Advanced course in foundations of mathematics

Part 1

Dr Roberta Bonacina

roberta.bonacina@fsci.uni-tuebingen.de

University of Tübingen
Carl Friedrich von Weizsäcker Center

a.a. 2020/21

The course

The main topics of the second half of "Advanced course in foundations of mathematics" are

- history of Mathematics
- an introduction to first order logic
- computable functions
- limiting results
  - Peano arithmetic
  - Gödel's incompleteness theorems
  - natural incompleteness results
- constructive Mathematics
  - λ calculus
  - type theories
  - propositions-as-types interpretation

Teaching

I will mainly teach sharing those slides, which are available via Moodle and at the website

https://robertabonacina.com/advanced-course-in-foundations-of-mathematics

Questions and interactions in general are welcome!

Feel free to interrupt me at any time during the lectures, or to write in the chat.

You can also contact me by email at roberta.bonacina@fsci.uni-tuebingen.de or roberta.bonacina@univr.it.

Syllabus

History of Mathematics and its foundations:

- developments of Mathematics
- foundational crisis
- constructive mathematics
- recent results

History of Mathematics

The history of Mathematics is the background to justify why and how Mathematical logic, and in particular the research of a foundation of Mathematics, developed.

The fundamental idea of *theorem*, a statement wich holds because its truth is achieved by correct reasoning, has been introduced by the Pythagorean school.

Important to mention are: Heraclitus, who introduced the term *logos* to mean reasoning, and Zeno of Elea, who introduced *reductio ad absurdum*, that is, proof by contradiction, and analysed the idea of infinity in its famous paradoxes.

A turning point was Aristotele's Organon, in which he introduced the theory of *syllogism*, the first formal system. His work emphasises the formal nature of reasoning, and he was the first to deal with the principles of contradiction and excluded middle in a systematic way.

Modality and abstraction

The stoic school, mainly Chrysippus, introduced the notion of *modality*, the theory of conditionals, which led to a systematic understanding of implication as a connective, and the relation between meaning and truth, posing the basis for semantics.

Centuries later, after the fall of the Roman Empire, and when Islam saw its golden age, the works of Al-Farabi, Ibn Sina (Avicenna) and others introduced further developments of the Greek tradition, in particular identifying the basis of modal and temporal logics. In particular Avicenna's "ma'na", the notion of sign in the mind that does not necessarily represent an existing thing, is the first attempt to capture the nature of abstract or ideal objects.

Although interesting, we skip over Medieval logic as its influence in Mathematical logic is minor, but it was fundamental to frame the basic of scientific reasoning.

Descartes

The Greek geometric tradition and the algebraic tradition, mainly developed in the Islamic world and in the early Italian school, merged together in the outstanding work of Descartes, with his analytic geometry. The importance of this contribution is difficult to overestimate: the notions of *curve* and *figure* become first class objects, which could be described by equations, no longer limiting the domain of geometry to lines, circles, and conics.

Also, the notion of *space* is a natural consequence of Descartes' work: it is the set of points which can be described by *coordinates*, tuples of numbers of fixed length. It was possible to conceive multi-dimensional spaces and to smoothly apply the algebraic methods to them.

Newton and Leibniz

Then Newton and Leibniz invented mathematical analysis. For Newton, this was the language in which to express his theory of gravitation, in fact, the milestone that marks the born of Science. In fact, Newton developed the *calculus of fluxions* by basing it on Euclidean geometry, trying to make it well founded, justified in the Euclid's sense. Oppositely Leibniz wanted an agile system: today, we still use his notation for integrals and derivatives. This more intuitive approach, based on infinitesimals, was extremely influencing, and, in fact, dominated the development of Mathematics till the 19^{th} Century.

Leibniz also conceived the idea of *characteristica universalis*, an attempt to devise a formal theory of thought, allowing to mechanically calculate the act of reasoning. This ideas anticipate symbolic logic with striking insights, like using the prime factorisation theorem to code propositions, anticipating Gödel's numbering.

Analysis

At the beginning of the 19^{th} Century, analysis was dominating mathematics, and it has become a very abstract field, a character that become predominant in Mathematics. This race for abstraction revealed a number of deep problems: functions in their full generality were defeating intuition, causing doubts on proofs. In fact, most of stunning achievements of analysis simply failed for general functions, showing that limits, continuity, derivation and integration are subtle concepts which require precise definitions. In turn, these definitions failed because mathematicians realised that the very notion of real number was inadequately understood.

This crisis eventually led to modern mathematical analysis: Bolzano's, Dedekind's, Cauchy's constructions of real numbers out of rationals; the notion of limit, continuity, and the definition of derivatives and integrals in formal terms. This immense effort due to Gauss, Cauchy, Weierstrass, Riemann, Dedekind an many others cleared the foundations of analysis but also showed that many fundamental mathematical concepts, taken for granted, required further study.

Algebra and geometry

In the first half of the 19^{th} Century a revolution started in the mathematical world: the advent of abstract algebra, and the discovery of non-Euclidean geometries.

In algebra, Abel and Galois developed group theory, providing a deep insight on the solution of polynomial equations. In fact, they were the first ones to provide *limiting results*: they showed that no general algebraic method can exist for solving polynomial equations of degree greater than four.

In geometry, Gauss, Bolyai, and Lobachevsky developed alternative models of spaces in which the parallel postulate does not hold. Then, Riemann vastly generalised the idea showing how every space admits a geometry which best describes it, whose lines are the *geodetic*, the curves of minimal length between a pair of points.

Klein, later, has shown the bridge between geometry and algebra: a geometry is, in fact, a group of transformations acting on a space.

Abstraction

The sudden raise in abstraction, the availability of novel and powerful instruments in algebra, and the pressure from analysis led to a deeper study of the fundamental ingredients of mathematical thinking: spaces, numbers, sets.

The need for a general notion of space going beyond the Euclidean intuition eventually led to topology. The study of numbers led to abstract away what was not needed in their definition, to keep their essential properties in a domain: this process led to abstract algebra, the notions of groups, rings, fields, vector spaces. The study of the notion of set by Cantor is particularly relevant to us. By comparing sets through functions, Cantor discovered that the idea of infinite is not unique: he showed that real numbers are more numerous than natural numbers, leading to the notion of *cardinality*.

In algebra, we have to mention the work of Boole, who invented an algebraic system to represent logical propositions in the sense of Aristotle. This is the starting point, together with Cantor's work, of a new discipline, Mathematical Logic.

Frege and Russell

Gottlob Frege in his Begriffsschrift (1879) introduced variables, quantifiers, and a rigorous treatment of functions, basing on intuitive set theory.

His purpose was to show that arithmetic is a branch of logic, and no intuition is needed to understand it, an approach which takes the name of Logician in philosophy. Also, his work can be considered the first attempt to provide a rigorous foundation to the whole Mathematics.

A principle used by Frege is the axiom of unlimited comprehension: if P is a formula depending on just x, then $R := \{x : P(x)\}$ is a set. In 1903, Bertrand Russell wrote a letter to Frege in which his famous paradox shows how Frege's system is inconsistent, as it contains a contradiction.

Consider $R := \{x | x \notin R\}$. Then $R \in R$ if and only if $R \notin R$. This contradiction cannot be avoided but dropping that axiom.

Hilbert

In 1889, Giuseppe Peano defined a formal theory for arithmetic which bears his name. He and Dedekind recognised induction as the characterising principle of natural numbers.

In 1899, David Hilbert developed a complete set of axioms for Euclidean geometry (Grundlagen der Geometrie), which makes formal Euclid's Elements.

In 1900, at the International Conference of Mathematicians, David Hilbert posed a list of 23 problems. For example, two of them required to resolve the Continuum Hypothesis and to find method to decide whether a multivariate polynomial equation over the integer has a solution.

The solution to these problems and their consequences are some of the fundamental results in Mathematical logic and foundations of Mathematics.

Hilbert

Hilbert's program aimed at developing a theory of Mathematics to provide a solid, definitive foundations. The pillar of his programs were

- formalisation: all mathematical statements have to be written, at least in principle, in a precise formal language and manipulated according to a fixed, precise and formal set of rules;
- consistency: the whole corpus of Mathematics have to be proved to be contradiction free by means of a formal proof inside Mathematics itself;
- finitistic: the language, the rules of inference, and the proofs have to be finite and effective (i.e., possible to calculate mechanically). In particular, the consistency proof has to be finitistic.

Sets, types, limiting results

Ernst Zermelo in 1908 introduced a formal system for set theory, in which the Axiom of Choice is stated for the first time. Later Abraham Fraenkel added the Axiom of Replacement obtaining **ZFC**, which is usually regarded as the reference for formal set theory.

In 1910, the first volume of Principia Mathematica by Russell and Whitehead appeared. This monumental work is an attempt to reconstruct the whole Mathematics from a single formal system, avoiding internal contradictions. It is based on a peculiar form of type theory.

Leopold Löwenheim (1915) and Thoralf Skolem (1920) obtained limiting results, summarised in their Theorem, saying that the first order theories cannot influence the cardinality of their infinite models. Skolem went so far to realise that formal set theory must have a countable model, which is completely counterintuitive.

Gödel

In 1929, Kurt Gödel proved the Completeness Theorem for first order logic in his doctoral dissertation. As a consequence, he derived compactness, which proves the finitary nature of first order systems.

In 1931, *On formally undecidable propositions of Principia Mathematica and Related Systems* was published. This is a milestone in human knowledge. In that work, Gödel proved that every sufficiently strong yet effective system is either inconsistent or contains a true but unprovable statement. He also showed that the consistency of such a formal system is one of those unprovable statements, giving a negative answer to Hilbert's program and closing the efforts of Russell and Whitehead's quest for a universal system.

In 1936, Gerard Gentzen proved the consistency of Peano arithmetic using methods which could not be formalised in Peano arithmetic itself; he proved it in a finitistic system together with the principle of transfinite induction up to the ϵ_0 ordinal, which can be considered a measure of the proving power of arithmetic. This result introduced also cut-elimination. In fact, this is the starting point of Proof Theory.

Constructive Mathematics

In the '20s, the notion of computability was in the air. Early works, like the definition of function by Schönfinkel, eventually led Gödel, Turing, Kleene and Church to recognise that effective systems and computable functions are two faces of the same coin. In particular, the notion of computable function proved to be very robust, leading to the Church-Turing thesis, and eventually generated an entirely new discipline: Computer Science.

It is worth noting how the solution of the Halting Problem (i.e., that determining whether a program will terminate or not is undecidable), posed by Hilbert in 1928, is strictly related to Gödel's incompleteness result and the existence of non-computable functions.

An important turnpoint in logic was Intuitionism, a philosophical line of thought promoted by Brouwer and formalised by Heyting. This line emphasises *constructions* as the building blocks of Mathematics. In time, it turned out that intuitionistic logic is strictly related with computability and plays a fundamental role in the foundation of Mathematics, for example, in topos theory.

Recent developments

At this point in history, around 1940, it becomes difficult to keep track of developments and achievements. We limit ourselves to mention

- the impact of category theory, which ultimately leads to abandon sets in favour of a more abstract but also more regular algebraic structure;
- the impact of type theories as the computational counterparts of constructive formal systems, which have recently been shown to have strict links with algebraic topology in Homotopy type theory;
- the development of ordinal analysis to classify theories and problems according to the proving power, essentially extending and generalising the limiting results;
- the immense amount of logics which have been studied: modal, temporal, epistemic, deontic, paraconsistent, many-valued, fuzzy just to name a few with their semantics, proof theory, and an ever growing amount of applications in every field.

References

All those lectures are mainly based on Marco Benini's slides for his course Mathematical Logic at the Insubria University. The original slides, together with some more material, can be found in the course website https://marcobenini.me/lectures/mathematical-logic/.

About the history of Mathematics, I suggest *Morris Kline*, Mathematical Thought from Ancient to Modern Times, Oxford University Press (1972). *Michael J. Beeson*, Foundations of Constructive Mathematics, Springer (1985) focuses on the history of constructive mathematics.
Finally, on Stanford Encyclopedia of Philosophy https://plato.stanford.edu/ can be found all the cited authors and topics.

Advanced course in foundations of mathematics

Part 2

Dr Roberta Bonacina

roberta.bonacina@fsci.uni-tuebingen.de

University of Tübingen
Carl Friedrich von Weizsäcker Center

a.a. 2020/21

Syllabus

First order logic:

- Language
- Substitution
- Natural deduction
- Informal meaning
- Semantics
- Soundness, completeness and compactness

Language

Definition 2.1 (Signature)

A *signature* $\Sigma = \langle S; F; R \rangle$ is composed by

- a set S of symbols for *sorts*.
- a set F of symbols for *functions*. Each symbol $f \in F$ is uniquely associated with a *type* $s_1 \times \cdots \times s_n \to s_0$, with $s_i \in S$ for each $0 \leq i \leq n$. When $n = 0$, we say that f is a *constant* of type s_0.
- a set R of symbols for *relations*. Each symbol $r \in R$ is uniquely associated with a *type* $s_1 \times \cdots \times s_n$, with $s_i \in S$ for each $1 \leq i \leq n$.

The notation $f : s_1 \times \cdots \times s_n \to s_0 \in F$ and $r : s_1 \times \cdots \times s_n \in R$ means that f is a function symbol whose type is $s_1 \times \cdots \times s_n \to s_0$, and r is a relation symbol whose type is $s_1 \times \cdots \times s_n$, respectively. Also, we require that S, F, and R do not contain the logical connectives and quantifiers.

A signature describes a first-order language: sorts stand for collections of elements, functions are used to denote elements, while relations are used to form basic formulae.

Example 2.2

The signature

$$\mathcal{N} = \langle \{\mathbb{N}\}; \{0 : \mathbb{N}, S : \mathbb{N} \to \mathbb{N}; + : \mathbb{N} \times \mathbb{N} \to \mathbb{N}, \cdot : \mathbb{N} \times \mathbb{N} \to \mathbb{N}\}; \{= : \mathbb{N} \times \mathbb{N}\} \rangle$$

specifies the basic language for arithmetic. There is one sort, which, in the intended interpretation, stands for the collection of natural numbers. There is a constant, 0, denoting the zero natural number, there is a function S, which stands for 'successor', denoting the next natural number, so that, in the intended interpretation, $S(5) = 6$, while the functions $+$ and $\cdot$ denote addition and multiplication.

There is only one relation symbol, denoting equality.

Of course, the theory of arithmetic should be devised in such a way that, as far as possible, the formal behaviour, that is, what we can prove, conforms to the intended interpretation.

Terms

The first-order language has two purposes: to provide a syntax to denote elements in the universe, i.e., in the collections denoted by the sorts, and to provide a syntax to denote properties of those elements.

The first issue is addressed by *terms*.

Definition 2.3 (Term)

Let $\Sigma = \langle S; F; R \rangle$ be a signature, and let V be an infinite set of symbols, called *variables*, such that $V \cap (S \cup F \cup R) = \emptyset$. Also, assume that each variable $x \in V$ has a uniquely associated type $s \in S$, denoted by $x : s$. We require that there is an infinite amount of variables for each type $s \in S$. A *term*, along with the set of its *free variables*, is inductively defined as:

- if $x : s \in V$, then x is a term of type s, and $\mathrm{FV}(x) = \{x\}$;
- if $f : s_1 \times \cdots \times s_n \to s_0 \in F$ and $t_1, \ldots, t_n$ are terms of type $s_1, \ldots, s_n$, respectively, then $f(t_1, \ldots, t_n)$ is a term of type s_0, and $\mathrm{FV}(f(t_1, \ldots, t_n)) = \bigcup_{i=1}^{n} \mathrm{FV}(t_i)$.

We use the notation $t : s$ to say that the term t has type s.

Terms

Example 2.4

Using the signature $\mathscr{N}$ of arithmetic, 0, $S(0)$, $S(S(0))$, ... are terms of type $\mathbb{N}$. Also $+(x,0)$ and $\cdot(x,+(S(0),S(S(0))))$ are terms of type $\mathbb{N}$. Notice how $x+0$ and $x(1+2)$ are **not** terms.

Formulae

As terms are used to denote elements, formulae are used to denote properties of elements.

Definition 2.5 (Formula)

Fixed a signature $\Sigma = \langle S;\ F; R\rangle$ and a set of variables as for terms, a *formula*, along with the set of its *free variables*, is inductively defined as

- $\top$ and $\bot$ are formulae, and $\mathrm{FV}(\top) = \mathrm{FV}(\bot) = \emptyset$.
- if $r : s_1 \times \cdots \times s_n \in R$ is a relation symbol, and $t_1 : s_1, \ldots, t_n : s_n$ are terms, then $r(t_1, \ldots, t_n)$ is an *atomic* formula, and $\mathrm{FV}\left(r(t_1, \ldots, t_n)\right) = \bigcup_{i=1}^{n} \mathrm{FV}(t_i)$.
- if A and B are formulae, so are $\neg A$, $A \wedge B$, $A \vee B$, and $A \supset B$, and $\mathrm{FV}(\neg A) = \mathrm{FV}(A)$, $\mathrm{FV}(A \wedge B) = \mathrm{FV}(A \vee B) = \mathrm{FV}(A \supset B) = \mathrm{FV}(A) \cup \mathrm{FV}(B)$.
- if $x : s$ is a variable and A is a formula, so are $\forall x : s. A$ and $\exists x : s. A$, and $\mathrm{FV}(\forall x : s. A) = \mathrm{FV}(\exists x : s. A) = \mathrm{FV}(A) \setminus \{x\}$.

Formulae

Notice that in quantified formulae the variable is **not** free. We say that quantified variables are *bounded*.

The notion of bounded variable is not new: for example, the expression $\int_a^b f(x)\,\mathrm{d}x$ does not really depend on the variable x. In fact, the x is a placeholder, to give some name to the argument of the f function. A bounded variable does not denote a value, but rather it acts as a placeholder which allows to write a formula or a term. Its meaning is controlled by the quantifier, and not by the way variables are interpreted, as in the integral, the x does not denote a real or complex number, but rather what is allowed to vary in the function.

Substitution

Variables are subject to a fundamental operation: substitution. In fact, from a formula A where the variable x appears free, we may obtain another formula, $A[t/x]$, where the term t is substituted for x. For example, in the language of arithmetic, x can be substituted in $x+0=x$ to obtain $2+0=2$.

Substitution is fundamental in describing the inference rules governing quantifiers. And bounded variables make substitution not immediately intuitive.

There are many equivalent ways to describe the substitution operation: we will use a method which is not the most immediate, but it will become very handy later in the course.

Substitution

Definition 2.6 (Substitution on terms)

Fixed a signature and a term t on it, the *substitution* of the variable $x : s$ with the term $r : s$, yielding $t[r/x]$, is defined by induction on the structure of the term t:

- if $t \equiv x$, then $t[r/x] = r$;
- if t is a variable, but $t \not\equiv x$, $t[r/x] = t$;
- if $t \equiv f(t_1, \ldots, t_n)$, then $t[r/x] = f(t_1[r/x], \ldots, t_n[r/x])$.

Notice that the substitution operation is defined only when r and x share the same type.

Substitution

Definition 2.7 (Substitution on formulae)

Fixed a signature and a formula A on it, the *substitution* of the variable $x : s$ with the term $t : s$, yielding $A[t/x]$, is defined by induction on the structure of the formula A:

- if $A \equiv \top$ or $A \equiv \bot$, then $A[t/x] = A$;
- if $A \equiv r(t_1, \ldots, t_n)$, then $A[t/x] = r(t_1[t/x], \ldots, t_n[t/x])$;
- if $A \equiv \neg B$, then $A[t/x] = \neg B[t/x]$;
- if $A \equiv B \wedge C$, $A \equiv B \vee C$, or $A \equiv B \supset C$, then $A[t/x] = B[t/x] \wedge C[t/x]$, $A[t/x] = B[t/x] \vee C[t/x]$, or $A[t/x] = B[t/x] \supset C[t/x]$, respectively;
- if $A \equiv \forall y : r.B$, or $A \equiv \exists y : r.B$, and $y : r \equiv x : s$, then $A[t/x] = A$;
- if $A \equiv \forall y : r.B$, or $A \equiv \exists y : r.B$, and $y : r \not\equiv x : s$, then $A[t/x] = \forall z : r.(B[z/y])[t/x]$, or $A[t/x] = \exists z : r.(B[z/y])[t/x]$, respectively, where $z : r \notin \mathrm{FV}(B) \cup \mathrm{FV}(t)$.

Substitution

The first clauses in the definition are obvious: we substitute the variable x with the term t where it appears.

The last but one clause means that a bounded variable cannot be substituted: this is simple to understand, as it does not make sense to substitute x with 5 in the formula $\exists x : \mathbb{N}. x^2 = x^3$. In fact, the formula is true, because $1^2 = 1 = 1^3$, but, evidently, it happens just for **some** values of x, which the existential quantifier is meant to single out.

The last clause is a bit cryptic. It says that, before performing the substitution of x with t on the quantified formula B, we should rename the quantified variable y with a **new** variable, which does not appear in B and t.

