A Denotational Semantics for Low-Level Probabilistic Programs with Nondeterminism

Di Wang [1] Jan Hoffmann [1] Thomas Reps [2,3]

[1]Carnegie Mellon University

[2]University of Wisconsin

[3]GrammaTech, Inc.

Probabilistic Programs

Draw random **data** from distributions

Condition **control-flow** at random

Low-Level Probabilistic Programs

High-Level Features:

- Functional (Borgström et al. 2016)
- Higher-order (Ehrhard, Pagani, and Tasson 2018)
- Recursive types (Vákár, Kammar, and Staton 2019)

Formal semantics has been well studied.

$\xrightarrow{\textsc{Compiler}}$

Low-Level Features:

- Imperative
- Unstructured control-flow

Operational semantics:
(Ferrer Fioriti and Hermanns 2015)

Denotational semantics:
This work

Benefits of A Denotational Semantics

- Abstraction from details about program executions
- Compositionality

Low-Level Probabilistic Programs

High-Level Features:

- Functional (Borgström et al. 2016)
- Higher-order (Ehrhard, Pagani, and Tasson 2018)
- Recursive types (Vákár, Kammar, and Staton 2019)

Formal semantics has been well studied.

$\xrightarrow{\text{COMPILER}}$

Low-Level Features:

- Imperative
- Unstructured control-flow

Operational semantics: (Ferrer Fioriti and Hermanns 2015)

Denotational semantics: **This work**

Benefits of A Denotational Semantics

- Abstraction from details about program executions
- Compositionality

Low-Level Probabilistic Programs

Example

The following code implements a variant of geometric distributions.

```
n := 0;
while prob(0.9) do
    n := n + 1;
    if n ≥ 10 then break
    else continue
od
```

There are multiple possible executions of the program, e.g., n could end up with 0, 3, or 10.

Principle

Probabilistic programs establish input/**output-distribution** relations. A probabilistic program can be modeled as a function in $X \longrightarrow \mathcal{D}(X)$, where X is a program state space and $\mathcal{D}(X)$ consists of probability distributions over X.

Low-Level Probabilistic Programs

Example

The following code implements a variant of geometric distributions.

$$
\begin{array}{l}
n := 0; \\
\textbf{while prob}(0.9)\ \textbf{do} \\
\quad n := n + 1; \\
\quad \textbf{if}\ n \geq 10\ \textbf{then break} \\
\quad \textbf{else continue} \\
\textbf{od}
\end{array}
$$

There are multiple possible executions of the program, e.g., n could end up with 0, 3, or 10.

Principle

Probabilistic programs establish input/**output-distribution** relations. A probabilistic program can be modeled as a function in $X \to \mathcal{D}(X)$, where X is a program state space and $\mathcal{D}(X)$ consists of probability distributions over X.

Low-Level Probabilistic Programs

Example

The following code implements a variant of geometric distributions.

```
n := 0;
while prob(0.9) do
    n := n + 1;
    if n ≥ 10 then break
    else continue
od
```

There are multiple possible executions of the program, e.g., n could end up with 0, 3, or 10.

Principle

Probabilistic programs establish input/**output-distribution** relations. A probabilistic program can be modeled as a function in $X \to \mathcal{D}(X)$, where X is a program state space and $\mathcal{D}(X)$ consists of probability distributions over X.

Nondeterminism

Sources

- Agents for Markov decisions processes (MDPs)
- Abstraction and refinement on programs

A Common Resolution

A nondeterministic function f from X to Y is a set-valued function that maps an input to a collection of outputs, i.e.,

$$f \in X \to \wp(Y).$$

Nondeterminism in Probabilistic Programming

A nondeterministic function f from X to $\mathcal{D}(X)$ should have the signature

$$f \in X \to \wp(\mathcal{D}(X)),$$

where $\mathcal{D}(X)$ consists of probability distributions over X.

Nondeterminism

Sources

- Agents for Markov decisions processes (MDPs)
- Abstraction and refinement on programs

A Common Resolution

A nondeterministic function f from X to Y is a set-valued function that maps an input to a collection of outputs, i.e.,

$$f \in X \to \wp(Y).$$

Nondeterminism in Probabilistic Programming

A nondeterministic function f from X to $\mathcal{D}(X)$ should have the signature

$$f \in X \to \wp(\mathcal{D}(X)),$$

where $\mathcal{D}(X)$ consists of probability distributions over X.

Nondeterminism

Sources

- Agents for Markov decisions processes (MDPs)
- Abstraction and refinement on programs

A Common Resolution

A nondeterministic function f from X to Y is a set-valued function that maps an input to a collection of outputs, i.e.,

$$f \in X \to \wp(Y).$$

Nondeterminism in Probabilistic Programming

A nondeterministic function f from X to $\mathcal{D}(X)$ should have the signature

$$f \in X \to \wp(\mathcal{D}(X)),$$

where $\mathcal{D}(X)$ consists of probability distributions over X.

When to Resolve Nondeterminism?

X is a program state space. $\mathcal{D}(X)$ consists of probability distributions over X.

The Common Resolution: Input Prior to Nondeterminism

$$f \in X \to \wp(\mathcal{D}(X))$$

What about: Nondeterminism Prior to Input?

$$f \in \wp(X \to \mathcal{D}(X))$$

Intuition: A nondeterministic program is a specification that models a **collection** of deterministic refinements.

When to Resolve Nondeterminism?

X is a program state space. $\mathcal{D}(X)$ consists of probability distributions over X.

The Common Resolution: Input **Prior to** Nondeterminism

$$f \in X \to \wp(\mathcal{D}(X))$$

What about: Nondeterminism **Prior to** Input?

$$f \in \wp(X \to \mathcal{D}(X))$$

Intuition: A nondeterministic program is a specification that models a **collection** of deterministic refinements.

When to Resolve Nondeterminism?

X is a program state space. $\mathcal{D}(X)$ consists of probability distributions over X.

The Common Resolution: Input **Prior to** Nondeterminism

$$f \in X \to \wp(\mathcal{D}(X))$$

What about: Nondeterminism **Prior to** Input?

$$f \in \wp(X \to \mathcal{D}(X))$$

Intuition: A nondeterministic program is a specification that models a **collection** of deterministic refinements.

Nondeterminism-**First**: Nondeterminism Prior to Input

Example

Consider the following program P where $\star$ represents nondeterminism.

$$\textbf{if}\ \textbf{prob}(\star)\ \textbf{then}\ t := t+1\ \textbf{else}\ t := t-1\ \textbf{fi}$$

The Common Resolution

$$t = 1$$

$t' = 2$ w.p. 0.5	$t' = 2$ w.p. 0.8
$t' = 0$ w.p. 0.5	$t' = 0$ w.p. 0.2

$\star$ resolved **after** t is given

Nondeterminism-First

$t = 1$

$t' = 2$	w.p. 0.5
$t' = 0$	w.p. 0.5

$\star$ resolved as 0.5

$t = 1$

$t' = 2$	w.p. 0.8
$t' = 0$	w.p. 0.2

$\star$ resolved as 0.8

$\star$ resolved **before** t is given

Nondeterminism-**First**: Nondeterminism Prior to Input

Example

Consider the following program P where $\star$ represents nondeterminism.

$$\textbf{if } \textbf{prob}(\star) \textbf{ then } t := t + 1 \textbf{ else } t := t - 1 \textbf{ fi}$$

The Common Resolution

$$t = 1$$

$$\boxed{\begin{array}{ll} t' = 2 & \text{w.p. } 0.5 \\ t' = 0 & \text{w.p. } 0.5 \end{array}} \quad \boxed{\begin{array}{ll} t' = 2 & \text{w.p. } 0.8 \\ t' = 0 & \text{w.p. } 0.2 \end{array}}$$

$\star$ resolved **after** t is given

Nondeterminism-First

$$t = 1 \qquad\qquad t = 1$$

$$\boxed{\begin{array}{ll} t' = 2 & \text{w.p. } 0.5 \\ t' = 0 & \text{w.p. } 0.5 \end{array}} \quad \boxed{\begin{array}{ll} t' = 2 & \text{w.p. } 0.8 \\ t' = 0 & \text{w.p. } 0.2 \end{array}}$$

