details

Mixed Markov Decision Processes (MMDP)

Link between MMDP and PA [Segala et al.]

Probabilistic Interface Theory (sketch)

Probabilistic models: existing approaches

- A good tutorial is [Sokolova and de Vink 2004]
- Objective: models that subsume both nondeterministic automata and Markov chains
- Ingredients: states q, actions α, and probabilities π (over states and possibly actions), with various mixes:

$q \xrightarrow{\alpha} \pi \rightsquigarrow q'$	Simple Segala PA	(Markov Decision Process)
$q \longrightarrow \pi \rightsquigarrow (\alpha, q')$	Segala PA	(Semi-Markov chain)

$\longrightarrow$: nondeterministic choice; $\rightsquigarrow$: probabilistic choice

Probabilistic models: existing approaches

$$q \xrightarrow{\alpha} \pi \rightsquigarrow q' \qquad \text{Simple Segala PA}$$
$$q \longrightarrow \pi \rightsquigarrow (\alpha, q') \qquad \text{Segala PA}$$

- The notions of simulation relation rely on extending relations between states to relations between probabilities on states:

$\rho \subseteq Q \times R$ extended to $\rho^{\mathcal{P}} \subseteq \mathcal{P}(Q) \times \mathcal{P}(R)$ defined by

$$(\pi_Q, \pi_R) \in \rho^{\mathcal{P}} \text{ iff } \exists \mu \in \mathcal{P}(Q \times R) : \begin{cases} \mu \text{ projects to } \pi_Q, \pi_R \\ \text{support}(\mu) = \rho \end{cases}$$

- (Bi)simulation theory works well.

Probabilistic models: existing approaches

$q \xrightarrow{\alpha} \pi \rightsquigarrow q'$	Simple Segala PA
$q \longrightarrow \pi \rightsquigarrow (\alpha, q')$	Segala PA

- Parallel composition is problematic
  (with the exception of simple Segala PA)

- The problem is the conflict between:
  - Performing probabilistic choice $\pi \rightsquigarrow$
  - Synchronizing on common actions

Probabilistic models: alternative idea for the $\|$

$q \xrightarrow{\alpha} \pi \rightsquigarrow q'$ Simple Segala PA keep as such

$q \longrightarrow \pi \rightsquigarrow (\alpha, q')$ Segala PA alternative

$$(q_1, q_2) \longrightarrow \pi_1 \otimes \pi_2 \rightsquigarrow ((\alpha_1, q_1'); (\alpha_2, q_2')) \stackrel{??}{\Longrightarrow} (\alpha_1 \alpha_2, (q_1', q_2'))$$

Idea: take "$\alpha_1 \alpha_2$ defined" as a constraint and use conditional probability distributions: $\pi_1 \otimes \pi_2$ given that $\alpha_1 \alpha_2$ is defined:

Probabilistic models: alternative idea for the $\|$

$q \xrightarrow{\alpha} \pi \rightsquigarrow q'$	Simple Segala PA	keep as such
$q \longrightarrow \pi \rightsquigarrow (\alpha, q')$	Segala PA	alternative

$$(q_1, q_2) \longrightarrow \pi_1 \otimes \pi_2 \rightsquigarrow ((\alpha_1, q_1'); (\alpha_2, q_2')) \stackrel{??}{\Longrightarrow} (\alpha_1\alpha_2, (q_1', q_2'))$$

Idea: take "$\alpha_1\alpha_2$ defined" as a constraint and use conditional probability distributions: $\pi_1 \otimes \pi_2$ given that $\alpha_1\alpha_2$ is defined:

$$(q_1, q_2) \longrightarrow \pi_1 \otimes \pi_2(. \mid \alpha_1\alpha_2 \text{ defined}) \quad \rightsquigarrow ((\alpha_1, q_1'); (\alpha_2, q_2')) \Longrightarrow (\alpha_1\alpha_2, (q_1', q_2'))$$

Requires $\pi_1 \otimes \pi_2(\alpha_1\alpha_2 \text{ defined}) > 0$, a consistency condition.

