Probabilistic relations for modelling epistemic and aleatoric uncertainty

Its semantics and automated reasoning with theorem proving

Kangfeng Ye, Jim Woodcock, Simon Foster

ROBOSTAR

robostar.cs.york.ac.uk

June 28, 2023

University of York

Engineering and Physical Sciences Research Council

Outline

ROBOSTAR

Outline

Why probability in robotics?

Uncertainties in autonomous robots:

- Unpredictable environment,
- Sensor: limits and noise,
- Actuator: control noise, mechanical failure,
- Model (abstraction of real world) error, and
- Control algorithmic approximations.

Probabilism: widely used in society and science to model uncertainty

> *"A theory that certainty is impossible especially in the sciences and that probability suffices to govern belief and action."* — *(Merriam-Webster dictionary)*

ROBOSTAR

Probabilistic models (PMs)

Ubiquitous: distributed systems, machine learning, artificial intelligence, robotics and autonomous systems, quantum computation etc.

Major impact on machine intelligence

Probabilistic models (PMs)

Ubiquitous: distributed systems, machine learning, artificial intelligence, robotics and autonomous systems, quantum computation etc.

Major impact on machine intelligence

Example (Intelligence)

Intersection without a signal, an autonomous vehicle slows down and coordinates its actions with others by gathering their probabilistic information.

Motivation: why semantics for PMs or probabilistic programs (PPs)?

1. Challenges to analyse PMs, subject to the size and complexity
2. Errors are easily introduced in the development stage from PMs to PPs
3. PPs are very difficult to be tested thoroughly

Motivation: why semantics for PMs or probabilistic programs (PPs)?

③ PPs are very difficult to be tested thoroughly

For non-probabilistic programs,

> *"Program testing can be used to show the presence of bugs, ..."*
>
> — *Edsger W. Dijkstra*

Motivation: why semantics for PMs or probabilistic programs (PPs)?

③ PPs are very difficult to be tested thoroughly

For non-probabilistic programs,

"Program testing can be used to show the presence of bugs, . . . "

— *Edsger W. Dijkstra*

For probabilistic programs,

"regular testing can't even establish that presence"

— *Annabelle McIver and Carroll Morgan*

ROBOSTAR

Motivation: why semantics for PMs or probabilistic programs (PPs)?

③ PPs are very difficult to be tested thoroughly

Challenges to debug a probabilistic program (locate and correct errors in the source code),

Example (Quantitative errors)

Quantitative information are the result of statistical analysis of many executions

Motivation: why semantics for PMs or probabilistic programs (PPs)?

1. Challenges to analyse PMs, subject to the size and complexity
2. Errors are easily introduced in the development stage from PMs to PPs
3. PPs are very difficult to be tested thoroughly

Needs: unambiguous and rigorous mathematical semantics, and analysed on a computer

ROBOSTAR

Motivation: Probabilistic semantics for RoboChart

RoboChart: reactive systems, CSP (nondeterminism, communication, and concurrency) with discrete-time (tock-CSP) semantics.
RoboSim: refinement of RoboChart to simulation level

ROBOSTAR

Motivation: Probabilistic semantics for RoboChart

RoboChart: reactive systems, CSP (nondeterminism, communication, and concurrency) with discrete-time (tock-CSP) semantics.
RoboSim: refinement of RoboChart to simulation level
What we want: Probabilistic semantics for RoboChart with all features + theorem proving

Motivation: Probabilistic semantics for RoboChart

RoboChart: reactive systems, CSP (nondeterminism, communication, and concurrency) with discrete-time (tock-CSP) semantics.
RoboSim: refinement of RoboChart to simulation level
What we want: Probabilistic semantics for RoboChart with all features + theorem proving
What we have now: Probabilistic semantics in

- Probabilistic designs [WCF+19, YFW21]: nondeterministic probabilistic sequential programming, finite states

ROBOSTAR

Motivation: Probabilistic semantics for RoboChart

RoboChart: reactive systems, CSP (nondeterminism, communication, and concurrency) with discrete-time (tock-CSP) semantics.

RoboSim: refinement of RoboChart to simulation level

What we want: Probabilistic semantics for RoboChart with all features + theorem proving

What we have now: Probabilistic semantics in

- Probabilistic designs [WCF+19, YFW21]: nondeterministic probabilistic sequential programming, finite states
- PRISM [YCF+22]: DTMC and MDP
  - time in Markov chains are different from that in RoboChart;
  - DTMC and MDP in PRISM: closed-world (no subject to inputs)
  - No concept of refinement and equivalence
  - State space explosion problem

ROBOSTAR

Motivation: Probabilistic semantics for RoboChart

RoboChart: reactive systems, CSP (nondeterminism, communication, and concurrency) with discrete-time (tock-CSP) semantics.
RoboSim: refinement of RoboChart to simulation level
What we want: Probabilistic semantics for RoboChart with all features + theorem proving

Our first thought

What on literatures: probabilistic process algebras

- Probabilistic extensions: PTS, CCS, CSP, ACP
- Markov models: concurrent, interactive, Markovian process algebra (PEPA)
- Probabilistic I/O automata, Probabilistic and time extension of automata

No practical tool support or not support theorem proving

Our pathways to the goal

Our goal

Probabilistic semantics for RoboChart supporting discrete time + nondeterminism + refinement + communication + concurrency + theorem proving

Pathways: a probabilistic programming language (PPL)

- A sequential PPL supporting discrete distributions with theorem proving
  - Discrete time
  - + Nondeterminism
  - + Refinement
  - + Continuous distributions
- A concurrent PPL with communication
- ...

ROBOSTAR

Our contributions

- An imperative sequential PPL supporting discrete distributions
- A probabilistic semantic framework: probabilistic relations
- Model both epistemic (subjective Bayesian) and aleatoric uncertainties

  Epistemic the lack of knowledge of information and reducible

  Aleatoric the natural randomness of physical processes and irreducible
- Support theorem proving with a set of algebraic laws for simplification and verification
- Six verified probabilistic examples

Outline

Flip a coin - How important it is? How difficult to reason about?

A fair die (Knuth and Yao[1]), any discrete distribution (McIver and Morgan[2])

Example (Flip a coin till heads)

while (outcome **is** tails) { outcome = flip a coin }

- What's its semantics?
- What's the probabilistic distribution?
- Does this loop terminate?
- On average, how many flips are needed?

[1] Knuth, D., Yao, A.: The complexity of nonuniform random number generation.
[2] McIver, A., Morgan, C. (2020): Correctness by Construction for Probabilistic Programs.

ROBOSTAR

Flip a coin - How important it is? How difficult to reason about?

A fair die (Knuth and Yao[1]), any discrete distribution (McIver and Morgan[2])

Example (Flip a coin till heads)

while (outcome **is** tails) { outcome = flip a coin }

- What's its semantics? the outcome is heads
- What's the probabilistic distribution?
- Does this loop terminate?
- On average, how many flips are needed?

[1] Knuth, D., Yao, A.: The complexity of nonuniform random number generation.

[2] McIver, A., Morgan, C. (2020): Correctness by Construction for Probabilistic Programs.

ROBOSTAR

Flip a coin - How important it is? How difficult to reason about?

A fair die (Knuth and Yao[1]), any discrete distribution (McIver and Morgan[2])

Example (Flip a coin till heads)

while (outcome **is** tails) { outcome = flip a coin }

- What's its semantics? the outcome is heads
- What's the probabilistic distribution? the outcome is heads in terms of iterations: $(1/2)^n$
- Does this loop terminate?
- On average, how many flips are needed?

[1] Knuth, D., Yao, A.: The complexity of nonuniform random number generation.

[2] McIver, A., Morgan, C. (2020): Correctness by Construction for Probabilistic Programs.

Flip a coin - How important it is? How difficult to reason about?

A fair die (Knuth and Yao[1]), any discrete distribution (McIver and Morgan[2])

Example (Flip a coin till heads)

while (outcome **is** tails) { outcome = flip a coin }

- What's its semantics? the outcome is heads
- What's the probabilistic distribution? the outcome is heads in terms of iterations: $(1/2)^n$
- Does this loop terminate? $\sum_{n=0}^{\infty}(1/2)^n = 1$, Almost-sure termination (AST)
- On average, how many flips are needed?

[1] Knuth, D., Yao, A.: The complexity of nonuniform random number generation.

[2] McIver, A., Morgan, C. (2020): Correctness by Construction for Probabilistic Programs.

Flip a coin - How important it is? How difficult to reason about?

