An introduction to denotational semantics of logic

Samuel Mimram

December 10, 2020

Introduction

I will try to introduce the concepts of semantics and its uses in logic.

The presentation may be more oriented to combinatorists in the audience: I suppose very little on your background in logic.

You are of course very welcome to ask questions.

What am I computing?

We should remember that programs are *not* computing functions.

It is just a bunch of electric currents going through wires.

What am I computing?

We should remember that programs are *not* computing functions.

It is just a bunch of electric currents going through wires.

In order to give a meaning to those we have to *interpret* them as functions.

What am I computing?

We should remember that programs are *not* computing functions.

It is just a bunch of electric currents going through wires.

In order to give a meaning to those we have to *interpret* them as functions.

Since we have that

$$\text{PROGRAM} = \text{PROOF}$$

(this is the Curry-Howard correspondence) we can play the same game for logics.

The virtuous circle

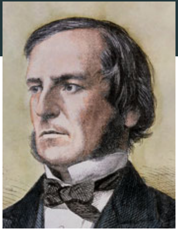
This is quite useful and productive:

- semantics helps to understand better the properties of logics
- logics helps to find the common structures behind the models

Part I

Boolean logic

The boolean semantics of logic

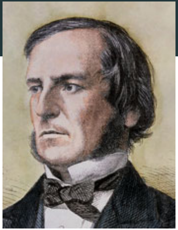


In fact, for most people, logic reduces to a semantical intuition: the boolean interpretation.

A **formula** is either

- X: a variable in a fixed countably infinite set $\mathcal{X}$,
- $A \wedge B$: a conjunction,
- $A \vee B$: a disjunction,
- $\top$: truth,
- $\bot$: falsity,
- $A \Rightarrow B$: an implication,

The boolean semantics of logic



In fact, for most people, logic reduces to a semantical intuition: the boolean interpretation.

A **formula** is either

- X: a variable in a fixed countably infinite set $\mathcal{X}$,
- $A \wedge B$: a conjunction,
- $A \vee B$: a disjunction,
- $\top$: truth,
- $\bot$: falsity,
- $A \Rightarrow B$: an implication,
- $\neg A$: a negation (can be coded by $A \Rightarrow \bot$).

The boolean semantics of logic

A **valuation** is a function $\rho : \mathcal{X} \to \{0, 1\}$.

The boolean semantics of logic

A **valuation** is a function $\rho : \mathcal{X} \to \{0,1\}$.

Given a valuation ρ, we can interpret any formula as a boolean by using the standard interpretation of connectives:

$A \wedge B$	0	1
0	0	0
1	0	1

$A \vee B$	0	1
0	0	1
1	1	1

$A \Rightarrow B$	0	1
0	1	1
1	0	1

$\neg A$	
0	1
1	0

The boolean semantics of logic

A **valuation** is a function $\rho : \mathcal{X} \to \{0,1\}$.

Given a valuation ρ, we can interpret any formula as a boolean by using the standard interpretation of connectives:

$A \wedge B$	0	1
0	0	0
1	0	1

$A \vee B$	0	1
0	0	1
1	1	1

$A \Rightarrow B$	0	1
0	1	1
1	0	1

$\neg A$	
0	1
1	0

For instance, consider ρ such that $\rho(X) = 1$ and $\rho(Y) = 0$.

The boolean semantics of logic

A **valuation** is a function $\rho : \mathcal{X} \to \{0, 1\}$.

Given a valuation ρ, we can interpret any formula as a boolean by using the standard interpretation of connectives:

$A \wedge B$	0	1
0	0	0
1	0	1

$A \vee B$	0	1
0	0	1
1	1	1

$A \Rightarrow B$	0	1
0	1	1
1	0	1

$\neg A$	
0	1
1	0

For instance, consider ρ such that $\rho(X) = 1$ and $\rho(Y) = 0$.

$$(X \Rightarrow Y) \Rightarrow (\neg X \vee Y)$$

The boolean semantics of logic

A **valuation** is a function $\rho : \mathcal{X} \to \{0, 1\}$.

Given a valuation ρ, we can interpret any formula as a boolean by using the standard interpretation of connectives:

$A \wedge B$	0	1
0	0	0
1	0	1

$A \vee B$	0	1
0	0	1
1	1	1

$A \Rightarrow B$	0	1
0	1	1
1	0	1

$\neg A$	
0	1
1	0

For instance, consider ρ such that $\rho(X) = 1$ and $\rho(Y) = 0$.

$$(\underbrace{X}_{1} \Rightarrow \underbrace{Y}_{0}) \Rightarrow (\neg \underbrace{X}_{1} \vee \underbrace{Y}_{0})$$

The boolean semantics of logic

A **valuation** is a function $\rho : \mathcal{X} \to \{0, 1\}$.

Given a valuation ρ, we can interpret any formula as a boolean by using the standard interpretation of connectives:

$A \wedge B$	0	1
0	0	0
1	0	1

$A \vee B$	0	1
0	0	1
1	1	1

$A \Rightarrow B$	0	1
0	1	1
1	0	1

$\neg A$	
0	1
1	0

For instance, consider ρ such that $\rho(X) = 1$ and $\rho(Y) = 0$.

$$(\underbrace{X \Rightarrow Y}_{0}) \Rightarrow (\neg \underbrace{X}_{1} \vee \underbrace{Y}_{0})$$

The boolean semantics of logic

A **valuation** is a function $\rho : \mathcal{X} \to \{0,1\}$.

Given a valuation ρ, we can interpret any formula as a boolean by using the standard interpretation of connectives:

$A \wedge B$	0	1
0	0	0
1	0	1

$A \vee B$	0	1
0	0	1
1	1	1

$A \Rightarrow B$	0	1
0	1	1
1	0	1

$\neg A$	
0	1
1	0

For instance, consider ρ such that $\rho(X) = 1$ and $\rho(Y) = 0$.

$$(\underbrace{X \Rightarrow Y}_{0}) \Rightarrow (\underbrace{\neg X}_{0} \vee \underbrace{Y}_{0})$$

The boolean semantics of logic

A **valuation** is a function $\rho : \mathcal{X} \to \{0, 1\}$.

Given a valuation ρ, we can interpret any formula as a boolean by using the standard interpretation of connectives:

$A \wedge B$	0	1
0	0	0
1	0	1

$A \vee B$	0	1
0	0	1
1	1	1

$A \Rightarrow B$	0	1
0	1	1
1	0	1

$\neg A$	
0	1
1	0

For instance, consider ρ such that $\rho(X) = 1$ and $\rho(Y) = 0$.

$$(\underbrace{X \Rightarrow Y}_{0}) \Rightarrow (\underbrace{\neg X \vee Y}_{0})$$

The boolean semantics of logic

A **valuation** is a function $\rho : \mathcal{X} \to \{0, 1\}$.

Given a valuation ρ, we can interpret any formula as a boolean by using the standard interpretation of connectives:

$A \wedge B$	0	1
0	0	0
1	0	1

$A \vee B$	0	1
0	0	1
1	1	1

$A \Rightarrow B$	0	1
0	1	1
1	0	1

$\neg A$	
0	1
1	0

For instance, consider ρ such that $\rho(X) = 1$ and $\rho(Y) = 0$.

$$\underbrace{(X \Rightarrow Y) \Rightarrow (\neg X \vee Y)}_{1}$$

The boolean semantics of logic

A **valuation** is a function $\rho : \mathcal{X} \to \{0,1\}$.

Given a valuation ρ, we can interpret any formula as a boolean by using the standard interpretation of connectives:

$A \wedge B$	0	1
0	0	0
1	0	1

$A \vee B$	0	1
0	0	1
1	1	1

$A \Rightarrow B$	0	1
0	1	1
1	0	1

$\neg A$	
0	1
1	0

For instance, consider ρ such that $\rho(X) = 1$ and $\rho(Y) = 0$.

$$\underbrace{(X \Rightarrow Y) \Rightarrow (\neg X \vee Y)}_{1}$$

A formula is **valid** when its interpretation is true for every value given to the variables.

Part II

Natural deduction

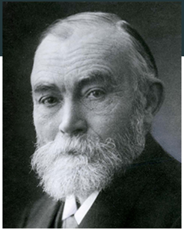
Formalizing logic

People started to formalize the rules of logic:

- we can show meta-theoretic properties of logical systems,
- we can reason on proofs,
- we can build bridges to other things.

Some important advances are

Formalizing logic

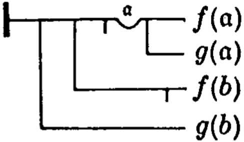


People started to formalize the rules of logic:

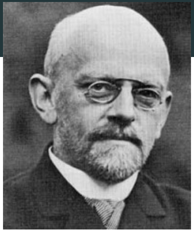
- we can show meta-theoretic properties of logical systems,
- we can reason on proofs,
- we can build bridges to other things.

Some important advances are

- 1880: Frege's *Begriffsschrift*



Formalizing logic



People started to formalize the rules of logic:

- we can show meta-theoretic properties of logical systems,
- we can reason on proofs,
- we can build bridges to other things.

Some important advances are

- 1880: Frege's *Begriffsschrift*
- 1990: Hilbert's quest for foundations of mathematics (following paradoxes such as Russell's)

Formalizing logic

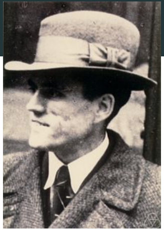
People started to formalize the rules of logic:

- we can show meta-theoretic properties of logical systems,
- we can reason on proofs,
- we can build bridges to other things.

Some important advances are

- 1880: Frege's *Begriffsschrift*
- 1990: Hilbert's quest for foundations of mathematics
- 1920: Brouwer's intuitionism

Formalizing logic



People started to formalize the rules of logic:

- we can show meta-theoretic properties of logical systems,
- we can reason on proofs,
- we can build bridges to other things.

Some important advances are

- 1880: Frege's *Begriffsschrift*
- 1990: Hilbert's quest for foundations of mathematics
- 1920: Brouwer's intuitionism
- 1930: Gentzen's natural deduction

Let's shift from provability to proofs.

Sequents

A **context** Γ is a list of formulas

$$A_1, \ldots, A_n$$

Sequents

A **context** Γ is a list of formulas

$$A_1, \ldots, A_n$$

A **sequent**

$$\Gamma \vdash A$$

consists of a context Γ together with a formula A.

Sequents

A **context** Γ is a list of formulas

$$A_1, \ldots, A_n$$

A **sequent**

$$\Gamma \vdash A$$

consists of a context Γ together with a formula A.

An **inference rule**

$$\frac{\Gamma_1 \vdash A_1 \qquad \ldots \qquad \Gamma_n \vdash A_n}{\Gamma \vdash A}$$

specifies when I can deduce a sequent from others / what I need to show in order to prove a sequent.

Intuitionistic natural deduction (NJ)

$$\frac{}{\Gamma, A, \Gamma' \vdash A}(\mathrm{ax})$$

$$\frac{\Gamma \vdash A \Rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B}(\Rightarrow_{\mathrm{E}}) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}(\Rightarrow_{\mathrm{I}})$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}(\wedge_{\mathrm{E}}^{\mathrm{l}}) \quad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}(\wedge_{\mathrm{E}}^{\mathrm{r}}) \qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}(\wedge_{\mathrm{I}})$$

$$\frac{\Gamma \vdash A \vee B \qquad \Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma \vdash C}(\vee_{\mathrm{E}}) \qquad \frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}(\vee_{\mathrm{I}}^{\mathrm{l}}) \quad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}(\vee_{\mathrm{I}}^{\mathrm{r}})$$

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash A}(\bot_{\mathrm{E}}) \qquad \frac{}{\Gamma \vdash \top}(\top_{\mathrm{I}})$$

Some remarks about the rules

Apart from the axiom, rules are either

- **elimination** rules: use a connective,
- **introduction** rules: show a connective.

