A Denotational Semantics for Low-Level Probabilistic Programs with Nondeterminism

Di Wang ¹  Jan Hoffmann ¹  Thomas Reps ²,³

¹Carnegie Mellon University

²University of Wisconsin

³GrammaTech, 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.

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.

Benefits of a Denotational Semantics

- Abstraction from details about program executions
- Compositionality

Low-Level Features:

- Imperative
- Unstructured control-flow

Operational semantics:
(Ferrer Fioriti and Hermanns 2015)

Denotational semantics:
This work
**Low-Level** Probabilistic Programs

**Example**

The following code implements a variant of geometric distributions.

```plaintext
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 $X \rightarrow D(X)$, where $X$ is a program state space and $D(X)$ consists of probability distributions over $X$. 
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 \( X \rightarrow \mathcal{D}(X) \), 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.

\[ n := 0; \]
\[ \text{while prob}(0.9) \text{ do} \]
\[ \quad n := n + 1; \]
\[ \quad \text{if } n \geq 10 \text{ then break} \]
\[ \quad \text{else continue} \]
\[ \text{od} \]

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 \( X \rightarrow D(X) \), where \( X \) is a program state space and \( D(X) \) 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.,

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

Nondeterminism in Probabilistic Programming

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

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

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.,

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

Nondeterminism in Probabilistic Programming

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

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

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.,

\[ f \in X \rightarrow \wp(Y). \]

Nondeterminism in Probabilistic Programming

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

\[ f \in X \rightarrow \wp(\mathcal{D}(X)), \]

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

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

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

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

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

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

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

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

Intuition: A nondeterministic program is a specification that models a collection of deterministic refinements.
When to Resolve Nondeterminism?

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

The Common Resolution: Input \textbf{Prior to Nondeterminism}

\[ f \in X \rightarrow \wp(D(X)) \]

What about: Nondeterminism \textbf{Prior to Input}?

\[ f \in \wp(X \rightarrow D(X)) \]

Intuition: A nondeterministic program is a specification that models a \textit{collection} of deterministic refinements.
Nondeterminism-**First**: Nondeterminism Prior to Input

**Example**

Consider the following program $P$ where ★ represents nondeterminism.

```plaintext
if prob(★) then t := t + 1 else t := t − 1 fi
```

**The Common Resolution**

- $t = 1$
- $t' = 2$ w.p. 0.5
- $t' = 0$ w.p. 0.5

**Nondeterminism-First**

- $t = 1$
- $t' = 2$ w.p. 0.5
- $t' = 0$ w.p. 0.5

★ resolved as 0.5

- $t = 1$
- $t' = 2$ w.p. 0.8
- $t' = 0$ w.p. 0.2

★ resolved as 0.8

★ resolved after $t$ is given

★ resolved before $t$ is given
**Nondeterminism-First**: Nondeterminism Prior to Input

**Example**

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

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

**The Common Resolution**

- $t = 1$
  - $t' = 2$ w.p. 0.5
  - $t' = 0$ w.p. 0.5
  - $\star$ 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.

\[
\text{if } \text{prob}(\star) \text{ then } t := t + 1 \text{ else } t := t - 1 \text{ fi}
\]

**The Common Resolution**

- $t = 1$
- $t' = 2$ w.p. 0.5
- $t' = 0$ w.p. 0.5

\[
\begin{align*}
\star \text{ resolved after } t \text{ is given}
\end{align*}
\]

**Nondeterminism-First**

- $t = 1$
- $t' = 2$ w.p. 0.5
- $t' = 0$ w.p. 0.5

\[
\begin{align*}
\star \text{ resolved as 0.5}
\end{align*}
\]

- $t = 1$
- $t' = 2$ w.p. 0.8
- $t' = 0$ w.p. 0.2

\[
\begin{align*}
\star \text{ resolved before } t \text{ is given}
\end{align*}
\]
Nondeterminism—**First**: What’s the Benefit?

**Example**

Consider the following program $P$ where ★ represents nondeterminism.

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

**Relational Reasoning about Refinements of a Program**

- For all refinements $P'$ of $P$, for all $t_1, t_2$, can we prove that $E_{t_1 \sim P'(t_1), t_2 \sim P'(t_2)}[t'_1 - t'_2] = t_1 - t_2$?
- 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.

\[
\text{if } \text{prob}(\star) \text{ then } t := t + 1 \text{ else } t := t - 1 \text{ fi}
\]

Relational Reasoning about Refinements of a Program

- For all refinements $P'$ of $P$, for all $t_1, t_2$, can we prove that $\exists t'_1 \sim P'(t_1), t'_2 \sim P'(t_2) [t'_1 - t'_2] = t_1 - t_2$?

- 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.

\[
\text{if } \text{prob(}\star\text{)} \text{ then } t := t + 1 \text{ else } t := t - 1 \text{ fi}
\]

**Relational Reasoning about Refinements of a Program**

- For all refinements $P'$ of $P$, for all $t_1, t_2$, can we prove that $\exists t_1' \sim P'(t_1), t_2' \sim P'(t_2) [t_1' - t_2'] = t_1 - t_2$?
- 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

\[ n \mod 2 = 0 \]
\[ n := n/2 \]
\[ n := 3 \times n + 1 \]

A tree-like hyper-path

\[ [n \neq 1] \]
\[ [n \mod 2 \neq 0] \]
\[ [n = 1] \]

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

