Synthetic probability theory: an axiomatic approach to nondeterminism

Paul Wang

Laboratoire d'Informatique de Paris 6
Sorbonne Université

Séminaire Général de Logique,
September 29th, 2025

Introduction

- Give an abstract, purely algebraic, axiomatic framework to reason about probability/nondeterminism.

Introduction

- Give an abstract, purely algebraic, axiomatic framework to reason about probability/nondeterminism.
- *Synthetic approach*, avoid dealing with the low-level (set-theoretic) details.

Introduction

- Give an abstract, purely algebraic, axiomatic framework to reason about probability/nondeterminism.
- *Synthetic approach*, avoid dealing with the low-level (set-theoretic) details. Analogy: machine code versus high-level programming language.

Introduction

- Give an abstract, purely algebraic, axiomatic framework to reason about probability/nondeterminism.
- *Synthetic approach*, avoid dealing with the low-level (set-theoretic) details. Analogy: machine code versus high-level programming language.
- Based on Fritz 2020, Cho and Jacobs 2019.

Introduction

- Give an abstract, purely algebraic, axiomatic framework to reason about probability/nondeterminism.
- *Synthetic approach*, avoid dealing with the low-level (set-theoretic) details. Analogy: machine code versus high-level programming language.
- Based on Fritz 2020, Cho and Jacobs 2019.
  Fritz's seminal paper also deals with conditional independence, sufficient statistics, almost sure equality...

Introduction

- There are several axiomatizations that one might consider, for instance to stick closer to probability theory.

Introduction

- There are several axiomatizations that one might consider, for instance to stick closer to probability theory. Some of them are not first-order.

Introduction

- There are several axiomatizations that one might consider, for instance to stick closer to probability theory. Some of them are not first-order.
- Today: we will focus on a core first-order theory.

Outline

Outline

Standard Borel Spaces

Standard Borel Spaces

- Consider the following collection of measurable spaces: the $\{1, \ldots, n\} \times \mathbb{N}^k \times \mathbb{R}^d$, for $n \geq 1$, $k \geq 0$, $d \geq 0$.

Standard Borel Spaces

- Consider the following collection of measurable spaces: the $\{1, \ldots, n\} \times \mathbb{N}^k \times \mathbb{R}^d$, for $n \geq 1$, $k \geq 0$, $d \geq 0$. Let *Borel* denote this set of spaces.

Standard Borel Spaces

- Consider the following collection of measurable spaces: the $\{1, \ldots, n\} \times \mathbb{N}^k \times \mathbb{R}^d$, for $n \geq 1$, $k \geq 0$, $d \geq 0$. Let *Borel* denote this set of spaces.
- Markov kernels $\approx$ "measurable functions with probabilistic noise" $\approx$ "measurable families of probability distributions"

Standard Borel Spaces

- Consider the following collection of measurable spaces: the $\{1, \ldots, n\} \times \mathbb{N}^k \times \mathbb{R}^d$, for $n \geq 1$, $k \geq 0$, $d \geq 0$. Let *Borel* denote this set of spaces.
- Markov kernels $\approx$ "measurable functions with probabilistic noise" $\approx$ "measurable families of probability distributions"
- For $X, Y \in \mathit{Borel}$, a *Markov kernel* $X \rightsquigarrow Y$ is given by a function $f : X \times \Sigma_Y \to [0, 1]$, such that

Standard Borel Spaces

- Consider the following collection of measurable spaces: the $\{1, \ldots, n\} \times \mathbb{N}^k \times \mathbb{R}^d$, for $n \geq 1$, $k \geq 0$, $d \geq 0$. Let *Borel* denote this set of spaces.
- Markov kernels $\approx$ "measurable functions with probabilistic noise" $\approx$ "measurable families of probability distributions"
- For $X, Y \in \mathit{Borel}$, a *Markov kernel* $X \rightsquigarrow Y$ is given by a function $f : X \times \Sigma_Y \to [0, 1]$, such that
  1. For all $x \in X$, the function $f(x, \cdot) : \Sigma_Y \to [0, 1]$ is a probability measure on (Y, Σ_Y).

Standard Borel Spaces

- Consider the following collection of measurable spaces: the $\{1, \ldots, n\} \times \mathbb{N}^k \times \mathbb{R}^d$, for $n \geq 1$, $k \geq 0$, $d \geq 0$. Let *Borel* denote this set of spaces.
- Markov kernels $\approx$ "measurable functions with probabilistic noise" $\approx$ "measurable families of probability distributions"
- For $X, Y \in \mathit{Borel}$, a *Markov kernel* $X \rightsquigarrow Y$ is given by a function $f : X \times \Sigma_Y \to [0, 1]$, such that
  1. For all $x \in X$, the function $f(x, \cdot) : \Sigma_Y \to [0, 1]$ is a probability measure on (Y, Σ_Y).
  2. For all $B \in \Sigma_Y$, the function $f(\cdot, B) : X \to [0, 1]$ is measurable.

Standard Borel Spaces

- Consider the following collection of measurable spaces: the $\{1, \ldots, n\} \times \mathbb{N}^k \times \mathbb{R}^d$, for $n \geq 1$, $k \geq 0$, $d \geq 0$. Let *Borel* denote this set of spaces.
- Markov kernels $\approx$ "measurable functions with probabilistic noise" $\approx$ "measurable families of probability distributions"
- For $X, Y \in \mathit{Borel}$, a *Markov kernel* $X \rightsquigarrow Y$ is given by a function $f : X \times \Sigma_Y \to [0, 1]$, such that
  1. For all $x \in X$, the function $f(x, \cdot) : \Sigma_Y \to [0, 1]$ is a probability measure on (Y, Σ_Y).
  2. For all $B \in \Sigma_Y$, the function $f(\cdot, B) : X \to [0, 1]$ is measurable.

  Let $\mathcal{F}$ denote the set of all such Markov kernels. We may write $f : X \to Y$ for deterministic functions, viewed as kernels.

Operations

Question

What are the key operations on the pair of sorts $(Borel, \mathcal{F})$, as far as probability theory is concerned?

Operations

Question

What are the key operations on the pair of sorts $(\mathit{Borel}, \mathcal{F})$, as far as probability theory is concerned?

- Product of spaces $\times : \mathit{Borel} \times \mathit{Borel} \to \mathit{Borel}$.

Operations

Question

What are the key operations on the pair of sorts $(\mathit{Borel}, \mathcal{F})$, as far as probability theory is concerned?

- Product of spaces $\times : \mathit{Borel} \times \mathit{Borel} \to \mathit{Borel}$.
- Independent coupling/tensor product of probability distributions/parallel product of kernels $\otimes : \mathcal{F} \times \mathcal{F} \to \mathcal{F}$.

Operations

Question

What are the key operations on the pair of sorts $(\mathit{Borel}, \mathcal{F})$, as far as probability theory is concerned?