$\star$ resolved as 0.5 | $\star$ resolved as 0.8

$\star$ resolved **before** t is given

Nondeterminism-**First**: Nondeterminism Prior to Input

Example

Consider the following program P where $\star$ represents nondeterminism.

$$\textbf{if } \mathbf{prob}(\star) \textbf{ then } t \coloneqq t+1 \textbf{ else } t \coloneqq t-1 \textbf{ fi}$$

The Common Resolution

$$\begin{tikzcd} & t = 1 \arrow[dl] \arrow[dr] & \\ \boxed{\begin{array}{ll} t' = 2 & \text{w.p. } 0.5 \\ t' = 0 & \text{w.p. } 0.5 \end{array}} & & \boxed{\begin{array}{ll} t' = 2 & \text{w.p. } 0.8 \\ t' = 0 & \text{w.p. } 0.2 \end{array}} \end{tikzcd}$$

$\star$ resolved **after** t is given

Nondeterminism-First

$$\begin{tikzcd} t = 1 \arrow[d] \\ \boxed{\begin{array}{ll} t' = 2 & \text{w.p. } 0.5 \\ t' = 0 & \text{w.p. } 0.5 \end{array}} \end{tikzcd}$$

$\star$ resolved as 0.5

$$\begin{tikzcd} t = 1 \arrow[d] \\ \boxed{\begin{array}{ll} t' = 2 & \text{w.p. } 0.8 \\ t' = 0 & \text{w.p. } 0.2 \end{array}} \end{tikzcd}$$

$\star$ resolved as 0.8

$\star$ resolved **before** t is given

Nondeterminism-**First**: What's the Benefit?

Example

Consider the following program P where $\star$ represents nondeterminism.

$$\textbf{if}\ \textbf{prob}(\star)\ \textbf{then}\ t \coloneqq t + 1\ \textbf{else}\ t \coloneqq t - 1\ \textbf{fi}$$

Relational Reasoning about Refinements of a Program

- For all refinements P' of P, for all t_1, t_2, can we prove that $\mathbb{E}_{t_1' \sim P'(t_1), t_2' \sim P'(t_2)}[t_1' - t_2'] = t_1 - t_2$?
- For all refinements P' of P, for all t_1, t_2, does P' exhibit similar execution time on t_1 and t_2?

Nondeterminism-**First**: What's the Benefit?

Example

Consider the following program P where $\star$ represents nondeterminism.

$$\textbf{if } \textbf{prob}(\star) \textbf{ then } t := t + 1 \textbf{ else } t := t - 1 \textbf{ fi}$$

Relational Reasoning about Refinements of a Program

- For all refinements P' of P, for all t_1, t_2, can we prove that $\mathbb{E}_{t_1' \sim P'(t_1), t_2' \sim P'(t_2)}[t_1' - t_2'] = t_1 - t_2$?
- For all refinements P' of P, for all t_1, t_2, does P' exhibit similar execution time on t_1 and t_2?

Nondeterminism-**First**: What's the Benefit?

Example

Consider the following program P where $\star$ represents nondeterminism.

$$\textbf{if } \textbf{prob}(\star) \textbf{ then } t := t + 1 \textbf{ else } t := t - 1 \textbf{ fi}$$

Relational Reasoning about Refinements of a Program

- For all refinements P' of P, for all t_1, t_2, can we prove that $\mathbb{E}_{t_1' \sim P'(t_1), t_2' \sim P'(t_2)}[t_1' - t_2'] = t_1 - t_2$?
- For all refinements P' of P, for all t_1, t_2, does P' exhibit similar execution time on t_1 and t_2?

Contributions

- We develop a denotational semantics for **low-level** probabilistic programs with unstructured control-flow, general recursion, and nondeterminism.
- We study different resolutions for nondeterminism and propose a new model that involves nondeterminacy among **state transformers**.
- We devise an **algebraic** framework for denotational semantics, which can be instantiated with different resolutions for nondeterminism.

Outline

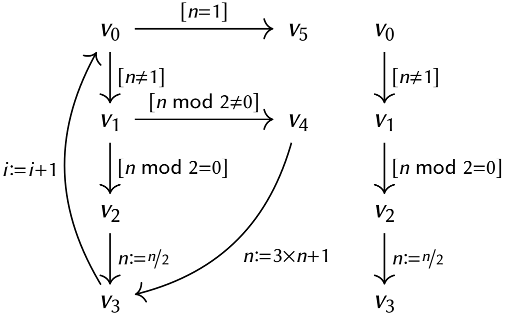
Motivation

Control-Flow Hyper-Graphs

Algebraic Denotational Semantics

Nondeterminism-First

Representation of Low-Level Probabilistic Programs



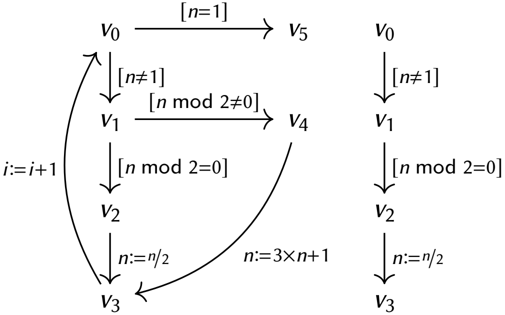
A standard CFG and an execution path

A tree-like **hyper-path**

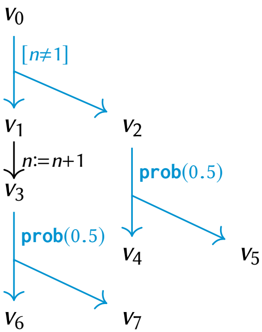
Principle

For probabilistic programs, execution paths are ***not* independent**. A formal semantics should reason about **distributions** over paths.

Representation of Low-Level Probabilistic Programs



A standard CFG and an execution path

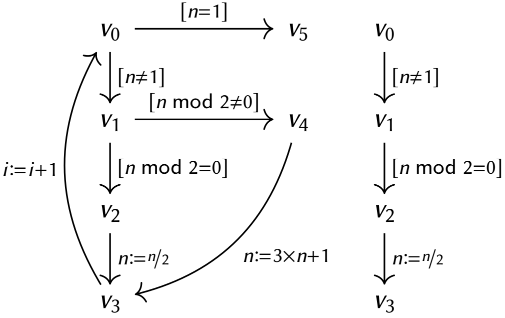


A tree-like **hyper-path**

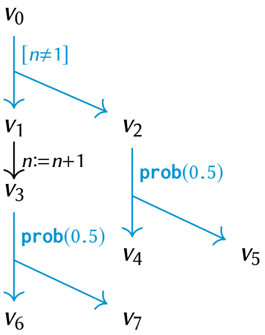
Principle

For probabilistic programs, execution paths are ***not independent***. A formal semantics should reason about **distributions** over paths.

Representation of Low-Level Probabilistic Programs



A standard CFG and an execution path



A tree-like **hyper-path**

Principle

For probabilistic programs, execution paths are ***not* independent**. A formal semantics should reason about **distributions** over paths.

