Truth and provability

Herman Ruge Jervell

December 11, 2007

Contents

0.1 Incompleteness 7
0.2 The pioneers 8

1 Data structures and syntax **9**
1.1 Skolems paper 9
1.2 Pairs 10
1.3 The language of pairs 10
1.4 Representing syntax by pairs 11
1.5 Natural numbers 13

2 Elementary theories of syntax **17**
2.1 The elementary theories 17
2.2 The minimal theory 17
2.3 Robinson theory 19
2.4 Non standard models 20
2.5 Induction 21
2.6 Skolem theory 21

3 Decidable theories **23**
3.1 Decidable versus undecidable 23
3.2 Automata theory 23
3.3 Logic of acceptance 25
3.4 Presburger arithmetic 25
3.5 Arithmetic with only multiplication 26
3.6 Numbers with addition and multiplication 26

4 Diagonalization **27**
4.1 Cantors argument 27
4.2 Russells paradox 28
4.3 Fix point of functions 29
4.4 Diagonal lemma 29
4.5 Self reference 30
4.6 Currys paradox 32

5 Provability **33**
5.1 Representing syntax . . . 33
5.2 Provability as necessity . . . 34
5.3 Tarskis theorem . . . 37
5.4 Consistency . . . 37
5.5 Gödels first incompleteness theorem . . . 38
5.6 Gödels second incompleteness theorem . . . 38

6 Modality **41**
6.1 The Gödel-Löb modal logic . . . 41
6.2 Solovays first completeness theorem . . . 45
6.3 Solovays second completeness teorem . . . 47

7 Direct proofs **51**
7.1 Auxiliary constructions . . . 51
7.2 Sequential calculus . . . 51
7.3 Completeness of the cut free rules . . . 54
7.4 Cut elimination . . . 56
7.5 Notations . . . 61
7.6 Infinitary proofs . . . 63
7.7 Direct proofs in syntactic theories . . . 64

8 Ordinals **65**
8.1 Well Orderings . . . 65
8.2 Arithmetical operations . . . 65
8.3 Transfinite induction . . . 66
8.4 Ordinal bounds . . . 69
8.5 Bounds for provable transfinite induction . . . 72

9 Quantifiers **73**
9.1 Skolem . . . 73
9.2 Gödel . . . 74
9.3 Interpretability . . . 76
9.4 Discussion . . . 78
9.5 Skolem functions . . . 79
9.6 Spector . . . 81
9.7 ϵ-calculus . . . 83

10 Well orderings **85**
10.1 β-logic . . . 85
10.2 β-completeness . . . 86
10.3 Discussion . . . 90

11 Finite trees as ordinals **93**
11.1 Ordering trees 93
11.2 Approximating trees 95
11.3 Trees are well ordered 99
11.4 Binary trees 100
11.5 Finding ordinals of trees 102

12 Labeled trees **105**
12.1 Extending finite trees 105
12.2 Gaps 108
12.3 Well ordering of finite labeled trees 108

13 Universes **113**
13.1 Data structures 113
13.2 Second order logic 114
13.3 Monotone operators 114
13.4 Fix point universes 115
13.5 Inductive universes 116

Preface

0.1 Incompleteness

There are two aspects of formal systems . We can view the formal system as a language and we can view it as a calculus. In the language part we build sentences from names of constants, from names of relations using connectives, quantifiers and other things. Often this language has a standard interpretation — for example it may talk about natural numbers, addition and multiplication — and it may make sense to talk about a sentence being true. On the other hand we may have a calculus — for example given as an axiom system — describing the provable sentences. We may hope that we in this way captures all true sentences. But there is often a tension between logic as language and logic as calculus. We may find facts expressed in the language which we know are true, but the calculus is not able to give a reason why it should be so.

The main focus of our investigation will be the tension between the true and the provable. It is often described as the phenomenon of incompleteness. Some high points here are:

- Gödels incompleteness theorems
- Provability logic
- Gentzens treatment of incompleteness in elementary arithmetic
- Ordinal notations
- Connections to computability and complexity

At first sight the incompleteness phenomenon may seem trivial. Why should we not be able to express things which cannot be proved? The interest comes up in the more precise demarcation between the provable and true and the not provable but still true.

But there is a long way before coming to this. We must be more precise with what we mean by a language, by truth and by provable. Then we look at how we could analyze the provability within certain formal systems in ways which are of interest in treating the demarcation between what can be proved and not proved within a system.

0.2 The pioneers

The real background of this monograph comes from the treatment by the pioneers — that is the works of

- Gottlob Frege (1848 - 1925)
- David Hilbert (1862 - 1943)
- Thoralf Skolem (1887 - 1963)
- Emil Post (1897 - 1954)
- Kurt Gödel (1906 - 1978)
- Gerhard Gentzen (1909 - 1945)
- Alan Turing (1912 - 1954)
- Stephen Kleene (1909 - 1994)
- Kurt Schütte (1909 - 1998)
- Gaisi Takeuti (1926 -)

Look up some of their papers — they are easily available — and read them. As a starting point you have the excellent collections [14], [4] and [2].

Chapter 1

Data structures and syntax

1.1 Skolems paper

Thoralf Skolem wrote under first world war and published in 1923 the paper *Begründung der elementären Arithmetik durch die rekurrierende Denkweise ohne Anwendung scheinbarer Veränderlichen mit unendlichem Ausdehnungsbereich* [10]. He had been reading Russell and Whiteheads *Principia Mathematica* and did not like it. In particular he thought it hopeless to use a long and rather involved proof of the statement $1 + 1 = 2$. Instead he introduced

- The unary numbers built up from 1 and successor.
- A language built on the data structure.
- A formal system built for proving statements in the language.

or in more modern terms

- unary numbers as a data structures
- primitive recursive functions as a programming language
- primitive recursive arithmetic as a programming logic

So he defined a universe — the unary numbers as built up from 1 and successor. On that universe he defined a language given by what we now know as the primitive recursive functions, and a logical system to show properties of the primitive recursive functions. The argument for $1+1 = 2$ is straightforward and reflects how a mathematician would think about it. We can think of Skolems paper as giving a treatment of natural numbers using three similar levels:

Universe: Built up from a starting point with one constructor.

Language: Built up from atomic pieces with a number of constructors.

Calculus: Built up from axioms using syntactical rules.

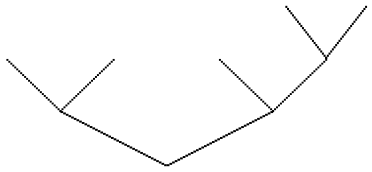
The tension between logic as a language and logic as a calculus has followed logic through history. With Skolems paper we have the means to approach the problem in an interesting way — a way leading through the works of Kurt Gödel, Gerhard Gentzen and Alan Turing in the 1930 up to present work in proof theory and theoretical computer science.

1.2 Pairs

The experience with programming languages like Lisp and Prolog tells us that a better data structure than unary numbers is the data structure of pairs. A pair is an element built up from

- the atom **nil**
- the binary constructor often called **cons** or written as $(x \cdot y)$

A typical pair can be visualized as a binary tree



On the top nodes we put the element **nil**. The data structure can be described by :

$$\mathcal{P} ::= \mathbf{nil} \mid (\mathcal{P} \cdot \mathcal{P})$$

The binary tree above can be written as:

$$((\mathbf{nil} \cdot \mathbf{nil}) \cdot (\mathbf{nil} \cdot (\mathbf{nil} \cdot \mathbf{nil})))$$

Connected to the data structure is a partial order $x \prec y$ meaning x is constructed before y.

1.3 The language of pairs

The language connected to the data structure is given by:

Constant: nil

Variables over pairs: $x, y, z, \ldots$

Binary function: $(\cdot)$

Binary relations: $\prec\ =$

Connectives: As usual $\neg, \wedge, \vee, \rightarrow$

Bounded quantifiers: $\forall x \prec y\ \exists x \prec y$

Quantifiers: $\forall x\ \exists x$

with formulas built up in the usual way. Note that we have singled out the bounded quantifiers. As usual we let $\mathbf{\Delta}_0$ be the formulas built up by only using connectives and bounded quantifiers. We have also the higher formula classes defined by

- $\mathbf{\Pi}_0 = \mathbf{\Sigma}_0 = \mathbf{\Delta}_0$
- $\mathbf{\Pi}_{n+1}$: $\forall$-quantifiers outside a $\mathbf{\Sigma}_n$ formula
- $\mathbf{\Sigma}_{n+1}$: $\exists$-quantifiers outside a $\mathbf{\Pi}_n$ formula

It is decidable whether a $\mathbf{\Delta}_0$-sentence (without free variables) is true or false. It is sufficient to look at sentences built up from literals (atomic or negation of atomic) using conjunction, disjunction and bounded quantifiers. We have

$$\begin{array}{lcl} \mathfrak{T} : F \wedge G & \Leftrightarrow & \mathfrak{T} : F \text{ and } \mathfrak{T} : G \\ \mathfrak{T} : F \vee G & \Leftrightarrow & \mathfrak{T} : F \text{ or } \mathfrak{T} : G \\ \mathfrak{T} : \forall x < t & \Leftrightarrow & \text{For all } x < t\ \mathfrak{T} : Fx \\ \mathfrak{T} : \exists x < t & \Leftrightarrow & \text{There exists } x < t\ \mathfrak{T} : Fx \end{array}$$

where we read $\mathfrak{T} : F$ as "F being true". The idea is to analyze sentences (no free variables) using the equivalences above. Note that in the analysis of a sentence (no free variables) we get from sentences to (shorter) sentences and the "for all $x < t$" and the "exists $x < t$" involves a finite number of cases. We write down the analysis as an AND-OR-tree with sentences at the nodes and literals at the topmost nodes. The branchings is bounded by the terms and the length of each branch is bounded by the formulas. By checking the analysis in the tree left first / depth first we get a decision procedure in polynomial space.

1.4 Representing syntax by pairs

Let us now look at the $\mathbf{\Delta}_0$-formulas as a language. As a start we introduce:

$$a \preceq b \leftrightarrow (a = b \vee a \prec b)$$

The pair is divided up into a head and a tail. We define:

Pair(x) **:** $\neg x = \mathbf{nil}$

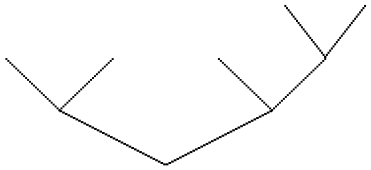
Head(x, y) **:** $\exists z \prec x.x = (y \cdot z)$

Tail(x, z) **:** $\exists y \prec x.x = (y \cdot z)$

The head of x **:** the element $y = \mathbf{hd}(x)$ such that **Head**(x, y)

The tail of x **:** the element $z = \mathbf{tl}(x)$ such that **Tail**(x, z)

Here we have used definite description — they can be eliminated in context in the usual way. We want to use pairs to represent syntactic elements. Then it is essential that we can pack in and pack out pieces of information. Let us look at the pair **P** that we have already looked at:



We can look at it as containing the three subtrees

- $\mathbf{hd}(\mathbf{P})$
- $\mathbf{hd}(\mathbf{tl}(\mathbf{P}))$
- $\mathbf{hd}(\mathbf{tl}(\mathbf{tl}(\mathbf{P})))$

In this way we have a simple way of packing in pieces of information.