- Product of spaces $\times : \mathit{Borel} \times \mathit{Borel} \to \mathit{Borel}$.
- Independent coupling/tensor product of probability distributions/parallel product of kernels $\otimes : \mathcal{F} \times \mathcal{F} \to \mathcal{F}$.
- Chaining/sequential composition of Markov kernels:

Operations

Question

What are the key operations on the pair of sorts $(\mathit{Borel}, \mathcal{F})$, as far as probability theory is concerned?

- Product of spaces $\times : \mathit{Borel} \times \mathit{Borel} \to \mathit{Borel}$.
- Independent coupling/tensor product of probability distributions/parallel product of kernels $\otimes : \mathcal{F} \times \mathcal{F} \to \mathcal{F}$.
- Chaining/sequential composition of Markov kernels: if $f : X \rightsquigarrow Y$ and $g : Y \rightsquigarrow Z$, we define $(f ; g) : X \rightsquigarrow Z$ via the *Chapman-Kolmogorov* formula:

Operations

Question

What are the key operations on the pair of sorts $(\mathit{Borel}, \mathcal{F})$, as far as probability theory is concerned?

- Product of spaces $\times : \mathit{Borel} \times \mathit{Borel} \to \mathit{Borel}$.
- Independent coupling/tensor product of probability distributions/parallel product of kernels $\otimes : \mathcal{F} \times \mathcal{F} \to \mathcal{F}$.
- Chaining/sequential composition of Markov kernels: if $f : X \rightsquigarrow Y$ and $g : Y \rightsquigarrow Z$, we define $(f ; g) : X \rightsquigarrow Z$ via the *Chapman-Kolmogorov* formula:

$$(f ; g)(x, C) = \int_Y g(y, C) f(x, dy),$$

Operations

Question

What are the key operations on the pair of sorts $(\mathit{Borel}, \mathcal{F})$, as far as probability theory is concerned?

- Product of spaces $\times : \mathit{Borel} \times \mathit{Borel} \to \mathit{Borel}$.
- Independent coupling/tensor product of probability distributions/parallel product of kernels $\otimes : \mathcal{F} \times \mathcal{F} \to \mathcal{F}$.
- Chaining/sequential composition of Markov kernels: if $f : X \rightsquigarrow Y$ and $g : Y \rightsquigarrow Z$, we define $(f ; g) : X \rightsquigarrow Z$ via the *Chapman-Kolmogorov* formula:

$$(f ; g)(x, C) = \int_Y g(y, C) f(x, dy),$$

for $x \in X$ and $C \in \Sigma_Z$.

Operations

Question

What are the key operations on the pair of sorts $(\mathit{Borel}, \mathcal{F})$, as far as probability theory is concerned?

- Product of spaces $\times : \mathit{Borel} \times \mathit{Borel} \to \mathit{Borel}$.
- Independent coupling/tensor product of probability distributions/parallel product of kernels $\otimes : \mathcal{F} \times \mathcal{F} \to \mathcal{F}$.
- Chaining/sequential composition of Markov kernels: if $f : X \rightsquigarrow Y$ and $g : Y \rightsquigarrow Z$, we define $(f ; g) : X \rightsquigarrow Z$ via the *Chapman-Kolmogorov* formula:

$$(f ; g)(x, C) = \int_Y g(y, C) f(x, dy),$$

for $x \in X$ and $C \in \Sigma_Z$. This can be encoded as a relation $R \subseteq \mathcal{F} \times \mathcal{F} \times \mathcal{F}$.

Operations, continued

There are also more generic functions, not specifically related to probability theory.

Operations, continued

There are also more generic functions, not specifically related to probability theory.

- Domain and codomain maps $\mathrm{dom}, \mathrm{cod} : \mathcal{F} \to \mathit{Borel}$.

Operations, continued

There are also more generic functions, not specifically related to probability theory.

- Domain and codomain maps $\mathrm{dom}, \mathrm{cod} : \mathcal{F} \to \mathit{Borel}$.
- Copy map $\mathrm{copy} : \mathit{Borel} \to \mathcal{F}$,

Operations, continued

There are also more generic functions, not specifically related to probability theory.

- Domain and codomain maps $\mathrm{dom}, \mathrm{cod} : \mathcal{F} \to \mathit{Borel}$.
- Copy map $\mathrm{copy} : \mathit{Borel} \to \mathcal{F}$, such that for all $X \in \mathit{Borel}$, we have that $\mathrm{copy}(X) : X \to X \times X$ is the diagonal function, seen as a Markov kernel.

Operations, continued

There are also more generic functions, not specifically related to probability theory.

- Domain and codomain maps $\mathrm{dom}, \mathrm{cod} : \mathcal{F} \to \mathit{Borel}$.
- Copy map $\mathrm{copy} : \mathit{Borel} \to \mathcal{F}$, such that for all $X \in \mathit{Borel}$, we have that $\mathrm{copy}(X) : X \to X \times X$ is the diagonal function, seen as a Markov kernel.
- Deletion map $\mathrm{del} : \mathit{Borel} \to \mathcal{F}$, where $\mathrm{del}(X) : X \to \{1\}$ is the constant function, for all X.

Operations, continued

There are also more generic functions, not specifically related to probability theory.

- Domain and codomain maps $\mathrm{dom}, \mathrm{cod} : \mathcal{F} \to \mathit{Borel}$.
- Copy map $\mathrm{copy} : \mathit{Borel} \to \mathcal{F}$, such that for all $X \in \mathit{Borel}$, we have that $\mathrm{copy}(X) : X \to X \times X$ is the diagonal function, seen as a Markov kernel.
- Deletion map $\mathrm{del} : \mathit{Borel} \to \mathcal{F}$, where $\mathrm{del}(X) : X \to \{1\}$ is the constant function, for all X.
- Identity function $\mathit{Borel} \to \mathcal{F}$.

Some first-order properties

Some first-order properties

- $(\mathit{Borel}, \{1\}, \times)$ is a commutative monoid.

Some first-order properties

- $(\mathit{Borel}, \{1\}, \times)$ is a commutative monoid.
  Note: it needed not be commutative on the nose; we might have allowed canonical symmetries
  $\sigma(X, Y) : X \times Y \to Y \times X$.

Some first-order properties

- $(\mathit{Borel}, \{1\}, \times)$ is a commutative monoid.
  Note: it needed not be commutative on the nose; we might have allowed canonical symmetries
  $\sigma(X, Y) : X \times Y \to Y \times X$.
- Associativity of parallel and sequential compositions.

