Categories, Proofs and Processes Lecture III
The Curry-Howard-Lambek Correspondence

Samson Abramsky

Oxford University Computing Laboratory

In A Nutshell

In A Nutshell

$$
\begin{tikzcd}
\text{Logic} \arrow[rr, leftrightarrow] \arrow[dr, leftrightarrow] & & \text{Computation} \arrow[dl, leftrightarrow] \\
& \text{Categories} &
\end{tikzcd}
$$

In A Nutshell

$$
\begin{tikzcd}
\text{Logic} \arrow[rr, leftrightarrow] \arrow[dr, leftrightarrow] & & \text{Computation} \arrow[dl, leftrightarrow] \\
& \text{Categories} &
\end{tikzcd}
$$

This connection has been known since the 1970's, and widely used in Computer Science. Beginning to be used in Quantum Informatics!

What is Logic about?

What is Logic about?

One main kind of answer focusses on **Truth**

What is Logic about?

One main kind of answer focusses on **Truth**

Another, on **Proof**.

What is Logic about?

One main kind of answer focusses on **Truth**

Another, on **Proof**.

We shall focus mainly on Proof:

$\boxed{\text{What follows from what?}}$

Formal Proofs

Formal Proofs

Traditional introductions to logic focus on Hilbert-style proof systems: generating the set of **theorems** of a system from a set of axioms by applying rules of inference (e.g. Modus Ponens).

Formal Proofs

Traditional introductions to logic focus on Hilbert-style proof systems: generating the set of **theorems** of a system from a set of axioms by applying rules of inference (e.g. Modus Ponens).

A key step in logic took place in the 1930's, with the advent of **Gentzen-style systems**. Instead of focussing on theorems, look more generally and symmetrically at

$\boxed{\text{What follows from what}}$

Proof from Assumptions

Proof from Assumptions

Cf. Open vs. Closed Systems.

Proof from Assumptions

Cf. Open vs. Closed Systems.

To capture this formally:
Proof of A from **assumptions** $A_1, \ldots, A_n$:

$$A_1, \ldots, A_n \vdash A$$

Proof from Assumptions

Cf. Open vs. Closed Systems.

To capture this formally:
Proof of A from **assumptions** $A_1, \ldots, A_n$:

$$A_1, \ldots, A_n \vdash A$$

We use Γ, Δ to range over finite sets of formulas, writing $\Gamma \vdash A$ etc.

Natural Deduction system for $\wedge$, $\supset$

Natural Deduction system for $\wedge$, $\supset$

Identity

$$\frac{}{\Gamma, A \vdash A}\ \mathsf{Id}$$

Natural Deduction system for $\wedge$, $\supset$

Identity

$$\frac{}{\Gamma, A \vdash A}\ \mathsf{Id}$$

Conjunction

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge\text{-intro} \quad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ \wedge\text{-elim-1} \quad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ \wedge\text{-elim-2}$$

Natural Deduction system for $\wedge$, $\supset$

Identity

$$\frac{}{\Gamma, A \vdash A}\ \mathsf{Id}$$

Conjunction

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge\text{-intro} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ \wedge\text{-elim-1} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ \wedge\text{-elim-2}$$

Implication

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B}\ \supset\text{-intro} \qquad \frac{\Gamma \vdash A \supset B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ \supset\text{-elim}$$

Structural Proof Theory

Structural Proof Theory

The idea is to study the 'space of formal proofs' as a mathematical structure in its own right, rather than to focus only on

$$\text{Provability} \longleftrightarrow \text{Truth}$$

(i.e. the usual notions of 'soundness and completeness').

Structural Proof Theory

The idea is to study the ‘space of formal proofs’ as a mathematical structure in its own right, rather than to focus only on

$$\text{Provability} \longleftrightarrow \text{Truth}$$

(i.e. the usual notions of ‘soundness and completeness’).

Why? One motivation comes from trying to understand and use the **computational content of proofs**. To make this precise, we look at the ‘Curry-Howard correspondence’.

Terms

Terms

λ-calculus: a pure calculus of functions.

Terms

λ-calculus: a pure calculus of functions.

Variables x, y, z, ...

Terms

λ-calculus: a pure calculus of functions.

Variables x, y, z, ...

Terms

$$t \quad ::= \quad x \mid \underbrace{tu}_{\text{application}} \mid \underbrace{\lambda x.\, t}_{\text{abstraction}}$$

Terms

λ-calculus: a pure calculus of functions.

Variables $x, y, z, \ldots$

Terms

$$t \ ::= \ x \ \mid \ \underbrace{tu}_{\text{application}} \ \mid \ \underbrace{\lambda x.\, t}_{\text{abstraction}}$$

Examples

$\lambda x.\, x + 1$	successor function
$\lambda x.\, x$	identity function
$\lambda f.\, \lambda x.\, fx$	application
$\lambda f.\, \lambda x.\, f(fx)$	double application
$\lambda f.\, \lambda g.\, \lambda x.\, g(f(x))$	composition $g \circ f$

Conversion and Reduction

Conversion and Reduction

The basic equation governing this calculus is β-**conversion**:

$$(\lambda x.\, t)u = t[u/x]$$

Conversion and Reduction

The basic equation governing this calculus is β-**conversion**:

$$(\lambda x.\, t)u = t[u/x]$$

E.g.

$$(\lambda f.\, \lambda x.\, f(fx))(\lambda x.\, x + 1)0 = \cdots 2.$$

Conversion and Reduction

The basic equation governing this calculus is β-**conversion**:

$$(\lambda x.\, t)u = t[u/x]$$

E.g.

$$(\lambda f.\, \lambda x.\, f(fx))(\lambda x.\, x + 1)0 = \cdots 2.$$

By orienting this equation, we get a 'dynamics' — β-**reduction**

$$(\lambda x.\, t)u \rightarrow t[u/x]$$

From type-free to typed

From type-free to typed

'Pure' λ-calculus is **very** unconstrained.