A fair die (Knuth and Yao[1]), any discrete distribution (McIver and Morgan[2])

Example (Flip a coin till heads)

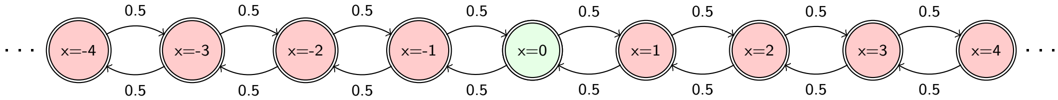
while (outcome **is** tails) { outcome = flip a coin }

- What's its semantics? the outcome is heads
- What's the probabilistic distribution? the outcome is heads in terms of iterations: $(1/2)^n$
- Does this loop terminate? $\sum_{n=0}^{\infty}(1/2)^n = 1$, Almost-sure termination (AST)
- On average, how many flips are needed? $\sum_{n=0}^{\infty}(1/2)^n * n = 2$, Positive AST

[1] Knuth, D., Yao, A.: The complexity of nonuniform random number generation.
[2] McIver, A., Morgan, C. (2020): Correctness by Construction for Probabilistic Programs.

(Symmetric) simple random walker

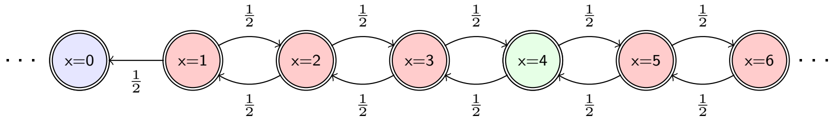
Start from the origin $x = 0$, will the robot always return to it (recurrent) infinitely often?



Symmetric random walk

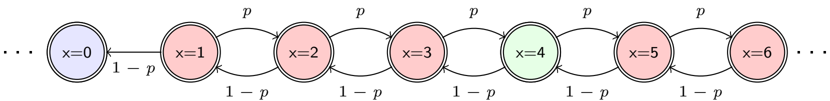
(Symmetric) simple random walker

Start from any position x, will the robot terminate at 0?
On average, how many steps?



(Symmetric) simple random walker

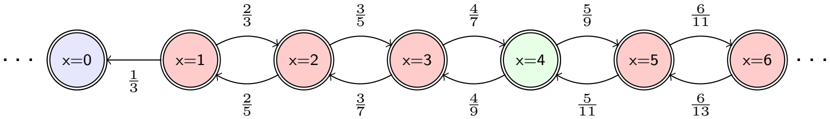
Start from any position x, will the robot terminate at 0?
On average, how many steps?



.

(Symmetric) simple random walker

Start from any position x, will the robot terminate at 0?
On average, how many steps?



The fair-in-the-limit random walk from McIver et al. [MMKK17]

Probabilistic models - hardness on termination analysis

Termination of non-probabilistic programs

- Absolute termination vs. non-termination (divergence)

Termination of probabilistic programs

- Almost-sure termination (AST) vs. non AST
- Positive AST vs. null AST
- Termination becomes an arithmetic problem

ROBOSTAR

Probabilistic models - hardness on termination analysis

Arithmetic (coin flip)

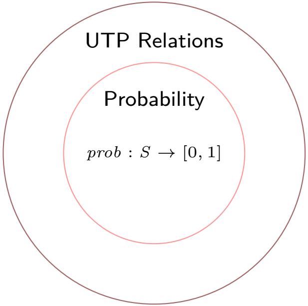
- Summation of sequences, (geometric) series: $\sum_{n=0}^{\infty}(1/2)^n * n = 2$
- Convergence: ratio test $f(n) \widehat{=} (1/2)^n * n$
- Solve an equation

$$\sum_{n=0}^{\infty} f(n+1) = \sum_{n=0}^{\infty} f(n) + f(0) = \underline{\sum_{n=0}^{\infty} f(n)}$$

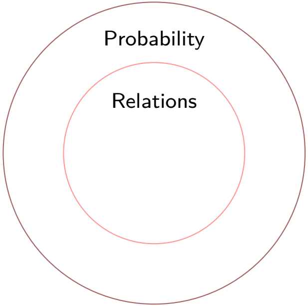
$$\sum_{n=0}^{\infty} f(n+1) = \sum_{n=0}^{\infty} (1/2)^{(n+1)} * n + \sum_{n=0}^{\infty} (1/2)^{(n+1)} = \underline{\left(\sum_{n=0}^{\infty} f(n)\right) /2 + 1}$$

ROBOSTAR

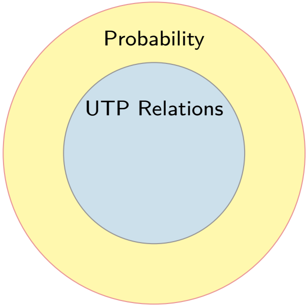
Probabilistic relations



Probabilistic designs
$(S \to [0,1]) \times (S \to [0,1]) \to \mathbb{B}$



Hehner's PPP[1]



Our Probabilistic Relations
$S \times S \to \mathbb{R}$

[1] Eric Hehner, Probabilistic predicative programming (PPP), MPC 2004.

Probabilistic relations

- Formalise Hehner's syntax and semantics
- Iverson bracket notation $[\![P]\!]$
  - Separate UTP relations and distributions to simplify reasoning
- Bridge semantic gap for loops in Hehner's work
  - Enrich semantics domain to superdistributions and subdistributions
  - Unit interval and its pointwise function as complete lattices
- Mechanised in Isabelle/UTP: 60 definitions + 390 lemmas and theorems
- Six examples: 65 definitions + 170 lemmas and theorems

ROBOSTAR

Outline

Unit real interval: ureal

ureal and conversion

$$\mathit{ureal} \mathrel{\widehat{=}} \{0\,..\,1\}$$
$$\overline{u} \mathrel{\widehat{=}} (u :: \mathbb{R})$$
$$\underline{r} \mathrel{\widehat{=}} \mathit{min}\,(\mathit{max}\,(0, r)\,, 1)$$
$$u_1 + u_2 \mathrel{\widehat{=}} \underline{(\mathit{min}\,(1, \overline{u_1} + \overline{u_2}))}$$
$$u_1 - u_2 \mathrel{\widehat{=}} \underline{(\mathit{max}\,(0, \overline{u_1} - \overline{u_2}))}$$
$$u_1 * u_2 \mathrel{\widehat{=}} \underline{(\overline{u_1} * \overline{u_2})}$$

Theorem: ureal

$$u_1 < u_2 \Rightarrow \overline{u_1} < \overline{u_2}$$
$$\underline{(\overline{u})} = u$$
$$(r \geq 0 \land r \leq 1) \Rightarrow \overline{(\underline{r})} = r$$

Complete lattice

$$(\mathit{ureal}, \leq, <, \mathit{0}, \mathit{1}, \mathit{min}, \mathit{max}, \bigsqcap, \bigsqcup)$$

Unit real interval: ureal functions

Function type

$$[S]\mathit{urexpr} \mathrel{\widehat{=}} S \to \mathit{ureal}$$

constants

$$\dot{0} \mathrel{\widehat{=}} \lambda\, s \bullet 0 \qquad \dot{1} \mathrel{\widehat{=}} \lambda\, s \bullet 1$$

$$\mathring{0} \mathrel{\widehat{=}} \lambda\, s \bullet \mathit{0} \qquad \mathring{1} \mathrel{\widehat{=}} \lambda\, s \bullet \mathit{1}$$

Pointwise functions

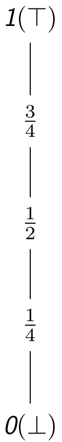
$$f - g \mathrel{\widehat{=}} (\lambda\, x \bullet f(x) - g(x))$$
$$f + g \mathrel{\widehat{=}} (\lambda\, x \bullet f(x) + g(x))$$
$$f \le g \mathrel{\widehat{=}} (\forall\, x \bullet f(x) \le g(x))$$
$$f < g \mathrel{\widehat{=}} (\forall\, x \bullet f(x) < g(x))$$

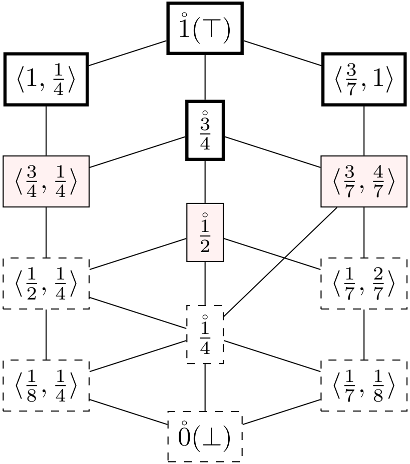
Complete lattice

$$([S]\mathit{urexpr}, \le, <, \mathring{0}, \mathring{1}, \sqcap, \sqcup, \bigsqcap, \bigsqcup)$$

ROBOSTAR

Unit real interval: complete lattice





Iverson brackets

Iverson bracket

$$[\![P]\!] : [S]pred \rightarrow (S \rightarrow \mathbb{R})$$

$$[\![P]\!] \mathrel{\widehat{=}} (\textbf{if } P \textbf{ then } 1 \textbf{ else } 0)_e$$

