From abstract mathematics to verified programs[1]

Ulrich Berger
Swansea University

Hilbert-Bernays Summer School on Logic and Computation
Göttingen, July 23-27, 2018

[1] available at `www.cs.swan.ac.uk/∼csulrich/slides.html`

Aims of the course

The aim of this course to show that **logic** is a natural bridge between **mathematics** and **computation**.

We will study how valid reasoning in abstract mathematics leads to provably correct algorithms and hence certified computer programs

Plan of the course

Lecture 1: Introduction to logic

Lecture 2: Proofs as programs

Lecture 3: The magic of induction

Lecture 4: Program extraction

What is logic?

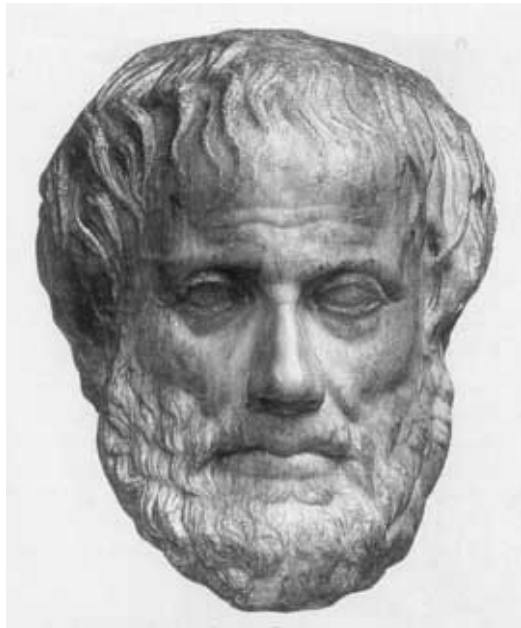
Logic is commonly described as the *science of reasoning* or the *study of the most general laws of truth* (Wikipedia).

Sherlock Holmes (Conan Doyle):

"Crime is common. Logic is rare. Therefore it is upon the logic rather than upon the crime that you should dwell."

"It is of the highest importance in the art of detection to be able to recognize, out of a number of facts, which are incidental and which vital. Otherwise your energy and attention must be dissipated instead of being concentrated."

Aristotle (384 - 322 BC)



Syllogisms (a form of logical deduction):

$$\frac{\text{All men are mortal} \qquad \text{Aristotle is a man}}{\text{Aristotle is mortal}}$$

Important branches of logic today

(1) Philosophical Logic

(2) Mathematical Logic

(3) Logic in Computer Science

Our course takes place mainly in (3) but we will heavily use tools from (2) and motivation from (1).

Background Reading

[1] D van Dalen, Logic and Structure, 3rd edition, Springer, 1994.

[2] A S Troelstra, D van Dalen, Constructivism in Mathematics, Vol. I, N-H, 1988.

[3] D Velleman, How to Prove It, 2nd edition, CUP, 1994.

[4] M Huth, M Ryan, Logic in Computer Science, CUP, 2004.

[5] B, K Miyamoto, H Schwichtenberg, M Seisenberger, Minlog - A Tool for Program Extraction for Supporting Algebra and Coalgebra, LNCS 6859, 2011.

[6] B, From coinductive proofs to exact real arithmetic: theory and applications, Logical Methods in Comput. Sci. 7(1), 2011,
`http://www.lmcs-online.org/ojs/viewarticle.php?id=704&layout=abstract`

[7] B, Logic for Computer Science, Swansea University Course Notes, 2017
`http://www.cs.swan.ac.uk/~csulrich/slides.html`

[8] Minlog
`http://www.mathematik.uni-muenchen.de/$\sim$logik/minlog/`

[9] O Petrovska, B, Prawf - an interactive proof systems.
`http://www.cs.swan.ac.uk/~csulrich/slides.html`

Contents of Lecture 1: Introduction to logic

- Propositional logic
- Applications in CS: Circuit minimization and SAT solving
- Proofs
- Predicate logic
- Undecidability and completeness

Propositional logic

If the Butler or the Maid is guilty, and the Maid or the Cook is guilty, then the Butler or the Cook is guilty.

$$(B \vee M) \wedge (M \vee C) \rightarrow B \vee C$$

Propositional formulas are built from

atomic propositions (or *propositional variables*) , here B, M, C, by

logical connectives, $\vee, \wedge, \rightarrow$.

Logical connectives

The logical connectives $\wedge, \vee, \rightarrow$ act as *Boolean functions*, that is, operations on the Boolean truth values

$$0 \text{ (“False”)}, 1 \text{ (“True”)}$$

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

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

A	B	$A \rightarrow B$
0	0	1
0	1	1
1	0	0
1	1	1

Negation

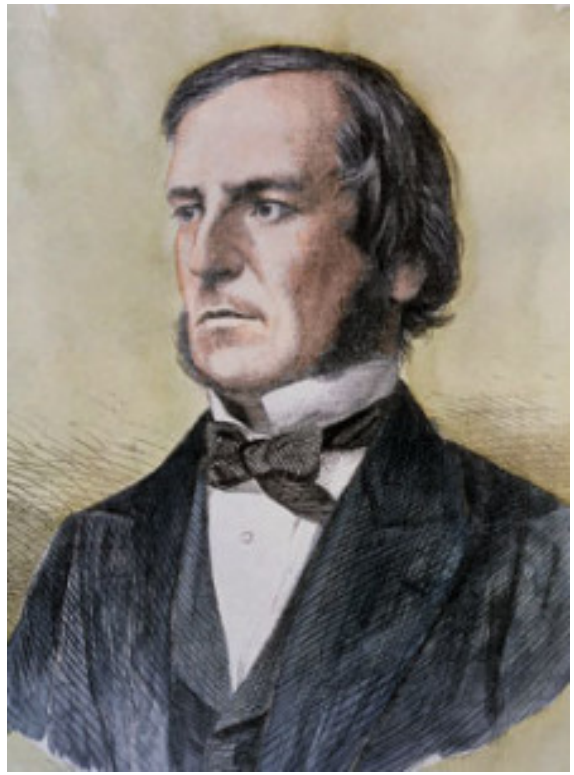
Negation can be defined by

$$\neg A = A \to \bot \qquad (\text{not } A)$$

where $\bot$ is a constant denoting 0 (Falsity)

A	$\neg A$
0	1
1	0

Boole



George Boole (1815 - 1864)

Boolean Algebras, a class of mathematical structures, are named after him. The simplest such structure is the Boolean Algebra of truth values $(\{0, 1\}, \wedge, \vee, \neg)$.