An example may clarify why this must be done: let $A \equiv \exists x : \mathbb{N}. x + y = 2y$, and let $t \equiv 2x$. If we do not rename variables, $A[t/y]$ would give $\exists x : \mathbb{N}. x + 2x = 2(2x)$, that is, $\exists x : \mathbb{N}. 3x = 4x$. We notice the A holds whenever $x = y$, but $A[t/y]$ does not. The problem is that the x in t and the one in A should be kept distinct—and we do this by renaming before performing the substitution.

Natural deduction

Definition 2.8 (Theory)

Fixed a language, a *theory* T is a set of formulae, each one usually referred to as an *axiom*.
When $T = \emptyset$, we will speak of the theory as *pure logic*.

Definition 2.9 (Proof)

Fixed a language and a theory T in it, a *proof* or *deduction* of the formula A, the *conclusion*, from a set Γ of formulae, the *hypotheses* or *assumptions*, is inductively defined by a set of inference rules summarised in the next slides.
A formula A which is the conclusion of a proof with no assumptions, is called a *theorem* in the theory T.

Natural deduction

The inference rules governing conjunctions are:

$$\frac{A \quad B}{A \wedge B}\,\wedge\mathsf{I} \qquad \frac{A \wedge B}{A}\,\wedge\mathsf{E}_1 \qquad \frac{A \wedge B}{B}\,\wedge\mathsf{E}_2$$

we have an introduction rule and two elimination rules.

Those governing disjunctions are:

$$\frac{A}{A \vee B}\,\vee\mathsf{I}_1 \qquad \frac{B}{A \vee B}\,\vee\mathsf{I}_2 \qquad \frac{A \vee B \quad \begin{array}{c}[A]\\ \vdots\\ C\end{array} \quad \begin{array}{c}[B]\\ \vdots\\ C\end{array}}{C}\,\vee\mathsf{E}$$

Natural deduction

Implication and negation are subject to the following rules:

$$\frac{\begin{array}{c}[A]\\ \vdots \\ B\end{array}}{A \supset B}\supset\mathsf{I} \qquad \frac{A \supset B \quad A}{B}\supset\mathsf{E}$$

$$\frac{\begin{array}{c}[A]\\ \vdots \\ \bot\end{array}}{\neg A}\neg\mathsf{I} \qquad \frac{\neg A \quad A}{\bot}\neg\mathsf{E}$$

Natural deduction

True and false are governed by the following rules:

$$\frac{}{\top}\,\top\mathsf{I} \qquad \frac{\bot}{A}\,\bot\mathsf{E}$$

If A is an axiom of the theory T, i.e., if $A \in T$, we are allowed to deduce it:

$$\frac{}{A}\,\mathsf{ax}$$

If A is an assumption, i.e., if $A \in \Gamma$, we can deduce it

$$A$$

Natural deduction

For every formula A, either A is true or it is false. This is expressed by the Law of Excluded Middle:

$$\frac{}{A \vee \neg A}\ \text{lem}$$

The Law of Excluded Middle is *delicate*, and it has a special status. Indeed it is used in classical logic, but not in intuitionistic logic.

Natural deduction

The rules for universal quantification are

$$\frac{A}{\forall x\colon s.A}\,\forall\mathsf{I} \qquad \frac{\forall x\colon s.A}{A[t/x]}\,\forall\mathsf{E}$$

provided that

- in $\forall E$, t is a term of type s;
- in $\forall I$, the variable $x\colon s$ does not *occur free in the proof* of the antecedent, which means that, for every assumption G, $x\colon s \notin \mathsf{FV}(G)$. This condition is, sometimes, referred to by saying that $x\colon s$ is an *eigenvariable*.

Notice the similarity between the rules for $\forall$ and for $\wedge$.

Natural deduction

Similarly, the rules for existential quantification are

$$\frac{A[t/x]}{\exists x : s.\, A}\,\exists\mathsf{I} \qquad \frac{\exists x : s.\, B \quad \begin{array}{c}[B]\\ \vdots\\ A\end{array}}{A}\,\exists\mathsf{E}$$

provided that

- in $\exists I$, t is a term of type s;
- in $\exists E$, the variable $x : s$ does not occur free in the proof of the second antecedent, that is, for every assumption G in the second subproof, except for B, $x : s \notin \mathrm{FV}(G)$ **and** $x \notin \mathrm{FV}(A)$. Again, $x : s$ is said to be an eigenvariable. Notice how this inference rule discharges the assumption B.

Notice the similarity between the rules for $\exists$ and for $\vee$.

Natural deduction

Equality is a special relation, and this is captured in a series of ad-hoc inference rules. When the language has an equality relation for some sort s, it is subject to the following rules:

$$\frac{}{\forall x : s.\, x = x}\ \text{refl} \qquad \frac{}{\forall x : s.\forall y : s.\, x = y \supset y = x}\ \text{sym}$$

$$\frac{}{\forall x : s.\forall y : s.\forall z : s.\, x = y \wedge y = z \supset x = z}\ \text{trans}$$

$$\frac{A[t/x] \quad t = r}{A[r/x]}\ \text{subst}$$

$$\frac{}{\forall x_1 : s_1 \ldots \forall x_n : s_n.\exists! z : s_0.\, z = f(x_1, \ldots, x_n)}\ \text{fun}$$

where, t and r are terms of type s, and $f : s_1 \times \cdots \times s_n \to s_0$ is a function symbol of the language.

Informal meaning

Fixed a signature $\langle S; F; R \rangle$, the intended interpretation of a sort $s \in S$ is a specific set; the intended interpretation of a function symbol is a function; and the intended interpretation of a relation symbol is a relation.

The intended meaning of equality, $=: s \times s$, when present in the language, is the identity of its arguments.

Thus, the intended meaning of a term is an element, which is identified via the interpretation of functions and the evaluation of variables, in the universe, the collection of all the sets denoted by sorts.

Informal meaning

In turn, formulae stands for a truth value, either true or false.

Atomic formulae, $r(t_1, \ldots, t_n)$, are true when the argument $(t_1, \ldots, t_n)$ is in the relation denoted by r.

A formula is universally valid, that is, $\forall x : s. A$ holds, when A is true in whatever way we interpret x as an element of the set denoted by s.

Symmetrically, a formula is existentially valid, that is, $\exists x : s. A$ holds, when there is an element e in the set denoted by s such that interpreting x as e makes A true.

Semantics

The standard semantics for first-order logic, due to Alfred Tarski, directly formalises the intended interpretation.

Definition 2.10 (Σ-structure)

Let $\Sigma = \langle S; F, R \rangle$ be a first-order signature.
Then, a Σ-*structure* $\mathscr{M} = \langle U; \mathscr{F}; \mathscr{R} \rangle$ is composed by

- a collection $U = \{u_s\}_{s \in S}$ of non-empty sets, called the *universe*,
- a collection of functions over the universe
  $\mathscr{F} = \left\{ g_f : u_{s_1} \times \cdots \times u_{s_n} \to u_{s_0} \mid f : s_1 \times \cdots \times s_n \to s_0 \in F \right\}$,
- a collection of relations over the universe
  $\mathscr{R} = \left\{ \rho_r : u_{s_1} \times \cdots \times u_{s_n} \mid r : s_1 \times \cdots \times s_n \in R \right\}$.

Semantics

To make clear the relation between a signature and a Σ-structure, we use the following notation:

- for each $s \in S$, $[\![s]\!] = u_s$;
- for each $f : s_1 \times \cdots \times s_n \to s_0 \in F$, $[\![f]\!] = g_f$;
- for each $r : s_1 \times \cdots \times s_n \in R$, $[\![r]\!] = \rho_r$.

This is called the *interpretation of the signature* in the Σ-structure.

Semantics

Definition 2.11 (Interpretation of terms)

Let $\Sigma = \langle S; F, R \rangle$ be a signature, and let $\mathscr{M}$ be a Σ-structure, with the notation as before. Also, let $\nu = \{\nu_s\}_{s \in S}$ be a collection of functions $\nu_s \colon \{v \colon v \colon s \in V\} \to [\![s]\!]$, mapping the variables of type s into the corresponding set $[\![s]\!]$.
Then, a term t is interpreted according to the following inductive definition on its structure:

- if $t \in V$ is a variable of type s, then $[\![t]\!] = \nu_s(t)$;
- if $t \equiv f(t_1, \ldots, t_n)$, then $[\![t]\!] = [\![f]\!]([\![t_1]\!], \ldots, [\![t_n]\!])$.

Semantics

Definition 2.12 (Interpretation of formulae)

Let $\Sigma = \langle S; F, R\rangle$ be a signature, let $\mathscr{M}$ be a Σ-structure, and let ν be an *evaluation of variables*, with the notation as before.
Then, a formula A is interpreted according to the following inductive definition on its structure:

- if $A \equiv \top$, $[\![A]\!] = 1$;
- if $A \equiv \bot$, $[\![A]\!] = 0$;
- if $A \equiv r(t_1, \ldots, t_n)$, $[\![A]\!] = 1$ if $([\![t_1]\!], \ldots, [\![t_n]\!]) \in [\![r]\!]$, and $[\![A]\!] = 0$ otherwise;
- if $A \equiv \neg B$, $A \equiv B \wedge C$, $A \equiv B \vee C$, $A \equiv B \supset C$, then $[\![A]\!]$ is defined as in the truth-table semantics;
- if $A \equiv \forall x\colon s.\, B$ or $A \equiv \exists x\colon s.\, B$, let $\xi = \{\xi_s\}_{s \in S}$ be an evaluation of variables such that, $\xi_\alpha = \nu_\alpha$, for each $\alpha \neq s$, and $\xi_s(v) = \nu_s(v)$ for each $v \neq x$. Then, $[\![\forall x\colon s.\, B]\!] = 1$ if, for all the possible ξ, $[\![B]\!] = 1$, and $[\![\forall x\colon s.\, B]\!] = 0$ otherwise. Also, $[\![\exists x\colon s.\, B]\!] = 1$ if, there is a ξ such that $[\![B]\!] = 1$, and $[\![\exists x\colon s.\, B]\!] = 0$ otherwise.

Semantics

We stipulate that, when equality is in the language, $[\![t_1 = t_2]\!] = 1$ exactly when $[\![t_1]\!] = [\![t_2]\!]$.

If one prefers, $[\![=_s]\!]$, the equality on the sort s, represents the *diagonal relation* $\{(x,x) : x \in [\![s]\!]\}$.

It is worth remarking that equality is always typed: $t_1 = t_2$ is a valid formula if and only if t_1 and t_2 are terms of the same sort s, and the relation $=$ should be read as a shorthand for $=_s$, which stands for the diagonal relation on the set denoted by the sort s.

Properties of the semantics

Definition 2.13 (Validity)

A formula A is *valid* or *true* in a Σ-structure $\mathscr{M}$ together with an interpretation ν of variables, when $[\![A]\!] = 1$.
A set of formulae is *valid* or *true* when each formula in the set is valid.

Definition 2.14 (Model)

Given a signature Σ for a theory T, a *model* for it is a Σ-structure together with an interpretation of variables ν which makes true the theory T.

Properties of the semantics

Theorem 2.15 (Soundness)

Given a model for T which makes true the assumptions in the finite set Δ, if A is the conclusion of a proof π from Δ in T, then A is valid.

Theorem 2.16 (Completeness)

If every model for which the finite set of assumptions Γ makes A true, then $\Gamma \vdash A$.

Theorem 2.17 (Compactness)

For any set of formulae Γ, if every finite subset of Γ has a model, then Γ has a model too.

References

Usually, first-order logic is presented in a simplified way, by avoiding the multi-sorted language, and by using a reduced number of connectives. Although this approach simplifies the initial presentation, it makes difficult to pass to other logical system, e.g., intuitionistic logic, and to deal with real mathematical theories, where multiple sorts are often present.

A good text which introduces the first-order language in a formal way is *John Bell* and *Moshé Machover*, A Course in Mathematical Logic, North-Holland, (1977), ISBN 0-7204-28440, which covers our treatment of definitions, too.

Natural deduction is described in many textbooks. This lesson follows *A.S. Troelstra* and *H. Schwichtenberg*, Basic Proof Theory, Cambridge Tracts in Theoretical Computer Science 43, Cambridge: Cambridge University Press, (1996).

References

The illustrated interpretation of formulae has been formalised first by Alfred Tarski. This is a classical definition, and it can be found in most textbooks.

The notion of model, that is, a Σ-structure which satisfies all the axioms in a theory, is analysed in depth in the branch of Logic called *model theory*. A standard reference is *C.C. Chang* and *H.J. Keisler*, Model Theory, Studies in Logic and the Foundations of Mathematics, 3^{rd} edition, Elsevier, (1990), ISBN 008088007X. Nevertheless, this text is quite dated, and an introduction to the basics of contemporary model theory can be found in *W. Hodges*, A Shorter Model Theory, Cambridge University Press, (1997), ISBN 0-521-58713-1.

The soundness theorem is a classical result and its proof can be found in most textbooks. For example, the already cited *John Bell* and *Moshé Machover*, A Course in Mathematical Logic, North-Holland, (1977), ISBN 0-7204-28440.

References

The first completeness proof for first-order logic has been given by Kurt Gödel.

In *John Bell* and *Moshé Machover*, A Course in Mathematical Logic, North-Holland, (1977), ISBN 0-7204-28440, is presented a proof which uses the techniques introduced by Leon Henkin.

Gödel's proof was his doctoral dissertation, and it is based on a obscure formalism. Henkin's proof is a substantial reorganisation of Gödel's proof, emphasising that it involves the construction of a model.

Advanced course in foundations of mathematics

Part 3

Dr Roberta Bonacina

roberta.bonacina@fsci.uni-tuebingen.de

University of Tübingen
Carl Friedrich von Weizsäcker Center

a.a. 2020/21

Syllabus

Computability:

- Motivation
- Recursive functions
- Main properties
- Comparing sets

Motivation

Computability theory is the branch of logic which studies the notion of 'computation'. Generally, it is considered in the borderline between mathematics and theoretical computer science, but, at least historically, it has been the part of logic from which computer science was born.

From a mathematical point of view, describing what can be really computed is an essential part of the XX Century's mathematics. Consider the notion of algorithm and how fundamental it revealed in many fields.

For logicians, computability theory is an essential ingredient to understand the reasons behind constructive mathematics. But it is also the fundamental tool to prove the results about the limit of formal reasoning.

Computable functions

Computability theory aims at describing the functions $\mathbb{N} \to \mathbb{N}$ which can be effectively calculated.

We notice how the vast majority of functions from naturals to naturals cannot be calculated. In fact, if we think that calculation is a process which mechanically transforms the argument of a function in its result, we have to pose a few limits on this process:

- it must take a finite amount of time;
- it must operate on a finitely generated formal language;
- it must rely on a finite description of the process which precisely describes the steps to be performed.

At least, we have a language on a finite alphabet, which is used to describe the process. No matter how we interpret the language, we know that the set of all the possible procedures is contained in the collection of finite sequences of symbols in the alphabet. So, the set of all possible procedures is in bijective correspondence with $\mathbb{N}$: it can't be bigger because the alphabet is finite, nor smaller since se may write an infinite amount of procedures. But most functions are **not** computable; indeed, the cardinality of the set of functions from $\mathbb{N}$ to $\mathbb{N}$ is $2^{|\mathbb{N}|}$, which is strictly greater than $|\mathbb{N}|$.

Computable functions

There are many ways to describe computations. For our purposes, which are not aimed at studying computations, but rather using the computable functions to reason about what can be effectively proved inside a formal system, we will use *partial recursive functions*.

In fact, we admit a computation may not terminate, hence partial functions, in which non termination is modelled as the function being undefined for the non terminating input.

Instead of using some abstract machine which 'performs' the computation, we will directly define computable functions as the class of functions that can be written in a special form. Although it is not immediately clear that this class contains all the computable functions, it is best suited to application in logic.

Primitive recursive functions

Definition 3.1 (Primitive recursive functions)

A function $f : \mathbb{N}^k \to \mathbb{N}$ is *primitive recursive* when

1. f is the *zero* function $\underline{0}(n) = 0$ for all $n \in \mathbb{N}$;
2. f is the *successor* function $\underline{\mathrm{succ}}(n) = n + 1$ for all $n \in \mathbb{N}$;
3. f is a *projection* function $U_i^k(n_1, \ldots, n_k) = n_i$ with $k \geq 1$, $1 \leq i \leq k$;
4. f is obtained by *substitution*: if $g, h_0, \ldots, h_m$ are primitive recursive functions, $f(n_1, \ldots, n_k) = g(h_0(n_1, \ldots, n_k), \ldots, h_m(n_1, \ldots, n_k))$;
5. f is obtained by *primitive recursion*: if g and h are primitive recursive functions, $f(n_1, \ldots, n_k, 0) = g(n_1, \ldots, n_k)$ and $f(n_1, \ldots, n_k, m+1) = h(n_1, \ldots, n_k, m, f(n_1, \ldots, n_k, m))$.

It is clear that primitive recursive functions are computable. It is also evident that there are computable functions which are not primitive recursive: for example, the function undefined everywhere.

Primitive recursive functions

Example 3.2

The *identity* function $\mathrm{id}(x) = x$ is primitive recursive: $\mathrm{id} = U_1^1$.

Example 3.3

The constant function $\underline{k}(x) = k$ is primitive recursive. In fact, by induction on k, if $k = 0$, $\underline{0}$ is primitive recursive by definition; if $k = k' + 1$, $\underline{k} = \mathrm{succ} \circ \underline{k'}$ by substitution, and $\underline{k'}$ is primitive recursive by induction hypothesis.

Example 3.4

Addition, multiplication and exponentiation are primitive recursive.

$$n + 0 = n \qquad n \cdot 0 = 0$$

$$n + (m+1) = \mathrm{succ}\left(U_3^3(n, m, n+m)\right) \qquad n \cdot (m+1) = n + \underline{0}(m) + n \cdot m$$

$$n^0 = \underline{1}(n)$$

$$n^{m+1} = n \cdot \underline{1}(m) \cdot n^m$$

Notice how $0^0 = 1$, which sounds odd.

Primitive recursive functions

Example 3.5

The *predecessor* function, defined by

$$\mathrm{pred}(n) = \begin{cases} n-1 & \text{when } n > 0 \\ 0 & \text{otherwise} \end{cases}$$

is primitive recursive: $\mathrm{pred}(0) = \underline{0}(0)$, and $\mathrm{pred}(n+1) = U_1^2(n, \mathrm{pred}(n))$.

Example 3.6

The *recursive difference*, defined by

$$m \dot{-} n = \begin{cases} m-n & \text{if } m \geq n \\ 0 & \text{otherwise} \end{cases}$$

is primitive recursive: $m \dot{-} 0 = m$ and $m \dot{-} (n+1) = \mathrm{pred}(m \dot{-} n)$.

Primitive recursive functions

Example 3.7

The *absolute difference* $|m-n|$ is primitive recursive:

$$|m-n| = (m \dot{-} n) + (n \dot{-} m) \ .$$

Example 3.8

The *sign* function, defined by

$$\mathrm{sg}(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{otherwise} \end{cases}$$

is primitive recursive: $\mathrm{sg}(0) = \underline{0}(0)$, and $\mathrm{sg}(n+1) = U_2^2(n, \underline{1}(n))$.

Similarly, integer division, the remainder function, integer logarithm are primitive recursive.

Primitive recursive functions

There are functions which are computable but not primitive recursive.

Definition 3.9 (Ackermann)

The *Ackermann's function* A is defined as

$$\begin{aligned} A(m,0) &= m+1 \\ A(0,n+1) &= A(1,n) \\ A(m+1,n+1) &= A\big(A(m,n+1),n\big) \quad . \end{aligned}$$

To give an impression: $A(0,0)=1,\ A(1,1)=3,\ A(2,2)=7,\ A(3,3)=61,$ but

$$A(4,4) = 2^{2^{65536}} \quad .$$

The function $\mathbb{N} \to \mathbb{N}$ given by $n \mapsto A(n,n)$ can be shown to grow faster than any primitive recursive function, so it is **not** primitive recursive.

Partial recursive functions

Definition 3.10 (Partial recursive functions)

A partial function $f : \mathbb{N}^k \to \mathbb{N}$ is *recursive* when

1. f is the *zero* function $\underline{0}(n) = 0$ for all $n \in \mathbb{N}$;
2. f is the *successor* function $\underline{\mathrm{succ}}(n) = n + 1$ for all $n \in \mathbb{N}$;
3. f is a *projection* function $U_i^k(n_1, \ldots, n_k) = n_i$ with $k \geq 1$, $1 \leq i \leq k$;
4. f is obtained by *substitution*: if $g, h_0, \ldots, h_m$ are partial recursive functions, $f(n_1, \ldots, n_k) = g(h_0(n_1, \ldots, n_k), \ldots, h_m(n_1, \ldots, n_k))$;
5. f is obtained by *primitive recursion*: if g and h are partial recursive functions, $f(n_1, \ldots, n_k, 0) = g(n_1, \ldots, n_k)$ and $f(n_1, \ldots, n_k, m+1) = h(n_1, \ldots, n_k, m, f(n_1, \ldots, n_k, m))$;
6. f is obtained by *minimalisation*: if g is a partial recursive function, then $f(n_1, \ldots, n_k) = \mu m.\,(g(n_1, \ldots, n_k, m) = 0)$, with $\mu m.\,P(m) = m_0$ if and only if $P(m_0)$ holds, and, for all $m < m_0$, $P(m)$ does not.

