A brief introduction to Logic and its applications

Classical, Intuitionistic and Hoare

Benoît Viguier

October 7, 2016

Overview

Classical Logic

Components

Constants:

- `true`
- `false`

Logical propositions:

p, q, r ...

operator	semantic	C
$\neg$	not	!
$\wedge$	and	&&
$\vee$	or	||
$\Rightarrow$	imply	...? ...: 1
$\Leftrightarrow$	if and only if	

Truth Table

p	q	$p \wedge q$	$p \vee q$	$p \Rightarrow q$	$p \Leftrightarrow q$
false	false	false	false	true	true
false	true	false	true	true	false
true	false	false	true	false	false
true	true	true	true	true	true

p	$\neg p$
false	true
true	false

A logical proposition P composed of atomic literals $(p, q, \ldots)$ can therefore be evaluated and exhaustively tested : same principle as the boolean.

Truth Table

p	q	$p \wedge q$	$p \vee q$	$p \Rightarrow q$	$p \Leftrightarrow q$
false	false	false	false	true	true
false	true	false	true	true	false
true	false	false	true	false	false
true	true	true	true	true	true

p	$\neg p$
false	true
true	false

A logical proposition P composed of atomic literals $(p, q, \ldots)$ can therefore be evaluated and exhaustively tested : same principle as the boolean.

Truth table are useful to prove simple statements such as : $\neg\neg p \Rightarrow p$

p	$\neg p$	$\neg\neg p$	$\neg\neg p \Rightarrow p$
false	true	false	true
true	false	true	true

Truth Table

p	q	$p \wedge q$	$p \vee q$	$p \Rightarrow q$	$p \Leftrightarrow q$
false	false	false	false	true	true
false	true	false	true	true	false
true	false	false	true	false	false
true	true	true	true	true	true

p	$\neg p$
false	true
true	false

A logical proposition P composed of atomic literals $(p, q, \ldots)$ can therefore be evaluated and exhaustively tested : same principle as the boolean.

Truth table are useful to prove simple statements such as : $\neg\neg p \Rightarrow p$

p	$\neg p$	$\neg\neg p$	$\neg\neg p \Rightarrow p$
false	true	false	true
true	false	true	true

Complexity

If we have n atomic propositions, the truth table will contain 2^n rows...

Brief History of Logic and Formalism

David Hilbert (1862 – 1943)

Entscheidungsproblem (1928):

There should be an algorithm for deciding the truth or falsity of any mathematical statement.

Precondition:

Logic completeness = every provable statement is true and every true statement is provable.

Kurt Gödels (1906 – 1978)

Incompleteness theorem (1931):

Any consistent formal system that includes enough of the theory of the natural numbers is incomplete: there are true statements expressible in its language that are unprovable within the system.

Any logic that includes arithmetic could encode :
"This statement is not provable".

"This statement is not provable"

If it is False...

then it is provable, and **you would have proven something False**...

“This statement is not provable”

If it is False...

then it is provable, and **you would have proven something False**...

try to prove it is True

and therefore unprovable...

At this time, to prove something, you didn’t need a formal definition of a proof. Just write its steps (kind of algorithm) and it is done.

"This statement is not provable"

If it is False...

then it is provable, and **you would have proven something False**...

try to prove it is True

and therefore unprovable...

At this time, to prove something, you didn't need a formal definition of a proof. Just write its steps (kind of algorithm) and it is done.

Better prove it is undecidable

This require a **formal definition of Proof**. Hence we need a the formal foundations of what is an algorithm

Alonzo Church (1903 - 1995)

Lambda calculus (1932):

Expression:

$$
\begin{array}{lll}
e & ::= x & \text{variable} \\
  & \mid \lambda x.e & \text{abstraction} \\
  & \mid ee & \text{application}
\end{array}
$$

λ-calculus (1/2)

$$
\begin{aligned}
e \quad ::= &\ x && \text{variable} \\
\mid &\ \lambda x.e && \text{abstraction} \\
\mid &\ ee && \text{application}
\end{aligned}
$$

λ-expression

$\lambda x.\ t$
Define a function of x where t is the body of the function.