But we can do more than that. We get to the more involved description by using bounded quantifiers. The following example show how this can be done.

Let us call a subtree for a tail element if it can be obtained by using **tl** one or more times. That y is a tail element of x can be defined by:

$$y \prec x \wedge \forall z \preceq x.(y \prec z \rightarrow y \preceq \mathbf{tl}(z))$$

and using standard techniques we can write it as a $\mathbf{\Delta}_0$-formula.

Exercise 1.4.1 *You can use the following as a representation of the natural numbers in* $\mathcal{P}$*:*

- $\underline{0} = (\mathbf{nil}, \mathbf{nil})$
- $\underline{n+1} = (\mathbf{nil}, \underline{n})$

Show that you can define that something is a natural number as a $\mathbf{\Delta}_0$*-formula in* $\mathcal{P}$*.*

Exercise 1.4.2 *Give a* $\mathbf{\Delta}_0$*-representation of finite sequences of natural numbers. Show that we have a* Δ_0*-representation of*

- *the length of a finite sequence*
- *the n-th item of a sequence*

This is as complicated coding as we need to consider. We claim

Background 1 *A property or an operation is syntactical if and only if it can be represented as a Δ_0-formula over $\mathcal{P}$.*

In this monograph we shall not go into much more details with actual coding. When we refer to a property as syntactical we claim that it can be represented as a $\mathbf{\Delta}_0$-formula in the language of pairs, and leave it to the reader to verify that it is actually so.

It may be useful to think of formulas from the higher formula classes in the following way:

Invariant property: A $\mathbf{\Pi}_1$-formula expressing that for every input a syntactical property holds

Provability property: A $\mathbf{\Sigma}_1$-formula which says that there exists a proof which satisfies some syntactical property.

Specification property: A $\mathbf{\Pi}_2$-formula expressing that for every input there is an output satisfying some syntactical relation.

1.5 Natural numbers

We have chosen the data structure of pairs. The pioneers — Skolem and Gödel — used unary numbers. Then the language is not rich enough to express syntax. There have been the following ways around it:

- Add to the language addition, multiplication and exponentiation. Then we can use the prime number decomposition of numbers to pack in / pack out information. The formulas expressing syntax are $\mathbf{\Delta}_0$.
- Add to the language addition and multiplication. Then we use $\mathbf{\Sigma}_1$-formulas to express syntax.

The first way is straightforward. Let us look at the second which was Gödels way. Here we need the Chinese remainder theorem from number theory to do the coding.

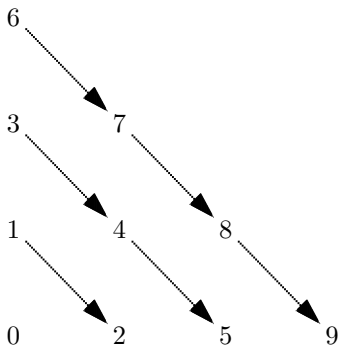
Let us start with unary numbers, addition and multiplication. A standard way to represent a pair of two numbers is

$$\langle x, y\rangle = x + \frac{(x+y+1)(x+y)}{2} = x + \Sigma_{i=0}^{x+y} i$$

or

$$z = \langle x, y \rangle \Leftrightarrow 2z = 2x + (x + y)(x + y + 1)$$

The function represent the pairs as the points in the plane enumerated as follows



But we can still not represent syntax. We get into problems where we represent properties like ´´being a tail element" which we treated above in the setting of pairs. For Gödel the way around was to use the Chinese remainder theorem.

Let us start with a representation of all pairs $\langle x, y \rangle$ of natural numbers with $x < 3$ and $y < 5$. Since there are 15 such pairs we need at least 15 distict numbers to represent them. It turns out that the numbers from 0 to 14 suffice. This can be seen from the following table:

Number	Remainder modulo 3	Remainder modulo 5
0	0	0
1	1	1
2	2	2
3	0	3
4	1	4
5	2	0
6	0	1
7	1	2
8	2	3
9	0	4
10	1	0
11	2	1
12	0	2
13	1	3
14	2	4

As an example we let the number 8 represent the pair $\langle 2, 3 \rangle$. From number theory we know that it is essential that the numbers 3 and 5 are relatively prime. This observation generalizes to arbitrary long sequences

Theorem 1 (Chinese remainder theorem) *Suppose that we have numbers $d_0, \ldots, d_{n-1}$ which are relatively prime. For a number x let r_i be the remainder of x modulo d_i. We then let x code the finite sequence $\langle r_0, \ldots, r_{n-1}\rangle$. Then the coding function*

$$x \mapsto \langle r_0, \ldots, r_{n-1}\rangle$$

from numbers $< d_0 \times \cdots \times d_{n-1}$ to sequences in $d_0 \times \cdots \times d_{n-1}$ is bijective (i.e. 1-1 and onto).

Proof: Let us first show that the function is 1-1. Assume we have two numbers $x \leq y < d_0 \times \cdots \times d_{n-1}$ with the same code $\langle r_0, \ldots, r_{n-1}\rangle$. But then all d_i will divide the difference $y - x$. Since the d_i are relatively prime then the product $d_0 \times \cdots \times d_{n-1}$ will also divide $y - x$. But $0 \leq y - x < d_0 \times \cdots \times d_{n-1}$ which shows that $0 = y - x$ and hence $x = y$.

So the coding function is 1-1. But both the domain and the range of the function are finite and contain the same number of elements. Therefore the coding function is also onto. ■

We can therefore code long sequences of numbers if we have long lists of relatively prime numbers. We now use

Theorem 2 *Let $n > 0$ and $d = n! =$ the factorial of n. Then the numbers $1 + d, 1 + 2d, \ldots, 1 + (n + 1)d$ are relatively prime.*

Proof: Consider $1 + id$ and $1 + jd$ with $1 \leq i \leq j \leq n + 1$. Let p be a prime number dividing both numbers. Then $p > n$ and p divides the difference $(j - i)d$. Since p is a prime $> n$, then p cannot divide $d = n!$. Hence p divides $j - i$. But $0 \leq j - i < n$ and hence $i = j$. ■

This gives us the famous β-function of Gödel.

$$\beta(c, d, i) = \text{the remainder of } c \text{ modulo } 1 + (i + 1)d$$

Theorem 3 *Let $a_0, \ldots, a_{n-1}$ be a sequence of natural numbers. Then there are c and d such that for all $i = 0, 1, \ldots, n - 1$ we have*

$$\beta(c, d, i) = a_i$$

So we get a $\mathbf{\Sigma}_1$-formula representing that something is code for a sequence. We need the extra quantifier to get large enough numbers to feed into the β-function. We used factorials in getting large enough numbers and hence went beyond the addition and multiplication used in the language. These large numbers are hidden by the extra $\exists$-quantifiers.

But for the language of pairs it is much simpler.

Chapter 2

Elementary theories of syntax

2.1 The elementary theories

We work within the language of pairs $\mathcal{P}$. Our theories are done in classical predicate logic with equality and in addition some axioms connected with the pairs. We consider mainly the following:

The minimal theory MIN: As axioms we take all true $\mathbf{\Delta}_0$ sentences.

Predicative induction $\mathbf{\Sigma}_1$-IND: In addition to **MIN** we have induction over $\mathbf{\Sigma}_1$-formulas which may contain free variables.

Full induction IND: This corresponds to Peano arithmetic.

There are two other theories we look at:

Robinson theory: This is a slight extension of **MIN** which have a finite number of axioms.

Skolem theory: This is an extension **MIN** by Δ_0-induction where in the language we allow names for terms for functions and predicates defined by primitive recursion over pairs. It turns out that $\mathbf{\Sigma}_1$-induction is derivable in the theory.

If we do not mention otherwise, all our theories extend **MIN** , have as a model the standard model of pairs and it is partially computable whether some formula is provable in the theory.

2.2 The minimal theory

We know that the truth or falsity of $\mathbf{\Delta}_0$-sentences can be decided by a PSPACE-algorithm. We do not here go down to weaker theories of pairs. It is remarkable

that the incompleteness phenomena starts already here. The short reason for this is as follows:

- That x is a terminating computation of program p is a syntactical property and can hence be expressed as a $\mathbf{\Delta}_0$-formula.
- That program p terminates can be expressed as a $\mathbf{\Sigma}_1$-formula $F(p)$
- Given a theory $\mathbf{T}$ extending **MIN** which is consistent and where derivability is partially computable.
- If program p terminates, then $F(p)$ is derivable in $\mathbf{T}$.
- If $\neg F(p)$ is derivable in $\mathbf{T}$, then program p does not terminate.
- It is not decidable whether program p terminates — by the halting problem from computability theory.
- There must be a program q which does not terminate and where we are not able to prove $\neg F(q)$
- For any such theory $\mathbf{T}$, there are true Π_1-sentences which cannot be derived in $\mathbf{T}$.

We now flesh out the argument. Assume we have a theory $\mathbf{T}$ as above. Since $\mathbf{T}$ extends **MIN** then all true Δ_0-sentences are provable in $\mathbf{T}$. We write this as

$$\mathbf{\Delta}_0 - \mathbf{TRUE} \Rightarrow \mathbf{\Delta}_0 - \mathbf{PROVABLE}$$

Since $\mathbf{T}$ is also consistent, then

$$\mathbf{\Delta}_0 - \mathbf{TRUE} \Leftarrow \mathbf{\Delta}_0 - \mathbf{PROVABLE}$$

A short argument gives further

$$\mathbf{\Sigma}_1 - \mathbf{TRUE} \Rightarrow \mathbf{\Sigma}_1 - \mathbf{PROVABLE}$$

For assume we have $\exists x.F(x)$ true with $F(x)$ a $\mathbf{\Delta}_0$-formula. Then there exists p with $F(p)$ true. By the above $F(p)$ is provable. By ordinary predicate logic $\exists x.F(x)$ is provable.

These implications are true for all such appropriate $\mathbf{T}$. For theories $\mathbf{T}$ where derivability is partially computable we do not have the converse for $\mathbf{\Sigma}_1$-sentences. If we had, then we could have made a decision procedure for the halting problem. A strengthening of such a theory $\mathbf{T}$ can be thought of as a way of making more true $\mathbf{\Pi}_1$-sentences provable. But we can never make all of them provable without going beyond consistency or partial computability. This could have been called the zeroth incompleteness theorem. There are some problems:

- we refer to truth/falsity of sentences
- we take consistency as being true in a model

Instead we want to have the incompleteness as a theorem only involving syntax. This is later done by defining

- a theory is incomplete if there are sentence F such that neither F nor $\neg F$ were provable
- a theory is consistent if there is a sentence which is not provable

We then get to the real incompleteness theorem by Gödel. Much more about them later.

