Markov chains and Markov decision processes in Isabelle/HOL

Johannes Hölzl

January 2016

TU München, Germany

Introduction

Formalize probabilistic models:

- Discrete *infinite* state spaces

Formalize probabilistic models:

- Discrete *infinite* state spaces
- Trace space & transition system

Formalize probabilistic models:

- Discrete *infinite* state spaces
- Trace space & transition system
- Support non-determinism

Formalize probabilistic models:

- Discrete *infinite* state spaces
- Trace space & transition system
- Support non-determinism
- Compare different system types

Formalize probabilistic models:

- Discrete *infinite* state spaces
- Trace space & transition system
- Support non-determinism
- Compare different system types

Approach:

- *Coalgebraic view* on transition systems

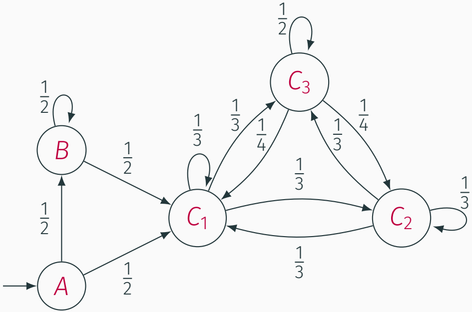
Formalize probabilistic models:

- Discrete *infinite* state spaces
- Trace space & transition system
- Support non-determinism
- Compare different system types

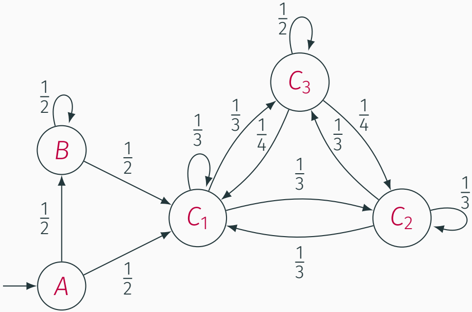
Approach:

- *Coalgebraic view* on transition systems
- *Fixed points* to define queries on trace space

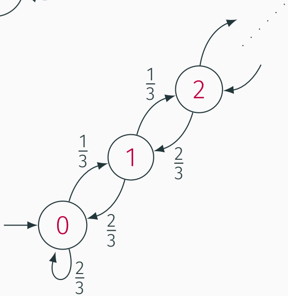
Markov chains



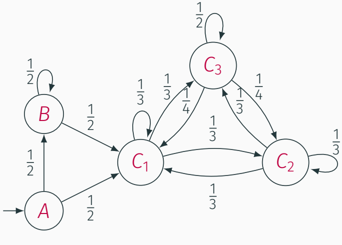
(a) A finite process



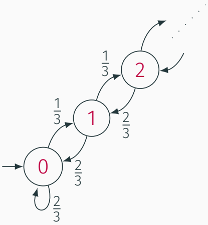
(a) A finite process



(b) An infinite birth-death process

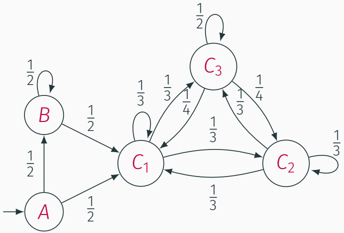


(a) A finite process

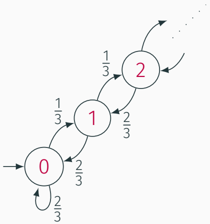


(b) An infinite birth-death process

- $\Pr_A(\Diamond C_3) = ?$

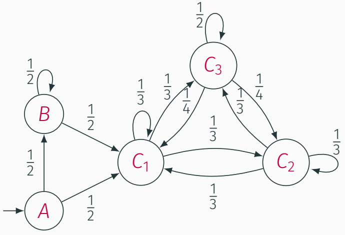


(a) A finite process

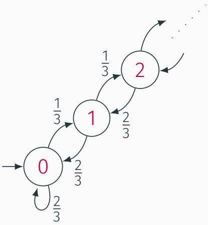


(b) An infinite birth-death process

- $\Pr_A(\Diamond C_3) = ?$
- $\Pr_A(\Box \neg C_3) = ?$

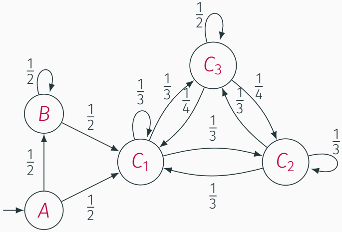


(a) A finite process

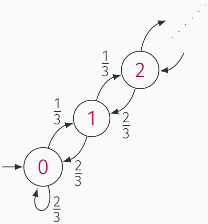


(b) An infinite birth-death process

- $\Pr_A(\Diamond C_3) = ?$
- $\Pr_A(\Box \neg C_3) = ?$
- $\Pr_A(\Box\Diamond\{C_1, C_2, C_3\}) = ?$

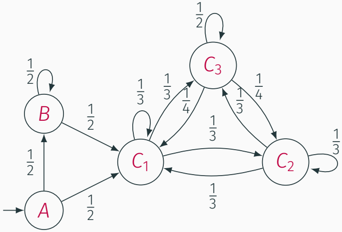


(a) A finite process

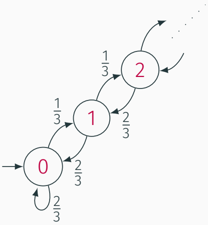


(b) An infinite birth-death process

- $\Pr_A(\Diamond C_3) = ?$
- $\Pr_A(\Box \neg C_3) = ?$
- $\Pr_A(\Box\Diamond\{C_1, C_2, C_3\}) = ?$
- $\lim_{n\to\infty} \Pr_A(\omega_n = C_3) > \lim_{n\to\infty} \Pr_A(\omega_n = C_2)$?



(a) A finite process



(b) An infinite birth-death process

- $\Pr_A(\Diamond C_3) = ?$
- $\Pr_A(\Box \neg C_3) = ?$
- $\Pr_A(\Box\Diamond\{C_1, C_2, C_3\}) = ?$
- $\lim_{n\to\infty} \Pr_A(\omega_n = C_3) > \lim_{n\to\infty} \Pr_A(\omega_n = C_2)$?
- $\int_\omega f_0\ \omega\ d\mathcal{T}_0 = ?$ – f: first occurence

Markov Chains — A Coalgebraic View

How to represent Markov chains?