Does implication express causality?

The intuitive understanding of an implication, $A \to B$, is that A is a cause for B.

It rains $\to$ the street is wet

x is divisible by 4 $\to$ x is divisible by 2

$1 < 2 \to 3 + 4 = 7$

$1 = 0 \to$ I am the pope

Logic gates

Every formula defines a *Boolean function* or *Logic gate*.

Example of a ternary logic gate $g : \{0,1\}^3 \to \{0,1\}$:

A	B	C	$g(A,B,C)$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

g can be defined by the formula

$$(\neg A \vee B \vee C) \wedge (\neg A \vee B \vee \neg C) \wedge (\neg A \vee \neg B \vee C)$$

but also by

$$A \to (B \wedge C)$$

Equivalence

Two formulas are *equivalent* (written $A \equiv B$) if they define the same logic gate.

Hence,

$$(\neg A \vee B \vee C) \wedge (\neg A \vee B \vee \neg C) \wedge (\neg A \vee \neg B \vee C)$$

and

$$A \rightarrow (B \wedge C)$$

are equivalent formulas.

Circuit minimization

Logic gates are the basic building blocks of digital circuits which in turn are the basis of computer hardware.

Circuit minimization, that is, finding the shortest representations of a logic gate is an important and difficult problem in hardware design.

All 4 unary and all 16 binary logic gates

A			NOT	
0	0	0	1	1
1	0	1	0	1

A	B		AND					XOR	OR
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1
A	B	NOR	EQU				IMP	NAND	
0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1

- How many ternary, more generally n-ary, logic gates are there?
- Can we define all logic gates by formulas?

The number of 12-ary logic gates

104438888141315250669175271071662438257996424904738378038423348328395390797155745684882681193499755834089010671443926283798757343818579360726323608785136527794595697654370999834036159013438371831442807001185594622637631883939771274567233468434458661749680790870580370407128404874011860911446797778359802900668693897688178778594690563019026094059957945343282346930302669644305902501597239986771421554169383555988529148631823791443449673408781187263949647510018904134900841706167509366833385055103297208826955076998361636941193301521379682583718809183365675122131849284636812555022599830041234478486259567449219461702380650591324561082573183538008760862210283427019769820231316901767800667519548507992163641937028537512478401490715913545998279051339961155179427110683113409058427288427979155484978295432353451706522326906139490598769300212296339568778287894844061600741294567491982305057164237715481632138063104590291613692670834285644073044789997190178146576347322385026725305989979599609079946920177462481771844986745565925017832907047311943316555080756822184657174637329688491281952031745700244092661691087414838507841192980452298185733897764810312608590300130241346718972667321649151113160292078173803343609024380470834040315419033 6

Equivalence laws

De Morgan's laws

$$\begin{aligned} \neg(A \wedge B) &\equiv \neg A \vee \neg B \\ \neg(A \vee B) &\equiv \neg A \wedge \neg B \end{aligned}$$

Distributivity

$$\begin{aligned} A \wedge (B \vee C) &\equiv (A \wedge B) \vee (A \wedge C) \\ A \vee (B \wedge C) &\equiv (A \vee B) \wedge (A \vee C) \end{aligned}$$

Material implication

$$A \to B \equiv \neg A \vee B$$

Validity and satisfiability

A formula is *logically valid* (or a *tautology*) if it is true under *all* assignments of truth values to its variables, that is, the logic gate it defines always returns the value 1.

A formula is *satisfiable* if it is true under *at least one* assignment of truth values to its variables, that is, the logic gate it defines does *not* always return the value 0.

Exercise: Which of the following formulas are valid/satisfiable?

$$A \vee \neg A, \quad A \wedge \neg A, \quad (A \to B) \vee (B \to A), \quad (A \to B) \to (B \to A)$$

Exercise

Recall:

If the Butler or the Maid is guilty, and the Maid or the Cook is guilty, then the Butler or the Cook is guilty.

$$(B \vee M) \wedge (M \vee C) \rightarrow B \vee C$$

Draw the logic gate defined by this formula and decide whether it is valid.

Satisfiability testing (SAT)

Many problems in computer science and mathematics can be encoded into the question whether a certain formula is satisfiable.

In computer science, typically, properties of possible states of a computing system (for example a computer program or a hardware component) are encoded into a CNF such that 'bad' states correspond to satisfying assignments of the CNF.

Therefore, in order to show that the system is safe one has to show that the CNF is unsatisfiable.

Testing a formula for satisfiability can be done, in principle, by trying out all 2^n assignments of truth values to the n variables of the formula. However, this is a very inefficient method.

Better methods are known, but still it is a hard problem, more precisely, the satisfiability problem is **NP**-complete.

Proofs

A *proof system* is a collection of rules to derive logically valid formulas.

There are many different proof systems. A very popular one is due to Gerhard Gentzen. It is called *Natural Deduction* since its rules are very close to natural human reasoning.