We will speak of *recursive* functions when we will consider only computable total functions.

Partial recursive functions

Definition 3.11

Let S be a set and R a relation. The *characteristic functions* of S and R are given by

$$\chi_S(x) = \begin{cases} 1 & \text{if } x \in S \\ 0 & \text{if } x \notin S \end{cases}$$

$$\chi_R(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } (x_1, \ldots, x_n) \in R \\ 0 & \text{otherwise} \end{cases}$$

We say that S or R is *recursive* when χ_S or χ_R are total recursive functions. We say they are *primitive recursive* when the corresponding characteristic functions are.

Example 3.12

The relation $\leq \subseteq \mathbb{N} \times \mathbb{N}$ is primitive recursive: $\chi_{\leq}(n, m) = 1 \dot{-} \mathrm{sg}(n \dot{-} m)$.

Partial recursive functions

Example 3.13

If P and Q are (primitive) recursive relations on $\mathbb{N}^k$, then so are $\neg P$, $P \wedge Q$, and $P \vee Q$.

$$\begin{aligned}
\chi_{\neg P}(x_1,\ldots,x_k) &= 1 \dot{-} \chi_P(x_1,\ldots,x_k) \\
\chi_{P\wedge Q}(x_1,\ldots,x_k) &= \chi_P(x_1,\ldots,x_k)\cdot\chi_Q(x_1,\ldots,x_k) \\
\chi_{P\vee Q}(x_1,\ldots,x_k) &= \mathrm{sg}\left(\chi_P(x_1,\ldots,x_k)+\chi_Q(x_1,\ldots,x_k)\right) \quad .
\end{aligned}$$

Example 3.14

Every finite set is primitive recursive.

Example 3.15

If R and S are primitive recursive subsets of $\mathbb{N}$, so are $\mathbb{N}\setminus R$, $R\cap S$, and $R\cup S$.

Partial recursive functions

Proposition 3.16

If $R(n_1, \ldots, n_k, m)$ is a recursive relation, then $f : \mathbb{N}^k \to \mathbb{N}$ defined by

$$f(n_1, \ldots, n_k) = \mu m.\, R(n_1, \ldots, n_k, m)$$

i.e., the least m such that $R(n_1, \ldots, n_k, m)$ holds, is partial recursive.

Proof.

Immediate by noticing that $f(n_1, \ldots, n_k) = \mu m.\, (\chi_{\neg R}(n_1, \ldots, n_k, m) = 0)$. □

Church-Turing Thesis

A function $f : \mathbb{N}^k \to \mathbb{N}$ is *computable* exactly when f is partial recursive.

Comparing sets

Comparing two sets means to establish a correspondence between them. A function, mapping all the elements of one set in the element of another does not say much. But, when the function is bijective, we may think that the two sets are equal except for a renaming of the elements in their extensions. We write $A \cong B$ to indicate that there is bijective map between the sets A and B.

Intuitively, a set A is smaller than a set B when it can be embedded into B modulo a renaming: formally, this intuition is modelled by the existence of an injective function $A \to B$. Symmetrically, A is greater than B when there is a surjective function $A \to B$.

Comparing sets

This way of comparing sets is the standard, and it works as one expects when dealing with finite sets. But, on infinite sets, it reveals that sets are far more complex objects than we may imagine at a first sight.

Theorem 3.17 (Schröder-Bernstein)

If $f : A \to B$ is injective and $g : B \to A$ is injective then $A \cong B$.

Proof. (i)

Let $C_0 = A \setminus g(B)$ and, by induction, $C_{n+1} = \{g(x) : x \in D_n\}$ and $D_n = \{f(x) : x \in C_n\}$. Define

$$h(x) = \begin{cases} f(x) & \text{if } x \in C_n \text{ for some } n \\ g^{-1}(x) & \text{otherwise} \end{cases}$$

This definition makes sense, as g is injective and $g^{-1}(x)$ is defined on $g(B)$.
↪

Comparing sets

↪ Proof. (ii)

Now, let us prove that h is bijective, i.e., $A \cong B$.

Let $x, y \in A$. Suppose $h(x) = h(y)$: if $x \in C_m$ and $y \in C_k$ for some m and k, then $f(x) = f(y)$, so $x = y$ being f injective; if $x \notin C_n$ and $y \notin C_n$ for any n, then $g^{-1}(x) = g^{-1}(y)$, so $x = y$ being g injective; if $x \in C_m$ for some m and $y \notin C_n$ for any n, $f(x) = g^{-1}(y)$, so $(g \circ f)(x) = y$, that is, $y \in C_{m+1}$, which is impossible. Thus h is injective.

We must show that $h(A) = B$. Firstly, for any n and any $z \in D_n$, $z = f(x)$ for some $x \in C_n$, so, by definition, $z = h(x)$. Then, let $z \in B \setminus \bigcup_n D_n$. Evidently, by induction on n, $g(z) \notin C_n$ for any n, thus $h(g(z)) = g^{-1}(g(z)) = z$. So h is surjective. □

Comparing sets

Example 3.18

Let $P = \{2n : n \in \mathbb{N}\}$. Since $f : P \to \mathbb{N}$ such that $f(x) = x$ is injective, and $g : \mathbb{N} \to P$ such that $g(x) = 2x$ is injective, by Theorem 3.17 we conclude that $P \cong \mathbb{N}$.

In general, an infinite set A is such that it is possible to find a proper subset $B \subset A$ such that $A \cong B$. We can even use this property as a *definition* of being infinite.

Comparing sets

Example 3.19
$\mathbb{N} \times \mathbb{N} \cong \mathbb{N}$

Evidently, the function $f : \mathbb{N} \to \mathbb{N} \times \mathbb{N}$ mapping $x \mapsto (x, x)$ is injective.
Oppositely, the function $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ defined as
$g(x, y) = (x + y)(x + y + 1)/2 + y$ is injective, as it is easy to prove. Informally, it counts the pairs using diagonals which justifies the claim of being injective: the formal proof is just arithmetic.
Thus, by Theorem 3.17 the result follows.

By induction, it follows that $\mathbb{N}^k \cong \mathbb{N}$ for any $k > 0$.

Comparing sets

Example 3.20

The collection of finite sequences of naturals $\mathbb{N}^* \cong \mathbb{N}$

Obviously, the function $f : \mathbb{N} \to \mathbb{N}^*$ mapping $x \mapsto \{x\}$ is injective.
Oppositely, calling $g_n : \mathbb{N}^n \to \mathbb{N}$ the bijection from the Cartesian product of $n \geq 1$ copies of $\mathbb{N}$ to $\mathbb{N}$, we may define a function $h : \mathbb{N}^* \to \mathbb{N} \times \mathbb{N}$ by $h(\{x_i\}_{1 \leq i \leq n}) = (n, g_n(x_1, \ldots, x_n))$. For $n = 0$, let $h(\emptyset) = (0, 0)$.
Evidently, h is injective since g_n is, for each $n \geq 1$. So, the composition $g_2 \circ h$ is injective, and the result follows by Theorem 3.17.

Comparing sets

Example 3.21

$\wp(\mathbb{N}) \ncong \mathbb{N}$.

This result, which specialises a famous Theorem by Cantor, says that the collection of subsets of $\mathbb{N}$ is **not** in bijection with $\mathbb{N}$. The proof is a classical masterpiece that introduces a technique called *diagonalisation*.

We can identify each subset $A \subseteq \mathbb{N}$ with its characteristic function $\chi_A : \mathbb{N} \to \{0,1\}$. Suppose that all these functions are in bijection with $\mathbb{N}$: then, there is a bijective function e which enumerates them. So, we have a sequence $\wp(\mathbb{N}) \cong \{\chi_{A_i}\}_{i \in \mathbb{N}}$ such that the i-th function is given by $e(i)$. Define a function $\Delta : \mathbb{N} \to \{0,1\}$ as $\Delta(x) = 1 - \chi_{A_x}(x)$. Thus Δ must appear somewhere in the sequence, i.e., $\Delta = \chi_{A_k}$ for some $k \in \mathbb{N}$. Which is impossible since $\chi_{A_k}(k) = \Delta(k) = 1 - \chi_{A_k}(k)$ and $\chi_{A_k} \in \{0,1\}$. Hence, the characteristic functions are not in bijection with $\mathbb{N}$, that is, $\wp(\mathbb{N}) \ncong \mathbb{N}$.

As a side effect, since the functions $\mathbb{N} \to \{0,1\}$ are in evident bijection with the real interval $[0,1]$, we get that $\mathbb{R} > \mathbb{N}$ strictly. In other words, infinity is not unique!

Comparing sets

Let us recall what we said at the beginning about computable functions. The language used to describe the process is based on a finite alphabet, and the set of all the possible procedures is contained in the collection of finite sequences of symbols in the alphabet.

Hence, by Example 3.20 it is at most in bijective correspondence with $\mathbb{N}$. Indeed, the set of all possible procedures is exactly in bijection with $\mathbb{N}$, because we can write an infinite amount of procedures.

Conversely, by Example 3.21, it is clear that not all functions $\mathbb{N} \to \mathbb{N}$ can be written in this language.

Universal function

Theorem 3.22 (Enumeration)

There is a partial recursive function $e(x,y)$ that enumerates all the partial recursive functions, that is, defining $\phi_x(y) = e(x,y)$, $\{\phi_x\}_{x\in\mathbb{N}}$ is the collection of all the partial recursive functions.

Proof. (i)

In the first place, we notice that, since, for any $k \in \mathbb{N}$, $\mathbb{N}^k \cong \mathbb{N}$ and the bijection is computable, we may safely reduce to enumerate the computable functions $\mathbb{N} \to \mathbb{N}$.

Partial recursive functions can be coded as naturals:

- $[\underline{0}] = 2$;
- $[\underline{\text{succ}}] = 3$
- $[U_i^k] = 5 \cdot 17^k \cdot 19^i$;
- substitution:
  $[g(h_0(n_1,\ldots,n_k),\ldots,h_m(n_1,\ldots,n_k))] = 7 \cdot 17^{[g]} \cdot 19^{[h_0]} \cdot \ldots \cdot p_{7+m}^{[h_m]}$, with $\{p_i\}_{i\in\mathbb{N}}$ the sequence of prime numbers;

↪

Universal function

↪ Proof. (ii)

- primitive recursion: $[f] = 11 \cdot 17^{[g]} \cdot 19^{[h]}$;
- minimalisation: $[f] = 13 \cdot 17^{[g]}$.

The coding is injective, so invertible, thanks to the unique factorisation in primes of any natural number. Moreover, it is computable, and the inverse is computable, too. Precisely, the coding is primitive recursive, as it is immediate to check.

Defining $\underline{\perp}$ as the partial function which is everywhere undefined, we can invert the $[\_]$ coding:

$$\phi_n = \begin{cases} f & \text{if there is } f \text{ such that} [f] = n \\ \underline{\perp} & \text{otherwise} \end{cases}$$

Since $\underline{\perp}(x) = \mu m.(\underline{1}(x) = 0)$, the decoding is computable.

Then, $e(x,y) = \phi_x(y)$. It enjoys the enumeration property by construction. □

Universal function

Proposition 3.23

There is no $\{f_x\}_{x\in\mathbb{N}}$ of all total computable functions which admits an enumeration function $e(x,z) = f_x(z)$.

Proof.

Consider the function $h(x) = f_x(x)+1$. It is total, since each f_x is.
Assume there is a recursive function e enumerating $\{f_x\}_{x\in\mathbb{N}}$. Then, $h(x) = e(x,x)+1$, so h is recursive.
But h also occurs in $\{f_x\}_{x\in\mathbb{N}}$, so there is $k\in\mathbb{N}$ such that $f_k = h$.
Thus, $h(k) = e(k,k)+1 = f_k(k)+1 = h(k)+1$, hence $0 = 1$, a contradiction. □

Universal function

Theorem 3.24

Let $m, n \geq 1$. Then, there is a computable function $S_n^m : \mathbb{N}^{m+1} \to \mathbb{N}$ such that

$$f_\alpha(x_1, \ldots, x_m, y_1, \ldots, y_n) = f_{S_n^m(\alpha, x_1, \ldots, x_m)}(y_1, \ldots, y_n) \ .$$

Although we will not prove the theorem, we want to remark its meaning: it shows that considering some arguments as parameters is an admissible operation in the computational world.

We can start the study of computable functions by considering an enumeration of them, which has a couple of properties: being computable, and satisfying the S_n^m theorem. Then

Theorem 3.25 (Turing, 1936)

There is a computable partial function $U : \mathbb{N}^2 \to \mathbb{N}$ such that $f_n(x) = U(n, x)$.

Such a function is called *universal*, and it is the first computer. But this is another story...

Fixed points

Theorem 3.26 (Kleene)

If f is a computable partial function, then exists $k \in \mathbb{N}$ for which $\phi_{f(k)} = \phi_k$ in any good enumeration of the partial recursive functions.

Proof.

Let $h(x) = \phi_x(x)$. This partial function is computable because it can be written as $h(x) = U(x, x)$. Then, $f \circ h$ is computable, too. So, $f \circ h = \phi_e$ for some $e \in \mathbb{N}$.

Therefore, $\phi_{f(h(e))} = \phi_{\phi_e(e)} = \phi_{h(e)}$. Thus $k = h(e)$ is the sought fixed point. □

References

Computability theory, also known as recursion theory is a major branch of mathematical logic. A very nice introductory text is *Barry Cooper*, Computability Theory, Chapman & Hall/CRC Mathematics, (2004), ISBN 1-58488-237-9.

Advanced course in foundations of mathematics

Part 4

Dr Roberta Bonacina

roberta.bonacina@fsci.uni-tuebingen.de

University of Tübingen
Carl Friedrich von Weizsäcker Center

a.a. 2020/21

Syllabus

Limiting results:

- Peano arithmetic
- Induction
- Standard and non-standard models
- Representable entities

Peano arithmetic

Peano arithmetic is the standard formal theory describing natural numbers and their properties.

It is composed by a series of axioms, divided into groups, and it is interpreted in classical first order logic.

The very same theory, interpreted in intuitionistic first order logic is called Heyting arithmetic. Despite they are syntactically identical, their interpretations are quite different. For example, in Peano arithmetic it is possible to show that there are functions which cannot be computed, while every function which can be proved to exist in Heyting arithmetic, is computable, because of the constructive nature of the logic.

Peano arithmetic

Peano arithmetic is based on the language generated by the the signature

$$\langle \{\mathbb{N}\}; \{0 : \mathbb{N}, S : \mathbb{N} \to \mathbb{N}, +, \cdot : \mathbb{N} \times \mathbb{N} \to \mathbb{N}\}; \{= : \mathbb{N} \times \mathbb{N}\} \rangle \ .$$

The first group of axioms defines what is a natural number:

$$\forall x, y. S\, x = S\, y \supset x = y \ ; \tag{1}$$

$$\forall x. S\, x \neq 0 \ . \tag{2}$$

The idea is that natural numbers are the elements of the free algebra generated by 0 and S. The successor function S, given a number x, calculates the next number, $x+1$. So natural numbers are written in the unary representation, and they are naturally equipped with a total order structure with minimum.

Peano arithmetic

The second group of axioms define addition and multiplication:

$$\forall x.0 + x = x \ ; \tag{3}$$

$$\forall x, y.S\,x + y = S(x + y) \ ; \tag{4}$$

$$\forall x.0 \cdot x = 0 \ ; \tag{5}$$

$$\forall x, y.S\,x \cdot y = x \cdot y + y \ . \tag{6}$$

It is worth remarking the inductive nature of these definitions.

Peano arithmetic

The third and last group of axioms is a schema: for any formula A,

$$A[0/x] \wedge (\forall x. A \supset A[S\,x/x]) \supset \forall x. A \tag{7}$$

This schema formalises induction on the structure of natural numbers:

- if A holds on 0
- and, assuming that A holds on x, we can show that it holds on $S\,x$,
- then, A holds for every $x \in \mathbb{N}$.

Induction

There is a link between induction and recursion: an inductive definition induces a recursive procedure that allows to calculate/generate the defining objects, and vice versa, a recursive procedure induces an inductive definition of its results.

Example 4.1

The axioms (3) and (4) provide a recursive schema that allows to calculate the addition:

$$x + y = \text{if } x = 0 \text{ then } y \text{ else let } x = S\,z \text{ in } S(z + y) \ ;$$

Conversely, we may say that the result of the sum is identified by induction of the first summand.

Standard model

The standard model for Peano arithmetic is the structure which interprets the signature as

- the unique sort into the set of natural numbers, denoted by $\mathbb{N}$;
- the function symbols into the zero number, the successor function, and the usual addition and multiplication, respectively.

Any model, i.e., any pair $(\mathscr{M},\sigma)$ is said to be *standard* when $\mathscr{M}$ is the structure above while no restriction is posed on the evaluation σ of variables. Although it may be confusing, we adopt the standard notation which uses the same symbols to denote the formal elements of the syntax, and their intended interpretation. In any standard model, this convention makes no difference.

Since the purpose of the theory of arithmetic is to characterise the class of standard models, it would be nice if these were the only models of the theory. Unfortunately, this is not the case.

Non-standard models

Definition 4.2 (Non-standard model)

Any structure $\mathcal{N}$ on the language of Peano arithmetic which is not isomorphic to the standard model $\mathcal{M}$ but, for any evaluation σ of variables is a model $(\mathcal{N}, \sigma)$ of Peano arithmetic, is called a *non-standard model*.

In the definition above, an isomorphism between structures $f : \mathcal{N} \to \mathcal{M}$ is

- an invertible function between the universes;
- for each term t, $f([\![t]\!]_{\mathcal{N}}) = [\![t]\!]_{\mathcal{M}}$.

If a non-standard model exists, it means that there is a structure $\mathcal{N}$ which makes Peano arithmetic true but interprets some term into an element e in the universe which cannot be mapped in some natural number.

Notice that the element e must be the image of a term under the interpretation function: so, for example, the real numbers consisting of all the non-negative integers, is **not** a non-standard model, even if it is constructed in a very different way from the naturals (all the reals are a quotient of Cauchy sequences).

Non-standard models

Proposition 4.3

There is a non-standard model for Peano arithmetic.

Proof. (i)

Define $S^0(0) = 0$, and $S^{i+1}(0) = S\,S^i(0)$. Evidently the term $S^n(0)$ gets interpreted in n in any model.

Let $\Sigma_n = \{x \neq S^i(0) \colon i < n\}$ be a collection of formulae, and let $\Sigma = \bigcup_{n\in\mathbb{N}} \Sigma_n$.

Calling $\mathscr{M}$ the structure of the standard model, and defining σ_n such that $\sigma_n(x) = n$, evidently the standard model $(\mathscr{M}, \sigma_n)$ makes Σ_n valid, together with all the axioms of Peano arithmetic.

Thus, any finite $A \subset \Sigma$ has a model, because it is contained in Σ_n for some n. Thus, by the Compactness Theorem 2.17, Σ has a model $(\mathscr{N}, \sigma)$ which makes true also all the axioms of Peano arithmetic. ↪

Non-standard models

↪ Proof. (ii)

In this model, $\sigma(x) \neq n$ for any $n \in \mathbb{N}$ because $[\![S^n(0)]\!]_{\mathscr{N}} = n$ but $x \neq S^n(0)$ occurs in Σ, so, by definition of interpretation, $\sigma(x) \neq [\![S^n(0)]\!]_{\mathscr{N}}$.

Hence, there is an element $k \notin \mathbb{N}$ such that $\sigma(x) = k$. But interpreting x on $\mathscr{M}$ leads to some $n \in \mathbb{N}$, whatever evaluation of variables we may choose. So, any function mapping $\mathscr{N}$ to $\mathscr{M}$ has to be non-invertible on the term x.

Thus, $(\mathscr{N}, \sigma)$ is a model of Peano arithmetic, which is not isomorphic to any standard model, so it is non-standard. □

Discussion

The existence of a non-standard model for Peano arithmetic shows that this theory does not describe **exactly** the natural numbers and their properties which can be expressed in the language. Here, not exactly means not only.

The first thought is to try to *complete* Peano arithmetic to prevent the construction of a model like the $(\mathcal{N}, \sigma)$ above. Clearly, the shape of the proof, using the Compactness Theorem, does not allow to obtain this result in a direct way.

However, it is not evident whether the existence of a non-standard model is disturbing: we cannot use the proof of Proposition 4.3 to write a formula which holds in the non-standard model while it does not in any standard model. In fact, we used this property to synthesise the non-standard model from the standard ones.

Discussion

Of course, we can use a theory to separate the non-standard model from any standard one: this is exactly the purpose of the Σ theory in Proposition 4.3.