Markov Chains — A Coalgebraic View

How to represent Markov chains?

$$\boxed{K :: \sigma \Rightarrow \sigma\ pmf}$$

Markov Chains — A Coalgebraic View

How to represent Markov chains?

$$\boxed{K :: \sigma \Rightarrow \sigma\ pmf}$$

K – *Markov kernel*: the transitions for each state

σ – type of states

$\sigma\ pmf$ – probability mass functions (discrete distributions)

Probability Mass Function

Model probabilistic transitions!

$$\mu :: \sigma\ \mathrm{pmf} \quad \approx \quad \mu :: \sigma \Rightarrow [0,1], \quad \textstyle\sum_x \mu\, x = 1$$

$$\approx \quad \mu :: \sigma\ \mathrm{measure}, \quad \mu\, \mathcal{U} = 1, \quad \textit{discrete}$$

Similar to Audebaud & Paulin-Mohring [MPC 2006]

Probability Mass Function

Model probabilistic transitions!

$$\mu :: \sigma \text{ pmf} \quad \approx \quad \mu :: \sigma \Rightarrow [0,1], \quad \textstyle\sum_x \mu\, x = 1$$
$$\approx \quad \mu :: \sigma \text{ measure}, \quad \mu\, \mathcal{U} = 1, \quad \textit{discrete}$$

Similar to Audebaud & Paulin-Mohring [MPC 2006]

- map $f\,\mu = \lambda x.\ \sum_{f\,y = x} \mu\, y$
- set $\mu = \{x \mid \mu\, x \neq 0\}$
- $\mu \ggg \nu = \lambda x.\ \sum_{y} \mu\, y \cdot \nu_y\, x$
- return $x = \lambda x'.\ \begin{cases} 1 & \text{if } x = x' \\ 0 & \text{else} \end{cases}$

- Bernoulli
- Uniform
- Binomial
- Geometric
- Poisson
- Conditional

Trace Space

Necessary to define $\int_{\omega} d\mathcal{T}_s$ (and $\Pr_s(P\ \omega)$)

Necessary to define $\int_{\omega} d\mathcal{T}_s$ (and $\Pr_s(P\ \omega)$)

$$\texttt{codatatype}\ \sigma\ \mathit{stream} = \sigma \cdot (\sigma\ \mathit{stream}) \qquad \approx \mathbb{N} \Rightarrow \sigma$$

Necessary to define $\int_{\omega} d\mathcal{T}_s$ (and $\Pr_s(P\ \omega)$)

$$\texttt{codatatype}\ \sigma\ \mathit{stream} = \sigma \cdot (\sigma\ \mathit{stream}) \qquad \approx \mathbb{N} \Rightarrow \sigma$$

Given K construct $\mathcal{T} :: \sigma \Rightarrow \sigma\ \text{stream measure}$ where:

$$\mathcal{T}_s = \text{do}\ \{t \leftarrow K_s\ ;\ \omega \leftarrow \mathcal{T}_t\ ;\ \text{return}\ (t \cdot \omega)\}$$

Equivalently: for f Borel-measurable:

$$\int_{\omega} f(\omega)\ d\mathcal{T}_s = \int_t \left(\int_{\omega} f(t \cdot \omega)\ d\mathcal{T}_t \right)\ dK_s$$

Necessary to define $\int_\omega d\mathcal{T}_s$ (and $\Pr_s(P\ \omega)$)

$$\texttt{codatatype}\ \sigma\ \mathit{stream} = \sigma \cdot (\sigma\ \mathit{stream}) \qquad \approx \mathbb{N} \Rightarrow \sigma$$

Given K construct $\mathcal{T} :: \sigma \Rightarrow \sigma$ stream measure where:

$$\mathcal{T}_s = \text{do}\ \{t \leftarrow K_s\ ;\ \omega \leftarrow \mathcal{T}_t\ ;\ \text{return}\ (t \cdot \omega)\}$$

Equivalently: for f Borel-measurable:

$$\int_\omega f(\omega)\ d\mathcal{T}_s = \int_t \left(\int_\omega f(t \cdot \omega)\ d\mathcal{T}_t \right)\ dK_s$$

This construction is unique!

Construct trace space for Markov chains

- Traditional solution: use Caratheodory's extension theorem

Construct trace space for Markov chains

- Traditional solution: use Caratheodory's extension theorem
  - Generate trace space by $\{\omega \mid \omega \textit{ starts with } xs\}$

Construct trace space for Markov chains

- Traditional solution: use Caratheodory's extension theorem
  - Generate trace space by $\{\omega \mid \omega \textit{ starts with } xs\}$
  - Countable additivity of pre-measure on cylinder sets

Construct trace space for Markov chains

- Traditional solution: use Caratheodory's extension theorem
  - Generate trace space by $\{\omega \mid \omega \textit{ starts with } xs\}$
  - Countable additivity of pre-measure on cylinder sets
- Our solution: reuse infinite product of probability spaces
  Measure space of *decisions* D:

$$D = \prod_{n::\mathbb{N}} \prod_{s::\sigma} K_s$$

$$run(s, d{\cdot}X) = s{\cdot}run(d\ s, X)$$

$$\mathcal{T}_s = D(run(s, \cdot))$$

Construct trace space for Markov chains

- Traditional solution: use Caratheodory's extension theorem
  - Generate trace space by $\{\omega \mid \omega \textit{ starts with } xs\}$
  - Countable additivity of pre-measure on cylinder sets
- Our solution: reuse infinite product of probability spaces
  Measure space of *decisions* D:

$$D = \prod_{n::\mathbb{N}} \prod_{s::\sigma} K_s$$

$$run(s, d{\cdot}X) = s{\cdot}run(d\ s, X)$$

$$\mathcal{T}_s = D(run(s, \cdot))$$

- Future Solution: Theorem by *Ionescu-Tuclea*

Queries on the trace space

Eventually ϕ: $\Diamond_{\phi}\ \omega =$

Always ϕ: $\Box_{\phi}\ \omega =$

ψ Until ϕ: ${}_{\psi}U_{\phi}\ \omega =$

First hit ϕ: $f_{\phi}\ \omega =$

Counting ϕ: $c_{\phi}\ \omega =$

Queries on the trace space

Eventually ϕ:	$\Diamond_\phi\ \omega$	$=$	$\exists n.\ \phi\ \omega_n$
Always ϕ:	$\Box_\phi\ \omega$	$=$	
ψ Until ϕ:	${}_\psi U_\phi\ \omega$	$=$	
First hit ϕ:	$f_\phi\ \omega$	$=$	
Counting ϕ:	$c_\phi\ \omega$	$=$	

Queries on the trace space

Eventually ϕ:	$\Diamond_\phi\ \omega$	$=$	$\exists n.\ \phi\ \omega_n$
Always ϕ:	$\Box_\phi\ \omega$	$=$	$\forall n.\ \phi\ \omega_n$
ψ Until ϕ:	${}_\psi U_\phi\ \omega$	$=$	
First hit ϕ:	$f_\phi\ \omega$	$=$	
Counting ϕ:	$c_\phi\ \omega$	$=$	

Queries on the trace space

Eventually ϕ:	$\Diamond_{\phi}\ \omega$	$=$	$\exists n.\ \phi\ \omega_n$
Always ϕ:	$\Box_{\phi}\ \omega$	$=$	$\forall n.\ \phi\ \omega_n$
ψ Until ϕ:	${}_{\psi}U_{\phi}\ \omega$	$=$	$\exists N.\ (\forall n < N.\ \psi\ \omega_n) \wedge \phi\ \omega_N$
First hit ϕ:	$f_{\phi}\ \omega$	$=$	
Counting ϕ:	$c_{\phi}\ \omega$	$=$	

Queries on the trace space

Eventually ϕ:	$\Diamond_{\phi}\ \omega$	$=$	$\exists n.\ \phi\ \omega_n$
Always ϕ:	$\Box_{\phi}\ \omega$	$=$	$\forall n.\ \phi\ \omega_n$
ψ Until ϕ:	${}_{\psi}U_{\phi}\ \omega$	$=$	$\exists N.\ (\forall n < N.\ \psi\ \omega_n) \wedge \phi\ \omega_N$
First hit ϕ:	$f_{\phi}\ \omega$	$=$	$\text{LEAST}\ n.\ \phi\ \omega_n$
Counting ϕ:	$c_{\phi}\ \omega$	$=$	