From type-free to typed

'Pure' λ-calculus is **very** unconstrained.

For example, it allows terms like $\omega \equiv \lambda x.\, xx$ — self-application.

From type-free to typed

'Pure' λ-calculus is **very** unconstrained.

For example, it allows terms like $\omega \equiv \lambda x.\, xx$ — self-application.

Hence $\Omega \equiv \omega\omega$, which **diverges**:

$$\Omega \rightarrow \Omega \rightarrow \cdots$$

From type-free to typed

'Pure' λ-calculus is **very** unconstrained.

For example, it allows terms like $\omega \equiv \lambda x.\, xx$ — self-application.

Hence $\Omega \equiv \omega\omega$, which **diverges**:

$$\Omega \rightarrow \Omega \rightarrow \cdots$$

Also, $\mathbf{Y} \equiv \lambda f.\, (\lambda x.\, f(xx))(\lambda x.\, f(xx))$ — recursion.

$$\mathbf{Y}t \rightarrow (\lambda x.\, t(xx))(\lambda x.\, t(xx)) \rightarrow t((\lambda x.\, t(xx))(\lambda x.\, t(xx))) = t(\mathbf{Y}t).$$

From type-free to typed

'Pure' λ-calculus is **very** unconstrained.

For example, it allows terms like $\omega \equiv \lambda x.\, xx$ — self-application.

Hence $\Omega \equiv \omega\omega$, which **diverges**:

$$\Omega \rightarrow \Omega \rightarrow \cdots$$

Also, $\mathbf{Y} \equiv \lambda f.\, (\lambda x.\, f(xx))(\lambda x.\, f(xx))$ — recursion.

$$\mathbf{Y}t \rightarrow (\lambda x.\, t(xx))(\lambda x.\, t(xx)) \rightarrow t((\lambda x.\, t(xx))(\lambda x.\, t(xx))) = t(\mathbf{Y}t).$$

Historically, Curry extracted **Y** from an analysis of Russell's Paradox.

From type-free to typed

'Pure' λ-calculus is **very** unconstrained.

For example, it allows terms like $\omega \equiv \lambda x.\, xx$ — self-application.

Hence $\Omega \equiv \omega\omega$, which **diverges**:

$$\Omega \rightarrow \Omega \rightarrow \cdots$$

Also, $\mathbf{Y} \equiv \lambda f.\, (\lambda x.\, f(xx))(\lambda x.\, f(xx))$ — recursion.

$$\mathbf{Y}t \rightarrow (\lambda x.\, t(xx))(\lambda x.\, t(xx)) \rightarrow t((\lambda x.\, t(xx))(\lambda x.\, t(xx))) = t(\mathbf{Y}t).$$

Historically, Curry extracted $\mathbf{Y}$ from an analysis of Russell's Paradox.

Solution: introduce Types.

From type-free to typed

'Pure' λ-calculus is **very** unconstrained.

For example, it allows terms like $\omega \equiv \lambda x.\, xx$ — self-application.

Hence $\Omega \equiv \omega\omega$, which **diverges**:

$$\Omega \rightarrow \Omega \rightarrow \cdots$$

Also, $\mathbf{Y} \equiv \lambda f.\, (\lambda x.\, f(xx))(\lambda x.\, f(xx))$ — recursion.

$$\mathbf{Y}t \rightarrow (\lambda x.\, t(xx))(\lambda x.\, t(xx)) \rightarrow t((\lambda x.\, t(xx))(\lambda x.\, t(xx))) = t(\mathbf{Y}t).$$

Historically, Curry extracted **Y** from an analysis of Russell's Paradox.

Solution: introduce Types.

$\boxed{\textbf{Types are there to stop you doing (bad) things}}$

From type-free to typed

'Pure' λ-calculus is **very** unconstrained.

For example, it allows terms like $\omega \equiv \lambda x.\, xx$ — self-application.

Hence $\Omega \equiv \omega\omega$, which **diverges**:

$$\Omega \rightarrow \Omega \rightarrow \cdots$$

Also, $\mathbf{Y} \equiv \lambda f.\, (\lambda x.\, f(xx))(\lambda x.\, f(xx))$ — recursion.

$$\mathbf{Y}t \rightarrow (\lambda x.\, t(xx))(\lambda x.\, t(xx)) \rightarrow t((\lambda x.\, t(xx))(\lambda x.\, t(xx))) = t(\mathbf{Y}t).$$

Historically, Curry extracted $\mathbf{Y}$ from an analysis of Russell's Paradox.

Solution: introduce Types.

Types are there to stop you doing (bad) things

N.B. One of the most fruitful **positive** ideas in Computer Science!