Some remarks about the rules

Apart from the axiom, rules are either

- **elimination** rules: use a connective,
- **introduction** rules: show a connective.

The leaves are axiom (or $\top$ introduction) rules: all the other rules have premises.

Some remarks about the rules

Apart from the axiom, rules are either

- **elimination** rules: use a connective,
- **introduction** rules: show a connective.

The leaves are axiom (or $\top$ introduction) rules: all the other rules have premises.

The axiom is the only way to "use" a formula in the context.

Some remarks about the rules

Apart from the axiom, rules are either

- **elimination** rules: use a connective,
- **introduction** rules: show a connective.

The leaves are axiom (or $\top$ introduction) rules: all the other rules have premises.

The axiom is the only way to "use" a formula in the context.

There is no introduction of $\bot$ and elimination of $\top$.

Some remarks about the rules

Apart from the axiom, rules are either

- **elimination** rules: use a connective,
- **introduction** rules: show a connective.

The leaves are axiom (or $\top$ introduction) rules: all the other rules have premises.

The axiom is the only way to "use" a formula in the context.

There is no introduction of $\bot$ and elimination of $\top$.

The **principal** premise is the leftmost premise of an elimination rule.

Proofs

A **proof** is a tree formed with the derivation rules of NJ.

Proofs

A **proof** is a tree formed with the derivation rules of NJ.

A sequent $\Gamma \vdash A$ is **provable** when it is the conclusion of some proof.

Proofs

A **proof** is a tree formed with the derivation rules of NJ.

A sequent $\Gamma \vdash A$ is **provable** when it is the conclusion of some proof.

A formula A is **provable** when the sequent $\vdash A$ is.

A proof of $A \Rightarrow A$

$$\vdash A \Rightarrow A$$

A proof of $A \Rightarrow A$

$$\frac{A \vdash A}{\vdash A \Rightarrow A} \; (\Rightarrow_{\mathrm{I}})$$

A proof of $A \Rightarrow A$

$$\dfrac{\dfrac{}{A \vdash A}\ (\mathsf{ax})}{\vdash A \Rightarrow A}\ (\Rightarrow_{\mathrm{I}})$$

A proof of $A \wedge B \Rightarrow A \vee B$

$$\vdash A \wedge B \Rightarrow A \vee B$$

A proof of $A \wedge B \Rightarrow A \vee B$

$$\frac{A \wedge B \vdash A \vee B}{\vdash A \wedge B \Rightarrow A \vee B}\ (\Rightarrow_{\mathrm{I}})$$

A proof of $A \wedge B \Rightarrow A \vee B$

$$\dfrac{\dfrac{A \wedge B \vdash A}{A \wedge B \vdash A \vee B}\,(\vee^{l}_{I})}{\vdash A \wedge B \Rightarrow A \vee B}\,(\Rightarrow_{I})$$

A proof of $A \wedge B \Rightarrow A \vee B$

$$\dfrac{\dfrac{\dfrac{A \wedge B \vdash A \wedge B}{A \wedge B \vdash A}\,(\wedge_{\mathsf{E}}^{\mathsf{l}})}{A \wedge B \vdash A \vee B}\,(\vee_{\mathsf{I}}^{\mathsf{l}})}{\vdash A \wedge B \Rightarrow A \vee B}\,(\Rightarrow_{\mathsf{I}})$$

A proof of $A \wedge B \Rightarrow A \vee B$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \wedge B \vdash A \wedge B}\ (\mathrm{ax})}{A \wedge B \vdash A}\ (\wedge_{\mathrm{E}}^{\mathrm{l}})}{A \wedge B \vdash A \vee B}\ (\vee_{\mathrm{I}}^{\mathrm{l}})}{\vdash A \wedge B \Rightarrow A \vee B}\ (\Rightarrow_{\mathrm{I}})$$

Correctness of the boolean interpretation

Proposition (Correctness)
The boolean interpretation is correct with respect to the rules:

if A is provable then A is valid.

Correctness of the boolean interpretation

Proposition (Correctness)
The boolean interpretation is correct with respect to the rules:

if A is provable then A is valid.

Proof.
We say that a sequent $\Gamma \vdash A$ is *valid* if A is true for every valuation ρ which makes the formulas in Γ true.

Correctness of the boolean interpretation

Proposition (Correctness)
The boolean interpretation is correct with respect to the rules: if A is provable then A is valid.

Proof.
We say that a sequent $\Gamma \vdash A$ is *valid* if A is true for every valuation ρ which makes the formulas in Γ true.

Show that for every deduction rule if the premises are valid then the conclusion is valid. □

Correctness of the boolean interpretation

Proposition (Correctness)
The boolean interpretation is correct with respect to the rules:

if A is provable then A is valid.

Proof.
We say that a sequent $\Gamma \vdash A$ is *valid* if A is true for every valuation ρ which makes the formulas in Γ true.

Show that for every deduction rule if the premises are valid then the conclusion is valid. □

By contraposition: if A is not valid then it is not provable.

Correctness of the boolean interpretation

An major property of a logical system is that it is **consistency**: there is at least one formula which is not provable.

Correctness of the boolean interpretation

An major property of a logical system is that it is **consistency**: there is at least one formula which is not provable.

Lemma
A logical system is consistent if and only if $\bot$ *is not provable.*

Proof.

Recall that the elimination rule for negation is $\dfrac{\Gamma \vdash \bot}{\Gamma \vdash A}(\bot_E)$. □

Correctness of the boolean interpretation

An major property of a logical system is that it is **consistency**: there is at least one formula which is not provable.

Lemma
A logical system is consistent if and only if $\bot$ is not provable.

Proof.

Recall that the elimination rule for negation is $\dfrac{\Gamma \vdash \bot}{\Gamma \vdash A}(\bot_E)$. □

Proposition
The system NJ is consistent.

Proof.
If $\bot$ was provable then, by correctness, it would be valid wrt the boolean interpretation, which it is not, by definition. □

Completeness of the boolean interpretation

The **completeness** of the boolean interpretation would be the converse: every valid formula is provable.

Completeness of the boolean interpretation

The **completeness** of the boolean interpretation would be the converse: every valid formula is provable.

This is not true: essentially, we miss the fact that $\neg\neg A \Rightarrow A$, more on this later.

Part III

The Curry-Howard correspondence

Types in functional programming languages

In modern languages, everything has a type:

expression	type
3	

Types in functional programming languages

In modern languages, everything has a type:

expression	type
3	`int`

Types in functional programming languages

In modern languages, everything has a type:

expression	type
3	`int`
`true`	

Types in functional programming languages

In modern languages, everything has a type:

expression	type
`3`	`int`
`true`	`bool`

Types in functional programming languages

In modern languages, everything has a type:

expression	type
`3`	`int`
`true`	`bool`
`fun x -> 2 * x`	

Types in functional programming languages

In modern languages, everything has a type:

expression	type
`3`	`int`
`true`	`bool`
`fun x -> 2 * x`	`int -> int`

Types in functional programming languages

In modern languages, everything has a type:

expression	type
`3`	`int`
`true`	`bool`
`fun x -> 2 * x`	`int -> int`
`fun x -> (2 * x, "A")`	

Types in functional programming languages

In modern languages, everything has a type:

expression	type
`3`	`int`
`true`	`bool`
`fun x -> 2 * x`	`int -> int`
`fun x -> (2 * x, "A")`	`int -> int * string`

Types in functional programming languages

In modern languages, everything has a type:

expression	type
`3`	`int`
`true`	`bool`
`fun x -> 2 * x`	`int -> int`
`fun x -> (2 * x, "A")`	`int -> int * string`
`fun x -> (x, "A")`	

Types in functional programming languages

In modern languages, everything has a type:

expression	type
`3`	`int`
`true`	`bool`
`fun x -> 2 * x`	`int -> int`
`fun x -> (2 * x, "A")`	`int -> int * string`
`fun x -> (x, "A")`	`'a -> 'a * string`

Types in functional programming languages

In modern languages, everything has a type:

expression	type
`3`	`int`
`true`	`bool`
`fun x -> 2 * x`	`int -> int`
`fun x -> (2 * x, "A")`	`int -> int * string`
`fun x -> (x, "A")`	`'a -> 'a * string`
`fun x y -> x`	

Types in functional programming languages

In modern languages, everything has a type:

expression	type
`3`	`int`
`true`	`bool`
`fun x -> 2 * x`	`int -> int`
`fun x -> (2 * x, "A")`	`int -> int * string`
`fun x -> (x, "A")`	`'a -> 'a * string`
`fun x y -> x`	`'a -> 'b -> 'a`

Types in functional programming languages

In modern languages, everything has a type:

expression	type
`3`	`int`
`true`	`bool`
`fun x -> 2 * x`	`int -> int`
`fun x -> (2 * x, "A")`	`int -> int * string`
`fun x -> (x, "A")`	`'a -> 'a * string`
`fun x y -> x`	`'a -> 'b -> 'a`
`fun x -> x x`	

Types in functional programming languages

In modern languages, everything has a type:

expression	type
`3`	`int`
`true`	`bool`
`fun x -> 2 * x`	`int -> int`
`fun x -> (2 * x, "A")`	`int -> int * string`
`fun x -> (x, "A")`	`'a -> 'a * string`
`fun x y -> x`	`'a -> 'b -> 'a`
`fun x -> x x`	

```
Error: This expression has type 'a -> 'b but an expression
was expected of type 'a The type variable 'a occurs inside 'a -> 'b
```

Types in functional programming languages

In modern languages, everything has a type:

expression	type
`3`	`int`
`true`	`bool`
`fun x -> 2 * x`	`int -> int`
`fun x -> (2 * x, "A")`	`int -> int * string`
`fun x -> (x, "A")`	`'a -> 'a * string`
`fun x y -> x`	`'a -> 'b -> 'a`
`fun x -> x x`	

Namely, the type of `x` should be of the form `'a -> 'b` with `'a` = (`'a -> 'b`), i.e.

```
((... -> 'b) -> 'b) -> 'b
```

Types in functional programming languages

In good languages, typing is

- **static**: checked at compilation time

Types in functional programming languages

In good languages, typing is

- **static**: checked at compilation time
- **safe**: if a program of a given type produces a values then it is of the expected form

Types in functional programming languages

In good languages, typing is

- **static**: checked at compilation time
- **safe**: if a program of a given type produces a values then it is of the expected form

  e.g. a program of type `int` will produce an integer, typically

  ```
  let n : int = 6 + 2
  ```

Types in functional programming languages

In good languages, typing is

- **static**: checked at compilation time
- **safe**: if a program of a given type produces a values then it is of the expected form

  e.g. a program of type `int` will produce an integer, typically

  ```
  let n : int = 6 + 2
  ```

  but the following is rejected

  ```
  let n : int = 3 + "a"
  ```

Types in functional programming languages

In good languages, typing is

- **static**: checked at compilation time
- **safe**: if a program of a given type produces a values then it is of the expected form

  e.g. a program of type `int` will produce an integer, typically

  ```
  let n : int = 6 + 2
  ```

  but the following is rejected

  ```
  let n : int = 3 + "a"
  ```

  but the following is accepted

  ```
  let rec loop x = loop x

  let n : int = loop "a"
  ```

Simply typed λ-calculus

For simplicity, let us consider a language where **types** are of the form

- constants (e.g. `int`, `bool`, ...),
- $A \to B$: a function taking an A and producing a B,
- $A \times B$: a pair of an A and a B,
- 1: unit.