Paths vs. Hyper-Paths

Example

$$\begin{array}{lll} \textbf{if } \star & \textbf{then} & \textbf{if } \textbf{prob}(0.5) \textbf{ then } t := 0 \textbf{ else } t := 1 \textbf{ fi} \\ & \textbf{else} & \textbf{if } \textbf{prob}(0.8) \textbf{ then } t := 0 \textbf{ else } t := 1 \textbf{ fi} \quad \textbf{fi} \end{array}$$

Paths Annotated with Probabilities

$$\begin{array}{cccc} \bullet & \bullet & \bullet & \bullet \\ \downarrow{\scriptstyle 0.5} & \downarrow{\scriptstyle 0.5} & \downarrow{\scriptstyle 0.8} & \downarrow{\scriptstyle 0.2} \\ t' = 0 & t' = 1 & t' = 0 & t' = 1 \end{array}$$

Hyper-Paths, each of which stands for a **distribution**

$$\begin{tikzcd} \bullet \arrow[d, "0.5"] \arrow[dr, "0.5"] & & \bullet \arrow[d, "0.8"] \arrow[dr, "0.2"] & \\ t' = 0 & t' = 1 & t' = 0 & t' = 1 \end{tikzcd}$$

Paths vs. Hyper-Paths

Example

$$
\begin{array}{lll}
\texttt{if}\ \star & \texttt{then} & \texttt{if}\ \texttt{prob}(0.5)\ \texttt{then}\ t := 0\ \texttt{else}\ t := 1\ \texttt{fi} \\
 & \texttt{else} & \texttt{if}\ \texttt{prob}(0.8)\ \texttt{then}\ t := 0\ \texttt{else}\ t := 1\ \texttt{fi} \quad \texttt{fi}
\end{array}
$$

Paths Annotated with Probabilities

$$
\begin{tikzcd}
\bullet \arrow[d, "0.5"] & \bullet \arrow[d, "0.5"] & \bullet \arrow[d, "0.8"] & \bullet \arrow[d, "0.2"] \\
t' = 0 & t' = 1 & t' = 0 & t' = 1
\end{tikzcd}
$$

Hyper-Paths, each of which stands for a **distribution**

$$
\begin{tikzcd}
\bullet \arrow[d, "0.5"] \arrow[dr, "0.5"] & & \bullet \arrow[d, "0.8"] \arrow[dr, "0.2"] & \\
t' = 0 & t' = 1 & t' = 0 & t' = 1
\end{tikzcd}
$$

Paths vs. Hyper-Paths

Example

$$\textbf{if}\ \star\ \ \textbf{then}\ \ \textbf{if}\ \textbf{prob}(0.5)\ \textbf{then}\ t := 0\ \textbf{else}\ t := 1\ \textbf{fi}$$
$$\textbf{else}\ \ \textbf{if}\ \textbf{prob}(0.8)\ \textbf{then}\ t := 0\ \textbf{else}\ t := 1\ \textbf{fi}\ \ \textbf{fi}$$

Paths Annotated with Probabilities

$$\begin{array}{cccc} \bullet & \bullet & \bullet & \bullet \\ \downarrow 0.5 & \downarrow 0.5 & \downarrow 0.8 & \downarrow 0.2 \\ t' = 0 & t' = 1 & t' = 0 & t' = 1 \end{array}$$

Hyper-Paths, each of which stands for a **distribution**

$$\begin{tikzcd} \bullet \arrow[d, "0.5"] \arrow[dr, "0.5"] & & \bullet \arrow[d, "0.8"] \arrow[dr, "0.2"] & \\ t' = 0 & t' = 1 & t' = 0 & t' = 1 \end{tikzcd}$$

Control-Flow Hyper-Graphs

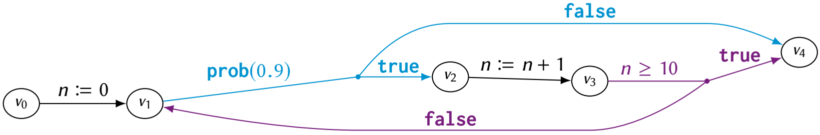
- Hyper-graphs are directed graphs with **hyper-edges** that could have multiple destinations. **Hyper-paths** are made up of hyper-egdes.
- The following hyper-graph

represents the control-flow of the example program

```
n := 0;
while prob(0.9) do
    n := n + 1;
    if n ≥ 10 then break
    else continue
od
```

Control-Flow Hyper-Graphs

- Hyper-graphs are directed graphs with **hyper-edges** that could have multiple destinations. **Hyper-paths** are made up of hyper-egdes.
- The following hyper-graph



represents the control-flow of the example program

$$
\begin{array}{l}
n := 0; \\
\textbf{while } \texttt{prob}(0.9) \textbf{ do} \\
\quad n := n + 1; \\
\quad \textbf{if } n \geq 10 \textbf{ then break} \\
\quad \textbf{else continue} \\
\textbf{od}
\end{array}
$$

Outline

An Algebraic Denotational Semantics

Goal

Develop a denotational semantics that can be instantiated with different resolutions of nondeterminism.

An Algebraic Approach

- Perform reasoning in some **abstract** space of program states and state transformers.
- The state transformers should obey some algebraic **laws**.
- For example, the command **`skip`** should be interpreted as an **identity** element for sequencing in the algebra of transformers.

Outcome

The semantics is a good fit for developing static analyses (Wang, Hoffmann, and Reps 2018).

An Algebraic Denotational Semantics

Goal

Develop a denotational semantics that can be instantiated with different resolutions of nondeterminism.

An Algebraic Approach

- Perform reasoning in some **abstract** space of program states and state transformers.
- The state transformers should obey some algebraic **laws**.
- For example, the command `skip` should be interpreted as an **identity** element for sequencing in the algebra of transformers.

Outcome

The semantics is a good fit for developing static analyses (Wang, Hoffmann, and Reps 2018).

An Algebraic Denotational Semantics

Goal

Develop a denotational semantics that can be instantiated with different resolutions of nondeterminism.

An Algebraic Approach

- Perform reasoning in some **abstract** space of program states and state transformers.
- The state transformers should obey some algebraic **laws**.
- For example, the command **`skip`** should be interpreted as an **identity** element for sequencing in the algebra of transformers.

Outcome

The semantics is a good fit for developing static analyses (Wang, Hoffmann, and Reps 2018).

The Algebra

Actions

`skip`

$x := x + 5$

$k \sim \text{Binomial}(10, 0.5)$

$\ldots$

$\xmapsto{\textsc{Semantic Function}}$

State Transformers $\mathcal{M}$

equipped with

sequencing $\otimes$

conditional-choice ${}_{\varphi}\diamondsuit$

nondeterministic-choice $\uplus$

$$\left\langle \mathcal{M}, \sqsubseteq, \otimes, {}_{\varphi}\diamondsuit, \uplus, \bot, 1 \right\rangle$$

- $\langle \mathcal{M}, \sqsubseteq \rangle$ forms a directed complete partial order (dcpo) with $\bot$ as its least element.
- $\langle \mathcal{M}, \otimes, 1 \rangle$ forms a monoid.
- Nondeterministic-choice $\uplus$ is a semilattice operation.

The Algebra

Actions		**State Transformers** M
`skip`		equipped with
$x := x + 5$	$\xmapsto{\textsc{Semantic Function}}$	*sequencing* $\otimes$
$k \sim \text{Binomial}(10, 0.5)$		*conditional-choice* ${}_{\varphi}\diamond$
$\cdots$		*nondeterministic-choice* $\uplus$

$$\left\langle M, \sqsubseteq, \otimes, {}_{\varphi}\diamond, \uplus, \perp, 1 \right\rangle$$

- $\langle M, \sqsubseteq \rangle$ forms a directed complete partial order (dcpo) with $\perp$ as its least element.
- $\langle M, \otimes, 1 \rangle$ forms a monoid.
- Nondeterministic-choice $\uplus$ is a semilattice operation.

The Algebra

Actions		State Transformers M
skip	SEMANTIC FUNCTION $\longmapsto$	equipped with
$x := x + 5$		*sequencing* $\otimes$
$k \sim \text{Binomial}(10, 0.5)$		*conditional-choice* ${}_{\varphi}\diamond$
$\cdots$		*nondeterministic-choice* $\uplus$

$$\left\langle M, \sqsubseteq, \otimes, {}_{\varphi}\diamond, \uplus, \bot, 1 \right\rangle$$

- $\langle M, \sqsubseteq \rangle$ forms a directed complete partial order (dcpo) with $\bot$ as its least element.
- $\langle M, \otimes, 1 \rangle$ forms a monoid.
- Nondeterministic-choice $\uplus$ is a semilattice operation.