β-reduction

$(\lambda x.\ t)s = t[x := s]$
Replace every occurence of x in t by s.

Examples:

$\lambda x.\ x$ is the identity function ($f : x \mapsto x$).
$\lambda x.\ y$ is the constant function ($f : x \mapsto y$).

λ-calculus (2/2)

$square\_add(x, y) = x \times x + y \times y$.
$square\_add(5, 2) = 25 + 4 = 29$.

```
/* In Java 8 since 2014 ! */
(x,y) -> x * x + y * y
```

```
/* Since C++14 ! */
[](auto a, auto b) { return a * a + b * b; }
```

In λ-calculus:

$$\begin{aligned} \lambda x.\ (\lambda y.\ (x * x + y * y))\ 5\ 2 &= \lambda y.\ (5 * 5 + y * y)\ 2 && (\beta\text{-reduction}) \\ &= (5 * 5 + 2 * 2) && (\beta\text{-reduction}) \\ &= 29 \end{aligned}$$

This is the root of functional programming
(Lisp *60*, Caml *85*, Haskell *87*, Coq *88*.

What is the link with Logic?

Gerhard Gentzen (1909 – 1945)

Natural Deduction and Sequent Calculus (1934):

Kind of proof calculus in which logical reasoning is expressed by inference rules closely related to the “natural” way of reasoning.

Notation:

$assumption \vdash goal$

Natural Deduction : Some rules (not all)

Modus Ponens

$$\frac{\vdash A \qquad \vdash A \Rightarrow B}{\vdash B}$$

If *I have* A and A implies B
Then *I can infer* B.

Notation: $\neg A := A \Rightarrow \bot$

$$\frac{A \vdash B}{\vdash A \Rightarrow B} \qquad \frac{\neg A \vdash \bot}{\vdash A} \qquad \frac{A \vdash \bot}{\vdash \neg A} \qquad \frac{}{\bot(\text{False}) \vdash}$$

$$\frac{}{\vdash \top(\text{True})} \qquad \frac{\vdash A \qquad \vdash B}{\vdash A \wedge B} \qquad \frac{\vdash A \wedge B}{\vdash A} \qquad \frac{\vdash A \wedge B}{\vdash B}$$

$$\frac{A, B \vdash}{A \wedge B \vdash} \qquad \frac{A \vdash \qquad B \vdash}{A \vee B \vdash} \qquad \frac{\vdash A}{\vdash A \vee B} \qquad \frac{\vdash B}{\vdash A \vee B}$$

Natural Deduction : Proof example

$$\dfrac{\dfrac{\dfrac{\overline{A, B \vdash A}\ \text{(assumption.)} \qquad \overline{A, B \vdash B}\ \text{(assumption.)}}{A, B \vdash A \wedge B}\ \text{(split.)}}{B \wedge A \vdash A \wedge B}\ \text{(destruct H.)}}{\vdash B \wedge A \Rightarrow A \wedge B}\ \text{(intro H.)}$$

Remark:

A proof is written from bottom to top (↑)
but read from top to bottom (↓).

Simply typed λ-Calculus (Church, 1940)

$$\frac{x : A \vdash N : B}{\vdash \lambda .x\ N : A \to B}$$

$$\frac{\vdash \lambda .x\ N : A \to B \qquad \vdash y : A}{\vdash \lambda .x\ N\ y : B}$$

Let's add the Pair structure :

$$\frac{\vdash x : A \qquad \vdash y : B}{\vdash (x, y) : A \times B}$$

$$\frac{\vdash p : A \times B}{\vdash fst\ p : A} \qquad\qquad \frac{\vdash p : A \times B}{\vdash snd\ p : B}$$

From proof to programs

$$\dfrac{\dfrac{\dfrac{z : B \times A \vdash z : B \times A}{z : B \times A \vdash snd\ z : A} \quad \dfrac{z : B \times A \vdash z : B \times A}{z : B \times A \vdash fst\ z : B}}{z : B \times A \vdash (snd\ z, fst\ z) : A \times B}}{\vdash \lambda.z\ (snd\ z, fst\ z) : B \times A \to A \times B}$$

This is called the Curry-Howard Correspondence (1969)

Isomorphisme between computer programs and logical proofs.