Some first-order properties

- $(\mathit{Borel}, \{1\}, \times)$ is a commutative monoid.
  Note: it needed not be commutative on the nose; we might have allowed canonical symmetries
  $\sigma(X, Y) : X \times Y \to Y \times X$.
- Associativity of parallel and sequential compositions.
- Parallel and sequential composition of kernels are compatible:

Some first-order properties

- $(\mathit{Borel}, \{1\}, \times)$ is a commutative monoid.
  Note: it needed not be commutative on the nose; we might have allowed canonical symmetries
  $\sigma(X, Y) : X \times Y \to Y \times X$.
- Associativity of parallel and sequential compositions.
- Parallel and sequential composition of kernels are compatible: we have $(f; g) \otimes (f'; g') = (f \otimes f'); (g \otimes g')$ whenever these make sense

Some first-order properties

- $(\mathit{Borel}, \{1\}, \times)$ is a commutative monoid.
  Note: it needed not be commutative on the nose; we might have allowed canonical symmetries
  $\sigma(X, Y) : X \times Y \to Y \times X$.
- Associativity of parallel and sequential compositions.
- Parallel and sequential composition of kernels are compatible: we have $(f; g) \otimes (f'; g') = (f \otimes f'); (g \otimes g')$ whenever these make sense, e.g. $f : X \rightsquigarrow Y$, $g : Y \rightsquigarrow Z$, $f' : X' \rightsquigarrow Y'$, $g' : Y' \rightsquigarrow Z'$.

Some first-order properties

- $(\mathit{Borel}, \{1\}, \times)$ is a commutative monoid.
  Note: it needed not be commutative on the nose; we might have allowed canonical symmetries
  $\sigma(X, Y) : X \times Y \to Y \times X$.
- Associativity of parallel and sequential compositions.
- Parallel and sequential composition of kernels are compatible: we have $(f; g) \otimes (f'; g') = (f \otimes f'); (g \otimes g')$ whenever these make sense, e.g. $f : X \rightsquigarrow Y$, $g : Y \rightsquigarrow Z$, $f' : X' \rightsquigarrow Y'$, $g' : Y' \rightsquigarrow Z'$.
- Basic equations involving copy, del, ...

Outline

The Language

Let us now try to generalize from the example above. Our first-order language is defined as follows:

The Language

Let us now try to generalize from the example above. Our first-order language is defined as follows:

- There are two sorts $\mathcal{S}$ and $\mathcal{F}$, for *spaces* and *nondeterministic functions*/generalized Markov kernels, respectively.

The Language

Let us now try to generalize from the example above. Our first-order language is defined as follows:

- There are two sorts $\mathcal{S}$ and $\mathcal{F}$, for *spaces* and *nondeterministic functions*/generalized Markov kernels, respectively. There is a constant symbol $1 \in \mathcal{S}$, which we might also write $*$.

The Language

Let us now try to generalize from the example above. Our first-order language is defined as follows:

- There are two sorts $\mathcal{S}$ and $\mathcal{F}$, for *spaces* and *nondeterministic functions*/generalized Markov kernels, respectively. There is a constant symbol $1 \in \mathcal{S}$, which we might also write $*$.
- We have a ternary relation $R \subseteq \mathcal{F} \times \mathcal{F} \times \mathcal{F}$.

The Language

Let us now try to generalize from the example above. Our first-order language is defined as follows:

- There are two sorts $\mathcal{S}$ and $\mathcal{F}$, for *spaces* and *nondeterministic functions*/generalized Markov kernels, respectively. There is a constant symbol $1 \in \mathcal{S}$, which we might also write $*$.
- We have a ternary relation $R \subseteq \mathcal{F} \times \mathcal{F} \times \mathcal{F}$.
- We have function symbols $\mathrm{dom} : \mathcal{F} \to \mathcal{S}$, $\mathrm{cod} : \mathcal{F} \to \mathcal{S}$, $\mathrm{id} : \mathcal{S} \to \mathcal{F}$,

The Language

Let us now try to generalize from the example above. Our first-order language is defined as follows:

- There are two sorts $\mathcal{S}$ and $\mathcal{F}$, for *spaces* and *nondeterministic functions*/generalized Markov kernels, respectively. There is a constant symbol $1 \in \mathcal{S}$, which we might also write $*$.
- We have a ternary relation $R \subseteq \mathcal{F} \times \mathcal{F} \times \mathcal{F}$.
- We have function symbols $\mathrm{dom} : \mathcal{F} \to \mathcal{S}$, $\mathrm{cod} : \mathcal{F} \to \mathcal{S}$, $\mathrm{id} : \mathcal{S} \to \mathcal{F}$, $\mathrm{copy} : \mathcal{S} \to \mathcal{F}$, $\mathrm{del} : \mathcal{S} \to \mathcal{F}$,

The Language

Let us now try to generalize from the example above. Our first-order language is defined as follows:

- There are two sorts $\mathcal{S}$ and $\mathcal{F}$, for *spaces* and *nondeterministic functions*/generalized Markov kernels, respectively. There is a constant symbol $1 \in \mathcal{S}$, which we might also write $*$.
- We have a ternary relation $R \subseteq \mathcal{F} \times \mathcal{F} \times \mathcal{F}$.
- We have function symbols $\mathrm{dom} : \mathcal{F} \to \mathcal{S}$, $\mathrm{cod} : \mathcal{F} \to \mathcal{S}$, $\mathrm{id} : \mathcal{S} \to \mathcal{F}$, $\mathrm{copy} : \mathcal{S} \to \mathcal{F}$, $\mathrm{del} : \mathcal{S} \to \mathcal{F}$, $\times : \mathcal{S} \times \mathcal{S} \to \mathcal{S}$, $\otimes : \mathcal{F} \times \mathcal{F} \to \mathcal{F}$,

The Language

Let us now try to generalize from the example above. Our first-order language is defined as follows:

- There are two sorts $\mathcal{S}$ and $\mathcal{F}$, for *spaces* and *nondeterministic functions*/generalized Markov kernels, respectively. There is a constant symbol $1 \in \mathcal{S}$, which we might also write $*$.
- We have a ternary relation $R \subseteq \mathcal{F} \times \mathcal{F} \times \mathcal{F}$.
- We have function symbols $\mathrm{dom} : \mathcal{F} \to \mathcal{S}$, $\mathrm{cod} : \mathcal{F} \to \mathcal{S}$, $\mathrm{id} : \mathcal{S} \to \mathcal{F}$, $\mathrm{copy} : \mathcal{S} \to \mathcal{F}$, $\mathrm{del} : \mathcal{S} \to \mathcal{F}$, $\times : \mathcal{S} \times \mathcal{S} \to \mathcal{S}$, $\otimes : \mathcal{F} \times \mathcal{F} \to \mathcal{F}$, $\sigma : \mathcal{S} \times \mathcal{S} \to \mathcal{F}$.

Graphical Notation

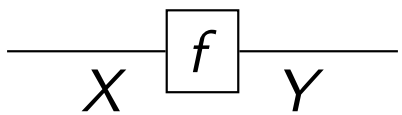
We wish to be able to write down equations for parallel and sequential composites of kernels.

Graphical Notation

We wish to be able to write down equations for parallel and sequential composites of kernels. To that end, the following graphical language will be convenient.