Gerhard Gentzen (1909 - 1945)

Natural Deduction

Assumption rule u:A (assumptions are cancelled by $\rightarrow^+ u : A$)		
	Introduction rules	Elimination rules
$\wedge$	$\dfrac{A \quad B}{A \wedge B}\ \wedge^+$	$\dfrac{A \wedge B}{A}\ \wedge^-_{\mathsf{l}} \qquad \dfrac{A \wedge B}{B}\ \wedge^-_{\mathsf{r}}$
$\rightarrow$	$\dfrac{B}{A \rightarrow B}\ \rightarrow^+ u : A$	$\dfrac{A \rightarrow B \quad A}{B}\ \rightarrow^-$
$\vee$	$\dfrac{A}{A \vee B}\ \vee^+_{\mathsf{l}} \qquad \dfrac{B}{A \vee B}\ \vee^+_{\mathsf{r}}$	$\dfrac{A \vee B \quad A \rightarrow C \quad B \rightarrow C}{C}\ \vee^-$
$\bot$		$\dfrac{\bot}{A}\ \mathsf{efq} \qquad \dfrac{\neg\neg A}{A}\ \mathsf{raa}$

Natural Deduction (version with explicit assumptions)

Assumption rule $\dfrac{}{\Gamma, A \vdash A}\ \text{use}$

	Introduction rules	Elimination rules
$\wedge$	$\dfrac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge^{+}$	$\dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ \wedge_{\mathsf{l}}^{-} \qquad \dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ \wedge_{\mathsf{r}}^{-}$
$\rightarrow$	$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B}\ \rightarrow^{+}$	$\dfrac{\Gamma \vdash A \rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ \rightarrow^{-}$
$\vee$	$\dfrac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ \vee_{\mathsf{l}}^{+} \qquad \dfrac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\ \vee_{\mathsf{r}}^{+}$	$\dfrac{\Gamma \vdash A \vee B \qquad \Gamma \vdash A \rightarrow C \qquad \Gamma \vdash B \rightarrow C}{\Gamma \vdash C}\ \vee^{-}$
$\bot$		$\dfrac{\Gamma \vdash \bot}{\Gamma \vdash A}\ \text{efq} \qquad \dfrac{\Gamma \vdash \neg\neg A}{\Gamma \vdash A}\ \text{raa}$

Example

$$\dfrac{\dfrac{u : A \wedge B \to C \qquad \dfrac{v : A \qquad w : B}{A \wedge B}\,\wedge^{+}}{C}\,\to^{-}}{\dfrac{\dfrac{\dfrac{C}{B \to C}\,\to^{+} w : B}{A \to (B \to C)}\,\to^{+} v : A}{(A \wedge B \to C) \to (A \to (B \to C))}\,\to^{+} u : A \wedge B \to C}$$

Example

$$\dfrac{u : A \vee B \qquad \dfrac{\dfrac{\dfrac{v : A}{B \vee A}\vee^+_{\mathsf{r}}}{A \to B \vee A}\to^+ v : A \qquad \dfrac{\dfrac{\dfrac{w : B}{B \vee A}\vee^+_{\mathsf{l}}}{B \to B \vee A}\to^+ w : B}{B \vee A}\vee^-}{A \vee B \to B \vee A}\to^+ u : A \vee B$$

Exercises

Prove the *Law of Excluded Middle*: $A \vee \neg A$.

Predicate logic

If all men are mortal and Aristotle is a man, then Aristotle is mortal.

This statement cannot be expressed in propositional logic since it is about

- properties of persons ("*Aristotle is a man*")
- and quantification ("*all men are mortal*").

To express such statements we need an extension of propositional logic called *predicate logic*.

Frege

Gottlob Frege (1848 - 1925)

Predicate logic was introduced by Frege in his *Begriffsschrift*.

The language of predicate logic

Predicate logic, also known as *first-order logic*, extends propositional logic by:

- Adding *terms*, that is, expressions denoting objects, for example 0, $\mathrm{Aristotle}$, but also *variables* x, that is, unknown objects.

- adding structure to atomic formulas: An atomic formula now states a *property* (*predicate*) of an object, for example, $\mathrm{mortal}(x)$, $\mathrm{even}(x)$, $\mathrm{mortal}(\mathrm{Aristotle})$, or a *relation* between objects, for example, $3x = y$;

- adding quantification over objects, for example,

$\forall x\,(\mathrm{man}(x) \to \mathrm{mortal}(x))$	all men are mortal
$\forall x\,\mathrm{even}(4x)$	*for all* x, $4x$ is an even number
$\exists x\,(3x = 6)$	*there exists* x such that $3x = 6$

Exercises

Write in predicate logic:

If all men are mortal and Aristotle is a man, then Aristotle is mortal.

Every positive number has a square root.

(use only the constant 0, multiplication, and the relation $<$; it is understood that variables range over numbers)

If no females are in the building and all club members are in the building and Sam is a club member, then some club member is not female.

(use the constant S (for "Sam") and the predicates F ("is female"), B ("is in the building"), C ("is a club member"))

Deciding truth in first-order logic

Computer scientists want to decide *automatically* whether a given formula (expressing a property of a program) is true.

Is this always possible?

In order to answer this question we need to understand

- what it means for a formula to be true;
- what it means to 'decide automatically'.

Hilbert

David Hilbert (1862 - 1943)

In 1926 Hilbert posed the *Entscheidungsproblem* (Decision Problem):

Find an algorithm (automatic method) to decide whether a formula in first-order logic is logically valid.

Models

In order to know what a formula means one needs to determine first:

- a universe of discourse (the objects the formula talks about)
- an interpretation of the constants and function symbols $(0, 1, 2, +, *, \ldots)$
- an interpretation of the predicate and relation symbols $(\mathrm{even}, <, \leq, \mathrm{sorted}, \ldots)$

These ingredients form a *model* $\mathcal{M}$.

In a given model $\mathcal{M}$, any formula A is either true or false.