Queries on the trace space

Eventually ϕ:	$\Diamond_\phi\ \omega$	$=$	$\exists n.\ \phi\ \omega_n$
Always ϕ:	$\Box_\phi\ \omega$	$=$	$\forall n.\ \phi\ \omega_n$
ψ Until ϕ:	${}_\psi U_\phi\ \omega$	$=$	$\exists N.\ (\forall n < N.\ \psi\ \omega_n) \land \phi\ \omega_N$
First hit ϕ:	$f_\phi\ \omega$	$=$	$\text{LEAST}\ n.\ \phi\ \omega_n$
Counting ϕ:	$c_\phi\ \omega$	$=$	$\sum_n \lfloor \phi\ \omega_n \rfloor$

Queries on the trace space

Eventually ϕ:	$\Diamond_\phi\ \omega$	$=$	$\exists n.\ \phi\ \omega_n$
		$\overset{\text{lfp}}{=}$	$\phi\ \omega\ \lor\ \Diamond_\phi\ (\text{tl}\ \omega)$
Always ϕ:	$\Box_\phi\ \omega$	$=$	$\forall n.\ \phi\ \omega_n$
ψ Until ϕ:	${}_\psi U_\phi\ \omega$	$=$	$\exists N.\ (\forall n < N.\ \psi\ \omega_n) \land \phi\ \omega_N$
First hit ϕ:	$f_\phi\ \omega$	$=$	$\text{LEAST}\ n.\ \phi\ \omega_n$
Counting ϕ:	$c_\phi\ \omega$	$=$	$\sum_n \lfloor \phi\ \omega_n \rfloor$

Queries on the trace space

Eventually ϕ:	$\Diamond_\phi\ \omega$	$=$	$\exists n.\ \phi\ \omega_n$
		$\overset{\text{lfp}}{=}$	$\phi\ \omega\ \ \vee\ \ \Diamond_\phi\ (\text{tl}\ \omega)$
Always ϕ:	$\Box_\phi\ \omega$	$=$	$\forall n.\ \phi\ \omega_n$
		$\overset{\text{gfp}}{=}$	$\phi\ \omega\ \ \wedge\ \ \Box_\phi\ (\text{tl}\ \omega)$
ψ Until ϕ:	${}_\psi U_\phi\ \omega$	$=$	$\exists N.\ (\forall n < N.\ \psi\ \omega_n) \wedge \phi\ \omega_N$
First hit ϕ:	$f_\phi\ \omega$	$=$	$\text{LEAST}\ n.\ \phi\ \omega_n$
Counting ϕ:	$c_\phi\ \omega$	$=$	$\sum_n \lfloor \phi\ \omega_n \rfloor$

Queries on the trace space

Eventually ϕ:	$\Diamond_\phi\ \omega$	$=$	$\exists n.\ \phi\ \omega_n$
		$\stackrel{\text{lfp}}{=}$	$\phi\ \omega\ \lor\ \Diamond_\phi\ (\mathrm{tl}\ \omega)$
Always ϕ:	$\Box_\phi\ \omega$	$=$	$\forall n.\ \phi\ \omega_n$
		$\stackrel{\text{gfp}}{=}$	$\phi\ \omega\ \land\ \Box_\phi\ (\mathrm{tl}\ \omega)$
ψ Until ϕ:	${}_\psi U_\phi\ \omega$	$=$	$\exists N.\ (\forall n < N.\ \psi\ \omega_n) \land \phi\ \omega_N$
		$\stackrel{\text{lfp}}{=}$	$\left(\psi\ \omega\ \land\ {}_\psi U_\phi\ (\mathrm{tl}\ \omega)\right)\ \lor\ \phi\ \omega$
First hit ϕ:	$f_\phi\ \omega$	$=$	$\mathrm{LEAST}\ n.\ \phi\ \omega_n$
Counting ϕ:	$c_\phi\ \omega$	$=$	$\sum_n \lfloor \phi\ \omega_n \rfloor$

Queries on the trace space

Eventually ϕ:	$\Diamond_\phi\ \omega$	$=$	$\exists n.\ \phi\ \omega_n$
		$\overset{\text{lfp}}{=}$	$\phi\ \omega \ \vee\ \Diamond_\phi\ (\mathrm{tl}\ \omega)$
Always ϕ:	$\Box_\phi\ \omega$	$=$	$\forall n.\ \phi\ \omega_n$
		$\overset{\text{gfp}}{=}$	$\phi\ \omega \ \wedge\ \Box_\phi\ (\mathrm{tl}\ \omega)$
ψ Until ϕ:	${}_\psi U_\phi\ \omega$	$=$	$\exists N.\ (\forall n < N.\ \psi\ \omega_n) \wedge \phi\ \omega_N$
		$\overset{\text{lfp}}{=}$	$\left(\psi\ \omega \ \wedge\ {}_\psi U_\phi\ (\mathrm{tl}\ \omega)\right) \ \vee\ \phi\ \omega$
First hit ϕ:	$f_\phi\ \omega$	$=$	$\text{LEAST } n.\ \phi\ \omega_n$
		$\overset{\text{lfp}}{=}$	$\begin{cases} 1 + f_\phi\ (\mathrm{tl}\ \omega) & \text{if } \neg\ \phi\ \omega \\ 0 & \text{otherwise} \end{cases}$
Counting ϕ:	$c_\phi\ \omega$	$=$	$\sum_n \lfloor \phi\ \omega_n \rfloor$

Queries on the trace space

Eventually ϕ:	$\Diamond_\phi\ \omega$	$=$	$\exists n.\ \phi\ \omega_n$
		$\stackrel{\text{lfp}}{=}$	$\phi\ \omega\ \vee\ \Diamond_\phi\ (\text{tl}\ \omega)$
Always ϕ:	$\Box_\phi\ \omega$	$=$	$\forall n.\ \phi\ \omega_n$
		$\stackrel{\text{gfp}}{=}$	$\phi\ \omega\ \wedge\ \Box_\phi\ (\text{tl}\ \omega)$
ψ Until ϕ:	${}_\psi U_\phi\ \omega$	$=$	$\exists N.\ (\forall n < N.\ \psi\ \omega_n) \wedge \phi\ \omega_N$
		$\stackrel{\text{lfp}}{=}$	$\left(\psi\ \omega\ \wedge\ {}_\psi U_\phi\ (\text{tl}\ \omega)\right)\ \vee\ \phi\ \omega$
First hit ϕ:	$f_\phi\ \omega$	$=$	$\text{LEAST } n.\ \phi\ \omega_n$
		$\stackrel{\text{lfp}}{=}$	$\begin{cases} 1 + f_\phi\ (\text{tl}\ \omega) & \text{if } \neg\ \phi\ \omega \\ 0 & \text{otherwise} \end{cases}$
Counting ϕ:	$c_\phi\ \omega$	$=$	$\sum_n \lfloor \phi\ \omega_n \rfloor$
		$\stackrel{\text{lfp}}{=}$	$\begin{cases} 1 + c_\phi\ (\text{tl}\ \omega) & \text{if } \phi\ \omega \\ c_\phi\ (\text{tl}\ \omega) & \text{otherwise} \end{cases}$

