Introduction to Intuitionistic Logic

Albert Lai

November 30, 2022

Motivation

Motivation: Classical Logic Surprises 1/2

Abbreviation:
RH: Riemann Hypothesis (Wikipedia entry)
P≠NP: A famous CS conjecture (Wikipedia entry)

Motivation: Classical Logic Surprises 1/2

Abbreviation:
RH: Riemann Hypothesis (Wikipedia entry)
P≠NP: A famous CS conjecture (Wikipedia entry)

The following holds in classical logic. How surprised are you?

$$(\mathrm{RH} \to \mathrm{P{\neq}NP}) \vee (\mathrm{P{\neq}NP} \to \mathrm{RH})$$

Motivation: Classical Logic Surprises 1/2

Abbreviation:
RH: Riemann Hypothesis (Wikipedia entry)
P≠NP: A famous CS conjecture (Wikipedia entry)

The following holds in classical logic. How surprised are you?

$$(\mathrm{RH} \rightarrow \mathrm{P}{\neq}\mathrm{NP}) \vee (\mathrm{P}{\neq}\mathrm{NP} \rightarrow \mathrm{RH})$$

Reason of surprise: You were thinking:

(proved RH → P≠NP) or (proved P≠NP → RH)

Motivation: Classical Logic Surprises 2/2

There exists an algorithm for the following. How surprised are you?

Input: Number n.
Output n if RH is true, else output $n + 1$.

(Hint: Generalizable from RH to any statement.)

Motivation: Classical Logic Surprises 2/2

There exists an algorithm for the following. How surprised are you?

Input: Number n.
Output n if RH is true, else output $n+1$.

(Hint: Generalizable from RH to any statement.)

Solution in classical logic:
RH is true or false.
If true, use this algorithm: Output n.
If false, use this algorithm: Output $n+1$.
One of them is the right algorithm, I don't know/care which.

Motivation: Classical Logic Surprises 2/2

There exists an algorithm for the following. How surprised are you?

Input: Number n.
Output n if RH is true, else output $n + 1$.

(Hint: Generalizable from RH to any statement.)

Solution in classical logic:
RH is true or false.
If true, use this algorithm: Output n.
If false, use this algorithm: Output $n + 1$.
One of them is the right algorithm, I don't know/care which.

Reason of surprise: You didn't expect the lack of commitment.
Excluded middle enables uninformative arguments.

Goal of Intuitionistic Logic

Goal: Avoid those uninformative surprises.

prove ...	Brouwer-Heyting-Kolmogorov interpretation
$A \wedge B$	prove A and prove B
$A \vee B$	prove A or prove B (tell me which)
$A \rightarrow B$	map proof of A to proof of B
$\neg A$	map proof of A to proof of anything
$\forall x \ldots$	map x to proof of ...
$\exists x \ldots$	construct example and prove ...

Goal of Intuitionistic Logic

Goal: Avoid those uninformative surprises.

prove …	Brouwer-Heyting-Kolmogorov interpretation
$A \wedge B$	prove A and prove B
$A \vee B$	prove A or prove B (tell me which)
$A \to B$	map proof of A to proof of B
$\neg A$	map proof of A to proof of anything
$\forall x \ldots$	map x to proof of …
$\exists x \ldots$	construct example and prove …

Is that constructive logic too? I didn't dig deeper, but:

Constructive logic

From Wikipedia, the free encyclopedia
Redirect page

↳ Intuitionistic logic

Talk Plan

This talk: Just propositional logic (no $\forall$ $\exists$).

- Proof rules of intuitionistic logic.
  - Gentzen's Natural Deduction.
  - What to add for classical logic.

Talk Plan

This talk: Just propositional logic (no $\forall$ $\exists$).

- Proof rules of intuitionistic logic.
  - Gentzen's Natural Deduction.
  - What to add for classical logic.
- Kripke's semantics of intuitionistic logic.
  - Warmup: Semantics of classical logic. (Truth values.)
  - Kripke's multiple world/state semantics. (Each state has its own truth values!)
  - (Why have semantics: concreteness; cross-check proof rule sanity; counterexamples for unprovable statements.)