Simply-Typed λ-calculus

Simply-Typed λ-calculus

Base types

$$B \quad ::= \quad \iota \mid \ldots$$

Simply-Typed λ-calculus

Base types

$$B \quad ::= \quad \iota \mid \ldots$$

General Types

$$T \quad ::= \quad B \mid T \to T \mid T \times T$$

Simply-Typed λ-calculus

Base types

$$B \quad ::= \quad \iota \mid \ldots$$

General Types

$$T \quad ::= \quad B \mid T \rightarrow T \mid T \times T$$

Examples

$\iota \rightarrow \iota \rightarrow \iota$ first-order function type

$(\iota \rightarrow \iota) \rightarrow \iota$ second-order function type

Simply-Typed λ-calculus

Base types

$$B \ ::= \ \iota \mid \ldots$$

General Types

$$T \ ::= \ B \mid T \to T \mid T \times T$$

Examples

$$\iota \to \iota \to \iota \qquad \text{first-order function type}$$

$$(\iota \to \iota) \to \iota \qquad \text{second-order function type}$$

Typed terms. Typing judgement:

$$x_1 : T_1, \ldots x_k : T_k \vdash t : T$$

Simply-Typed λ-calculus

Base types

$$B \ ::= \ \iota \mid \ldots$$

General Types

$$T \ ::= \ B \mid T \rightarrow T \mid T \times T$$

Examples

$$\iota \rightarrow \iota \rightarrow \iota \qquad \text{first-order function type}$$

$$(\iota \rightarrow \iota) \rightarrow \iota \qquad \text{second-order function type}$$

Typed terms. Typing judgement:

$$x_1 : T_1, \ldots x_k : T_k \vdash t : T$$

The term t has type T **under the assumption** (or: **in the context**) that the variable x_1 has type T_1, ..., x_k has type T_k.

The System of Simply-Typed λ-calculus

Variable

$$\frac{}{\Gamma, x : T \vdash x : T}$$

Product

$$\frac{\Gamma \vdash t : T \qquad \Gamma \vdash u : U}{\Gamma \vdash \langle t, u \rangle : T \times U} \qquad \frac{\Gamma \vdash v : T \times U}{\Gamma \vdash \pi_1 v : T} \qquad \frac{\Gamma \vdash v : T \times U}{\Gamma \vdash \pi_2 v : U}$$

Function

$$\frac{\Gamma, x : U \vdash t : T}{\Gamma \vdash \lambda x.\, t : U \rightarrow T} \qquad \frac{\Gamma \vdash t : U \rightarrow T \qquad \Gamma \vdash u : U}{\Gamma \vdash tu : T}$$

Reduction rules

Reduction rules

Computation rules (β-reductions):

$$
\begin{array}{lcl}
(\lambda x.\, t)u & \longrightarrow & t[u/x] \\
\pi_1\langle t, u\rangle & \longrightarrow & t \\
\pi_2\langle t, u\rangle & \longrightarrow & u
\end{array}
$$

Reduction rules

Computation rules (β-reductions):

$$\begin{array}{lcl} (\lambda x.\, t)u & \longrightarrow & t[u/x] \\ \pi_1\langle t, u\rangle & \longrightarrow & t \\ \pi_2\langle t, u\rangle & \longrightarrow & u \end{array}$$

Also, η-laws (extensionality principles):

$$\begin{array}{lcll} t & = & \lambda x.\, tx & x \text{ not free in } t\text{, at function types} \\ v & = & \langle \pi_1 v, \pi_2 v\rangle & \text{at product types} \end{array}$$

The Types/Proofs Gestalt

Simple Type System for $\times$, $\rightarrow$.

Variable

$$\frac{}{\Gamma, x : t \vdash x : T}$$

Product

$$\frac{\Gamma \vdash t : T \qquad \Gamma \vdash u : U}{\Gamma \vdash \langle t, u \rangle : T \times U} \qquad \frac{\Gamma \vdash v : T \times U}{\Gamma \vdash \pi_1 v : T} \qquad \frac{\Gamma \vdash v : T \times U}{\Gamma \vdash \pi_2 v : U}$$

Function

$$\frac{\Gamma, x : U \vdash t : T}{\Gamma \vdash \lambda x.\, t : U \rightarrow T} \qquad \frac{\Gamma \vdash t : U \rightarrow T \qquad \Gamma \vdash u : U}{\Gamma \vdash tu : T}$$

The Types/Proofs Gestalt

Natural Deduction System for $\wedge$, $\supset$

Identity

$$\frac{}{\Gamma, A \vdash A}\ \mathsf{Id}$$

Conjunction

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge\text{-intro} \quad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ \wedge\text{-elim-1} \quad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ \wedge\text{-elim-2}$$

Implication

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B}\ \supset\text{-intro} \qquad \frac{\Gamma \vdash A \supset B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ \supset\text{-elim}$$

The Curry-Howard Correspondence

The Curry-Howard Correspondence

If we equate

$$\begin{array}{ccc} \wedge & \equiv & \times \\ \supset & \equiv & \rightarrow \end{array}$$

they are the same!

The Curry-Howard Correspondence

If we equate

$$\begin{array}{ccc} \wedge & \equiv & \times \\ \supset & \equiv & \longrightarrow \end{array}$$

they are the same!

This is the **Curry-Howard correspondence** (sometimes: 'Curry-Howard isomorphism').

The Curry-Howard Correspondence

If we equate

$$\begin{array}{ccc} \wedge & \equiv & \times \\ \supset & \equiv & \rightarrow \end{array}$$

they are the same!

This is the **Curry-Howard correspondence** (sometimes: 'Curry-Howard isomorphism').

It works on three levels:

Formulas	Types
Proofs	Terms
Proof transformations	Term reductions

The Curry-Howard Correspondence

If we equate

$$\begin{array}{ccc} \wedge & \equiv & \times \\ \supset & \equiv & \rightarrow \end{array}$$

they are the same!