Iverson brackets

Theorems

$$[\![\mathit{false}]\!] = \dot{0} \qquad [\![\mathit{true}]\!] = \dot{1} \qquad Q \sqsubseteq P \Rightarrow [\![P]\!] \le [\![Q]\!] \qquad [\![\neg\, P]\!] = (1 - [\![P]\!])_e$$

$$[\![P \wedge Q]\!] = ([\![P]\!] * [\![Q]\!])_e \qquad [\![P \vee Q]\!] = ([\![P]\!] + [\![Q]\!] - [\![P]\!] * [\![Q]\!])_e$$

$$[\![\lambda\, s \bullet s \in A \cap B]\!] = ([\![\lambda\, s \bullet s \in A]\!] * [\![\lambda\, s \bullet s \in B]\!])_e$$

$$([\![\lambda\, s \bullet s \in A]\!] + [\![\lambda\, s \bullet s \in B]\!])_e = ([\![\lambda\, s \bullet s \in A \cap B]\!] + [\![\lambda\, s \bullet s \in A \cup B]\!])_e$$

$$(\mathit{max}\,(x, y))_e = (x * [\![x > y]\!] + y * [\![x \le y]\!])_e \qquad (\mathit{min}\,(x, y))_e = \cdots$$

$$\sum_{P(k)} f\,(k) = \sum_{k} (f * [\![P]\!])_e\,(k)$$

Type Abbreviation and conversions

Types

$$[V, S]\mathit{expr} \mathrel{\widehat=} S \to V$$
$$[S]\mathit{rexpr} \mathrel{\widehat=} [\mathbb{R}, S]\mathit{expr}$$
$$[S_1, S_2]\mathit{rvfun} \mathrel{\widehat=} [\mathbb{R}, S_1 \times S_2]\mathit{expr}$$
$$[S]\mathit{rvhfun} \mathrel{\widehat=} [S, S]\mathit{rvfun}$$
$$[S]\mathit{urexpr} \mathrel{\widehat=} [\mathit{ureal}, S]\mathit{expr}$$
$$[S_1, S_2]\mathit{prfun} \mathrel{\widehat=} [\mathit{ureal}, S_1 \times S_2]\mathit{urexpr}$$
$$[S]\mathit{prhfun} \mathrel{\widehat=} [S, S]\mathit{prfun}$$

Conversion

$$P : [S_1, S_2]\mathit{prfun}$$
$$f : [S_1, S_2]\mathit{rvfun}$$
$$\overline{P} \mathrel{\widehat=} \mathit{rvfun\_of\_prfun}(P)$$
$$\underline{f} \mathrel{\widehat=} \mathit{prfun\_of\_rvfun}(f)$$

Distribution functions

Probability and distribution functions

$$\underbracket{p} \mathrel{\widehat{=}} \forall\, s \bullet (p)_e\,(s) \qquad \textit{is\_prob}(p) \mathrel{\widehat{=}} \underbracket{p \geq 0 \land p \leq 1}$$

$$\textit{is\_dist}(p) \mathrel{\widehat{=}} \textit{is\_prob}(p) \land \Sigma_\infty s \bullet p(s) = 1$$

$$\textit{is\_subdist}(p) \mathrel{\widehat{=}} \textit{is\_prob}(p) \land \Sigma_\infty s \bullet p(s) > 0 \land \Sigma_\infty s \bullet p(s) \leq 1$$

Theorems

$$\textit{is\_prob}([\![p]\!]) \quad \textit{is\_dist}(p) \Rightarrow \textit{is\_subdist}(p) \quad \textit{is\_prob}\left(\overline{P}\right) \quad \textit{is\_prob}\left(\dot{1} - \overline{P}\right)$$

$$\underline{\left(\overline{P}\right)} = P \quad \text{if } P : [S_1, S_2]\textit{prfun} \qquad \textit{is\_prob}(p) \Rightarrow \overline{\underline{(p)}} = p \qquad \overline{\left(\underline{[\![p]\!]}\right)} = [\![p]\!]$$

ROBOSTAR

Distribution functions

Probability and distribution functions over final states

$\tilde{p} \mathrel{\widehat{=}} \lambda\, s\, s' \bullet p(s, s')$ $\quad$ $\mathit{is\_final\_prob}(p) \mathrel{\widehat{=}} \underbrace{\mathit{is\_prob}(\tilde{p})}$

$\mathit{is\_final\_dist}(p) \mathrel{\widehat{=}} \underbrace{\mathit{is\_dist}(\tilde{p})}$ $\quad$ $\mathit{is\_final\_subdist}(p) \mathrel{\widehat{=}} \underbrace{\mathit{is\_subdist}(\tilde{p})}$

$\mathit{summable\_on\_final}(p) \mathrel{\widehat{=}} (\forall\, s \bullet \mathit{summable}\,(\tilde{p}(s), \mathbb{U}))$

$\mathit{summable\_on\_final2}(p, q) \mathrel{\widehat{=}} \left(\forall\, s \bullet \mathit{summable}\left(\lambda\, s' \bullet p(s, s') * q(s, s'), \mathbb{U}\right)\right)$

$\mathit{final\_reachable}(p) \mathrel{\widehat{=}} \left(\forall\, s \bullet \exists\, s' \bullet p(s, s') > 0\right)$

$\mathit{final\_reachable2}(p, q) \mathrel{\widehat{=}} \left(\forall\, s \bullet \exists\, s' \bullet p(s, s') > 0 \wedge q(s, s') > 0\right)$

ROBOSTAR

Distribution functions

Final distributions and subdistributions

$$\mathit{is\_final\_dist}(p) \Rightarrow \left(\begin{array}{l} \mathit{is\_prob}(p) \land (\forall s \bullet \Sigma_\infty s' \bullet p(s, s') = 1) \land \\ \mathit{summable\_on\_final}(p) \land \mathit{final\_reachable}(p) \end{array} \right)$$

$$\mathit{is\_final\_subdist}(p) \Rightarrow \left(\begin{array}{l} \mathit{is\_prob}(p) \land (\forall s \bullet \Sigma_\infty s' \bullet p(s, s') > 0) \land \\ (\forall s \bullet \Sigma_\infty s' \bullet p(s, s') \leq 1) \\ \mathit{summable\_on\_final}(p) \land \mathit{final\_reachable}(p) \end{array} \right)$$

ROBOSTAR

Normalisation

Normalisation

$$\mathcal{N}(p) \mathrel{\widehat=} (p / (\Sigma_\infty s : S \bullet p(s)))_e$$
$$\mathcal{N}_f(p) \mathrel{\widehat=} \left(p / \left(\Sigma_\infty v_0 : S_2 \bullet p[v_0/\mathbf{v}']\right)\right)_e$$
$$\mathcal{N}_\alpha(x, p) \mathrel{\widehat=} \left(p / \left(\Sigma_\infty x_0 : T_x \bullet p[x_0/x']\right)\right)_e \quad \text{Alphabetised}$$
$$\mathcal{U}(x, A) \mathrel{\widehat=} \mathcal{N}_\alpha\left(x, \left[\!\left[\bigsqcup v \in A \bullet x := v \right]\!\right]\right) \quad \text{Uniform distributions}$$

Normalisation is final distribution

$$\mathit{is\_nonneg}(p) \land \mathit{final\_reachable}(p) \land \mathit{summable\_on\_final}(p) \Rightarrow \mathit{is\_final\_dist}(\mathcal{N}_f(p))$$

ROBOSTAR

Outline

Syntax and semantics

$$\mathit{II}_p \mathrel{\widehat=} \underline{[\![\mathit{II}]\!]} \qquad \text{(skip)}$$

$$(x :=_p e) \mathrel{\widehat=} \underline{[\![x := e]\!]} \qquad \text{(assignment)}$$

$$(P \oplus_r Q) \mathrel{\widehat=} \underline{\left(\overline{r} * \overline{P} + (\dot{1} - \overline{r}) * \overline{Q}\right)_e} \qquad \text{(probabilistic choice)}$$

$$(\mathbf{if}_c\ b\ \mathbf{then}\ P\ \mathbf{else}\ Q) \mathrel{\widehat=} \underline{\left(\mathbf{if}\ b\ \mathbf{then}\ \overline{P}\ \mathbf{else}\ \overline{Q}\right)_e} \qquad \text{(conditional choice)}$$

$$P ; Q \mathrel{\widehat=} \underline{\left(\Sigma_\infty v_0 \bullet \overline{P}[v_0/\mathbf{v}'] * \overline{Q}[v_0/\mathbf{v}]\right)_e} \qquad \text{(sequential composition)}$$

$$R \parallel T \mathrel{\widehat=} \underline{\mathcal{N}_f\ (R * T)_e} \qquad \text{(parallel composition)}$$

$$\mathcal{F}_P^b(X) \mathrel{\widehat=} \mathcal{F}\,(b, P, X) \mathrel{\widehat=} \mathbf{if}_c\ b\ \mathbf{then}\ (P ; X)\ \mathbf{else}\ \mathit{II}_p \qquad \text{(loop characterisation function)}$$

$$\textbf{\textit{while}}_p\ b\ \textbf{\textit{do}}\ P\ \textbf{\textit{od}} \mathrel{\widehat=} \mu_p\, X \bullet \mathcal{F}_P^b(X) \qquad \text{(while loop by least fixed point)}$$

$$\textbf{\textit{while}}_p^\top\ b\ \textbf{\textit{do}}\ P\ \textbf{\textit{od}} \mathrel{\widehat=} \nu_p\, X \bullet \mathcal{F}_P^b(X) \qquad \text{(while loop by strongest fixed point)}$$