Talk Plan

This talk: Just propositional logic (no $\forall$ $\exists$).

- Proof rules of intuitionistic logic.
  - Gentzen's Natural Deduction.
  - What to add for classical logic.
- Kripke's semantics of intuitionistic logic.
  - Warmup: Semantics of classical logic. (Truth values.)
  - Kripke's multiple world/state semantics. (Each state has its own truth values!)
  - (Why have semantics: concreteness; cross-check proof rule sanity; counterexamples for unprovable statements.)
- Correspondence with programming.
  - Curry-Howard correspondence.
  - statements $\rightarrow$ types, proofs $\rightarrow$ expressions
  - Formalization of constructive BHK view.

Proof Rules

Natural Deduction

Gentzen's idea: An operator op should come with:

- Introduction rules: How to deduce statements of the form $S\ op\ T$.
- Elimination rules: How to use statements of the form $S\ op\ T$ to deduce more statements. (Perhaps "consumption rules" is better.)

Probably obvious to you (both idea above and actual rules later.) Probably what you've always used. Hence "natural".

Proof Format in This Talk

I use a format similar to 1st-year baby-step proofs: One line per [intermediate] result, later lines deduced from earlier lines.

With indentation and markers for subproof structure to clarify scopes of local assumptions and results.

assumption A1
A1 holds here
assumption A2
A1 and A2 hold here
result R2
$\therefore$ A2 $\rightarrow$ R2
A1 and A2 $\rightarrow$ R2 hold, A2 and R2 don't
result R1
$\therefore$ A1 $\rightarrow$ R1

(Gentzen's original format was a tree.)

$\to$ Rules

(S, T are statements.)

$\to$-introduction: To deduce $S \to T$, make a subproof that assumes S, deduces T. The assumption is local to the subproof (and its subsubproofs etc.).

S	assume
...steps ...	
T	
$S \to T$	$\to$-intro

$\rightarrow$ Rules

(S, T are statements.)

$\rightarrow$-introduction: To deduce $S \rightarrow T$, make a subproof that assumes S, deduces T. The assumption is local to the subproof (and its subsubproofs etc.).

$\quad S$	assume
$\quad \dots$ steps $\dots$	
$\quad T$	
$S \rightarrow T$	$\rightarrow$-intro

$\rightarrow$-elimination: If $S \rightarrow T$ and S hold, can deduce T. "hold" includes in-scope assumptions and results.

$S \rightarrow T$	
S	
T	$\rightarrow$-elim

Example Proof Using $\rightarrow$ Rules

Prove $P \rightarrow ((P \rightarrow Q) \rightarrow Q)$:

1	P	assume
2	$P \rightarrow Q$	assume
3	Q	$\rightarrow$-e 2,1
4	$(P \rightarrow Q) \rightarrow Q$	$\rightarrow$-i
5	$P \rightarrow ((P \rightarrow Q) \rightarrow Q)$	$\rightarrow$-i

(Advice: Write or read from outer to inner.)

Example Proof Using $\rightarrow$ Rules

Prove $P \rightarrow ((P \rightarrow Q) \rightarrow Q)$:

1	P	assume
2	$P \rightarrow Q$	assume
3	Q	$\rightarrow$-e 2,1
4	$(P \rightarrow Q) \rightarrow Q$	$\rightarrow$-i
5	$P \rightarrow ((P \rightarrow Q) \rightarrow Q)$	$\rightarrow$-i

(Advice: Write or read from outer to inner.)

(You know what, in order from i to v :D

ii	P	assume
iv	$P \rightarrow Q$	assume
v	Q	$\rightarrow$-e iv,ii
iii	$(P \rightarrow Q) \rightarrow Q$	$\rightarrow$-i
i	$P \rightarrow ((P \rightarrow Q) \rightarrow Q)$	$\rightarrow$-i

)

$\wedge$ Rules

(S, T are statements.)

$\wedge$**-introduction**:

S
T
$S \wedge T$ $\quad$ $\wedge$-intro

$\wedge$ Rules

(S, T are statements.)

$\wedge$-introduction:

S	
T	
$S \wedge T$	$\wedge$-intro

$\wedge$-elimination: Two of them:

$S \wedge T$		$S \wedge T$	
S	$\wedge$-elim	T	$\wedge$-elim

$\vee$ Rules

(S, T, R are statements.)