```plaintext
if ★ then if prob(0.5) then \( t := 0 \) else \( t := 1 \) fi
else if prob(0.8) then \( t := 0 \) else \( t := 1 \) fi fi
```

Paths Annotated with Probabilities

- \( t' = 0 \) with prob 0.5
- \( t' = 1 \) with prob 0.5
- \( t' = 0 \) with prob 0.8
- \( t' = 1 \) with prob 0.2

Hyper-Paths, each of which stands for a distribution

- \( t' = 0 \) with prob 0.5
- \( t' = 1 \) with prob 0.5
- \( t' = 0 \) with prob 0.8
- \( t' = 1 \) with prob 0.2
Paths vs. Hyper-Paths

Example

```
if ★ then if prob(0.5) then t := 0 else t := 1 fi
else if prob(0.8) then t := 0 else t := 1 fi fi
```

Paths Annotated with Probabilities

- $t' = 0$
- $t' = 1$
- $t' = 0$
- $t' = 1$

Hyper-Paths, each of which stands for a distribution
Paths vs. Hyper-Paths

Example

\[
\begin{align*}
&\text{if } \star \text{ then if prob}(0.5) \text{ then } t := 0 \text{ else } t := 1 \text{ fi} \\
&\quad \text{else if prob}(0.8) \text{ then } t := 0 \text{ else } t := 1 \text{ fi fi}
\end{align*}
\]

Paths Annotated with Probabilities

\[
\begin{array}{cccc}
\bullet & \downarrow 0.5 & \bullet & \downarrow 0.5 \\
t' = 0 & t' = 1 & t' = 0 & t' = 1
\end{array}
\]

Hyper-Paths, each of which stands for a distribution

\[
\begin{array}{cccc}
\bullet & \downarrow 0.5 & 0.5 & \bullet \\
t' = 0 & t' = 1 & t' = 0 & t' = 1
\end{array}
\]
Control-Flow Hyper-Graphs

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

- The following hyper-graph

\[
\begin{align*}
    &v_0 \quad n := 0 \\
    &v_1 \quad \text{prob}(0.9) \\
    &v_2 \quad n := n + 1 \\
    &v_3 \quad n \geq 10 \\
    &v_4 \\
\end{align*}
\]

represents the control-flow of the example program

\[
\begin{align*}
    n &:= 0; \\
    \text{while prob}(0.9) \text{ do} \\
    &n := n + 1; \\
    &\text{if } n \geq 10 \text{ then break} \\
    &\text{else continue} \\
    \text{od}
\end{align*}
\]
Control-Flow Hyper-Graphs

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

```
\begin{align*}
V_0 & \rightarrow V_1 & n := 0; \\
\text{true} & \rightarrow V_2 & n := n + 1; \\
\text{false} & \rightarrow V_3 & n \geq 10; \\
\text{false} & \rightarrow V_3 & \text{break;} \\
\text{true} & \rightarrow V_4 & \text{continue;}
\end{align*}
```

represents the control-flow of the example program