2.3 Robinson theory

The theory **MIN** has an infinite number of axioms. By strengthening the theory slightly we can make do with a finite number of axioms. Robinson theory is the universal closure of the following five axioms:

R1: $\neg \mathbf{nil} = (x \cdot y)$

R2: $(x \cdot y) = (u, v) \rightarrow x = u \wedge y = v$

R3: $x = \mathbf{nil} \vee \exists u, v.x = (u \cdot v)$

R4: $\neg x \prec \mathbf{nil}$

R5: $x \prec (u \cdot v) \leftrightarrow x = u \vee x \prec u \vee x = v \vee x \prec v$

Exercise 2.3.1 *Show that all true $\mathbf{\Delta}_0$-sentences are provable in Robinsons theory.*

We can then transform provability of F in Robinson theory to provability of $R1 \wedge R2 \wedge R3 \wedge R4 \wedge R5 \rightarrow F$ in predicate logic with equality. Now we know that provability in Robinson theory is undecidable. Hence so is provability in predicate logic with equality. A little more work shows that we can eliminate equality and get that provability in predicate logic is also undecidable.

Exercise 2.3.2 *Show undecidability of provability in predicate logic.*

Robinson theory is incomplete — we know that there are true $\mathbf{\Pi}_1$-sentences which are not provable. In fact the following is such a one

$$\forall x. \neg x = (x \cdot x)$$

We can prove this by making a model of Robinsons theory where there is an element $\star$ with $\star = (\star, \star)$. The model is made up of all terms

$$\mathfrak{T} \equiv \mathbf{nil} | \star | (\mathfrak{T} \cdot \mathfrak{T})$$

where we have as extra identity

$$\star = (\star \cdot \star)$$

This is a well defined universe with obvious interpretations of the sentences in the language. Observe that each of the axioms of the Robinsons theory is true in the model and in addition we have the sentence $\forall x.\neg x = (x \cdot x)$ false in the model.

Here we could directly construct a true $\mathbf{\Pi}_1$-sentence which is not provable. For stronger theories this is far from easy. The argument above shows only that there is such a sentence. By looking closer at the argument we can also find explicitly such a sentence. But to find one which is readily understandable for us is something different. The work by Gerhard Gentzen — and later work by Gaisi Takeuti and Kurt Schütte — has given us more understandable constructions of such sentences for stronger theories.

2.4 Non standard models

Thoralf Skolem gave another argument that we could not characterize the natural numbers in an axiomatic theory. In fact one could not characterize any infinite domain. For the natural numbers we introduce a new constant c and extra axioms

$$0 < c\ ,\ 1 < c\ ,\ 2 < c\ ,\ 3 < c\ ,\ \ldots$$

and then use the compactness theorem in predicate logic to get a new model of the natural numbers which is not isomorphic with the standard model. This construction is well known. It differs from our constructions in the following way

Skolem: We get a model which is not isomorphic to the standard model.

Gödel: We get a sentence and a model M — the sentence is true in the standard model and false in M

It is important that we can distinguish the models with a sentence from our language. It may happen that we still have non standard models a la Skolem but the language is not expressive enough to separate between the models. In fact this is so for the languages:

Presburger, Skolem: Natural numbers with only addition

Skolem: Natural numbers with only multiplication

Tarski: Real numbers with addition and multiplication

In Skolems non standard models there is very little about the tension between logic as language and logic as calculus. In Gödels incompleteness theorems this is a main theme.

2.5 Induction

In the universe of pairs induction is defined for formulas Fx as follows:

$$F(\mathbf{nil}) \wedge \forall x, y.(Fx \wedge Fy \rightarrow F(x \cdot y)) \rightarrow \forall z.Fz$$

We have full induction if there are no restriction on Fx, predicative induction if Fx is restricted to $\mathbf{\Sigma}_1$-formula. In all cases we allow additional free variables in the formulas.

We are working within the universe of pairs. It is liberating to get away from the usual data structure of unary numbers which we know is a poor data structure. We must add extra functions — as addition and multiplication — to represent syntax. And even then we must use extra tricks. Here are some other data structures we could have used instead:

- strings over an alphabet with at least two symbols
- hereditarily finite sets
- finite trees with finite branchings

In those cases we have the appropriate constructions in the data structure. Then we define elementary theories as above starting with the minimal theories where all true $\mathbf{\Delta}_0$-sentences are added as axioms. So we look for a Robinson theory and then have induction axioms. We leave such developments as exercises for the reader.

2.6 Skolem theory

A data structure gives a way to build up the universe from below — and induction reflects this way of building up as long as the induction formula does not refer to the whole universe but only the universe built up so far. This tells us that quantifier free induction is conceptually simpler than full induction.

Thoralf Skolem pushed the quantifier free formulas a little by allowing as terms the primitive recursive functions. Let us call

Skolem theory: We extend the language by allowing terms for functions defined by primitive recursion (over pairs). The Skolem theory is then the theory where we have as axioms all true $\mathbf{\Delta}_0$-sentences as in **MIN**, and induction over $\mathbf{\Delta}_0$-formulas.

The Skolem theory can be considered as a predicative theory — we do not refer to the totality of all pairs in arguing that the theory is sound.

We have used the language of pairs. In Skolem theory this does not really matter. The language is rich enough to represent syntax with $\mathbf{\Delta}_0$ predicates even if we have as the underlying data structure the unary numbers. There it is sufficient to note that we have terms representing addition, multiplication and exponentiation.

In the Skolem theory the two most important properties are:

Primitive recursive realizers: To every provable specification we have a primitive recursive realizer. That is if $\vdash \forall x \exists y F\, x\, y$ with F a $\mathbf{\Delta}_0$ formula, then there is a primitive recursive term tx with $\vdash \forall x F\, x\, tx$

$\mathbf{\Sigma}_1$-induction: We can derive Σ_1-induction in Skolem theory. In fact we can derive induction for formulas which are disjunctions of $\mathbf{\Sigma}_1$ and $\mathbf{\Pi}_1$ formulas.

The first property shows that the formula $\forall x \exists y A x y$ expressing that the Ackermann function is total is true but not provable in Skolem theory.

Chapter 3

Decidable theories

3.1 Decidable versus undecidable

We have seen that we get undecidability for consistent systems when

- We have a language strong enough to express the usual syntactic constructions as Δ_0-formulas
- The axiom system is strong enough to derive all true Δ_0-sentences

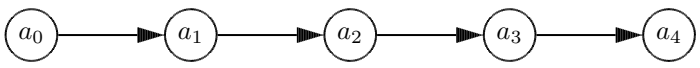
For some systems we can derive decidability. But then either the language is not strong enough or the system is not strong enough. Let us look a little closer at this phenomenon.

3.2 Automata theory

We start with the theory of finite automata. For our purposes here we can consider the automata as finite coloring devices. Given a finite alphabet $\mathcal{A}$. We consider

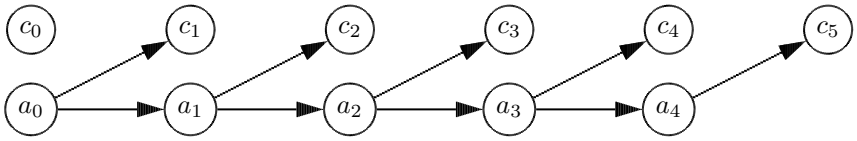
- strings over $\mathcal{A}$
- (binary) trees over $\mathcal{A}$

The restriction to binary trees is not of much importance — it makes the presentation simpler. Given a string over $\mathcal{A}$

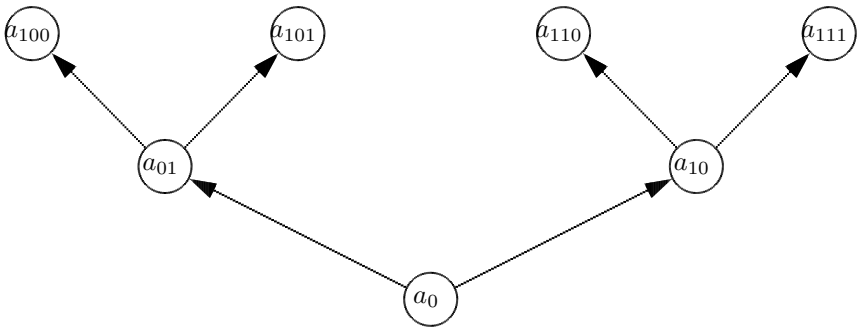


An automaton over $\mathcal{A}$ is a coloring scheme. There is given a finite set of colors $\mathcal{C}$ and an association $\mathcal{A} \times \mathcal{C} \to \mathcal{C}$. If this association is a function, then

the automaton is deterministic. Given a color for the first element in the string, we get a coloring for all elements. We can think of this as follows



We start with the string and the first color c_0. From c_0 and a_0 we get c_1, from c_1 and a_1 we get c_2, and so on. Note that this coloring scheme also works for infinite strings. Let us now go to binary trees. Below is one.



We get a coloring scheme for this whenever we have

- A starting color — the color of a_0
- An association $\mathcal{A} \times \mathcal{C} \to \mathcal{C} \times \mathcal{C}$

As before this works also for infinite binary trees. The automata is defined by

- A finite alphabet $\mathcal{A}$
- A finite set of colors $\mathcal{C}$
- A starting color
- An association which for strings are of form $\mathcal{A} \times \mathcal{C} \to \mathcal{C}$ and for binary trees of form $\mathcal{A} \times \mathcal{C} \to \mathcal{C} \times \mathcal{C}$
- An acceptance criterion

An automaton is deterministic if the association above is a function. For indeterministic automata we may have many different colorings. An input is accepted if there is a coloring which is accepted.

So far we have not said anything about the acceptance criterion. Here we have a difference between finite and infinite input. For finite strings we end up with a color at the end. For infinite strings the acceptance have to do with which colors occur infinitely often. Let us list up the acceptance criteria:

Finite string We have a subset of colors — the accepting colors. A coloring is accepted if it ends up with an accepting color

Infinite string There is a set of subsets of colors. A coloring is accepted if the colors which occur infinitely often are exactly one of these subsets.

Finite or infinite tree A tree is accepted if all its branches are

The logical complexity of acceptance can be quite complicated. In the treatment of indeterminism we say "There exists a coloring ...", and for the trees we say "For all branches ...". It is surprising that questions about acceptance may turn out to be decidable – even for infinite strings and trees. But more about this below.

3.3 Logic of acceptance

We consider on the one hand automata



And on the other hand formulas where the variables range over input-data, and a formula $\Phi(x)$ is true for x if the corresponding automaton accepts x. We now note that there is a correspondence between logical operations on formulas and constructions on automata. Some examples:

Negation: For deterministic automata we get negation by taking as new accepting colors the complement of the old. For nondeterministic automata we first change them into deterministic automata by the usual subset construction and then taking the complement for detrministic automata.

Conjunction: We just take the cartesian product of the automata.

Existential quantifier: We use nondeterministic choice as an extra input.

3.4 Presburger arithmetic

We can use finite automata to add numbers. Numbers are then given in the form:

- Binary numbers (or some higher number system beyond the unary numbers)
- The numbers are read from right to left

But then natural numbers with addition is decidable. In this theory we can also define order.