$\vee$-introduction: Two of them:

S		T	
$S \vee T$	$\vee$-intro	$S \vee T$	$\vee$-intro

$\vee$ Rules

(S, T, R are statements.)

$\vee$**-introduction**: Two of them:

S		T	
$S \vee T$	$\vee$-intro	$S \vee T$	$\vee$-intro

$\vee$**-elimination**: You know it as case analysis:

$S \vee T$		
	S	assume
	$\dots$	
	R	
	T	assume
	$\dots$	
	R	
R		$\vee$-elim

Example Proof 2

Prove $((P \to R) \vee (Q \to R)) \to ((P \wedge Q) \to R)$:

1	$(P \to R) \vee (Q \to R)$	assume
2	$P \wedge Q$	assume
3	$P \to R$	assume
4	P	$\wedge$-e 2
5	R	$\to$-e 3,4
6	$Q \to R$	assume
7	Q	$\wedge$-e 2
8	R	$\to$-e 6,7
9	R	$\vee$-e 1
10	$(P \wedge Q) \to R$	$\to$-i
11	$((P \to R) \vee (Q \to R)) \to ((P \wedge Q) \to R)$	$\to$-i

Rules for $\perp$ “false” and $\top$ “true”

$\perp$-elimination: From $\perp$ “false” deduce anything. S is any statement you like:

$$\begin{array}{ll} \perp & \\ S & \perp\text{-elim} \end{array}$$

Closely related to $\neg$. (How to obtain $\perp$ in the first place? From contradictory results. Forward reference: $\neg$-elim.)

Rules for $\perp$ "false" and $\top$ "true"

$\perp$-elimination: From $\perp$ "false" deduce anything. S is any statement you like:

$$\frac{\perp}{S}\ \perp\text{-elim}$$

Closely related to $\neg$. (How to obtain $\perp$ in the first place? From contradictory results. Forward reference: $\neg$-elim.)

$\top$-introduction: Can always deduce $\top$ "true".

$$\top \quad \top\text{-intro}$$

$\top$ doesn't help you deduce anything else, so no elim rule.

Looks useless, but elegant dual of $\perp$, and corresponds to something useful in programming.

¬ Rules

(S is a statement.)

¬-introduction:

$\quad S$	assume
$\quad \ldots$	
$\quad \bot$	
$\neg S$	¬-intro

¬ Rules

(S is a statement.)

¬-introduction:

S	assume
$\dots$	
$\perp$	
$\neg S$	¬-intro

¬-elimination (also how to deduce $\perp$):

$\neg S$	
S	
$\perp$	¬-elim

$\neg$ Rules

(S is a statement.)

$\neg$-introduction:

$$\begin{array}{ll} \quad S & \text{assume} \\ \quad \ldots & \\ \quad \bot & \\ \neg S & \neg\text{-intro} \end{array}$$

$\neg$-elimination (also how to deduce $\bot$):

$$\begin{array}{ll} \neg S & \\ S & \\ \bot & \neg\text{-elim} \end{array}$$

Equivalently, $\neg S$ is syntax sugar for $S \to \bot$, use $\to$ rules.

Example Proof 3

Prove $\neg(P \vee Q) \rightarrow \neg P$:

1	$\neg(P \vee Q)$	assume
2	P	assume
3	$P \vee Q$	$\vee$-i 2
4	$\perp$	$\neg$-e 1,3
5	$\neg P$	$\neg$-i
6	$\neg(P \vee Q) \rightarrow \neg P$	$\rightarrow$-i

Gentzen Tree Format Example

Example 2 in Gentzen's format:

$$
\dfrac{
  \dfrac{
    \dfrac{}{(P \to R) \lor (Q \to R)}\,1
    \qquad
    \dfrac{
      \dfrac{}{P \to R}\,3
      \quad
      \dfrac{\dfrac{}{P \land Q}\,2}{P}\,\land\mathrm{e}
    }{R}\,\to\mathrm{e}
    \qquad
    \dfrac{
      \dfrac{}{Q \to R}\,6
      \quad
      \dfrac{\dfrac{}{P \land Q}\,2}{Q}\,\land\mathrm{e}
    }{R}\,\to\mathrm{e}
  }{
    \dfrac{R}{(P \land Q) \to R}\,\to\mathrm{i}^{2}
  }\,\lor\mathrm{e}^{3,6}
}{((P \to R) \lor (Q \to R)) \to ((P \land Q) \to R)}\,\to\mathrm{i}^{1}
$$

"$\lor\mathrm{e}^{3,6}$" means it consumes assumptions 3 and 6. Similar for others.