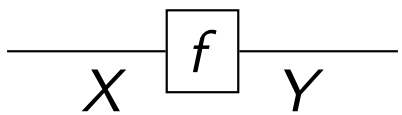
Graphical Notation

We wish to be able to write down equations for parallel and sequential composites of kernels. To that end, the following graphical language will be convenient.
Kernels $f : X \rightsquigarrow Y$ will be denoted as boxes, reading from left to right:

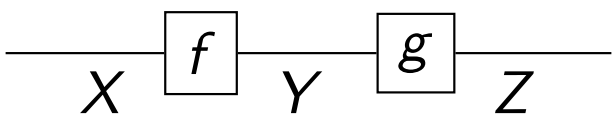


Graphical Notation

We wish to be able to write down equations for parallel and sequential composites of kernels. To that end, the following graphical language will be convenient.
Kernels $f : X \rightsquigarrow Y$ will be denoted as boxes, reading from left to right:



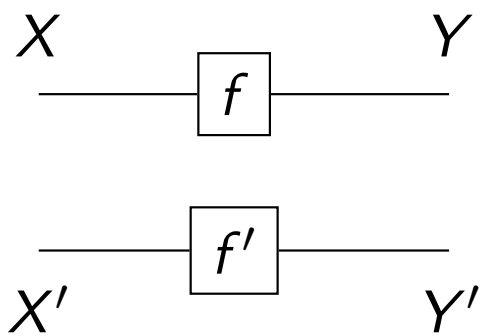
Sequential composites $f ; g$ shall be drawn as



Graphical Notation

Graphical Notation

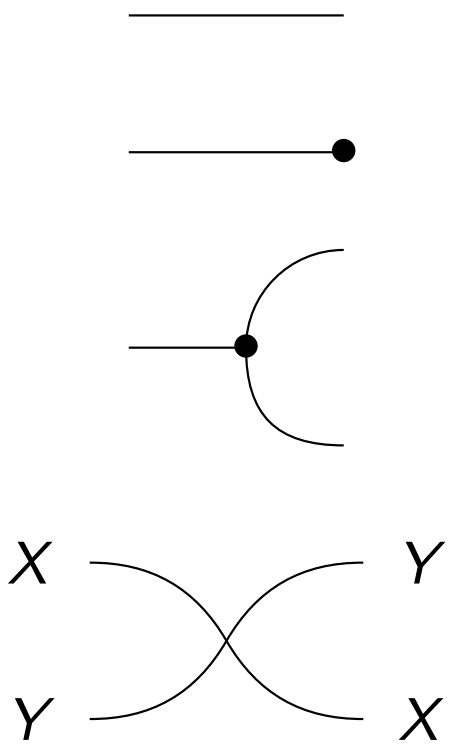
Parallel composites $f \otimes f'$ will be depicted as



Graphical Notation

Graphical Notation

Identities, deleting, copying and symmetry kernels will be drawn as follows:



Axioms

Axioms

- The relation R is the graph of a sequential composition operation denoted ";", that respects dom, cod.

Axioms

- The relation R is the graph of a sequential composition operation denoted ";", that respects dom, cod. Identities are neutral for this composition, and "$\mathrm{id} \otimes \mathrm{id} = \mathrm{id}$".

Axioms

- The relation R is the graph of a sequential composition operation denoted ";", that respects dom, cod. Identities are neutral for this composition, and "$\mathrm{id} \otimes \mathrm{id} = \mathrm{id}$".
- Associativity of $\times$, $\otimes$, and sequential composition.

Axioms

- The relation R is the graph of a sequential composition operation denoted ";", that respects dom, cod. Identities are neutral for this composition, and "$\mathrm{id} \otimes \mathrm{id} = \mathrm{id}$".
- Associativity of $\times$, $\otimes$, and sequential composition.
- The object 1 is the neutral element for $\times$.

Axioms

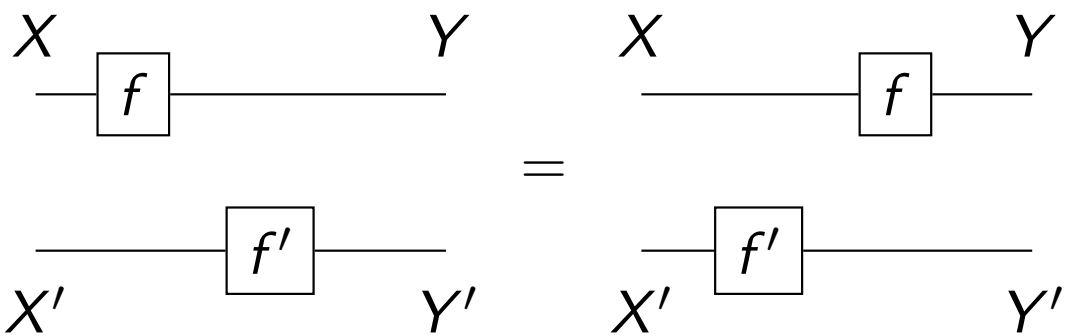
- The relation R is the graph of a sequential composition operation denoted ";", that respects $\mathrm{dom}, \mathrm{cod}$. Identities are neutral for this composition, and "$\mathrm{id} \otimes \mathrm{id} = \mathrm{id}$".
- Associativity of $\times$, $\otimes$, and sequential composition.
- The object 1 is the neutral element for $\times$.
- Compatibility between ; and $\otimes$. We have $(f; g) \otimes (f'; g') = (f \otimes f'); (g \otimes g')$ whenever these make sense.

Axioms

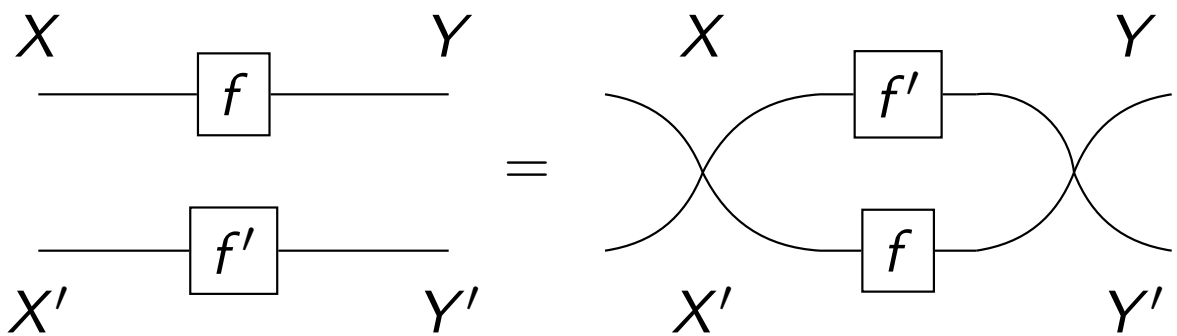
- The relation R is the graph of a sequential composition operation denoted ";", that respects dom, cod. Identities are neutral for this composition, and "$\mathrm{id} \otimes \mathrm{id} = \mathrm{id}$".
- Associativity of $\times$, $\otimes$, and sequential composition.
- The object 1 is the neutral element for $\times$.
- Compatibility between ; and $\otimes$. We have $(f ; g) \otimes (f' ; g') = (f \otimes f') ; (g \otimes g')$ whenever these make sense.