The Algebra

Actions		State Transformers M
skip		equipped with
$x := x + 5$	$\xmapsto{\text{Semantic Function}}$	*sequencing* $\otimes$
$k \sim \text{Binomial}(10, 0.5)$		*conditional-choice* ${}_{\varphi}\diamond$
$\cdots$		*nondeterministic-choice* $\uplus$

$$\left\langle M, \sqsubseteq, \otimes, {}_{\varphi}\diamond, \uplus, \perp, 1 \right\rangle$$

- $\langle M, \sqsubseteq \rangle$ forms a directed complete partial order (dcpo) with $\perp$ as its least element.
- $\langle M, \otimes, 1 \rangle$ forms a monoid.
- Nondeterministic-choice $\uplus$ is a semilattice operation.

The Algebra

Actions		State Transformers M
`skip`		equipped with
$x := x + 5$	$\xmapsto{\text{Semantic Function}}$	*sequencing* $\otimes$
$k \sim \text{Binomial}(10, 0.5)$		*conditional-choice* ${}_{\varphi}\!\diamond$
$\cdots$		*nondeterministic-choice* $\uplus$

$$\left\langle M, \sqsubseteq, \otimes, {}_{\varphi}\!\diamond, \uplus, \bot, 1 \right\rangle$$

- $\langle M, \sqsubseteq\rangle$ forms a directed complete partial order (dcpo) with $\bot$ as its least element.
- $\langle M, \otimes, 1\rangle$ forms a monoid.
- Nondeterministic-choice $\uplus$ is a semilattice operation.

Fixpoint Semantics for Hyper-Graphs

Principle

The semantics of a node in the control-flow hyper-graph is a summary of computation that **continues from** that node.

Recall the control-flow hyper-graph below.

```
n := 0;
while prob(0.9) do
  n := n + 1;
  if n ≥ 10 then break
  else continue
od
```


Semantics is defined as the **least** solution to the following equation system

$$\mathcal{S}(v_0) = seq[n := 0](\mathcal{S}(v_1)) \qquad \mathcal{S}(v_2) = seq[n := n + 1](\mathcal{S}(v_3)) \qquad \mathcal{S}(v_4) = 1$$
$$\mathcal{S}(v_1) = prob[0.9](\mathcal{S}(v_2), \mathcal{S}(v_4)) \qquad \mathcal{S}(v_3) = cond[n \geq 10](\mathcal{S}(v_4), \mathcal{S}(v_1))$$

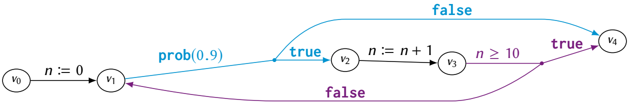
Fixpoint Semantics for Hyper-Graphs

Principle

The semantics of a node in the control-flow hyper-graph is a summary of computation that **continues from** that node.

Recall the control-flow hyper-graph below.

$n := 0;$
while **prob**(0.9) **do**
  $n := n + 1;$
  if $n \geq 10$ **then break**
  else continue
od



Semantics is defined as the **least** solution to the following equation system

$$\mathcal{S}(v_0) = seq[n := 0](\mathcal{S}(v_1)) \qquad \mathcal{S}(v_2) = seq[n := n + 1](\mathcal{S}(v_3)) \qquad \mathcal{S}(v_4) = 1$$

$$\mathcal{S}(v_1) = prob[0.9](\mathcal{S}(v_2), \mathcal{S}(v_4)) \qquad \mathcal{S}(v_3) = cond[n \geq 10](\mathcal{S}(v_4), \mathcal{S}(v_1))$$

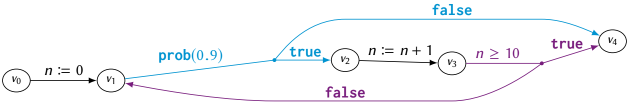
Fixpoint Semantics for Hyper-Graphs

Principle

The semantics of a node in the control-flow hyper-graph is a summary of computation that **continues from** that node.

Recall the control-flow hyper-graph below.

```
n := 0;
while prob(0.9) do
  n := n + 1;
  if n ≥ 10 then break
  else continue
od
```



Semantics is defined as the **least** solution to the following equation system

$$\mathcal{S}(v_0) = seq[n := 0](\mathcal{S}(v_1)) \qquad \mathcal{S}(v_2) = seq[n := n+1](\mathcal{S}(v_3)) \qquad \mathcal{S}(v_4) = 1$$

$$\mathcal{S}(v_1) = prob[0.9](\mathcal{S}(v_2), \mathcal{S}(v_4)) \qquad \mathcal{S}(v_3) = cond[n \geq 10](\mathcal{S}(v_4), \mathcal{S}(v_1))$$

Fixpoint Semantics for Hyper-Graphs

Semantics is defined as the **least** solution to the following equation system

$$\mathcal{S}(v_0) = \mathit{seq}[n := 0](\mathcal{S}(v_1)) \qquad \mathcal{S}(v_2) = \mathit{seq}[n := n+1](\mathcal{S}(v_3)) \qquad \mathcal{S}(v_4) = 1$$
$$\mathcal{S}(v_1) = \mathit{prob}[0.9](\mathcal{S}(v_2), \mathcal{S}(v_4)) \qquad \mathcal{S}(v_3) = \mathit{cond}[n \geq 10](\mathcal{S}(v_4), \mathcal{S}(v_1))$$

Use the algebra to reinterpret the equation system

$$\mathcal{S}(v_0) = [\![n := 0]\!] \otimes \mathcal{S}(v_1) \qquad \mathcal{S}(v_2) = [\![n := n+1]\!] \otimes \mathcal{S}(v_3) \qquad \mathcal{S}(v_4) = 1$$
$$\mathcal{S}(v_1) = \mathcal{S}(v_2) \,{}_{\mathbf{prob}(0.9)}\!\Diamond\, \mathcal{S}(v_4) \qquad \mathcal{S}(v_3) = \mathcal{S}(v_4) \,{}_{n \geq 10}\!\Diamond\, \mathcal{S}(v_1)$$

where $[\![\cdot]\!]$ maps actions into state transformers in M.

Fixpoint Semantics for Hyper-Graphs

Semantics is defined as the **least** solution to the following equation system

$$\mathcal{S}(v_0) = \mathit{seq}[n := 0](\mathcal{S}(v_1)) \qquad \mathcal{S}(v_2) = \mathit{seq}[n := n+1](\mathcal{S}(v_3)) \qquad \mathcal{S}(v_4) = 1$$

$$\mathcal{S}(v_1) = \mathit{prob}[0.9](\mathcal{S}(v_2), \mathcal{S}(v_4)) \qquad \mathcal{S}(v_3) = \mathit{cond}[n \geq 10](\mathcal{S}(v_4), \mathcal{S}(v_1))$$

Use the algebra to reinterpret the equation system

$$\mathcal{S}(v_0) = [\![n := 0]\!] \otimes \mathcal{S}(v_1) \qquad \mathcal{S}(v_2) = [\![n := n+1]\!] \otimes \mathcal{S}(v_3) \qquad \mathcal{S}(v_4) = 1$$

$$\mathcal{S}(v_1) = \mathcal{S}(v_2) \;{}_{\mathbf{prob}(0.9)}\Diamond\; \mathcal{S}(v_4) \qquad \mathcal{S}(v_3) = \mathcal{S}(v_4) \;{}_{n \geq 10}\Diamond\; \mathcal{S}(v_1)$$

where $[\![\cdot]\!]$ maps actions into state transformers in M.