Truth, Validity, Logical Consequence

$\mathcal{M} \models A$ (formula A is true in model $\mathcal{M}$, or $\mathcal{M}$ satisfies A)

A is *logically valid* ($\models A$) $\stackrel{\text{Def}}{=}$ for *all* models $\mathcal{M}$, $\mathcal{M} \models A$.

(A is true in *all* models)

A is a *logical consequence* of a set of formulas Γ ($\Gamma \models A$) $\stackrel{\text{Def}}{=}$ for *all* models $\mathcal{M}$, if $\mathcal{M} \models \Gamma$, then $\mathcal{M} \models A$.

(A is true in *all* models of Γ, or Γ logically implies A)

Where $\mathcal{M} \models \Gamma$ means $\mathcal{M} \models B$ for all $B \in \Gamma$.

Tarski

Alfred Tarski (1901-1983)

The precise definition of the truth of formulas in predicate logic is due to Tarski. It is therefore also called *Tarskian semantics*.

Exercises

Let $(\mathcal{N}, +)$ be the structure of natural numbers $\{0, 1, 2, \ldots\}$ with the operation of addition.

Let $A \stackrel{\text{Def}}{=} \exists z \, \forall x \forall y \, (x + y = z \rightarrow x = z)$

1. Is A true in $(\mathcal{N}, +)$?

2. Is A logically valid?

Let $A' \stackrel{\text{Def}}{=} \exists z \, \forall x \, (\exists y \, (x + y = z) \rightarrow x = z)$

3. Is the formula $A \rightarrow A'$ logically valid?
(That is, does $\{A'\} \models A$ hold?)

More exercises

Are the following formulas logically valid?

Drinker's paradox: $\exists x\,(D(x) \to \forall y\, D(x))$

(*there is a person x (in the pub) such that if x drinks, then everybody drinks*)

The formula expressing

If no females are in the building and all club members are in the building and Sam is a club member, then some club member is not female.

Computability

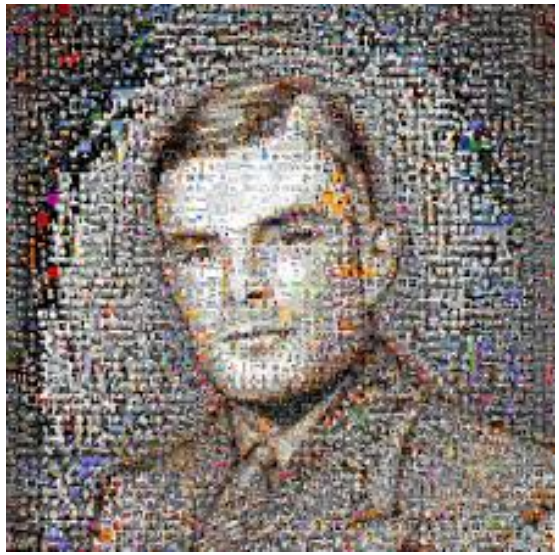
To 'decide automatically' whether a formula is true (in a fixed model, or in all models) means to have an *algorithm* to solve this problem, that is, a way of mechanically computing for every formula A the correct answer to the question whether A is true.

At first glance, this looks a like very vague and imprecise definition of computability because it is unclear what is meant by 'a way of mechanically computing'.

Luckily, the opposite is the case: Although there are many different notions of mechanical computation (using different machine models and programming languages), they all define the same class of computable problems.

A surprisingly simple way of defining computability is by way of so-called *Turing-machines*.

The Church-Turing Thesis



Alan Turing (1912 - 1954)
(picture computer generated by Arnold Beckmann)

It is commonly accepted that Turing machines provide a *universal model of computation*. This is commonly known as the

Church-Turing Thesis

The Church-Turing Thesis is supported by the fact that all other known approaches to a universal model of computation lead to the same class of computable functions.

Church

Alonzo Church (1903 - 1995)

Turing and Church independently proved the unsolvability of Hilbert's Entscheidungsproblem.

The *lambda-calculus*, invented by Church, is the theoretical basis of *Functional Programming*.

Turing was one of Church's many PhD students who became famous Logicians and Computer Scientists.

Completeness

In 1929 Kurt Gödel proved that there is a *sound and complete* proof calculus for first-order logic:

Completeness Theorem

A formula in first-order logic is logically valid if and only it is provable.

$$\models A \quad \Leftrightarrow \quad \vdash A$$

Since proofs can generated automatically, this entails that there is a computable procedure that produces, one-by-one, all logically valid formulas.

Using terminology of computability theory: The set of logically valid formulas is *computably enumerable*.

An equivalent way of stating this is that there is a program V such that for every formula A,

$V(A)$ halts if and only if A is logically valid.

Gödel

Kurt Gödel (1906-1978)

Gödel is one of the central figures in mathematical logic. He is most famous for his *Completeness Theorem* and his two *Incompleteness Theorems*.

Quantifier rules for Natural Deduction

	Introduction rules	Elimination rules
$\forall$	$\dfrac{A(x)}{\forall x\, A(x)}\ \forall^+ \quad (*)$	$\dfrac{\forall x\, A(x)}{A(t)}\ \forall^-$
$\exists$	$\dfrac{A(t)}{\exists x\, A(x)}\ \exists^+$	$\dfrac{\exists x\, A(x) \qquad \forall x\, (A(x) \to C)}{C}\ \exists^- \quad (**)$

Variable conditions:

(*) x must not occur free in any free (that is, uncancelled) assumption.

(**) x must not occur free in C.

Adding these rules to natural deduction yields a complete proof system.

Lecture 2: Proofs as programs

- Classical vs. intuitionistic logic
- The BHK interpretation of formulas
- The Lambda calculus and the Curry-Howard correspondence
- Realizability

Classical logic

Predicate logic, with Tarskian semantics and the complete proof calculus, is often called *classical logic* because it is the most traditional and widely used logic.

The following example shows that proofs in classical logic may lack an expected constructive content.