Simply typed λ-calculus

For simplicity, let us consider a language where **types** are of the form

- constants (e.g. `int`, `bool`, …),
- $A \to B$: a function taking an A and producing a B,
- $A \times B$: a pair of an A and a B,
- 1: unit.

A **terms** t (= a program) is of the form

- a constant (natural numbers, booleans, etc.)
- a variable x,
- $\lambda x^A.t$: the function which to x associates t (in OCaml: `fun (x : A) -> t`),
- $t\,u$: we apply the function t to u,
- $\langle t, u \rangle$: a pair,
- $\pi_l(t)$, $\pi_r(t)$: the projections,
- $\langle\rangle$: unit.

Typing rules

A **context** Γ is a list

$$x_1 : A_1, \ldots, x_n : A_n$$

of pairs consisting of a variable x_i and a type A_i (all the variables we know of).

Typing rules

A **context** Γ is a list

$$x_1 : A_1, \ldots, x_n : A_n$$

of pairs consisting of a variable x_i and a type A_i (all the variables we know of).

A **sequent** is a triple

$$\Gamma \vdash t : A$$

consisting of a context Γ a term t and a type A (a typing judgment).

Typing rules

A **context** Γ is a list

$$x_1 : A_1, \dots, x_n : A_n$$

of pairs consisting of a variable x_i and a type A_i (all the variables we know of).

A **sequent** is a triple

$$\Gamma \vdash t : A$$

consisting of a context Γ a term t and a type A (a typing judgment).

A term t **has type** A when $\vdash t : A$ can be derived using the typing rules.

Typing rules

$$\frac{}{\Gamma, x : A, \Gamma' \vdash x : A}(\text{ax})$$

$$\frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\, u : B}(\to_{\mathsf{E}}) \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x^A.t : A \to B}(\to_{\mathsf{I}})$$

$$\frac{\Gamma \vdash t : A \times B}{\Gamma \vdash \pi_l(t) : A}(\times_{\mathsf{E}}^{\mathsf{l}}) \quad \frac{\Gamma \vdash t : A \times B}{\Gamma \vdash \pi_r(t) : B}(\times_{\mathsf{E}}^{\mathsf{r}}) \qquad \frac{\Gamma \vdash t : A \qquad \Gamma \vdash u : B}{\Gamma \vdash \langle t, u \rangle : A \times B}(\times_{\mathsf{I}})$$

$$\frac{}{\Gamma \vdash \langle\rangle : 1}(1_{\mathsf{I}})$$

A typing example

$$\vdash \lambda f^{A \to A}.\lambda x^{A}.f\,(f\;x) : (A \to A) \to A \to A$$

A typing example

$$\frac{f : A \to A \vdash \lambda x^{A}.f(f\ x) : A \to A}{\vdash \lambda f^{A \to A}.\lambda x^{A}.f(f\ x) : (A \to A) \to A \to A}\ (\to_{\mathsf{I}})$$

A typing example

$$\dfrac{\dfrac{f : A \to A, x : A \vdash f(f\ x) : A}{f : A \to A \vdash \lambda x^A.f(f\ x) : A \to A}\ (\to_{\mathsf{I}})}{\vdash \lambda f^{A \to A}.\lambda x^A.f(f\ x) : (A \to A) \to A \to A}\ (\to_{\mathsf{I}})$$

A typing example

$$
\dfrac{\dfrac{\dfrac{\Gamma \vdash f : A \to A \qquad \Gamma \vdash f\,x : A}{f : A \to A, x : A \vdash f(f\,x) : A}\;(\to_E)}{f : A \to A \vdash \lambda x^A.f(f\,x) : A \to A}\;(\to_I)}{\vdash \lambda f^{A \to A}.\lambda x^A.f(f\,x) : (A \to A) \to A \to A}\;(\to_I)
$$

A typing example

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash f : A \to A}\ (\mathrm{ax}) \qquad \Gamma \vdash f\,x : A}{f : A \to A, x : A \vdash f(f\,x) : A}\ (\to_{\mathrm{E}})}{f : A \to A \vdash \lambda x^{A}.f(f\,x) : A \to A}\ (\to_{\mathrm{I}})}{\vdash \lambda f^{A \to A}.\lambda x^{A}.f(f\,x) : (A \to A) \to A \to A}\ (\to_{\mathrm{I}})$$

A typing example

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{}{\Gamma \vdash f : A \to A}\;(\text{ax})
      \qquad
      \dfrac{\Gamma \vdash f : A \to A \qquad \Gamma \vdash x : A}{\Gamma \vdash f\,x : A}\;(\to_{\mathsf{E}})
    }{f : A \to A, x : A \vdash f(f\,x) : A}\;(\to_{\mathsf{E}})
  }{f : A \to A \vdash \lambda x^{A}.f(f\,x) : A \to A}\;(\to_{\mathsf{I}})
}{\vdash \lambda f^{A \to A}.\lambda x^{A}.f(f\,x) : (A \to A) \to A \to A}\;(\to_{\mathsf{I}})
$$

A typing example

$$
\dfrac{
  \dfrac{}{\Gamma \vdash f : A \to A}\,(\mathrm{ax})
  \qquad
  \dfrac{
    \dfrac{}{\Gamma \vdash f : A \to A}\,(\mathrm{ax})
    \qquad
    \Gamma \vdash x : A
  }{\Gamma \vdash f\,x : A}\,(\to_{\mathrm{E}})
}{
  \dfrac{f : A \to A, x : A \vdash f(f\,x) : A}{
    \dfrac{f : A \to A \vdash \lambda x^{A}.f(f\,x) : A \to A}{\vdash \lambda f^{A\to A}.\lambda x^{A}.f(f\,x) : (A \to A) \to A \to A}\,(\to_{\mathrm{I}})
  }\,(\to_{\mathrm{I}})
}\,(\to_{\mathrm{E}})
$$

A typing example

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{}{\Gamma \vdash f : A \to A}\,(\mathrm{ax})
      \qquad
      \dfrac{
        \dfrac{}{\Gamma \vdash f : A \to A}\,(\mathrm{ax})
        \qquad
        \dfrac{}{\Gamma \vdash x : A}\,(\mathrm{ax})
      }{\Gamma \vdash f\,x : A}\,(\to_{\mathrm{E}})
    }{f : A \to A, x : A \vdash f(f\,x) : A}\,(\to_{\mathrm{E}})
  }{f : A \to A \vdash \lambda x^{A}.f(f\,x) : A \to A}\,(\to_{\mathrm{I}})
}{\vdash \lambda f^{A \to A}.\lambda x^{A}.f(f\,x) : (A \to A) \to A \to A}\,(\to_{\mathrm{I}})
$$

A typing example

$$\dfrac{\dfrac{\overline{f : A \to A, x : A \vdash x : A}\ (\mathsf{ax})}{f : A \to A \vdash \lambda x^A.x : A \to A}\ (\to_\mathsf{I})}{\vdash \lambda f^{A \to A}.\lambda x^A.x : (A \to A) \to A \to A}\ (\to_\mathsf{I})$$

Uniqueness of typing

Note that depending on a term at most one rule applies:

Proposition (Uniqueness of typing)
Given a term t such that $\Gamma \vdash t : A$ and $\Gamma \vdash t : A'$ are derivable then $A = A'$

Proof.
By induction on the derivations. □

Uniqueness of typing

Note that depending on a term at most one rule applies:

Proposition (Uniqueness of typing)
Given a term t *such that* $\Gamma \vdash t : A$ *and* $\Gamma \vdash t : A'$ *are derivable then* $A = A'$ *and given two derivations*

$$\frac{\pi}{\Gamma \vdash t : A} \qquad\qquad \frac{\pi'}{\Gamma \vdash t : A}$$

we have $\pi = \pi'$.

Proof.
By induction on the derivations. □

The Curry-Howard correspondence

A very simple observation is that if we “erase” terms in typing rules and slightly change the notations of connectives

typing	logic
$\to$	$\Rightarrow$
$\times$	$\wedge$
1	$\top$

we obtain the rules of logic, for instance:

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash u : B}{\Gamma \vdash \langle t, u \rangle : A \times B}(\times_\mathrm{I}) \qquad \rightsquigarrow \qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}(\wedge_\mathrm{I})$$

The Curry-Howard correspondence

$$\frac{}{\Gamma, x : A, \Gamma' \vdash x : A}(\mathsf{ax})$$

$$\frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\, u : B}(\to_{\mathsf{E}}) \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x^A . t : A \to B}(\to_{\mathsf{I}})$$

$$\frac{\Gamma \vdash t : A \times B}{\Gamma \vdash \pi_l(t) : A}(\times_{\mathsf{E}}^{\mathsf{l}}) \quad \frac{\Gamma \vdash t : A \times B}{\Gamma \vdash \pi_r(t) : B}(\times_{\mathsf{E}}^{\mathsf{r}}) \qquad \frac{\Gamma \vdash t : A \qquad \Gamma \vdash u : B}{\Gamma \vdash \langle t, u \rangle : A \times B}(\times_{\mathsf{I}})$$

$$\frac{}{\Gamma \vdash \langle\rangle : 1}(1_{\mathsf{I}})$$

The Curry-Howard correspondence

$$\frac{}{\Gamma, A, \Gamma' \vdash A}\,(\mathrm{ax})$$

$$\frac{\Gamma \vdash A \Rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B}\,(\Rightarrow_{\mathrm{E}}) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\,(\Rightarrow_{\mathrm{I}})$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\,(\wedge^{\mathrm{l}}_{\mathrm{E}}) \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\,(\wedge^{\mathrm{r}}_{\mathrm{E}}) \qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\,(\wedge_{\mathrm{I}})$$

$$\frac{}{\Gamma \vdash \top}\,(\top_{\mathrm{I}})$$

The Curry-Howard correspondence

Theorem
There is a bijection between given a context Γ and a type A

- **i.** *terms t such that $\Gamma \vdash t : A$ is derivable,*
- **ii.** *typing derivations $\Gamma \vdash t : A$ for some t,*
- **iii.** *proofs of $\Gamma \vdash A$.*

The Curry-Howard correspondence

Theorem
There is a bijection between given a context Γ and a type A

i. *terms t such that $\Gamma \vdash t : A$ is derivable,*

ii. *typing derivations $\Gamma \vdash t : A$ for some t,*

iii. *proofs of $\Gamma \vdash A$.*

This extends to richer fragments (disjunctions, quantifications, etc.).