Subjective Bayesian

Sequential composition: conditional probability (for actions)
Parallel composition: joint probability (for new knowledge)

$$\text{posterior} = \frac{\text{prior} * \text{likelihood}}{\text{evidence}} \quad \text{or} \quad P(A \mid B) = \frac{P(A)P(B \mid A)}{P(B)}$$

Programs

$$(prior;\ action) \parallel (likelihood)$$

ROBOSTAR

Top, bottom, skip, and assignment

Top and bottom

$$\top = \mathring{1} \qquad \perp = \mathring{0} \qquad \underline{\dot{1}} = \mathring{1}$$

$$\underline{\dot{0}} = \mathring{0} \qquad \overline{\mathring{1}} = \dot{1} \qquad \overline{\mathring{0}} = \dot{0}$$

$$P \leq \mathring{1} \qquad P \geq \mathring{0} \qquad P * \mathring{1} = P$$

$$p * \dot{0} = \dot{0} \qquad p * \dot{1} = p \qquad P * \mathring{0} = \mathring{0}$$

Skip and assignment

$$\mathit{II}_p = (x :=_p x)$$

$$\textit{is\_final\_dist}\,(\overline{\mathit{II}_p})$$

$$\overline{\left(\underline{[\![\mathit{II}]\!]}\right)} = [\![\mathit{II}]\!]$$

$$\textit{is\_final\_dist}\,(\overline{x :=_p e})$$

Probabilistic choice

Probabilistic choice

$$is\_final\_dist\left(\overline{P}\right) \land is\_final\_dist\left(\overline{Q}\right) \Rightarrow is\_final\_dist\left(\overline{P \oplus_r Q}\right)$$

$$\left(P \oplus_{\mathring{0}} Q\right) = Q \qquad \left(P \oplus_{\mathring{1}} Q\right) = P$$

$$\left(P \oplus_r Q\right) = \left(Q \oplus_{\mathring{1}-r} P\right) \qquad \left(P \oplus_r Q\right) = \underline{\overline{r} * \overline{P} + \left(\mathring{1} - \overline{r}\right) * \overline{Q}}$$

$$r^{\Uparrow} \mathrel{\widehat{=}} \lambda(s, s') \bullet r(s)$$

$$\left(\begin{array}{l} \underbrace{\left(\mathring{1} - w_1\right) * \left(\mathring{1} - w_2\right) = \left(\mathring{1} - r_2\right)} \\ \land \underbrace{w_1 = r_1 * r_2} \end{array}\right) \Rightarrow \left(P \oplus_{w_1^{\Uparrow}} \left(Q \oplus_{w_2^{\Uparrow}} R\right)\right) = \left(\left(P \oplus_{r_1^{\Uparrow}} Q\right) \oplus_{r_2^{\Uparrow}} R\right)$$

ROBOSTAR

Conditional choice

Conditional choice

$$\mathit{is\_final\_dist}\left(\overline{P}\right) \wedge \mathit{is\_final\_dist}\left(\overline{Q}\right) \Rightarrow \mathit{is\_final\_dist}\left(\overline{\mathbf{if}_c\ b\ \mathbf{then}\ P\ \mathbf{else}\ Q}\right)$$

$$(\mathbf{if}_c\ b\ \mathbf{then}\ P\ \mathbf{else}\ P) = P$$

$$(\mathbf{if}_c\ b\ \mathbf{then}\ P\ \mathbf{else}\ Q) = \left(P \oplus_{\underline{[\![b]\!]}}\ Q\right)$$

$$(P_1 \le P_2 \wedge Q_1 \le Q_2) \Rightarrow (\mathbf{if}_c\ b\ \mathbf{then}\ P_1\ \mathbf{else}\ Q_1) \le (\mathbf{if}_c\ b\ \mathbf{then}\ P_2\ \mathbf{else}\ Q_2)$$

Sequential composition

Sequential composition

$$\textit{is\_final\_dist}\left(\overline{P}\right) \wedge \textit{is\_final\_dist}\left(\overline{Q}\right) \Rightarrow \textit{is\_final\_dist}\left(\overline{P\,;\,Q}\right)$$

$$\mathring{0}\,;\,P = \mathring{0} \qquad P\,;\,\mathring{0} = \mathring{0} \qquad I\!I_p\,;\,P = P \qquad P\,;\,I\!I_p = P$$

$$\textit{is\_final\_dist}(\overline{P}) \Rightarrow P\,;\,\mathring{1} = \mathring{1} \qquad (P_1 \leq P_2 \wedge Q_1 \leq Q_2) \Rightarrow (P_1\,;\,Q_1) \leq (P_2\,;\,Q_2)$$

$$\textit{is\_final\_subdist}\left(\overline{P}\right) \wedge \cdots \left(\overline{Q}\right) \wedge \cdots \left(\overline{R}\right) \Rightarrow (P\,;(Q\,;\,R) = (P\,;\,Q)\,;\,R)$$

$$\textit{is\_final\_subdist}\left(\overline{P}\right) \Rightarrow$$

$$\left(P\,;\,(\mathbf{if}_c\ b\ \mathbf{then}\ Q\ \mathbf{else}\ R) = \underline{\left(\overline{(P\,;\,([\![b]\!] * Q))} + \overline{(P\,;\,([\![\neg\, b]\!] * R))}\right)_e}\right)$$

Sequential composition

Sequential composition

$$\underline{\llbracket p \rrbracket} \,;\, \underline{\llbracket q \rrbracket} = \underline{\left(\Sigma_\infty v_0 \bullet \llbracket p[v_0/\mathbf{v}'] \wedge q[v_0/\mathbf{v}] \rrbracket\right)_e}$$

$$c_1 \neq c_2 \Rightarrow \underline{\llbracket x' = c_1 \rrbracket} \,;\, \underline{\llbracket x = c_2 \rrbracket} = \mathring{0}$$

$$\underline{\llbracket x = c_0 \wedge x := c_1 \rrbracket} \,;\, \underline{\llbracket x = c_1 \rrbracket} = \underline{\llbracket x = c_0 \rrbracket}$$

$$\underline{\llbracket x = c_0 \wedge x := c_1 \rrbracket} \,;\, \underline{\llbracket x = c_1 \wedge x := c_2 \rrbracket} = \underline{\llbracket x = c_0 \wedge x' = c_2 \rrbracket}$$

ROBOSTAR

Uniform distribution

Uniform distribution

$$\mathcal{U}\,(x, \varnothing) = \dot{0}$$

$$\mathit{finite}(A) \Rightarrow \mathit{is\_prob}\,(\mathcal{U}\,(x, A))$$

$$\mathit{finite}(A) \wedge A \neq \varnothing \Rightarrow \mathit{is\_final\_dist}\,(\mathcal{U}\,(x, A))$$

$$\mathit{finite}(A) \wedge A \neq \varnothing \Rightarrow (\forall\, v \in A \bullet \mathcal{U}\,(x, A)\,;[\![x = v]\!] = (1/\,\mathit{card}(A))_e)$$

$$\mathit{finite}(A) \wedge A \neq \varnothing \Rightarrow \left(\mathcal{U}\,(x, A) = \left[\!\!\left[\bigcup v \in A \bullet x := v\right]\!\!\right] /\, \mathit{card}(A)\right)$$

$$\mathit{finite}(A) \wedge A \neq \varnothing \Rightarrow \left(\underline{\mathcal{U}\,(x, A)}\,; P = \underline{(\Sigma_\infty v \in A \bullet \overline{P}[v/x]) \,/\, \mathit{card}(A)}\right)$$

Parallel composition

Parallel composition

$$\textit{is\_nonneg}(p * q) \Rightarrow \textit{is\_prob}\left(\mathcal{N}_f\ (p * q)_e\right)$$

$$\left(\begin{array}{l} \textit{is\_final\_prob}(p) \wedge \textit{is\_final\_prob}(q) \wedge \\ (\textit{summable\_on\_final}(p) \vee \textit{summable\_on\_final}(q)) \\ \wedge\ \textit{final\_reachable2}(p, q) \end{array}\right) \Rightarrow \textit{is\_final\_dist}\,(p \parallel q)$$

$$\left(\ \textit{is\_nonneg}(p) \wedge \textit{is\_nonneg}(q) \wedge \neg\ \textit{final\_reachable2}(p, q)\ \right) \Rightarrow p \parallel q = \mathring{0}$$

$$\dot{0} \parallel p = \mathring{0} \qquad p \parallel \dot{0} = \mathring{0} \qquad p \parallel q = q \parallel p$$

$$c \neq 0 \wedge \textit{is\_final\_dist}(p) \Rightarrow (\lambda\, s \bullet c) \parallel p = \underline{p}$$

$$c \neq 0 \wedge \textit{is\_final\_dist}(p) \Rightarrow p \parallel (\lambda\, s \bullet c) = \underline{p}$$