This is the **Curry-Howard correspondence** (sometimes: 'Curry-Howard isomorphism').

It works on three levels:

Formulas	Types
Proofs	Terms
Proof transformations	Term reductions

The λ-term assigned to a Natural Deduction proof encodes the entire tree structure of the proof. The **open assumptions** are the free variables; the **uses** of the assumptions are the **occurrences** of the free variables.

The Curry-Howard Correspondence

If we equate

$$\begin{array}{ccc} \wedge & \equiv & \times \\ \supset & \equiv & \rightarrow \end{array}$$

they are the same!

This is the **Curry-Howard correspondence** (sometimes: 'Curry-Howard isomorphism').

It works on three levels:

Formulas	Types
Proofs	Terms
Proof transformations	Term reductions

The λ-term assigned to a Natural Deduction proof encodes the entire tree structure of the proof. The **open assumptions** are the free variables; the **uses** of the assumptions are the **occurrences** of the free variables.

Elimination of detours in ND proofs corresponds to **normalization** of λ-terms.

Reduction revisited

Reduction revisited

Reduction results in a **normal form**; a proof in which all **lemmas** have been eliminated, resulting in an **explicit but much longer expression**.

Reduction revisited

Reduction results in a **normal form**; a proof in which all **lemmas** have been eliminated, resulting in an **explicit but much longer expression**.

Even simply typed lambda calculus has enormous (**non-elementary**) complexity.

Reduction revisited

Reduction results in a **normal form**; a proof in which all **lemmas** have been eliminated, resulting in an **explicit but much longer expression**.

Even simply typed lambda calculus has enormous (**non-elementary**) complexity.

β-reduction:

$$(\lambda x.\, u)v \longrightarrow u[v/x]$$

Reduction revisited

Reduction results in a **normal form**; a proof in which all **lemmas** have been eliminated, resulting in an **explicit but much longer expression**.

Even simply typed lambda calculus has enormous (**non-elementary**) complexity.

β-reduction:

$$(\lambda x.\, u)v \longrightarrow u[v/x]$$

A **redex** of a term t is a subexpression of the form of the left-hand-side of the above rule, to which β-reduction can be applied. A term is in **normal form** of it contains no redexes. We write $t \twoheadrightarrow u$ if u can be obtained from t by a number of applications of β-reduction. Thus $\twoheadrightarrow$ is a reflexive and transitive relation.

Reduction revisited

Reduction results in a **normal form**; a proof in which all **lemmas** have been eliminated, resulting in an **explicit but much longer expression**.

Even simply typed lambda calculus has enormous (**non-elementary**) complexity.

β-reduction:

$$(\lambda x.\, u)v \longrightarrow u[v/x]$$

A **redex** of a term t is a subexpression of the form of the left-hand-side of the above rule, to which β-reduction can be applied. A term is in **normal form** of it contains no redexes. We write $t \twoheadrightarrow u$ if u can be obtained from t by a number of applications of β-reduction. Thus $\twoheadrightarrow$ is a reflexive and transitive relation.

Substitution:

$$x[t/x] = t \qquad y[t/x] = y \quad (x \neq y)$$

$$(\lambda z.\, u)[t/x] = \lambda z.\, (u[t/x]) \qquad (*)$$

$$(uv)[t/x] = (u[t/x])(v[t/x])$$

Normalization in simple types is non-elementary

Normalization in simple types is non-elementary

Define $e(m,n)$ by $e(m,0) = m$, $e(m,n+1) = 2^{e(m,n)}$. Thus $e(m,n)$ is an exponential 'stack' of n 2's with an m at the top:

$$e(m,n) \quad = \quad 2^{2^{\cdot^{\cdot^{\cdot^{2^{m}}}}}}$$

Normalization in simple types is non-elementary

Define $e(m,n)$ by $e(m,0) = m$, $e(m,n+1) = 2^{e(m,n)}$. Thus $e(m,n)$ is an exponential 'stack' of n 2's with an m at the top:

$$e(m,n) \;=\; 2^{2^{\cdot^{\cdot^{\cdot^{2^m}}}}}$$

We can prove that a term of degree d and height h has a normal form of height bounded by $e(h,d)$.

Normalization in simple types is non-elementary

Define $e(m,n)$ by $e(m,0) = m$, $e(m,n+1) = 2^{e(m,n)}$. Thus $e(m,n)$ is an exponential 'stack' of n 2's with an m at the top:

$$e(m,n) \;=\; 2^{2^{\cdot^{\cdot^{\cdot^{2^m}}}}}$$

We can prove that a term of degree d and height h has a normal form of height bounded by $e(h,d)$.

However, there is no **elementary** bound (*i.e.* an exponential stack of fixed height).

Constructive reading of formulas

Constructive reading of formulas

The 'Brouwer-Heyting-Kolmogorov interpretation'.

- A proof of an implication $A \supset B$ is a construction which transforms any proof of A into a proof of B.
- A proof of $A \wedge B$ is a pair consisting of a proof of A and a proof of B.

Constructive reading of formulas

The 'Brouwer-Heyting-Kolmogorov interpretation'.

- A proof of an implication $A \supset B$ is a construction which transforms any proof of A into a proof of B.
- A proof of $A \wedge B$ is a pair consisting of a proof of A and a proof of B.

These readings motivate identifying $A \wedge B$ with $A \times B$, and $A \supset B$ with $A \longrightarrow B$.

Constructive reading of formulas

The 'Brouwer-Heyting-Kolmogorov interpretation'.