The Curry-Howard correspondence

$$\lambda f^{A \to A}.\lambda x^{A}.f\ (f\ x)$$

The Curry-Howard correspondence

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash f : A \to A}\,(\text{ax}) \qquad \dfrac{\dfrac{}{\Gamma \vdash f : A \to A}\,(\text{ax}) \quad \dfrac{}{\Gamma \vdash x : A}\,(\text{ax})}{\Gamma \vdash f\,x : A}\,(\to_{\text{E}})}{f : A \to A, x : A \vdash f\,(f\,x) : A}\,(\to_{\text{E}})}{f : A \to A \vdash \lambda x^{A}.f\,(f\,x) : A \to A}\,(\to_{\text{I}})}{\vdash \lambda f^{A \to A}.\lambda x^{A}.f\,(f\,x) : (A \to A) \to A \to A}\,(\to_{\text{I}})
$$

$$
\dfrac{
  \dfrac{
    \dfrac{}{\Gamma \vdash A \Rightarrow A}\,(\text{ax})
    \qquad
    \dfrac{
      \dfrac{}{\Gamma \vdash A \Rightarrow A}\,(\text{ax})
      \quad
      \dfrac{}{\Gamma \vdash A}\,(\text{ax})
    }{\Gamma \vdash A}\,(\Rightarrow_{\text{E}})
  }{A \Rightarrow A,\ A \vdash A}\,(\Rightarrow_{\text{E}})
}{
  \dfrac{A \Rightarrow A \vdash A \Rightarrow A}{\vdash (A \Rightarrow A) \Rightarrow A \Rightarrow A}\,(\Rightarrow_{\text{I}})
}\,(\Rightarrow_{\text{I}})
$$

Part IV

Semantics of propositional logic

The set-theoretic interpretation

This suggests that the interpretation of formulas as booleans is very poor.

We would rather like to interpret formulas as sets, typically

$$[\![\texttt{int}]\!] = \mathbb{N}$$

The set-theoretic interpretation

This suggests that the interpretation of formulas as booleans is very poor.

We would rather like to interpret formulas as sets, typically

$$[\![\texttt{int}]\!] = \mathbb{N}$$

Suppose fixed an interpretation of base types as above, we extend the interpretation as

- $[\![A \Rightarrow B]\!] = [\![A]\!] \to [\![B]\!]$ is the set of functions from $[\![A]\!]$ to $[\![B]\!]$,
- $[\![A \wedge B]\!] = [\![A]\!] \times [\![B]\!]$ is the cartesian product of $[\![A]\!]$ and $[\![B]\!]$,
- $[\![\top]\!] = \{\star\}$ is the set with one element.

The set-theoretic interpretation

We extend the interpretation to contexts

$$\Gamma = A_1, \ldots, A_n$$

by

$$[\![\Gamma]\!] = [\![A_1]\!] \times \ldots \times [\![A_n]\!]$$

The set-theoretic interpretation

We extend the interpretation to contexts

$$\Gamma = A_1, \ldots, A_n$$

by

$$[\![\Gamma]\!] = [\![A_1]\!] \times \ldots \times [\![A_n]\!]$$

Finally, we extend the interpretation to proofs by

$$[\![\Gamma \vdash t : A]\!] \in [\![\Gamma]\!] \to [\![A]\!]$$

by induction on their derivation.

The set-theoretic interpretation

For instance, suppose that our proof ends with

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash u : B}{\Gamma \vdash \langle t, u \rangle : A \times B}(\times_\text{I})$$

The set-theoretic interpretation

For instance, suppose that our proof ends with

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash u : B}{\Gamma \vdash \langle t, u \rangle : A \times B}(\times_\mathsf{I})$$

By induction hypothesis we have

$$[\![\Gamma \vdash t : A]\!] \in [\![\Gamma]\!] \to [\![A]\!] \qquad\qquad [\![\Gamma \vdash u : B]\!] \in [\![\Gamma]\!] \to [\![B]\!]$$

The set-theoretic interpretation

For instance, suppose that our proof ends with

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash u : B}{\Gamma \vdash \langle t, u \rangle : A \times B}(\times_\mathsf{I})$$

By induction hypothesis we have

$$[\![\Gamma \vdash t : A]\!] \in [\![\Gamma]\!] \to [\![A]\!] \qquad\qquad [\![\Gamma \vdash u : B]\!] \in [\![\Gamma]\!] \to [\![B]\!]$$

and we define

$$\begin{aligned} [\![\Gamma \vdash \langle t, u \rangle : A \times B]\!] : [\![\Gamma]\!] &\to [\![A]\!] \times [\![B]\!] \\ g &\mapsto ([\![\Gamma \vdash t : A]\!](g), [\![\Gamma \vdash u : B]\!](g)) \end{aligned}$$

The set-theoretic interpretation

For instance, suppose that our proof ends with

$$\frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\,u : B}(\to_{\mathsf{E}})$$

The set-theoretic interpretation

For instance, suppose that our proof ends with

$$\frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\,u : B}(\to_{\mathsf{E}})$$

By induction hypothesis we have

$$[\![\Gamma \vdash t : A \to B]\!] \in [\![\Gamma]\!] \to ([\![A]\!] \to [\![B]\!]) \qquad [\![\Gamma \vdash u : A]\!] \in [\![\Gamma]\!] \to [\![A]\!]$$

The set-theoretic interpretation

For instance, suppose that our proof ends with

$$\frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\,u : B}(\to_{\mathsf{E}})$$

By induction hypothesis we have

$$[\![\Gamma \vdash t : A \to B]\!] \in [\![\Gamma]\!] \to ([\![A]\!] \to [\![B]\!]) \qquad [\![\Gamma \vdash u : A]\!] \in [\![\Gamma]\!] \to [\![A]\!]$$

and we define

$$\begin{aligned} [\![\Gamma \vdash t\,u : B]\!] : [\![\Gamma]\!] &\to [\![B]\!] \\ g &\mapsto [\![\Gamma \vdash t : A \to B]\!](g)([\![\Gamma \vdash u : A]\!](g)) \end{aligned}$$

The set-theoretic interpretation

For instance, suppose that our proof ends with

$$\frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\,u : B}(\to_{\mathsf{E}})$$

By induction hypothesis we have

$$[\![\Gamma \vdash t : A \to B]\!] \in [\![\Gamma]\!] \to ([\![A]\!] \to [\![B]\!]) \qquad [\![\Gamma \vdash u : A]\!] \in [\![\Gamma]\!] \to [\![A]\!]$$

and we define

$$\begin{aligned} [\![\Gamma \vdash t\,u : B]\!] : [\![\Gamma]\!] &\to [\![B]\!] \\ g &\mapsto [\![\Gamma \vdash t : A \to B]\!](g)([\![\Gamma \vdash u : A]\!](g)) \end{aligned}$$

Other cases are “similar”.

The set-theoretic interpretation

This can further be extended to other connectives by setting

$$[\![A \vee B]\!] = [\![A]\!] \sqcup [\![B]\!]$$

and

$$[\![\bot]\!] = \emptyset$$

and

$$[\![\neg A]\!] = [\![A \Rightarrow \bot]\!] = [\![A]\!] \to \emptyset$$

The set-theoretic interpretation

We recover the previous interpretation by considering whether a set is empty or not:

$A \Rightarrow B$	$\emptyset$	1
$\emptyset$	1	1
1	$\emptyset$	1

$A \wedge B$	$\emptyset$	1
$\emptyset$	$\emptyset$	$\emptyset$
1	$\emptyset$	1

$A \vee B$	$\emptyset$	1
$\emptyset$	$\emptyset$	1
1	1	1

$\neg A$	
$\emptyset$	1
1	$\emptyset$

We have shifted from provability to proofs!

Classical logic

The interpretation of negation is

$$[\![\neg A]\!] = [\![A \Rightarrow \bot]\!] = [\![A]\!] \to \emptyset$$

Classical logic

The interpretation of negation is

$$[\![\neg A]\!] = [\![A \Rightarrow \bot]\!] = [\![A]\!] \to \emptyset$$

Thus,

$$[\![\neg A]\!] = \begin{cases} \emptyset & \text{if } A \neq \emptyset, \\ \{\star\} & \text{if } A = \emptyset, \end{cases}$$

Classical logic

The interpretation of negation is

$$[\![\neg A]\!] = [\![A \Rightarrow \bot]\!] = [\![A]\!] \to \emptyset$$

Thus,

$$[\![\neg A]\!] = \begin{cases} \emptyset & \text{if } A \neq \emptyset, \\ \{\star\} & \text{if } A = \emptyset, \end{cases} \qquad \text{and} \qquad [\![\neg\neg A]\!] = \begin{cases} \emptyset & \text{if } [\![A]\!] = \emptyset, \\ \{\star\} & \text{if } [\![A]\!] \neq \emptyset. \end{cases}$$

Classical logic

The interpretation of negation is

$$[\![\neg A]\!] = [\![A \Rightarrow \bot]\!] = [\![A]\!] \to \emptyset$$

Thus,

$$[\![\neg A]\!] = \begin{cases} \emptyset & \text{if } A \neq \emptyset, \\ \{\star\} & \text{if } A = \emptyset, \end{cases} \qquad \text{and} \qquad [\![\neg\neg A]\!] = \begin{cases} \emptyset & \text{if } [\![A]\!] = \emptyset, \\ \{\star\} & \text{if } [\![A]\!] \neq \emptyset. \end{cases}$$

We thus understand why we cannot expect to have a proof of

$$\neg\neg A \Rightarrow A$$

in general!

Classical logic

The interpretation of negation is

$$[\![\neg A]\!] = [\![A \Rightarrow \bot]\!] = [\![A]\!] \to \emptyset$$

Thus,

$$[\![\neg A]\!] = \begin{cases} \emptyset & \text{if } A \neq \emptyset, \\ \{\star\} & \text{if } A = \emptyset, \end{cases} \qquad \text{and} \qquad [\![\neg\neg A]\!] = \begin{cases} \emptyset & \text{if } [\![A]\!] = \emptyset, \\ \{\star\} & \text{if } [\![A]\!] \neq \emptyset. \end{cases}$$

We thus understand why we cannot expect to have a proof of

$$\neg\neg A \Rightarrow A$$

in general! And that doubly negated formulas behave as in the boolean interpretation.

Classical logic

If we add the rule

$$\frac{\Gamma \vdash \neg\neg A}{\Gamma \vdash A} \ (\neg\neg_E)$$

we obtain classical logic.

Theorem
The boolean interpretation is correct and complete for classical logic.

Part V

Dynamics

Executing programs

What makes programs interesting is that one can execute them.

Executing programs

What makes programs interesting is that one can execute them.

In λ-calculus, execution is called **β-reduction** and consists in the rule

$$(\lambda x.t)\, u \qquad \longrightarrow_\beta \qquad t[x := u]$$

which can be applied anywhere in a term.

Executing programs

What makes programs interesting is that one can execute them.

In λ-calculus, execution is called **β-reduction** and consists in the rule

$$(\lambda x.t)\,u \qquad \longrightarrow_\beta \qquad t[x := u]$$

which can be applied anywhere in a term.

For instance, if we define

$$\texttt{double} \qquad = \qquad \lambda x.x + x$$

we have

$$\texttt{double}\,5 = (\lambda x.x + x)\,5 \longrightarrow_\beta 5 + 5$$

Executing programs

We should also add rules for products:

$$
\begin{aligned}
(\lambda x.t)\, u &\longrightarrow_\beta t[x := u] \\
\pi_\mathsf{l}\, \langle t, u \rangle &\longrightarrow_\beta t \\
\pi_\mathsf{r}\, \langle t, u \rangle &\longrightarrow_\beta u
\end{aligned}
$$

A **redex** is a subterm which can reduce.