A Denotational Semantics without Nondeterminism

- $X \stackrel{\text{def}}{=} \mathsf{Var} \rightharpoonup_{\text{fin}} \mathbb{Q}$ and $M \stackrel{\text{def}}{=} X \to \underline{\mathcal{D}}(X)$.
- $\underline{\mathcal{D}}(X)$ stands for **sub-probability distributions** on X, i.e., $\Delta \in \underline{\mathcal{D}}(X)$ iff $\Delta : X \to [0,1]$ and $\sum_{x \in X} \Delta(x) \le 1$.
- For actions act, we have $[\![\mathsf{act}]\!] \in M$.
- For conditions φ, we have $[\![\varphi]\!] : X \to [0,1]$, e.g., $[\![\mathbf{prob}(p)]\!] \stackrel{\text{def}}{=} \lambda\_.p$.
- $f \sqsubseteq g \stackrel{\text{def}}{=} \forall x \in X : \forall x' \in X : f(x)(x') \le g(x)(x')$.
- $f \otimes g \stackrel{\text{def}}{=} \lambda x.\lambda x''. \sum_{x' \in X} f(x,x') \cdot g(x', x'')$.
- $f \;{}_{\varphi}\!\diamond\; g \stackrel{\text{def}}{=} \lambda x.\lambda x'. [\![\varphi]\!](x) \cdot f(x)(x') + (1 - [\![\varphi]\!](x)) \cdot g(x)(x')$.
- $\bot \stackrel{\text{def}}{=} \lambda\_.\lambda\_.0$.
- $1 \stackrel{\text{def}}{=} \lambda x.\delta(x)$ where the **point distribution** $\delta(x) \stackrel{\text{def}}{=} \lambda x'.[x = x']$.

A Denotational Semantics without Nondeterminism

- $X \stackrel{\text{def}}{=} \mathrm{Var} \rightarrow_{\mathrm{fin}} \mathbb{Q}$ and $M \stackrel{\text{def}}{=} X \rightarrow \underline{\mathcal{D}}(X)$.
- $\underline{\mathcal{D}}(X)$ stands for **sub-probability distributions** on X, i.e., $\Delta \in \underline{\mathcal{D}}(X)$ iff $\Delta : X \rightarrow [0, 1]$ and $\sum_{x \in X} \Delta(x) \leq 1$.
- For actions act, we have $[\![\mathrm{act}]\!] \in M$.
- For conditions φ, we have $[\![\varphi]\!] : X \rightarrow [0, 1]$, e.g., $[\![\mathbf{prob}(p)]\!] \stackrel{\text{def}}{=} \lambda\_.p$.
- $f \sqsubseteq g \stackrel{\text{def}}{=} \forall x \in X : \forall x' \in X : f(x)(x') \leq g(x)(x')$.
- $f \otimes g \stackrel{\text{def}}{=} \lambda x.\lambda x''. \sum_{x' \in X} f(x, x') \cdot g(x', x'')$.
- $f \;{}_{\varphi}\!\diamond\; g \stackrel{\text{def}}{=} \lambda x.\lambda x'. [\![\varphi]\!](x) \cdot f(x)(x') + (1 - [\![\varphi]\!](x)) \cdot g(x)(x')$.
- $\perp \stackrel{\text{def}}{=} \lambda\_.\lambda\_.0$.
- $1 \stackrel{\text{def}}{=} \lambda x.\delta(x)$ where the **point distribution** $\delta(x) \stackrel{\text{def}}{=} \lambda x'.[x = x']$.

A Denotational Semantics without Nondeterminism

- $X \stackrel{\text{def}}{=} \mathrm{Var} \to_{\text{fin}} \mathbb{Q}$ and $M \stackrel{\text{def}}{=} X \to \underline{\mathcal{D}}(X)$.
- $\underline{\mathcal{D}}(X)$ stands for **sub-probability distributions** on X, i.e., $\Delta \in \underline{\mathcal{D}}(X)$ iff $\Delta : X \to [0, 1]$ and $\sum_{x \in X} \Delta(x) \leq 1$.
- For actions act, we have $[\![\mathrm{act}]\!] \in M$.
- For conditions φ, we have $[\![\varphi]\!] : X \to [0, 1]$, e.g., $[\![\mathbf{prob}(p)]\!] \stackrel{\text{def}}{=} \lambda\_.p$.
- $f \sqsubseteq g \stackrel{\text{def}}{=} \forall x \in X : \forall x' \in X : f(x)(x') \leq g(x)(x')$.
- $f \otimes g \stackrel{\text{def}}{=} \lambda x.\lambda x''. \sum_{x' \in X} f(x, x') \cdot g(x', x'')$.
- $f\ {}_{\varphi}\!\diamond\ g \stackrel{\text{def}}{=} \lambda x.\lambda x'. [\![\varphi]\!](x) \cdot f(x)(x') + (1 - [\![\varphi]\!](x)) \cdot g(x)(x')$.
- $\perp \stackrel{\text{def}}{=} \lambda\_.\lambda\_.0$.
- $1 \stackrel{\text{def}}{=} \lambda x.\delta(x)$ where the **point distribution** $\delta(x) \stackrel{\text{def}}{=} \lambda x'.[x = x']$.

A Denotational Semantics without Nondeterminism

- $X \overset{\text{def}}{=} \mathrm{Var} \to_{\text{fin}} \mathbb{Q}$ and $M \overset{\text{def}}{=} X \to \underline{\mathcal{D}}(X)$.
- $\underline{\mathcal{D}}(X)$ stands for **sub-probability distributions** on X, i.e., $\Delta \in \underline{\mathcal{D}}(X)$ iff $\Delta : X \to [0,1]$ and $\sum_{x \in X} \Delta(x) \leq 1$.
- For actions act, we have $[\![\mathrm{act}]\!] \in M$.
- For conditions φ, we have $[\![\varphi]\!] : X \to [0,1]$, e.g., $[\![\mathbf{prob}(p)]\!] \overset{\text{def}}{=} \lambda\_.p$.
- $f \sqsubseteq g \overset{\text{def}}{=} \forall x \in X : \forall x' \in X : f(x)(x') \leq g(x)(x')$.
- $f \otimes g \overset{\text{def}}{=} \lambda x.\lambda x''. \sum_{x' \in X} f(x, x') \cdot g(x', x'')$.
- $f \; {}_{\varphi}\!\diamond \; g \overset{\text{def}}{=} \lambda x.\lambda x'. [\![\varphi]\!](x) \cdot f(x)(x') + (1 - [\![\varphi]\!](x)) \cdot g(x)(x')$.
- $\perp \overset{\text{def}}{=} \lambda\_.\lambda\_.0$.
- $1 \overset{\text{def}}{=} \lambda x.\delta(x)$ where the **point distribution** $\delta(x) \overset{\text{def}}{=} \lambda x'.[x = x']$.

A Denotational Semantics without Nondeterminism

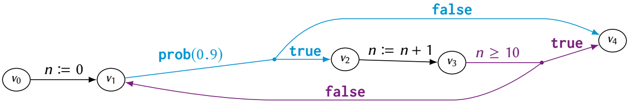
- $X \stackrel{\text{def}}{=} \mathsf{Var} \rightharpoonup_{\text{fin}} \mathbb{Q}$ and $M \stackrel{\text{def}}{=} X \to \underline{\mathcal{D}}(X)$.
- $\underline{\mathcal{D}}(X)$ stands for **sub-probability distributions** on X, i.e., $\Delta \in \underline{\mathcal{D}}(X)$ iff $\Delta : X \to [0,1]$ and $\sum_{x \in X} \Delta(x) \leq 1$.
- For actions act, we have $[\![\mathsf{act}]\!] \in M$.
- For conditions φ, we have $[\![\varphi]\!] : X \to [0,1]$, e.g., $[\![\mathbf{prob}(p)]\!] \stackrel{\text{def}}{=} \lambda\_.p$.
- $f \sqsubseteq g \stackrel{\text{def}}{=} \forall x \in X : \forall x' \in X : f(x)(x') \leq g(x)(x')$.
- $f \otimes g \stackrel{\text{def}}{=} \lambda x.\lambda x''. \sum_{x' \in X} f(x,x') \cdot g(x',x'')$.
- $f \;{}_{\varphi}\!\diamond\; g \stackrel{\text{def}}{=} \lambda x.\lambda x'. [\![\varphi]\!](x) \cdot f(x)(x') + (1 - [\![\varphi]\!](x)) \cdot g(x)(x')$.
- $\perp \stackrel{\text{def}}{=} \lambda\_.\lambda\_.0$.
- $1 \stackrel{\text{def}}{=} \lambda x.\delta(x)$ where the **point distribution** $\delta(x) \stackrel{\text{def}}{=} \lambda x'.[x = x']$.