But, still, it is not clear whether there is *closed* formula, i.e., a formula with no free variables, allowing to separate standard models from non-standard ones.

This would be crucial, since such a formula ϕ does not depend on the evaluation of variables, thus its truth value would be defined by the structure of the model only. In a sense, ϕ, if it exists, cannot be provable, even if it is true in any standard model, because it would be false in some non-standard model, thus, by the Soundness Theorem, it cannot be proved.

If such a ϕ exists, it means that we have a way to separate models **within** the theory of Peano arithmetic, just by adding a single axiom, ϕ, or its complement, $\neg\phi$.

Proof in natural deduction

We prove using the inference rules of first order logic a result which will be useful in the following: $\vdash A = \neg\neg A$

$$\dfrac{\dfrac{\dfrac{[\neg A]^1 \quad [A]^2}{\bot}\,\neg\mathrm{E}}{\neg\neg A}\,\neg\mathrm{I}^1}{A \supset \neg\neg A}\,\supset\mathrm{I}^2$$

$$\dfrac{\dfrac{}{A \vee \neg A}\,\mathrm{lem} \qquad \dfrac{\dfrac{\dfrac{[\neg A]^1 \quad [\neg\neg A]^2}{\bot}\,\neg\mathrm{E}}{A}\,\bot\mathrm{E} \qquad [A]^1}{A}\,\vee\mathrm{E}^1}{\neg\neg A \supset A}\,\supset\mathrm{I}^2$$

Notice that the proof of $\neg\neg A \supset A$ uses the law of excluded middle, hence the result holds in classical logic, but not in intuitionistic logic. That's why intuitionism refuses proofs by contradiction.

As an exercise, prove $\vdash \neg A = \neg\neg\neg A$ without using the law of excluded middle.

Representable entities

Definition 4.4 (Numerals)

Given $n \in \mathbb{N}$, the *numeral* $\overline{n}$ representing n is defined as $\overline{0} \equiv 0$, and $\overline{n+1} \equiv S\ \overline{n}$.

Definition 4.5 (Representation)

A relation $R \subseteq \mathbb{N}^k$ is *representable* in Peano arithmetic if and only if there is a formula ϕ such that

- if $(n_1, \ldots, n_k) \in R$ then $\vdash_{\mathrm{PA}} \phi(\overline{n_1}, \ldots, \overline{n_k})$;
- if $(n_1, \ldots, n_k) \notin R$ then $\vdash_{\mathrm{PA}} \neg\phi(\overline{n_1}, \ldots, \overline{n_k})$;

where $\vdash_{PA}$ means 'provable in Peano arithmetic'.
A function $f : \mathbb{N}^k \to \mathbb{N}$ is *representable* in Peano arithmetic if the relation $R = \{(n_1, \ldots, n_k, m) : m = f(n_1, \ldots, n_k)\}$ is representable.
A set $S \subset \mathbb{N}$ is *representable* in Peano arithmetic if its characteristic function is representable.

Representable entities

Example 4.6

Equality is representable in Peano arithmetic.

Proposition 4.7

If the relations $P, Q \subseteq \mathbb{N}^k$ are representable in Peano arithmetic, so are $\neg P$, $P \wedge Q$, and $P \vee Q$.

Proof.

Since P and Q are representable, there are ϕ_P and ϕ_Q as in Definition 4.5.
So, $(n_1, \ldots, n_k) \in \neg P$ if and only if $(n_1, \ldots, n_k) \notin P$. Thus, $\neg\phi_P$ represents $\neg P$, because $\neg\neg\phi_P(n_1, \ldots, n_k) = \phi_P(n_1, \ldots, n_k)$.
Also, $(n_1, \ldots, n_k) \in P \wedge Q$ if and only if $(n_1, \ldots, n_k) \in P$ and $(n_1, \ldots, n_k) \in Q$.
Thus, $\phi_{P \wedge Q} = \phi_P \wedge \phi_Q$. Similarly, $\phi_{P \vee Q} = \phi_Q \vee \phi_Q$. □

Representable entities

Proposition 4.8

The $\underline{0}$ function is representable.

Proof.

Since $\underline{0}\colon \mathbb{N} \to \mathbb{N}$, we have to find a formula representing $Z = \{(n,m)\colon m = \underline{0}(n)\}$. Consider $\phi_{\underline{0}}(x,y) \equiv (y = 0)$.

- If $(n,m) \in Z$, then $m = \underline{0}(n)$, so $m = 0$. Thus, $\phi_{\underline{0}}(\overline{n},\overline{m}) \equiv (\overline{m} = \overline{0}) \equiv (\overline{0} = \overline{0})$, so $\vdash_{\mathsf{PA}} \phi_{\underline{0}}(\overline{n},\overline{m})$, by reflexivity.
- If $(n,m) \notin Z$, then $m \neq \underline{0}(n)$, so $m \neq 0$. Thus, $\overline{m} \equiv S\ \overline{m'}$ and $\phi_{\underline{0}}(\overline{n},\overline{m}) \equiv (\overline{m} = \overline{0}) \equiv (S\ \overline{m'} = \overline{0})$, so $\vdash_{\mathsf{PA}} \neg\phi_{\underline{0}}(\overline{n},\overline{m})$, by axiom.

□

Representable entities

Example 4.9
The successor and projection function are representable. So are also addition and multiplication.

Example 4.10
If g and $h_o, \ldots, h_k$ are representable, so is f obtained by substitution.

Example 4.11
If g is representable, so is f obtained by minimalisation.

Example 4.12
Addition and multiplication are representable.

Representable entities

Theorem 4.13

All recursive functions are representable in Peano arithmetic.

Corollary 4.14

All recursive sets and relations are representable in Peano arithmetic.

These proofs can be found in *Elliott Mendelson*, Introduction to Mathematical Logic, CRC Press. The proof is by induction on the structure of partial recursive functions and it is far too complex to be detailed here: in fact, it is usually absent in most textbooks.
But it is a constructive proof: given a partial recursive function f, it provides an effective method to build a formula representing f.

Exercise

Write a proof in Peano arithmetic that every $n \in \mathbb{N}$ is equal or greater than zero, where $n \geq m$ is defined as $\exists x.m + x = n$.

Discuss what happens if one defines $n \geq m$ as $\exists x.x + m = n$.

References

Peano arithmetic is illustrated in most textbooks about logic.

The relation between induction and recursion is deep and complex: an introduction to this is beyond the scope of the present course. The interested reader could refer to *Benjamin C. Pierce*, Types and Programming Languages, MIT Press, (2002), ISBN 978-0-262-16209-8.

The existence of non-standard models can be shown in many different ways. Proposition 4.3 is adapted from *John Bell* and *Moshé Machover*, A Course in Mathematical Logic, North-Holland, (1977), ISBN 0-7204-28440.

The representation of relations, sets, and functions is taken from *Barry Cooper*, Computability Theory, Chapman & Hall/CRC Mathematics, (2004), ISBN 1-58488-237-9.

Advanced course in foundations of mathematics

Part 5

Dr Roberta Bonacina

roberta.bonacina@fsci.uni-tuebingen.de

University of Tübingen
Carl Friedrich von Weizsäcker Center

a.a. 2020/21

Syllabus

Limiting results:

- Gödel's First Incompleteness Theorem
- The idea behind the proof
- Coding terms
- Coding formulae
- Gödel's Second Incompleteness Theorem
- Meaning and consequences

Induction, again

The induction principle says that, fixed a property $P \subseteq \mathbb{N}$, if $0 \in P$ and, for any $n \in \mathbb{N}$, if $n \in P$ then $n+1 \in P$, then $P = \mathbb{N}$.

Clearly, the induction schema (7) in Peano arithmetic is just an approximation of the real induction principle: since $\left|\wp(\mathbb{N})\right| = 2^{|\mathbb{N}|}$ while the collection of formulae on the language of arithmetic has cardinality $|\mathbb{N}|$, we have not enough formulae to represent all the possible properties.

The gap between what can be formalised and what is the intended meaning about the structure of natural numbers, the induction principle at the first place, is responsible for non-standard models.

Incompleteness theorem

Theorem 5.1 (Gödel's Incompleteness Theorem)

Let T be an effective theory which is consistent, and able to represent all the recursive functions. Then, there is a closed formula G such that $T \nvdash G$ and $T \nvdash \neg G$.

A theory is said to be *consistent* when it does not happen that both A and $\neg A$ are derivable for any formula A, and *effective* when the set of axioms is recursive, that is, applying a *coding* to its axioms so that they become a set of numbers, this set is recursive.
A coding of Peano arithmetic, or, more in general, of recursive functions, is a total map g from the expressions of the syntax (terms, formulae, proofs) to $\mathbb{N}$ such that

- g is injective;
- g is recursive;
- g^{-1} on the image of g is recursive, too.

Strategy

The proof of the incompleteness theorem is complex. It has a difficult part, the fixed point lemma, and a lot of technicalities.

The strategy is to consider the sentence "this sentence is not provable".

- we will show that there is a coding function that maps terms, formulae and proofs into natural numbers;
- hence, it is possible to write a formula which says "there is a number p which is the code of a proof of the sentence x";
- negating that formula, we can express the fact that x is not provable;
- we will show a fixed point theorem saying that there exists a fixed point of the transformation which maps each sentence x to the code of the sentence expressing that x is not provable;
- thus, the sentence G becomes the formula stating that x is not provable with x substituted with the fixed point;
- the meaning of G is that G is not provable;
- but G must be true in the standard model, otherwise the theory would be contradictory, so the result follows.

Coding terms

In the following, for the sake of simplicity, we will assume the set of variables in the language of Peano arithmetic to be $V = \{x_i : i \in \mathbb{N}\}$.

Definition 5.2 (Coding terms)

The *Gödel's coding function* g on terms is inductively defined as follows:

- $g(0) = 2 \cdot 3$;
- $g(x_i) = 2 \cdot 3^2 \cdot 5^{i+1}$;
- $g(S\,t) = 2 \cdot 3^3 \cdot 5^{g(t)}$;
- $g(t+s) = 2 \cdot 3^4 \cdot 5^{g(t)} \cdot 7^{g(s)}$;
- $g(t \cdot s) = 2 \cdot 3^5 \cdot 5^{g(t)} \cdot 7^{g(s)}$.

Thanks to the theorem saying that natural numbers admit a unique factorisation in primes, g is computable, injective, and g^{-1} is computable.

Coding terms

A few remarks are needed:

- each code for a term is of the form $2 \cdot n$, with n odd;
- the exponent of the factor 3 tells whether the term is 0, a variable, a successor, a sum, or a multiplication;
- the parameters of a term, i.e., the index of the variable, or the arguments of the successor, of the sum, or the multiplication, are the exponents of the factors 5 and 7, in that order.

Hence, intuitively, it is possible to write a formula in Peano arithmetic that tells whether its argument is a code of a term. This can be formalised by showing that the set of codes for terms is recursive, so that Corollary 4.14 yields the result.

Coding formulae

Definition 5.3 (Coding formulae)

The *Gödel's coding function* g on formulae extends the coding of terms and it is inductively defined as follows:

- $g(\top) = 2^2 \cdot 3$;
- $g(\bot) = 2^2 \cdot 3^2$;
- $g(t = s) = 2^2 \cdot 3^3 \cdot 5^{g(t)} \cdot 7^{g(s)}$;
- $g(\neg A) = 2^2 \cdot 3^4 \cdot 5^{g(A)}$;
- $g(A \wedge B) = 2^2 \cdot 3^5 \cdot 5^{g(A)} \cdot 7^{g(B)}$;
- $g(A \vee B) = 2^2 \cdot 3^6 \cdot 5^{g(A)} \cdot 7^{g(B)}$;
- $g(A \supset B) = 2^2 \cdot 3^7 \cdot 5^{g(A)} \cdot 7^{g(B)}$;
- $g(\forall x.\, A) = 2^2 \cdot 3^8 \cdot 5^{g(A)} \cdot 7^{g(x)}$;
- $g(\exists x.\, A) = 2^2 \cdot 3^9 \cdot 5^{g(A)} \cdot 7^{g(x)}$.

Again, the coding g is computable, injective, and g^{-1} is computable, too.

Coding formulae

A few remarks are needed:

- each code for a formula is of the form $2^2 \cdot n$, with n odd, so we can separate the codes of terms from the ones of formulae just looking the exponent of the factor 2;
- the exponent of the factor 3 tells which kind of formula the code represents;
- the parameters of a formula are the exponents of the factors 5 and 7, in that order.

Hence, intuitively, it is possible to write a formula in Peano arithmetic that tells whether its argument is a code of a formula. This can be formalised by showing that the set of codes for formulae is recursive, so that Corollary 4.14 yields the result.

Coding sequences

Definition 5.4 (Coding finite sequences)

The *Gödel's coding function* g of a finite sequence $n_1, \ldots, n_k$ of natural numbers is $g(n_1, \ldots, n_k) = 2^3 \cdot \prod_{1 \leq i \leq k} p_{i+1}^{n_i+1}$, with p_j the j-th prime number.

It is clear that the coding function is injective, computable, and its inverse is computable, too. Also, the codes for sequences can be separated by the codes of terms and formulae, and the set of codes for sequences can be represented, in the sense of Corollary 4.14, by some formula of Peano arithmetic.

Coding proofs

Definition 5.5 (Coding proofs)

The *Gödel's coding function* g on proofs extends the previous coding g and it is inductively defined as:

- $g\left(\dfrac{\pi_1 : \Gamma \vdash A \quad \pi_2 : \Gamma \vdash B}{A \wedge B}\,\wedge\mathsf{I}\right) = 2^4 \cdot 3 \cdot 5^{g(\pi_1 : \Gamma \vdash A)} \cdot 7^{g(\pi_2 : \Gamma \vdash B)} \cdot 13^{g(A \wedge B)}$;
- $g\left(\dfrac{\pi : \Gamma \vdash A \wedge B}{A}\,\wedge\mathsf{E}_1\right) = 2^4 \cdot 3^2 \cdot 5^{g(\pi : \Gamma \vdash A \wedge B)} \cdot 13^{g(A)}$;
- $g\left(\dfrac{\pi : \Gamma \vdash A \wedge B}{B}\,\wedge\mathsf{E}_2\right) = 2^4 \cdot 3^3 \cdot 5^{g(\pi : \Gamma \vdash A \wedge B)} \cdot 13^{g(B)}$;
- $g\left(\dfrac{\pi : \Gamma \vdash A}{A \vee B}\,\vee\mathsf{I}_1\right) = 2^4 \cdot 3^4 \cdot 5^{g(\pi : \Gamma \vdash A)} \cdot 13^{g(A \vee B)}$;
- $g\left(\dfrac{\pi : \Gamma \vdash B}{A \vee B}\,\vee\mathsf{I}_2\right) = 2^4 \cdot 3^5 \cdot 5^{g(\pi : \Gamma \vdash B)} \cdot 13^{g(A \vee B)}$;

↪

Coding proofs

↪ (Coding proofs)

- $g\left(\dfrac{\pi_1 : \Gamma \vdash A \vee B \quad \pi_2 : \Gamma, A \vdash C \quad \pi_3 : \Gamma, B \vdash C}{C}\,\vee\mathsf{E}\right) =$
  $2^4 \cdot 3^6 \cdot 5^{g(\pi_1 : \Gamma \vdash A \vee B)} \cdot 7^{g(\pi_2 : \Gamma, A \vdash C)} \cdot 11^{g(\pi_3 : \Gamma, B \vdash C)} \cdot 13^{g(C)}$;
- $g\left(\dfrac{\pi : \Gamma, A \vdash B}{A \supset B}\,\supset\mathsf{I}\right) = 2^4 \cdot 3^7 \cdot 5^{g(\pi : \Gamma, A \vdash B)} \cdot 13^{g(A \supset B)}$;
- $g\left(\dfrac{\pi_1 : \Gamma \vdash A \supset B \quad \pi_2 : \Gamma \vdash A}{B}\,\supset\mathsf{E}\right) = 2^4 \cdot 3^8 \cdot 5^{g(\pi_1 : \Gamma \vdash A \supset B)} \cdot 7^{g(\pi_2 : \Gamma \vdash A)} \cdot 13^{g(B)}$;
- $g\left(\dfrac{\pi : \Gamma, A \vdash \bot}{\neg A}\,\neg\mathsf{I}\right) = 2^4 \cdot 3^9 \cdot 5^{g(\pi : \Gamma, A \vdash \bot)} \cdot 13^{g(\neg A)}$;
- $g\left(\dfrac{\pi_1 : \Gamma \vdash \neg A \quad \pi_2 : \Gamma \vdash A}{\bot}\,\neg\mathsf{E}\right) = 2^4 \cdot 3^{10} \cdot 5^{g(\pi_1 : \Gamma \vdash \neg A)} \cdot 7^{g(\pi_2 : \Gamma \vdash A)} \cdot 13^{g(\bot)}$;
- $g\left(\dfrac{}{\top}\,\top\mathsf{I}\right) = 2^4 \cdot 3^{11} \cdot 13^{g(\top)}$;

↪

Coding proofs

↪ (Coding proofs)

- $g\left(\dfrac{\pi : \Gamma \vdash \bot}{A}\,\bot\mathsf{E}\right) = 2^4 \cdot 3^{12} \cdot 5^{g(\pi : \Gamma \vdash \bot)} \cdot 13^{g(A)}$;
- $g\left(\dfrac{}{A \vee \neg A}\,\mathsf{lem}\right) = 2^4 \cdot 3^{13} \cdot 13^{g(A \vee \neg A)}$;
- $g\left(\dfrac{\pi : \Gamma \vdash A}{\forall x.\, A}\,\forall\mathsf{I}\right) = 2^4 \cdot 3^{14} \cdot 5^{g(\pi : \Gamma \vdash A)} \cdot 13^{g(\forall x.\, A)} \cdot 19^{g(x)}$;
- $g\left(\dfrac{\pi : \Gamma \vdash \forall x.\, A}{A[t/x]}\,\forall\mathsf{E}\right) = 2^4 \cdot 3^{15} \cdot 5^{g(\pi : \Gamma \vdash \forall x.\, A)} \cdot 13^{g(A[t/x])} \cdot 17^{g(t)} \cdot 19^{g(x)}$;
- $g\left(\dfrac{\pi : \Gamma \vdash A[t/x]}{\exists x.\, A}\,\exists\mathsf{I}\right) = 2^4 \cdot 3^{16} \cdot 5^{g(\pi : \Gamma \vdash A[t/x])} \cdot 13^{g(\exists x.\, A)} \cdot 17^{g(t)} \cdot 19^{g(x)}$;
- $g\left(\dfrac{\pi_1 : \Gamma \vdash \exists x.\, A \quad \pi_2 : \Gamma, A \vdash B}{B}\,\exists\mathsf{E}\right) =$
  $2^4 \cdot 3^{17} \cdot 5^{g(\pi_1 : \Gamma \vdash \exists x.\, A)} \cdot 7^{g(\pi_2 : \Gamma, A \vdash B)} \cdot 13^{g(B)} \cdot 19^{g(x)}$;

↪

Coding proofs

↪ (Coding proofs)

- $g\left(\dfrac{}{\forall x.x=x}\,\mathsf{ax}\right) = 2^4 \cdot 3^{18} \cdot 13^{g(\forall x.x=x)} \cdot 19^{g(x)}$;
- $g\left(\dfrac{}{\forall x,y.x=y \supset y=x}\,\mathsf{ax}\right) = 2^4 \cdot 3^{19} \cdot 13^{g(\forall x,y.x=y\supset y=x)} \cdot 19^{g(x,y)}$;
- $g\left(\dfrac{}{\forall x,y,z.x=y \wedge y=z \supset x=z}\,\mathsf{ax}\right) =$
  $2^4 \cdot 3^{20} \cdot 13^{g(\forall x,y,z.x=y\wedge y=z\supset x=z)} \cdot 19^{g(x,y,z)}$;
- $g\left(\dfrac{\pi_1 : \Gamma \vdash A[t/x] \quad \pi_2 : \Gamma \vdash t=r}{A[r/x]}\,\mathsf{ax}\right) =$
  $2^4 \cdot 3^{21} \cdot 5^{g(\pi_1 : \Gamma\vdash A[t/x])} \cdot 7^{g(\pi_2 : \Gamma\vdash t=r)} \cdot 13^{g(A[r/x])} \cdot 19^{g(x)}$;
- $g\left(\dfrac{}{\forall x_1,\ldots,x_n.\exists! z.z=f(x_1,\ldots,x_n)}\,\mathsf{ax}\right) =$
  $2^4 \cdot 3^{22} \cdot 13^{g(\forall x_1,\ldots,x_n.\exists! z.z=f(x_1,\ldots,x_n))} \cdot 17^{g(f(x_1,\ldots,x_n))} \cdot 19^{g(x_1,\ldots,x_n,z)}$;
- $g\left(\dfrac{}{\forall x.Sx\neq 0}\,\mathsf{ax}\right) = 2^4 \cdot 3^{23} \cdot 13^{g(\forall x.Sx\neq 0)} \cdot 19^{g(x)}$;

↪

Coding proofs

↪ (Coding proofs)

- $g\left(\dfrac{}{\forall x,y.\,Sx=Sy\supset x=y}\,\mathsf{ax}\right)=2^4\cdot 3^{24}\cdot 13^{g(\forall x,y.\,Sx=Sy\supset x=y)}\cdot 19^{g(x,y)}$;
- $g\left(\dfrac{}{\forall x.\,0+x=x}\,\mathsf{ax}\right)=2^4\cdot 3^{25}\cdot 13^{g(\forall x.\,0+x=x)}\cdot 19^{g(x)}$;
- $g\left(\dfrac{}{\forall x,y.\,Sx+y=S(x+y)}\,\mathsf{ax}\right)=2^4\cdot 3^{26}\cdot 13^{g(\forall x,y.\,Sx+y=S(x+y))}\cdot 19^{g(x,y)}$;
- $g\left(\dfrac{}{\forall x.\,0\cdot x=0}\,\mathsf{ax}\right)=2^4\cdot 3^{27}\cdot 13^{g(\forall x.\,0\cdot x=0)}\cdot 19^{g(x)}$;
- $g\left(\dfrac{}{\forall x,y.\,Sx\cdot y=x\cdot y+y)}\,\mathsf{ax}\right)=2^4\cdot 3^{28}\cdot 13^{g(\forall x,y.\,Sx\cdot y=x\cdot y+y))}\cdot 19^{g(x,y)}$;
- $g\left(\dfrac{}{A[0/x]\wedge(\forall x.\,A\supset A[Sx/x])\supset\forall x.\,A}\,\mathsf{ax}\right)=$
  $2^4\cdot 3^{29}\cdot 5^{g(A)}\cdot 13^{g(A[0/x]\wedge(\forall x.\,A\supset A[Sx/x])\supset\forall x.\,A)}\cdot 19^{g(x)}$;
- if $A\in\Gamma$ is a proof by assumption, $g(A)=2^4\cdot 3^{30}\cdot 5^{g(A)}\cdot 7^{g(\Gamma)}\cdot 13^{g(A)}$ with $\Gamma=\{\gamma_1,\ldots,\gamma_n\}$ and $g(\Gamma)=g(\gamma_1,\ldots,\gamma_n)$.