Parallel composition

Parallel composition

$$\left(\begin{array}{l} \mathit{is\_nonneg}(p) \land \mathit{is\_nonneg}(q) \land \mathit{is\_nonneg}(r) \land \\ \mathit{summable\_on\_final2}(p, q) \land \mathit{summable\_on\_final2}(q, r) \land \\ \mathit{final\_reachable2}(p, q) \land \mathit{final\_reachable2}(q, r) \end{array}\right)$$

$$\Rightarrow (p \parallel q) \parallel r = p \parallel (q \parallel r)$$

$$\mathit{summable\_on\_final}(\overline{Q}) \Rightarrow (\overline{P} \parallel \overline{Q}) \parallel \overline{R} = \overline{P} \parallel (\overline{Q} \parallel \overline{R})$$

$$\mathit{finite}(A) \land A \neq \varnothing \Rightarrow$$

$$\mathcal{U}\,(x, A) \parallel p = \underline{\left(\left(\Sigma_\infty v \in A \bullet [\![x := v]\!] * p[v/x']\right) / \left(\Sigma_\infty v \in A \bullet p[v/x']\right)\right)_e}$$

Semantic gap for loops in Hehner's PPP

Semantic gap for loops

1. PPP: semantics for basic constructs like sequential composition, probabilistic and conditional choice
2. ???
3. ???
4. ???
5. PPP: find a fixed point

Semantic gap for loops in Hehner's PPP

Our approach

1. PPP: semantics for basic constructs like sequential composition, probabilistic and conditional choice
2. Scott-Continuity
3. ???
4. ???
5. PPP: find a fixed point

Semantic gap for loops in Hehner's PPP

Our approach

1. PPP: semantics for basic constructs like sequential composition, probabilistic and conditional choice
2. Scott-Continuity
3. Kleene fixed point theorem
4. ???
5. PPP: find a fixed point

Semantic gap for loops in Hehner's PPP

Our approach

1. PPP: semantics for basic constructs like sequential composition, probabilistic and conditional choice
2. Scott-Continuity
3. Kleene fixed point theorem
4. Unique fixed point theorem
5. PPP: find a fixed point

Semantic gap for loops in Hehner's PPP

Our approach

1. PPP: semantics for basic constructs like sequential composition, probabilistic and conditional choice
2. Scott-Continuity
3. Kleene fixed point theorem
4. Unique fixed point theorem
5. PPP: find a fixed point

Only for probabilistic programs P whose possible final states are always finite

ROBOSTAR

Semantics of loops

Knaster–Tarski fixed-point theorem

- Provided $(X, \leq)$ is a complete lattice and $F : X \to X$ is monotonic,
- then the set of fixed points of F also forms a complete lattice.
- The LFP is the infimum of the pre-fixed points, and the GFP is the supremum of the post-fixed points.

  $\mu F \mathrel{\widehat{=}} \bigsqcap \{u : X \mid F(u) \leq u\}$

  $\nu F \mathrel{\widehat{=}} \bigsqcup \{u : X \mid u \leq F(u)\}$

While loops

While loops

$$is\_final\_dist\left(\overline{P}\right) \Rightarrow \textbf{\textit{while}}_p\ b\ \textbf{do}\ P\ \textbf{od} = \mathcal{F}_P^b\left(\textbf{\textit{while}}_p\ b\ \textbf{do}\ P\ \textbf{od}\right)$$

$$\textbf{\textit{while}}_p\ \textbf{\textit{false}}\ \textbf{do}\ P\ \textbf{od} = I\!I_p$$

$$\textbf{\textit{while}}_p\ \textbf{\textit{true}}\ \textbf{do}\ P\ \textbf{od} = \mathring{0}$$

$$is\_final\_dist\left(\overline{P}\right) \Rightarrow \textbf{\textit{while}}_p^\top\ b\ \textbf{do}\ P\ \textbf{od} = \mathcal{F}_P^b\left(\textbf{\textit{while}}_p^\top\ b\ \textbf{do}\ P\ \textbf{od}\right)$$

$$\textbf{\textit{while}}_p^\top\ \textbf{\textit{false}}\ \textbf{do}\ P\ \textbf{od} = I\!I_p$$

$$is\_final\_dist\left(\overline{P}\right) \Rightarrow \textbf{\textit{while}}_p^\top\ \textbf{\textit{true}}\ \textbf{do}\ P\ \textbf{od} = \mathring{1}$$

Continuity and Kleene fixed-point theorem

Scott continuity

- Suppose $(X, \leq)$ and $(X', \leq')$ are complete lattices,
- A function $F : X \to X'$ is Scott-continuous or continuous if, for every non-empty chain $S \subseteq X$,
  - $F\left(\bigsqcup_{\mathbf{X}} S\right) = \bigsqcup_{\mathbf{X'}} F(S)$
- $F(S) \mathrel{\widehat{=}} \{d \in S \bullet F(d)\}$: the relational image of S under F or the range of F domain restricted to S.

Continuity and Kleene fixed-point theorem

Kleene fixed-point theorem

- Provided $(X, \leq)$ is a complete lattice, and $F : X \to X$ is continuous,
- then F has a least fixed point μF and a greatest fixed point νF,

  $\mu F = \bigsqcup_{n \geq 0} F^n(\bot)$

  $\nu F = \bigsqcap_{n \geq 0} F^n(\top)$
- Here we use $\bigsqcup_{n \geq 0} F^n(\bot)$ to denote $\bigsqcup \{n : \mathbb{N} \bullet F^n(\bot)\}$

ROBOSTAR

Continuity and Kleene fixed-point theorem: loop iterations

Loop iteration and iteration difference

$$\mathcal{I}\,(n, b, P, X) \mathrel{\widehat=} \left(\textbf{if}\ n = 0\ \textbf{then}\ X\ \textbf{else}\ \mathcal{F}_P^b\,(\mathcal{I}\,(n-1, b, P, X))\right)$$

$$\mathcal{F}_0(b, P, X) \mathrel{\widehat=} \textbf{if}_c\, b\ \textbf{then}\ (P\,;\,X)\ \textbf{else}\ \mathring{0}$$

$$\mathcal{ID}\,(n, b, P, X) \mathrel{\widehat=} (\textbf{if}\ n = 0\ \textbf{then}\ X\ \textbf{else}\ \mathcal{F}_0\,(b, P, \mathcal{ID}\,(n-1, b, P, X)))$$

Increasing and decreasing chains

$$\mathit{is\_final\_dist}\,(\overline{P}) \Rightarrow \mathit{incseq}\,(\lambda\, n \bullet \mathcal{I}\,(n, b, P, \mathring{0}))$$

$$\mathit{is\_final\_dist}\,(\overline{P}) \Rightarrow \mathit{decseq}\,(\lambda\, n \bullet \mathcal{I}\,(n, b, P, \mathring{1}))$$

Continuity of loop iteration functions

Finite possible final states

$$\mathit{finite\_final}(P) \mathrel{\widehat{=}} \forall\, s \bullet \mathit{finite}\left\{s' : S \mid P(s, s') > 0\right\}$$

Continuity of loop iteration functions

$$\left(\mathit{is\_final\_dist}\left(\overline{P}\right) \land \mathit{finite\_final}(P)\right) \Rightarrow \left(\begin{array}{l} \mathcal{F}_P^b\left(\bigsqcup n \bullet \mathcal{I}\left(n, b, P, \mathring{0}\right)\right) = \left(\bigsqcup n \bullet \mathcal{I}\left(n, b, P, \mathring{0}\right)\right) \\ \mathcal{F}_P^b\left(\bigsqcap n \bullet \mathcal{I}\left(n, b, P, \mathring{1}\right)\right) = \left(\bigsqcap n \bullet \mathcal{I}\left(n, b, P, \mathring{1}\right)\right) \end{array}\right)$$

Kleene fixed-point theorem of probabilistic loops

Semantics of probabilistic loops by iterations

$$\left(\mathit{is\_final\_dist}\left(\overline{P}\right) \wedge \mathit{finite\_final}(P)\right)$$
$$\Rightarrow \left(\begin{array}{l} \textbf{\textit{while}}_p\ b\ \textbf{\textit{do}}\ P\ \textbf{\textit{od}} = \left(\bigsqcup n \bullet \mathcal{I}\left(n, b, P, \mathring{0}\right)\right) \\ \textbf{\textit{while}}_p^{\top}\ b\ \textbf{\textit{do}}\ P\ \textbf{\textit{od}} = \left(\bigsqcap n \bullet \mathcal{I}\left(n, b, P, \mathring{1}\right)\right) \end{array}\right)$$