A classical proof

Theorem

There are irrational numbers a and b such that a^b is rational.

Proof. We do case analysis according to whether or not $\sqrt{2}^{\sqrt{2}}$ is rational.

Case $\sqrt{2}^{\sqrt{2}}$ is rational. Then we can take $a = b \stackrel{\text{Def}}{=} \sqrt{2}$, because, as we all know, $\sqrt{2}$ is irrational.

Case $\sqrt{2}^{\sqrt{2}}$ is irrational. Then take $a \stackrel{\text{Def}}{=} \sqrt{2}^{\sqrt{2}}$ and $b \stackrel{\text{Def}}{=} \sqrt{2}$ and we have

$$a^b = (\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = \sqrt{2}^{\sqrt{2}*\sqrt{2}} = \sqrt{2}^2 = 2$$

Although this is a nice and short proof, it is somewhat unsatisfactory, since we still do not know what a and b are.

Intuitionistic logic

The case analysis in the previous theorem made use of the *law of excluded middle*

$$A \vee \neg A$$

(with $A \stackrel{\text{Def}}{=}$ '$\sqrt{2}^{\sqrt{2}}$ is rational') which is valid in *classical logic*.

Intuitionistic logic is, roughly speaking, classical logic without the law of excluded middle.

A proof system for intuitionistic logic is obtained by removing the principle of proof by contradiction ($\neg\neg A \to A$, that is, raa) from natural deduction.

Existence Theorem for Intuitionistic logic

From an intuitionistic proof of a formula of the form $\exists x\, A(x)$ one can extract a term t such that $A(t)$ is provable.

Because of this theorem intuitionistic logic may also be called *constructive* because proofs of existential formulas include the construction of witnesses.

Semantics of Intuitionistic logic

Intuitionistic logic is incomplete w.r.t. Tarskian semantics, since the (classically valid) law of excluded middle is not provable.

However, there are other styles of semantics for which intuitionistic logic is complete and which better bring to light its constructive nature.

We study an interpretation due to Brouwer, Heyting, and Kolmogorov.

L E Jan Brouwer
(1881 - 1966)

A Heyting
(1898 - 1980)

A Kolmogorov
(1903 - 1987)

The BHK interpretation

According to the BHK interpretation a formula expresses a *computational problem* which is defined by a description of how to solve it:

A solution to $A \wedge B$ is a pair (a, b) such that

a solves A and b solves B.

A solution to $A \vee B$ is

either $(0, a)$ where a solves A

or $(1, b)$ where b solves B.

A solution to $A \to B$ is a construction that transforms

any solution of A to a solution of B.

The lambda calculus

In the BHK interpretation it is left open what a "construction" is.

Church's *lambda calculus* provides a good notion of construction:

The lambda calculus consists of

- *lambda terms* generated by the rules

x	Variables
$\lambda x\,.\,M$	lambda-abstraction
$M\,N$	Application

- *beta-reduction*

$$(\lambda x\,.\,M)N \quad \to_\beta \quad M[N/x]$$

$M[N/x]$ denotes substitution of the term N for x in the term M.

One usually writes $M\,N\,K$ for $(M\,N)\,K$.

Lambda calculus with types

Types are like propositional formulas with $\to$ as the only connective.

Let $\Gamma = x_1 : A_1, \ldots x_n, : A_n$ be a *context*, that is a type assignment to variables.

We define inductively the relation $\Gamma \vdash M : A$
(M has type A in context Γ).

$$\Gamma, x : A \vdash x : A$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x\, M : A \to B} \qquad \frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\, N : B}$$

β-reduction and -equality

Theorem

β-reduction is *strongly normalizing*, that is, every reduction sequence $M_1 \to_\beta M_2 \to_\beta M_3, \ldots$ terminates.

Theorem

β-reduction is *confluent*, that is, if $M \to_\beta^* N_1$ and $M \to_\beta^* N_2$, then there exists a term N such that $N_1 \to_\beta^* N$ and $N_2 \to_\beta^* N$.

Theorem

The relation of β-equality, defined by

$$M =_\beta N \quad :\Leftrightarrow \quad \exists K \, (M \to_\beta^* K \wedge N \to_\beta^* K)$$

is decidable.

Extension to products and sums

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash N : B}{\Gamma \vdash (M, N) : A \times B}$$

$$\frac{\Gamma \vdash M : A \times B}{\Gamma \vdash \pi_0(M) : A} \qquad \frac{\Gamma \vdash M : A \times B}{\Gamma \vdash \pi_1(M) : B}$$

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash (0, M) : A + B} \qquad \frac{\Gamma \vdash M : B}{\Gamma \vdash (1, M) : A + B}$$

$$\frac{\Gamma \vdash M : A + B \qquad \Gamma \vdash N : A \to C \qquad \Gamma \vdash K : B \to C}{\Gamma \vdash \mathbf{case}(M, N, K)}$$

$$\begin{aligned}
\pi_0(M, N) &\to_\beta M \\
\pi_1(M, N) &\to_\beta N \\
\mathbf{case}((0, M), N, K) &\to_\beta N\, M \\
\mathbf{case}((1, M), N, K) &\to_\beta K\, M
\end{aligned}$$

The Curry-Howard correspondence

The *Curry-Howard correspondence* is the observation that intuitionistic natural deduction proofs are in a natural correspondence with the typed lambda calculus or the *typed combinator calculus*.

Since typed lambda terms are the core of functional programming languages such as ML and Haskell (named after Haskell B Curry) one can also say that intuitionistic proofs correspond to programs.