↪

Coding proofs

↪ (Coding proofs)

It should be remarked that $g(e_1,\ldots,e_n)$, when e_i are not numbers should be read as $g(g(e_1),\ldots,g(e_n))$, i.e., the code of the sequence of codes of the elements.

Although it is long and tedious to verify, g is injective, computable, and g^{-1} is recursive. Also, the coding function is written down to make easy to tell pieces apart. For example, the code of the conclusion is always the exponent of the 13 factor.

Numeral

Definition 5.6 (Numeral)

The *numeral* $\ulcorner A \urcorner$ of a formula A is defined as $\ulcorner A \urcorner = S^{g(A)}(0)$, that is, the code of A written in the syntax of Peano arithmetic.

Similarly, the numeral of a term t is $\ulcorner t \urcorner = S^{g(t)}(0)$, the numeral of a proof π is $\ulcorner \pi \urcorner = S^{g(\pi)}(0)$, and the numeral of a sequence is $\ulcorner e_1, \ldots, e_n \urcorner = S^{g(e_1,\ldots,e_n)}(0)$.

Numerals allow to *internalise* the codes: we can, indirectly, speak of a formula (term, proof, sequence) by stating a property of its code. As soon as the property does not rely on the value, but on the "meaning" of the code, this is a perfectly reasonable way to proceed.

Fixed point lemma

Lemma 5.7 (Fixed point)

Let Ξ be a theory in which every (primitive) recursive function is representable, and let A be a formula such that $\mathrm{FV}(A) = \{y\}$. Then, there is a formula δ_A such that $\mathrm{FV}(\delta_A) = \emptyset$ and $\vdash_\Xi \delta_A = A[\ulcorner \delta_A \urcorner / y]$.

Proof. (i)

Let $\Delta_{\mathscr{F}}$ be the map from formulae to formulae defined by $\Delta_{\mathscr{F}}(B) \equiv \exists x_1 . x_1 = \ulcorner B \urcorner \wedge B$. This function is total, computable and injective.

Thus, the map $\Delta_{\mathbb{N}}$ defined by $\Delta_{\mathbb{N}}(g(B)) = g(\Delta_{\mathscr{F}}(B))$ is total on the image of g, (primitive) recursive, and injective.

By hypothesis, there is a formula Δ with $\mathrm{FV}(\Delta) = \{x, y\}$ such that Δ represents the function $\Delta_{\mathbb{N}}$.

Let $F \equiv \exists y . \Delta[x_1/x] \wedge A$. Clearly, $\mathrm{FV}(F) = \{x_1\}$. Also, let $\delta_A = \Delta_{\mathscr{F}}(F)$, that is, $\delta_A \equiv \exists x_1 . x_1 = \ulcorner F \urcorner \wedge F$. Thus, $\mathrm{FV}(\delta_A) = \emptyset$. ↪

Fixed point lemma

↪ Proof. (ii)

Since $\exists y.\Delta[\ulcorner F \urcorner/x] \wedge A$ implies $\exists x_1, y.\Delta[x_1/x] \wedge A$ with $x_1 = \ulcorner F \urcorner$, we can prove that $\exists x_1.x_1 = \ulcorner F \urcorner \wedge \exists y.\Delta[x_1/x] \wedge A$, which is just δ_A, Hence, we have shown that $\vdash (\exists y.\Delta[\ulcorner F \urcorner/x] \wedge A) \supset \delta_A$.

Conversely, $\delta_A \equiv \exists x_1.x_1 = \ulcorner F \urcorner \wedge \exists y.\Delta[x_1/x] \wedge A$, so δ_A implies $\exists x_1, y.\Delta[x_1/x] \wedge A$ with $x_1 = \ulcorner F \urcorner$, thus we can prove that $\exists y.\Delta[\ulcorner F \urcorner/x] \wedge A$. Hence, we have shown that $\vdash \delta_A \supset (\exists y.\Delta[\ulcorner F \urcorner/x] \wedge A)$, thus δ_A and $\exists y.\Delta[\ulcorner F \urcorner/x] \wedge A$ are equivalent.

But Δ represents $\Delta_{\mathbb{N}}$, so Ξ allows to prove, for each $n \in \mathbb{N}$, $\vdash \Delta[S^n(0)/x] = (y = S^{\Delta_{\mathbb{N}}(n)}(0))$. Specialising to $n = g(F)$, we obtain $\vdash \forall y.\Delta[\ulcorner F \urcorner/x] = (y = S^{\Delta_{\mathbb{N}}(g(F))}(0))$. ↪

Fixed point lemma

↪ Proof. (iii)

So the previous equivalence $\vdash \delta_A = (\exists y.\, \Delta[\ulcorner F \urcorner / x] \wedge A)$ allows to derive $\vdash \delta_A = (\exists y.\, y = S^{\Delta_{\mathbb{N}}(g(F))}(0) \wedge A)$.

Evidently, we can prove $\vdash A\left[S^{\Delta_{\mathbb{N}}(g(F))}(0)/y\right] = \left(\exists y.\, y = S^{\Delta_{\mathbb{N}}(g(F))}(0) \wedge A\right)$, thus we can immediately prove $\vdash \delta_A = A\left[S^{\Delta_{\mathbb{N}}(g(F))}(0)/y\right]$.

But $\ulcorner \delta_A \urcorner = S^{g(\delta_A)}(0) = S^{g(\Delta_{\mathscr{F}}(F))}(0) = S^{\Delta_{\mathbb{N}}(g(F)))}(0)$. Thus, the proof above can be rephrased as $\vdash \delta_A = A[\ulcorner \delta_A \urcorner / y]$. □

Fixed point lemma

Let us look at the proof in another order

Proof. (i)

We need to prove that there is δ_A such that $\vdash \delta_A = A[\ulcorner \delta_A \urcorner / y]$. The idea is to synthesize δ_A.

Let us see A as a transformation ψ: Formulae $\rightarrow$ Formulae such that $\psi(B) = A[\ulcorner B \urcorner / y]$. Our objective is to prove $\vdash \delta_A = \psi(\delta_A)$.

It would be better to separate A from substitution: more generally, we show $\vdash B[k/z] = (\exists z.z = k \wedge B)$. One direction is obvious for $\exists E$, and subst, the other is easy: refl, then $\exists I$, $\exists E$ and subst give B by hyothesis, then $\wedge I$ and $\exists I$ complete the proof. $\hookrightarrow$

Fixed point lemma

↪ Proof. (ii)

Define $\Delta_{\mathscr{F}}$, which separate a formula from the substitution of its coding: $\Delta_{\mathscr{F}}(B) \equiv \exists x_1.x_1 = \ulcorner B \urcorner \wedge B$. Hence, $\vdash B[k/z] = \Delta_{\mathscr{F}}(B)$.

We imagine that it's easier if $\delta_A = \Delta_{\mathscr{F}}(F)$. The idea is to synthesize F and then define δ_A as $\Delta_{\mathscr{F}}(F)$.

$\Delta_{\mathscr{F}}(F)$ is a total computable function. If we codify the formulae with the coding g we obtain $\Delta_{\mathbb{N}}\colon g(\text{Formulae}) \to \mathbb{N}$, with $\Delta_{\mathbb{N}}(g(B)) \equiv g(\Delta_{\mathscr{F}}(B))$. This function $\Delta_{\mathbb{N}}$ is computable, hence representable. Let it be represented by the formula Δ. ↪

Fixed point lemma

↪ Proof. (iii)

Suppose the existence of δ_A and calculate:
$A[\ulcorner \delta_A \urcorner / y] = A[\ulcorner \Delta_{\mathscr{F}}(F) \urcorner / y] = A[\ulcorner \Delta_{\mathbb{N}}(g(F)) \urcorner / y] = \exists y.(y = \ulcorner \Delta_{\mathbb{N}}(g(F)) \urcorner) \wedge A =$
$\exists y.(y = \Delta(\ulcorner F \urcorner)) \wedge A = \exists x_1.(x_1 = \ulcorner F \urcorner) \wedge (\exists y.\Delta[x_1/x] \wedge A)$.

Hence, the choice of F is obvious: $F \equiv \exists y.\Delta[x_1/x] \wedge A$.

Indeed, the previous chain of equalities becomes
$A[\ulcorner \delta_A \urcorner / y] = \exists x_1.(x_1 = \ulcorner F \urcorner) \wedge F = \Delta_{\mathscr{F}}(F) = \delta_A$, which completes the proof. □

Provability predicate

Definition 5.8 (Provability predicate)

The formula $\mathscr{D}$ with $\mathrm{FV}(\mathscr{D}) = \{x, y\}$ is defined as
$\mathscr{D} \equiv \exists z.13^y \cdot z = x \wedge \mathrm{isExpr}(x) \wedge \mathrm{isExpr}(y) \wedge \mathrm{isProof}(x) \wedge \mathrm{isFormula}(y)$.

The *provability predicate* T is the formula $\exists x.\mathscr{D}$, having $FV(T) = \{y\}$.

Clearly, $\mathscr{D}[\ulcorner \pi \urcorner / x, \ulcorner A \urcorner / y]$ holds exactly when A is the conclusion of the proof $\pi \colon \vdash A$. And, consequently, $T[\ulcorner A \urcorner / y]$ holds when A is provable.

The formulae isExpr(x), isExpr(y), isProof(x), and isFormula(y) in the definition of $\mathscr{D}$ have not been made explicit. While isProof(x) can be defined as $\exists z.2^4 \cdot z = x$, and isFormula(y) can be defined as $(\exists z.2^3 \cdot z = x) \wedge \neg$isProof(x), the definition of isExpr comes from the fact that the collection of codes forms a recursive set. It could be written down in an explicit way, but it is a cumbersome formula.

Incompleteness theorem

Theorem 5.9 (Gödel's Incompleteness Theorem)

Let Ξ be an effective theory which is consistent, and able to represent all the recursive functions. Then, there is a closed formula G such that $\Xi \nvdash G$ and $\Xi \nvdash \neg G$.

Proof.

Consider the formula $\neg T[x/y]$: applying the fixed point lemma, there is G such that $\mathrm{FV}(G) = \emptyset$ and $\vdash G = \neg T[\ulcorner G \urcorner / y]$.

Assume there is $\pi : \vdash G$. Then $\vdash \neg T[\ulcorner G \urcorner / y]$. But, because $\pi : \vdash G$, it holds that $\vdash \mathscr{D}[\ulcorner \pi \urcorner / x, \ulcorner G \urcorner / y]$, and thus $\vdash \exists x.\mathscr{D}[\ulcorner G \urcorner / y]$, that is, $\vdash T[\ulcorner G \urcorner / y]$, making the theory non consistent. Hence $\nvdash G$.

Oppositely, suppose there is $\pi : \vdash \neg G$. Then $\vdash T[\ulcorner G \urcorner / y]$ by definition of G, so $\vdash \exists x.\mathscr{D}[\ulcorner G \urcorner / y]$. But this means that there exists $\theta : \vdash G$ with $x = \ulcorner \theta \urcorner$. Thus, again, we get a contradiction. Thus $\nvdash \neg G$. □

Conditions on T

Notice that the three conditions on the theory Ξ are necessary:

- The collection of all the true formulae on the natural numbers with the same signature as Peano arithmetic is able to represent all the partial recursive functions, and it is consistent. Evidently, it is not effective. However, it is also complete. Hence, to apply the incompleteness theorem the theory must be effective.
- The empty theory is effective and consistent, and it corresponds to the pure logic, which is complete. Evidently, it does not allow to represent the partial recursive functions.
- The theory containing all the formulae which could be written in the language of Peano arithmetic is clearly effective and represents all the recursive functions. Clearly, it is not consistent, and, obviously, it is complete in a trivial sense.

Provability predicate

Notice that in the proof of Theorem 5.9 we implicitly used that $\vdash T[\ulcorner A \urcorner / y]$ holds when A is provable.

Remember that $\vdash \exists x.\mathscr{D}[\ulcorner A \urcorner / y]$ means that there is a code x for a proof. Since the codes are natural numbers, we suppose that x is interpreted in a natural. Which is obvious in the standard models for Peano arithmetic, but we have seen that the non standard models contain elements which aren't numbers.

Hence, in the standard model $\vdash T[\ulcorner A \urcorner / y]$, $\vdash \exists x.\mathscr{D}[\ulcorner A \urcorner / y]$ and "A is provable" are equivalent, while this is not clear in non standard models. Hence, to use the provability predicate T it becomes useful to study its properties.

Properties of provability

Proposition 5.10

For any pair of formulae A and B in Peano arithmetic,

1. $\vdash T[\ulcorner A \urcorner / y]$ *if and only if* $\vdash A$;
2. $\vdash T[\ulcorner A \supset B \urcorner / y] \wedge T[\ulcorner A \urcorner / y] \supset T[\ulcorner B \urcorner / y]$;
3. $\vdash T[\ulcorner A \urcorner / y] = T[\ulcorner T[\ulcorner A \urcorner / y] \urcorner / y]$;
4. $\vdash (T[\ulcorner A \urcorner / y] \wedge T[\ulcorner B \urcorner / y]) = T[\ulcorner A \wedge B \urcorner / y]$;
5. *if* $\vdash A \supset B$ *then* $\vdash T[\ulcorner A \urcorner / y] \supset T[\ulcorner B \urcorner / y]$;
6. *if* $\vdash (T[\ulcorner A \urcorner / y] \wedge A) \supset B$, *then* $\vdash T[\ulcorner A \urcorner / y] \supset T[\ulcorner B \urcorner / y]$.

These properties show that (i) the provability predicate T allows to prove A whenever there is a proof that A is provable; (ii) it acts naturally with respect to implication and conjunction; (iii) proving provability is equivalent to prove that provability is provable.

Properties of provability

Proposition 5.11

In Peano arithmetic, if $\vdash A = \neg T[\ulcorner A \urcorner / y]$*, then* $\vdash T[\ulcorner A \urcorner / y] = T[\ulcorner \bot \urcorner / y]$.

Again, without proving it, the proposition says that every formula, which behaves like Gödel's G, is provable if and only if $\bot$ is provable, a fact that captures the content of Theorem 5.9. But, and this is important, the proposition proves that this fact holds **inside** the theory, which is not obvious.

Second incompleteness theorem

Theorem 5.12 (Gödel's second incompleteness theorem)

There is no provable formula C in Peano arithmetic which codes the consistency of the theory, i.e., such that $\vdash C \supset \neg T[\ulcorner \bot \urcorner / y]$.

Proof.

Suppose there is C such that $\vdash C$ and $\vdash C \supset \neg T[\ulcorner \bot \urcorner / y]$. Then, $\vdash \neg T[\ulcorner \bot \urcorner / y]$, which means that $\bot$ is not provable, that is, Peano arithmetic cannot contain a contradiction, hence it is consistent.

From Theorem 5.9, there is a formula G such that $\vdash G = \neg T[\ulcorner G \urcorner / y]$, but $\nvdash G$. By Proposition 5.11, $\vdash T[\ulcorner G \urcorner / y] = T[\ulcorner \bot \urcorner / y]$, so $\nvdash \neg T[\ulcorner \bot \urcorner / y]$. Thus, we have a contradiction, showing that C cannot exist. □

Mathematical meaning

The incompleteness theorems closes the quest for a universal, self-contained foundation of Mathematics which is able to prove its own consistency. Simply, such a system cannot exist.

Nevertheless, these theorems opened the way to many developments, and to some of the other fundamental results in 20^{th} Century:

- the effective construction of non-computable functions
- the idea of coding lead to reason "modulo a coding function", which has been influential in algebra, algebraic geometry, algebraic topology, number theory, ...
- examples of independent statements arose in many fields, and they shed lights to a variety of hidden aspects of apparently clean notions.

Foundational consequences

Having a mathematical theory T which is powerful enough to represent Peano arithmetic has the consequence that we cannot prove its consistency within T. We need a theory T', containing T, and more powerful.

This fact led to the development of many hierarchies of formal systems to classify the power of mathematical theories: we scratched just the surface, by showing that the consistency of Peano arithmetic can be proved in a stronger system. But, how much stronger? Since the proof of Gödel's results, much deeper analyses have been conducted, and nowadays this part of Logic is a complex, intricate, difficult field on its own.

In constructive mathematics, the same fact led to doubt that "truth" is the right concept to analyse, and there are approaches favouring the notion of provability as the real foundation of Mathematics. This has a number of consequences, which we do not want to discuss here.

Understanding

For a very long time, mathematicians regarded the incompleteness theorems as strange beasts: something which is important, but, essentially, with no influence in the mathematical practise.

For example, the textbook of Bell and Machover we referred to many times, explicitly says that the sentences which are not provable in Peano arithmetic are not important in arithmetic, because they have no "arithmetical" content, but just a logical content. This is true for the sentence G, and for most other sentences we can construct within the logical analysis.

Unfortunately, there are purely arithmetical properties of genuine interest for mathematicians not working in logic, which are independent from Peano arithmetic.

Incompleteness and computability

The incompleteness results have proved to be extremely useful in the study of computability. In fact, using the coding techniques developed to establish Gödel's theorem, a number of limiting results about what is computable have been derived.

Also, analogously to the notion of independence, it is possible to develop hierarchies of machines, computing modulo an oracle, that allow to classify the difficulty in solving problems, either by showing their distance to what is computable, e.g, the *arithmetic hierarchy*, or comparing them to efficient procedures to solve problems, e.g., the *polynomial hierarchy*.

Incompleteness and computability

As an example we show that, fixed an enumeration $\{\phi_i\}$ of all recursive functions,

Proposition 5.13

The set $H = \{i \in \mathbb{N} : \phi_i(i) \textit{is defined}\}$ *is not recursive.*

Proof.

Suppose it is. Then, its characteristic function χ_H has to be recursive and total. So

$$f(x) = \begin{cases} 0 & \text{when } \chi_H(x) = 0 \\ 1 + \phi_x(x) & \text{otherwise} \end{cases}$$

has to be recursive and total, too. Thus, there is $j \in \mathbb{N}$ such that $f = \phi_j$. If $f(j) = 0$, then $\phi_j(j) = 0$, so $\chi_H(j) = 0$, which makes $j \notin H$, that is, $\phi_j(j) = f(j)$ is not defined, despite it has the value 0. Hence, $f(j) \neq 0$. So $f(j) = 1 + \phi_j(j) = 1 + f(j)$, thus $0 = 1$. So, H cannot be recursive. □

Natural incompleteness

Theorem 5.14 (Paris, Harrington)

For all $e, r, k \in \mathbb{N}$, there is $M \in \mathbb{N}$ such that, for every $f : \{F \subseteq \{0, \ldots, M\} : |F| = e\} \to \{0, \ldots, r\}$, there is $H \subseteq \{0, \ldots, M\}$ such that

- $|H| \geq \max\{k, \min H\}$, *and*
- *exists $v \leq r$ such that, for all $F \subseteq H$ with $|F| = n$, $f(x) = v$ for each $x \in F$.*

By using the Infinite Ramsey Theorem, it is not too difficult to derive a value $M \in \mathbb{N}$ which makes the statement true on naturals. This proof is carried out either in second-order arithmetic, with the full induction principle, or in a suitable set theory. Nevertheless, it is possible to show, *within Peano arithmetic*, that the combinatorial principle in Theorem 5.14 implies the consistency of Peano arithmetic, thus it is impossible to prove in that theory, according to Gödel's second incompleteness theorem.