```
n := 0;
while \text{prob}(0.9) do
    n := n + 1;
    if n \geq 10 then break
    else continue
od
```
Outline

Motivation

Control-Flow Hyper-Graphs

Algebraic Denotational Semantics

Nondeterminism-First
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 ~ Binomial(10, 0.5)`

- `⋯`

**State Transformers** $\mathcal{M}$ equipped with

- *sequencing* $\otimes$

- *conditional-choice* $\varphi \diamond$

- *nondeterministic-choice* $\sqcup$

\[
\langle \mathcal{M}, \sqsubseteq, \otimes, \varphi \diamond, \sqcup, \bot, 1 \rangle
\]

- $\langle \mathcal{M}, \sqsubseteq \rangle$ forms a directed complete partial order (dcpo) with $\bot$ as its least element.

- $\langle \mathcal{M}, \otimes, 1 \rangle$ forms a monoid.

- Nondeterministic-choice $\sqcup$ is a semilattice operation.
The Algebra

<table>
<thead>
<tr>
<th>Actions</th>
<th>State Transformers $\mathcal{M}$ equipped with</th>
</tr>
</thead>
<tbody>
<tr>
<td>skip</td>
<td>sequencing $\otimes$</td>
</tr>
<tr>
<td>$x := x + 5$</td>
<td>conditional-choice $\phi$</td>
</tr>
<tr>
<td>$k \sim \text{Binomial}(10, 0.5)$</td>
<td>nondeterministic-choice $\bigcirc$</td>
</tr>
</tbody>
</table>

$\langle \mathcal{M}, \sqsubseteq, \otimes, \phi \bigcirc, \sqcup, \bot, 1 \rangle$

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

**Actions**

- *skip*
  
  \[ x := x + 5 \]
  
  \[ k \sim \text{Binomial}(10, 0.5) \]
  
  \[ \ldots \]

**State Transformers** \( M \)

equipped with

- *sequencing* \( \otimes \)
- *conditional-choice* \( \varphi \diamond \)
- *nondeterministic-choice* \( \triangledown \)

\[
\langle M, \sqsubseteq, \otimes, \varphi \diamond, \triangledown, \bot, 1 \rangle
\]

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

Actions

\[ \text{skip} \]
\[ x := x + 5 \]
\[ k \sim \text{Binomial}(10, 0.5) \]
\[ \ldots \]

State Transformers \( M \)
equipped with

- Sequencing \( \otimes \)
- Conditional-choice \( \varphi \diamond \)
- Nondeterministic-choice \( \triangledown \)

\[ \left\langle M, \sqsubseteq, \otimes, \varphi \otimes, \triangledown, \bot, 1 \right\rangle \]

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

Actions

skip

\[ x := x + 5 \]
\[ k \sim \text{Binomial}(10, 0.5) \]
\[ \ldots \]

State Transformers \( \mathcal{M} \)
equipped with

- sequencing \( \otimes \)
- conditional-choice \( \varphi \)
- nondeterministic-choice \( \triangledown \)

\[ \langle \mathcal{M}, \sqsubseteq, \otimes, \varphi \triangledown, \triangledown, \bot, 1 \rangle \]

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

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

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

\[
S(v_0) = seq[n := 0](S(v_1)) \\
S(v_1) = prob[0.9](S(v_2), S(v_4)) \\
S(v_2) = seq[n := n + 1](S(v_3)) \\
S(v_3) = cond[n ≥ 10](S(v_4), S(v_1)) \\
S(v_4) = 1
\]
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.

\[
\begin{align*}
n &:= 0; \\
\text{while } \text{prob}(0.9) \text{ do} \\
& \quad n := n + 1; \\
& \quad \text{if } n \geq 10 \text{ then break} \\
& \quad \text{else } \text{continue} \\
& \text{od}
\end{align*}
\]

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

\[
\begin{align*}
S(v_0) &= \text{seq}[n := 0](S(v_1)) \\
S(v_2) &= \text{seq}[n := n + 1](S(v_3)) \\
S(v_4) &= 1 \\
S(v_1) &= \text{prob}[0.9](S(v_2), S(v_4)) \\
S(v_3) &= \text{cond}[n \geq 10](S(v_4), S(v_1))
\end{align*}
\]
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.

\[
\begin{align*}
n &:= 0; \\
\text{while } &\text{prob}(0.9) \text{ do} \\
&\quad n := n + 1; \\
&\quad \text{if } n \geq 10 \text{ then break} \\
&\quad \text{else continue} \\
\text{end}
\end{align*}
\]