Subject reduction

One of the most important properties of typing systems is called **subject reduction**:

Theorem
If $\Gamma \vdash t : A$ *is derivable and* $t \longrightarrow_\beta t'$ *then* $\Gamma \vdash t' : A$ *is also derivable.*

It means that if a term is check to have type `int`, it will never reduce to `"a"`.

Subject reduction

One of the most important properties of typing systems is called **subject reduction**:

Theorem
If $\Gamma \vdash t : A$ *is derivable and* $t \longrightarrow_\beta t'$ *then* $\Gamma \vdash t' : A$ *is also derivable.*

Proof.
Transform the proof of $\Gamma \vdash t : A$ into a proof of $\Gamma \vdash t' : A$. □

It means that if a term is check to have type `int`, it will never reduce to `"a"`.

Subject reduction: proof

Suppose that $\Gamma \vdash t : A$ and $t \longrightarrow_\beta t'$ using the rule

$$\pi_\mathsf{l}\langle u, v\rangle \longrightarrow_\beta u$$

Subject reduction: proof

Suppose that $\Gamma \vdash t : A$ and $t \longrightarrow_\beta t'$ using the rule

$$\pi_\text{l}\langle u, v\rangle \longrightarrow_\beta u$$

This means that the typing derivation of t will contain a subproof of the form

$$\Gamma \vdash \pi_\text{l}\langle u, v\rangle : A$$

Subject reduction: proof

Suppose that $\Gamma \vdash t : A$ and $t \longrightarrow_\beta t'$ using the rule

$$\pi_\mathsf{l}\langle u, v\rangle \longrightarrow_\beta u$$

This means that the typing derivation of t will contain a subproof of the form

$$\frac{\Gamma \vdash \langle u, v\rangle : A \times B}{\Gamma \vdash \pi_\mathsf{l}\langle u, v\rangle : A}\ (\times_\mathsf{E}^\mathsf{l})$$

Subject reduction: proof

Suppose that $\Gamma \vdash t : A$ and $t \longrightarrow_\beta t'$ using the rule

$$\pi_\mathsf{l}\langle u, v\rangle \longrightarrow_\beta u$$

This means that the typing derivation of t will contain a subproof of the form

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma \vdash u : A} \qquad \dfrac{\vdots}{\Gamma \vdash v : B}}{\Gamma \vdash \langle u, v\rangle : A \times B}\,(\times_\mathsf{I})}{\Gamma \vdash \pi_\mathsf{l}\langle u, v\rangle : A}\,(\times_\mathsf{E}^\mathsf{l})$$

Subject reduction: proof

Suppose that $\Gamma \vdash t : A$ and $t \longrightarrow_\beta t'$ using the rule

$$\pi_\mathsf{l}\langle u, v\rangle \longrightarrow_\beta u$$

This means that the typing derivation of t will contain a subproof of the form

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma \vdash u : A} \qquad \dfrac{\vdots}{\Gamma \vdash v : B}}{\Gamma \vdash \langle u, v\rangle : A \times B}\,(\times_\mathsf{I})}{\Gamma \vdash \pi_\mathsf{l}\langle u, v\rangle : A}\,(\times_\mathsf{E}^\mathsf{l}) \qquad \rightsquigarrow \qquad \dfrac{\vdots}{\Gamma \vdash u : A}$$

Subject reduction: proof

Suppose that $\Gamma \vdash t : A$ and $t \longrightarrow_\beta t'$ using the rule

$$\pi_\mathsf{r}\langle u, v\rangle \longrightarrow_\beta v$$

This means that the typing derivation of t will contain a subproof of the form

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma \vdash u : A} \qquad \dfrac{\vdots}{\Gamma \vdash v : B}}{\Gamma \vdash \langle u, v\rangle : A \times B}\,(\times_\mathsf{I})}{\Gamma \vdash \pi_\mathsf{r}\langle u, v\rangle : A}\,(\times_\mathsf{E}^\mathsf{r}) \qquad \rightsquigarrow \qquad \dfrac{\vdots}{\Gamma \vdash v : B}$$

Subject reduction: proof

Suppose that $\Gamma \vdash t : A$ and $t \longrightarrow_\beta t'$ using the rule

$$\pi_{\mathsf{r}}\langle u, v\rangle \longrightarrow_\beta v$$

This means that the typing derivation of t will contain a subproof of the form

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma \vdash u : A} \qquad \dfrac{\vdots}{\Gamma \vdash v : B}}{\Gamma \vdash \langle u, v\rangle : A \times B}\,(\times_{\mathsf{I}})}{\Gamma \vdash \pi_{\mathsf{r}}\langle u, v\rangle : A}\,(\times_{\mathsf{E}}^{\mathsf{r}}) \qquad \rightsquigarrow \qquad \dfrac{\vdots}{\Gamma \vdash v : B}$$

There is a "similar" transformation in the case where the rule is

$$(\lambda x.t)u \longrightarrow_\beta t[x := u]$$

Cut elimination

By the Curry-Howard correspondence, we can interpret these operations as proof transformations:

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma \vdash u : A} \qquad \dfrac{\vdots}{\Gamma \vdash v : B}}{\Gamma \vdash \langle u, v \rangle : A \times B}\ (\times_\mathsf{I})}{\Gamma \vdash \pi_\mathsf{l}\langle u, v \rangle : A}\ (\times_\mathsf{E}^\mathsf{l}) \qquad \rightsquigarrow \qquad \dfrac{\vdots}{\Gamma \vdash u : A}$$

Cut elimination

By the Curry-Howard correspondence, we can interpret these operations as proof transformations:

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma \vdash A} \qquad \dfrac{\vdots}{\Gamma \vdash B}}{\Gamma \vdash A \wedge B}\,(\wedge_{\mathrm{I}})}{\Gamma \vdash A}\,(\wedge_{\mathrm{E}}^{\mathrm{l}}) \qquad \rightsquigarrow \qquad \dfrac{\vdots}{\Gamma \vdash A}$$

Such a situation is called a **cut** (= introduction and elimination of a connective) and this transformation is called **cut elimination**.

Cut elimination

In the case of $\Rightarrow$, it relies on the fact that the following rule is admissible:

$$\frac{\Gamma \vdash A \qquad \Gamma, A \vdash B}{\Gamma \vdash B} \text{ (cut)}$$

which corresponds to preservation of typing under substitutions.

Cut elimination

In the case of $\Rightarrow$, it relies on the fact that the following rule is admissible:

$$\frac{\Gamma \vdash A \qquad \Gamma, A \vdash B}{\Gamma \vdash B}\ (\text{cut})$$

which corresponds to preservation of typing under substitutions.

Note that this rule has the following particular case:

$$\frac{A \vdash B \qquad B \vdash C}{A \vdash C}\ (\text{cut})$$

Cut-elimination

The major theorem of proof theory:

Theorem (Strong normalization)
The reduction of any typable term $\Gamma \vdash t : A$ *eventually stops.*

Cut-elimination

The major theorem of proof theory:

Theorem (Strong normalization)
The reduction of any typable term $\Gamma \vdash t : A$ *eventually stops.*

Theorem (Cut elimination)
If a sequent $\Gamma \vdash A$ *is provable then it admits a proof without cut.*

Cut-elimination

The major theorem of proof theory:

Theorem (Strong normalization)
The reduction of any typable term $\Gamma \vdash t : A$ *eventually stops.*

Theorem (Cut elimination)
If a sequent $\Gamma \vdash A$ *is provable then it admits a proof without cut.*

Proof.
Apply previous theorem through Curry-Howard. □

Cut-elimination

The major theorem of proof theory:

Theorem (Strong normalization)
The reduction of any typable term $\Gamma \vdash t : A$ *eventually stops.*

Theorem (Cut elimination)
If a sequent $\Gamma \vdash A$ *is provable then it admits a proof without cut.*

Proof.
Apply previous theorem through Curry-Howard. □

There are wonderful consequences of this (coherence, improvability of $\neg\neg A \Rightarrow A$, etc.) but this will be for another time.

Correctness of set-theoretic semantics

Theorem
The set-theoretic semantics is correct, *in the sense that it is invariant under reduction (= cut elimination): given a derivation of* $\Gamma \vdash t : A$ *if* $t \longrightarrow_\beta t'$ *then*

$$[\![\Gamma \vdash t : A]\!] \quad = \quad [\![\Gamma \vdash t' : A]\!]$$

This can be seen as a modularity principle: the behavior of the whole program can be determined from its components only (and not the interactions they can have).

Denotational semantics

Definition
A semantics of a programming language is **denotational** when it is invariant under reduction (and "co-reduction" / extensionality).

Denotational semantics

Definition
A semantics of a programming language is **denotational** when it is invariant under reduction (and "co-reduction" / extensionality).

There are many alternatives to plain sets and functions:

- we might want to model more features:
  fixpoints [`while`] (domains), equality (spaces / HoTT), etc.
- we might want to capture more precisely the language:
  the interactive behavior (game semantics), etc.
- we might want to remove all functions which are not interpretations of programs.

η-reduction

The "co-cuts" correspond to η-**reduction** which express some form of extensionality

$$\lambda x.tx \longrightarrow_\eta t \qquad\qquad \langle \pi_{\mathsf{l}} t, \pi_{\mathsf{r}} t \rangle \longrightarrow_\eta t$$

η-reduction

The "co-cuts" correspond to η-**reduction** which express some form of extensionality

$$\lambda x.tx \longrightarrow_\eta t \qquad\qquad \langle \pi_l t, \pi_r t\rangle \longrightarrow_\eta t$$

On the typing side:

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma \vdash t : A \times B}}{\Gamma \vdash \pi_l t : A}\,(\times_E^l) \qquad \dfrac{\dfrac{\vdots}{\Gamma \vdash t : A \times B}}{\Gamma \vdash \pi_r t : B}\,(\times_E^r)}{\Gamma \vdash \langle \pi_l t, \pi_r t\rangle : A \times B}\,(\times_I) \quad \rightsquigarrow \quad \dfrac{\vdots}{\Gamma \vdash t : A \times B}$$

Part VI

Categorical semantics

Categorical axiomatization

Instead of checking each time that the model is denotational, people have studied categorical axiomatizations.

We define algebraic structures on categories which ensure that

- *operations*: there is a canonical interpretation of proofs in those categories,
- *axioms*: the interpretation of proofs is invariant under reduction.

Categories

A **category** $\mathcal{C}$ consists of

- a set of objects $A, B, \ldots$
- a set of morphisms $\mathcal{C}(A, B)$ for every objects A and B
- a composition operations and identities

such that

- composition is associative: $h \circ (g \circ f) = (h \circ g) \circ f$
- identities are neutral elements: $\mathrm{id} \circ f = f = f \circ \mathrm{id}$.

Typically:

- the category Set of sets and functions,
- Top, Vect, ...

Categories

The identity will correspond to the interpretation of

$$\dfrac{}{A \vdash A}\ (\text{ax})$$

and composition to

$$\dfrac{A \vdash B \qquad B \vdash C}{A \vdash C}\ (\text{cut})$$

Categories

The identity will correspond to the interpretation of

$$\frac{}{A \vdash A}\ (\mathrm{ax})$$

and composition to

$$\frac{A \vdash B \qquad B \vdash C}{A \vdash C}\ (\mathrm{cut})$$

The axioms of categories will ensure that the interpretation is invariant under cut-elimination.