Interception: Least/Greatest Fixed Points

Monotone functions f, g

Least fixed point: $\mathrm{lfp}\, f = f\,(\mathrm{lfp}\, f) \qquad (\forall x.\; f\,x \leqslant x \implies \mathrm{lfp}\, f \leqslant x)$

$$\mathrm{lfp}\, f = \overbrace{f \circ f \circ f \circ f \circ f \circ f \circ f \circ f \circ f \circ \cdots \circ}^{\infty} \bot$$

Interception: Least/Greatest Fixed Points

Monotone functions f, g

Least fixed point: $\mathrm{lfp}\ f = f\ (\mathrm{lfp}\ f) \qquad (\forall x.\ f\ x \leqslant x \implies \mathrm{lfp}\ f \leqslant x)$

Rolling rule: $g\ (\mathrm{lfp}\ (f \circ g)) = \mathrm{lfp}\ (g \circ f)$

$$g \circ \overbrace{f \circ g} \circ \overbrace{f \circ g} \circ \overbrace{f \circ g} \circ \overbrace{f \circ g} \circ \ldots$$

Interception: Least/Greatest Fixed Points

Monotone functions f, g

Least fixed point: $\text{lfp}\ f = f\ (\text{lfp}\ f) \qquad (\forall x.\ f\ x \leqslant x \implies \text{lfp}\ f \leqslant x)$

Rolling rule: $g\ (\text{lfp}\ (f \circ g)) = \text{lfp}\ (g \circ f)$

$$\overbrace{g \circ f} \circ \overbrace{g \circ f} \circ \overbrace{g \circ f} \circ \overbrace{g \circ f} \circ g \circ \ldots$$

Interception: Least/Greatest Fixed Points

Monotone functions f, g

Least fixed point: $\text{lfp}\ f = f\ (\text{lfp}\ f) \qquad (\forall x.\ f\ x \leqslant x \implies \text{lfp}\ f \leqslant x)$

Rolling rule: $g\ (\text{lfp}\ (f \circ g)) = \text{lfp}\ (g \circ f)$

Iteration rule: $\text{lfp}\ (f \circ f) = \text{lfp}\ f$

Interception: Least/Greatest Fixed Points

Monotone functions f, g

Least fixed point: $\mathrm{lfp}\ f = f\ (\mathrm{lfp}\ f) \qquad (\forall x.\ f\ x \leqslant x \implies \mathrm{lfp}\ f \leqslant x)$

Rolling rule: $g\ (\mathrm{lfp}\ (f \circ g)) = \mathrm{lfp}\ (g \circ f)$

Iteration rule: $\mathrm{lfp}\ (f \circ f) = \mathrm{lfp}\ f$

Nesting rule: $\mathrm{lfp}\ (\lambda x.\ \mathrm{lfp}\ (f\ x)) = \mathrm{lfp}\ (\lambda x.\ f\ x\ x)$

Interception: Least/Greatest Fixed Points

$\forall C \in \mathbb{N} \to X.\ \text{monotone}\ C \implies f\,(\bigsqcup_i C_i) = \bigsqcup_i f\, C_i$

Least fixed point: $\mathsf{lfp}\ f = f\ (\mathsf{lfp}\ f) \qquad (\forall x.\ f\ x \leqslant x \implies \mathsf{lfp}\ f \leqslant x)$

Rolling rule: $g\ (\mathsf{lfp}\ (f \circ g)) = \mathsf{lfp}\ (g \circ f)$

Iteration rule: $\mathsf{lfp}\ (f \circ f) = \mathsf{lfp}\ f$

Nesting rule: $\mathsf{lfp}\ (\lambda x.\ \mathsf{lfp}\ (f\ x)) = \mathsf{lfp}\ (\lambda x.\ f\ x\ x)$

Transfer rule:

$$\sqcup\text{-continuous}\ \alpha, f, g$$

$$\frac{\alpha\ \bot = \bot \qquad \alpha \circ f = g \circ \alpha}{\alpha(\mathsf{lfp}\ f) = \mathsf{lfp}\ g}$$

$$\begin{aligned} \alpha(\mathsf{lfp}\ f) &= \alpha \circ f \circ f \circ f \circ f \circ f \circ \cdots \circ \bot \\ &= g \circ g \circ g \circ \alpha \circ f \circ f \circ \cdots \circ \bot \\ &= g \circ g \circ g \circ g \circ g \circ \cdots \circ \alpha\ \bot \\ &= \mathsf{lfp}\ g \end{aligned}$$

Interception: Least/Greatest Fixed Points

Monotone functions f, g

Least fixed point: $\mathrm{lfp}\ f = f\ (\mathrm{lfp}\ f) \qquad (\forall x.\ f\ x \leqslant x \implies \mathrm{lfp}\ f \leqslant x)$

Rolling rule: $g\ (\mathrm{lfp}\ (f \circ g)) = \mathrm{lfp}\ (g \circ f)$

Iteration rule: $\mathrm{lfp}\ (f \circ f) = \mathrm{lfp}\ f$

Nesting rule: $\mathrm{lfp}\ (\lambda x.\ \mathrm{lfp}\ (f\ x)) = \mathrm{lfp}\ (\lambda x.\ f\ x\ x)$

$\sqcup$−continuous α, f, g

Transfer rule: $$\frac{\alpha\ \bot = \bot \qquad \alpha \circ f = g \circ \alpha}{\alpha(\mathrm{lfp}\ f) = \mathrm{lfp}\ g}$$

Interception: Least/Greatest Fixed Points

Monotone functions f, g

Least fixed point: $\text{lfp}\ f = f\ (\text{lfp}\ f) \qquad (\forall x.\ f\ x \leqslant x \implies \text{lfp}\ f \leqslant x)$

Rolling rule: $g\ (\text{lfp}\ (f \circ g)) = \text{lfp}\ (g \circ f)$

Iteration rule: $\text{lfp}\ (f \circ f) = \text{lfp}\ f$

Nesting rule: $\text{lfp}\ (\lambda x.\ \text{lfp}\ (f\ x)) = \text{lfp}\ (\lambda x.\ f\ x\ x)$

$\sqcup$−continuous α, f, g

Transfer rule: $$\frac{\alpha\ \bot = \bot \qquad \alpha \circ f = g \circ \alpha}{\alpha(\text{lfp}\ f) = \text{lfp}\ g}$$

$\alpha\ f = \int f\ d\mathcal{M}$ for f Borel-measurable

Equation for queries under integration

Example (First hitting time ϕ on states)

Define f:

$$f_{\phi} \stackrel{\text{def}}{=} \text{lfp}\ (\lambda\, f\, (s \cdot \omega).\ \lfloor \neg \phi\ s \rfloor \cdot (1 + f\ \omega))$$

Equation for queries under integration

Example (First hitting time ϕ on states)