3.5 Arithmetic with only multiplication

We can multiply using tree automata. The numbers are then given in the form

- We use prime number decomposition
- The exponents are binary numbers read from right to left

We multiply two numbers by adding the exponents. For acceptance of a multiplication $a \times b = c$ we had to check whether all exponents match. For tree automata we have acceptance if all branches are accepted.

Therefore the theory of natural numbers with multiplication is decidable.

3.6 Numbers with addition and multiplication

If we now consider theories with addition and multiplication, then we have

- Natural numbers — we have shown that the theory is undecidable.
- Integers — we can define the natural numbers by using the fact that they are exactly those integers which can be written as sums of four squares. This shows that the theory is undecidable.
- Rationals — Julia Robinson have used more advanced number theory to define the integers within the rationals. And we get that this theory is undecidable.
- Reals — Alfred Tarski has shown that this theory is decidable. In fact he showed that one could eliminate quantifiers in the theory of reals with addition and multiplication.

One may wonder why the reals are so much simpler than the natural numbers. This has to do with the poor language of reals with addition and multiplication. For the coding of syntax the crucial thing seems to be able to handle finite sequences, and we are not able to do that. Quantifier statements can be reduced to questions of whether some equation is solvable. Often we can answer that by just looking at the coefficients and need not find the solution. The typical case is an an equation of odd degree

$$x^{2n+1} + a_{2n}x^{2n} + \cdots + a_1 x + a_0 = 0$$

which always has a solution in the reals no matter what the coeffisients are.

Chapter 4

Diagonalization

4.1 Cantors argument

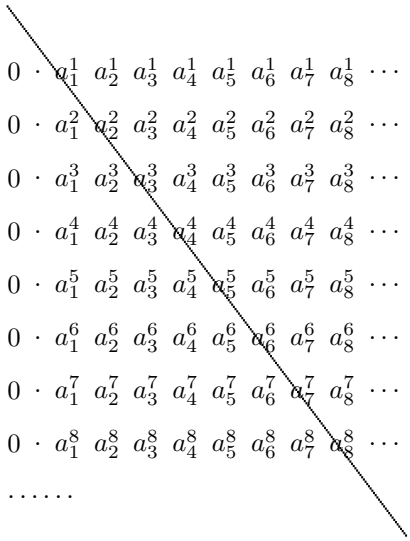
We all know Cantors diagonal argument. In this chapter we spell it out in a little more detail than usual to make clear the underlying technique.

Georg Cantor used first a topological argument to show that the real numbers is not countable. Later he found the diagonal argument.

Assume the real numbers between 0 and 1 is countable. Then we could put them below each other:

$$
\begin{array}{l}
0 \cdot a_1^1\ a_2^1\ a_3^1\ a_4^1\ a_5^1\ a_6^1\ a_7^1\ a_8^1 \cdots \cdots \\
0 \cdot a_1^2\ a_2^2\ a_3^2\ a_4^2\ a_5^2\ a_6^2\ a_7^2\ a_8^2 \cdots \cdots \\
0 \cdot a_1^3\ a_2^3\ a_3^3\ a_4^3\ a_5^3\ a_6^3\ a_7^3\ a_8^3 \cdots \cdots \\
0 \cdot a_1^4\ a_2^4\ a_3^4\ a_4^4\ a_5^4\ a_6^4\ a_7^4\ a_8^4 \cdots \cdots \\
0 \cdot a_1^5\ a_2^5\ a_3^5\ a_4^5\ a_5^5\ a_6^5\ a_7^5\ a_8^5 \cdots \cdots \\
0 \cdot a_1^6\ a_2^6\ a_3^6\ a_4^6\ a_5^6\ a_6^6\ a_7^6\ a_8^6 \cdots \cdots \\
0 \cdot a_1^7\ a_2^7\ a_3^7\ a_4^7\ a_5^7\ a_6^7\ a_7^7\ a_8^7 \cdots \cdots \\
0 \cdot a_1^8\ a_2^8\ a_3^8\ a_4^8\ a_5^8\ a_6^8\ a_7^8\ a_8^8 \cdots \cdots \\
\ldots\ldots \\
\ldots\ldots
\end{array}
$$

Now it is easy to find a number not in the enumeration. Draw the diagonal through a_i^i



$$
\begin{array}{llllllllll}
0 \cdot & a_1^1 & a_2^1 & a_3^1 & a_4^1 & a_5^1 & a_6^1 & a_7^1 & a_8^1 & \cdots \cdots \\
0 \cdot & a_1^2 & a_2^2 & a_3^2 & a_4^2 & a_5^2 & a_6^2 & a_7^2 & a_8^2 & \cdots \cdots \\
0 \cdot & a_1^3 & a_2^3 & a_3^3 & a_4^3 & a_5^3 & a_6^3 & a_7^3 & a_8^3 & \cdots \cdots \\
0 \cdot & a_1^4 & a_2^4 & a_3^4 & a_4^4 & a_5^4 & a_6^4 & a_7^4 & a_8^4 & \cdots \cdots \\
0 \cdot & a_1^5 & a_2^5 & a_3^5 & a_4^5 & a_5^5 & a_6^5 & a_7^5 & a_8^5 & \cdots \cdots \\
0 \cdot & a_1^6 & a_2^6 & a_3^6 & a_4^6 & a_5^6 & a_6^6 & a_7^6 & a_8^6 & \cdots \cdots \\
0 \cdot & a_1^7 & a_2^7 & a_3^7 & a_4^7 & a_5^7 & a_6^7 & a_7^7 & a_8^7 & \cdots \cdots \\
0 \cdot & a_1^8 & a_2^8 & a_3^8 & a_4^8 & a_5^8 & a_6^8 & a_7^8 & a_8^8 & \cdots \cdots \\
\ldots\ldots & & & & & & & & & \\
\ldots\ldots & & & & & & & & &
\end{array}
$$

It is sufficient to find a number that differs from each of the enumerated numbers on the diagonal. There is a small extra problem for the real numbers — two numbers may be equal even if they have different decimal expansion like.

$$1.0000000\ldots = 0.99999999\ldots$$

A way around this is to chose the number $0.b_1b_2b_3b_4b_5\ldots$ where

$$b_i = \begin{cases} 3 \text{ om } a_i^i \neq 3 \\ 7 \text{ om } a_i^i = 3 \end{cases}$$

And we get a real number $0.b_1b_2b_3b_4b_5\ldots$ not in the enumeration.
This well known argument contains some ingredients

- Make a 2-dimensional schema.
- Look at the diagonal.
- Transform the diagonal to something like one of the rows in the schema.
- Look at the place where the row meets the diagonal.

Below we give some typical uses of the argument.

4.2 Russells paradox

We make our schema with the predicate $x \in y$. On the axes in the schema we have sets and on the point belonging to column x and row y we put the truth value of $x \in y$. As transformation we apply negation $\neg$ to all elements. The transformed diagonal contains all

$$\neg x \in x$$

In Russells paradox we look for a set R giving the transformed diagonal. Hence

$$x \in R \leftrightarrow \neg x \in x$$

And this is impossible by looking at the point where the line R meets the diagonal $R \in R$.

4.3 Fix point of functions

For functions with one argument we make the two dimensional schema by writing functions f_k along x-axis and y-axis. At point (k, l) we write the composition of the two functions

$$\lambda x.f_k(f_l(x)) = f_k \circ f_l$$

Assume that we can transform this by using a function Φ. Then the transformed diagonal looks like

$$\lambda X.\Phi(X \circ X)$$

Under some quite general conditions the transformed diagonal is equal to some row $M = \lambda X.\Phi(X \circ X)$. The point where the row meets the diagonal we get

$$M \circ M = \Phi(M \circ M)$$

This is the fix point for the transformation Φ and we can write it as

$$(\lambda X.\Phi(X \circ X))(\lambda X.\Phi(X \circ X))$$

And we have the usual fix point construction from recursion theory and from lambda-calculus.

4.4 Diagonal lemma

Using coding we can compose formulas in one free variable x. The two formulas Ax and Bx can be composed as $A\lceil Bx\rceil$ where we have substituted for x in Ax the code of the formula Bx with free variable x.

We can describe this with a substitution function

$$\mathbf{sub}(a, b)$$

which does the following syntactical operation: Let a be the code for a formula Ax, then $\mathbf{sub}(a, b)$ is the code for the formula obtained by substituting b for x in Ax.

We are now ready for a diagonalization argument. Let Φx be a formula with free variable x. Then put up the two dimensional schema obtained by composition of formulae and then apply Φx on the diagonal. We then get a formula ψx such that

$$\psi x \leftrightarrow \Phi\lceil \psi x \rceil$$

A closer look reveals the following

Lemma 1 (Fix point) *Let x be a variable and Ax a formula with free variable x. There is then a sentence B with*

$$B \leftrightarrow A\lceil B \rceil$$

Proof: Suppose the language is rich enough to define the syntactical operation $\mathbf{sub}(a, b)$ used above. This is so for the language of pairs and the operation can be defined by a $\mathbf{\Delta}_0$-formula. Then

$$\begin{aligned} Cx &= A(\mathbf{sub}(x, x)) \\ m &= \lceil Cx \rceil \\ B &= Cm \end{aligned}$$

And

$$\begin{aligned} B &\leftrightarrow Cm \\ &\leftrightarrow A(\mathbf{sub}(m, m)) \\ &\leftrightarrow A(\mathbf{sub}(\lceil Cx \rceil, m)) \\ &\leftrightarrow A(\lceil Cm \rceil) \\ &\leftrightarrow A\lceil B \rceil \end{aligned}$$

Here we must replace the term **sub** with appropriate formulas.
■

4.5 Self reference

The phenomena around incompleteness depends on the language rich enough that the system can talk about itself. Raymond Smullyan has developed a simple argument to show the phenomenon in [12]. Assume we have a computer printing expressions in the 5 symbols

$$\sim \quad P \quad N \quad (\quad)$$

We then define

Expression: A non-empty word in the alphabet above.

Norm: The norm of the expression X is the expression $X(X)$

Sentence: An expression of one of the following forms (where X is an arbitrary expression

- $P(X)$
- $PN(X)$
- $\sim P(X)$
- $\sim PN(X)$

As semantics for the language we let

P**:** printable

N**:** the norm of

$\sim$**:** not

Then the expressions get a double role

- they are printed out by the computer
- they talk about what can be printed out — before or later

Assume now that *the computer is accurate* — all expressions printed out are true. So if the computer prints out $P(X)$,, then the computer will also print out X, and if computer prints out $PN(X)$, then will it print out $X(X)$. The problem now is whether it is possible to make an accurate machine which prints out exactly all true sentences. The answer is NO. For assume we had such a computer. Then it would have problems with the sentence

$$\sim PN(\sim PN)$$

This sentence is true if and only if the computer does not print out the norm of $\sim PN$ — that is it does not print out $\sim PN(\sim PN)$. Then there are two possibilities — either the sentence is true but will never be printed, or the sentence is false and is printed. Since the computer was accurate we get the first possibility. The sentence $\sim PN(\sim PN)$ is true and $PN(\sim PN)$ is untrue. Hence the computer do not print $PN(\sim PN)$. So there is a sentence $PN(\sim PN)$ such that the computer neither prints out it or its negation $\sim PN(\sim PN)$.