  Note: these axioms are "baked in" the graphical representation.

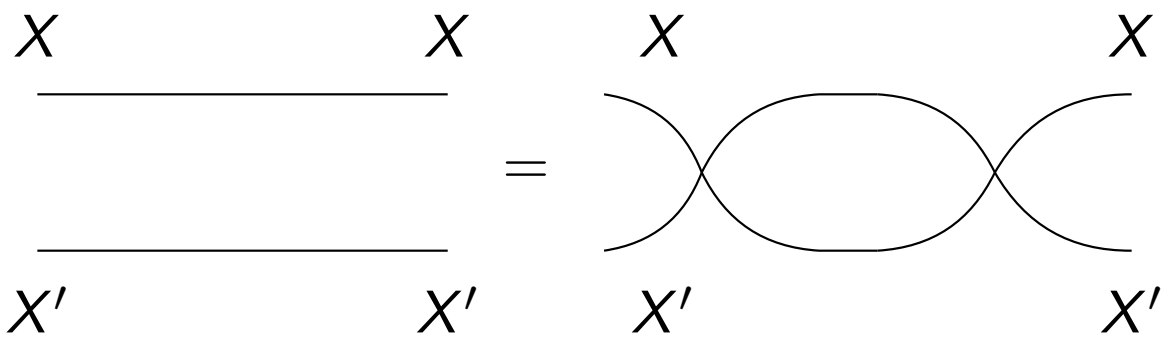
Axioms, graphically



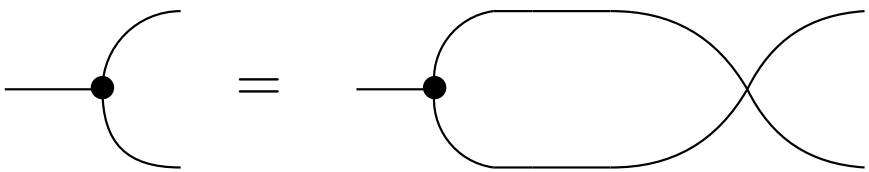
More axioms, graphically

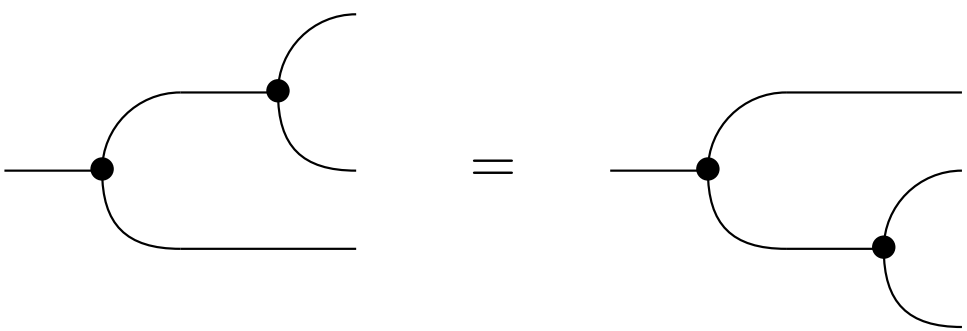


More axioms, graphically

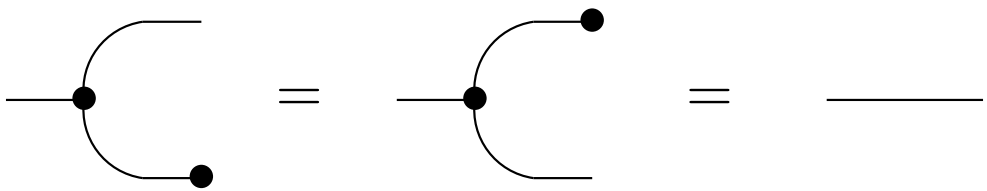


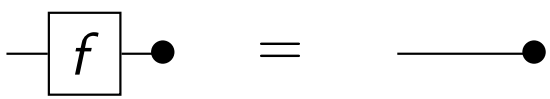
More axioms, graphically



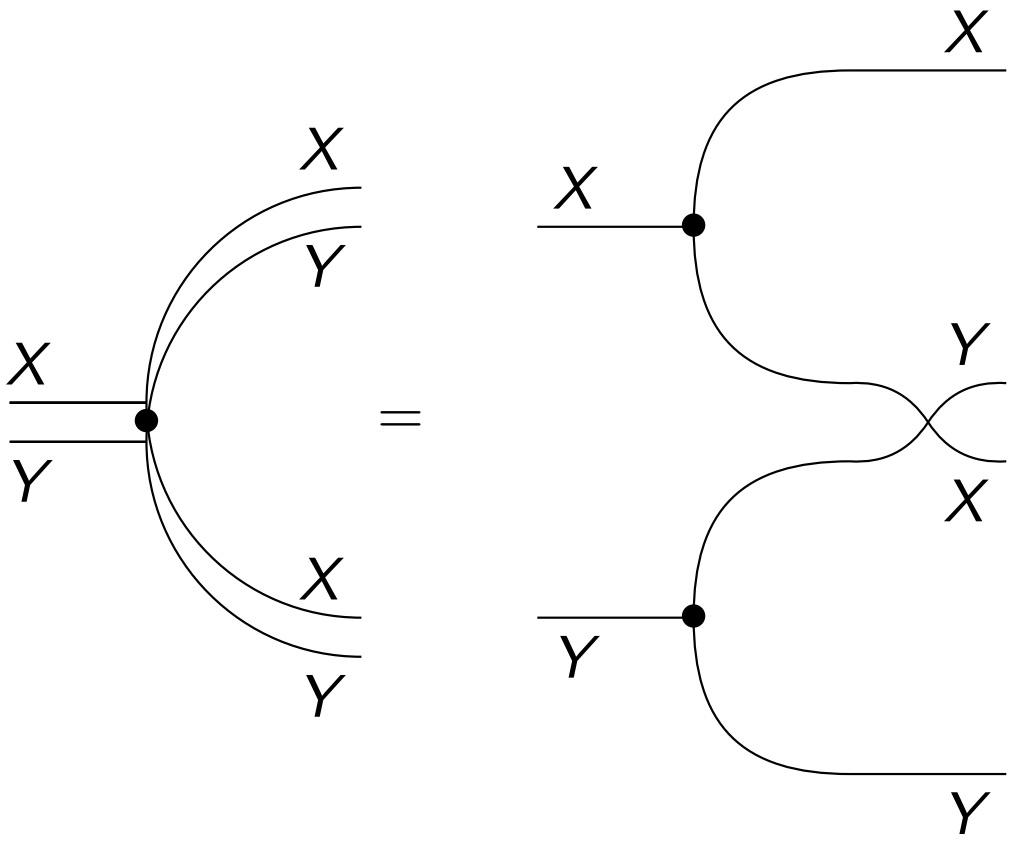


More axioms, graphically





More axioms, graphically



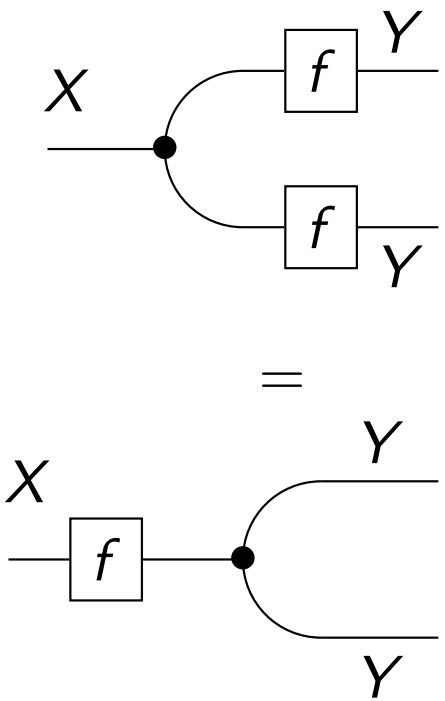
Determinism

Determinism

We say that a generalized Markov kernel $f : X \rightsquigarrow Y$ is deterministic if $\mathrm{copy}(X); (f \otimes f) = f; \mathrm{copy}(Y)$,

Determinism

We say that a generalized Markov kernel $f : X \rightsquigarrow Y$ is deterministic if $\mathrm{copy}(X); (f \otimes f) = f; \mathrm{copy}(Y)$, i.e.



Outline

A Simple Programming Language

A Simple Programming Language

- Goal: define a toy example of nondeterministic programming language, and give semantics in models of the theory above.

A Simple Programming Language

- Goal: define a toy example of nondeterministic programming language, and give semantics in models of the theory above.
- The following was inspired by a paper (Liell-Cock and Staton 2025) on imprecise probability, a more involved notion.

A Simple Programming Language

- Goal: define a toy example of nondeterministic programming language, and give semantics in models of the theory above.
- The following was inspired by a paper (Liell-Cock and Staton 2025) on imprecise probability, a more involved notion.
- We consider a simple first-order functional language without recursion.

A Simple Programming Language

- Goal: define a toy example of nondeterministic programming language, and give semantics in models of the theory above.
- The following was inspired by a paper (Liell-Cock and Staton 2025) on imprecise probability, a more involved notion.
- We consider a simple first-order functional language without recursion. We use a many-sorted point of view.

A Simple Programming Language

- Goal: define a toy example of nondeterministic programming language, and give semantics in models of the theory above.
- The following was inspired by a paper (Liell-Cock and Staton 2025) on imprecise probability, a more involved notion.
- We consider a simple first-order functional language without recursion. We use a many-sorted point of view.
- The language includes sequencing with variable assignements,

A Simple Programming Language

- Goal: define a toy example of nondeterministic programming language, and give semantics in models of the theory above.
- The following was inspired by a paper (Liell-Cock and Staton 2025) on imprecise probability, a more involved notion.
- We consider a simple first-order functional language without recursion. We use a many-sorted point of view.
- The language includes sequencing with variable assignements, of the form $(x \leftarrow t; u)$, where x is a variable, t, u are terms.

A Simple Programming Language