Unique fixed-point theorem: motivation example

Flip a fair coin till heads

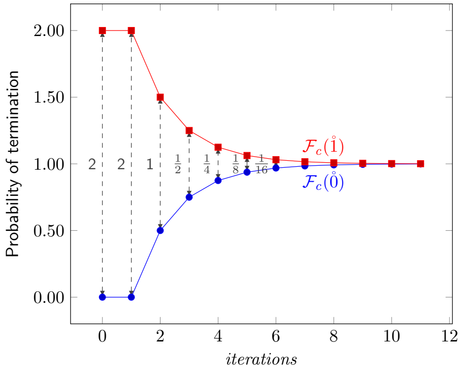
$Tcoin ::= hd \mid tl$

alphabet $cstate = c :: Tcoin$

$cflip \mathrel{\widehat{=}} c :=_p hd \oplus_{1/2} c :=_p tl$

$flip \mathrel{\widehat{=}} \textbf{while}_p\ c = tl\ \textbf{do}\ cflip\ \textbf{od}$

$\mathcal{F}_c \mathrel{\widehat{=}} \mathcal{F}^{c=tl}_{cflip}(X)$



Unique fixed-point theorem

Unique fixed-point theorem

$$\begin{pmatrix} \textit{is\_final\_dist}\left(\overline{P}\right) \land \\ \textit{finite\_final}(P) \land \\ \left(\forall s \bullet \left(\lambda n \bullet \overline{\mathcal{ID}\left(n, b, P, \mathring{1}\right)}(s)\right) \xrightarrow{n \to \infty} 0\right) \land \\ \mathcal{F}_P^b\left(fp\right) = fp \end{pmatrix}$$

$$\Rightarrow \left(\textbf{while}_p\ b\ \textbf{do}\ P\ \textbf{od} = fp\right) \land \left(\textbf{while}_p^\top\ b\ \textbf{do}\ P\ \textbf{od} = fp\right)$$

Unique fixed-point theorem

Unique fixed-point theorem

$$\left(\begin{array}{l} \mathit{is\_final\_dist}\left(\overline{P}\right) \land \\ \mathit{finite\_final}(P) \land \\ \left(\forall s \bullet \left(\lambda n \bullet \overline{\mathcal{ID}\left(n, b, P, \mathring{1}\right)}(s)\right) \xrightarrow{n \to \infty} 0\right) \land \\ \mathcal{F}_P^b\left(fp\right) = fp \end{array}\right)$$
$$\Rightarrow \left(\textbf{\textit{while}}_p\ b\ \textbf{\textit{do}}\ P\ \textbf{\textit{od}} = fp\right) \land \left(\textbf{\textit{while}}_p^\top\ b\ \textbf{\textit{do}}\ P\ \textbf{\textit{od}} = fp\right)$$

Finding the semantics of a probabilistic loop is merely to prove the four assumptions:

- Hehner: the first and fourth assumptions

Semantic gap for loops in Hehner's PPP is filled

Semantic gap for loops: our approach

1. PPP: semantics for basic constructs like sequential composition, probabilistic and conditional choice
2. Scott-Continuity
3. Kleene fixed point theorem
4. Unique fixed point theorem
5. PPP: find a fixed point

Only for probabilistic programs P whose possible final states are always finite

Outline

Doctor Who's Tardis Attack

Two robots, the Cyberman C and the Dalek D, attack Doctor Who's Tardis once a day between them.

C has a probability of 1/2 of a successful attack, while D has a probability of 3/10 of a successful attack.

C attacks more often than D, with a probability of 3/5 on a particular day (and so D attacks with a probability of 2/5 on that day).

What is the probability that there is a successful attack today?

Doctor Who's Tardis Attack

DWTA

$$Attacker ::= C \mid D \qquad Status ::= S \mid F$$

$$\mathtt{alphabet}\ dwtastate = a :: Attacker \qquad s :: Status$$

$$dwta \mathrel{\widehat{=}} \begin{array}{l} \Big((a :=_p C) \,;\, \big(s :=_p S \oplus_{1/2} s :=_p F\big)\Big) \oplus_{3/5} \\ \Big((a :=_p D) \,;\, \big(s :=_p S \oplus_{3/10} s :=_p F\big)\Big) \end{array}$$

Doctor Who's Tardis Attack

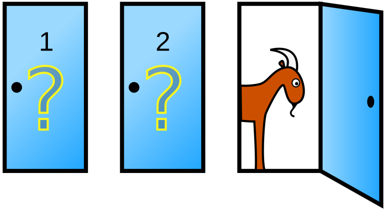
DWTA

$$dwta = \left(\begin{array}{l} 3/10 * [\![a' = C \land s' = S]\!] + \\ 3/10 * [\![a' = C \land s' = F]\!] + \\ 6/50 * [\![a' = D \land s' = S]\!] + \\ 14/50 * [\![a' = D \land s' = F]\!] \end{array} \right)_e$$

$$\overline{dwta} \,; [\![s = S]\!] = (21/50)_e$$

Monty hall problem

From wikipedia:



A game, three doors, one car and two goats, the host knows which door has the car, the contestant is offered to choose a door (let's say door 1), then the host opens door 3, the contestant has an opportunity to change the door. Should the contestant switch to door 2?

Monty hall problem

Modelling

$$\texttt{alphabet}\ mhstate = p :: \mathbb{N} \qquad c :: \mathbb{N} \qquad m :: \mathbb{N}$$

$$init \mathrel{\widehat{=}} \mathcal{U}(p, \{0\,..\,2\})\,;\mathcal{U}(c, \{0\,..\,2\})$$

$$mc \mathrel{\widehat{=}} \left((m :=_p (c+1) \bmod 3) \oplus_{1/2} (m :=_p (c+2) \bmod 3)\right)$$

$$mha \mathrel{\widehat{=}} \left(\mathbf{if}_c\, c = p\ \mathbf{then}\ mc\ \mathbf{else}\ m := 3 - c - p \right)$$

$$mha\_nc \mathrel{\widehat{=}} init\,;mha\,;\mathit{II}_p$$

$$mha\_c \mathrel{\widehat{=}} init\,;mha\,;c := 3 - c - m$$

Monty hall problem

Win probability

$$\overline{mha\_nc}\,;[\![c=p]\!]=(1/3)_e$$
$$\overline{mha\_c}\,;[\![c=p]\!]=(2/3)_e$$

So the contestant should switch because of the higher probability ($2/3$ vs. $1/3$) to win.

Forgetful Monty problem

Suppose now that Monty forgets which door has the prize behind it. He just opens either of the doors not chosen by the contestant.
If the prize is revealed ($m' = p'$), then obviously the contestant switches their choice to that door. So the contestant will surely win.
However, if the prize is not revealed ($m' \neq p'$), should the contestant switch?

New fact learned

Forgetful Monty problem

Modelling

$$\mathbf{alphabet}\ mhstate = p :: \mathbb{N} \qquad c :: \mathbb{N} \qquad m :: \mathbb{N}$$

$$init \mathrel{\widehat{=}} \mathcal{U}\,(p, \{0\,..\,2\})\,;\mathcal{U}\,(c, \{0\,..\,2\})$$

$$mc \mathrel{\widehat{=}} \Big((m :=_p (c+1) \text{ mod } 3) \oplus_{1/2} (m :=_p (c+2) \text{ mod } 3)\Big)$$

$$forgetful\_monty \mathrel{\widehat{=}} init\,;\,mc$$

$$learn\_fact \mathrel{\widehat{=}} forgetful\_monty \parallel \underline{[\![m' \neq p']\!]}$$

Forgetful Monty problem

Win probability

$$learn\_fact = \underline{\left(\begin{array}{l} [\![p' \in \{0\,..\,2\}]\!] * [\![c' \in \{0\,..\,2\}]\!] * [\![m' \neq p']\!] * \\ ([\![m' = (c'+1)\%3]\!] + [\![m' = (c'+2)\%3]\!])\,/12 \end{array} \right)_e}$$

$$\overline{learn\_fact}\,;[\![c = p]\!] = (1/2)_e$$

So it doesn't matter whether the contestant switches or not.

ROBOSTAR

Robot localisation

A circular room has two doors and a wall. A robot is equipped with a noisy door sensor which maps position to $door$ *or* $wall$*.*
Doors are at position 0 and 2, and position 1 is a blank wall.
ine a predicate $door(p) \mathrel{\widehat{=}} p = 0 \vee p = 2$ *and introduce a program variable* $bel \in \{0\,..\,2\}$ *to denote the position of the robot that we believe.*
When the reading of the door sensor is $door$*, it has four times more likely to be right than wrong.*
We are interested in questions like how many measurements and moves are necessary to get a confident estimation of the robot's location?