A Denotational Semantics without Nondeterminism

- $X \stackrel{\text{def}}{=} \mathrm{Var} \rightharpoonup_{\text{fin}} \mathbb{Q}$ and $M \stackrel{\text{def}}{=} X \to \underline{\mathcal{D}}(X)$.
- $\underline{\mathcal{D}}(X)$ stands for **sub-probability distributions** on X, i.e., $\Delta \in \underline{\mathcal{D}}(X)$ iff $\Delta : X \to [0,1]$ and $\sum_{x \in X} \Delta(x) \leq 1$.
- For actions act, we have $[\![\mathrm{act}]\!] \in M$.
- For conditions φ, we have $[\![\varphi]\!] : X \to [0,1]$, e.g., $[\![\mathbf{prob}(p)]\!] \stackrel{\text{def}}{=} \lambda\_.p$.
- $f \sqsubseteq g \stackrel{\text{def}}{=} \forall x \in X : \forall x' \in X : f(x)(x') \leq g(x)(x')$.
- $f \otimes g \stackrel{\text{def}}{=} \lambda x.\lambda x''. \sum_{x' \in X} f(x,x') \cdot g(x',x'')$.
- $f \; {}_{\varphi}\!\diamond \; g \stackrel{\text{def}}{=} \lambda x.\lambda x'. [\![\varphi]\!](x) \cdot f(x)(x') + (1 - [\![\varphi]\!](x)) \cdot g(x)(x')$.
- $\bot \stackrel{\text{def}}{=} \lambda\_.\lambda\_.0$.
- $1 \stackrel{\text{def}}{=} \lambda x.\delta(x)$ where the **point distribution** $\delta(x) \stackrel{\text{def}}{=} \lambda x'.[x = x']$.

A Denotational Semantics without Nondeterminism

$n := 0;$
while **prob**(0.9) **do**
  $n := n + 1;$
  if $n \geq 10$ **then break**
  else continue
od



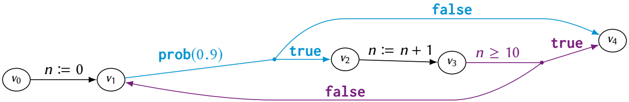
Because Var $= \{n\}$ is a singleton, we present the semantics as if $X \stackrel{\text{def}}{=} \mathbb{Z}$.

$$\mathcal{S}(v_0) = \lambda\_.\ \sum_{k=0}^{9} (0.1 \times 0.9^k) \cdot \delta(k) + 0.3486784401 \cdot \delta(10)$$

$\delta(n_0)$ represents a point distribution at n_0.

A Denotational Semantics without Nondeterminism

$n := 0$;
while prob(0.9) **do**
  $n := n + 1$;
  if $n \geq 10$ **then break**
  else continue
od



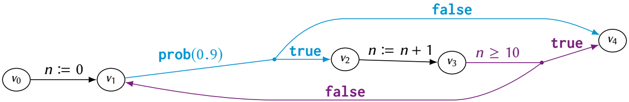
Because Var $= \{n\}$ is a singleton, we present the semantics as if $X \stackrel{\text{def}}{=} \mathbb{Z}$.

$$\mathcal{S}(v_0) = \lambda\_.\ \sum_{k=0}^{9} (0.1 \times 0.9^k) \cdot \delta(k) + 0.3486784401 \cdot \delta(10)$$

$\delta(n_0)$ represents a point distribution at n_0.

A Denotational Semantics without Nondeterminism

```
n := 0;
while prob(0.9) do
   n := n + 1;
   if n ≥ 10 then break
   else continue
od
```



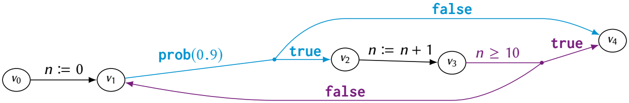
Because Var $= \{n\}$ is a singleton, we present the semantics as if $X \stackrel{\text{def}}{=} \mathbb{Z}$.

$$\mathcal{S}(v_0) = \lambda\_.\ \sum_{k=0}^{9} (0.1 \times 0.9^k) \cdot \delta(k) + 0.3486784401 \cdot \delta(10)$$

$\delta(n_0)$ represents a point distribution at n_0.

A Denotational Semantics without Nondeterminism

```
n := 0;
while prob(0.9) do
   n := n + 1;
   if n ≥ 10 then break
   else continue
od
```



Recall the equation

$$\mathcal{S}(v_0) = [\![n := 0]\!] \otimes \mathcal{S}(v_1)$$

Obtain $\mathcal{S}(v_0)$ from $\mathcal{S}(v_1)$

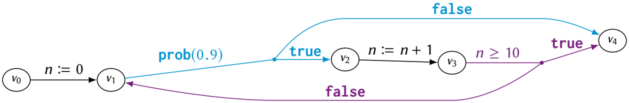
$$\mathcal{S}(v_0) = \lambda\_.\sum_{k=0}^{9}(0.1 \times 0.9^k) \cdot \delta(k) + 0.3486784401 \cdot \delta(10)$$

$$[\![n := 0]\!] = \lambda\_.\delta(0)$$

$$\mathcal{S}(v_1) = \lambda n.[n \geq 9] \cdot (0.1 \cdot \delta(n) + 0.9 \cdot \delta(n+1)) + [n < 9] \cdot \left(\sum_{k=n}^{\infty}(0.1 \times 0.9^{k-n}) \cdot \delta(\min\{k, 10\})\right)$$

A Denotational Semantics without Nondeterminism

```
n := 0;
while prob(0.9) do
  n := n + 1;
  if n ≥ 10 then break
  else continue
od
```



Recall the equation

$$\mathcal{S}(v_0) = [\![n := 0]\!] \otimes \mathcal{S}(v_1)$$

Obtain $\mathcal{S}(v_0)$ from $\mathcal{S}(v_1)$

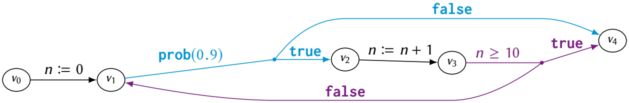
$$\mathcal{S}(v_0) = \lambda\_.\sum_{k=0}^{9}(0.1 \times 0.9^k) \cdot \delta(k) + 0.3486784401 \cdot \delta(10)$$

$$[\![n := 0]\!] = \lambda\_.\delta(0)$$

$$\mathcal{S}(v_1) = \lambda n.[n \geq 9] \cdot (0.1 \cdot \delta(n) + 0.9 \cdot \delta(n+1)) + [n < 9] \cdot \left(\sum_{k=n}^{\infty}(0.1 \times 0.9^{k-n}) \cdot \delta(\min\{k, 10\}) \right)$$

A Denotational Semantics without Nondeterminism

$n := 0;$
while prob(0.9) **do**
  $n := n + 1;$
  if $n \geq 10$ **then break**
  else continue
od



Recall the equation

$$\mathcal{S}(v_0) = [\![n := 0]\!] \otimes \mathcal{S}(v_1)$$

Obtain $\mathcal{S}(v_0)$ from $\mathcal{S}(v_1)$

$$\mathcal{S}(v_0) = \lambda\_.\sum_{k=0}^{9}(0.1 \times 0.9^k) \cdot \delta(k) + 0.3486784401 \cdot \delta(10)$$

$$[\![n := 0]\!] = \lambda\_.\delta(0)$$

$$\mathcal{S}(v_1) = \lambda n.[n \geq 9] \cdot (0.1 \cdot \delta(n) + 0.9 \cdot \delta(n+1)) + [n < 9] \cdot \left(\sum_{k=n}^{\infty}(0.1 \times 0.9^{k-n}) \cdot \delta(\min\{k, 10\})\right)$$

Outline

Motivation

Control-Flow Hyper-Graphs

Algebraic Denotational Semantics

Nondeterminism-First

Sub-Probability Kernels

Definition

A function $\kappa : X \to \underline{\mathcal{D}}(X)$ is called a **sub-probability kernel**. The set of kernels is denoted by $\underline{\mathcal{K}}(X)$.