4.6 Currys paradox

Haskell Curry has shown that we cannot have a system with both self application and logic in the same system. Suppose we have a system with sufficient

Self application: We have the following fix point theorem: *To every expression $F(X)$ with free variable X there is a constant G with G and $F(G)$ equivalent.*

Logic: The conditionals are treated in the usual way.

Then we can derive everything. For let F be an arbitrary expression. By the fix point theorem there is a G with G and $G \to F$ equivalent. Then we have the following deduction

$$\begin{array}{c} G \to G \\ G \to (G \to F) \\ G \to F \\ G \\ F \end{array}$$

Note that in our fix point theorem we do not get G and $F(G)$ equivalent, but rather G and $F(\lceil G \rceil)$. In the context $F(X)$ we use not G but the code of G.

Chapter 5

Provability

The main references to this chapter and the next is Smoryńskis article in [2] and the monographs by Boolos [3] and Smoryński [11].

5.1 Representing syntax

We have seen that syntax can be represented as $\mathbf{\Delta}_0$-formulas in the language of pairs. The next step is to consider provability within a theory. The provability of a formula F is analyzed as

- There exists a syntactical object p
- The object p represents a proof of the formula F

which is written as

$$\exists p.\mathbf{PROOF}(p, \lceil F \rceil)$$

Here $\lceil F \rceil$ is a pair representing the formula F. The quantifier $\exists p$ runs over pairs. We now make the crucial assumption about the theory and our representation

The predicate **PROOF**(p,q) is $\mathbf{\Delta}_0$, and provability is $\mathbf{\Sigma}_1$

This is so for the all usual theories and it excludes the case where we have as axioms all true sentences of pairs.

Now we want to investigate the $\mathbf{\Sigma}_1$ formula expressing that some formula F is provable. We write:

$$\Box F$$

We must be careful with the notation here. The F is not a sub formula. It is rather a pair representing F. The $\Box$ expresses that there is a pair representing

a proof of F. If we want to emphasize that the proof is within a system T we write

$$\Box_T F$$

The construction here is quite robust — the details of the actual representation does not matter much. Later we shall demand the following:

Fair coding of equality: $\forall x, y.(x = y \rightarrow \Box x = y)$

Fair coding of inequality: $\forall x, y.(x \neq y \rightarrow \Box x \neq y)$

Observe that neither of these are obvious — and are extra conditions on the representations.

5.2 Provability as necessity

We shall consider the $\Box$ as a modality operator. In fact we shall prove the following:

GL0: $\vdash F \Rightarrow \vdash \Box F$

GL1: $\vdash \Box F \wedge \Box(F \rightarrow G) \rightarrow \Box G$

GL2: $\vdash \Box F \rightarrow \Box\Box F$

GL3: $\vdash \Box(\Box F \rightarrow F) \rightarrow \Box F$

Here GL is from Gödel and Löb. Let us look at these properties.

GL0: $\vdash F \Rightarrow \vdash \Box F$

This reflects only that the formalized provability $\Box F$ is a fair representation of real provability $\vdash F$. As long as we can represent provability as a formula this should be true.

GL1: $\vdash \Box F \wedge \Box(F \rightarrow G) \rightarrow \Box G$

Here we have the formalized version of modus ponens. In many cases the theories are such that modus ponens is just an extra step in the formalized proof. But note that we may have notions of provability where this is not any longer true

- as directly provable
- as F provable and there is no shorter proof of $\neg F$

In our provability the modus ponens is a rule and **GL1** is true. With **GL0** and **GL1** we get a modal logic for provability. Note that we then get the following rule: *If* $\vdash \Box F$ *and* G *is a tautological consequence of* F*, then* $\vdash \Box G$*.*

The two extra properties characterize our modal logic and are harder to prove. As a preparation we prove the following interesting property:

Theorem 4 ($\mathbf{\Sigma}_1$ completeness) *Assume that the theory contains Σ_1-induction and we have a fair coding of equality and inequality. Then for all Σ_1 formulas G*

$$\vdash G \rightarrow \Box G$$

Here G may contain free variables.

Proof: Note that the theory will also contain Robinson theory. The proof is first by induction over the build up of $\mathbf{\Delta}_0$ formulas. Then we show that the principle still holds if we have $\exists$-quantifiers outermost.

Literals: The fair coding of equality and of inequality gives the principle for $x = y$ and for $x \neq y$. It also gives the principle for $x = (u \cdot z)$ and $x \neq (u \cdot z)$. Just substitute $(u \cdot z)$ for y. For $x < y$ we use $\mathbf{\Sigma}_1$ induction over y. In the induction start $y = \mathbf{nil}$ we have $\vdash \neg x < \mathbf{nil}$ and trivially the principle. Assume the principle true for y and z. We then get it for $(y \cdot z)$ by using the axiom of Robinson theory:

$$x < (y \cdot z) \leftrightarrow x = y \vee x < y \vee x = z \vee x < z$$

Now to $\neg x < y$. Again we use $\mathbf{\Sigma}_1$-induction over y. For the induction start $y = \mathbf{nil}$ we note

$$\vdash \Box \neg x < \mathbf{nil}$$

In the induction step we again use the equivalence above.

Conjunction: Assume $\vdash F \rightarrow \Box F$ and $\vdash G \rightarrow \Box G$. But from **GL0** we have $\vdash \Box(F \rightarrow (G \rightarrow F \wedge G))$ and using **GL1** and propositional logic we get $\vdash F \wedge G \rightarrow \Box F \wedge G$.

Disjunction: Here we use **GL0** with $\vdash \Box(F \rightarrow F \vee G)$ and $\vdash \Box(G \rightarrow F \vee G)$.

Bounded quantifiers: We use $\mathbf{\Sigma}_1$-induction over y to prove

$$\vdash \exists x < y.Gx \rightarrow \Box \exists x < y.Gx$$

Note that the formula $\exists x < y.Gx$ is $\mathbf{\Delta}_0$ and hence that the whole formula is $\mathbf{\Sigma}_1$. For the induction we use the axioms from Robinson theory:

R1: $\neg \mathbf{nil} = (x \cdot y)$

R2: $(x \cdot y) = (u, v) \rightarrow x = u \wedge y = v$

R3: $x = \mathbf{nil} \vee \exists u, v.x = (u \cdot v)$

R4: $\neg x < \mathbf{nil}$

R5: $x < (u \cdot v) \leftrightarrow x = u \vee x < u \vee x = v \vee x < v$

In the same way we prove by induction

$$\vdash \forall x < y.Gx \to \Box \forall x < y.Gx$$

We conclude that the principle is true for all $\mathbf{\Delta}_0$ formulas. Now we note that it can be extended with $\exists$-quantifiers in front.

Existential quantifier: We assume $\vdash Fx \to \Box Fx$ for arbitrary x. Furthermore by **GL0** $\vdash \Box(Fx \to \exists y.Fy)$. Then $\vdash Fx \to \Box \exists y.Fy$ and $\vdash \exists y.Fy \to \exists y.Fy$.

■

So the principle is true for all $\mathbf{\Sigma}_1$ formulas. In particular it is true for the $\mathbf{\Sigma}_1$ formula $\Box F$ and we get for theories with $\mathbf{\Sigma}_1$-induction

GL2: $\vdash \Box F \to \Box\Box F$

We now prove Löbs rule

Löbs rule: If $\vdash \Box S \to S$, then $\vdash S$.

Here we use **GL0**, **GL1** and **GL2** and in addition the fix point theorem. From the fix point theorem there is an I with $\vdash I \leftrightarrow (\Box I \to S)$. We then have

$\vdash I \to (\Box I \to S)$
$\vdash \Box I \to (\Box\Box I \to \Box S)$
$\vdash \Box I \to \Box S$
$\vdash \Box S \to S$, assumption
$\vdash \Box I \to S$
$\vdash I$
$\vdash \Box I$
$\vdash S$, conclusion

We are then ready to prove

GL3: $\vdash \Box(\Box F \to F) \to \Box F$

We abbreviate $B = \Box(\Box F \to F)$, $C = \Box F$ and $D = B \to C$. Then

$\vdash \Box D \to (\Box B \to \Box C)$
$\vdash B \to (\Box C \to C)$
$\vdash B \to \Box B$, since B starts with $\Box$
$\vdash \Box D \to (B \to C)$
$\vdash \Box D \to D$
$\vdash D$, by Löbs rule

And we are done. Note that we had to use $\mathbf{\Sigma}_1$-induction to prove **GL2** and this was again used in the proof of **GL3**.

5.3 Tarskis theorem

Assume that we have a formalized notion of truth one within our theory. That is there is a predicate **TR** such that for all sentences S

$$\vdash \mathbf{TR}(\lceil S \rceil) \leftrightarrow S$$

The point is that the S on the left hand side occurs as the coded representation of it. We can then use the fix point theorem to get a sentence T with

$$\vdash \neg\mathbf{TR}(\lceil T \rceil) \leftrightarrow T$$

and we have an immediate contradiction.

Theorem 5 (Tarski) *There is no formalized theory of truth in a theory where we can prove the fix point theorem.*

5.4 Consistency

We now want to replace the semantical assumption "only true sentences are provable within the theory" with some weaker syntactical notion. Gödel considered two notions

- A theory is *consistent* if $\nvdash \bot$.
- A theory is ω*-consistent* if there are no Fx with $\vdash \exists x.Fx$ and $\vdash \neg Fp$ for all pairs p

We could also have formulated consistency as demanding that there are no formulas F with both $\vdash F$ and $\vdash \neg F$. Then we see that ω-consistency implies consistency. Instead of ω-consistency it may be more perspicuous to use the following consequence

Theorem 6 *If the theory is ω-consistent and provability is $\mathbf{\Sigma}_1$, then*

$$\vdash \Box F \Rightarrow \vdash F$$

Proof: For assume $\vdash \Box F$ or $\vdash \exists p.\mathbf{PROOF}(p, \lceil F \rceil)$. Then by ω-consistency there must be a pair q with $\vdash \mathbf{PROOF}(q, \lceil F \rceil)$. But this is a $\mathbf{\Delta}_0$ sentence and hence true. Therefore $\vdash F$. ■

5.5 Gödels first incompleteness theorem

We have already seen that we cannot expect that a theory of pairs could treat the true sentences in a reasonable way. Under quite general assumptions:

- all true $\mathbf{\Delta}_0$-sentences are provable
- only true sentences are provable
- provability is partially computable

we conclude that there must be a true $\mathbf{\Pi}_1$-sentence which is not provable. Here we shall show a way to get around the second assumption, and let the incompleteness be a more syntactical matter.

Theorem 7 (First incompleteness) *Assume we have a theory in the language of pairs where all true $\mathbf{\Delta}_0$ sentences are provable. Let G be a sentence which is fix point such that*

$$\vdash G \leftrightarrow \neg\Box G$$

Then under the assumptions below we get:

Consistent: *Then $\nvdash G$*

ω-consistent: *Then $\nvdash \neg G$*

Proof:
Assume that the theory is consistent. Assume $\vdash G$. Then $\vdash \Box G$ by **GL0** and from the definition of G we get $\vdash \neg G$ contradicting consistency.
Assume that the theory is ω-consistent. Assume $\vdash \neg G$. Then $\vdash \Box G$ and from the above by ω-consistency $\vdash G$. This contradicts the consistency of the theory. ■

Under the weak assumption of ω-consistency of the theory we get a sentence G such that neither it nor its negation $\neg G$ is provable. So the theory is not complete.