Robot localisation

Modelling

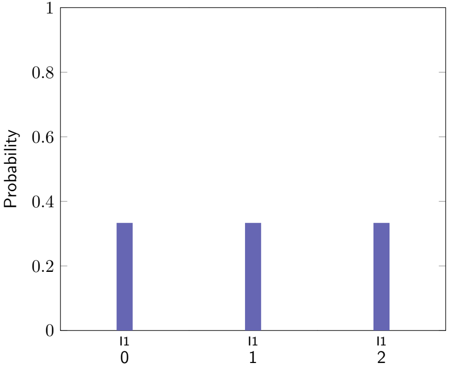
$$\texttt{alphabet}\ state = bel :: \mathbb{N}$$
$$door(p) \mathrel{\widehat{=}} p = 0 \lor p = 2$$
$$scale\_door \mathrel{\widehat{=}} \left(3 * [\![door(bel')]\!] + 1\right)_e$$
$$scale\_wall \mathrel{\widehat{=}} \left(3 * [\![\neg\ door(bel')]\!] + 1\right)_e$$
$$init \mathrel{\widehat{=}} \mathcal{U}\left(bel, \{0 \mathinner{..} 2\}\right)$$
$$move\_right \mathrel{\widehat{=}} \left(bel := (bel + 1) \bmod 3\right)$$

ROBOSTAR

Robot localisation

Belief

$$init = \underline{\left(\begin{array}{l} 1/3 * [\![bel' = 0]\!] + \\ 1/3 * [\![bel' = 1]\!] + \\ 1/3 * [\![bel' = 2]\!] \end{array}\right)}$$

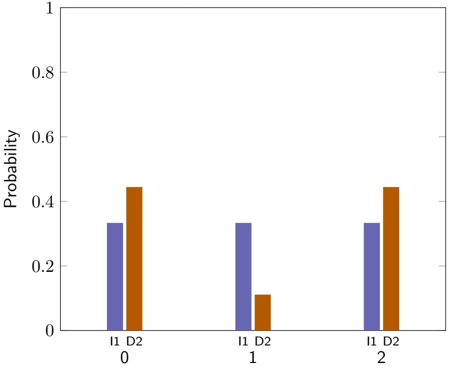


Robot localisation

Belief

$$init \parallel scale\_door$$

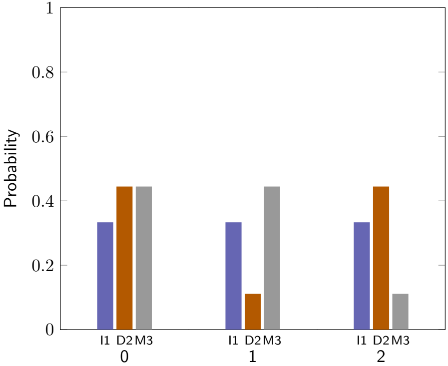
$$= \frac{\begin{pmatrix} 4/9 * [\![bel' = 0]\!] + \\ 1/9 * [\![bel' = 1]\!] + \\ 4/9 * [\![bel' = 2]\!] \end{pmatrix}}{}$$



Robot localisation

Belief

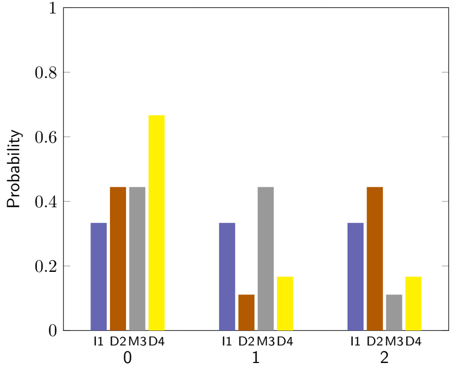
$$(init \parallel scale\_door) \, ; move\_right$$
$$= \underline{\left(\begin{array}{l} 4/9 * [\![bel' = 0]\!] + \\ 4/9 * [\![bel' = 1]\!] + \\ 1/9 * [\![bel' = 2]\!] \end{array} \right)}$$



Robot localisation

Belief

$$
\begin{aligned}
& init \parallel scale\_door \\
& \quad ; move\_right \\
& \qquad \parallel scale\_door \\
&= \left(\begin{array}{l} 2/3 * [\![bel' = 0]\!] + \\ 1/6 * [\![bel' = 1]\!] + \\ 1/6 * [\![bel' = 2]\!] \end{array} \right)
\end{aligned}
$$

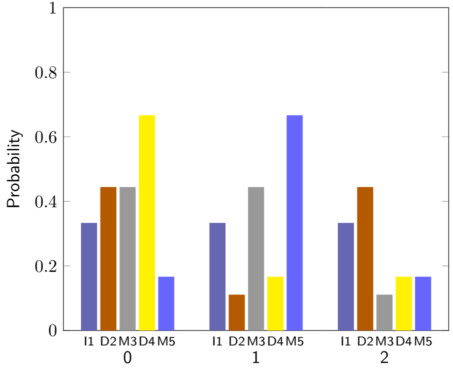


ROBOSTAR

Robot localisation

Belief

$$
\begin{aligned}
& init \parallel scale\_door \\
& \quad ; move\_right \\
& \qquad \parallel scale\_door \\
& \qquad ; move\_right \\
& = \underline{\left(\begin{array}{l} 1/6 * [\![bel' = 0]\!] + \\ 2/3 * [\![bel' = 1]\!] + \\ 1/6 * [\![bel' = 2]\!] \end{array} \right)}
\end{aligned}
$$

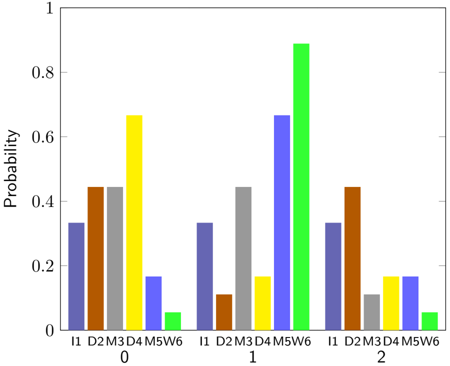


Robot localisation

Belief

$$\begin{aligned} & init \parallel scale\_door \\ & \quad ; move\_right \\ & \qquad \parallel scale\_door \\ & \qquad\quad ; move\_right \\ & \qquad\qquad \parallel scale\_wall \end{aligned}$$

$$= \underline{\left(\begin{array}{l} 1/18 * [\![bel' = 0]\!] + \\ 8/9 * [\![bel' = 1]\!] + \\ 1/18 * [\![bel' = 2]\!] \end{array} \right)}$$



ROBOSTAR

COVID diagnosis

Consider people use a COVID test to diagnose if they may or may not have contracted COVID. The test result is binary and could be positive or negative. The test, however, is imperfect. It doesn't not always give a correct result. We are interested in several questions.

How likely a randomly selected person has covid if the first test result is positive?

Is it necessary to have the second test to reassure the result?

How much can the second test contribute to the diagnosis?

COVID diagnosis

Modelling

$$CovidTest ::= Pos \mid Neg \quad \mathbf{alphabet}\ cdstate = c :: bool \qquad ct :: CovidTest$$

$$Init \mathrel{\widehat=} \mathbf{if}_p\ p_1\ \mathbf{then}\ c := True\ \mathbf{else}\ c := False$$

$$TestAction \mathrel{\widehat=} \mathbf{if}_c\ c\ \mathbf{then}\ (ct :=_p Pos \oplus_{p_2} ct :=_p Neg)\ \mathbf{else}\ (ct :=_p Pos \oplus_{p_3} ct :=_p Neg)$$

$$FirstTestPos \mathrel{\widehat=} (Init\,;\,TestAction) \parallel [\![ct' = Pos]\!]$$

$$SecondTestPos \mathrel{\widehat=} (FirstTestPos\,;\,TestAction) \parallel [\![ct' = Pos]\!]$$

The prior probability of a randomly selected patient having COVID is p_1.
The sensitivity (true positive) of the test is p_2, and
The specificity (true negative) is $1 - p_3$.

ROBOSTAR

COVID diagnosis

Results

$$FirstTestPos = \left(\left(\begin{array}{l} [\![c']\!] * [\![ct' = Pos]\!] * p_1 * p_2 + \\ [\![\neg\, c']\!] * [\![ct' = Pos]\!] * (1 - p_1) * p_3 \end{array} \right) \Big/ (p_1 * p_2 + (1 - p_1) * p_3) \right)_e$$

$$SecondTestPos = \left(\left(\begin{array}{l} [\![c']\!] * [\![ct' = Pos]\!] * p_1 * p_2^2 + \\ [\![\neg\, c']\!] * [\![ct' = Pos]\!] * (1 - p_1) * p_3^2 \end{array} \right) \Big/ (p_1 * p_2^2 + (1 - p_1) * p_3^2) \right)_e$$

Provided $p_1 = 0.002$, $p_2 = 0.89$, and $p_3 = 0.05$, the probability of the patient having COVID is 3.4% (given one positive test) and 38.84% (given two positive tests).