Natural incompleteness

Actually, a simplified version of Theorem 5.14 suffices:

Theorem 5.15

For all $n \in \mathbb{N}$, there is $M \in \mathbb{N}$ such that, for every function $f : \{F \subseteq \{0,\ldots,M\} : |F| = n\} \to \{0,1\}$, there is $H \subseteq \{0,\ldots,M\}$ for which, for all $F \subseteq H$ with $|F| = n$, $f(F) = \{0\}$, and $|H| > n(2^{n \min H} + 1)$.

This theorem and the previous one are *natural* in the sense that, changing the first condition in Theorem 5.14 to $|H| \geq k$, we get the Finite Ramsey Theorem, which is provable inside Peano arithmetic, and which is the starting point for a large branch of Combinatorics.

Natural incompleteness

Another important theorem from a different branch of combinatorics is independent from Peano arithmetic: it holds in the standard model, but we cannot prove it in the theory. This is the famous Kruskal's theorem on trees. A simplified version suffices to yield the independence result.

Theorem 5.16

There is some $n \in \mathbb{N}$ such that, if $T_1, \ldots, T_n$ is a finite sequence of trees, where T_k has $k+n$ vertices, then, for some $i < j$, there is an injective map $f : T_i \to T_j$ between the vertices of the trees which preserves paths.

The independence proof for this theorem follows a different pattern: it is possible to show that any function which provably exists in Peano arithmetic cannot grow too fast, but the above theorem allows to construct a function which grows even faster. And this suffices to establish the fact that the theorem is unprovable in Peano arithmetic.

Natural incompleteness

Kruskal's Theorem plays an important role in the algebra of well quasi orders, a topic which has shown relevance in proving the termination of algorithms, so the above independence result has a direct, negative, application to Computer Science, for example.

In this sense, Kruskal's Theorem is "natural" and practically significant.

Incompleteness in set theory

Axiom 5.17 (Choice)

For any non empty family $\{X_i\}_{i\in I}$ of non empty sets such that $X_i \cap X_j = \emptyset$ for any $i, j \in I$, $i \neq j$, there exists a function $f : I \to \bigcup_{i\in I} X_i$ such that $f(i) \in X_i$ for every $i \in I$.

The meaning is that, whenever we are given such a family, we have the ability to make a choice that simultaneously pick an element from each set.

Some important results in Mathematics require the Axiom of Choice to be proved: as a small collection of examples, take

- every non empty vector space has a base;
- every field has an algebraic closure, which is unique modulo isomorphisms;
- the notion of adjunction in category theory;
- the compactness theorem in first order logic.

Incompleteness in set theory

The Axiom of Choice, the Continuum Hypothesis, and the Generalised Continuum Hypothesis are independent from **ZF**.

All these statements are "natural", as they state properties of sets which are inherently of interest, either because of their consequences, or because they impose a regular structure over the objects we want to study.

In fact, the independence results in set theory and in Peano arithmetic are related.

Ordinal analysis

There is a branch of proof theory devoted to study the "power" of deductive systems.

This is a deep, delicate, difficult, and complex part of logic, still in development: it is sometimes referred to as "reverse mathematics" when the goal is to find the minimal theory in which a given statement can be shown to hold.

References

The original proof of the first incompleteness theorem can be found in *Kurt Gödel*, Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I, Monatshefte für Mathematik und Physik 38, 173–198, (1931).

The proof has been generalised and polished by Rosser, and we have shown a slightly reworked version of Rosser's result. The reference is *John Barkley Rosser*, Extensions of some theorems of Gödel and Church, Journal of Symbolic Logic 1, 87–91 (1936).

Nevertheless, the proof of the first incompleteness theorem presented here roughly follows some unpublished notes from the course held by Silvio Valentini in 1991.

In *John Bell* and *Moshé Machover*, A Course in Mathematical Logic, North-Holland, (1977), ISBN 0-7204-28440 can be found a proof of both the first and second incompleteness theorem.

References

The discussion is general, and there is no specific reference for it. Some ideas could be found in *Jon Barwise*, Handbook of Mathematical Logic, Studies in Logic and the Foundations of Mathematics 90, North-Holland, (1977), ISBN 0-444-863888-5.

As an example of a (very) popular book which deals with incompleteness, we signal *D. Hofstadter*, Gödel, Escher, Bach: an Eternal Golden Braid, Basic books, (1979), ISBN 0-465-02656-7. It is an enjoyable account for non-specialists, but it also contains many debatable points and opinions. Nevertheless, the mathematical content is, essentially, precise—and the author won the Pulitzer prize for non-fiction.

A nice introduction to computability theory, which explains also the origin of the results can be found in *Barry Cooper*, Computability Theory, Chapman & Hall/CRC Mathematics, (2004), ISBN 1-58488-237-9.

References

A dated, but still valid reference for Ramsey theory is *R. Graham, B. Rothschild, J.H. Spencer*, Ramsey Theory, 2nd edition, John Wiley and Sons, (1990), ISBN 0-4715-0046-1.

The original paper *J.B. Kruskal*, Well-quasi-ordering, the tree theorem, and Vazsonyi's conjecture, Transactions of the American Mathematical Society 95(2), pp. 210–225, American Mathematical Society, (1960), is an inspiring introduction to the theorem and its motivation.

Although there are many texts providing a general overview of combinatorics, a good one is *M. Bóna*, A Walk Through Combinatorics, 2nd edition, World Scientific, (2006), ISBN 981-256-885-9.

References

The link between Kruskal's theorem and logic is analysed in depth in *J.H. Gallier*, What's so special about Kruskal's theorem and the ordinal Γ_0? A survey of some results in proof theory, Annals of Pure and Applied Logic 53(3), pp. 199-260, (1991).

The original publication about the Paris-Harrington theorem can be found in *Jon Barwise*, Handbook of Mathematical Logic, Studies in Logic and the Foundations of Mathematics 90, North-Holland, (1977), ISBN 0-444-863888-5.

A nice reference to elementary set theory, which explains the nature of the Axiom of Choice with some detail is *P. Suppes*, Axiomatic Set Theory, Dover Publishing, (1972), ISBN 0-486-61630-4.

References

The classical text *Kenneth Kunen*, Set Theory: An Introduction to Independence Proofs, Studies in Logic and the Foundations of Mathematics 102, Elsevier, (1980) provides a more in-depth discussion, extending far beyond the limits of this course.

The continuum hypothesis is the main subject of the essays in *Paul J. Cohen*, Set Theory and the Continuum Hypothesis, Dover Publishing, (2008), ISBN 0-486-46921-2. This text contains the proof that the continuum hypothesis is independent from the other axioms of **ZFC**. Students should be warned that its content is advanced material.

Finally, a fine introduction to ordinal analysis can be found in *Michael Rathjen*, The art of ordinal analysis, Proceedings of the International Congress of Mathematicians, volume 2, pp. 45–70, (2006), ISBN 978-3-03719-022-7, written by a master of the field.

Advanced course in foundations of mathematics

Part 6

Dr Roberta Bonacina

roberta.bonacina@fsci.uni-tuebingen.de

University of Tübingen
Carl Friedrich von Weizsäcker Center

a.a. 2020/21

Syllabus

Proofs and computations:

- λ calculus
- the simple theory of types
- propositions as types
- normalisation
- Martin-Löf type theory
- Homotopy type theory

λ-calculus

The λ-calculus is a family of formal systems, based on Alonzo Church's work in the 1930s. These systems are deputed to describe computable functions using the simplest syntax. Surprisingly, not only they describe computable functions, but, when equipped with types, they show a hidden link between logic and computability, which is what is mainly of interest for us.

We will introduce the λ-calculus, its simplest typed version and then study some more recent developments. Our aim is to illustrate the general aspects of the theory and to derive a few results.

In many cases, we will avoid proving the results we will introduce. This is done on purpose: the simplicity of the formal system has as a natural counterpart a deep and complex technical development.

λ-calculus

Definition 6.1 (λ-term)

Fixed a set V of *variables*, which is both infinite and recursive, a *λ-term* is inductively defined as:

- any $x \in V$ is a λ-term, and $\mathrm{FV}(x) = \{x\}$;
- if M and N are λ-terms, so is $(M \cdot N)$, called *application*, and $\mathrm{FV}(MN) = \mathrm{FV}(M) \cup \mathrm{FV}(N)$;
- if $x \in V$ and M is a λ-term, so is $(\lambda x.\, M)$, called *abstraction*, and $\mathrm{FV}(\lambda x.\, M) = \mathrm{FV}(M) \setminus \{x\}$.

The set $\mathrm{FV}(M)$ is called the set of *free variables* in M, and the variables in M not occurring in $\mathrm{FV}(M)$ are said to be *bound*.

Example 6.2

$(\lambda x.x)$ is a λ-term with no free variables, representing the identity function.

λ-calculus

To simplify notation, we introduce a number of conventions:

- outermost parentheses are not written: $\lambda x.x$ instead of $(\lambda x.x)$;
- a sequence of consecutive abstraction is grouped: $\lambda x, y.x \cdot y$ instead of $\lambda x.(\lambda y.x \cdot y)$;
- we treat application as a product, omitting the dot: xy instead of $x \cdot y$;
- we assume application associates to the left: xyz instead of $(xy)z$.

Also, we use the term *combinator* to denote a λ-term having no free variables.

Example 6.3

The following are combinators

- $I \equiv \lambda x.x$;
- $K \equiv \lambda x, y.x$;
- $S \equiv \lambda x, y, z.(xz)(yz)$;
- $\Omega \equiv (\lambda x.xx)(\lambda x.xx)$.

Definition 6.4 (Substitution)

For any M, N λ-terms, and x variable, $M[N/x]$ is the *substitution* of x with N in M, defined by induction on M as:

- $x[N/x] \equiv N$;
- $y[N/x] \equiv y$, when $x \not\equiv y$;
- $(PQ)[N/x] \equiv (P[N/x])(Q[N/x])$;
- $(\lambda x.P)[N/x] \equiv \lambda x.P$;
- $(\lambda y.P)[N/x] \equiv \lambda y.P[N/x]$, when $x \not\equiv y$ and $y \notin \mathrm{FV}(N)$;
- $(\lambda y.P)[N/x] \equiv \lambda z.(P[z/y])[N/x]$, when $x \not\equiv y$ and $y \in \mathrm{FV}(N)$ and $z \notin \mathrm{FV}(P) \cup \mathrm{FV}(N)$.

In the last clause, the z variable is said to be *new*, and it is always possible to choose a z which satisfies the constraint.

The purpose of the last clause is to prevent variable capturing.

Definition 6.5 (α-equivalence)

The λ-terms M and N are α-equivalent, $M =_\alpha N$, when

- $M \equiv N$;
- $M \equiv PQ$, $N \equiv P'Q'$, and $P =_\alpha P'$ and $Q =_\alpha Q'$;
- $M \equiv \lambda x.P$, $N \equiv \lambda x.P'$, and $P =_\alpha P'$;
- $M \equiv \lambda x.P$ and $N \equiv \lambda y.P[y/x]$

So, two λ-terms are α-equivalent when they differ for the names of bound variables only.

It is immediate to see that α-equivalence is an equivalence relation, but it is also a *congruence* with respect to substitution:

Proposition 6.6

If $M =_\alpha M'$ *and* $N =_\alpha N'$, *then* $M[N/x] =_\alpha M'[N'/x]$.

Definition 6.7 (β-reduction)

The binary relation between λ-terms $M \rhd_{1,\beta} N$, *M β-reduces to N in one step*, holds if and only if $M \equiv M'[(\lambda x.P) \cdot Q/z]$ and $N \equiv M'[(P[Q/x])/z]$.
We say that *M β-reduces* to N, $M \rhd_\beta N$, when there is a finite sequence $P_1, \ldots, P_n$ such that $M \equiv P_1$, $N \equiv P_n$ and, for each $1 \leq i < n$, $P_i \rhd_{1,\beta} P_{i+1}$.

In the λ-calculus, computation is performed by β-reduction.

Definition 6.8 (β-normal form)

A term N is said to be in *β-normal form* when it does not contain any subterm of the form $(\lambda x.P)Q$.

With respect to computations, λ-terms in β-normal form represent the values.

Theorem 6.9 (Church-Rosser)

If $M \rhd_\beta P$ and $M \rhd_\beta Q$, then there is a λ-term R such that $P \rhd_\beta R$ and $Q \rhd_\beta R$.

Corollary 6.10

If $M \rhd_\beta N$ and N is a β-normal form, then N is unique up to α-equivalence.

The Church-Rosser Theorem and its corollary say that, although the computation in λ-calculus is non-deterministic, the result, when it exists, is uniquely determined.

Definition 6.11 (β-equality)

We say that P is β-equivalent to Q, $P =_\beta Q$, when there is a finite sequence $R_1, \ldots, R_n$ such that $P \equiv R_1$, $Q \equiv R_n$, and, for all $1 \leq i < n$, $R_i \rhd_{1,\beta} R_{i+1}$, or $R_{i+1} \rhd_{1,\beta} R_i$, or $R_i =_\alpha R_{i+1}$.

Theorem 6.12 (Fixed point)

There is a combinator $\mathbf{Y}$ *such that* $\mathbf{Y}x =_\beta x(\mathbf{Y}x)$.

Proof.

Let $U \equiv \lambda u, x.x(uux)$, and let $\mathbf{Y} \equiv UU$. Then
$\mathbf{Y}x \equiv (\lambda u, x.x(uux))Ux \rhd_\beta (\lambda x.x(UUx))x \rhd_\beta x(UUx) \equiv x(\mathbf{Y}x)$. □

The proof of the fixed point theorem as above, is due to Alan Turing.

The fixed point theorem says that, every λ-term, when thought of as a function, has a fixed point which is calculated by the $\mathbf{Y}$ combinator. This is an important property which suggests that each function which can be represented as a λ-term, has to be continuous.

Representable functions

Definition 6.13 (Numerals)

For every $n \in \mathbb{N}$, the *Church numeral* $\overline{n}$ is a λ-term inductively defined as:

- $\overline{0} = \lambda x, y.\, y$;
- $\overline{n+1} = \lambda x, y.\, x(\overline{n}xy)$.

Definition 6.14 (Representable functions)

Let $f : \mathbb{N}^k \to \mathbb{N}$ be a partial function. A λ-term F is said to *represent* the function f when

- for all $n_1, \ldots, n_k \in \mathbb{N}$, if $f(n_1, \ldots, n_k) = m$, then $F\overline{n_1}, \ldots, \overline{n_k} = \overline{m}$;
- for all $n_1, \ldots, n_k \in \mathbb{N}$, if $f(n_1, \ldots, n_k)$ is undefined, then $F\overline{n_1}, \ldots, \overline{n_k}$ has no β-normal form.

Theorem 6.15

Every partial recursive function can be represented in the λ-calculus.

Representable functions

The proof of the theorem is difficult beyond the aim of this course. But we will show a few examples to justify it.

Example 6.16

The successor function is represented by $\lambda x, s, z.s(xsz)$.
Addition is represented by $\lambda x, y, s, z.xs(ysz)$.
Multiplication is represented by $\lambda x, y, s.x(ys)$.
Exponentiation is represented by $\lambda x, y.yx$

Example 6.17

The Boolean values $\top$ and $\bot$ are represented as $\lambda x, y.y$ and $\lambda x, y.x$, respectively.
Then, 'if x then y else z' is represented by $\lambda x, y, z.xzy$. In fact

- if $\overline{\bot}$ then A else $B \equiv (\lambda x, y, z.xzy)(\lambda x, y.x)AB =_\beta (\lambda y, z.(\lambda x, y.x)zy)AB =_\beta (\lambda y, z.z)AB =_\beta B$, while
- if $\overline{\top}$ then A else $B \equiv (\lambda x, y, z.xzy)(\lambda x, y.y)AB =_\beta (\lambda y, z.(\lambda x, y.y)zy)AB =_\beta (\lambda y, z.y)AB =_\beta A$.

Representable functions

To get a clue why these representations work, we could read them as computations over logical structures. For example, natural numbers are inductively defined from 0 and the successor. Hence, a model for the naturals is specified when we provide a set together with a way to interpret 0 as some specific element, and the successor as an injective function which transforms an element into another.

Consider $\overline{0} \equiv \lambda x, y.\, y$: this is a function from the model which provides an element of the model. The model is specified by providing the specification of the successor and the specification of zero. The result is the specification of 0.

Consider $\overline{n+1} \equiv \lambda x, y.\, x(\overline{n}xy)$: since $\overline{n}$ transforms a model into a number, the term $\overline{n}xy$ evaluates to n in the model (x, y). But x stands for the successor function, so we are taking the successor of n in the model.

Thus, $x + y$ is calculated by interpreting x in a model where the successor function is given, but the zero element is ysz, i.e., the number which stands for y in the model.

Similarly, the product xy is calculated by interpreting x in a model where the successor function moves by y steps at once.

Simple theory of types

Definition 6.18 (Type)

Fixed a denumerable set V_T of *type variables*, a *type* is inductively defined as follows:

- $x \in V_T$ is a type;
- 0 and 1 are types;
- if α and β are types, so are $(\alpha \times \beta)$, $(\alpha + \beta)$, and $(\alpha \to \beta)$.

As usual, we omit parentheses when they are not strictly needed: $\times$ binds stronger that $+$, and $+$ binds stronger than $\to$, so $\alpha \times \beta + \gamma \to (\alpha + \gamma) \times (\beta + \gamma)$ stands for $((\alpha \times \beta) + \gamma) \to ((\alpha + \gamma) \times (\beta + \gamma))$.

A type is used to constrain the main entity of interest in the theory of types, the *term*.

Simple theory of types

Definition 6.19 (Term)

Fixed a family $\{V_\alpha\}_\alpha$ of *variables*, indexed by the collection of types, such that, for each α, V_α is denumerable and distinct from the set of type variables, and such that $V_\alpha \cap V_\beta = \emptyset$ whenever $\alpha \neq \beta$, a *term* $t : \alpha$ of type α, along with the set of its *free variables*, is inductively defined as:

- if $x \in V_\alpha$ for some type α, $x : \alpha$ is a term, and $\mathrm{FV}(x : \alpha) = \{x : \alpha\}$;
- $* : 1$ is a term and $\mathrm{FV}(* : 1) = \emptyset$;
- for each type α, $\Box_\alpha : 0 \to \alpha$ is a term and $FV(\Box_\alpha : 0 \to \alpha) = \emptyset$;
- if $A : \alpha$ and $B : \beta$ are terms, $\langle A, B\rangle : \alpha \times \beta$ is a term and $\mathrm{FV}(\langle A, B\rangle : \alpha \times \beta) = \mathrm{FV}(A : \alpha) \cup \mathrm{FV}(B : \beta)$;
- if $A : \alpha \times \beta$ is a term, so are $\pi_1 A : \alpha$ and $\pi_2 A : \beta$, and $\mathrm{FV}(\pi_1 A : \alpha) = \mathrm{FV}(\pi_2 A : \beta) = \mathrm{FV}(A : \alpha \times \beta)$;

↪

Simple theory of types

$\hookrightarrow$ (Term)

- if $A\colon \alpha$ is a term, then, for any type β, $i_1^\beta A\colon \alpha+\beta$ and $i_2^\beta A\colon \beta+\alpha$ are terms and $\mathrm{FV}(i_1^\beta A\colon \alpha+\beta) = \mathrm{FV}(i_2^\beta A\colon \beta+\alpha) = \mathrm{FV}(A\colon \alpha)$;
- if $C\colon \alpha+\beta$, $A\colon \alpha\to\gamma$, and $B\colon \beta\to\gamma$ are terms, so is $\delta(C,A,B)\colon \gamma$, and $\mathrm{FV}(\delta(C,A,B)\colon \gamma) = \mathrm{FV}(C\colon \alpha+\beta)\cup\mathrm{FV}(A\colon \alpha\to\gamma)\cup\mathrm{FV}(B\colon \beta\to\gamma)$;
- if $A\colon \beta$ is a term and $x\in V_\alpha$, then $\lambda x\colon \alpha.A\colon \alpha\to\beta$ is a term and $\mathrm{FV}(\lambda x\colon \alpha.A\colon \alpha\to\beta) = FV(A\colon \beta)\setminus\{x\colon \alpha\}$;
- if $A\colon \alpha$ and $B\colon \alpha\to\beta$ are terms, then $B\cdot A\colon \beta$ is a term and $\mathrm{FV}(B\cdot A\colon \beta) = FV(A\colon \alpha)\cup\mathrm{FV}(B\colon \alpha\to\beta)$.