The assumption used before about the theory being partially computable is here reflected in the assumption that $\Box F$ is a $\mathbf{\Sigma}_1$ formula.

5.6 Gödels second incompleteness theorem

We now use the stronger assumption **GL2** of provability.

Theorem 8 (Second incompleteness) *Let G be the sentence used in the first incompleteness theorem. Then*

$$\vdash G \leftrightarrow \neg\Box\bot$$

Proof: We have $\vdash G \rightarrow \neg\Box G$. Then use $\vdash \bot \rightarrow G$ and **GL0** and **GL1** to get $\vdash \Box\bot \rightarrow \Box G$. Hence $\vdash G \rightarrow \neg\Box\bot$ which is half of the equivalence.

Conversely by **GL2** we have $\vdash \Box G \rightarrow \Box\Box G$ and hence $\vdash \Box G \rightarrow \Box\neg G$. Then $\vdash \Box G \rightarrow \Box(G \wedge \neg G)$ and $\vdash \Box G \rightarrow \Box\bot$ and $\vdash \neg G \rightarrow \Box\bot$ which is the other half of the equivalence. ■

Both incompleteness theorems use that provability is $\mathbf{\Sigma}_1$. The first incompleteness theorem requires that all true $\mathbf{\Delta}_0$ sentences are provable, while the second incompleteness theorem requires $\mathbf{\Sigma}_1$ induction in the theory — to get **GL2** — and that the coding represents equality and inequality in a fair way.

The second incompleteness theorem shows that it makes sense to talk about the Gödel sentence of a theory — and that this sentence is equivalent to $\neg\Box\bot$. The actual theories enters only through the representation of $\Box$.

Chapter 6

Modality

6.1 The Gödel-Löb modal logic

We have seen that we can interpret provability as a modal operator where we have the following:

GL0: $\vdash F \Rightarrow \vdash \Box F$

GL1: $\vdash \Box F \wedge \Box(F \to G) \to \Box G$

GL2: $\vdash \Box F \to \Box\Box F$

GL3: $\vdash \Box(\Box F \to F) \to \Box F$

From the literature this modal logic is known as the Gödel-Löb modal logic — abbreviated as GL. The logic GL is treated in a number of textbooks on modal logic. We write $\vdash_{GL} F$ for provability in the logic GL. Note that **GL2** can be derived from the rest:

$\vdash \Box(\Box(\Box F \wedge F) \to (\Box F \wedge F)) \to \Box(\Box F \wedge F)$ by **GL3**
$\vdash \Box(\Box F \wedge F) \to \Box\Box F$ by **GL0** and **GL1**
$\vdash \Box(\Box F \wedge F) \to \Box F$ by **GL0** and **GL1**
$\vdash \Box(\Box(\Box F \wedge F) \to (\Box F \wedge F)) \to \Box\Box F$ by logic
$\vdash F \to (\Box(\Box F \wedge F) \to (\Box F \wedge F))$ by logic
$\vdash \Box F \to \Box(\Box(\Box F \wedge F) \to (\Box F \wedge F))$ by **GL0** and **GL1**
$\vdash \Box F \to \Box\Box F$ by logic

The reason for putting **GL2** in the list defining the GL is as follows. We used **GL2** as a steppingstone in deriving **GL3**. It is also useful to point out **GL2** as an extra principle. The modal logic given by **GL0**, **GL1** and **GL2** is known as K4 and is complete for transitive frames.

We now develop a little more of theory behind GL. It is useful to write down a falsification scheme for GL. In ordinary logic we get a tree with AND and OR branches – as we did for true / false for $\mathbf{\Delta}_0$-sentences. We get a slight

simplification by considering formulas with negations innermost. In modal logic we then also need the modality $\Diamond$ given by

$$\Diamond F \leftrightarrow \neg\Box\neg F$$

A more important simplification is to consider sequents instead of single formulas. A sequent is a finite set of formulas. For derivability the sequent is interpreted disjunctively and for falsifiability it is interpreted conjunctively. A sequent Γ is falsifiable if there is a falsification of each of the formulas contained in it. We have the usual shorthand Γ, F and F, G and so on for sequents. We write $\mathfrak{F} : \Gamma$ for Γ being falsifiable. Then

$$\begin{aligned} \mathfrak{F} : \Gamma, F \wedge G &\Leftrightarrow \mathfrak{F} : \Gamma, F \text{ or } \mathfrak{F} : \Gamma, G \\ \mathfrak{F} : \Gamma, F \vee G &\Leftrightarrow \mathfrak{F} : \Gamma, F, G \end{aligned}$$

We se how the sequents are used to avoid AND-branchings in the analysis. Using these rules we get a tree with sequents with no outermost conjunction or disjunction at the leafs. We call these sequents for critical sequents. They are of the form

$$\Lambda, \Box\Gamma, \Diamond\Delta$$

where Λ are literals, $\Box\Gamma$ are formulas of form $\Box G$, and $\Diamond\Delta$ are formulas of form $\Diamond D$. We can go on and analyze the critical sequents. In the logic GL this is done by the rule

$$\mathfrak{F} : \Lambda, \Box\Gamma, \Diamond\Delta \quad \Leftrightarrow \quad \text{For all } G \in \Gamma\text{: } \mathfrak{F} : G, \neg G, \Delta, \Diamond\Delta$$

Here $\neg G$ is the formula where the $\neg$ are pushed innermost using de Morgan laws as usual. Note that we here get an AND-branching. There are two ways to stop the analysis of a critical sequent $\Lambda, \Box\Gamma, \Diamond\Delta$

Not falsifiable: Λ is not falsifiable by containing both an atomic formula and its negation. If $\bot$ and $\top$ is part of the language, then Λ is not falsifable if it contains $\top$

Falsifiable: Λ is falsifable and Γ is empty.

Given a sequent Γ we can write down the analysis tree over Γ as an AND-OR-tree with sequents at the nodes. The sequents at the topmost nodes are critical sequents and are hence either falsifiable or not falsifiable. The AND-branchings are always immediately above a critical sequent where we have analyzed some $\Box$'s and some $\Diamond$'s. The falsifiability / non falsifiability of the topmost nodes propagates downwards to the root of the analysis tree in the usual way respecting the AND- and the OR-branchings.

Let us look a little closer how properties propagate in an AND-OR-tree. We assume we investigate a property $\mathfrak{P}$ for nodes in the tree :

Top node: We have assigned $\mathfrak{P}$ or $\neg\mathfrak{P}$

AND-node: It has property $\mathfrak{P}$ iff all nodes immediately above has property $\mathfrak{P}$

OR-node: It has property $\mathfrak{P}$ iff there exist node immediately above with property $\mathfrak{P}$

Tree: The tree has property $\mathfrak{P}$ iff the root node has property $\mathfrak{P}$.

We have a witness for a tree having property $\mathfrak{P}$ if we have a subtree of AND-nodes and top nodes with the top nodes having property $\mathfrak{P}$ preserving the branching in the AND-nodes.

A GL-model is a Kripke model where the frame is a finite tree. That is we have

- a finite tree with nodes n
- to each node we assign a truth value of all atomic formulas
- the ordering among the nodes is given by the transitive (but not reflexive) closure of the tree ordering.

We can now prove using standard techniques that derivability in GL corresponds exactly to validity in all Kripke models over finite trees. We go through the main points in the proof:

Lemma 2 (Analysis) *Assume that the sequent Γ is falsifiable with a Kripke model on a finite tree as frame, then any analysis tree $\mathbb{T}$ over Γ is falsifiable.*

Proof: We assume as above and following the construction of $\mathbb{T}$ we construct a witnessing subtree of $\mathbb{T}$ proving $\mathbb{T}$ falsifiable. The nodes in the witnessing subtree contains a sequent falsifiable at a node in the Kripke model. Assume that sequent Θ are at node n in $\mathbb{T}$ and is falsified at node x in the Kripke model. Then depending on the analysis of the sequent we get:

- By the analysis rules for $\wedge$ and $\vee$ we get a critical sequent in $\mathbb{T}$ falsified at x. This critical sequent gives either a top node or an AND-node in the witnessing subtree.
- By the analysis rule for critical sequent $\Lambda, \Box\Gamma, \Diamond\Delta$ we get either:

  Stop: Λ is falsifiable, and Γ is empty. Then take this as a top node in the witnessing subtree. Observe that we there have $\Diamond\Delta$ trivially falsifiable.

  Continue: Λ is falsifiable and Γ is not empty. Then to each $G \in \Gamma$ we can find nodes x_G above x with $G, \Delta, \Diamond\Delta$ falsifiable. Let the x_G be the node highest up in the tree with $G, \Delta, \Diamond\Delta$ falsified. We then have also $G, \neg\Box G, \Delta, \Diamond\Delta$ falsified since for all higher nodes we have G true. We let the x_G be nodes in the witnessing subtree.

We construct a witnessing subtree for $\mathbb{T}$ and get $\mathbb{T}$ falsifiable. ■

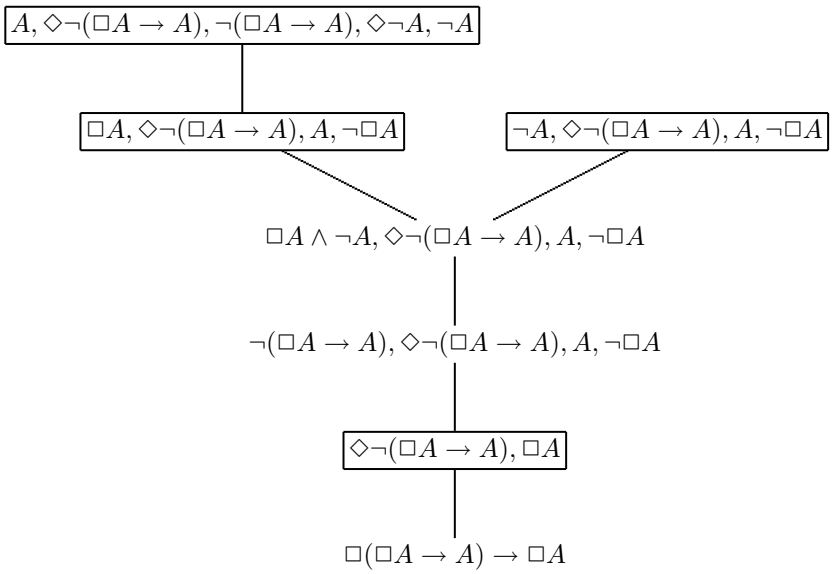
Lemma 3 (Synthesis) *Assume that the analysis tree $\mathbb{T}$ over Γ is falsified. Then we can find a Kripke model over a finite tree which falsifies Γ*

Proof: We use the witnessing subtree for $\mathbb{T}$ to construct a Kripke model. The frame is the witnessing subtree itself and the interpretation is such that all literals at each node is falsified. Then by induction over the formulas we show in a straightforward way that all formulas occurring at a node is false in Kripke model at that node. ■

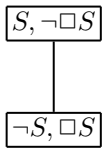
From these two lemmas we get

Theorem 9 (Completeness of GL) *A sequent Γ has a falsifiable analysis tree if and only if it can be falsified in a Kripke model over a finite tree (where we use transitive closure of the tree ordering).*

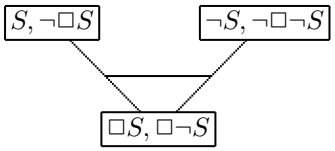
We look at an analysis tree for GL2:



And we see that **GL2** is valid as it should. Here are two examples where we are able to falsify the sequents:



We have as falsifying model the tree with two nodes, the upper most making S true and the down most making S false. We shall use this construction later to get a sentence which is true but not provable. A sentence S where neither S nor $\neg S$ is provable is constructed using:



The AND-branching is marked here with a horizontal line between the branches. The falsifying Kripke model is given by the three nodes indicated where for the two upper most, the left makes S false while the right makes it true. For the down most node we could make it either true or false — it does not matter.