Thm: If $(\alpha_1, \alpha_2) \mapsto \alpha_1\alpha_2$ is commutative and associative, then, this parallel composition is commutative and associative

Probabilistic models: alternative idea for the $\|$

$q \xrightarrow{\alpha} \pi \rightsquigarrow q'$	Simple Segala PA	keep as such
$q \longrightarrow \pi \rightsquigarrow (\alpha, q')$	Segala PA	alternative

$$(q_1, q_2) \longrightarrow \pi_1 \otimes \pi_2 \rightsquigarrow ((\alpha_1, q_1'); (\alpha_2, q_2')) \stackrel{??}{\Longrightarrow} (\alpha_1\alpha_2, (q_1', q_2'))$$

Idea: take "$\alpha_1\alpha_2$ defined" as a constraint and use conditional probability distributions: $\pi_1 \otimes \pi_2$ given that $\alpha_1\alpha_2$ is defined:

$$(q_1, q_2) \longrightarrow \pi_1 \otimes \pi_2(. \mid \alpha_1\alpha_2 \text{ defined}) \rightsquigarrow ((\alpha_1, q_1'); (\alpha_2, q_2')) \Longrightarrow (\alpha_1\alpha_2, (q_1', q_2'))$$

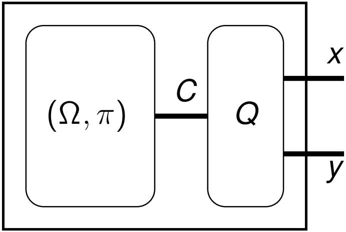
Requires $\pi_1 \otimes \pi_2(\alpha_1\alpha_2 \text{ defined}) > 0$, a consistency condition.

Thm: If $(\alpha_1, \alpha_2) \mapsto \alpha_1\alpha_2$ is commutative and associative, then, this parallel composition is commutative and associative

We can do much better

Mixed (static) Systems (MS) [Benveniste et al.1995]

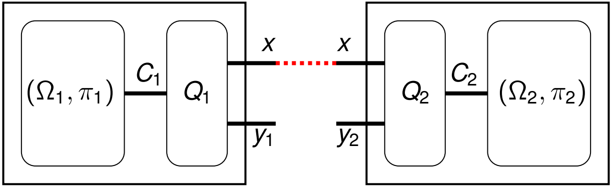
Intuition:

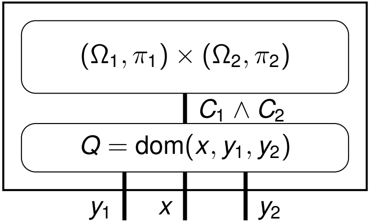


- (Ω, π): probability space, private
- Q: state space, $Q = \text{dom}(x, y)$, (x, y) visible variables
- $C(\omega; x, y) \subseteq \Omega \times Q$: a relation
- Operational semantics:
  1. draw ω at random according to conditional distrib $\pi(. \mid \exists q.C)$
  2. select q nondeterministically, such that $C(\omega, q)$ holds

Mixed (static) Systems (MS) [Benveniste et al.1995]

Parallel composition:





the parallel composition subsumes both

- the direct product of probability spaces
- the conjunction of systems of equations

Mixed (static) Systems (MS) [Benveniste et al.1995]

We assume an underlying set of variables $\mathcal{X}$; $\mathcal{Q} = \mathrm{Dom}(\mathcal{X})$

- $X \subseteq \mathcal{X}$ finite subset; $Q = \mathrm{Dom}(X)$; q generic element of Q
- $q_1 \sqcap q_2$ iff q_i agree on $X_1 \cap X_2$; $q_1 \sqcup q_2$ is the join of q_1 and q_2

Mixed (static) Systems (MS) [Benveniste et al.1995]