- A proof of an implication $A \supset B$ is a construction which transforms any proof of A into a proof of B.
- A proof of $A \wedge B$ is a pair consisting of a proof of A and a proof of B.

These readings motivate identifying $A \wedge B$ with $A \times B$, and $A \supset B$ with $A \rightarrow B$.

Moreover, these ideas have strong connections to computing. The λ-calculus is a 'pure' version of functional programming languages such as Haskell and SML. So we get a reading of

Proofs as Programs

The Connection to Categories

We now have our link

$$\text{Logic} \longleftrightarrow \text{Computation}$$

The Connection to Categories

We now have our link

$$\begin{tikzcd} \text{Logic} \arrow[rr, leftrightarrow] & & \text{Computation} \end{tikzcd}$$

We now complete the triangle by showing the connection to Categories:

$$\begin{tikzcd} \text{Logic} \arrow[rr, leftrightarrow] \arrow[dr, leftrightarrow] & & \text{Computation} \arrow[dl, leftrightarrow] \\ & \text{Categories} & \end{tikzcd}$$

Exponentials in **Set**

Exponentials in **Set**

In **Set**, given sets A, B, we can form the set of functions $B^A = \mathbf{Set}(A, B)$, **which is again a set**. This closure of **Set** under forming 'function spaces' is one of its most important properties.

Exponentials in **Set**

In **Set**, given sets A, B, we can form the set of functions $B^A = \mathbf{Set}(A, B)$, **which is again a set**. This closure of **Set** under forming 'function spaces' is one of its most important properties.

How can we axiomatize this situation? Once again, rather than asking what the elements of a function space **are**, we ask rather: what can we **do** with it operationally?

Exponentials in **Set**

In **Set**, given sets A, B, we can form the set of functions $B^A = \mathbf{Set}(A, B)$, **which is again a set**. This closure of **Set** under forming 'function spaces' is one of its most important properties.

How can we axiomatize this situation? Once again, rather than asking what the elements of a function space **are**, we ask rather: what can we **do** with it operationally?

Answer: apply functions to their arguments. That is, there is a map

$$\mathrm{ev}_{A,B} : B^A \times A \longrightarrow B \qquad \mathrm{ev}_{A,B}(f, a) = f(a)$$

Exponentials in **Set**

In **Set**, given sets A, B, we can form the set of functions $B^A = \mathbf{Set}(A, B)$, **which is again a set**. This closure of **Set** under forming 'function spaces' is one of its most important properties.

How can we axiomatize this situation? Once again, rather than asking what the elements of a function space **are**, we ask rather: what can we **do** with it operationally?

Answer: apply functions to their arguments. That is, there is a map

$$\mathsf{ev}_{A,B} : B^A \times A \longrightarrow B \qquad \mathsf{ev}_{A,B}(f, a) = f(a)$$

Think of the function as a 'black box': we can feed it inputs and observe the outputs.

Couniversal property of evaluation in **Set**

Couniversal property of evaluation in **Set**

For any $g : C \times A \longrightarrow B$, there is a unique map $\Lambda(g) : C \longrightarrow B^A$ such that:

$$
\begin{tikzcd}
B^A & & B^A \times A \arrow[r, "\mathrm{ev}_{A,B}"] & B \\
C \arrow[u, dashed, "\Lambda(g)"] & & C \times A \arrow[u, dashed, "\Lambda(g) \times \mathrm{id}_A"] \arrow[ur, "g"'] &
\end{tikzcd}
$$

Couniversal property of evaluation in **Set**

For any $g : C \times A \longrightarrow B$, there is a unique map $\Lambda(g) : C \longrightarrow B^A$ such that:

$$
\begin{tikzcd}
B^A & & B^A \times A \arrow[r, "\mathrm{ev}_{A,B}"] & B \\
C \arrow[u, dashed, "\Lambda(g)"] & & C \times A \arrow[u, dashed, "\Lambda(g) \times \mathrm{id}_A"] \arrow[ur, "g"'] &
\end{tikzcd}
$$

In **Set**, this is defined by

$$\Lambda(g)(c) = k : A \longrightarrow B \quad \text{where} \quad k(a) = g(c, a).$$

Couniversal property of evaluation in **Set**

For any $g : C \times A \longrightarrow B$, there is a unique map $\Lambda(g) : C \longrightarrow B^A$ such that:

$$
\begin{tikzcd}
B^A & & B^A \times A \arrow[r, "\mathsf{ev}_{A,B}"] & B \\
C \arrow[u, dashed, "\Lambda(g)"] & & C \times A \arrow[u, dashed, "\Lambda(g) \times \mathsf{id}_A"] \arrow[ur, "g"'] &
\end{tikzcd}
$$

In **Set**, this is defined by

$$\Lambda(g)(c) = k : A \longrightarrow B \;\text{ where }\; k(a) = g(c, a).$$

This process of transforming a function of two arguments into a function-valued function of one argument is known as **Currying** after H. B. Curry.

Couniversal property of evaluation in **Set**

For any $g : C \times A \longrightarrow B$, there is a unique map $\Lambda(g) : C \longrightarrow B^A$ such that:

$$\begin{tikzcd} B^A \\ \\ C \arrow[uu, dashed, "\Lambda(g)"] \end{tikzcd} \qquad \begin{tikzcd} B^A \times A \arrow[r, "\mathrm{ev}_{A,B}"] & B \\ \\ C \times A \arrow[uu, dashed, "\Lambda(g) \times \mathrm{id}_A"] \arrow[uur, "g"'] \end{tikzcd}$$

In **Set**, this is defined by

$$\Lambda(g)(c) = k : A \longrightarrow B \ \text{ where } \ k(a) = g(c, a).$$

This process of transforming a function of two arguments into a function-valued function of one argument is known as **Currying** after H. B. Curry.