6.2 Solovays first completeness theorem

In this section we shall show — following work of Robert Solovay [13] — that GL tells the whole story about provability in elementary theories with $\mathbf{\Sigma}_1$-induction.

We want to simulate trees and interpretations over a tree in an elementary theory. So assume that we have a representation of the tree given by a downmost node 0 and the accessibility relation $\prec$ and downmost world 0. For notational simplicity we use natural numbers and usual less than relation on them. Define a primitive recursive (or using $\mathbf{\Sigma}_1$ induction) climbing function h by

$$\begin{array}{rcl} h(0) & = & 0 \\ h(x+1) & = & \left\{ \begin{array}{ll} j & \text{, where } j \succ h(x) \text{ and } x \text{ proves } \exists y > x \cdot h(y) \succ j \\ h(x) & \text{, otherwise} \end{array} \right. \end{array}$$

So it describes a man climbing up the frame, and he can only climb up a step if he can prove that he will not stay there. (Or, it could be a refugee who is allowed to enter a country only if he can prove that he will go to a more favorable country.) We then define predicates S_i expressing that the climber will ultimately end up with world i

$$S_i = \exists x \forall y > x \cdot h(y) = i$$

We can then prove

$$\begin{array}{ll} \vdash S_i \rightarrow \Box\neg S_i & \text{for } i \succ 0 \\ \vdash S_i \rightarrow \neg\Box\neg S_j & \text{for } j \succ i \\ \vdash \neg(S_i \wedge S_j) & \text{for } i \neq j \\ \vdash \bigvee_i \cdot S_i & \\ \vdash S_i \rightarrow \Box \bigvee_{j \succ i} S_j & \text{for } i \succ 0 \end{array}$$

Only the last requires some work. We can derive within the theory using $i \succ 0$ and the formalized $\mathbf{\Sigma}_1$-completeness

$$
\begin{array}{l}
S_i \rightarrow \exists a \cdot h(a) = i \\
\exists a \cdot h(a) = i \rightarrow \bigvee_{j \succeq i} S_j \\
\Box \exists a \cdot h(a) = i \rightarrow \Box(S_i \vee \bigvee_{j \succ i} S_j) \\
\exists a \cdot h(a) = i \rightarrow \Box \exists a \cdot h(a) = i \text{ using } \mathbf{\Sigma}_1\text{-completeness} \\
S_i \rightarrow \Box(S_i \vee \bigvee_{j \succ i} S_j) S_i \rightarrow \Box(\neg S_i) \\
S_i \rightarrow \Box \bigvee_{j \succ i} S_j
\end{array}
$$

We are now ready to simulate any finite, transitive frame in arithmetic. So assume such a frame is given and let S_i be the corresponding sentences. There is also defined a provability operator. To any formula F in GL we define a formula $F^\star$ in arithmetic by

- if P is an atomic formula, then $P^\star = \bigvee\{S_i | P \text{ is true in } i\}$
- $(P \wedge Q)^\star = P^\star \wedge Q^\star \;\; (P \vee Q)^\star = P^\star \vee Q^\star$
- $(\Box P)^\star = \Box P^\star$

We say that $P^\star$ interprets P. This is justified by

Lemma 4

$$
\begin{array}{l}
i \models P \;\Rightarrow \vdash S_i \rightarrow P^\star \\
i \not\models P \;\Rightarrow \vdash S_i \rightarrow \neg P^\star
\end{array}
$$

Proof: We prove this by induction over the formula P. Observe that it is true for atomic formula and it is preserved by Boolean combinations. We are left to prove it for formula $\Box Q$ assuming it true for Q. So assume first $i \models \Box Q$. Then

$$
\begin{array}{l}
\forall j \succ i \cdot j \models Q \\
\forall j \succ i \cdot \vdash S_j \rightarrow Q^\star \\
\vdash \bigvee_{j \succ i} S_j \rightarrow Q^\star \\
\vdash \Box \bigvee_{j \succ i} S_j \rightarrow \Box Q^\star \\
\vdash S_i \rightarrow \Box Q^\star
\end{array}
$$

And assume $i \not\models \Box Q$. Then

$$
\begin{array}{l}
\exists j \succ i \cdot j \models Q \\
\exists j \succ i \cdot \vdash S_j \rightarrow \neg Q^\star \\
\exists j \succ i \cdot \vdash \neg \Box \neg S_j \rightarrow \neg \Box Q^\star \\
\vdash S_i \rightarrow \neg \Box Q^\star
\end{array}
$$

■

Theorem 10 (Solovays first completeness theorem) $\vdash_{GL} A \Leftrightarrow \forall \star \vdash_{\mathcal{S}} A^\star$

Proof: We have already proved the implication $\Rightarrow$. To the other way assume $\nvdash_{GL} A$. There exists then a finite, transitive, conversely wellfounded frame with downmost element 1 and $1 \not\models A$. Tack on a new element 0 in the frame below 1 and let the interpretations in 0 be arbitrary. We have

$$\begin{array}{l} \vdash S_1 \rightarrow A^\star \\ \vdash \neg\Box\neg S_1 \rightarrow \neg\Box A^\star \\ \vdash S_0 \rightarrow \neg\Box\neg S_1 \\ \vdash S_0 \rightarrow \neg\Box A^\star \end{array}$$

But S_0 is true (even if it is not provable). Therefore $\Box A^\star$ is false and hence $\nvdash_{\mathcal{S}} A^\star$.■

We have used $\mathbf{\Sigma}_1$-induction. The same arguments goes through for any stronger system.

6.3 Solovays second completeness teorem

In the proof of Solovays first completeness theorem we wandered between provability and truth. This is going to be done even more so in the Solovays second completeness theorem. First we introduce a new logical system GLS — Gödel Löb Solovay

- all valid GL-sentences
- $\Box A \rightarrow A$
- modus ponens

Note that we do not have the necessitation rule ($\vdash A \Rightarrow \vdash \Box A$) in GLS. Let F be any formula. We are going to use the formula

$$\bigwedge\{\Box A \rightarrow A | \Box A \text{ is a subformula of } F\} \rightarrow F$$

This is written shortly as $\bigwedge(\Box A \rightarrow A) \rightarrow F$.

Theorem 11 (Solovays second completeness theorem) *The following are equivalent*

1. $GL \vdash \bigwedge(\Box A \rightarrow A) \rightarrow F$
2. $GLS \vdash F$
3. $\forall \star F^\star$ *true*

Proof: Here it is straightforward to prove $1 \Rightarrow 2$ and $2 \Rightarrow 3$. So assume $GL \nvdash \bigwedge(\Box A \rightarrow A) \rightarrow F$. Then there is finite, transitive, conversely wellfounded frame with downmost element 1 giving a countermodel for it. Tack on a new

node 0 below 1 and now we assume that 0 has exactly the same literals true as 1. Then $1 \not\models F$ and $1 \models \bigwedge(\Box A \rightarrow A)$. Let the sentences S_i be defined by the climbing function as in the proof of the first completeness theorem. We first show for subformulas of F

$$1 \models B \ \Rightarrow \vdash S_0 \rightarrow B^\star$$
$$1 \not\models B \ \Rightarrow \vdash S_0 \rightarrow \neg B^\star$$

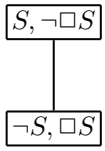
We use induction over B. If B is atomic, then $B^\star$ is a disjunction of S_is. Assume $1 \models B$. Then by construction of 0 we have $0 \models B$ and S_0 is a disjunct of $B^\star$. So $\vdash S_0 \rightarrow B^\star$. Now assume $1 \not\models B$. By construction of 0 we have $0 \not\models B$ and S_0 is not a disjunct of $B^\star$. So $\vdash S_0 \rightarrow \neg B^\star$. The properties are easily extended through Boolean combinations. It remains to prove it for $\Box C$ given the properties for C.

Assume $1 \models \Box C$. Then $\forall j \succ 1 \cdot j \models C$ and by properties proved in the first completeness theorem we have $\forall j \succ 1 \cdot \vdash S_j \rightarrow C^\star$. We now use that $1 \models \Box C \rightarrow C$ to conclude $\vdash S_1 \rightarrow C$. This gives $\vdash S_1 \vee \bigvee_{j \succ 1} S_j \rightarrow C^\star$. We now use $\vdash S_0 \vee S_1 \vee \bigvee_{j \succ 1} S_j$ to conclude $\vdash C^\star$ and $\vdash \Box C^\star$ and $\vdash S_0 \rightarrow \Box C^\star$.

Assume $1 \not\models \Box C$. Then $\exists j \succ 1 \vdash S_j \rightarrow \neg C^\star$ and $\exists j \succ 1 \vdash \neg\Box\neg S_j \rightarrow \neg\Box C^\star$. But $\vdash S_0 \rightarrow \neg\Box\neg S_j$. This gives $\vdash S_0 \rightarrow \neg\Box C^\star$.

This proves the properties. Now to the conclusion of the proof of the completeness theorem. We have assumed that $1 \not\models F$. But then $\vdash S_0 \rightarrow \neg A^\star$. But S_0 is true. Hence $A^\star$ is false.■

Let us give some simple applications. Assume we want to find a sentence S which is true but not provable in arithmetic. We then try to falsify $\neg S, \Box S$ in GL and get the following falsification tree which we have seen before



So we have two worlds — an upper with S false and a lower with S true. Observe now that in the lower one we have $\Box A \rightarrow A$ true for any subformula $\Box A$ of the sequent. There is only one $\Box F$ and we have $\Box F \rightarrow F$ true in the lower world. Now tack on a new world 0 below and we then have an interpretation for S. We can interpret it as $\neg S_2$. So in arithmetic we have $\neg S_2$ true, but not provable.