Haskell B (1900-1982)

Intuitionistic ND proofs vs typed lambda calculus

$$\frac{A \qquad B}{A \wedge B} \qquad \frac{M : A \qquad N : B}{(M, N) : A \times B}$$

$$\frac{A \wedge B}{A} \qquad \frac{A \wedge B}{B} \qquad \frac{M : A \times B}{\pi_0(M) : A} \qquad \frac{M : A \times B}{\pi_1(M) : B}$$

$$\frac{\begin{array}{c}[u : A] \\ B\end{array}}{A \to B} \to^+ u : A \qquad \frac{M : B}{\lambda x\, M : A \to B}$$

$$\frac{A \to B \qquad A}{B} \qquad \frac{M : A \to B \qquad N : B}{M\, N : B}$$

$$\frac{A}{A \vee B} \qquad \frac{B}{A \vee B} \qquad \frac{M : A}{(0, M) : A \vee B} \qquad \frac{M : B}{(1, M) : A \vee B}$$

$$\frac{A \vee B \qquad \begin{array}{c}[A] \\ C\end{array} \qquad \begin{array}{c}[B] \\ C\end{array}}{C} \qquad \frac{M : A \vee B \qquad N : A \to C \qquad K : B \to C}{\mathbf{case}(M, N, K) : C}$$

Realizability

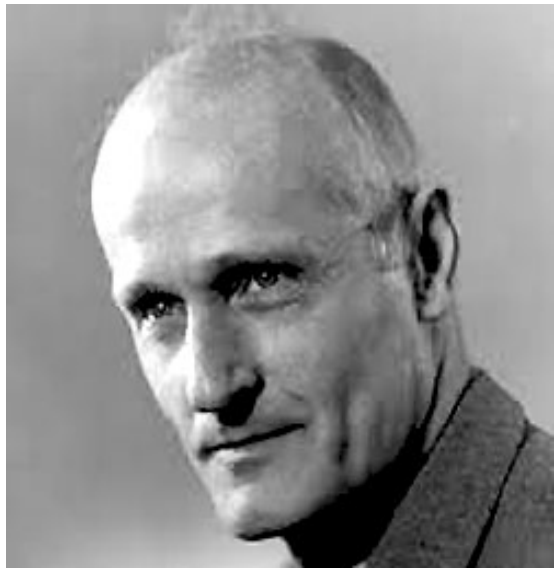
Realizability attaches meaning to the Curry-Howard correspondence.

Intuitively:

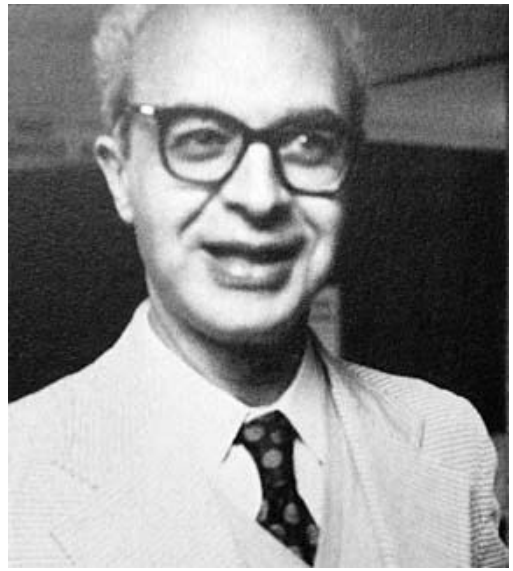
If $M : A$ (that is, M codes an intuitionistic ND proof of A), then M solves the problem A according to the BHK-interpretation.

This intuition is formalized in Kleene's and Kreisel's realizability interpretation of intuitionistic logic.

Kleene and Kreisel



Stephen Kleene
(1909 - 1994)



Georg Kreisel
(1923 - 2015)

Definition of (formalized modified) realizability

To every formula A one assigns a unary predicate $\mathbf{R}(A)$ in an extended language with a new sort of (potential) realizers for each formula (type) A.

We write $a\,\mathbf{r}\,A$ instead of $\mathbf{R}(A)(a)$.

$$
\begin{array}{lcll}
a\,\mathbf{r}\,A & \equiv & A \wedge a = \mathbf{Nil} & (A \text{ atomic}) \\
c\,\mathbf{r}\,(A \wedge B) & \equiv & c = (a, b) \wedge a\,\mathbf{r}\,A \wedge b\,\mathbf{r}\,B & \\
f\,\mathbf{r}\,(A \to B) & \equiv & \forall a\,(a\,\mathbf{r}\,A \to (f\ a)\,\mathbf{r}\,B) & \\
c\,\mathbf{r}\,(A \vee B) & \equiv & (c = (0, a) \wedge a\,\mathbf{r}\,A) \vee (c = (1, b) \wedge b\,\mathbf{r}\,B) &
\end{array}
$$

Soundness of realizability

If $M : A$ (that is, M codes an intuitionistic ND proof of A), then $M \,\mathbf{r}\, A$, that is, M realizes A.

The proof of A encoded by M can be automatically transformed into an intuitionistic ND proof of $M \,\mathbf{r}\, A$.

Realizability for predicate logic

The definition of realizability can be extended to predicate logic in a straightforward way:

$$\begin{aligned} a\,\mathbf{r}\,\forall x\,A(x) &\equiv \forall x\,a\,\mathbf{r}\,A(x) \\ a\,\mathbf{r}\,\exists x\,A(x) &\equiv \exists x\,a\,\mathbf{r}\,A(x) \end{aligned}$$

If p is an intuitionistic ND proof of A where A is a formula in *predicate* logic, then the lambda term M is no longer identical to p but is obtained by deleting all first-order parts (and more) from p.

In that case, M is called the program extracted from M and the Soundness Theorem reads:

If p is an intuitionistic ND proof of A, then one can extract from p a lambda term M such that $M\,\mathbf{r}\,A$.

Example

$$\dfrac{u : A \vee B \qquad \dfrac{\dfrac{\dfrac{v : A}{B \vee A}\vee_{\mathsf{r}}^{+}}{A \to B \vee A}\to^{+} v : A \qquad \dfrac{\dfrac{\dfrac{w : B}{B \vee A}\vee_{\mathsf{l}}^{+}}{B \to B \vee A}\to^{+} w : B}{B \vee A}\vee^{-}}{A \vee B \to B \vee A}\to^{+} u : A \vee B$$

What's the extracted program and what does it do?