It is an algebraic form of λ-**abstraction**.

General definition of exponentials

General definition of exponentials

Let $\mathcal{C}$ be a category with a terminal object and binary products. For each object A of $\mathcal{C}$, we can define a functor

$$- \times A : \mathcal{C} \longrightarrow \mathcal{C}$$

General definition of exponentials

Let $\mathcal{C}$ be a category with a terminal object and binary products. For each object A of $\mathcal{C}$, we can define a functor

$$- \times A : \mathcal{C} \longrightarrow \mathcal{C}$$

We say that $\mathcal{C}$ **has exponentials** if for all objects A and B of $\mathcal{C}$ there is a couniversal arrow from $- \times A$ to B, *i.e.* an object B^A of $\mathcal{C}$ and a morphism

$$\mathsf{ev}_{A,B} : B^A \times A \longrightarrow B$$

with the couniversal property: for every $g : C \times A \longrightarrow B$, there is a unique morphism $\Lambda(g) : C \longrightarrow B^A$ such that

$$\mathsf{ev}_{A,B} \circ (\Lambda(g) \times \mathsf{id}A) = g.$$

(Same as diagram on previous slide).

Cartesian Closed Categories

Cartesian Closed Categories

A category with a terminal object, products and exponentials is called a **Cartesian Closed Category** (CCC).

Cartesian Closed Categories

A category with a terminal object, products and exponentials is called a **Cartesian Closed Category** (CCC).

This notion is fundamental in understanding functional types, models of λ-calculus, and the structure of proofs.

Cartesian Closed Categories

A category with a terminal object, products and exponentials is called a **Cartesian Closed Category** (CCC).

This notion is fundamental in understanding functional types, models of λ-calculus, and the structure of proofs.

Notation The notation of B^A for exponential objects, and $\mathrm{ev}_{A,B}$ for evaluation, is standard in the category theory literature. However, for our purposes, the following notation will be more convenient: $A \Rightarrow B$ for exponential objects, and

$$\mathsf{Ap}_{A,B} : (A \Rightarrow B) \times A \longrightarrow B$$

for **application** (*i.e.* evaluation).

Cartesian Closed Categories

A category with a terminal object, products and exponentials is called a **Cartesian Closed Category** (CCC).

This notion is fundamental in understanding functional types, models of λ-calculus, and the structure of proofs.

Notation The notation of B^A for exponential objects, and $\mathsf{ev}_{A,B}$ for evaluation, is standard in the category theory literature. However, for our purposes, the following notation will be more convenient: $A \Rightarrow B$ for exponential objects, and

$$\mathsf{Ap}_{A,B} : (A \Rightarrow B) \times A \longrightarrow B$$

for **application** (*i.e.* evaluation).

A basic example of a cartesian closed category is **Set**.

Cartesian Closed Categories

A category with a terminal object, products and exponentials is called a **Cartesian Closed Category** (CCC).

This notion is fundamental in understanding functional types, models of λ-calculus, and the structure of proofs.

Notation The notation of B^A for exponential objects, and $\mathsf{ev}_{A,B}$ for evaluation, is standard in the category theory literature. However, for our purposes, the following notation will be more convenient: $A \Rightarrow B$ for exponential objects, and

$$\mathsf{Ap}_{A,B} : (A \Rightarrow B) \times A \longrightarrow B$$

for **application** (*i.e.* evaluation).

A basic example of a cartesian closed category is **Set**.
Others?

Example: Boolean Algebras

Example: Boolean Algebras

A Boolean algebra (*e.g.* a powerset $\mathcal{P}(X)$) is a CCC.

Example: Boolean Algebras

A Boolean algebra (*e.g.* a powerset $\mathcal{P}(X)$) is a CCC.

Products are given by conjunctions $A \wedge B$.

Example: Boolean Algebras

A Boolean algebra (*e.g.* a powerset $\mathcal{P}(X)$) is a CCC.

Products are given by conjunctions $A \wedge B$.

We define exponentials as **implications**:

$$A \Rightarrow B \;\; = \;\; \neg A \vee B$$

Example: Boolean Algebras

A Boolean algebra (*e.g.* a powerset $\mathcal{P}(X)$) is a CCC.

Products are given by conjunctions $A \wedge B$.

We define exponentials as **implications**:

$$A \Rightarrow B \;\; = \;\; \neg A \vee B$$

Evaluation is just Modus Ponens:

$$(A \Rightarrow B) \wedge A \leq B$$

Example: Boolean Algebras

A Boolean algebra (*e.g.* a powerset $\mathcal{P}(X)$) is a CCC.

Products are given by conjunctions $A \wedge B$.

We define exponentials as **implications**:

$$A \Rightarrow B \;\; = \;\; \neg A \vee B$$

Evaluation is just Modus Ponens:

$$(A \Rightarrow B) \wedge A \leq B$$

Couniversality is the 'Deduction Theorem':

$$C \wedge A \leq B \iff C \leq A \Rightarrow B.$$

Example: Boolean Algebras

A Boolean algebra (*e.g.* a powerset $\mathcal{P}(X)$) is a CCC.

Products are given by conjunctions $A \wedge B$.