Define f:

$$f_{\phi} \stackrel{\text{def}}{=} \mathrm{lfp}\ (\lambda\, f\ (s{\cdot}\omega).\ \lfloor \neg\phi\ s \rfloor \cdot (1 + f\ \omega))$$

$$f_{\phi}\ (s{\cdot}\omega) \stackrel{\text{lfp}}{=} \begin{cases} 1 + f_{\phi}\ \omega & \text{if } \neg\ \phi\ s \\ 0 & \text{otherwise} \end{cases}$$

Equation for queries under integration

Example (First hitting time ϕ on states)

Define f:

$$f_\phi \stackrel{\text{def}}{=} \text{lfp } (\lambda\, f\ (s{\cdot}\omega).\ \lfloor \neg\phi\ s \rfloor \cdot (1 + f\ \omega))$$

$$f_\phi\ (s{\cdot}\omega) \stackrel{\text{lfp}}{=} \begin{cases} 1 + f_\phi\ \omega & \text{if } \neg\ \phi\ s \\ 0 & \text{otherwise} \end{cases}$$

Prove computation rule by transfer rule:

$$\int_\omega f_\phi\ \omega\ d\mathcal{T}_s = lfp\ \left(\lambda g\ s.\ \int_t \lfloor \neg\phi\ t \rfloor \cdot (1 + g\ t)\ \ dK_s\right)\ s$$

Equation for queries under integration

Example (First hitting time ϕ on states)

Define f:

$$f_\phi \stackrel{\text{def}}{=} \text{lfp}\ (\lambda\, f\ (s \cdot \omega).\ \lfloor \neg\phi\ s \rfloor \cdot (1 + f\ \omega))$$

$$f_\phi\ (s \cdot \omega) \stackrel{\text{lfp}}{=} \begin{cases} 1 + f_\phi\ \omega & \text{if } \neg\ \phi\ s \\ 0 & \text{otherwise} \end{cases}$$

Prove computation rule by transfer rule:

$$\int_\omega f_\phi\ \omega\ d\mathcal{T}_s = lfp\ \left(\lambda g\ s.\ \int_t \lfloor \neg\phi\ t \rfloor \cdot (1 + g\ t)\ dK_s\right)\ s$$

For finite state space: lfp is a system of linear equations!

Proofs employing fixed point reasoning

Lemma (Fairness)

$$\Pr_s\Big(\Box\Diamond t \implies \Box\Diamond(t \wedge \bigcirc t')\Big) = 1 \qquad \textit{if } t' \in K_t$$

Proof.

Show that gfp $(\lambda g\ s.\ (\neg t)\ U\ (t \cdot \neg t' \cdot g))$ has probability 0. ☐

Proofs employing fixed point reasoning

Lemma (Fairness)

$$\Pr_s\Big(\Box\Diamond t \implies \Box\Diamond(t \wedge \bigcirc t')\Big) = 1 \qquad \textit{if } t' \in K_t$$

Proof.

Show that gfp $(\lambda g\ s.\ (\neg t)\ U\ (t \cdot \neg t' \cdot g))$ has probability 0. □

Lemma (Finite hitting time)

$$\int_\omega f_t\ \omega\ d\mathcal{T}_s < \infty \qquad \textit{if } \Pr_s(\Diamond t) = 1 \textit{ and finite state space}$$

Proofs employing fixed point reasoning

Lemma (Fairness)

$$\Pr_s\Big(\Box\Diamond t \implies \Box\Diamond(t \wedge \bigcirc t')\Big) = 1 \qquad \textit{if } t' \in K_t$$

Proof.

Show that gfp $(\lambda g\ s.\ (\neg t)\ U\ (t \cdot \neg t' \cdot g))$ has probability 0. □

Lemma (Finite hitting time)

$$\int_\omega f_t\ \omega\ d\mathcal{T}_s < \infty \qquad \textit{if } \Pr_s(\Diamond t) = 1 \textit{ and finite state space}$$

Proof size is reduced to $\approx 65\%$!

Stationary Distribution

N is a *stationary distribution* iff $(N \gg= K) = N$

Or: $K \times N = N$ — K as transition matrix

- When support set of N is essential (bottom SCC):

$$\int_{\omega} f_s\ \omega\ d\mathfrak{T}_s = \frac{1}{N\ s} - 1$$

- When essential and aperiodic:

$$\lim_{n \to \infty} \Pr_s(\omega_n = t) = N\ t$$

- Stationary distribution for b): $N = \mathit{geometric}\left(\frac{1}{2}\right)$

$$\text{DTMC}\ (M :: \alpha\ \text{measure})\ (X :: \text{nat} \Rightarrow \alpha \Rightarrow \sigma) =$$
$$\quad \text{prob-space}\ M \quad\wedge\quad (\forall n.\ X\ n \in M \to^{\sigma} \mathcal{U}) \quad\wedge$$
$$\quad (\exists S.\ \text{countable}\ S \wedge \forall n.\ \Pr(X\ n \in S) = 1) \quad\wedge$$

— **The stochastic process** X **is** ***memoryless*****:**

$$\Big(\forall n\ s\ t.$$
$$\quad \Pr(\forall n' \leqslant n.\ X\ n' = t\ n') \neq 0 \longrightarrow$$
$$\quad \Pr(X\ (n+1) = s \mid \forall n' \leqslant n.\ X\ n' = t\ n') =$$
$$\quad \Pr(X\ (n+1) = s \mid X\ n = t\ n)\Big) \quad\wedge$$

— **The stochastic process** X **is** ***time-homogeneous*****:**

$$\Big(\forall n\ n'\ s\ t.$$
$$\quad \Pr(X\ n = t) \neq 0 \wedge \Pr(X\ n' = t) \neq 0 \longrightarrow$$
$$\quad \Pr(X\ (n+1) = s \mid X\ n = t) = \Pr(X\ (n'+1) = s \mid X\ n' = t)\Big)$$

Markov decision processes

Markov decision process

Probabilistic & non-deterministic transitions

Probabilistic & non-deterministic transitions

- Kernels (coalgebras) of MDPs:

$$\boxed{K :: \sigma \Rightarrow \sigma \text{ pmf set}, \quad K_s \neq \emptyset}$$

Probabilistic & non-deterministic transitions

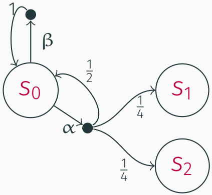
- Kernels (coalgebras) of MDPs:

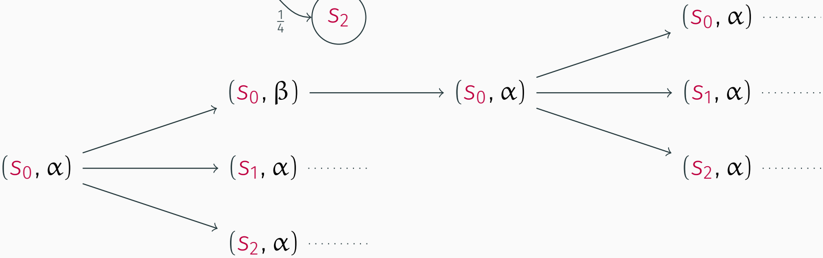
$$\boxed{K :: \sigma \Rightarrow \sigma \text{ pmf set}, \quad K_s \neq \emptyset}$$

- Traditional definition of schedulers:

$$sc :: \sigma \text{ list} \Rightarrow \sigma \text{ pmf}, \quad sc\ (h \cdot s) \in K_s$$

Configurations





Attention: the configuration includes the entire tree!