Lecture 3: The magic of induction

- Induction on natural numbers
- Strictly positive induction
- Realizability of induction
- Coinduction

Induction on the natural numbers

$$\frac{P(0) \qquad \forall x\,(P(x) \to P(x+1))}{\forall x \in \mathbb{N}\, P(x)}$$

Realizability:

$$\frac{a\,\mathbf{r}\,P(0) \qquad f\,\mathbf{r}\,(\forall x\,(P(x) \to P(x+1)))}{g\,\mathbf{r}\,(\forall x \in \mathbb{N}\, P(x))}$$

where $g = \mathbf{It}(a, n)$ with

$g : \mathbb{N} \to \tau(P) \quad (\tau(P) = \text{type of realizers of } P)$

$g(n) = f^n(a)$

$\mathbb{N}$ is a new predicate symbol and $\mathbf{It}(a, n)$ is a new program construct.

"$n\,\mathbf{r}\,\mathbb{N}(x)$" is defined as "$n$ is the unary representation of $x \in \mathbb{N}$".

Other forms of induction

Induction on lists, trees, ...

$$\frac{P([]) \qquad \forall x \in A \, \forall l \, (P(l) \to P(x : l))}{\forall x \in \mathbf{List}(A) \, P(x)}$$

Induction on ordinals (or any wellfounded relation $<$)

$$\frac{\forall x \, ((\forall y < x \, P(y)) \to P(x))}{\forall x < \alpha \, P(x)}$$

Bar induction

...

A unifying approach: Monotone induction

Let U be a set and $\mathcal{P}(U)$ the powerset of U.

An operator $\Phi : \mathcal{P}(X) \to \mathcal{P}(X)$ is *monotone* if for all $X, Y \in \mathcal{P}(U)$

$$X \subseteq Y \to \Phi(X) \subseteq \Phi(Y)$$

Every monotone operator $\Phi : \mathcal{P}(X) \to \mathcal{P}(X)$ has a *least fixed point*, $\mu(\Phi) \in \mathcal{P}(U)$, which can be defined by

$$\mu(\Phi) \stackrel{\text{Def}}{=} \bigcap \{X \in \mathcal{P}(U) \mid \Phi(X) \subseteq X\}$$

but also by

$$\mu(\Phi) \stackrel{\text{Def}}{=} \bigcup \{\Phi^{\alpha}(\emptyset) \mid \alpha \in \textbf{Ordinals}\}$$

Closure and induction

One can show (exercise) that indeed $\mu(\Phi)$ is a fixed point of Φ, that is,

$$\Phi(\mu(\Phi)) = \mu(\Phi)$$

Moreover, $\mu(\Phi)$ is the least element of

$$\mathbf{pfp}(\Phi) \stackrel{\text{Def}}{=} \{X \in \mathcal{P}(U) \mid \Phi(X) \subseteq X\}$$

which means that the following rules hold:

$$\frac{}{\Phi(\mu(\Phi)) \subseteq \mu(\Phi)}\ \textbf{Cl} \qquad \frac{\Phi(X) \subseteq X}{\mu(\Phi) \subseteq X}\ \textbf{Ind}$$

Formalizing induction

We add to the language of predicate logic *predicate variables* $X, Y, \ldots$.

Atomic formulas can now also be of the form $X(\vec{t})$.

If $A(X, \vec{x})$ is a formula where $\vec{x}$ is a tuple of individual variables and the predicate variable X occurs only *strictly positively*, that is, not on the left hand side of an implications, then we can form the predicate $\lambda\vec{x}\, A(X, \vec{x})$ and the monotone operator

$$\Phi \stackrel{\text{Def}}{=} \lambda X \lambda\vec{x}\, A(X, \vec{x})$$

and formally introduce the predicate $\mu(\Phi)$ with the intention that it denotes the least fixed point of Φ.

We express this intention by adding the derivation rules

$$\frac{}{\Phi(\mu(\Phi)) \subseteq \mu(\Phi)}\ \textbf{CI} \qquad \frac{\Phi(P) \subseteq P}{\mu(\Phi) \subseteq P}\ \textbf{Ind}$$

For every predicate P definable in the language.

Proof theory of inductive definitions

Inductive definitions add considerable strength to a theory.

The precise proof-theoretic strength of various fragments of the theory of inductive definitions (obtained by allowing operators Φ of varying complexity) is completely analysed in the book

```
@Book{BuFePoSi81,
  author = "Buchholz, W. and Feferman, F. and Pohlers, W. and Sieg, W."
  title =  "Iterated Inductive Definitions and Subsystems of
                      Analysis: Recent Proof--Theoretical Studies",
  publisher = "Springer",
  year =    "1981",
  volume =  "897",
  series =  "Lecture Notes in Mathematics",
  address = "Berlin"}
```

Example: Natural numbers

Consider a first-order axiomatisation of real closed fields (with the real numbers as intended model).

In particular we have the constants, functions and relation symbols $0, 1, +, *, |\cdot|, <, \ldots$ with the expected axioms.

We define the natural numbers as the least predicate that contains 0 and is closed under the $(+1)$ operation:

$$\mathbb{N} \stackrel{\text{Def}}{=} \mu(\Phi)$$

where $\Phi = \lambda X\, \lambda x\, (x = 0 \vee \exists y(X(y) \wedge x = y + 1))$

Equivalently, $\Phi = \lambda X\, \lambda x\, (x = 0 \vee X(x - 1))$.