- Goal: define a toy example of nondeterministic programming language, and give semantics in models of the theory above.
- The following was inspired by a paper (Liell-Cock and Staton 2025) on imprecise probability, a more involved notion.
- We consider a simple first-order functional language without recursion. We use a many-sorted point of view.
- The language includes sequencing with variable assignements, of the form $(x \leftarrow t; u)$, where x is a variable, t, u are terms.
- There is a special sort Bool, and a term constructor "If Then Else".

Syntax

$$FV(x \leftarrow t; u) = \begin{cases} FV(u) & \text{if } x \text{ not free in } u \\ FV(t) \cup (FV(u) \setminus \{x\}) & \text{otherwise} \end{cases}$$

Syntax

$$FV(x \leftarrow t; u) = \begin{cases} FV(u) & \text{if } x \text{ not free in } u \\ FV(t) \cup (FV(u) \setminus \{x\}) & \text{otherwise} \end{cases}$$

$$FV(\text{If } b \text{ Then } u \text{ Else } v) = FV(b) \cup FV(u) \cup FV(v).$$

Extending the language

Extending the language

- "Sorts = datatypes",

Extending the language

- "Sorts = datatypes", "Function symbols = primitives",

Extending the language

- "Sorts = datatypes", "Function symbols = primitives", "Constants = probability distributions"

Extending the language

- "Sorts = datatypes", "Function symbols = primitives", "Constants = probability distributions"
- For instance, one might add a sort I (for $[0, 1]$), and a function symbol $\mathrm{bernoulli} : I \to \mathrm{Bool}$.

Desiderata/Axioms

Desiderata/Axioms

- Affine: $(x \leftarrow t; u) = u$, if x not free in u.

Desiderata/Axioms

- Affine: $(x \leftarrow t; u) = u$, if x not free in u.
- Commutativity: $(x \leftarrow t; y \leftarrow u; v) = (y \leftarrow u; x \leftarrow t; v)$ if x not free in u and y not free in t.

Desiderata/Axioms

- Affine: $(x \leftarrow t; u) = u$, if x not free in u.
- Commutativity: $(x \leftarrow t; y \leftarrow u; v) = (y \leftarrow u; x \leftarrow t; v)$ if x not free in u and y not free in t.
- Hoisting equation:

$$\mathrm{If}\, b\, \mathrm{Then}(x \leftarrow t; u)\, \mathrm{Else}(x \leftarrow t; v) = x \leftarrow t; (\mathrm{If}\, b\, \mathrm{Then}\, u\, \mathrm{Else}\, v)$$

Desiderata/Axioms

- Affine: $(x \leftarrow t; u) = u$, if x not free in u.
- Commutativity: $(x \leftarrow t; y \leftarrow u; v) = (y \leftarrow u; x \leftarrow t; v)$
  if x not free in u and y not free in t.
- Hoisting equation:

  $$\mathrm{If}\, b\, \mathrm{Then}(x \leftarrow t; u)\, \mathrm{Else}(x \leftarrow t; v) = x \leftarrow t; (\mathrm{If}\, b\, \mathrm{Then}\, u\, \mathrm{Else}\, v)$$

  if x is not free in b.

A Semantics

A Semantics

- Let $\mathcal{M} = (\mathcal{S}, \mathcal{F}, 1, \ldots)$ be a model of the theory of the previous section.

A Semantics

- Let $\mathcal{M} = (\mathcal{S}, \mathcal{F}, 1, \ldots)$ be a model of the theory of the previous section. Let $S_1, \ldots, S_n \in \mathcal{S}$, and denote $S_1 = B$. Add sorts $S_2, \ldots, S_n$ to the language; recall there is already a sort Bool.