Configurations on MDPs

$$\texttt{codatatype}\ \sigma\ \mathit{cfg} = \mathit{Cfg}\ (\mathit{state} :\ \sigma)\ (\mathit{act} :\ \sigma\ \mathrm{pmf})\ (\mathit{cont} :\ \sigma \Rightarrow \sigma\ \mathrm{cfg})$$
$$\texttt{where}\ \mathit{state}\ (\mathit{cont}\ c\ s) = s$$

- Induces a *Markov chain*:

$$K^{MC} :: \sigma\ \mathrm{cfg} \Rightarrow \sigma\ \mathrm{cfg}\ \mathrm{pmf}$$
$$K^{MC}{}_c = \mathit{map}\ (\mathit{cont}\ c)\ (\mathit{act}\ c)$$

- Trace space: $\mathcal{T}_c = \mathit{map}_{\mathit{measure}}\ (\mathit{map}_{\mathit{stream}}\ \mathit{state})\ \mathcal{T}^{MC}{}_c$
- Valid Configuration: *act* is always compatible with K

Definition (Minimal Expectation)

$$\mathbb{E}^{\mathrm{min}}{}_{s}\left[f\right] = \prod_{c \in valid_s} \int f \, d\mathcal{T}_c$$

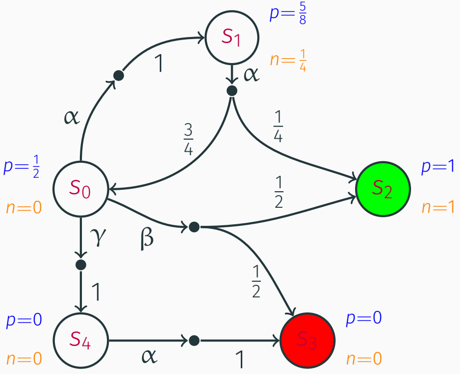
Definition (Minimal Expectation)

$$\mathbb{E}^{\min}{}_{s}[f] = \prod_{c \in valid_s} \int f \, d\mathcal{T}_c$$

Lemma (Iteration Rule)

$$\mathbb{E}^{\min}{}_{s}[f] = \prod_{D \in K_s} \int_t \mathbb{E}^{\min}{}_{s}[f\,(t \cdot \omega)] \; dD$$

Application: Reachability Problem Example



p is $\Pr_s^{\max}(S_1\ U\ S_2)$, n is $\Pr_s^{\min}(S_1\ U\ S_2)$

Application: Reachability problems on MDPs

Goal: certify solutions to reachability problems in MDPs

Application: Reachability problems on MDPs

Goal: certify solutions to reachability problems in MDPs

$\Longrightarrow$ Formalize MDPs and reachability problems
$\mathrm{Pr}_s^{\min}(S_1 \; U \; S_2) = \mathrm{lfp}(\cdots)$

Application: Reachability problems on MDPs

Goal: certify solutions to reachability problems in MDPs

$\Longrightarrow$ Formalize MDPs and reachability problems
$\mathrm{Pr}_s^{\min}(S_1 \ U \ S_2) = \mathrm{lfp}(\cdots)$

$\Longrightarrow$ Implement and verify certification algorithm
Currently: $v \leqslant \mathrm{Pr}^{\min}(S_1 \ U \ S_2)$ and $\mathrm{Pr}^{\max}(S_1 \ U \ S_2) \leqslant v$

Application: Reachability problems on MDPs

Goal: certify solutions to reachability problems in MDPs

$\Longrightarrow$ Formalize MDPs and reachability problems
$\mathrm{Pr}_{s}^{\min}(S_1\ U\ S_2) = \mathrm{lfp}(\cdots)$

$\Longrightarrow$ Implement and verify certification algorithm
Currently: $v \leqslant \mathrm{Pr}^{\min}(S_1\ U\ S_2)$ and $\mathrm{Pr}^{\max}(S_1\ U\ S_2) \leqslant v$

$\Longrightarrow$ Requires proof: $\exists$ optimal memoryless scheduler

Application: Reachability problems on MDPs

Goal: certify solutions to reachability problems in MDPs

$\Longrightarrow$ Formalize MDPs and reachability problems
$\mathsf{Pr}_s^{\min}(S_1 \; U \; S_2) = \mathsf{lfp}(\cdots)$

$\Longrightarrow$ Implement and verify certification algorithm
Currently: $v \leqslant \mathsf{Pr}^{\max}(S_1 \; U \; S_2)$ and $\mathsf{Pr}^{\max}(S_1 \; U \; S_2) \leqslant v$

$\Longrightarrow$ Requires proof: $\exists$ optimal memoryless scheduler

$\Longrightarrow$ Import results by executing algorithm in Isabelle/HOL

Application: pGCL semantics

Present the pGCL semantics similar to [Gretz, Katoen, McIver (2014)]:

$$
\begin{aligned}
\text{pgcl} \quad := \quad & \text{Skip} \\
| \quad & \text{Abort} \\
| \quad & \text{Assign } (\sigma \Rightarrow \sigma) \\
| \quad & \text{Seq pgcl pgcl} \\
| \quad & \text{Par pgcl pgcl} \\
| \quad & \text{If } (\sigma \Rightarrow \mathrm{bool}) \text{ pgcl pgcl} \\
| \quad & \text{Prob } [0,1] \text{ pgcl pgcl} \\
| \quad & \text{While } (\sigma \Rightarrow \text{bool}) \text{ pgcl pgcl}
\end{aligned}
$$

Weakest pre-expectation transformer

$$\text{wp} :: \text{pgcl} \Rightarrow \left(\sigma \Rightarrow \mathbb{R}^{\infty}_{\geqslant 0}\right) \Rightarrow \left(\sigma \Rightarrow \mathbb{R}^{\infty}_{\geqslant 0}\right)$$

$$\begin{aligned}
\text{wp Skip } f &= f \\
\text{wp Abort } f &= \bot \\
\text{wp (Assign } u\text{) } f &= f \circ u \\
\text{wp (Seq } c_1\ c_2\text{) } f &= \text{wp } c_1\ (\text{wp } c_2\ f) \\
\text{wp (Par } c_1\ c_2\text{) } f &= \text{wp } c_1\ f \sqcap \text{wp } c_2\ f \\
\text{wp (If } b\ c_1\ c_2\text{) } f &= \lambda s.\ \text{if } b\ s \text{ then wp } c_1\ f\ s \text{ else wp } c_2\ f\ s \\
\text{wp (Prob } p\ c_1\ c_2\text{) } f &= \lambda s.\ p \cdot \text{wp } c_1\ f\ s + (1 - p) \cdot \text{wp } c_2\ f\ s \\
\text{wp (While } b\ c\text{) } f &= \text{lfp } (\lambda g\ s.\ \text{if } b\ s \text{ then wp } c\ g\ s \text{ else } f\ s)
\end{aligned}$$