We use the following short notation for the definition of $\mathbb{N}$:

$$\mathbb{N}(x) \stackrel{\mu}{=} x = 0 \vee \mathbb{N}(x - 1)$$

Induction on $\mathbb{N}$

Recall that $\mathbb{N} = \mu(\Phi)$ where $\Phi = \lambda X\, \lambda x\, (x = 0 \vee X(x-1))$.

Let's work out the meaning of the rules

$$\frac{}{\Phi(\mathbb{N}) \subseteq \mathbb{N}}\ \textbf{CI} \qquad \frac{\Phi(P) \subseteq P}{\mathbb{N} \subseteq P}\ \textbf{Ind}$$

$\Phi(\mathbb{N}) \subseteq \mathbb{N}$ means $\forall x\, (\mathbb{N}(x) \to x = 0 \vee \mathbb{N}(x-1))$ which is equivalent to

$$\mathbb{N}(0) \wedge \forall x\, (\mathbb{N}(x) \to \mathbb{N}(x+1))$$

$\Phi(P) \subseteq P$ means $\forall x\, (x = 0 \vee P(x-1) \to P(x)$ which is equivalent to

$$P(0) \wedge \forall x\, (P(x) \to P(x+1))$$

Therefore, the induction rule is nothing but the familiar

$$\frac{P(0) \qquad \forall x\, (P(x) \to P(x+1))}{\forall x \in \mathbb{N}\, P(x)}\ \textbf{Ind}$$

Realizability of inductive definitions

$$\mathbf{R}(\mu(\Phi)) \stackrel{\text{Def}}{=} \mu(\mathbf{R}(\Phi))$$

where for $\Phi = \lambda X \lambda \vec{x}\, A(X, \vec{x})$ we introduce a new predicate variable $\tilde{X}$ with one argument place for realizers and set

$$\begin{aligned} a\,\mathbf{r}\,X(\vec{t}) &\stackrel{\text{Def}}{=} \tilde{X}(\vec{t}, a) \\ \mathbf{R}(\Phi) &\stackrel{\text{Def}}{=} \lambda \tilde{X} \lambda(\vec{x}, a)\, a\,\mathbf{r}\, A(X, \vec{x}) \end{aligned}$$

For $\mathbb{N}$ this works out as

$$a\,\mathbf{r}\,\mathbb{N}(x) \stackrel{\mu}{=} (a = (0, \mathbf{Nil}) \wedge x = 0) \vee (a = (1, b) \wedge b\,\mathbf{r}\,\mathbb{N}(x - 1))$$

If we define numerals as $\underline{0} = (0, \mathit{Nil})$, $\underline{n+1} = (1, \underline{n})$, we see that $a\,\mathbf{r}\,x$ iff $x \in \mathbb{N}$ and a is the numeral representing x.

Realizability of closure and induction

$\Phi = \lambda X \lambda \vec{x}\, A(X, \vec{x})$

It is easy to see that $(\lambda a\, a)\, \mathbf{r}\, (\Phi(\mu(\Phi)) \subseteq \mu(\Phi))$.

$$\frac{s\, \mathbf{r}\, (\Phi(P) \subseteq P)}{g\, \mathbf{r}\, (\mu(\Phi) \subseteq P)}$$

where g is defined recursively as

$$g = s \circ (\mathbf{map}\, g) \qquad (\circ \text{ means composition})$$

and **map** is a term such that $\mathbf{map}\, \mathbf{r}\, ((X \subseteq Y) \to \Phi(X) \subseteq \Phi(Y))$.

map can be constructed by induction on A using the fact that X may only occur strictly positive in A.

Realizability of induction on natural numbers

$g = s \circ (\mathbf{map}\, g)$, that is, $g\, a = s\, (\mathbf{map}\, g\, a)$

$\Phi = \lambda X \lambda x\, (x = 0 \vee X(x-1))$

$\mathbf{map}\, s\, (0, \mathbf{Nil}) = (0, \mathbf{Nil})$, $\mathbf{map}\, s\, (1, a) = (1, s\, a)$.

$a\, \mathbf{r}\, P(0)$, $f\, \mathbf{r}\, (\forall x\, (P(x) \to P(x+1))$.

$s\, (0, \mathbf{Nil}) = a$, $s\, (1, a) = f\, a$.

Therefore,

$$\begin{aligned} g\, (0, \mathbf{Nil}) &= s\, (0, \mathbf{Nil}) = a \\ g\, (1, a) &= s\, (1, g\, a) = f\, (g\, a) \end{aligned}$$

That is, $g\, \underline{n} = f^n a$.

Coinduction

Instead of *least* fixed points, $\mu(\Phi)$, one can also consider *greatest* fixed points, $\nu(\Phi)$.

$$\frac{}{\nu(\Phi) \to \Phi(\nu(\Phi))}\ \textbf{Cocl} \qquad \frac{P \subseteq \Phi(P)}{P \subseteq \mu(\Phi)}\ \textbf{Coind}$$

$$\frac{s\,\mathbf{r}\,(P \subseteq \Phi(P))}{g\,\mathbf{r}\,(P \subseteq \mu(\Phi))}$$

where g is defined recursively as

$$g = (\mathbf{map}\, g) \circ s$$

Example: Cauchy reals

Let $\mathbb{Q}(x)$ mean that x is a rational number (definable using $\mathbb{N}$).

$$\mathbf{A}(x) \stackrel{\nu}{=} \exists q \in \mathbb{Q}\, |x - q| \leq 1 \wedge \mathbf{A}(2x)$$

Then $a \,\mathbf{r}\, \mathbf{A}(x)$ iff a is an infinite stream of rational numbers,

$$a = (q_0, (q_1, (q_2 \ldots$$

such that $|x - q_n| \leq 2^n$ for all n.

Program extraction

- Natural numbers: Quotient and remainder
- Real numbers: Integration
- Lambda calculus: Extraction of normalization-by-evaluation
- Satisfiability: Extraction of a SAT-solver
- Extensions: Extraction of concurrent programs