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

\[
\begin{align*}
S(v_0) &= \text{seq}[n := 0](S(v_1)) \\
S(v_1) &= \text{prob}[0.9](S(v_2), S(v_4)) \\
S(v_2) &= \text{seq}[n := n + 1](S(v_3)) \\
S(v_3) &= \text{cond}[n \geq 10](S(v_4), S(v_1)) \\
S(v_4) &= 1
\end{align*}
\]
Fixpoint Semantics for Hyper-Graphs

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

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

Use the algebra to reinterpret the equation system

\[ S(v_0) = [n := 0] \otimes S(v_1) \quad S(v_2) = [n := n + 1] \otimes S(v_3) \quad S(v_4) = 1 \]
\[ S(v_1) = S(v_2) \text{ prob}(0.9) \triangle S(v_4) \quad S(v_3) = S(v_4) \text{ cond}[n \geq 10] \triangle S(v_1) \]

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

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

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

Use the algebra to reinterpret the equation system

\[
S(v_0) = [n := 0] \otimes S(v_1) \quad S(v_2) = [n := n + 1] \otimes S(v_3) \quad S(v_4) = 1
\]

\[
S(v_1) = S(v_2) \text{prob}(0.9) \diamond S(v_4) \quad S(v_3) = S(v_4) \text{n}_{\geq 10} \diamond S(v_1)
\]

where $[\cdot]$ maps actions into state transformers in $\mathcal{M}$. 
A Denotational Semantics without Nondeterminism

• \( X \overset{\text{def}}{=} \text{Var} \rightarrow_{\text{fin}} \mathbb{Q} \) and \( M \overset{\text{def}}{=} X \rightarrow \mathcal{D}(X) \).

• \( \mathcal{D}(X) \) stands for sub-probability distributions on \( X \), i.e., \( \Delta \in \mathcal{D}(X) \) iff \( \Delta : X \rightarrow [0, 1] \) and \( \sum_{x \in X} \Delta(x) \leq 1 \).

• For actions \( \text{act} \), we have \([\text{act}] \in M\).

• For conditions \( \varphi \), we have \([\varphi] : X \rightarrow [0, 1] \), e.g., \([\text{prob}(p)] \overset{\text{def}}{=} \lambda_.p\).

• \( f \sqsubseteq g \overset{\text{def}}{=} \forall x \in X : \forall x' \in X : f(x)(x') \leq g(x)(x') \).

• \( f \otimes g \overset{\text{def}}{=} \lambda x.\lambda x''. \sum_{x' \in X} f(x, x') \cdot g(x', x'') \).

• \( f \vartriangleleft g \overset{\text{def}}{=} \lambda x.\lambda x'. [\varphi](x) \cdot f(x)(x') + (1 - [\varphi](x)) \cdot g(x)(x') \).

• \( \bot \overset{\text{def}}{=} \lambda_.\lambda_.0 \).

• \( 1 \overset{\text{def}}{=} \lambda x.\delta(x) \) where the point distribution \( \delta(x) \overset{\text{def}}{=} \lambda x'.[x = x'] \).
A Denotational Semantics without Nondeterminism

- \( X \overset{\text{def}}{=} \text{Var} \rightarrow_{\text{fin}} \mathbb{Q} \) and \( M \overset{\text{def}}{=} X \rightarrow \mathcal{D}(X) \).

- \( \mathcal{D}(X) \) stands for sub-probability distributions on \( X \), i.e., \( \Delta \in \mathcal{D}(X) \) iff \( \Delta : X \rightarrow [0, 1] \) and \( \sum_{x \in X} \Delta(x) \leq 1 \).

- For actions \( \text{act} \), we have \([\text{act}] \in M \).

- For conditions \( \varphi \), we have \([\varphi] : X \rightarrow [0, 1] \), e.g., \([\text{prob}(p)] \overset{\text{def}}{=} \lambda . p \).

- \( f \sqsubseteq g \overset{\text{def}}{=} \forall x \in X : \forall x' \in X : f(x)(x') \leq g(x)(x') \).

- \( f \otimes g \overset{\text{def}}{=} \lambda x. \lambda x'' . \sum_{x' \in X} f(x, x') \cdot g(x', x'') \).

- \( f \varphi \Diamond g \overset{\text{def}}{=} \lambda x. \lambda x' . ([\varphi](x) \cdot f(x)(x') + (1 - [\varphi](x)) \cdot g(x)(x')) \).

- \( \bot \overset{\text{def}}{=} \lambda . \bot . 0 \).

- \( 1 \overset{\text{def}}{=} \lambda x . \delta(x) \) where the point distribution \( \delta(x) \overset{\text{def}}{=} \lambda x'. [x = x'] \).
A Denotational Semantics without Nondeterminism