Operational semantics as MDP

$$K :: (\text{pgcl} \times \sigma) \Rightarrow (\text{pgcl} \times \sigma)\ \text{pmf set}$$

$$\begin{aligned}
K\ (\text{Skip}, s) &= \ll \text{Skip}, s \gg \\
K\ (\text{Abort}, s) &= \ll \text{Abort}, s \gg \\
K\ (\text{Assign}\ u, s) &= \ll \text{Skip}, u\ s \gg \\
K\ (\text{Seq}\ c_1\ c_2, s) &= \\
&\quad K(c_1, s) \left[\lambda(c_1', s').\ \left\{ \begin{array}{ll} (\text{Seq}\ c_1'\ c_2, s') & \text{if } c_1' \neq \text{Skip} \\ (c_2, s') & \text{else} \end{array} \right\} \right] \\
K\ (\text{Par}\ c_1\ c_2, s) &= \ll c_1, s \gg \cup \ll c_2, s \gg \\
K\ (\text{If}\ b\ c_1\ c_2, s) &= \text{if } b\ s \text{ then } K\ (c_1, s) \text{ else } K\ (c_2, s) \\
K\ (\text{Prob}\ p\ c_1\ c_2, s) &= \{\{(c_1, s) \mapsto p, (c_2, s) \mapsto (1 - p)\}\} \\
K\ (\text{While}\ g\ c, s) &= \begin{cases} \ll \text{Seq}\ c\ (\text{While}\ g\ c), s \gg & \text{if } g\ s \\ \ll \text{Skip}, s \gg & \text{else} \end{cases}
\end{aligned}$$

Definition (Result of a Trace)

$$r\ f\ ((c, s)\cdot\omega) \stackrel{\text{lfp}}{=} \begin{cases} r\ f\ \omega & \text{if } c \neq \mathsf{Skip} \\ f\ s & \text{else} \end{cases}$$

Equate wp and K

Definition (Result of a Trace)

$$r\ f\ ((c, s)\cdot\omega) \stackrel{\text{lfp}}{=} \begin{cases} r\ f\ \omega & \text{if } c \neq \mathrm{Skip} \\ f\ s & \text{else} \end{cases}$$

Theorem (Operational semantics equals denotational semantics)

$$\mathbb{E}^{\min}{}_{(c,s)}(r\ f) = \mathrm{wp}\ \ c\ f\ s$$

Proof

$$\mathbb{E}^{\min}{}_{(c,s)}\,(r\,f) = \mathrm{lfp}\left(\lambda g\ s.\ \prod_{\mu \in K_{(c,s)}} \int_{(c,s)} \begin{cases} g\ (c,s) & \text{if } c \neq \mathrm{Skip} \\ f\ s & \text{else} \end{cases} d\mu\right)\ (c,s)$$

Case $c = \mathrm{Seq}\ c_1\ c_2$:

$$\mathbb{E}^{\min}{}_{(\mathrm{Seq}\ c_1\ c_2,s)}(r\,f) = \mathbb{E}^{\min}{}_{(c_1,s)}\left(r\ \left(\lambda s'.\ \mathbb{E}^{\min}{}_{(c_2,s')}(r\,f)\right)\right)$$

Case $c = \mathrm{While}\ b\ c'$: $\mathbb{E}^{\min}{}_{(\mathrm{While}\ g\ c',s)}(r\,f) = \mathrm{lfp}\ w\ s$

$$w\ g\ s = \prod_{\mu \in K_s} \int_{(d,t)} \begin{cases} g\ (d,t) & \text{if } d \neq \mathrm{Skip} \\ g\ (c',t) & \text{if } b\ t \\ f\ t & \text{else} \end{cases} d\mu$$

Probabilistic Hierarchy

Zoo of Probabilistic System Types

H., Traytel & Lochbihler [ITP 2015]

Ana Sokolva – Coalgebraic Analysis of Probabilistic Systems (2005):

4.4 The hierarchy 107

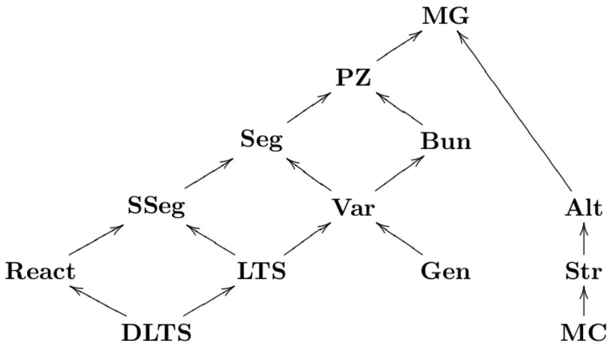


Figure 4.2: Hierarchy of probabilistic system types

Hierarchy of Probabilistic Systems Types

How to ...

Hierarchy of Probabilistic Systems Types

How to …

…model system types?

Hierarchy of Probabilistic Systems Types

How to …

…model system types?

…compare systems of same type?

Hierarchy of Probabilistic Systems Types

How to …

…model system types?

…compare systems of same type?

…compare different system types?

Hierarchy of Probabilistic Systems Types

How to …

…model system types? Coalgebras

…compare systems of same type?

…compare different system types?

Hierarchy of Probabilistic Systems Types

How to ...

- ...model system types? Coalgebras
- ...compare systems of same type? Bisimulation
- ...compare different system types?

Hierarchy of Probabilistic Systems Types

How to ...

...model system types?	Coalgebras
...compare systems of same type?	Bisimulation
...compare different system types?	Embedding respecting bisimulation

Hierarchy of Probabilistic Systems Types

How to …

…model system types? Coalgebras

…compare systems of same type? Bisimulation

…compare different system types? Embedding respecting bisimulation

Hierarchy of Probabilistic Systems Types

How to …

…model system types? Coalgebras

…compare systems of same type? Bisimulation

…compare different system types? Embedding respecting bisimulation

…formalize it in Isabelle/HOL?

Hierarchy of Probabilistic Systems Types

How to ...

...model system types?	Coalgebras
...compare systems of same type?	Bisimulation
...compare different system types?	Embedding respecting bisimulation
...formalize it in Isabelle/HOL?	**codatatype** + Probability Mass Func.

Coalgebras as Codatatypes in Isabelle/HOL

Idea: Analyse transition systems modulo *bisimulation*!

$$\text{Equality} :\Longleftrightarrow \text{Bisimulation}$$

Coalgebras as Codatatypes in Isabelle/HOL

Idea: Analyse transition systems modulo *bisimulation*!

$$\text{Equality} :\Longleftrightarrow \text{Bisimulation}$$

How to model all F-coalgebras as type?