With a tree you need to duplicate multi-used assumptions e.g. 2.

Exercise for You

Prove $\neg\neg(P \vee \neg P)$.

In general, for propositional logic, statement S is classically provable iff $\neg\neg S$ is intuitionistically provable.

With $\forall$ and $\exists$, a similar result holds, but more elaborate translation.

Wikipedia entry: double-negation translation.

Classical Rules

Add one for classical logic, one is enough, more is OK.

(S, T are statements.)

Excluded middle: $S \vee \neg S$ is an axiom.

$S \vee \neg S$ excluded middle

Classical Rules

Add one for classical logic, one is enough, more is OK.

(S, T are statements.)

Excluded middle: $S \vee \neg S$ is an axiom.

$S \vee \neg S$ excluded middle

$\neg\neg$-elimination: From $\neg\neg S$ deduce S. Equivalently make a subproof: assume $\neg S$, deduce $\bot$, i.e., proof by contradiction.

$\neg\neg S$
S $\neg\neg$-elim

Classical Rules

Add one for classical logic, one is enough, more is OK.

(S, T are statements.)

Excluded middle: $S \vee \neg S$ is an axiom.

$S \vee \neg S$ excluded middle

$\neg\neg$-elimination: From $\neg\neg S$ deduce S. Equivalently make a subproof: assume $\neg S$, deduce $\perp$, i.e., proof by contradiction.

$\neg\neg S$
S $\neg\neg$-elim

Pierce's law: $((S \to T) \to S) \to S$ is an axiom. Also has subproof equivalent (example next slide).

$((S \to T) \to S) \to S$ Pierce

Classical Rules

Add one for classical logic, one is enough, more is OK.

(S, T are statements.)

Excluded middle: $S \vee \neg S$ is an axiom.

$S \vee \neg S$ excluded middle

$\neg\neg$-elimination: From $\neg\neg S$ deduce S. Equivalently make a subproof: assume $\neg S$, deduce $\perp$, i.e., proof by contradiction.

$\neg\neg S$
S $\neg\neg$-elim

Pierce's law: $((S \to T) \to S) \to S$ is an axiom. Also has subproof equivalent (example next slide).

$((S \to T) \to S) \to S$ Pierce

There are others.

Example: Use Pierce for Excluded Middle

“To B? Or not to B?”

1	$(B \vee \neg B) \to \bot$	assume
2	B	assume
3	$B \vee \neg B$	$\vee$-i 2
4	$\bot$	$\to$-e 1,3
5	$\neg B$	$\neg$-i
6	$B \vee \neg B$	$\vee$-i 5
7	$B \vee \neg B$	Pierce

“That’s the resolution. :D”

Semantics

[Dis]Orientation

Suppose you study linear algebra:

"B is a basis iff B spans V and B is linearly independent."

Study linear algebra (green) using logic (black "iff", "and").
(Which logic? Up to you.)

[Dis]Orientation

Suppose you study linear algebra:

"B is a basis iff B spans V and B is linearly independent."

Study linear algebra (green) using logic (black "iff", "and").
(Which logic? Up to you.)

Suppose you study logic:

"I satisfies $S \wedge T$ iff I satisfies S and I satisfies T."

Study logic using... logic?!
"$\wedge$" vs "and", what's the difference?!

Orientation: Object And Meta

What you wish they said in a logic course

Normal in CS: Python interpreter written in some language. Which language? Currently C, but could be Python even. Most C compilers are written in C. “This is fine.”

Orientation: Object And Meta

What you wish they said in a logic course

Normal in CS: Python interpreter written in some language. Which language? Currently C, but could be Python even. Most C compilers are written in C. “This is fine.”