Cartesian categories

In order to interpret conjunction, we need for every objects A and B, an object $A \times B$:

$$[\![A \wedge B]\!] = [\![A]\!] \times [\![B]\!]$$

Cartesian categories

In order to interpret conjunction, we need for every objects A and B, an object $A \times B$:

$$[\![A \wedge B]\!] = [\![A]\!] \times [\![B]\!]$$

We also need "projection" morphisms:

$$\pi_{A,B} : A \times B \to A \qquad \pi'_{A,B} : A \times B \to B$$

which interpret

$$\dfrac{\dfrac{}{A \wedge B \vdash A \wedge B}\,(\mathrm{ax})}{A \wedge B \vdash A}\,(\wedge^{\mathrm{l}}_{\mathrm{E}}) \qquad \dfrac{\dfrac{}{A \wedge B \vdash A \wedge B}\,(\mathrm{ax})}{A \wedge B \vdash B}\,(\wedge^{\mathrm{r}}_{\mathrm{E}})$$

Cartesian categories

Given morphisms $f : C \to A$ and $g : C \to B$, we need a morphism $\langle f, g \rangle : C \to A \times B$:

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ (\wedge_\mathsf{I})$$

Cartesian categories

Given morphisms $f : C \to A$ and $g : C \to B$, we need a morphism $\langle f, g \rangle : C \to A \times B$:

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ (\wedge_\mathrm{I})$$

Moreover, invariance under cut elimination

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma \vdash A} \qquad \dfrac{\vdots}{\Gamma \vdash B}}{\Gamma \vdash A \wedge B}\ (\wedge_\mathrm{I})}{\Gamma \vdash A}\ (\wedge_\mathrm{E}^\mathrm{l}) \qquad \rightsquigarrow \qquad \dfrac{\vdots}{\Gamma \vdash A}$$

will impose

$$\pi \circ \langle f, g \rangle \qquad = \qquad f$$

Cartesian categories

Given morphisms $f : C \to A$ and $g : C \to B$, we need a morphism $\langle f, g \rangle : C \to A \times B$:

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\,(\wedge_I)$$

Moreover, invariance under cut elimination

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma \vdash A} \qquad \dfrac{\vdots}{\Gamma \vdash B}}{\Gamma \vdash A \wedge B}\,(\wedge_I)}{\Gamma \vdash A}\,(\wedge_E^l) \qquad \rightsquigarrow \qquad \dfrac{\vdots}{\Gamma \vdash A}$$

will impose

$$\begin{aligned} \pi \circ \langle f, g \rangle &= f \\ \pi' \circ \langle f, g \rangle &= g \end{aligned}$$

Cartesian categories

Given morphisms $f : C \to A$ and $g : C \to B$, we need a morphism $\langle f, g\rangle : C \to A \times B$:

$$\dfrac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ (\wedge_\mathsf{I})$$

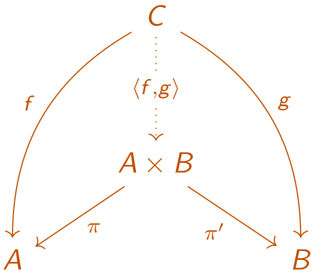
Moreover, invariance under cut elimination

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma \vdash A} \qquad \dfrac{\vdots}{\Gamma \vdash B}}{\Gamma \vdash A \wedge B}\ (\wedge_\mathsf{I})}{\Gamma \vdash A}\ (\wedge_\mathsf{E}^\mathsf{l}) \qquad \rightsquigarrow \qquad \dfrac{\vdots}{\Gamma \vdash A}$$

will impose

$$\begin{aligned}
\pi \circ \langle f, g\rangle &= f \\
\pi' \circ \langle f, g\rangle &= g \\
\langle \pi \circ f, \pi' \circ f\rangle &= f
\end{aligned}$$

A **cartesian product** of A and B in a category $\mathcal{C}$ is



Cartesian closed categories

In order to interpret our fragment of logic we need a **cartesian closed category**:

- a *category*

Cartesian closed categories

In order to interpret our fragment of logic we need a **cartesian closed category**:

- a *category*
- with *cartesian products*

Cartesian closed categories

In order to interpret our fragment of logic we need a **cartesian closed category**:

- a *category*
- with *cartesian products*
- with a *terminal object* 1 to interpret $\top$:

$$A \dashrightarrow 1$$

Cartesian closed categories

In order to interpret our fragment of logic we need a **cartesian closed category**:

- a *category*
- with *cartesian products*
- with a *terminal object* 1 to interpret $\top$:

$$A \dashrightarrow 1$$

- with an *exponential closure* to interpret $\Rightarrow$:

$$\mathcal{C}(A \times B, C) \simeq \mathcal{C}(A, B \to C)$$

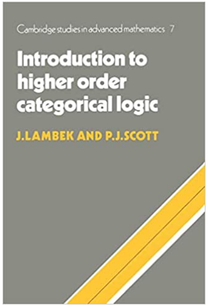
Cartesian closed categories

Theorem (Soundness)
We have a denotational semantics of our logic in every cartesian closed category.

Cartesian closed categories

Theorem (Soundness)
We have a denotational semantics of our logic in every cartesian closed category.

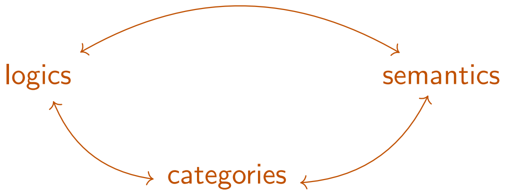
Theorem (Completeness)
Every categorical model of our logic has to be cartesian closed.



Cartesian closed categories

Theorem (Soundness)
We have a denotational semantics of our logic in every cartesian closed category.

Theorem (Completeness)
Every categorical model of our logic has to be cartesian closed.



Part VII

Linear logic

Reasoning about resources

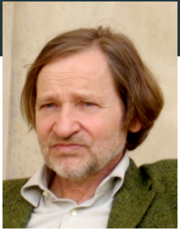
A λ-term can use its argument many times:

$$\lambda x.x + x$$

including zero times:

$$\lambda x.0$$

Reasoning about resources



A λ-term can use its argument many times:

$$\lambda x.x + x$$

including zero times:

$$\lambda x.0$$

Linear logic: we would like to distinguish between variables which can be used exactly once or not in order to

- take resources in account,
- take complexity in account,
- have a fine grained understanding of logic.

The linear λ-calculus

Let's try to modify the rules for the λ-calculus in order to capture **linear λ-terms** only, where each variable is used exactly once.

The linear λ-calculus

Let's try to modify the rules for the λ-calculus in order to capture **linear λ-terms** only, where each variable is used exactly once.

$\lambda x.\lambda y.\langle x, y\rangle$ $\quad$ $\lambda x.\lambda y.yx$ $\quad$ $\lambda x.\lambda y.x$ $\quad$ $\lambda x.\lambda y.xyy$

The linear λ-calculus

Let's try to modify the rules for the λ-calculus in order to capture **linear λ-terms** only, where each variable is used exactly once.

$\lambda x.\lambda y.\langle x, y\rangle$ — linear

$\lambda x.\lambda y.yx$

$\lambda x.\lambda y.x$

$\lambda x.\lambda y.xyy$

The linear λ-calculus

Let's try to modify the rules for the λ-calculus in order to capture **linear λ-terms** only, where each variable is used exactly once.

$\lambda x.\lambda y.\langle x, y\rangle$	$\lambda x.\lambda y.yx$	$\lambda x.\lambda y.x$	$\lambda x.\lambda y.xyy$
linear	linear		

The linear λ-calculus

Let's try to modify the rules for the λ-calculus in order to capture **linear λ-terms** only, where each variable is used exactly once.

$\lambda x.\lambda y.\langle x, y\rangle$	$\lambda x.\lambda y.yx$	$\lambda x.\lambda y.x$	$\lambda x.\lambda y.xyy$
linear	linear	not linear	

The linear λ-calculus

Let's try to modify the rules for the λ-calculus in order to capture **linear λ-terms** only, where each variable is used exactly once.

$\lambda x.\lambda y.\langle x, y\rangle$	$\lambda x.\lambda y.yx$	$\lambda x.\lambda y.x$	$\lambda x.\lambda y.xyy$
linear	linear	not linear	not linear

The linear λ-calculus

Let's try to modify the rules for the λ-calculus in order to capture **linear λ-terms** only, where each variable is used exactly once.

$\lambda x.\lambda y.\langle x, y\rangle$	$\lambda x.\lambda y.yx$	$\lambda x.\lambda y.x$	$\lambda x.\lambda y.xyy$
linear	linear	not linear	not linear

The main idea is that in sequents, the context Γ maintains the variables we are aware of, which can be thought of as resources that we should handle carefully.

Contraction

A rule is **admissible** when if we can derive the premises then we can derive the conclusion.

Contraction

A rule is **admissible** when if we can derive the premises then we can derive the conclusion.

Lemma
The following contraction *rule is admissible:*

$$\frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}$$

Contraction

Lemma
The following contraction *rule is admissible:*

$$\frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}$$

Categorically,

$$A \times A \xrightarrow{\quad f \quad} B$$

Contraction

Lemma
The following contraction *rule is admissible:*

$$\frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}$$

Categorically,

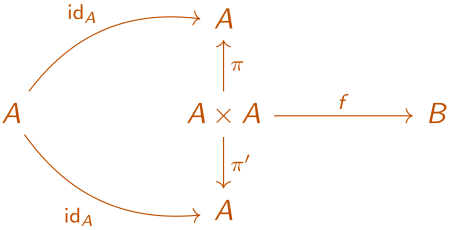
$$\begin{array}{ccc} A & & \\ \uparrow{\scriptstyle \pi} & & \\ A \times A & \xrightarrow{f} & B \\ \downarrow{\scriptstyle \pi'} & & \\ A & & \end{array}$$

Contraction

Lemma
The following contraction *rule is admissible:*

$$\frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}$$

Categorically,

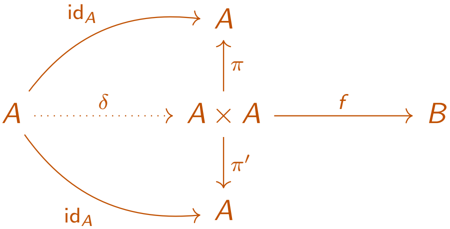


Contraction

Lemma
The following contraction *rule is admissible:*

$$\frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}$$

Categorically,



In sets: $\delta(x) = (x, x)$.

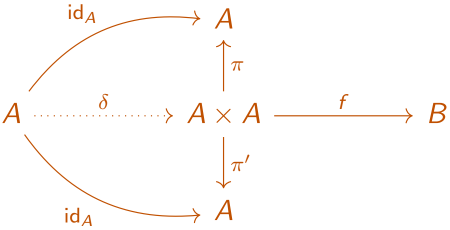
This means that from $(x, y) \mapsto f(x, y)$, we can construct $x \mapsto f(x, x)$!

Contraction

Lemma
The following contraction *rule is admissible:*

$$\frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}$$

Categorically,



Closely related to the fact that we have the proof $\dfrac{\dfrac{}{A \vdash A}\,(\text{ax}) \qquad \dfrac{}{A \vdash A}\,(\text{ax})}{A \vdash A \wedge A}\,(\wedge_\mathrm{I})$.