- $X \overset{\text{def}}{=} \text{Var} \rightarrow_{\text{fin}} \mathbb{Q}$ and $M \overset{\text{def}}{=} X \rightarrow \mathcal{D}(X)$.
- $\mathcal{D}(X)$ stands for sub-probability distributions on $X$, i.e., $\Delta \in \mathcal{D}(X)$ iff $\Delta : X \rightarrow [0, 1]$ and $\sum_{x \in X} \Delta(x) \leq 1$.
- For actions act, we have $\llbracket \text{act} \rrbracket \in M$.
- For conditions $\varphi$, we have $\llbracket \varphi \rrbracket : X \rightarrow [0, 1]$, e.g., $\llbracket \text{prob}(p) \rrbracket \overset{\text{def}}{=} \lambda_.p$.
- $f \sqsubseteq g \overset{\text{def}}{=} \forall x \in X : \forall x' \in X : f(x)(x') \leq g(x)(x')$.
- $f \otimes g \overset{\text{def}}{=} \lambda x.\lambda x''. \sum_{x' \in X} f(x, x') \cdot g(x', x'')$.
- $f \varphi \cdot g \overset{\text{def}}{=} \lambda x.\lambda x'. \llbracket \varphi \rrbracket (x) \cdot f(x)(x') + (1 - \llbracket \varphi \rrbracket (x)) \cdot g(x)(x')$.
- $\bot \overset{\text{def}}{=} \lambda_.\lambda_.0$.
- $1 \overset{\text{def}}{=} \lambda x.\delta(x)$ where the point distribution $\delta(x) \overset{\text{def}}{=} \lambda x'.[x = x']$. 
A Denotational Semantics without Nondeterminism

- \( X \overset{\text{def}}{=} \text{Var} \rightarrow_{\text{fin}} \mathbb{Q} \) and \( M \overset{\text{def}}{=} X \rightarrow \mathcal{D}(X) \).
- \( \mathcal{D}(X) \) stands for sub-probability distributions on \( X \), i.e., \( \Delta \in \mathcal{D}(X) \) iff \( \Delta : X \rightarrow [0, 1] \) and \( \sum_{x \in X} \Delta(x) \leq 1 \).
- For actions \( \text{act} \), we have \([\text{act}] \in M\).
- For conditions \( \varphi \), we have \([\varphi] : X \rightarrow [0, 1] \), e.g., \([\text{prob}(p)] \overset{\text{def}}{=} \lambda_\_p \).
- \( f \sqsubseteq g \overset{\text{def}}{=} \forall x \in X : \forall x' \in X : f(x)(x') \leq g(x)(x') \).
- \( f \otimes g \overset{\text{def}}{=} \lambda x.\lambda x''. \sum_{x' \in X} f(x, x') \cdot g(x', x'') \).
- \( f \varphi \Diamond g \overset{\text{def}}{=} \lambda x.\lambda x'. [\varphi](x) \cdot f(x)(x') + (1 - [\varphi](x)) \cdot g(x)(x') \).
- \( \bot \overset{\text{def}}{=} \lambda_\_.\lambda_\_.0 \).
- \( 1 \overset{\text{def}}{=} \lambda x.\delta(x) \) where the point distribution \( \delta(x) \overset{\text{def}}{=} \lambda x'.[x = x'] \).
A Denotational Semantics without Nondeterminism

- \( X \overset{\text{def}}{=} \text{Var} \overset{\text{fin}}{\rightarrow} \mathbb{Q} \) and \( M \overset{\text{def}}{=} X \rightarrow \mathcal{D}(X) \).

- \( \mathcal{D}(X) \) stands for sub-probability distributions on \( X \), i.e., \( \Delta \in \mathcal{D}(X) \) iff \( \Delta : X \rightarrow [0, 1] \) and \( \sum_{x \in X} \Delta(x) \leq 1 \).

- For actions \( \text{act} \), we have \( \llbracket \text{act} \rrbracket \in M \).

- For conditions \( \varphi \), we have \( \llbracket \varphi \rrbracket : X \rightarrow [0, 1] \), e.g., \( \llbracket \text{prob}(p) \rrbracket \overset{\text{def}}{=} \lambda_.p \).

- \( f \sqsubseteq g \overset{\text{def}}{=} \forall x \in X : \forall x' \in X : f(x)(x') \leq g(x)(x') \).

- \( f \otimes g \overset{\text{def}}{=} \lambda x.\lambda x''. \sum_{x' \in X} f(x, x') \cdot g(x', x'') \).

- \( f \varphi \downarrow g \overset{\text{def}}{=} \lambda x.\lambda x'. \llbracket \varphi \rrbracket (x) \cdot f(x)(x') + (1 - \llbracket \varphi \rrbracket (x)) \cdot g(x)(x') \).

- \( \bot \overset{\text{def}}{=} \lambda_.\lambda_.0 \).

- \( 1 \overset{\text{def}}{=} \lambda x.\delta(x) \) where the point distribution \( \delta(x) \overset{\text{def}}{=} \lambda x'.[x = x'] \).
A Denotational Semantics without Nondeterminism