Goal

The common resolution for nondeterminism admits the following signature

$$X \to \wp(\underline{\mathcal{D}}(X)),$$

while our **nondeterminism-first** model should have the following signature

$$\wp(X \to \underline{\mathcal{D}}(X)) \equiv \wp(\underline{\mathcal{K}}(X)).$$

Sub-Probability Kernels

Definition

A function $\kappa : X \to \underline{\mathcal{D}}(X)$ is called a **sub-probability kernel**. The set of kernels is denoted by $\underline{\mathcal{K}}(X)$.

Goal

The common resolution for nondeterminism admits the following signature

$$X \to \wp(\underline{\mathcal{D}}(X)),$$

while our **nondeterminism-first** model should have the following signature

$$\wp(X \to \underline{\mathcal{D}}(X)) \equiv \wp(\underline{\mathcal{K}}(X)).$$

Reasoning with Nondeterminism-First

Example

Recall the following nondeterministic program P

$$\textbf{if } \texttt{prob}(\star) \textbf{ then } t := t + 1 \textbf{ else } t := t - 1 \textbf{ fi}$$

Then the common resolution for nondeterminism derives

$$\lambda t.\{r \cdot \delta(t + 1) + (1 - r) \cdot \delta(t - 1) \mid r \in [0, 1]\},$$

but the nondeterminism-first model leads to

$$\{\lambda t.r \cdot \delta(t + 1) + (1 - r) \cdot \delta(t - 1) \mid r \in [0, 1]\}.$$

With the new model, we can prove that for every refinement P' with $\star$ resolved as $r \in [0, 1]$, for all t_1, t_2, we have

$$\begin{aligned}\mathbb{E}_{t_1' \sim P'(t_1), t_2' \sim P'(t_2)}[t_1' - t_2'] &= \mathbb{E}_{t_1' \sim P'(t_1)}[t_1'] - \mathbb{E}_{t_2' \sim P'(t_2)}[t_2'] \\ &= (r(t_1 + 1) + (1 - r)(t_1 - 1)) - (r(t_2 + 1) + (1 - r)(t_2 - 1)) \\ &= t_1 - t_2\end{aligned}$$

Reasoning with Nondeterminism-First

Example

Recall the following nondeterministic program P

$$\textbf{if } \texttt{prob}(\star) \textbf{ then } t \coloneqq t+1 \textbf{ else } t \coloneqq t-1 \textbf{ fi}$$

Then the common resolution for nondeterminism derives

$$\lambda t.\{r \cdot \delta(t+1) + (1-r) \cdot \delta(t-1) \mid r \in [0,1]\},$$

but the nondeterminism-first model leads to

$$\{\lambda t.r \cdot \delta(t+1) + (1-r) \cdot \delta(t-1) \mid r \in [0,1]\}.$$

With the new model, we can prove that for every refinement P' with $\star$ resolved as $r \in [0,1]$, for all t_1, t_2, we have

$$\begin{aligned}
\mathbb{E}_{t_1' \sim P'(t_1), t_2' \sim P'(t_2)}[t_1' - t_2'] &= \mathbb{E}_{t_1' \sim P'(t_1)}[t_1'] - \mathbb{E}_{t_2' \sim P'(t_2)}[t_2'] \\
&= (r(t_1+1) + (1-r)(t_1-1)) - (r(t_2+1) + (1-r)(t_2-1)) \\
&= t_1 - t_2
\end{aligned}$$

A Powerdomain for Nondeterminism-First

Necessary Conditions

We need to identify a subset $\mathcal{A}$ of $\wp(\underline{\mathcal{K}}(X))$ as the collection of admissible semantic objects.

- $\mathcal{A}$ admits a semilattice operation $\uplus$ (used as **nondeterministic-choice**), s.t. for all $A \in \mathcal{A}$, $A \uplus A = A$.
- $\mathcal{A}$ is equipped with a conditional-choice operation ${}_{\phi}\Diamond$ where $\phi : X \longrightarrow [0,1]$ represents a Boolean-valued random variable.
- For all $A_1, A_2 \in \mathcal{A}$ and $\phi : X \longrightarrow [0,1]$, if $\kappa_1 \in A_1$ and $\kappa_2 \in A_2$, then $\kappa_1 \,{}_{\phi}\Diamond\, \kappa_2$ should be in $A_1 \uplus A_2$.

A Convexity-Like Condition

For all $A \in \mathcal{A}$, we have $A \uplus A = A$, therefore we should also have $\forall \phi \in X \longrightarrow [0,1] : \forall \kappa_1, \kappa_2 \in A : \kappa_1 \,{}_{\phi}\Diamond\, \kappa_2 \in A$.

A Powerdomain for Nondeterminism-First

Necessary Conditions

We need to identify a subset $\mathcal{A}$ of $\wp(\underline{\mathcal{K}}(X))$ as the collection of admissible semantic objects.

- $\mathcal{A}$ admits a semilattice operation $\uplus$ (used as **nondeterministic-choice**), s.t. for all $A \in \mathcal{A}$, $A \uplus A = A$.
- $\mathcal{A}$ is equipped with a conditional-choice operation ${}_{\phi}\!\diamond$ where $\phi : X \to [0, 1]$ represents a Boolean-valued random variable.
- For all $A_1, A_2 \in \mathcal{A}$ and $\phi : X \to [0, 1]$, if $\kappa_1 \in A_1$ and $\kappa_2 \in A_2$, then $\kappa_1 \, {}_{\phi}\!\diamond \, \kappa_2$ should be in $A_1 \uplus A_2$.

A Convexity-Like Condition

For all $A \in \mathcal{A}$, we have $A \uplus A = A$, therefore we should also have $\forall \phi \in X \to [0, 1] : \forall \kappa_1, \kappa_2 \in A : \kappa_1 \, {}_{\phi}\!\diamond \, \kappa_2 \in A$.

A Powerdomain for Nondeterminism-First

Necessary Conditions

We need to identify a subset $\mathcal{A}$ of $\wp(\underline{\mathcal{K}}(X))$ as the collection of admissible semantic objects.

- $\mathcal{A}$ admits a semilattice operation $\uplus$ (used as **nondeterministic-choice**), s.t. for all $A \in \mathcal{A}$, $A \uplus A = A$.
- $\mathcal{A}$ is equipped with a conditional-choice operation ${}_{\phi}\Diamond$ where $\phi : X \to [0,1]$ represents a Boolean-valued random variable.
- For all $A_1, A_2 \in \mathcal{A}$ and $\phi : X \to [0,1]$, if $\kappa_1 \in A_1$ and $\kappa_2 \in A_2$, then $\kappa_1 \,{}_{\phi}\Diamond\, \kappa_2$ should be in $A_1 \uplus A_2$.

A Convexity-Like Condition

For all $A \in \mathcal{A}$, we have $A \uplus A = A$, therefore we should also have $\forall \phi \in X \to [0,1] : \forall \kappa_1, \kappa_2 \in A : \kappa_1 \,{}_{\phi}\Diamond\, \kappa_2 \in A$.

A Powerdomain for Nondeterminism-First

Necessary Conditions

We need to identify a subset $\mathcal{A}$ of $\wp(\underline{\mathcal{K}}(X))$ as the collection of admissible semantic objects.

- $\mathcal{A}$ admits a semilattice operation $\uplus$ (used as **nondeterministic-choice**), s.t. for all $A \in \mathcal{A}$, $A \uplus A = A$.
- $\mathcal{A}$ is equipped with a conditional-choice operation ${}_{\phi}\!\diamond$ where $\phi : X \to [0, 1]$ represents a Boolean-valued random variable.
- For all $A_1, A_2 \in \mathcal{A}$ and $\phi : X \to [0, 1]$, if $\kappa_1 \in A_1$ and $\kappa_2 \in A_2$, then $\kappa_1 \,{}_{\phi}\!\diamond\, \kappa_2$ should be in $A_1 \uplus A_2$.