Terms represent the primitive computational statements.

Simple theory of types

Terms can be *reduced* according to the following rules, where it is assumed that both sides of the equalities are correctly typed:

- $\pi_1\langle A, B\rangle = A$;
- $\pi_2\langle A, B\rangle = B$;
- $\langle \pi_1 A, \pi_2 A\rangle = A$;
- $(\lambda x\colon \alpha. B)\cdot A = B[A/x]$, the act of substituting A for x;
- $\lambda x\colon \alpha.(F\cdot x) = F$, when $x\colon \alpha \notin \mathrm{FV}(F\colon \alpha \to \beta)$;
- $\delta(i_1 C, A, B) = A\cdot C$;
- $\delta(i_2 C, A, B) = B\cdot C$.

It is clear that these rules satisfy the requirements on computable functions.

Simple theory of types

Definition 6.20 (Strongly normalisable)

A term $A\colon \alpha$ is *strongly normalisable* if every reduction sequence starting from it terminates.

Theorem 6.21 (Strong normalisation)

All the terms in the simple theory of types are strongly normalisable.

Structure of the proof by Girard:

- for each type α, inductively define a set $R(\alpha)$ of *reducible terms*;
- if $A \in R(\alpha)$, then A is strongly normalisable;
- if $A\colon \alpha$, then $A \in R(\alpha)$.

Simple theory of types

The simple theory of types can be extended with natural numbers, whose behaviour is expressed by the primitive recursion schema, obtaining Gödel's system T.

Girard's technique allows to prove that Gödel's system T is strongly normalisable. This result implies the consistency of Peano arithmetic.

Simple theory of types

If we restrict to the subsystem whose types are those generated by type variables, $\rightarrow$ and $\times$, and whose terms are, correspondingly, the variables, and those of the form $\lambda x\colon \alpha . A\colon \alpha \rightarrow \beta$, called *abstractions*, $A \cdot B\colon \beta$, called *applications*, $\langle A, B\rangle\colon \alpha \times \beta$, called *pairs*, $\pi_1 A\colon \alpha$ and $\pi_2 A\colon \beta$, called *projections*, we get a subsystem of special interest.

In fact, if we interpret $\times$ as the Cartesian product, and $\rightarrow$ as the function space, we can easily derive a representation of the natural numbers, together with the operations of addition, multiplication and exponentiation, the Boolean values, the if-then-else construction, and so on.
In fact, these representation are nothing but the same we used for the pure, non-typed λ-calculus.

Propositions as types

If we put side by side propositional logical formulae and types in the simple theory of types, we get:

types	formulae
variable	variable
0	$\bot$
1	$\top$
$\alpha \times \beta$	$\alpha \wedge \beta$
$\alpha + \beta$	$\alpha \vee \beta$
$\alpha \to \beta$	$\alpha \supset \beta$

This correspondence shows that we can translate any logical formula in a type and any type in a formula, by a one-to-one map.

Propositions as types

If we put side by side propositional proofs in the intuitionistic natural deduction system, and terms in the simple theory of types, we get:

proof	assumption	$\top I$	$\bot E$	$\wedge I$	$\wedge E_{1,2}$	$\vee I_{1,2}$	$\vee E$	$\supset I$	$\supset E$
term	variable	$*$	$\square_\alpha$	$\langle \_, \_ \rangle$	π_1, π_2	i_1^α, i_2^α	δ	λ	$\cdot$

There is an evident one-to-one correspondence, which perfectly matches the one on types.

Propositions as types

Let's examine a few examples:

- if $A\colon \alpha$ and $B\colon \beta$ are terms, so is $\langle A, B\rangle\colon \alpha \times \beta$ becomes

$$\dfrac{\begin{array}{cc}\vdots & \vdots \\ A & B \\ \alpha & \beta\end{array}}{\alpha \wedge \beta}\ \wedge\mathsf{I}$$

- if $A\colon \beta$ is a term and $x\colon \alpha$ a variable, then $\lambda x\colon \alpha.A\colon \alpha \to \beta$ becomes

$$\dfrac{\begin{array}{c}[\alpha]^{*} \\ \vdots \\ A \\ \beta\end{array}}{\alpha \supset \beta}\ \supset\mathsf{I}^{*}$$

where the label $*$ stands for x.

Propositions as types

- if $A\colon \alpha$ is a term, so is $i_1^{\beta}(A)\colon \alpha+\beta$ becomes

$$\dfrac{\begin{array}{c}\vdots\\ A\end{array}}{\alpha}\,{\vee}\mathsf{I}_2$$

Notice that $i_1^{\beta}(A)$ has two labels 1 and β which allow to keep track of the types from which it is constructed and to distinguish the two inclusions $i_1^{\beta}(A)\colon \alpha+\beta$ and $i_1^{\gamma}(A)\colon \alpha+\gamma$. This emphasizes the constructive nature of the theory of types.

Propositions as types

The correspondence illustrated so far is known as the *propositions-as-types interpretation*, and also as the *Curry-Howard isomorphism*.

At a first glance, the simple theory of types is just a way to write proofs and formulae as linear expressions instead of adopting the tree-like syntax of natural deduction.

But the logical syntax is coupled with a semantics, and the type theory with a computational meaning, given by the reduction rules.

Computations, logically

Since every formal proof in intuitionistic logic corresponds to a typed term, and typed terms are also λ-terms, each proof is a program which computes something.

It is possible to associate to each proof an object, which is an *evidence* of its type, or its conclusion, if you prefer. So, the evidence of $A \wedge B$ is a pair of evidences for A and B; the evidence of $A \vee B$ is a pair (w, e), with $w \in \{1, 2\}$ telling us which disjunct holds, and e an evidence for it; the evidence of $A \supset B$ is a function mapping any evidence of A into an evidence of B.

These evidences are the intermediate results of the computation performed by the λ-term associated to the proof. So, in a constructive system, proving a statement is, essentially, equivalent to write a computer program satisfying a specification given by the conclusion.

Proofs, computationally

Since typed terms are proofs under the correspondence, we can reduce them to a normal form. Formalising this process leads to state that every proof possesses a normal form.

Thus, considering any proof $\pi : \vdash A \vee B$, it can be reduced to a proof $\pi' : \vdash A \vee B$ in normal form, whose last step is either an instance of $\vee I_1$ or $\vee I_2$. Hence, the conclusion of the last but one step would be either A or B.

Similarly, considering any proof $\pi : \vdash \exists x : s.A$, it can be reduced to a proof $\pi' : \vdash \exists x : s.A$ in normal form, whose last step is an instance of $\exists I$. Hence, the conclusion of the last but one step would be $A[t/x]$ for some term t, providing a witness to the existential statement.

An auxiliary result

Lemma 6.22

If $\pi : \Gamma \cup \{A\} \vdash_T B$ and $\theta : \Gamma \vdash_T A$, then there is a proof $\nu : \Gamma \vdash_T B$.

Proof. (i)

By induction on the structure of the proof π.

- if π is an instance of the assumption rule either $B \in \Gamma$, so ν coincides with π, which does not depend on A, or $B \equiv A$, thus $\nu = \theta$.
- if π is an instance of the axiom rule, $B \in T$, so $\nu = \pi$, which does not depend on A.
- if π is an instance of $\top$-introduction, $B \equiv \top$, so $\nu = \pi$, which does not depend on A.
- if π is an instance of $\bot$-elimination, by induction hypothesis, there is $\xi : \Gamma \vdash_T \bot$, so applying the $\bot$-elimination rule to ξ gives the required ν.

↪

An auxiliary result

↪ Proof. (ii)

- if π is an instance of $\wedge$-introduction, $B \equiv C \wedge D$, and, by induction hypothesis, there are $\xi \colon \Gamma \vdash_T C$ and $\mu \colon \Gamma \vdash_T D$, so the required ν is obtained by applying $\wedge$-introduction to ξ and μ.
- if π is an instance of $\wedge_1$-elimination, by induction hypothesis, there is $\xi \colon \Gamma \vdash_T B \wedge C$, so ν is obtained by applying $\wedge_1$-elimination to ξ.
- if π is an instance of $\wedge_2$-elimination, by induction hypothesis, there is $\xi \colon \Gamma \vdash_T C \wedge B$, so ν is obtained by applying $\wedge_2$-elimination to ξ.

↪

An auxiliary result

↪ Proof. (iii)

- if π is an instance of $\vee_1$-introduction, then $B \equiv C \vee D$ and, by induction hypothesis, there is $\xi : \Gamma \vdash_T C$, so ν is obtained by applying $\vee_1$-introduction to ξ.
- if π is an instance of $\vee_2$-introduction, then $B \equiv C \vee D$ and, by induction hypothesis, there is $\xi : \Gamma \vdash_T D$, so ν is obtained by applying $\vee_2$-introduction to ξ.
- if π is an instance of $\vee$-elimination, by induction hypothesis, there are $\xi : \Gamma \vdash_T C \vee D$, $\mu_C : \Gamma \cup \{C\} \vdash_T B$, and $\mu_D : \Gamma \cup \{D\} \vdash_T B$, so, applying $\vee$-elimination to ξ, μ_C, and μ_D the required ν is constructed.

↪

An auxiliary result

↪ Proof. (iv)

- if π is an instance of $\supset$-introduction, then $B \equiv C \supset D$ and, by induction hypothesis, there is $\xi : \Gamma \cup \{C\} \vdash_T D$, so ν is obtained by applying $\supset$-introduction to ξ.
- if π is an instance of $\supset$-elimination, by induction hypothesis, there are $\xi : \Gamma \vdash_T C \supset B$ and $\mu : \Gamma \vdash_T C$, so ν is constructed applying $\supset$-elimination to ξ and μ.
- if π is an instance of $\neg$-introduction, $B \equiv \neg C$ and, by induction hypothesis, there is $\xi : \Gamma \cup \{C\} \vdash_T \bot$, thus ν is obtained applying $\neg$-introduction to ξ.
- if π is an instance of $\neg$-elimination, by induction hypothesis there are $\xi : \Gamma \vdash_T \neg C$ and $\mu : \Gamma \vdash_T C$, so ν is constructed applying $\neg$-elimination to ξ and μ.

An auxiliary result

↪ Proof. (v)

- if π is an instance of $\supset$-introduction, then $B \equiv C \supset D$ and, by induction hypothesis, there is $\xi : \Gamma \cup \{C\} \vdash_T D$, so ν is obtained by applying $\supset$-introduction to ξ.
- if π is an instance of $\supset$-elimination, by induction hypothesis, there are $\xi : \Gamma \vdash_T C \supset B$ and $\mu : \Gamma \vdash_T C$, so ν is constructed applying $\supset$-elimination to ξ and μ.
- if π is an instance of $\neg$-introduction, $B \equiv \neg C$ and, by induction hypothesis, there is $\xi : \Gamma \cup \{C\} \vdash_T \bot$, thus ν is obtained applying $\neg$-introduction to ξ.
- if π is an instance of $\neg$-elimination, by induction hypothesis there are $\xi : \Gamma \vdash_T \neg C$ and $\mu : \Gamma \vdash_T C$, so ν is constructed applying $\neg$-elimination to ξ and μ.
- if π is an instance of the Law of Excluded Middle, $B \equiv C \vee \neg C$, so $\nu = \pi$, which does not depend on A. □

Normalisation

The objective of normalisation is to eliminate the redundant steps in a proof, and to give it a standard format, *minimal*, in a sense.

A natural requirement for a proof in natural deduction is that no conclusion of an introduction rule must be the major premise of an elimination rule. The major premise is the formula containing as principal connective, the one which is eliminated by an elimination rule.

Also, another natural requirement is that discharged assumptions should be used in disjunction elimination, while the false elimination rule has to derive a conclusion which is not $\bot$.

Finally, although the previous requirements seem evident, they can be hidden, because of multiple subsequent elimination rules which can be permuted.

Normalisation

The *detour conversions* are deputed to eliminate detours, i.e., redundant elementary steps in a proof given by an introduction rule in the major premise of an elimination rule:

- $\wedge$ rules:

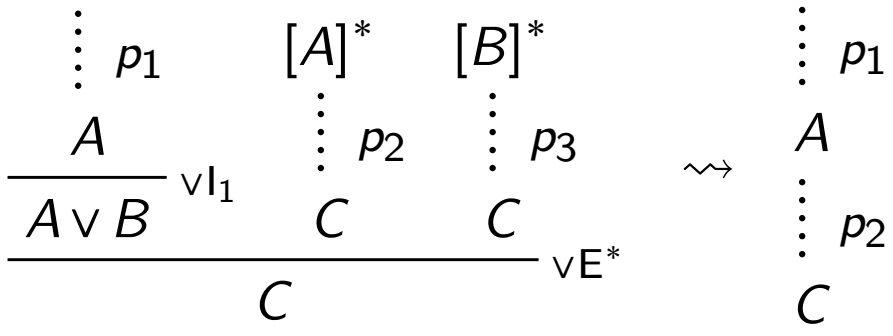
$$\dfrac{\dfrac{\begin{array}{cc}\vdots\,p_1 & \vdots\,p_2\\ A & B\end{array}}{A\wedge B}\wedge\mathrm{I}}{A}\wedge\mathrm{E}_1 \rightsquigarrow \begin{array}{c}\vdots\,p_1\\ A\end{array} \qquad \dfrac{\dfrac{\begin{array}{cc}\vdots\,p_1 & \vdots\,p_2\\ A & B\end{array}}{A\wedge B}\wedge\mathrm{I}}{B}\wedge\mathrm{E}_2 \rightsquigarrow \begin{array}{c}\vdots\,p_2\\ B\end{array}$$

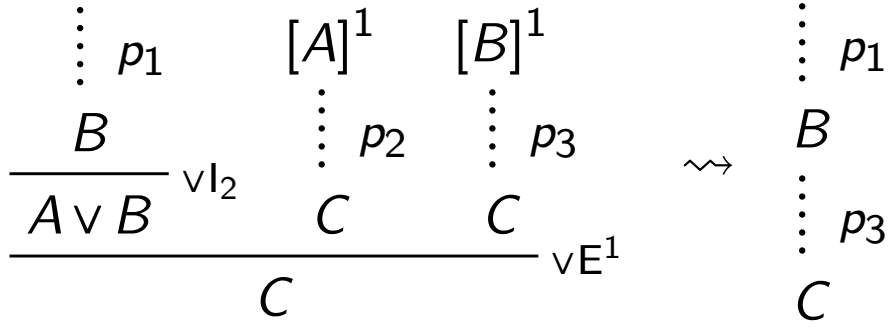
- $\supset$ rules:

$$\dfrac{\dfrac{\begin{array}{c}[A]^*\\ \vdots\,p_1\\ B\end{array}}{A\supset B}\supset\mathrm{I}^* \quad \begin{array}{c}\vdots\,p_2\\ A\end{array}}{B}\supset\mathrm{E} \rightsquigarrow \begin{array}{c}\vdots\,p_2\\ A\\ \vdots\,p_1\\ B\end{array}$$

Normalisation

- $\vee$ rules:





Normalisation

Since $\neg A \equiv A \supset \bot$, we do not need detour conversions for $\neg$ rules, as soon as we rewrite them as instances of the $\supset$ rules. The conversions for $\supset$ and $\vee$ are justified by Proposition 6.22, which allows to join proofs.

There are no detour conversions for $\bot$ and $\top$, since these connectives lack an introduction and elimination rule, respectively.

It is instructive to see these conversions through the propositions-as-types correspondence: for example, the detour conversion for $\wedge$ becomes $\pi_1\langle p_1, p_2\rangle = p_1$ and $\pi_2\langle p_1, p_2\rangle = p_2$. This observation shows how normalisation in proofs is the same as deriving a normal form for a term in the simple theory of types.

Normalisation

Detour conversions eliminate obviously redundant steps in a proof. However, there are instances of the disjunction elimination rule that are, in fact, redundant, those in which one of the discharged assumptions is not used. This fact leads to define the following *simplification conversions*: if, in

$$\dfrac{\begin{array}{ccc} & [A]^1 & [B]^1 \\ \vdots\, p_1 & \vdots\, p_2 & \vdots\, p_3 \\ A \vee B & C & C \end{array}}{C}\ \vee\mathrm{E}^1$$

either the assumption A in p_2 is not used, or the assumption B in p_3 is not used, then we can use p_2 or p_3, respectively to prove the conclusion.

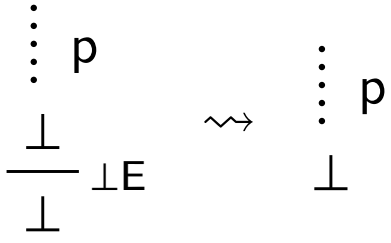
Normalisation

$$\dfrac{\begin{array}{ccccc} & & & [B]^1 & \\ \vdots\, p_1 & & \vdots\, p_2 & \vdots\, p_3 & \\ A \vee B & & C & C & \end{array}}{C}\ \vee\mathrm{E}^1 \quad \rightsquigarrow \quad \begin{array}{c} \vdots\, p_2 \\ C \end{array}$$

$$\dfrac{\begin{array}{ccccc} & & & [A]^1 & \\ \vdots\, p_1 & & \vdots\, p_2 & \vdots\, p_3 & \\ A \vee B & & C & C & \end{array}}{C}\ \vee\mathrm{E}^1 \quad \rightsquigarrow \quad \begin{array}{c} \vdots\, p_3 \\ C \end{array}$$

Normalisation

Moreover, the instances of the $\bot$ elimination rule in which the conclusion is $\bot$ are obviously redundant, and we can apply another *simplification conversion* to eliminate them.

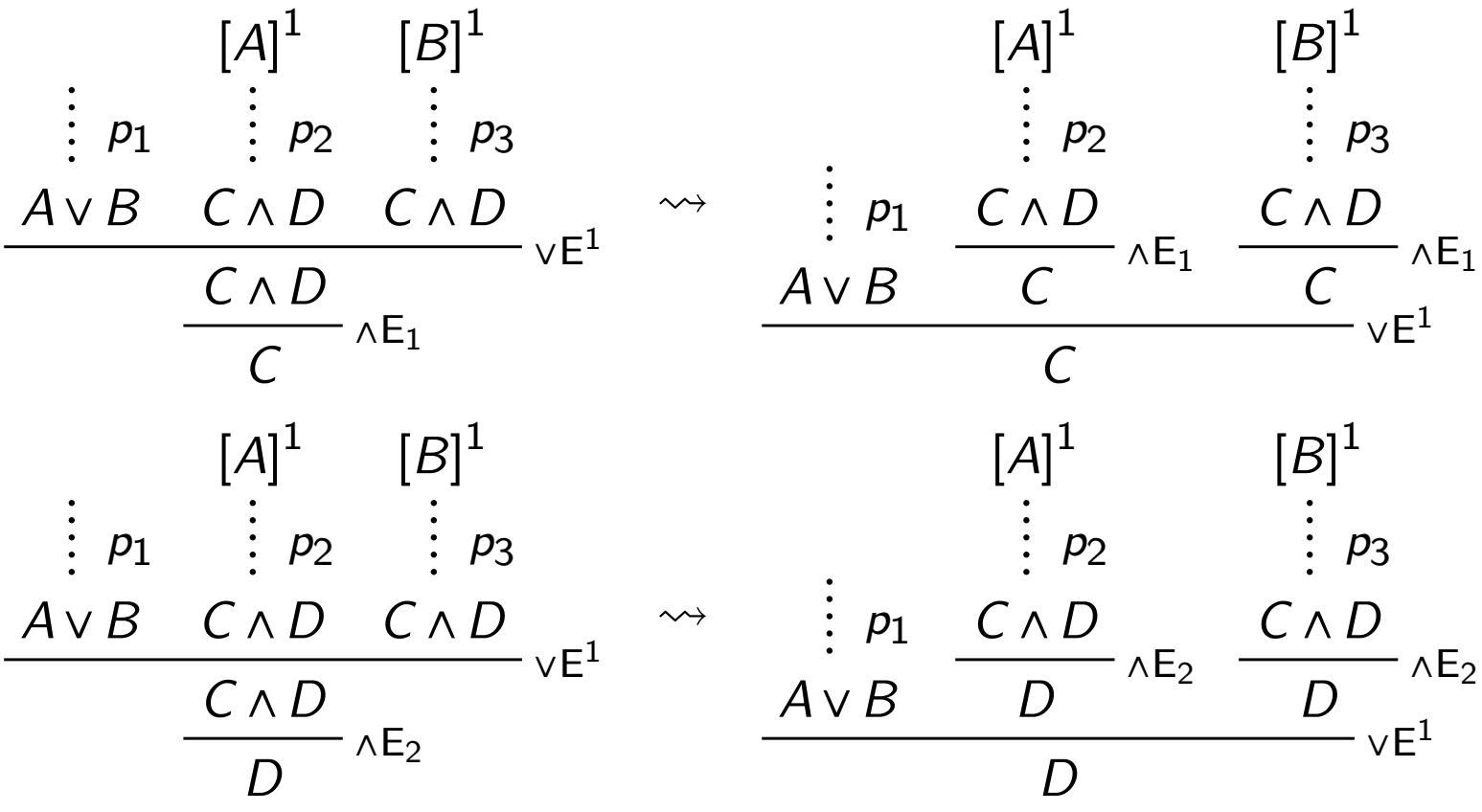


Sometimes, detours and simplifications cannot be directly applied, because they are hidden inside a proof. This happens when we apply an elimination rule whose major premise is an application of the disjunction elimination rule.