We define exponentials as **implications**:

$$A \Rightarrow B \;\; = \;\; \neg A \vee B$$

Evaluation is just Modus Ponens:

$$(A \Rightarrow B) \wedge A \leq B$$

Couniversality is the 'Deduction Theorem':

$$C \wedge A \leq B \iff C \leq A \Rightarrow B.$$

It is the Galois connection or adjunction we saw previously.

Example: Boolean Algebras

A Boolean algebra (*e.g.* a powerset $\mathcal{P}(X)$) is a CCC.

Products are given by conjunctions $A \wedge B$.

We define exponentials as **implications**:

$$A \Rightarrow B \;\; = \;\; \neg A \vee B$$

Evaluation is just Modus Ponens:

$$(A \Rightarrow B) \wedge A \leq B$$

Couniversality is the 'Deduction Theorem':

$$C \wedge A \leq B \iff C \leq A \Rightarrow B.$$

It is the Galois connection or adjunction we saw previously.

More generally, we have the notion of **residuated meet semilattice** as the poset version of cartesian closed category.

The Connection To Logic

The Connection To Logic

Let $\mathcal{C}$ be a category. We shall interpret Formulas (or Types) as Objects of $\mathcal{C}$.

The Connection To Logic

Let $\mathcal{C}$ be a category. We shall interpret Formulas (or Types) as Objects of $\mathcal{C}$.

A morphism $f : A \longrightarrow B$ will then correspond to a **proof of B from assumption** A, *i.e.* a proof of $A \vdash B$. Note that the bare structure of a category only supports proofs from a single assumption.

The Connection To Logic

Let $\mathcal{C}$ be a category. We shall interpret Formulas (or Types) as Objects of $\mathcal{C}$.

A morphism $f : A \longrightarrow B$ will then correspond to a **proof of** B **from assumption** A, *i.e.* a proof of $A \vdash B$. Note that the bare structure of a category only supports proofs from a single assumption.

Now suppose $\mathcal{C}$ has finite products. A proof of

$$A_1, \ldots, A_k \vdash A$$

will correspond to a morphism

$$f : A_1 \times \cdots \times A_k \longrightarrow A.$$

Axiom, Conjunction

Axiom, Conjunction

Axiom

$$\frac{}{\Gamma, A \vdash A}\ \mathsf{Id} \qquad\qquad \frac{}{\pi_2 : \Gamma \times A \longrightarrow A}$$

Axiom, Conjunction

Axiom

$$\frac{}{\Gamma, A \vdash A}\ \mathsf{Id} \qquad\qquad \frac{}{\pi_2 : \Gamma \times A \longrightarrow A}$$

Conjunction

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge\text{-intro} \qquad\qquad \frac{f : \Gamma \longrightarrow A \qquad g : \Gamma \longrightarrow B}{\langle f, g \rangle : \Gamma \longrightarrow A \times B}$$

Axiom, Conjunction

Axiom

$$\frac{}{\Gamma, A \vdash A}\ \mathsf{Id} \qquad\qquad \frac{}{\pi_2 : \Gamma \times A \longrightarrow A}$$

Conjunction

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge\text{-intro} \qquad\qquad \frac{f : \Gamma \longrightarrow A \qquad g : \Gamma \longrightarrow B}{\langle f, g \rangle : \Gamma \longrightarrow A \times B}$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ \wedge\text{-elim-1} \qquad\qquad \frac{f : \Gamma \longrightarrow A \times B}{\pi_1 \circ f : \Gamma \longrightarrow A}$$

Axiom, Conjunction

Axiom

$$\frac{}{\Gamma, A \vdash A}\ \mathsf{Id} \qquad\qquad \frac{}{\pi_2 : \Gamma \times A \longrightarrow A}$$

Conjunction

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge\text{-intro} \qquad\qquad \frac{f : \Gamma \longrightarrow A \qquad g : \Gamma \longrightarrow B}{\langle f, g \rangle : \Gamma \longrightarrow A \times B}$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ \wedge\text{-elim-1} \qquad\qquad \frac{f : \Gamma \longrightarrow A \times B}{\pi_1 \circ f : \Gamma \longrightarrow A}$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ \wedge\text{-elim-2} \qquad\qquad \frac{f : \Gamma \longrightarrow A \times B}{\pi_2 \circ f : \Gamma \longrightarrow B}$$

Implication

Implication

Now let $\mathcal{C}$ be cartesian closed.

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B} \supset\text{-intro} \qquad\qquad \frac{f : \Gamma \times A \longrightarrow B}{\Lambda(f) : \Gamma \longrightarrow (A \Rightarrow B)}$$

Implication

Now let $\mathcal{C}$ be cartesian closed.

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B}\ \supset\text{-intro} \qquad \frac{f : \Gamma \times A \longrightarrow B}{\Lambda(f) : \Gamma \longrightarrow (A \Rightarrow B)}$$

$$\frac{\Gamma \vdash A \supset B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ \supset\text{-elim} \qquad \frac{f : \Gamma \longrightarrow (A \Rightarrow B) \qquad g : \Gamma \longrightarrow A}{\mathsf{Ap}_{A,B} \circ \langle f, g \rangle : \Gamma \longrightarrow B}$$

Implication

Now let $\mathcal{C}$ be cartesian closed.

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B}\ \supset\text{-intro} \qquad\qquad \frac{f : \Gamma \times A \longrightarrow B}{\Lambda(f) : \Gamma \longrightarrow (A \Rightarrow B)}$$

$$\frac{\Gamma \vdash A \supset B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ \supset\text{-elim} \qquad \frac{f : \Gamma \longrightarrow (A \Rightarrow B) \qquad g : \Gamma \longrightarrow A}{\mathrm{Ap}_{A,B} \circ \langle f, g \rangle : \Gamma \longrightarrow B}$$

Moreover, the β- and η-equations are all then **derivable** from the equations of cartesian closed categories.