A Convexity-Like Condition

For all $A \in \mathcal{A}$, we have $A \uplus A = A$, therefore we should also have $\forall \phi \in X \to [0, 1] \colon \forall \kappa_1, \kappa_2 \in A \colon \kappa_1 \,{}_{\phi}\!\diamond\, \kappa_2 \in A$.

A Powerdomain for Nondeterminism-First

Necessary Conditions

We need to identify a subset $\mathcal{A}$ of $\wp(\underline{\mathcal{K}}(X))$ as the collection of admissible semantic objects.

- $\mathcal{A}$ admits a semilattice operation $\uplus$ (used as **nondeterministic-choice**), s.t. for all $A \in \mathcal{A}$, $A \uplus A = A$.
- $\mathcal{A}$ is equipped with a conditional-choice operation ${}_{\phi}\Diamond$ where $\phi : X \longrightarrow [0, 1]$ represents a Boolean-valued random variable.
- For all $A_1, A_2 \in \mathcal{A}$ and $\phi : X \longrightarrow [0, 1]$, if $\kappa_1 \in A_1$ and $\kappa_2 \in A_2$, then $\kappa_1 \,{}_{\phi}\Diamond\, \kappa_2$ should be in $A_1 \uplus A_2$.

A Convexity-Like Condition

For all $A \in \mathcal{A}$, we have $A \uplus A = A$, therefore we should also have $\forall \phi \in X \longrightarrow [0, 1] \colon \forall \kappa_1, \kappa_2 \in A \colon \kappa_1 \,{}_{\phi}\Diamond\, \kappa_2 \in A$.

Generalized Convexity

Let $\phi \cdot \kappa \stackrel{\text{def}}{=} \lambda x.\lambda x'.\phi(x) \cdot \kappa(x)(x')$ and $\kappa_1 + \kappa_2 \stackrel{\text{def}}{=} \lambda x.\lambda x.\kappa_1(x)(x') + \kappa_2(x)(x')$. Then $\kappa_1 \;{}_{\phi}\!\diamond\; \kappa_2$ can be represented as $\phi \cdot \kappa_1 + (\dot{1} - \phi) \cdot \kappa_2$.

Definition

A subset A of $\underline{\mathcal{K}}(X)$ is said to be **g-convex**, if for all sequences $\{\kappa_i\}_{i \in \mathbb{N}} \subseteq A$ and $\{\phi_i\}_{i \in \mathbb{N}} \subseteq X \to [0, 1]$ such that $\sum_{i=1}^{\infty} \phi_i = \dot{1}$, then $\sum_{i=1}^{\infty} \phi_i \cdot \kappa_i \in A$.

Clearly g-convexity of a set A implies that for all $\phi : X \to [0, 1]$ and $\kappa_1, \kappa_2 \in A$, we have $\kappa_1 \;{}_{\phi}\!\diamond\; \kappa_2 \in A$.

Generalized Convexity

Let $\phi \cdot \kappa \stackrel{\text{def}}{=} \lambda x.\lambda x'.\phi(x) \cdot \kappa(x)(x')$ and $\kappa_1 + \kappa_2 \stackrel{\text{def}}{=} \lambda x.\lambda x.\kappa_1(x)(x') + \kappa_2(x)(x')$. Then $\kappa_1 \,{}_\phi\!\diamond\, \kappa_2$ can be represented as $\phi \cdot \kappa_1 + (\dot{1} - \phi) \cdot \kappa_2$.

Definition

A subset A of $\underline{\mathcal{K}}(X)$ is said to be **g-convex**, if for all sequences $\{\kappa_i\}_{i \in \mathbb{N}} \subseteq A$ and $\{\phi_i\}_{i \in \mathbb{N}} \subseteq X \longrightarrow [0,1]$ such that $\sum_{i=1}^{\infty} \phi_i = \dot{1}$, then $\sum_{i=1}^{\infty} \phi_i \cdot \kappa_i \in A$.

Clearly g-convexity of a set A implies that for all $\phi : X \longrightarrow [0,1]$ and $\kappa_1, \kappa_2 \in A$, we have $\kappa_1 \,{}_\phi\!\diamond\, \kappa_2 \in A$.

Generalized Convexity

Let $\phi \cdot \kappa \stackrel{\text{def}}{=} \lambda x.\lambda x'.\phi(x) \cdot \kappa(x)(x')$ and $\kappa_1 + \kappa_2 \stackrel{\text{def}}{=} \lambda x.\lambda x.\kappa_1(x)(x') + \kappa_2(x)(x')$. Then $\kappa_1 \;{}_{\phi}\!\diamond\; \kappa_2$ can be represented as $\phi \cdot \kappa_1 + (\dot{1} - \phi) \cdot \kappa_2$.

Definition

A subset A of $\underline{\mathcal{K}}(X)$ is said to be **g-convex**, if for all sequences $\{\kappa_i\}_{i \in \mathbb{N}} \subseteq A$ and $\{\phi_i\}_{i \in \mathbb{N}} \subseteq X \to [0, 1]$ such that $\sum_{i=1}^{\infty} \phi_i = \dot{1}$, then $\sum_{i=1}^{\infty} \phi_i \cdot \kappa_i \in A$.

Clearly g-convexity of a set A implies that for all $\phi : X \to [0, 1]$ and $\kappa_1, \kappa_2 \in A$, we have $\kappa_1 \;{}_{\phi}\!\diamond\; \kappa_2 \in A$.

A G-Convex Powerdomain for Nondeterminism-First

Idea

Construct a Plotkin-style powerdomain on $\underline{\mathcal{K}}(X)$, ***except that* g-convexity** replaces standard convexity in the development.

Example

Consider the following nondeterministic program P

$$\texttt{if} \star \texttt{then}\ t := t + 1\ \texttt{else}\ t := t - 1\ \texttt{fi}$$

Let the state space $X \stackrel{\text{def}}{=} \mathbb{Z}$ represent the value of t. The common resolution for nondeterminism gives the following semantics

$$\lambda t.\{r \cdot \delta(t+1) + (1-r) \cdot \delta(t-1) \mid r \in [0,1]\},$$

while the nondeterminism-first resolution derives

$$\{\lambda t.\phi(t) \cdot \delta(t+1) + (1-\phi(t)) \cdot \delta(t-1) \mid \phi \in \mathbb{Z} \to [0,1]\}.$$

A G-Convex Powerdomain for Nondeterminism-First

Idea

Construct a Plotkin-style powerdomain on $\underline{\mathcal{K}}(X)$, ***except that* g-convexity** replaces standard convexity in the development.

Example

Consider the following nondeterministic program P

$$\texttt{if}\ \star\ \texttt{then}\ t := t + 1\ \texttt{else}\ t := t - 1\ \texttt{fi}$$

Let the state space $X \stackrel{\text{def}}{=} \mathbb{Z}$ represent the value of t. The common resolution for nondeterminism gives the following semantics

$$\lambda t.\{r \cdot \delta(t+1) + (1-r) \cdot \delta(t-1) \mid r \in [0,1]\},$$

while the nondeterminism-first resolution derives

$$\{\lambda t.\phi(t) \cdot \delta(t+1) + (1-\phi(t)) \cdot \delta(t-1) \mid \phi \in \mathbb{Z} \to [0,1]\}.$$

Summary

This Work

We have developed an **algebraic** framework for denotational semantics of **low-level** probabilistic programs, which can be instantiated with different models of nondeterminism, including the common resolution for nondeterminism and the new **nondeterminism-first**.

Limitations and Future Work

- The framework does not support for continuous distributions yet.
- We are looking for interesting applications of nondeterminism-first, especially for relational reasoning.

Summary

This Work

We have developed an **algebraic** framework for denotational semantics of **low-level** probabilistic programs, which can be instantiated with different models of nondeterminism, including the common resolution for nondeterminism and the new **nondeterminism-first**.

Limitations and Future Work

- The framework does not support for continuous distributions yet.
- We are looking for interesting applications of nondeterminism-first, especially for relational reasoning.