- \( X \overset{\text{def}}{=} \text{Var} \rightarrow_{\text{fin}} \mathbb{Q} \) and \( M \overset{\text{def}}{=} X \rightarrow \mathcal{D}(X) \).
- \( \mathcal{D}(X) \) stands for sub-probability distributions on \( X \), i.e., \( \Delta \in \mathcal{D}(X) \) iff \( \Delta : X \rightarrow [0, 1] \) and \( \sum_{x \in X} \Delta(x) \leq 1 \).
- For actions \( \text{act} \), we have \( \llbracket \text{act} \rrbracket \in M \).
- For conditions \( \varphi \), we have \( \llbracket \varphi \rrbracket : X \rightarrow [0, 1] \), e.g., \( \llbracket \text{prob}(p) \rrbracket \overset{\text{def}}{=} \lambda \_ . p \).
- \( f \sqsubseteq g \overset{\text{def}}{=} \forall x \in X : \forall x' \in X : f(x)(x') \leq g(x)(x') \).
- \( f \otimes g \overset{\text{def}}{=} \lambda x . \lambda x''. \sum_{x' \in X} f(x, x') \cdot g(x', x'') \).
- \( f \varphi \diamond g \overset{\text{def}}{=} \lambda x . \lambda x'. \llbracket \varphi \rrbracket (x) \cdot f(x)(x') + (1 - \llbracket \varphi \rrbracket (x)) \cdot g(x)(x') \).
- \( \bot \overset{\text{def}}{=} \lambda \_ . \lambda \_ . 0 \).
- \( 1 \overset{\text{def}}{=} \lambda x . \delta(x) \) where the point distribution \( \delta(x) \overset{\text{def}}{=} \lambda x'. [x = x'] \).
Because $\text{Var} = \{n\}$ is a singleton, we present the semantics as if $X \overset{\text{def}}{=} \mathbb{Z}$.

\[
S(v_0) = \lambda_. \sum_{k=0}^{9} (0.1 \times 0.9^k) \cdot \delta(k) + 0.3486784401 \cdot \delta(10)
\]

$\delta(n_0)$ represents a point distribution at $n_0$. 
A Denotational Semantics without Nondeterminism

\[
n := 0;\\
\text{while } \text{prob}(0.9) \text{ do}\\
\quad n := n + 1;\\
\quad \text{if } n \geq 10 \text{ then break}\\
\quad \text{else continue}\\
\text{od}
\]

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

\[
\mathcal{S}(v_0) = \lambda_. \sum_{k=0}^{9} (0.1 \times 0.9^k) \cdot \delta(k) + 0.3486784401 \cdot \delta(10)
\]

\(\delta(n_0)\) represents a point distribution at \(n_0\).
$n := 0$;
while $\text{prob}(0.9)$ do
    $n := n + 1$;
    if $n \geq 10$ then break
    else continue
    od

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

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

$\delta(n_0)$ represents a point distribution at $n_0$. 
A Denotational Semantics without Nondeterminism

\[
n := 0; \\
\text{while } \text{prob}(0.9) \text{ do} \\
\quad n := n + 1; \\
\quad \text{if } n \geq 10 \text{ then break} \\
\quad \text{else continue} \\
\text{od}
\]

Recall the equation

\[
S(v_0) = [n := 0] \otimes S(v_1)
\]

Obtain \(S(v_0)\) from \(S(v_1)\)

\[
S(v_0) = \lambda_\_. \sum_{k=0}^{9} (0.1 \times 0.9^k) \cdot \delta(k) + 0.3486784401 \cdot \delta(10)
\]

\[
[n := 0] = \lambda_. \delta(0)
\]

\[
S(v_1) = \lambda n. [n \geq 9] \cdot (0.1 \cdot \delta(n) + 0.9 \cdot \delta(n + 1)) + \\
[n < 9] \cdot \left(\sum_{k=n}^{\infty} (0.1 \times 0.9^{k-n}) \cdot \delta(\min\{k, 10\})\right)
\]
A Denotational Semantics without Nondeterminism

\[
\begin{align*}
n &:= 0; \\
\text{while } \text{prob}(0.9) \text{ do} & \\
& \quad n := n + 1; \\
& \quad \text{if } n \geq 10 \text{ then break} \\
& \quad \text{else continue} \\
\end{align*}
\]

Recall the equation

\[
S(v_0) = \left[ n := 0 \right] \otimes S(v_1)
\]

Obtain \(S(v_0)\) from \(S(v_1)\)

\[
S(v_0) = \lambda_. \sum_{k=0}^{9} (0.1 \times 0.9^k) \cdot \delta(k) + 0.3486784401 \cdot \delta(10)
\]

\[
\left[ n := 0 \right] = \lambda_. \delta(0)
\]

\[
S(v_1) = \lambda n. \left[ n \geq 9 \right] \cdot (0.1 \cdot \delta(n) + 0.9 \cdot \delta(n + 1)) + \\
\left[ n < 9 \right] \cdot \left( \sum_{k=n}^{\infty} (0.1 \times 0.9^{k-n}) \cdot \delta(\min\{k, 10\}) \right)
\]
A Denotational Semantics without Nondeterminism

\[ n := 0; \]