Intuitionistic Logic

The philosophy

Classical Logic

Propositional formulae are assign a Truth value (True or False).

Intuitionistic Logic (or Constructive Logic)

Propositional formulae in intuitionistic logic are considered **True** only when we have **direct evidence**, hence proof.
Propositional formulae in which there is no way to give evidence are therefore not provable.

Unavailables theorems

Reductio ad absurdum

$$\dfrac{\dfrac{\text{(unprovable)}}{((P \Rightarrow \bot) \Rightarrow \bot) \vdash P}}{\dfrac{\vdash ((P \Rightarrow \bot) \Rightarrow \bot) \Rightarrow P}{\vdash \neg\neg P \Rightarrow P}\text{(unfold not.)}}\text{(intro.)}$$

Tertium Non Datur

$$\dfrac{\dfrac{\dfrac{\dfrac{\text{(unprovable)}}{P \vdash \bot}}{\vdash P \Rightarrow \bot}\text{(intro.)}}{\vdash \neg P}\text{(unfold not.)}}{\vdash \neg P \vee P}\text{(left.)}$$

$$\dfrac{\dfrac{\text{(unprovable)}}{\vdash P}}{\vdash \neg P \vee P}\text{(right.)}$$

Curry-Howard and *Tertium Non Datur*

Another reason why one could not prove $P \lor \neg P$?

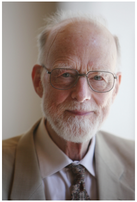
When you prove a statement such as $A \lor B$ you can extract a proof that answers whether A or B holds.

If we were able to prove the *excluded middle*, we could extract an algorithm that, given some proposition tells us whether it is valid or not (Curry-Howard).

This is not possible due to the undecidability :
if we take P to mean "program p halts on input x", the excluded middle would yield a decider for the halting problem, which cannot exist.

Hoare Logic

Sir Charles Antony Richard Hoare (1934 – T.B.D.)



Hoare Logic (1969):

It describes how the execution of a piece of code changes the state of the computation.

Notation: $\{P\}\ C\ \{Q\}$

Where P is the *pre-condition*, C is the *command* and Q is the *post-condition*. This is called a Hoare triple.

Toward the code Verification...

$$\frac{}{\{P\}\ \texttt{skip}\ \{P\}}\ \text{(skip)}$$

$$\frac{}{\{Q[e/x]\}\ \texttt{x:= e}\ \{Q\}}\ \text{(assign)}$$

$$\frac{\{P\}\ C_1\ \{Q\} \qquad \{Q\}\ C_2\ \{R\}}{\{P\}\ C_1;\ C_2\ \{R\}}\ \text{(seq)}$$

$$\frac{\{P \Rightarrow P'\} \qquad \{P'\}\ \texttt{C}\ \{Q'\} \qquad \{Q' \Rightarrow Q\}}{\{P\}\ \texttt{C}\ \{Q\}}\ \text{(consequence)}$$

$$\frac{\{B \wedge P\}\ C_1\ \{Q\} \qquad \{\neg B \wedge P\}\ C_2\ \{Q\}}{\{P\}\ \texttt{if B then}\ C_1\ \texttt{else}\ C_2\ \texttt{endif}\ \{Q\}}\ \text{(cond)}$$

Similar to Dataflow analysis, Operational Semantics...