Flip a coin till heads - Parametric model and verification

Modelling - parametric

$$Tcoin ::= hd \mid tl$$

$$\texttt{alphabet}\ cstate = c :: Tcoin$$

$$cflip \mathrel{\widehat=} c :=_p hd \oplus_{1/2} c :=_p tl$$

$$flip \mathrel{\widehat=} \textbf{while}_p\ c = tl\ \textbf{do}\ cflip\ \textbf{od}$$

$$pflip(p) \mathrel{\widehat=} \textbf{while}_p\ c = tl\ \textbf{do}\ c :=_p hd \oplus_p c :=_p tl\ \textbf{od}$$

$$\texttt{alphabet}\ cstate\_t = t :: \mathbb{N} \qquad c :: Tcoin$$

$$pflip\_t(p) \mathrel{\widehat=} \textbf{while}_p\ c = tl\ \textbf{do}\ \left(c :=_p hd \oplus_p c :=_p tl\right); t :=_p t + 1\ \textbf{od}$$

Flip a coin till heads - Parametric model and verification

Semantics

$$\mathit{flip} = \underline{[\![c' = hd]\!]} \qquad p \neq 0 \Rightarrow \mathit{pflip}(p) = \underline{[\![c' = hd]\!]}$$

$$p \neq 0 \Rightarrow \mathit{pflip\_t}(p) = \underline{\begin{pmatrix} [\![c = hd]\!] * [\![c' = hd]\!] * [\![t' = t]\!] + \\ [\![c = tl]\!] * [\![c' = hd]\!] * [\![t' \geq t+1]\!] * (1 - \overline{p})^{t'-t-1} * \overline{p} \end{pmatrix}_e}$$

Termination and expected run time

$$p \neq 0 \Rightarrow \overline{\mathit{pflip\_t}(p)} \, ; [\![c = hd]\!] = (1)_e$$

$$p \neq 0 \Rightarrow \overline{\mathit{flip\_t\_p}(p)} \, ; t = ([\![c = hd]\!] * t + [\![c = tl]\!] * (t + 1/\overline{p}))_e$$

ROBOSTAR

Throw a pair of dice

Modelling

$$Tdice ::= \{1..6\}$$

$$\texttt{alphabet}\ dstate\_t = t :: \mathbb{N} \qquad d_1 :: Tdice \qquad d_2 :: Tdice$$

$$dice\_t \mathrel{\widehat{=}} \textbf{\textit{while}}_p\ d_1 \neq d_2\ \textbf{\textit{do}}\ \underline{\mathcal{U}\left(d_1, Tdice\right)} ; \underline{\mathcal{U}\left(d_2, Tdice\right)} ; t :=_p t + 1\ \textbf{\textit{od}}$$

Throw a pair of dice

Semantics, termination, and expected run time

$$dice\_t = \left(\begin{array}{l} [\![d_1 = d_2]\!] * [\![t' = t \wedge d_1' = d_1 \wedge d_2' = d_2]\!] + \\ [\![d_1 \neq d_2]\!] * [\![d_1' = d_2']\!] * [\![t' \geq t + 1]\!] * (5/6)^{t'-t-1} * (1/36) \end{array} \right)_e$$

$$\overline{dice\_t} \,;\, [\![d_1 = d_2]\!] = (1)_e$$

$$\overline{dice\_t} \,;\, t = ([\![d1 = d2]\!] * t + [\![d1 \neq d2]\!] * (t + 6))_e$$

Outline

Background and motivations

Complexity of probabilistic reasoning and our approach

Basic definitions: *ureal*, Iverson brackets, and distributions

Probabilistic Relations: syntax and semantics

Examples

Conclusion

Conclusion

- Probabilistic relations (PR): A PPL and also a probabilistic semantics framework
- Syntax and semantics of PR
- A collection of algebraic laws for each construct of PR
- Particularly, semantics for probabilistic loops using iterations
- Probabilistic examples: two contain loops
- Mechanisation of the PPL, theorems, and examples in Isabelle/UTP

ROBOSTAR

Future work

- Discrete time
  - Time primitives of RoboChart: clock, reset, since, deadline, wait etc.
  - Semantics to (PRISM) DTMCs, equivalence to simplify models (tractable for model checking)
    - Make time in DTMCs explicitly
    - Combine more commands into fewer (i.e. simultaneous assignments)

Future work

- Discrete time
- Nondeterminism
  - Parametrised models
    - $P \sqcap Q \Longrightarrow (PQ(b) \mathrel{\widehat{=}} \mathbf{if}_c\ b\ \mathbf{then}\ P\ \mathbf{else}\ Q)$: boolean parameter
    - $\boldsymbol{while}_p\ b\ \boldsymbol{do}\ P \sqcap Q\ \boldsymbol{od} \Longrightarrow (PQWhile(bf : \mathbb{N} \to \mathbb{B}) \mathrel{\widehat{=}} \boldsymbol{while}_p\ b\ \boldsymbol{do}\ PQ(bf(t))\ \boldsymbol{od})$
    - Functions (bf whose domain is the iteration index) as parameters
  - Schedule or policy becomes the full or partial instantiation of models by their parameters
  - The complexity of nondeterministic models is the same as the deterministic models
  - Semantics to (PRISM) MDPs, equivalence and parametric (tractable for model checking)

Future work

- Discrete time
- Nondeterminism
- Refinement and abstraction
  1. From nondeterministic models to deterministic models,
  2. Superdistributions to distributions,
  3. Discrete distributions into uniform distributions such as random number generators
  4. Data refinement: abstract types (such as int) into concrete types (int32, int64 etc.)

  Correctness-by-Construction

  Abstraction to simplify PRISM DTMCs and MDPs

ROBOSTAR

Future work

- Discrete time
- Nondeterminism
- Refinement and abstraction
- Probabilistic programs that have infinite possible final states in the loop body
  - Cousot's constructive version of the Knaster–Tarski fixed-point theorem[1]
  - Weaken continuity into monotonicity
  - Treat the least fixed point as the stationary limit of transfinite iteration sequences

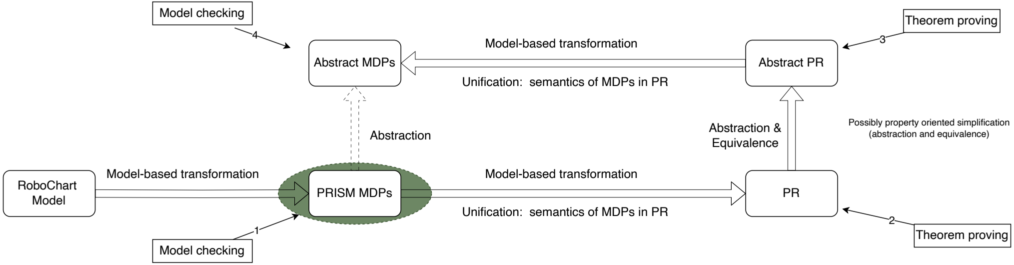
[1] P. Cousot, R. Cousot, Constructive versions of Tarski's fixed point theorems, Pacific Journal of Mathematics 81 (1) (1979) 43–57.

ROBOSTAR

Future work

- Discrete time
- Nondeterminism
- Refinement and abstraction
- Probabilistic programs that have infinite possible final states in the loop body
- Continuous distributions: measure theory

An potential application



Thank you!

https://robostar.cs.york.ac.uk/

Annabelle McIver, Carroll Morgan, Benjamin Lucien Kaminski, and Joost-Pieter Katoen.
A New Proof Rule for Almost-Sure Termination.
Proc. ACM Program. Lang., 2(POPL), dec 2017.

Jim Woodcock, Ana Cavalcanti, Simon Foster, Alexandre Mota, and Kangfeng Ye.
Probabilistic Semantics for RoboChart.
In Pedro Ribeiro and Augusto Sampaio, editors, *Unifying Theories of Programming*, pages 80–105, Cham, 2019. Springer International Publishing.

Kangfeng Ye, Ana Cavalcanti, Simon Foster, Alvaro Miyazawa, and Jim Woodcock.
Probabilistic modelling and verification using robochart and prism.
21:667–716, 2022.

Kangfeng Ye, Simon Foster, and Jim Woodcock.
Automated reasoning for probabilistic sequential programs with theorem proving.

In Uli Fahrenberg, Mai Gehrke, Luigi Santocanale, and Michael Winter, editors, *Relational and Algebraic Methods in Computer Science*, pages 465–482, Cham, 2021. Springer International Publishing.