\[ \text{while } \text{prob}(0.9) \text{ do} \]
\[ n := n + 1; \]
\[ \text{if } n \geq 10 \text{ then break} \]
\[ \text{else continue} \]
\[ \text{od} \]

Recall the equation

\[ S(v_0) = [n := 0] \otimes S(v_1) \]

Obtain \( S(v_0) \) from \( S(v_1) \)

\[ S(v_0) = \lambda_\_. \sum_{k=0}^{9} (0.1 \times 0.9^k) \cdot \delta(k) + 0.3486784401 \cdot \delta(10) \]

\[ [n := 0] = \lambda_. \delta(0) \]

\[ S(v_1) = \lambda n.[n \geq 9] \cdot (0.1 \cdot \delta(n) + 0.9 \cdot \delta(n + 1)) + \]

\[ [n < 9] \cdot \left( \sum_{k=n}^{\infty} (0.1 \times 0.9^{k-n}) \cdot \delta(\min\{k, 10\}) \right) \]
Outline

Motivation

Control-Flow Hyper-Graphs

Algebraic Denotational Semantics

Nondeterminism-First
Sub-Probability Kernels

**Definition**

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

**Goal**

The common resolution for nondeterminism admits the following signature

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

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

$$\wp(X \to \mathcal{D}(X)) \equiv \wp(\mathcal{K}(X)).$$
Sub-Probability Kernels

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

Goal
The common resolution for nondeterminism admits the following signature

$$X \to \wp(D(X)),$$

while our nondeterminism-first model should have the following signature

$$\wp(X \to \mathcal{D}(X)) \equiv \wp(\mathcal{K}(X)).$$
Reasoning with Nondeterminism-First

Example

Recall the following nondeterministic program $P$

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

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

\[
\mathbb{E}_{t_1 \sim P'(t_1), t_2' \sim P'(t_2)}[t_1' - t_2'] = \mathbb{E}_{t_1' \sim P'(t_1)}[t_1'] - \mathbb{E}_{t_2' \sim P'(t_2)}[t_2']
\]

\[
= (r(t_1 + 1) + (1 - r)(t_1 - 1)) - (r(t_2 + 1) + (1 - r)(t_2 - 1))
\]

\[
= t_1 - t_2
\]
Example

Recall the following nondeterministic program \( P \)

\[
\text{if } \text{prob}(\star) \text{ then } t := t + 1 \text{ else } t := t - 1 \text{ fi}
\]

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

\[
\mathbb{E}_{t_1 \sim P'(t_1), t_2 \sim P'(t_2)}[t'_1 - t'_2] = \mathbb{E}_{t'_1 \sim P'(t_1)}[t'_1] - \mathbb{E}_{t'_2 \sim P'(t_2)}[t'_2]
\]

\[
= (r(t_1 + 1) + (1 - r)(t_1 - 1)) - (r(t_2 + 1) + (1 - r)(t_2 - 1))
\]

\[
= t_1 - t_2
\]
A Powerdomain for Nondeterminism-First

Necessary Conditions

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

- $\mathcal{A}$ admits a semilattice operation $\cup$ (used as **nondeterministic-choice**), s.t. for all $A \in \mathcal{A}$, $A \cup A = A$.
- $\mathcal{A}$ is equipped with a conditional-choice operation $\diamond$ 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 $\kappa_1 \diamond \kappa_2$ should be in $A_1 \cup A_2$.

A Convexity-Like Condition

For all $A \in \mathcal{A}$, we have $A \cup A = A$, therefore we should also have $\forall \phi \in X \to [0, 1] : \forall \kappa_1, \kappa_2 \in A : \kappa_1 \diamond \kappa_2 \in A$. 
A Powerdomain for Nondeterminism-First

Necessary Conditions

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

- $\mathcal{A}$ admits a semilattice operation $\cup$ (used as nondeterministic-choice), s.t. for all $A \in \mathcal{A}$, $A \cup A = A$.
- $\mathcal{A}$ 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 $A_1 \cup A_2$.

A Convexity-Like Condition

For all $A \in \mathcal{A}$, we have $A \cup A = A$, 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 $\wp(K(X))$ as the collection of admissible semantic objects.

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

A Convexity-Like Condition