We assume an underlying set of variables $\mathcal{X}$; $\mathcal{Q} = \text{Dom}(\mathcal{X})$

- $X \subseteq \mathcal{X}$ finite subset; $Q = \text{Dom}(X)$; q generic element of Q
- $q_1 \sqcap q_2$ iff q_i agree on $X_1 \cap X_2$; $q_1 \sqcup q_2$ is the join of q_1 and q_2

Mixed Systems (MS) $S = ((\Omega, \pi), Q, C)$, where

$$
\begin{array}{rcl}
(\Omega, \pi) & : & \text{(finite or denumerable) probability space} \\
Q & : & \text{state space} \\
C & : & \text{relation } \subseteq \Omega \times Q \\
\text{semantics } S \rightsquigarrow q & : & \text{draw } \omega \text{ using } \pi(. \mid \exists q.C);\ \text{select } q \in C(\omega, .)
\end{array}
$$

$\pi(. \mid \exists q.C)$ is the conditional distribution π given that $\exists q.C$ holds.
Ensures that there will exist a q related to randomly generated ω.

Mixed (static) Systems (MS) [Benveniste et al.1995]

Mixed Systems (MS): $S = ((\Omega, \pi), Q, C)$, where

$$\begin{array}{rcl} (\Omega, \pi) & : & \text{probability space} \\ Q & : & \text{state space} \\ C & : & \text{relation } \subseteq \Omega \times Q \\ \text{semantics } S \rightsquigarrow q & : & \text{draw } \omega \text{ using } \pi(. \mid \exists q.C); \text{ select } q \in C(\omega, .) \end{array}$$

Compressing S

$\omega \sim \omega'$ iff $C(\omega, .) = C(\omega', .)$ (ω and ω' are indistinguishable via Q)

define $[S] = ((\Omega, \pi)/\sim, Q, C/\sim)$ compressed form

Thm: S and $[S]$ possess identical semantics.
The compressed form is a canonical form.

Mixed (static) Systems (MS) [Benveniste et al.1995]

Mixed Systems (MS): $S = ((\Omega, \pi), Q, C)$, where

$$\begin{array}{rcl} (\Omega,\pi) & : & \text{probability space} \\ Q & : & \text{state space} \\ C & : & \text{relation } \subseteq \Omega \times Q \\ \text{semantics } S \rightsquigarrow q & : & \text{draw } \omega \text{ using } \pi(. \mid \exists q.C); \text{ select } q \in C(\omega,.) \end{array}$$

Extending a relation ρ, from states to Mixed Systems

$\rho \subseteq Q \times R$ extended to $\rho^{\mathcal{S}} \subseteq \mathcal{S}(Q) \times \mathcal{S}(R)$ defined by

$$(S_Q, S_R) \in \rho^{\mathcal{S}} \text{ iff } \exists \mu \in \mathcal{P}(\Omega_Q \times \Omega_R) : \left\{ \begin{array}{l} \mu \text{ projects to } \pi_Q, \pi_R \\ \mu \text{ has support } (C,C)^{-1}(\rho) \end{array} \right.$$

Thm: Compression is a congruence wrt lifted relations:

$$(S_1, S_2) \in \rho^{\mathcal{S}} \text{ and } [S_1'] = [S_1] \text{ imply } (S_1', S_2) \in \rho^{\mathcal{S}}$$

Mixed (static) Systems (MS) [Benveniste et al.1995]

Mixed Systems (MS): $S = ((\Omega, \pi), Q, C)$, where

$$\begin{array}{rcl} (\Omega, \pi) & : & \text{probability space} \\ Q & : & \text{state space} \\ C & : & \text{relation } \subseteq \Omega \times Q \\ \text{semantics } S \rightsquigarrow q & : & \text{draw } \omega \text{ using } \pi(. \mid \exists q.C);\ \text{select } q \in C(\omega, .) \end{array}$$