Example...

$$\dfrac{\dfrac{\dfrac{}{\{a \geq b \vdash a+1 > b-1\}}\,(\text{omega})}{\{a \geq b \Rightarrow a+1 > b-1\}}\,(\text{intro}) \qquad \dfrac{}{\{a+1 > b-1\}\ \texttt{c := a + 1}\ \{c > b-1\}}\,(\text{ass})}{\{a \geq b\}\ \texttt{c := a + 1}\ \{c > b-1\}}\,(\text{con})$$

$$\dfrac{\vdots \qquad \dfrac{}{\{c > b-1\}\ \texttt{b := b - 1}\ \{c > b\}}\,(\text{ass})}{\{a \geq b\}\ \texttt{c := a + 1; b:= b - 1}\ \{c > b\}}\,(\text{seq})$$

Conclusion

To sum up

Classical Logic

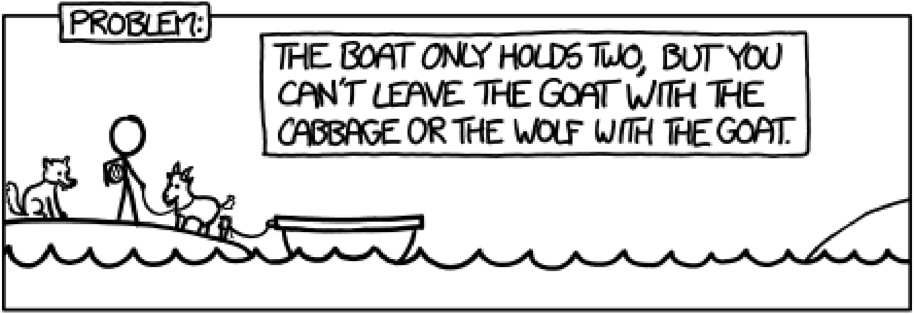
Propositional formulae are assign a Truth value (True or False).

Intuitionistic Logic (or Constructive Logic)

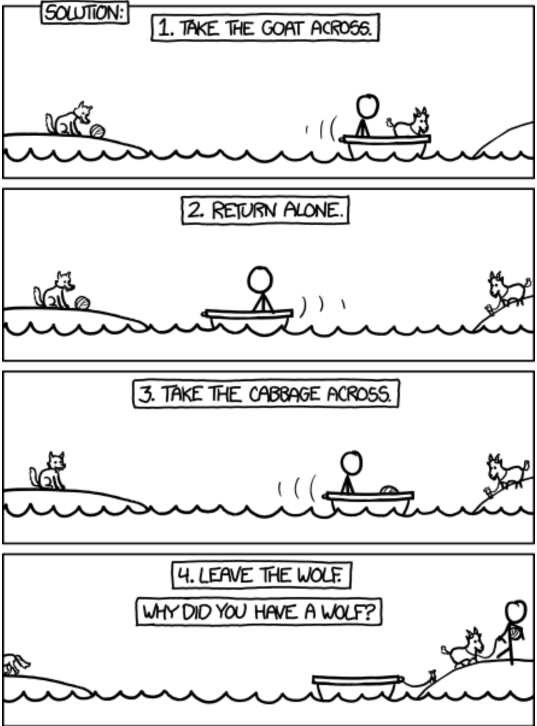
Propositional formulae in intuitionistic logic are considered **True** only when we have **direct evidence**, hence proof.
Calculations can also be included in a proof (e.g. 4-color theorem).

Hoare Logic

Formal model to prove the correctness of a program.



https://xkcd.com/1134/


SOLUTION:
1. TAKE THE GOAT ACROSS.
2. RETURN ALONE.
3. TAKE THE CABBAGE ACROSS.
4. LEAVE THE WOLF.
WHY DID YOU HAVE A WOLF?

Further Readings. . .

Intuitionistic Logic - *Stanford Encyclopedia of Philosophy*
Propositions as Types by *Philip Wadler* (paper)
Propositions as Types by *Philip Wadler* (video)
Introduction to Type Systems by *Delphine Demange*
Why are logical connectives and booleans separate in Coq?
Operational Semantics by *Delphine Demange*
Background reading on Hoare Logic by *Mike Gordon*