Study {classical, intuitionistic, other} logic using {classical, intuitionistic, other} logic.

Terminology: Study “object logic/language” using “meta logic/language”.

This talk: Meta logic (“iff”, “and”, etc.) is classical. (Some people use intuitionsitic logic, more work but same results.)

“I satisfies $S \wedge T$”: “S”, “T” in black meta, placeholders for object-level things, “meta variables”.

Semantics of Classical Propositional Logic

(Statements are made of operators and atomic propositions (aka propositional variables).)

An interpretation I consists of: Function π_I from atomic propositions to booleans $\{0, 1\}$. (Truth assignment.)

Semantics of Classical Propositional Logic

(Statements are made of operators and atomic propositions (aka propositional variables).)

An interpretation I consists of: Function π_I from atomic propositions to booleans $\{0, 1\}$. (Truth assignment.)

"I satisfies S", notated "$I \Vdash S$", defined below. Intention: "S is true when interpreted under I".

"I does not satisfy S", "I falsifies S", notated "$I \nVdash S$".

$I \Vdash \top$		
$I \nVdash \bot$		
$I \Vdash p$	iff	$\pi_I(p) = 1$ (for atomic propositions)
$I \Vdash S \land T$	iff	$I \Vdash S$ and $I \Vdash T$
$I \Vdash S \lor T$	iff	$I \Vdash S$ or $I \Vdash T$
$I \Vdash S \to T$	iff	$I \nVdash S$ or $I \Vdash T$
$I \Vdash \neg S$	iff	$I \nVdash S$

Soundness, Completeness, Counterexamples

Soundness theorem: If S provable, then $I \Vdash S$ for all I.
(Proof rules are safe.)

Completeness theorem: If $I \Vdash S$ for all I, then S provable.
(Proof rules are sufficient.)

Example application of soundness:

(P, Q atomic propositions.) $P \to P \land Q$ unprovable because can falsify (counterexample):

$\pi_I(P) = 1$
$\pi_I(Q) = 0$

Interpretation of Intuitionistic Propositional Logic

Kripke's idea: Multiple worlds, each its own truth.

A Kripke structure M consists of the following data:

- Set of "worlds / states / world states".
- Each state w: Function π_w from atomic propositions to booleans $\{0, 1\}$.
- Pre-order $\sqsubseteq_M$ over states. (Wlog partial order, reflexive transitive closure of directed [acyclic] graph.)
  When $v \sqsubseteq_M w$: v ancestor, past; w descendent, future.
- Start state r_M, ancestor of all states in M.

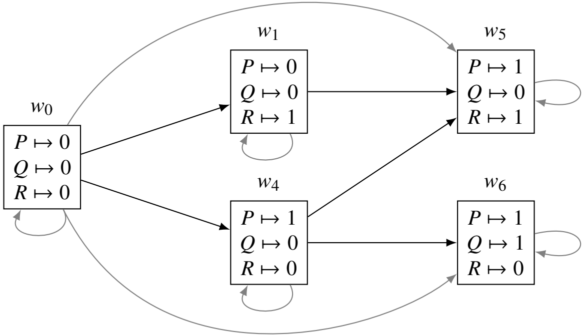
under this constraint:

- Persistence: If $v \sqsubseteq_M w$ and $\pi_v(p) = 1$, then $\pi_w(p) = 1$.

Example: Next slide.

Example Kripke Structure

(Black edges DAG. Reflexive transitive closure adds gray edges.)



Intuitionistic Satisfaction

Satisfaction depends on states too. "$M, v \Vdash S$".

Main idea: $\to$ and $\neg$ check all futures; the rest can look classical.

$M, v \Vdash \top$		
$M, v \nVdash \bot$		
$M, v \Vdash p$	iff	$\pi_v(p) = 1$ (for atomic propositions)
$M, v \Vdash S \wedge T$	iff	$M, v \Vdash S$ and $M, v \Vdash T$
$M, v \Vdash S \vee T$	iff	$M, v \Vdash S$ or $M, v \Vdash T$
$M, v \Vdash S \to T$	iff	for all $w \sqsupseteq_M v$, $M, w \nVdash S$ or $M, w \Vdash T$
$M, v \Vdash \neg S$	iff	for all $w \sqsupseteq_M v$, $M, w \nVdash S$

Intuitionistic Satisfaction

Satisfaction depends on states too. "$M, v \Vdash S$".