Implication

Now let $\mathcal{C}$ be cartesian closed.

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B} \supset\text{-intro} \qquad\qquad \frac{f : \Gamma \times A \longrightarrow B}{\Lambda(f) : \Gamma \longrightarrow (A \Rightarrow B)}$$

$$\frac{\Gamma \vdash A \supset B \qquad \Gamma \vdash A}{\Gamma \vdash B} \supset\text{-elim} \qquad \frac{f : \Gamma \longrightarrow (A \Rightarrow B) \qquad g : \Gamma \longrightarrow A}{\mathsf{Ap}_{A,B} \circ \langle f, g \rangle : \Gamma \longrightarrow B}$$

Moreover, the β- and η-equations are all then **derivable** from the equations of cartesian closed categories.
See the Notes for details.

Implication

Now let $\mathcal{C}$ be cartesian closed.

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B}\ \supset\text{-intro} \qquad \frac{f : \Gamma \times A \longrightarrow B}{\Lambda(f) : \Gamma \longrightarrow (A \Rightarrow B)}$$

$$\frac{\Gamma \vdash A \supset B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ \supset\text{-elim} \qquad \frac{f : \Gamma \longrightarrow (A \Rightarrow B) \qquad g : \Gamma \longrightarrow A}{\mathsf{Ap}_{A,B} \circ \langle f, g \rangle : \Gamma \longrightarrow B}$$

Moreover, the β- and η-equations are all then **derivable** from the equations of cartesian closed categories.
See the Notes for details.

So cartesian closed categories are **models** of $\wedge$, $\supset$-logic, at the level of **proofs** and **proof transformations**, and of simply typed λ-calculus, at the level of **terms** and **equations between terms**.

Implication

Now let $\mathcal{C}$ be cartesian closed.

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B} \supset\text{-intro} \qquad\qquad \frac{f : \Gamma \times A \longrightarrow B}{\Lambda(f) : \Gamma \longrightarrow (A \Rightarrow B)}$$

$$\frac{\Gamma \vdash A \supset B \qquad \Gamma \vdash A}{\Gamma \vdash B} \supset\text{-elim} \qquad \frac{f : \Gamma \longrightarrow (A \Rightarrow B) \qquad g : \Gamma \longrightarrow A}{\mathsf{Ap}_{A,B} \circ \langle f, g \rangle : \Gamma \longrightarrow B}$$

Moreover, the β- and η-equations are all then **derivable** from the equations of cartesian closed categories.
See the Notes for details.

So cartesian closed categories are **models** of $\wedge$, $\supset$-logic, at the level of **proofs** and **proof transformations**, and of simply typed λ-calculus, at the level of **terms** and **equations between terms**.

This is the **Curry-Howard-Lambek correspondence**.

Consequences and Implications of the Curry-Howard-Lambek Correspondence

Consequences and Implications of the Curry-Howard-Lambek Correspondence

- For Logic: computational content of proofs.

Consequences and Implications of the Curry-Howard-Lambek Correspondence

- For Logic: computational content of proofs.

  Question: does this extend to other logics (e.g. Classical logic)?

Consequences and Implications of the Curry-Howard-Lambek Correspondence

- For Logic: computational content of proofs.

  Question: does this extend to other logics (e.g. Classical logic)?

- For Computer Science: a foundation for functional programming and its type systems. Hindley-Milner type-checking algorithm, Haskell, OCAML, C#, etc.

Consequences and Implications of the Curry-Howard-Lambek Correspondence

- For Logic: computational content of proofs.

  Question: does this extend to other logics (e.g. Classical logic)?

- For Computer Science: a foundation for functional programming and its type systems. Hindley-Milner type-checking algorithm, Haskell, OCAML, C#, etc.

- For Category theory. A link to formal languages and logic.

Consequences and Implications of the Curry-Howard-Lambek Correspondence

- For Logic: computational content of proofs.

  Question: does this extend to other logics (e.g. Classical logic)?

- For Computer Science: a foundation for functional programming and its type systems. Hindley-Milner type-checking algorithm, Haskell, OCAML, C#, etc.

- For Category theory. A link to formal languages and logic.

  λ-calculus as the 'internal language' of cartesian closed categories.

Consequences and Implications of the Curry-Howard-Lambek Correspondence

- For Logic: computational content of proofs.

  Question: does this extend to other logics (e.g. Classical logic)?

- For Computer Science: a foundation for functional programming and its type systems. Hindley-Milner type-checking algorithm, Haskell, OCAML, C#, etc.

- For Category theory. A link to formal languages and logic.

  λ-calculus as the 'internal language' of cartesian closed categories.

  We can use λ-calculus to do calculations in cartesian closed categories — and vice versa!

Consequences and Implications of the Curry-Howard-Lambek Correspondence

- For Logic: computational content of proofs.

  Question: does this extend to other logics (e.g. Classical logic)?

- For Computer Science: a foundation for functional programming and its type systems. Hindley-Milner type-checking algorithm, Haskell, OCAML, C#, etc.

- For Category theory. A link to formal languages and logic.

  λ-calculus as the 'internal language' of cartesian closed categories.

  We can use λ-calculus to do calculations in cartesian closed categories — and vice versa!

  Extension to categories corresponding to richer logics, e.g. **topos theory**.