$S_1 \parallel S_2 = ((\Omega, \pi), Q, C)$, where:

$$\begin{array}{rcl} (\Omega, \pi) & = & (\Omega_1 \times \Omega_2, \pi_1 \otimes \pi_2) \\ Q & = & Q_1 \sqcup Q_2 =_{\text{def}} \{q_1 \sqcup q_2 \mid q_i \in Q_i, q_1 \sqcap q_2\} \\ C & = & \{((\omega_1, \omega_2); q_1 \sqcup q_2) \mid (\omega_i, q_i) \in C_i, q_1 \sqcap q_2\} \end{array}$$

Thm: $\parallel$ is associative and commutative, and $[S_1 \parallel S_2] = [[S_1] \parallel [S_2]]$

Mixed Markov Decision Processes (MMDP)

Idea: use Mixed Systems as targets of transitions

from $q \xrightarrow{\alpha} q'$ to $q \xrightarrow{\alpha} \pi$ and to $q \xrightarrow{\alpha} S$

Mixed Markov Decision Processes (MMDP)

- q: state
- α: action
- S: Mixed System

Mixed Markov Decision Processes (MMDP)

- MMDP $M = (A, Q, \rightarrow)$, where

$$q \xrightarrow{\alpha} S = ((\Omega, \pi), Q, C) \rightsquigarrow q'$$

$$\alpha\alpha' \text{ defined iff } \alpha = \alpha' \text{ and then } \alpha\alpha' =_{\text{def}} \alpha$$

- Simulation relation $q_1 \leq q_2$

$$\forall \alpha : \; q_1 \xrightarrow{\alpha} S_1 \quad \Longrightarrow \quad \exists S_2 : \; q_2 \xrightarrow{\alpha} S_2 \text{ and } S_1 \leq^{\mathcal{S}} S_2$$

  and define $M_1 \leq M_2$ iff $q_{0,1} \leq q_{0,2}$.

- $M_1 \parallel M_2$

$$q_1 \sqcup q_2 \xrightarrow{\alpha} S_1 \parallel S_2 \rightsquigarrow q_1' \sqcup q_2'$$

Thm: $\parallel$ is associative, commutative, and monotonic:
$M_i' \leq M_i$ implies $M_1' \parallel M_2' \leq M_1 \parallel M_2$

Mixed Markov Decision Processes (MMDP)

```
S1 and S2 and S3 where

system S1 (visible:u,y) =
    v = f(u) + pre y
and y = if fail then z
                else v
and fail ∼ Bernoulli(10^-6)
and z ∼ Gauss(v,sigma)
end

system S2 (visible:y) =
observe y
end

system S3 (visible:u) =
observe u>0
end
```

- Conservative extension of Lustre with observers
- Only the visible variables participate in interactions (define state space Q)
- Prior distributions are independent (in space and time)
- `observe y` specifies that the value of `y` is known; `observe u>0` states that this predicate is true

Link between MMDP and PA [Segala et al.]

- Simple PA (SPA):
  - There exists an embedding SPA $\mapsto$ MMDP preserving both simulation and parallel composition
  - There exists an embedding MMDP $\mapsto$ SPA preserving simulation. Parallel composition cannot be preserved.
- PA:
  - There exists an embedding PA $\mapsto$ MMDP preserving simulation. Parallel composition not preserved.
  - There exists an embedding MMDP $\mapsto$ PA preserving simulation. Parallel composition cannot be preserved.
- Sokolova & de Vink Most General Model can be mapped to MMDP by preserving simulation (but not parallel composition).

The interest of MMDP is its unique clean parallel composition and support for conditioning.

Mixed Interfaces (sketch)

The following is not developed in this presentation (a Fossacs rejection):

- We have developed an interface theory on top of MMDP, called Mixed Interfaces
- It offers all the algebra of contracts (except for the quotient) $\Longrightarrow$ support for multi-view and concurrent system design
- Would make sense to formulate a logic for Mixed Interfaces as a specification formalism for probabilistic programming

mixed feelings
about
mixed systems?