Tensor

In linear logic, the traditional *additive* conjunction is noted $\&$

$$\frac{\Gamma \vdash A \& B}{\Gamma \vdash A}\ (\&_E^l) \qquad \frac{\Gamma \vdash A \& B}{\Gamma \vdash B}\ (\&_E^r) \qquad\qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \& B}\ (\&_I)$$

Tensor

In linear logic, the traditional *additive* conjunction is noted $\&$

$$\frac{\Gamma \vdash A \& B}{\Gamma \vdash A}\ (\&_E^l) \qquad \frac{\Gamma \vdash A \& B}{\Gamma \vdash B}\ (\&_E^r) \qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \& B}\ (\&_I)$$

It allows deducing

$$\frac{\dfrac{}{A \vdash A}\ (\mathrm{ax}) \qquad \dfrac{}{A \vdash A}\ (\mathrm{ax})}{A \vdash A \& A}\ (\&_I)$$

Tensor

In linear logic, the traditional *additive* conjunction is noted $\&$

$$\frac{\Gamma \vdash A \& B}{\Gamma \vdash A}\ (\&_E^l) \qquad \frac{\Gamma \vdash A \& B}{\Gamma \vdash B}\ (\&_E^r) \qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \& B}\ (\&_I)$$

It allows deducing

$$\frac{\dfrac{}{A \vdash A}\ (\mathrm{ax}) \qquad \dfrac{}{A \vdash A}\ (\mathrm{ax})}{A \vdash A \& A}\ (\&_I)$$

We want to replace it by a *multiplicative* conjunction $\otimes$ which does not allow this

$$\frac{\Gamma \vdash A \qquad \Gamma' \vdash B}{\Gamma, \Gamma' \vdash A \otimes B}\ (\otimes_I)$$

Tensor

In linear logic, the traditional *additive* conjunction is noted $\&$

$$\frac{\Gamma \vdash A \& B}{\Gamma \vdash A}\,(\&^l_E) \qquad \frac{\Gamma \vdash A \& B}{\Gamma \vdash B}\,(\&^r_E) \qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \& B}\,(\&_I)$$

It allows deducing

$$\frac{\dfrac{}{A \vdash A}\,(\text{ax}) \qquad \dfrac{}{A \vdash A}\,(\text{ax})}{A \vdash A \& A}\,(\&_I)$$

We want to replace it by a *multiplicative* conjunction $\otimes$ which does not allow this

$$\frac{\Gamma \vdash A \otimes B \qquad \Gamma', A, B \vdash C}{\Gamma, \Gamma' \vdash C}\,(\otimes_E) \qquad \frac{\Gamma \vdash A \qquad \Gamma' \vdash B}{\Gamma, \Gamma' \vdash A \otimes B}\,(\otimes_I)$$

Linear implication

Similarly, we change the rules for implication to

$$\frac{\Gamma \vdash A \multimap B \qquad \Gamma' \vdash A}{\Gamma, \Gamma' \vdash B}\ (\multimap_{\mathsf{E}}) \qquad\qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}\ (\multimap_{\mathsf{I}})$$

Axiom and unit

We replace the axiom rule

$$\frac{}{\Gamma, A \vdash A}\ (\text{ax})$$

by

$$\frac{}{A \vdash A}\ (\text{ax})$$

Axiom and unit

We replace the axiom rule

$$\frac{}{\Gamma, A \vdash A}\ (\text{ax})$$

by

$$\frac{}{A \vdash A}\ (\text{ax})$$

Similarly, we replace

$$\frac{}{\Gamma \vdash \top}\ (\top_\text{I})$$

by

$$\frac{}{\vdash 1}\ (1_\text{I})$$

Exchange

We also need to allow exchanging hypothesis in the context

$$\frac{\Gamma, B, A, \Gamma' \vdash C}{\Gamma, A, B, \Gamma' \vdash C}$$

Theorem
The λ-terms typable with $\otimes$, 1 and $\multimap$ are precisely the λ-terms which

- *are typable in the previous sense and*
- *linear.*

The simply-typed linear λ-calculus

The models of this logic are symmetric monoidal closed categories:

- **monoidal**: there is an object 1, and objects and morphisms equipped with a binary operation $\otimes$ together with isomorphisms

$$\alpha_{A,B,C} : (A \otimes B) \otimes C \simeq A \otimes (B \otimes C) \qquad \lambda_A : 1 \otimes A \simeq A$$

$$\rho_A : A \otimes 1 \simeq A$$

  satisfying axioms

The simply-typed linear λ-calculus

The models of this logic are symmetric monoidal closed categories:

- **monoidal**: there is an object 1, and objects and morphisms equipped with a binary operation $\otimes$ together with isomorphisms

$$\alpha_{A,B,C} : (A \otimes B) \otimes C \simeq A \otimes (B \otimes C) \qquad \lambda_A : 1 \otimes A \simeq A$$

$$\rho_A : A \otimes 1 \simeq A$$

  satisfying axioms
- **symmetric**:

$$\gamma_{A,B} : B \otimes A \to A \otimes B$$

The simply-typed linear λ-calculus

The models of this logic are symmetric monoidal closed categories:

- **monoidal**: there is an object 1, and objects and morphisms equipped with a binary operation $\otimes$ together with isomorphisms

$$\alpha_{A,B,C} : (A \otimes B) \otimes C \simeq A \otimes (B \otimes C) \qquad \lambda_A : 1 \otimes A \simeq A$$

$$\rho_A : A \otimes 1 \simeq A$$

  satisfying axioms

- **symmetric**:

$$\gamma_{A,B} : B \otimes A \to A \otimes B$$

- **closed**: there is a closure

$$\mathrm{Hom}(A \otimes B, C) \simeq \mathrm{Hom}(A, B \multimap C)$$

The simply-typed linear λ-calculus

Every typing derivation of a linear λ-term is a valid non-linear one: in terms of models, we have that every cartesian closed category is a symmetric monoidal closed category.

The simply-typed linear λ-calculus

Every typing derivation of a linear λ-term is a valid non-linear one: in terms of models, we have that every cartesian closed category is a symmetric monoidal closed category.

Conversely, a symmetric monoidal closed category is a cartesian closed one when we have the ability of duplicating and erasing objects:

The simply-typed linear λ-calculus

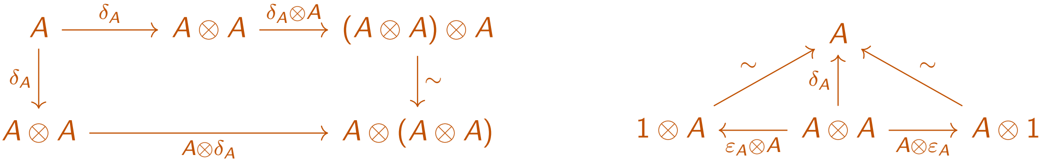
Every typing derivation of a linear λ-term is a valid non-linear one: in terms of models, we have that every cartesian closed category is a symmetric monoidal closed category.

Conversely, a symmetric monoidal closed category is a cartesian closed one when we have the ability of duplicating and erasing objects:

Theorem
A monoidal category is cartesian precisely when for every object A is equipped with

$$\delta_A : A \to A \otimes A \qquad\qquad \varepsilon_1 : A \to 1$$

compatible with other morphisms and coassociative, counital and cocommutative:



Linear logic

Linear logic builds on these observations and incorporates both worlds:

	conjunction
multiplicative	$\otimes$
additive	$\&$

Linear logic

Linear logic builds on these observations and incorporates both worlds:

	conjunction	disjunction
multiplicative	$\otimes$	$\parr$
additive	$\&$	$\oplus$

Linear logic

Linear logic builds on these observations and incorporates both worlds:

	conjunction	disjunction
multiplicative	$\otimes$	$\parr$
additive	$\&$	$\oplus$

We write A^* for the **dual** of a formula:

$$(A \otimes B)^* = A^* \parr B^* \qquad (A \parr B)^* = A^* \otimes B^* \qquad (A \,\&\, B)^* = A^* \oplus B^* \qquad \ldots$$

Linear logic

Linear logic builds on these observations and incorporates both worlds:

	conjunction	disjunction
multiplicative	$\otimes$	$⅋$
additive	$\&$	$\oplus$

We write A^* for the **dual** of a formula:

$$(A \otimes B)^* = A^* ⅋ B^* \qquad (A ⅋ B)^* = A^* \otimes B^* \qquad (A \,\&\, B)^* = A^* \oplus B^* \qquad \ldots$$

It can be shown that we can define

$$A \multimap B \qquad = \qquad A^* ⅋ B$$

(akin $A \Rightarrow B = \neg A \vee B$ in classical logic).

The exponential

In order to relate both worlds, linear logic introduces a last connective the **exponential** $!$

The exponential

In order to relate both worlds, linear logic introduces a last connective the **exponential** $!$ (and its dual $?$).

The exponential

In order to relate both worlds, linear logic introduces a last connective the **exponential** $!$ (and its dual $?$).

The intuition is that $!A$ is an A which an be used any number of times, something like

$$!A = \bigotimes_{n=0} A^{\otimes n} / \sim$$

we will see the formal rules later on (if we have time), but we should have maps

$$!A \multimap A \qquad\qquad !A \multimap !!A$$

The exponential

The main properties of exponential are that

- it turns additives into multiplicatives:

$$!(A \mathbin{\&} B) \quad \multimap\!\!\circ \quad !A \otimes !B$$

The exponential

The main properties of exponential are that

- it turns additives into multiplicatives:

$$!(A \,\&\, B) \quad \multimapboth \quad !A \otimes !B$$

- we can recover the usual implication as

$$A \Rightarrow B \quad = \quad !A \multimap B$$

Part VIII

The relational model

The relational model

In order to gain intuition, we can build the following model of the logic we have so far, in the category Rel of sets and relations.

We interpret A as a set:

- $[\![A \otimes B]\!] = [\![A \parr B]\!] = [\![A \multimap B]\!] = [\![A]\!] \times [\![B]\!]$
- $[\![A \mathbin{\&} B]\!] = [\![A \oplus B]\!] = [\![A]\!] \sqcup [\![B]\!]$

The relational model

In order to gain intuition, we can build the following model of the logic we have so far, in the category Rel of sets and relations.

We interpret A as a set:

- $[\![A \otimes B]\!] = [\![A \parr B]\!] = [\![A \multimap B]\!] = [\![A]\!] \times [\![B]\!]$
- $[\![A \,\&\, B]\!] = [\![A \oplus B]\!] = [\![A]\!] \sqcup [\![B]\!]$

A proof π of $A \vdash B$ is interpreted as a **relation** between $[\![A]\!]$ and $[\![B]\!]$:

$$[\![\pi]\!] \subseteq [\![A]\!] \times [\![B]\!]$$

(intuitively, $(a, b) \in [\![\pi]\!]$ means that b is a possible result for a)

The relational model: axiom

The axiom

$$\overline{A \vdash A} \ (\mathsf{ax})$$

is interpreted as the identity relation:

$$[\![A \vdash A]\!] = \{(a, a) \mid a \in [\![A]\!]\} \subseteq [\![A]\!] \times [\![A]\!]$$

The relational model: cut

For the cut rule

$$\dfrac{\dfrac{\rho}{A \vdash B} \qquad \dfrac{\sigma}{B \vdash C}}{A \vdash C}\ (\text{cut})$$

The relational model: cut

For the cut rule

$$\dfrac{\dfrac{\rho}{A \vdash B} \qquad \dfrac{\sigma}{B \vdash C}}{A \vdash C}\ (\text{cut})$$

given the respective interpretations

$$R \subseteq [\![A]\!] \times [\![B]\!] \qquad\qquad S \subseteq [\![B]\!] \times [\![B]\!]$$

of ρ and σ,

The relational model: cut

For the cut rule

$$\dfrac{\dfrac{\rho}{A \vdash B} \qquad \dfrac{\sigma}{B \vdash C}}{A \vdash C}\ \text{(cut)}$$

given the respective interpretations

$$R \subseteq [\![A]\!] \times [\![B]\!] \qquad\qquad S \subseteq [\![B]\!] \times [\![B]\!]$$

of ρ and σ, we define the interpretation of the proof as

$$S \circ R = \{(a,c) \in [\![A]\!] \times [\![C]\!] \mid (a,b) \in R \text{ and } (b,c) \in S\}$$

The relational model

We thus interpret proof in the category Rel with

- sets as objects,
- relations as morphisms,
- compositions and identities defined as above.