Idea: Analyse transition systems modulo *bisimulation*!

$$\text{Equality} :\Longleftrightarrow \text{Bisimulation}$$

How to model all *F*-coalgebras as type?

$$\textbf{codatatype}\ \tau_F = \mathsf{C}\ (\tau_F\ F)$$

Coalgebras as Codatatypes in Isabelle/HOL

Idea: Analyse transition systems modulo *bisimulation*!

$$\text{Equality} :\Longleftrightarrow \text{Bisimulation}$$

How to model all F-coalgebras as type?

$$\textbf{codatatype } \tau_F = \mathsf{C}\ (\tau_F\ F)$$

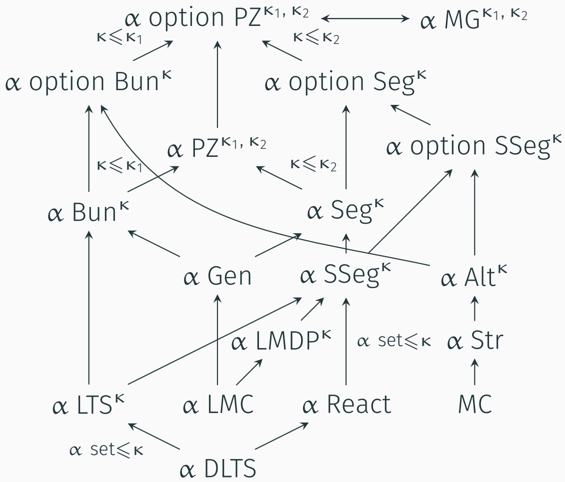
Example (Labeled Markov Chains where $F = \alpha \times \Box\ \mathsf{pmf}$):

$$\textbf{codatatype } \alpha\ \mathsf{mc} = \mathsf{MC}\ (\alpha \times \alpha\ \mathsf{mc}\ \mathsf{pmf})$$

System Types

Name	Functor	Codatatype
Markov chain	σ pmf	MC
Labeled MC	$\alpha \times \sigma$ pmf	α LMC
Labeled MDP	$\alpha \times \sigma$ pmf set_1^{κ}	α LMDP^{κ}
Det. automaton	$\alpha \Rightarrow \sigma$ option	α DLTS
Non-det. automaton	$(\alpha \times \sigma)$ set^{κ}	α LTS^{κ}
Reactive system	$\alpha \Rightarrow \sigma$ pmf option	α React
Generative system	$(\alpha \times \sigma)$ pmf option	α Gen
Stratified system	σ pmf $+$ $(\alpha \times \sigma)$ option	α Str
Alternating system	σ pmf $+$ $(\alpha \times \sigma)$ set^{κ}	α Alt^{κ}
Simple Segala system	$(\alpha \times \sigma$ pmf$)$ set^{κ}	α SSeg^{κ}
Segala system	$(\alpha \times \sigma)$ pmf set^{κ}	α Seg^{κ}
Bundle system	$(\alpha \times \sigma)$ set^{κ} pmf	α Bun^{κ}
Pnueli-Zuck system	$(\alpha \times \sigma)$ set^{κ_1} pmf set^{κ_2}	α $\text{PZ}^{\kappa_1,\kappa_2}$
Most general system	$(\alpha \times \sigma + \sigma)$ set^{κ_1} pmf set^{κ_2}	α $\text{MG}^{\kappa_1,\kappa_2}$

Hierarchy


α option PZ^κ1,κ2
α MG^κ1,κ2
κ⩽κ1
κ⩽κ2
α option Bun^κ
α option Seg^κ
α PZ^κ1,κ2
α option SSeg^κ
κ⩽κ1
κ⩽κ2
α Bun^κ
α Seg^κ
α Gen
α SSeg^κ
α Alt^κ
α LMDP^κ
α set⩽κ
α Str
α LTS^κ
α LMC
α React
MC
α set⩽κ
α DLTS

Hierarchy of Probabilistic System Types

Ana Sokolva – Coalgebraic Analysis of Probabilistic Systems (2005):

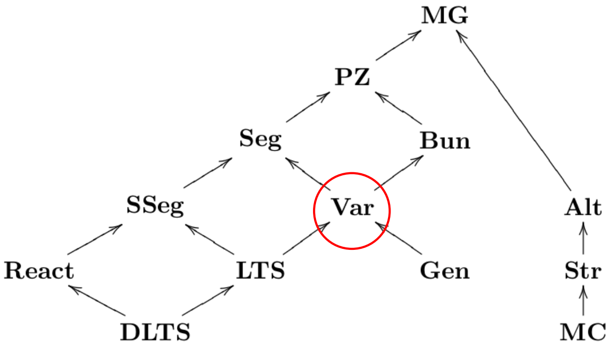


Figure 4.2: Hierarchy of probabilistic system types

Conclusion

Related Work

Formalizing probabilistic trace spaces:

- Formal verification of probabilistic algorithms
  Hurd [thesis 2002]
- Formal reasoning about classified Markov chains in HOL
  Liu, Hasan, Aravantinos, and Tahar [ITP 2013]

Formalizing probabilistic transition systems:

- Probabilistic guarded commands mechanized in HOL
  Hurd, McIver, and Morgan [Theor. Comput. Sci. 2005]
- Proofs of randomized algorithms in Coq
  Audebaud and Paulin-Mohring [MPC 2006]
- Verifying probabilistic correctness in Isabelle with pGCL
  Cock [SSV 2012]

Conclusion

- Coalgebraic & Fixed point approach simplified out theory (also smaller proofs)

Conclusion

- Coalgebraic & Fixed point approach simplified out theory (also smaller proofs)
- Very usable for our applications

Conclusion

- Coalgebraic & Fixed point approach simplified out theory (also smaller proofs)
- Very usable for our applications
    - Probabilistic model checking

Conclusion

- Coalgebraic & Fixed point approach simplified out theory (also smaller proofs)
- Very usable for our applications
  - Probabilistic model checking
  - pGCL semantics equivalence

Conclusion

- Coalgebraic & Fixed point approach simplified out theory (also smaller proofs)
- Very usable for our applications
    - Probabilistic model checking
    - pGCL semantics equivalence
    - Small examples on fixed models

Conclusion

- Coalgebraic & Fixed point approach simplified out theory (also smaller proofs)
- Very usable for our applications
    - Probabilistic model checking
    - pGCL semantics equivalence
    - Small examples on fixed models
- Formalized hierarchy of probabilistic systems types
  Found *two* flaws

Conclusion

- Coalgebraic & Fixed point approach simplified out theory (also smaller proofs)
- Very usable for our applications
  - Probabilistic model checking
  - pGCL semantics equivalence
  - Small examples on fixed models
- Formalized hierarchy of probabilistic systems types
  Found *two* flaws
- Probability theory also used for:

Conclusion

- Coalgebraic & Fixed point approach simplified out theory (also smaller proofs)
- Very usable for our applications
    - Probabilistic model checking
    - pGCL semantics equivalence
    - Small examples on fixed models
- Formalized hierarchy of probabilistic systems types
  Found *two* flaws
- Probability theory also used for:
    - Density Compiler [Eberl, H., Nipkow (ESOP 2015)]

Conclusion

- Coalgebraic & Fixed point approach simplified out theory (also smaller proofs)
- Very usable for our applications
    - Probabilistic model checking
    - pGCL semantics equivalence
    - Small examples on fixed models
- Formalized hierarchy of probabilistic systems types
  Found *two* flaws
- Probability theory also used for:
    - Density Compiler [Eberl, H., Nipkow (ESOP 2015)]
    - Central Limit Theorem [Avigad, H., Serafin (2014)]

Conclusion

- Coalgebraic & Fixed point approach simplified out theory (also smaller proofs)
- Very usable for our applications
  - Probabilistic model checking
  - pGCL semantics equivalence
  - Small examples on fixed models
- Formalized hierarchy of probabilistic systems types
  Found *two* flaws
- Probability theory also used for:
  - Density Compiler [Eberl, H., Nipkow (ESOP 2015)]
  - Central Limit Theorem [Avigad, H., Serafin (2014)]
- Future Work:
  Average Runtime Analysis, Probabilistic Programming