Main idea: $\to$ and $\neg$ check all futures; the rest can look classical.

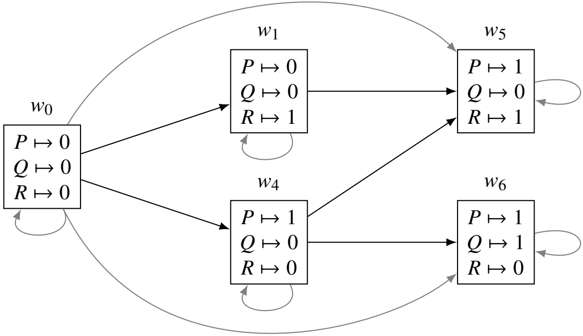
$M, v \Vdash \top$		
$M, v \nVdash \bot$		
$M, v \Vdash p$	iff	$\pi_v(p) = 1$ (for atomic propositions)
$M, v \Vdash S \wedge T$	iff	$M, v \Vdash S$ and $M, v \Vdash T$
$M, v \Vdash S \vee T$	iff	$M, v \Vdash S$ or $M, v \Vdash T$
$M, v \Vdash S \to T$	iff	for all $w \sqsupseteq_M v$, $M, w \nVdash S$ or $M, w \Vdash T$
$M, v \Vdash \neg S$	iff	for all $w \sqsupseteq_M v$, $M, w \nVdash S$

Motivated by another intuitionistic philosophy: Truth is subjective, constructed; $\neg$ ("impossible") and $\to$ ("must follow") must consider possible future developments.

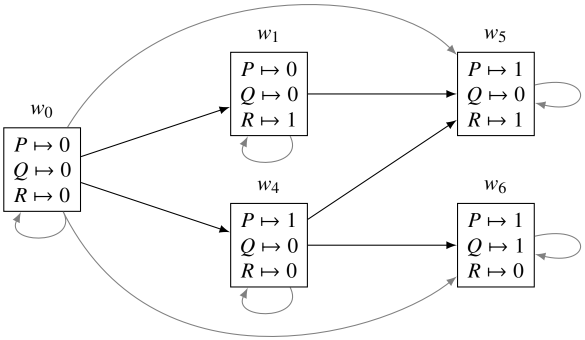
If hard to swallow for you, perhaps consider a constructed civilization...

Motivation for Kripke Semantics: Gamify :D

Design a strategy game. Fictional civilization, set of techs.
Player chooses techs to enable and when. Restrictions apply.
Enabled techs persist (until end of game).
Induces state diagram of permitted evolutions.
Example: ("$\mapsto 0$" not [yet] enabled, "$\mapsto 1$" enabled)



Motivation for Kripke Semantics



For this fictional civilization:

When at w_0 or w_4, some future has Q, premature to claim $\neg Q$.

If they hit w_1 or w_5, Q unobtainable forever, can claim $\neg Q$.

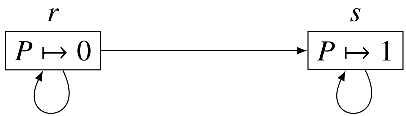
Soundness, Completeness, Counterexamples

Soundness theorem: If S provable, then $M, r_M \Vdash S$ for all M.

Completeness theorem: If $M, r_M \Vdash S$ for all I, then S provable.

Example application of soundness:

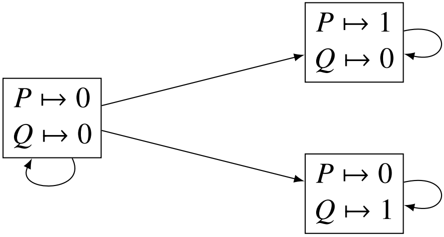
(P atomic proposition.) $P \vee \neg P$ unprovable because can falsify (counterexample):



$M, r \nVdash P$
$M, r \nVdash \neg P$ because in one future $M, s \Vdash P$.
So $M, r \nVdash P \vee \neg P$.