The relational model

We thus interpret proof in the category **Rel** with

- sets as objects,
- relations as morphisms,
- compositions and identities defined as above.

When we interpret

$$[\![A \otimes B]\!] = [\![A]\!] \times [\![B]\!] \qquad\qquad [\![A \,\&\, B]\!] = [\![A]\!] \sqcup [\![B]\!]$$

The cartesian product (in terms of the categorical property) is the second one, the first only induces a monoidal structure.

Exponential

The **exponential** $!A$ is interpreted as

$$[\![!A]\!] \quad = \quad \mathcal{M}_{\mathsf{fin}}([\![A]\!])$$

the set of finite multisets of elements of $[\![A]\!]$, i.e. lists of elements of $[\![A]\!]$ considered up to permutation.

Exponential

The **exponential** $!A$ is interpreted as

$$[\![!A]\!] \quad = \quad \mathcal{M}_{\mathsf{fin}}([\![A]\!])$$

the set of finite multisets of elements of $[\![A]\!]$, i.e. lists of elements of $[\![A]\!]$ considered up to permutation.

For instance, we can check that we have the required isomorphism

$$[\![!(A \mathbin{\&} B)]\!] \quad \simeq \quad [\![!A \otimes !B]\!]$$

Enriching over sets

This model is not the most informative (every connective is interpreted as its dual, it is not fully abstract, etc.).

A refined variant of this it can be obtained as follows.

The interpretation of a proof π of $A \vdash B$ is

$$[\![\pi]\!] \subseteq [\![A]\!] \times [\![B]\!]$$

and $(a, b) \in [\![\pi]\!]$ means that b is a possible result for a, but we do not track why!

Enriching over sets

A relation

$$R \quad \subseteq \quad X \times Y$$

can alternatively be seen as a function

$$(X \times Y) \to \{0, 1\}$$

Enriching over sets

A relation

$$R \quad \subseteq \quad X \times Y$$

can alternatively be seen as a function

$$(X \times Y) \to \{0, 1\}$$

We should get a better behaved model by switching to functions

$$(X \times Y) \to \mathsf{Set}$$

Enriching over sets

We can extend previous constructions in order to get a “model” of linear logic.

Enriching over sets

We can extend previous constructions in order to get a “model” of linear logic.

Given two “relations”

$$R : X \times Y \to \mathsf{Set} \qquad\qquad S : Y \times Z \to \mathsf{Set}$$

their composite is given by

$$S \circ R(x, z) \quad = \quad \bigsqcup_{y \in Y} R(x, y) \times S(y, z)$$

it is not strictly associative but only associative up to isomorphism.

Enriching over sets

We can extend previous constructions in order to get a "model" of linear logic.

Given two "relations"

$$R : X \times Y \to \mathsf{Set} \qquad\qquad S : Y \times Z \to \mathsf{Set}$$

their composite is given by

$$S \circ R(x, z) \quad = \quad \coprod_{y \in Y} R(x, y) \times S(y, z)$$

it is not strictly associative but only associative up to isomorphism.

We thus get a (bi)category of sets and "generalized relations", which corresponds to Girard's model of normal functors or Kock's polynomial functors.

Enriching over sets

In this model, the functions

$$A \Rightarrow B \quad = \quad !A \multimap B$$

are interpreted as “relations” R in

$$(\mathcal{M}_{\mathsf{fin}}[\![A]\!] \times [\![B]\!]) \to \mathsf{Set}$$

Enriching over sets

In this model, the functions

$$A \Rightarrow B \quad = \quad !A \multimap B$$

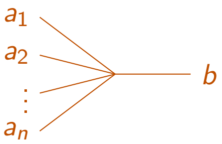
are interpreted as "relations" R in

$$(\mathcal{M}_{\text{fin}}[\![A]\!] \times [\![B]\!]) \to \text{Set}$$

An element of

$$R(a_1 a_2 \dots a_n, b)$$

can be interpreted as an operation



and composition corresponds to the expected composition of trees.

The exponential is interpreted as

$$!A \quad = \quad \mathcal{M}_{\mathsf{fin}}([\![A]\!]) \quad = \quad \{a_1 \ldots a_n \in A^* \mid n \in \mathbb{N}\} / \sim$$

where we identify

$$a_1 \ldots a_n \sim a_{\sigma(1)} \ldots a_{\sigma(n)}$$

for every permutation $\sigma \in \Sigma_n$.

Generalized species

The exponential is interpreted as

$$!A \quad = \quad \mathcal{M}_{\mathsf{fin}}(\llbracket A \rrbracket) \quad = \quad \{a_1 \dots a_n \in A^* \mid n \in \mathbb{N}\}/\sim$$

where we identify

$$a_1 \dots a_n \sim a_{\sigma(1)} \dots a_{\sigma(n)}$$

for every permutation $\sigma \in \Sigma_n$.

We would like to avoid quotienting and keep this action explicit.

Generalized species

This suggests another generalization of the model [Fiore, Gambino, Hyland]:

- we model objects as groupoids,
- morphisms are functors (= profunctors)

$$(X \times Y) \to \mathrm{Set}$$

- the exponential $!X$ is the free symmetric monoidal category on X.

Generalized species

In particular, morphisms $!1 \to 1$

- where functions $\mathbb{N} \to \mathsf{Set}$ in previous model,
- are now functors $\mathsf{Bij} \to \mathsf{Set}$.

In this sense morphisms $!A \to B$ are generalized species.

This suggests studying categorical structures present in this category, as well as further generalizations of it.

Questions?

Part IX

Rules for linear logic

Sequent calculus

In order to formulate the usual rules of linear logic, we perform two changes:

- we use sequent calculus instead of natural deduction: we change

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \& B}\ (\&_\mathrm{R})$$

  to

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, A \& B \vdash C}\ (\&_\mathrm{L}^\mathrm{l}) \qquad \frac{\Gamma, B \vdash \Delta}{\Gamma, A \& B \vdash C}\ (\&_\mathrm{L}^\mathrm{r}) \qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \& B}\ (\&_\mathrm{R})$$

Sequent calculus

In order to formulate the usual rules of linear logic, we perform two changes:

- we use sequent calculus instead of natural deduction: we change

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \,\&\, B}\ (\&_{\mathrm{R}})$$

  to

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, A \,\&\, B \vdash C}\ (\&_{\mathrm{L}}^{\mathrm{l}}) \qquad \frac{\Gamma, B \vdash \Delta}{\Gamma, A \,\&\, B \vdash C}\ (\&_{\mathrm{L}}^{\mathrm{r}}) \qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \,\&\, B}\ (\&_{\mathrm{R}})$$

- we shift to classical logic by allowing multiple sequents on the right: sequents are of the form

$$\Gamma \vdash \Delta$$

Linear logic: categorical rules

Categorical rules:

$$\frac{}{\Gamma \vdash \Gamma}\ (\text{ax}) \qquad \frac{\Gamma \vdash A, \Delta \qquad \Gamma', A \vdash B, \Delta'}{\Gamma, \Gamma' \vdash B, \Delta, \Delta'}\ (\text{cut})$$

Structural rules:

$$\frac{\Gamma, B, A, \Gamma' \vdash \Delta}{\Gamma, A, B, \Gamma' \vdash \Delta} \qquad \frac{\Gamma \vdash \Delta, B, A, \Delta'}{\Gamma \vdash \Delta, A, B, \Delta'}$$

Linear logic: multiplicative rules

Multiplicative conjunction:

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \otimes B \vdash \Delta}\ (\otimes_L) \qquad \frac{\Gamma \vdash A, \Delta \qquad \Gamma' \vdash B, \Delta}{\Gamma, \Gamma' \vdash A \otimes B, \Delta, \Delta'}\ (\otimes_R)$$

Multiplicative truth:

$$\frac{\Gamma \vdash A}{\Gamma, 1 \vdash A}\ (1_L) \qquad \frac{}{\vdash 1}\ (1_R)$$

Multiplicative disjunction:

$$\frac{\Gamma, A \vdash \Delta \qquad \Gamma', B \vdash \Delta'}{\Gamma, \Gamma', A \parr B \vdash \Delta, \Delta'}\ (\parr_L) \qquad \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \parr B, \Delta}\ (\parr_R)$$

Multiplicative falsity:

$$\frac{}{\bot \vdash}\ (\bot_L) \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \bot, \Delta}\ (\bot_R)$$

Linear logic: additives

Additive conjunction:

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, A \,\&\, B \vdash \Delta}\,(\&_L^l) \qquad \frac{\Gamma, B \vdash \Delta}{\Gamma, A \,\&\, B \vdash \Delta}\,(\&_L^r) \qquad \frac{\Gamma \vdash A, \Delta \qquad \Gamma \vdash B, \Delta}{\Gamma \vdash A \,\&\, B, \Delta}\,(\&_R)$$

Additive truth:

$$\frac{}{\Gamma \vdash \top, \Delta}\,(\top_R)$$

Additive disjunction:

$$\frac{\Gamma, A \vdash \Delta \qquad \Gamma, B \vdash \Delta}{\Gamma, A \oplus B \vdash \Delta}\,(\oplus_L) \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash A \oplus B, \Delta}\,(\oplus_R^l) \qquad \frac{\Gamma \vdash B, \Delta}{\Gamma \vdash A \oplus B, \Delta}\,(\oplus_R^r)$$

Additive falsity:

$$\frac{}{\Gamma, 0 \vdash \Delta}\,(0_L)$$

Linear logic: exponentials

Bang:

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, !A \vdash \Delta} \qquad \frac{\Gamma \vdash \Delta}{\Gamma, !A \vdash \Delta} \qquad \frac{\Gamma, !A, !A \vdash \Delta}{\Gamma, !A \vdash \Delta} \qquad \frac{!\Gamma \vdash A, ?\Delta}{!\Gamma \vdash !A, ?\Delta}$$

Maybe:

$$\frac{\Gamma \vdash A, \Delta}{\Gamma \vdash ?A, \Delta} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash ?A, \Delta} \qquad \frac{\Gamma \vdash ?A, ?A, \Delta}{\Gamma \vdash ?A, \Delta} \qquad \frac{!\Gamma, A \vdash ?\Delta}{!\Gamma, ?A \vdash ?\Delta}$$

(dereliction / weakening / contraction / promotion)