A Semantics

- Let $\mathcal{M} = (\mathcal{S}, \mathcal{F}, 1, \ldots)$ be a model of the theory of the previous section. Let $S_1, \ldots, S_n \in \mathcal{S}$, and denote $S_1 = B$. Add sorts $S_2, \ldots, S_n$ to the language; recall there is already a sort Bool.
- We shall interpret (products of) sorts as sets, and terms as functions.

A Semantics

- Let $\mathcal{M} = (\mathcal{S}, \mathcal{F}, 1, \ldots)$ be a model of the theory of the previous section. Let $S_1, \ldots, S_n \in \mathcal{S}$, and denote $S_1 = B$. Add sorts $S_2, \ldots, S_n$ to the language; recall there is already a sort Bool.
- We shall interpret (products of) sorts as sets, and terms as functions.
- Interpretation: $|S_i| = \{f \in \mathcal{F} \mid f : 1 \rightsquigarrow S_i\}$, for $i = 1, \ldots, n$.

A Semantics

- Let $\mathcal{M} = (\mathcal{S}, \mathcal{F}, 1, \ldots)$ be a model of the theory of the previous section. Let $S_1, \ldots, S_n \in \mathcal{S}$, and denote $S_1 = B$. Add sorts $S_2, \ldots, S_n$ to the language; recall there is already a sort Bool.
- We shall interpret (products of) sorts as sets, and terms as functions.
- Interpretation: $|S_i| = \{f \in \mathcal{F} \mid f : 1 \rightsquigarrow S_i\}$, for $i = 1, \ldots, n$.
- Twist: if $X_1, \ldots X_k$ are sorts in the programming language, we interpret $X_1 \times \cdots \times X_k$ via $|X_1 \times \cdots \times X_k| = \{f \in \mathcal{F} \mid f : 1 \rightsquigarrow X_1 \times \cdots \times X_k\}$

A Semantics

- Let $\mathcal{M} = (\mathcal{S}, \mathcal{F}, 1, \ldots)$ be a model of the theory of the previous section. Let $S_1, \ldots, S_n \in \mathcal{S}$, and denote $S_1 = B$. Add sorts $S_2, \ldots, S_n$ to the language; recall there is already a sort Bool.
- We shall interpret (products of) sorts as sets, and terms as functions.
- Interpretation: $|S_i| = \{f \in \mathcal{F} \mid f : 1 \rightsquigarrow S_i\}$, for $i = 1, \ldots, n$.
- Twist: if $X_1, \ldots X_k$ are sorts in the programming language, we interpret $X_1 \times \cdots \times X_k$ via
  $|X_1 \times \cdots \times X_k| = \{f \in \mathcal{F} \mid f : 1 \rightsquigarrow X_1 \times \cdots \times X_k\}$
  $\neq |X_1| \times \cdots \times |X_k|$.

A Semantics

Assumption

A Semantics

Assumption

We assume that B is equipped with deterministic kernels $\top, \bot : 1 \to B$, that have the following property:

A Semantics

Assumption

We assume that B is equipped with deterministic kernels $\top, \bot : 1 \to B$, that have the following property: for all $X, Y \in \mathcal{S}$, for all $f, g : X = X \times 1 \rightsquigarrow Y$, there exists a unique $h : X \times B \rightsquigarrow Y$ such that $(\mathrm{id}(X) \otimes \top); h = f$ and $(\mathrm{id}(X) \otimes \bot); h = g$.

A Semantics

Assumption

We assume that B is equipped with deterministic kernels $\top, \bot : 1 \to B$, that have the following property: for all $X, Y \in \mathcal{S}$, for all $f, g : X = X \times 1 \rightsquigarrow Y$, there exists a unique $h : X \times B \rightsquigarrow Y$ such that $(\mathrm{id}(X) \otimes \top); h = f$ and $(\mathrm{id}(X) \otimes \bot); h = g$.
We write $h = f + g$.

A Semantics

Assumption

We assume that B is equipped with deterministic kernels $\top, \bot : 1 \to B$, that have the following property: for all $X, Y \in \mathcal{S}$, for all $f, g : X = X \times 1 \rightsquigarrow Y$, there exists a unique $h : X \times B \rightsquigarrow Y$ such that $(\mathrm{id}(X) \otimes \top); h = f$ and $(\mathrm{id}(X) \otimes \bot); h = g$.
We write $h = f + g$.

Proposition

Under this assumption, there exists an (inductive) interpretation for terms of the programming language, that satisfies the desiderata above.

Proof Ideas

Proof Ideas

The interpretation $|(t, u)|$ is equal to

Proof Ideas

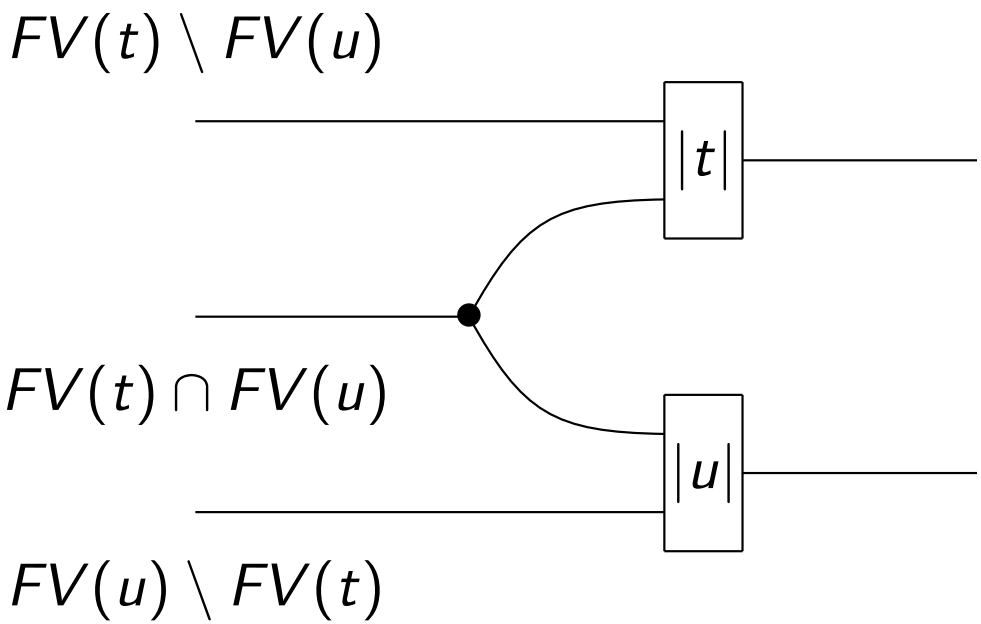
The interpretation $|(t, u)|$ is equal to