One More Counterxample

(P, Q atomic propositions.) Falsify $(P \to Q) \vee (Q \to P)$:



$P \to Q$ ruined by one possible future.
$Q \to P$ ruined by the other.

Correspondence with Programming

Church's Typed Lambda Calculus

plus some data type constructions

Toy programming language with

- Function types $S \to T$
  Functions without names.
  You might write $x \mapsto foo$. I follow Church: $\lambda x \cdot foo$
- Cartesian product types $S \times T$
- Disjoint union types (sum types) $S + T$
  Pools values from both S and T, with tagging to remember origin.
- A single-value type "unit", value name "•".
  Useful for e.g. $S + \text{unit}$.
- A no-value type "empty".
  Sounds useless but recall: For every set S, there exists a unique function $\emptyset \to S$.

Typing Rules: Motivation

Need rules to say which expressions have which types, in fact which expressions are legal at all.

Plus, they say what you can do with values according to types. (APIs of types.)

Let me put it this way: You want to know:

- Introduction rules: How to make values of a type.
- Elimination rules: How to use values of a type to make more values of more types.

Sounds familiar? ;)

Notation: "*expression* : *type*"

$\rightarrow$ Rules

(Types S, T. Variable x. Expressions e, f.)

$\rightarrow$-introduction:

$$\begin{array}{ll} \left[\begin{array}{l} x : S \\ \ldots \text{steps} \ldots \\ e : T \end{array}\right. & \text{local var} \\ (\lambda x \cdot e) : S \rightarrow T & \rightarrow\text{-intro} \end{array}$$

$\rightarrow$-elimination:

$$\begin{array}{ll} f : S \rightarrow T & \\ e : S & \\ f(e) : T & \rightarrow\text{-elim} \end{array}$$

Deja Vu?

Example Program Using → Rules

Write a function $P \to ((P \to Q) \to Q)$:

1	$x : P$	local var
2	$g : P \to Q$	local var
3	$g(x) : Q$	$\to$-e 2,1
4	$(\lambda g \cdot g(x)) : (P \to Q) \to Q$	$\to$-i
5	$(\lambda x \cdot \lambda g \cdot g(x)) : P \to ((P \to Q) \to Q)$	$\to$-i

Did I just copy-paste?

$\times$ Rules

(Types S, T. Expressions s, t, e.)

$\times$-introduction:

$s : S$
$t : T$
$(s, t) : S \times T \quad \times\text{-intro}$

$\times$-elimination: Two of them:

$e : S \times T$
$\text{fst}(e) : S \quad \times\text{-elim}$

$e : S \times T$
$\text{snd}(e) : T \quad \times\text{-elim}$

(fst “first”, snd “second”. Projections.)

Did I just search-replace?

+ Rules

(Types S, T, R. Expressions s, t, e, r_1, r_2. Variables x, y.)

+**-introduction**: Two of them:

$$\frac{s : S}{\mathsf{inl}(s) : S + T}\ \text{+-intro} \qquad \frac{t : T}{\mathsf{inr}(t) : S + T}\ \text{+-intro}$$

inl "inject left", inr "inject right". Tags.

+ Rules

(Types S, T, R. Expressions s, t, e, r_1, r_2. Variables x, y.)

+-introduction: Two of them:

$s : S$
inl$(s) : S + T$ +-intro

$t : T$
inr$(t) : S + T$ +-intro

inl "inject left", inr "inject right". Tags.

+-elimination:

$e : S + T$

$x : S$ local var
$\ldots$
$r_1 : R$

$y : T$ local var
$\ldots$
$r_2 : R$

(case e of inl$(x) \mapsto r_1$; inr$(y) \mapsto r_2$) $: R$ +-elim

Example Program 2

Program for $((P \to R) + (Q \to R)) \to ((P \times Q) \to R)$:

$x : (P \to R) + (Q \to R)$	local var
$y : P \times Q$	local var
$f : P \to R$	local var
$\mathsf{fst}(y) : P$	$\times$-e
$f(\mathsf{fst}(y)) : R$	$\to$-e
$g : Q \to R$	local var
$\mathsf{snd}(y) : Q$	$\times$-e
$g(\mathsf{snd}(y)) : R$	$\to$-e
$C : R$	$+$-e
$C = \text{case } x \text{ of inl}(f) \mapsto f(\mathsf{fst}(y)); \text{inr}(g) \mapsto g(\mathsf{snd}(y))$	
$(\lambda y \cdot C) : (P \times Q) \to R$	$\to$-i
$(\lambda x \cdot \lambda y \cdot C) : ((P \to R) + (Q \to R)) \to ((P \times Q) \to R)$	$\to$-i

Logic Wednesday Buy Proof Get Program Free?

unit, empty, ¬

unit-introduction: Just one value, trivial to make:

$\bullet : \text{unit}$ unit-intro

No information (cf. information theory), no elim rule.

Useful! unit + unit for booleans, $S \to (T + \text{unit})$ partial functions.

unit, empty, ¬

unit-introduction: Just one value, trivial to make:

$$\frac{}{\bullet : \text{unit}}\ \text{unit-intro}$$

No information (cf. information theory), no elim rule.

Useful! unit + unit for booleans, $S \to (T + \text{unit})$ partial functions.

empty-elimination: No value, no intro, unlimited information extractible (cf. information theory).

$$\frac{e : \text{empty}}{\text{miracle}(e) : S}\ \text{empty-elim}$$

unit, empty, ¬

unit-introduction: Just one value, trivial to make:

$\bullet$: unit   unit-intro

No information (cf. information theory), no elim rule.

Useful! unit + unit for booleans, $S \to (T + \text{unit})$ partial functions.

empty-elimination: No value, no intro, unlimited information extractible (cf. information theory).

e : empty
miracle(e) : S empty-elim

¬-definition: $\neg S$ syntax sugar for $S \to$ empty.

All logical operators accounted for!

Example Program 3

Program for $\neg(P + Q) \to \neg P$

i.e., $((P + Q) \to \text{empty}) \to (P \to \text{empty})$

$x : (P + Q) \to \text{empty}$	local var
$p : P$	local var
$\text{inl}(p) : P + Q$	+-i
$x(\text{inl}(p)) : \text{empty}$	$\to$-e
$(\lambda p \cdot x(\text{inl}(p))) : P \to \text{empty}$	$\to$-i
$(\lambda x \cdot \lambda p \cdot x(\text{inl}(p))) : ((P + Q) \to \text{empty}) \to (P \to \text{empty})$	$\to$-i

Your training is complete!

Curry-Howard Correspondence

Curry and Howard independently were the first to notice the uncanny resemblance.

statements	$\leftrightarrow$	types
proofs	$\leftrightarrow$	values, expressions, programs
normalize a proof (not in this talk)	$\leftrightarrow$	evaluate an expression

Not obvious back then, even Church missed it! Obvious today by notation design with benefit of hindsight.

Some logicians even go: Proofs are elements of statements.

Recall

$$(\lambda x \cdot \lambda g \cdot g(x)) : P \to ((P \to Q) \to Q)$$

as per BHK, literally maps proofs of P to proofs of $(P \to Q) \to Q$.

Example: The Lean Theorem Prover

A year ago we had talks on Lean.

Latest of several theorem provers based on the correspondence: one language for both proofs and programs.

Lean is a functional programming language; tactical proofs you saw are syntax sugar for programs.

My example proof-programs in Lean: File examples.lean

Can load optional library for classical logic, or add your own classical axioms (or any axioms). (My example file adds my own Pierce axiom.)

Further Readings (What I Read)

Further Readings (What I Read)

Philip Wadler. Propositions as Types. (Paper and lecture video.) Communications of the ACM, 58(12):75–84, 2015

Richard Bornat. Proof and Disproof in Formal Logic: An Introduction for Programmers. Oxford University Press.

Saul A. Kripke. Semantical Analysis of Intuitionistic Logic I. Formal Systems and Recursive Functions (Proceedings of the Eighth Logic Colloquium at Oxford, July, 1963), 92–130.

Gerhard Gentzen. Untersuchungen über das logische Schließen I. Mathematische Zeitschrift, 39:176–210, 1935.