For all $A \in \mathcal{A}$, we have $A \sqcup A = A$, therefore we should also have $\forall \phi \in X \rightarrow [0, 1] : \forall \kappa_1, \kappa_2 \in A : \kappa_1 \phi^\Diamond \kappa_2 \in A$. 
A Powerdomain for Nondeterminism-First

Necessary Conditions

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

- $\mathcal{A}$ admits a semilattice operation $\cup$ (used as \textbf{nondeterministic-choice}), s.t. for all $A \in \mathcal{A}$, $A \cup A = A$.
- $\mathcal{A}$ 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 $A_1 \cup A_2$.

A Convexity-Like Condition

For all $A \in \mathcal{A}$, we have $A \cup A = A$, 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 $\wp(K(X))$ as the collection of admissible semantic objects.

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

A Convexity-Like Condition

For all $A \in \mathcal{A}$, we have $A \sqcup A = A$, therefore we should also have $\forall \phi \in X \rightarrow [0, 1] : \forall \kappa_1, \kappa_2 \in A : \kappa_1 \phi \diamond \kappa_2 \in A$. 
Generalized Convexity

Let \( \phi \cdot \kappa \overset{\text{def}}{=} \lambda x.\lambda x'.\phi(x) \cdot \kappa(x)(x') \) and \( \kappa_1 + \kappa_2 \overset{\text{def}}{=} \lambda x.\lambda x.\kappa_1(x)(x') + \kappa_2(x)(x') \). Then \( \kappa_1 \phi \diamond \kappa_2 \) can be represented as \( \phi \cdot \kappa_1 + (\hat{1} - \phi) \cdot \kappa_2 \).

Definition

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

Clearly g-convexity of a set \( A \) implies that for all \( \phi : X \rightarrow [0,1] \) and \( \kappa_1, \kappa_2 \in A \), we have \( \kappa_1 \phi \diamond \kappa_2 \in A \).
Generalized Convexity

Let $\phi \cdot \kappa \overset{\text{def}}{=} \lambda x.\lambda x'. \phi(x) \cdot \kappa(x)(x')$ and $\kappa_1 + \kappa_2 \overset{\text{def}}{=} \lambda x.\lambda x. \kappa_1(x)(x') + \kappa_2(x)(x')$. Then $\kappa_1 \phi \lozenge \kappa_2$ can be represented as $\phi \cdot \kappa_1 + (\hat{1} - \phi) \cdot \kappa_2$.

Definition

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

Clearly g-convexity of a set $A$ implies that for all $\phi : X \rightarrow [0, 1]$ and $\kappa_1, \kappa_2 \in A$, we have $\kappa_1 \phi \lozenge \kappa_2 \in A$. 
Generalized Convexity

Let \( \phi \cdot \kappa \overset{\text{def}}{=} \lambda x.\lambda x'.\phi(x) \cdot \kappa(x)(x') \) and \( \kappa_1 + \kappa_2 \overset{\text{def}}{=} \lambda x.\lambda x.\kappa_1(x)(x') + \kappa_2(x)(x') \). Then \( \kappa_1 \phi \diamond \kappa_2 \) can be represented as \( \phi \cdot \kappa_1 + (\hat{1} - \phi) \cdot \kappa_2 \).

**Definition**

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

Clearly g-convexity of a set \( A \) implies that for all \( \phi : X \rightarrow [0, 1] \) 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 $\mathcal{K}(X)$, except that g-convexity replaces standard convexity in the development.

Example

Consider the following nondeterministic program $P$

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

Let the state space $X \overset{\text{def}}{=} \mathbb{Z}$ 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 $\mathcal{K}(X)$, except that g-convexity replaces standard convexity in the development.

**Example**

Consider the following nondeterministic program $P$

```plaintext
if ⋄ then t := t + 1 else t := t − 1 fi
```

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

$$\lambda t.\{r \cdot \delta(t + 1) + (1 - r) \cdot \delta(t - 1) \mid r \in \mathbb{R}_0^1\},$$

while the nondeterminism-first resolution derives

$$\{\lambda t.\phi(t) \cdot \delta(t + 1) + (1 - \phi(t)) \cdot \delta(t - 1) \mid \phi \in \mathbb{Z} \rightarrow \mathbb{R}_0^1\}.$$
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.