In those cases, we can move the disjunction elimination downwards, eventually revealing hidden detours and simplifications. The rules to do so are called *permutation conversions*.

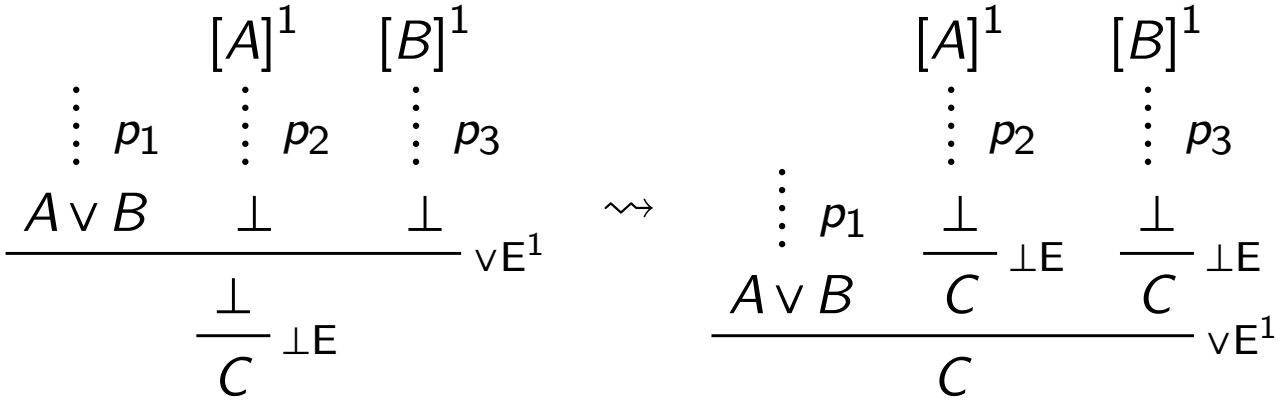
Normalisation

- $\wedge$ elimination:



Normalisation

- $\bot$ elimination:



Normalisation

- $\supset$ elimination:

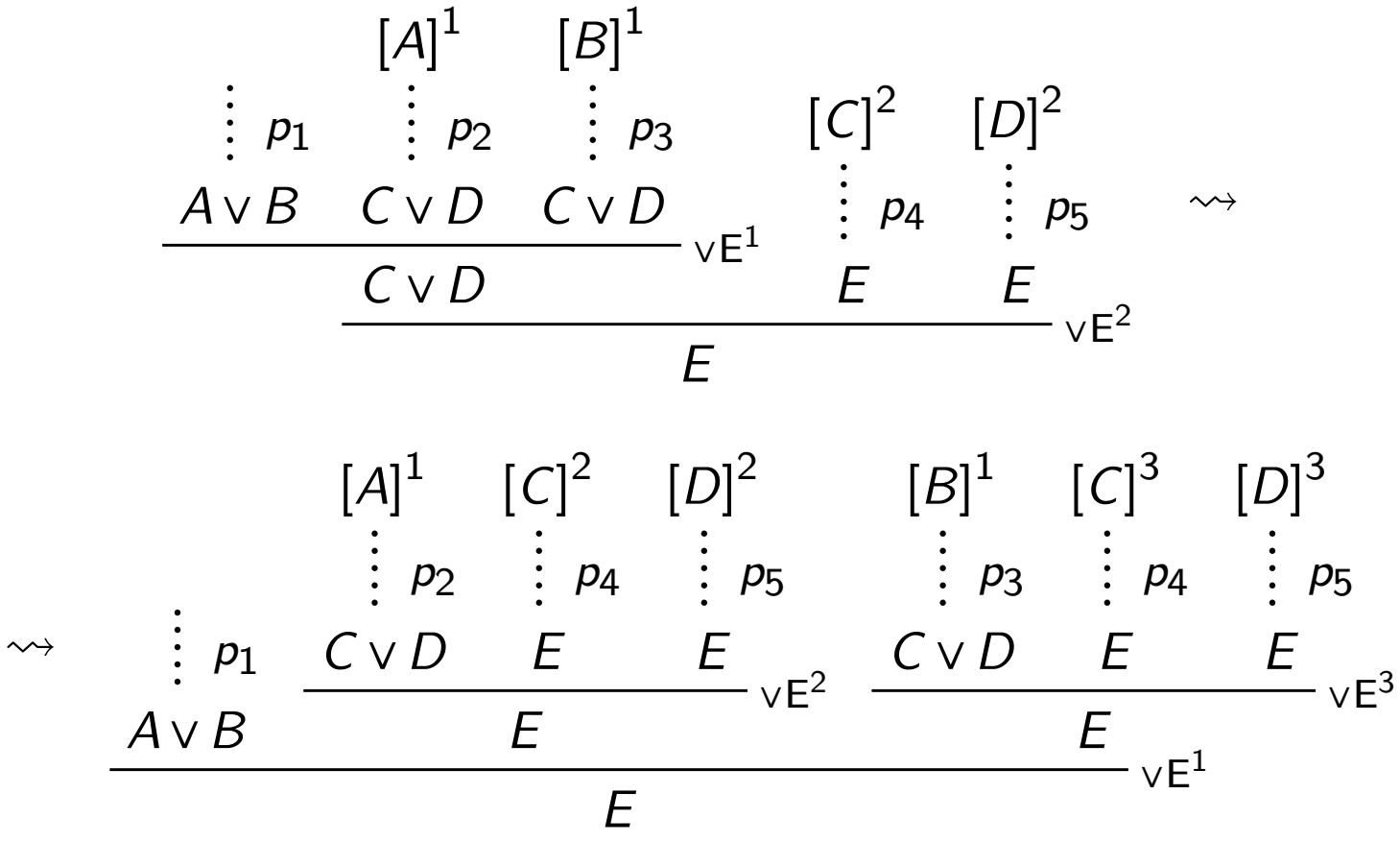
$$\dfrac{\dfrac{\begin{array}{ccc}[A]^1 & & [B]^1 \\ \vdots\, p_1 & & \vdots\, p_2 \\ A \vee B & C \supset D & C \supset D\end{array}}{C \supset D}\vee E^1 \qquad \begin{array}{c}\vdots\, p_3 \\ C\end{array}}{D}\supset E$$

$$\rightsquigarrow$$

$$\dfrac{\begin{array}{ccc}\vdots\, p_1 & \dfrac{\begin{array}{cc}[A]^1 & \\ \vdots\, p_2 & \vdots\, p_3 \\ C \supset D & C\end{array}}{D}\supset E & \dfrac{\begin{array}{cc}[B]^1 & \\ \vdots\, p_2 & \vdots\, p_3 \\ C \supset D & C\end{array}}{D}\supset E \\ A \vee B & D & D\end{array}}{D}\vee E^1$$

Normalisation

- $\vee$ elimination:



Normalisation

By applying all these conversions, mimicking the reduction process of the simple theory of types, we get the following result

Theorem 6.23 (Normalisation)

Each derivation in intuitionistic natural deduction reduces to a normal derivation, in which none of the detour, simplification and permutation conversions can be applied.

Although we are not going the see the details of the proof, since they rely on a complex double induction, we are able to derive a few consequences which are relevant.

Theorem 6.24 (Subformula property)

Let $\pi : \Gamma \vdash A$ be a normal derivation in intuitionistic propositional logic. Then each formula in π is a subformula of some formula in $\Gamma \cup \{A\}$.

Normalisation

By looking at the proof of the Normalisation Theorem,

Corollary 6.25

Let $\pi : \Gamma \vdash A$ be a normal derivation in intuitionistic propositional logic. If A is not atomic or $\bot$, then the last step is an introduction rule.

An immediate consequence is that disjunction is decidable.

Corollary 6.26 (Disjunction property)

Let $\pi : \Gamma \vdash A \vee B$ be a normal derivation in intuitionistic propositional logic. Then, there is a subproof π' of π whose conclusion is either A or B.

Similar results hold for intuitionistic first order logic, and, in particular

Corollary 6.27 (Explicit definability)

Let $\pi : \Gamma \vdash \exists x.A$ be a normal derivation in intuitionistic first order logic. Then, there is a subproof π' of π whose conclusion is either $A[t/x]$ for some term t.

Normalisation

It is important to remark that we have proved these results about normalisation in the natural deduction system for pure propositional logic. Choosing a different deductive system, although sound and complete, does not necessarily lead to the same result.

Also, adding a theory, and, thus, instances of the axiom rule may lead to alternative normalisation procedures, or to systems in which normalisation cannot be obtained.

In these cases, the constructive nature of intuitionistic logic, stemming from Corollaries 6.26 and 6.27, is not automatically achieved.
As an obvious counterexample, consider that classical logic is just intuitionistic logic plus the theory $\{A \vee \neg A \colon A \text{ formula}\}$.

Variations on the theme

The simple theory of types is just the simplest type theory: many other systems have been analysed, and many of them have a propositions-as-types interpretation, computationally characterising some logical system.

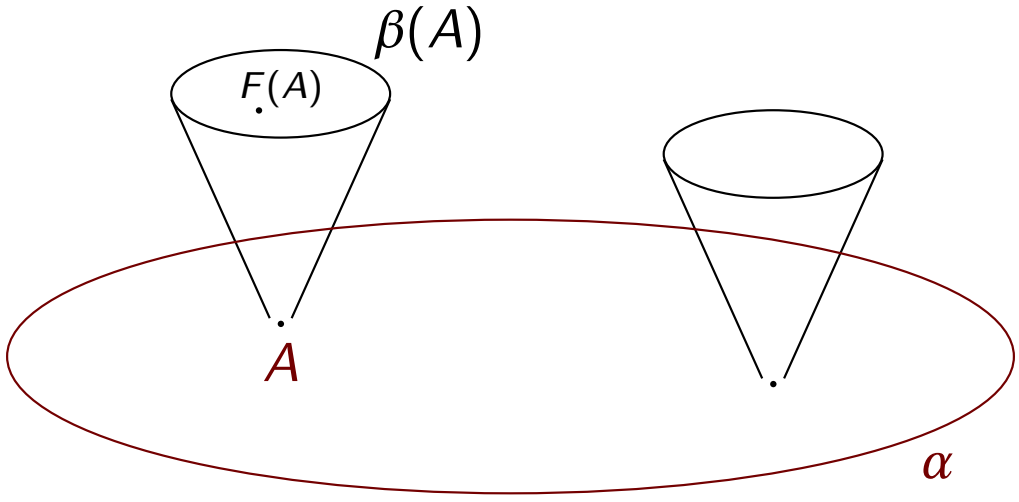
In some cases, like in the constructive type theory, the corresponding logical system is part of the type theory itself. This reflection allows to use such a system to describe mathematical theories, like set theory, inside the type system, becoming part of it. Thus, the type system acts as a *universal* theory, which contains the whole mathematics representable in its logical counterpart.

This way of proceeding has recently lead to a promising approach, which explains computation in terms of algebraic topology (and vice versa). It is called *homotopy type theory*, and it is part of the contemporary frontier of mathematical research. The basic idea is that, by adding a pair of axioms to constructive type theory, one can interpret a computation as a path in some homotopy space. It turns out that paths which are homotopy equivalent can be represented by the same term.

Martin-Löf type theory

The propositions-as-types interpretation can be extended to first order intuitionistic logic by introducing the notion of dependent types.

The concept of function $F : \alpha \to \beta$ can be extended to the case in which β has a *dependent structure*: if β is a function with domain α, a function F from α to β can be defined such that $F(A) : \beta(A)$ for each $A : \alpha$, i.e., the codomain of the function depends on the choice of the element in the domain.



The same can be done for pairs.

Martin-Löf type theory

Thus, are introduced

- $\Pi x\colon \alpha.\beta$ type of the dependent functions
- $\Sigma x\colon \alpha.\beta$ type of the dependent pairs

such that, if β does not depend on $A\colon \alpha$, then

- $\Pi x\colon \alpha.\beta = \alpha \to \beta$;
- $\Sigma x\colon \alpha.\beta = \alpha \times \beta$.

This allows to extend the propositions-as-types correspondence as

types	formulae
$\Pi x\colon \alpha.\beta$	$\forall x \in \alpha.\beta$
$\Sigma x\colon \alpha.\beta$	$\exists x \in \alpha.\beta$

Example 6.28
A dependent pair is

$$\Sigma_{n:\,\mathbb{N}}\mathbb{N}^n$$

containing all the n-tuples of naturals for each $n:\mathbb{N}$.

Example 6.29
An example of term of dependent product is

$$\mathsf{zeros}:\,(\Pi_{n:\,\mathbb{N}}\mathbb{N}^n)$$

whose terms are such that $\mathsf{zeros}\,n$ is the n-component vector $(0,\ldots,0)$.

Martin-Löf type theory

Martin-Löf type theory consists in the simple theory of types extended with Π, Σ, Natural numbers and an equality type.

Naturals are defined inductively:

- $\mathbb{N}$ is a type;
- $0 : \mathbb{N}$;
- if $n : \mathbb{N}$, then $\mathrm{succ}(n) : \mathbb{N}$.

Given $A : \alpha$ and $B : \alpha$, is defined the propositional equality type $A =_\alpha B$ by introducing the term $\mathrm{refl}_A : A =_\alpha A$, stating that A is equal to itself, i.e., that quality is reflexive.
The idea behind propositional equality is that if there is a term of type $A =_\alpha B$ then A and B are equal. In general $A =_\alpha B$ is empty, but nothing prevents that an equality type contains terms other than refl.

Martin-Löf type theory

An important feature of Martin-Löf type theory is the notion of *induction*, which generalises the one on natural numbers: to prove a statement on all naturals, it suffices to prove it for 0 and succ(n), provided that it holds for n.

The same idea lies behind the δ term in the simple theory of types: to prove a statement about all the terms of a certain type, it is enough to prove it for the canonical terms.

For example, in the case of $+$, to prove that a type γ depending on $\alpha + \beta$ is inhabited it is enough to show that we can construct a term of γ from

- $i_1^{\beta} A \colon \alpha + \beta$ and
- $i_2^{\alpha} B \colon \alpha + \beta$.

This allows, for example, to prove that propositional equality is symmetric

Proposition 6.30

For every type α and every $A, B : \alpha$ there is a function $(A =_\alpha B) \to (B =_\alpha A)$ mapping $P : A =_\alpha B$ to $P^{-1} : B =_\alpha A$ such that $\mathrm{refl}_A^{-1} \equiv \mathrm{refl}_A$ for each $A : \alpha$.

Proof.

Given $A, B : \alpha$ and $P : A =_\alpha B$ we want to construct a term $P^{-1} : B =_\alpha A$. By induction, it is enough to do this in the trivial case in which B is A and P is refl_A; hence, P^{-1} has to be of type $A =_\alpha A$. Thus, we can just define refl_A^{-1} as refl_A, and the general case follows by induction. □

Propositional equality is also transitive, and those operations are well behaved

Proposition 6.31

For every type α and every $A, B, C : \alpha$ there is a function $(A =_\alpha B) \to (B =_\alpha C) \to (A =_\alpha C)$ mapping $P : (A =_\alpha B)$ to $Q : (B =_\alpha C)$ to $P \cdot Q : (A =_\alpha C)$ such that $\mathsf{refl}_A \cdot \mathsf{refl}_A \equiv \mathsf{refl}_A$ for each $A : \alpha$.

Proposition 6.32

Let α be a type, $A, B, C, D : \alpha$, $P : (A =_\alpha B)$, $Q : (B =_\alpha C)$ and $R : (C =_\alpha D)$. Then

- $P = P \cdot \mathsf{refl}_B$ *and* $P = \mathsf{refl}_A \cdot P$;
- $P^{-1} \cdot P = \mathsf{refl}_B$ *and* $P \cdot P^{-1} = \mathsf{refl}_A$;
- $(P^{-1})^{-1} = P$;
- $P \cdot (Q \cdot R) = (P \cdot Q) \cdot R$.

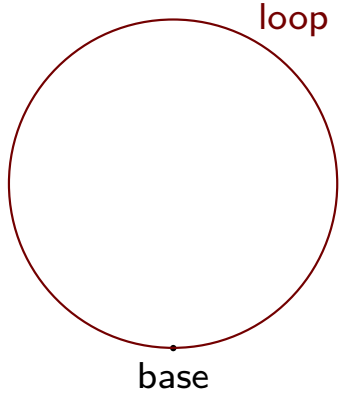
Homotopy type theory

An insight by Voevodsky and the other mathematicians behind Homotopy type theory allowed to easily formalise some complex mathematical objects using type theory.

A term P of a propositional equality type $A =_\alpha B$ can be seen as a *path* from A to B.

This allows, among the other things, to construct some *topological spaces*.

For example, the circle is constructed as the type $\mathbb{S}^1$ such that base: $\mathbb{S}^1$ and loop: base $=_{\mathbb{S}^1}$ base.
Hence the circle is constructed by a point, base, and a path from base to base, called loop.
The paths loop^{-1}, loop^3 etc can be obtained by inverting and composing loop.



Homotopy type theory

For each type α, a type $||\alpha||$ can be constructed such that

- for any $A : \alpha$ there is $|A| : ||\alpha||$;
- for any $x, y : ||\alpha||$, $x =_{||\alpha||} y$.

This means that all the inhabitants of the type $||\alpha||$ are equal or, through the *propositions-as-types* correspondence, that all the proofs of the proposition $||\alpha||$ can be identified. It is a way to translate intuitionistic logic into classical logic.

Indeed, while it can be proved that there is a type α such that $\alpha + \neg\alpha$ and $\neg\neg\alpha \to \alpha$ are not inhabited, this does not hold for truncated types.

References

A classical, and still excellent introduction to λ-calculus and the simple theory of types is *J.R. Hindley* and *J.P. Seldin*, Lambda-Calculus and Combinators, Cambridge University Press, (2008), ISBN 978-0-521-89885-0.

The link between logical systems, their semantics, and the simple theory of types is illustrated in *P. Johnstone*, Sketches of an Elephant: A Topos Theory Compendium, two volumes, Oxford University Press (2002), ISBN 978-0-19-853425-9 and 978-0-19-851598-2.

The link between λ-calculus, continuous functions, and topological spaces is explained in the fundamental paper *D. Scott*, Continuous lattices, in F.W. Lawvere ed., Toposes, Algebraic Geometry and Logic: Dalhousie University, Halifax, January 16–19, 1971, pp. 97–136, Springer (1972), ISBN 978-3-540-37609-5.

The normalisation result for the simple theory of types is illustrated in *J. Girard, Y. Lafont, P. Taylor* Proofs and Types, Cambridge Tracts in Theoretical Computer Science (1989).

References

The propositions-as-types interpretation and the normalisation theorem are illustrated in many textbooks: the lesson has been adapted from *A.S. Troelstra* and *H. Schwichtenberg*, Basic Proof Theory, Cambridge Tracts in Theoretical Computer Science 43, Cambridge: Cambridge University Press, (1996).

An analysis of normalisation can be found in *S. Negri* and *J. Von Plato*, Structural Proof Theory, Cambridge University Press (2001).

A more computer science oriented text is *B.C. Pierce*, Types and Programming Languages, The MIT Press, (2002), ISBN 978-0262-16209-8.

Homotopy Type Theory is introduced in *The Univalent Foundation Program*, Homotopy Type Theory, Institute of Advanced Studies, Princeton, (2013). The chapter on type theory is also a very nice introduction to the theory of dependent types.

Exercises

1. Prove in intuitionistic propositional logic that $(a \supset (b \supset c)) \supset (a \wedge b \supset c)$ and translate this proof into a term in the simple theory of types.
2. Show that the Axiom of Choice implies the Law of Excluded Middle [hint: let P be a proposition, and let x be a variable not appearing in P. Define $U = \{x \in \{0,1\} : P \vee (x = 0)\}$ and $V = \{x \in \{0,1\} : P \vee (x = 1)\}$. There must be a choice function on $\{U, V\}$, hence...].
3. Prove in Peano arithmetic that $\forall x.(x = 0) \vee (\exists y.x = Sx)$.
4. Show that if g and $h_o, \ldots, h_k$ are representable in Peano arithmetic, so is f obtained by substitution.
5. Show that if g is representable in Peano arithmetic, so is f obtained by minimalisation.
6. β-reduce $(\lambda x.xxy)(\lambda x.xxy)$, $(\lambda x.(\lambda y.yx)z)$, and the combinator Ω.
7. Show that the reminder function is primitive recursive.
8. Show that if $f(m_1, \ldots, m_k, n)$ is primitive recursive, then the bounded sum $h(m_1, \ldots, m_k, p) = \Sigma_{n \leq p} f(m_1, \ldots, m_k, n)$ is primitive recursive.