$$\mathrm{copy}(FV(t) \cap FV(u)); (|t| \otimes |u|)$$

Proof Ideas

The interpretation $|(t, u)|$ is equal to

$$\mathrm{copy}(FV(t) \cap FV(u)); (|t| \otimes |u|)$$



Proof Ideas

Proof Ideas

The interpretation $|x \leftarrow t; u|$ is equal to

Proof Ideas

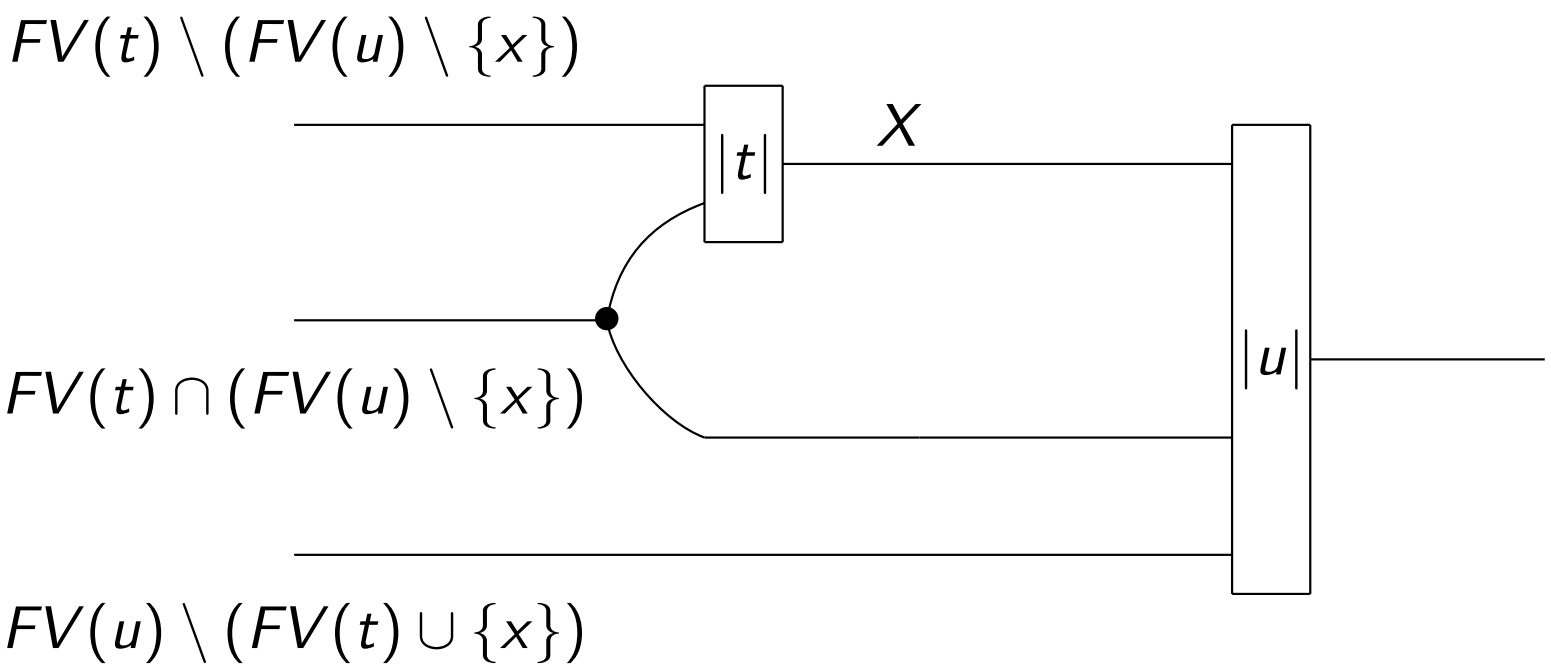
The interpretation $|x \leftarrow t; u|$ is equal to

$$\begin{cases} |u| & \text{if } x \text{ not free in } u \\ (\mathrm{copy}(FV(t) \cap (FV(u) \setminus \{x\})); |t|); |u| & \text{otherwise} \end{cases}$$

Proof Ideas

The interpretation $|x \leftarrow t; u|$ is equal to

$$\begin{cases} |u| & \text{if } x \text{ not free in } u \\ (\mathrm{copy}(FV(t) \cap (FV(u) \setminus \{x\})); |t|); |u| & \text{otherwise} \end{cases}$$



Proof Ideas

Proof Ideas

The interpretation $|\mathrm{If}\ b\ \mathrm{Then}\ u\ \mathrm{Else}\ v|$ is defined as

Proof Ideas

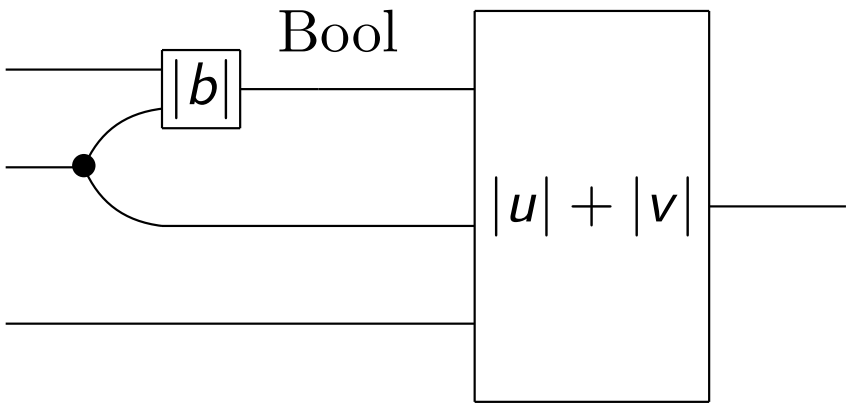
The interpretation $|\,\mathrm{If}\ b\ \mathrm{Then}\ u\ \mathrm{Else}\ v|$ is defined as

$$|\,\mathrm{If}\ b\ \mathrm{Then}\ u\ \mathrm{Else}\ v| = |b|; (|u| + |v|)$$

Proof Ideas

The interpretation $|\,\mathrm{If}\ b\ \mathrm{Then}\ u\ \mathrm{Else}\ v\,|$ is defined as

$$|\,\mathrm{If}\ b\ \mathrm{Then}\ u\ \mathrm{Else}\ v\,| = |b|; (|u| + |v|)$$



Proof Ideas

Showing that the axioms hold with these interpretations amounts to computing in the structure $\mathcal{M}$;

Proof Ideas

Showing that the axioms hold with these interpretations amounts to computing in the structure $\mathcal{M}$; the graphical language is useful for that!

Thank you!

Thank you!

References


Cho, K. and B. Jacobs (2019). "Disintegration and Bayesian inversion via string diagrams". *Mathematical Structures in Computer Science* 29.7.


Fritz, T. (2020). "A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics". *Advances in Mathematics* 370.


Liell-Cock, J. and S. Staton (Jan. 2025). "Compositional Imprecise Probability: A Solution from Graded Monads and Markov Categories". *Proc. ACM Program. Lang.* 9.POPL, 54:1596–54:1626.