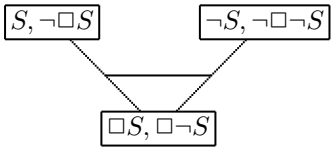
What is the complexity of the interpreted sentences? We observe that for a topmost node j we can write $S_j = \exists x \cdot h(x) = j$ since we cannot climb further, and we get S_j a $\mathbf{\Sigma}_1$-sentence. For a node i lower down we can write

$$S_i = \exists x \cdot h(x) = i \wedge \bigwedge_{k \succ i} \neg\exists y \cdot h(y) = k$$

and we get S_i a Boolean combination of $\mathbf{\Sigma}_1$-sentences.

In our example we have a true $\mathbf{\Pi}_1$-sentence which is not provable.

Let us now try to find a sentence S such that neither S nor $\neg S$ is provable. We are then led to the following falsification tree



As before observe that in the lower world we have $\Box A \to A$ true for all subformulas $\Box A$ of the sequent. Tack on a new node below and as before we have an interpretation for S — and a formula which is neither provable nor its negation is provable. This is a variant of the Rosser construction.

It is surprising that the incompleteness phenomena can be expressed with the decidable modal logic GL. On the other hand it gives a limitation on our analysis so far. The second incompleteness theorem tells us that we cannot prove $\neg\Box\bot$. The system enters in our understanding of $\Box$, but this understanding is not expressed in the language of GL.

Chapter 7

Direct proofs

7.1 Auxiliary constructions

Skolem had in his 1923 paper three levels:

Universe: Built up from a starting point with one constructor.

Language: Built up from atomic pieces with a number of constructors.

Calculus: Built up from axioms using syntactical rules.

Then Gödel showed that these levels are similar, and used this to get the incompleteness. If we now look closer at the levels we see a difference of the levels:

Direct construction: The first two levels build up from some simple elements getting more and more complicated pieces.

Constructions via detours: In the last level we may construct something quite simple by having a detour through some quite complicated constructions.

These constructions via detours is what the second incompleteness theorem indicates. The sentence $\neg\Box\bot$ is not provable — we cannot exclude the possibility that there is a very complicated way of proving $\bot$ using some tricky detours.

Gerhard Gentzen did all the main work towards an understanding of direct constructions in the 1930's [5]. This is — in our view — the way forward from Gödel.

7.2 Sequential calculus

Let us first do Gentzens analysis for predicate logic. For simplicity let

Sequent: A finite set of formulas

Negation normal form: The formulas are built up from literals using the connectives $\wedge$ and $\vee$ and the quantifiers $\forall$ and $\exists$

Negation: A syntactical operation on formulas defined by de Morgans laws and double negation by

- $\neg(F \wedge G) = \neg F \vee \neg G$
- $\neg(F \vee G) = \neg F \wedge \neg G$
- $\neg\forall x.Fx = \exists x.\neg Fx$
- $\neg\exists x.Fx = \forall x.\neg Fx$
- $\neg\neg F = F$

Then we have as axioms:

$$\Gamma, L, \neg L \text{ where } L \text{ is a literal}$$

and could have $\Gamma, \top$. As rules we have:

$$\frac{\Gamma, F \quad \Gamma, G}{\Gamma, F \wedge G}\ \wedge$$

$$\frac{\Gamma, F, G}{\Gamma, F \vee G}\ \vee$$

$$\frac{\Gamma, Fa}{\Gamma, \forall x.Fx}\ \forall$$

$$\frac{\Gamma, \exists x.Fx, Ft}{\Gamma, \exists x.Fx}\ \exists$$

In rule $\forall$ we assume that the eigen parameter a does not occur in $\Gamma, \forall x.Fx$, while in rule $\exists$ there is a term t. So far the axioms and rules give a system where we build up from simple pieces more and more complicated pieces. Gentzen added to these rules a new rule:

$$\frac{\Gamma, F \quad \Gamma, \neg F}{\Gamma}\ \textbf{cut}$$

We may also have the trivial rule

$$\frac{\Gamma}{\Gamma}\ \textbf{triv}$$

This system is well known as Gentzens sequential calculus. There are a number of variations and refinements of the calculus. If we have equality, then we add as axiom:

$$\Gamma, s = s$$

and as rule

$$\frac{\Gamma, \Gamma(s/t, t/s), s = t}{\Gamma, s = t} =$$

where the sequent $\Gamma(s/t, t/s)$ is obtained from $\Gamma, s = t$ by substituting some s for t and some t for s.

We have also infinitary calculi. The two most important are

Countable conjunctions and disjunctions: We then have as rules

$$\frac{\Gamma, F_k, \bigvee_i F_i}{\Gamma, \bigvee_i F_i} \bigvee$$

and

$$\frac{\cdots \quad \Gamma, F_j \quad \cdots}{\Gamma, \bigwedge_i F_i} \bigwedge$$

ω-logic: The quantifiers are supposed to run over some countable data structure as for example natural numbers. We replace the quantifiers with the appropriate countable conjunctions and disjunctions.

And then many sorted languages and so on and so on.

To get the sequential calculus version of our elementary systems we use predicate logic with equality, extra axioms we have for true Δ_0 sentences F, rules for bounded quantifiers and induction rule. For the data structure of pairs, the induction rule is

$$\frac{\Gamma, \neg Fa, \neg Fb, F(a \cdot b)}{\Gamma, \neg F(nil), Ft} \textbf{ ind}$$

As usual we may have restriction on the induction formula.

The one extraordinary rule in the sequential calculus is the cut rule:

$$\frac{\Gamma, F \quad \Gamma, \neg F}{\Gamma} \textbf{ cut}$$

It is extraordinary on many accounts:

Direct proofs: All the other rules pass from some premisses to a conclusion which is at least as complicated as the premisses. This is not so for the cut rule. Here the cut formula F may be much more complicated than the conclusion Γ.

Mathematical texts: In almost all mathematical texts we have lemmas and other auxiliary notions. We get at these using the cut rule in the form:

$$\frac{\textbf{assumptions} \rightarrow \textbf{lemma} \quad \textbf{assumptions}, \textbf{lemma} \rightarrow \textbf{theorem}}{\textbf{assumptions} \rightarrow \textbf{theorem}}$$

Proof search: Given a conclusion we can search for a possible proof of it. This search stops at the cut rule. There is in general no way to guess the appropriate cut formula.

In our provability logic we used the cut rule to argue for

GL1: $\vdash \Box F \wedge \Box(F \to G) \to \Box G$

7.3 Completeness of the cut free rules

We can do without the cut rule in predicate calculus — the cut free rules are complete. The argument is well known, but is worth repeating. We can look at the rules from conclusion to the premisses as falsification rules. We write $\mathfrak{F} : \Gamma$ for Γ being falsifiable. Then we have

$$\begin{array}{rcl} \mathfrak{F} : \Gamma, F \wedge G & \Leftrightarrow & \mathfrak{F} : \Gamma, F \vee \mathfrak{F} : \Gamma, G \\ \mathfrak{F} : \Gamma, F \vee G & \Leftrightarrow & \mathfrak{F} : \Gamma, F, G \\ \mathfrak{F} : \Gamma, \forall x.F & \Leftrightarrow & \text{there exists } t \text{ with } \mathfrak{F} : \Gamma, Ft \\ \mathfrak{F} : \Gamma, \exists x.F & \Leftrightarrow & \text{for all } t\ \mathfrak{F} : \Gamma, Ft \end{array}$$

Observe that these conditions are very close to the rules in sequential calculus. There are some extra complications related to the quantifiers. The two quantifier rules are:

$$\frac{\Gamma, Fa}{\Gamma, \forall x.Fx}\ \forall$$

$$\frac{\Gamma, \exists x.Fx, Ft}{\Gamma, \exists x.Fx}\ \exists$$

Here the a in $\forall$-rule is a new parameter — we can think of it as a name for the falsifying term in the equivalence above This is of course the crucial step in the completeness theorem. Instead of working directly with the individuals in the falsifying universe, we work with names for the individuals and have a mechanism for supplying with as many names as we want. We may not know what the falsifying term is, but we can readily supply a name for it. The extra problem that a term may have many names are treated separately — this becomes a minor problem when we treat predicate logic with equality and there we have universes built up on equivalence classes of names.

In the $\exists$-rule it is crucial that we repeat the $\exists x.Fx$ in the premiss. We call such a formula for a critical formula, and we must use the $\exists$-rule repeatedly to get at the equivalence above.

Given a sequent Γ that we want to falsify. We write down a falsification tree above Γ as follows:

- start with the sequent Γ at the root.
- a sequent is closed if it is an axiom
- a sequent is open if it contains only literals and is not an axiom
- pick a topmost sequent which is neither open nor closed and use one of the rules to get new sequents above — two if we use $\wedge$-rule else we get one sequent.

The main points in the proof of the completeness theorem is below. The falsification tree is *fair* if all formulas which could be analyzed sooner or later is analyzed. Then

Lemma 5 (Synthesis) *Suppose we have a fair falsification tree over* Γ *with a branch not containing an axiom, then there is a falsification of* Γ*.*

Proof: For the falsification we let

Universe: All terms which occur in the tree

Literals: A literal is interpreted as false if it occurs in the branch

Then by induction over the build up of formulas, we prove that the formulas which occur in the branch are interpreted as false. In particular the formulas in Γ are false. ■

Lemma 6 (Analysis) *Assume we have a falsification of* Γ*. Then every falsification tree over* Γ *contains a branch without any axiom.*

Proof: Assume we have a falsification I of Γ. By induction over the build up of the falsification tree we prove that there is a branch and an extension of the interpretation to the new eigen parameters such that all sequents in the branch are interpreted as false. ■

Theorem 12 (Completeness of predicate logic) *A sequent* Γ *is valid if and only if there is a falsification tree over* Γ *contain axioms in all branches if and only if any fair falsification tree over* Γ *contains axioms in all branches.*

This proof is extended to:

Predicate logic with equality: We change our interpretation to the universe of all terms modulo the equivalence relation given by terms s, t with $\neg s = t$ occurring in the branch.

Countable conjunctions and disjunctions: We must be careful that the disjunctions are analyzed in a fair way.

ω-logic: We must be careful that the existential quantifiers are analyzed in a fair way.

In our falsification procedure we get the term universe in a dynamic way. We get new eigen parameters as we go on in constructing the falsification tree. New eigen parameters are introduced in the analysis of $\forall$, and if we falsify an $\exists\forall$-combination we get a process where we construct an infinite number of eigen parameters. Of course we could have given the term universe in advance. This is — from a falsification point of view — what we do with Skolem functions. We leave to the reader to explore the connection.

Above we did not use use cut in our falsification procedure. We proved completeness for the cut free rules. But look a little closer at the proof. Our falsification is constructed from an open and fair branch in our falsification tree. The falsification comes from assigning false to all literals occurring in the branch. This is sufficient to get that all formulas in the branch are assigned false. But the falsification does not need to assign truth values to all atomic formulas. The falsification is usually partial.

On the other hand we could have used cuts in our falsification procedure. Let us use all possible cuts in a fair way — for each cut formula sooner or later it is used in every branch which does not stop. If we then have an open and fair branch, then the falsification is total. For every formula F : F is false in the falsification if and only if F is in the branch.

Given a theory T and a proof that it is consistent, then we can use the falsification procedure to get an open and fair branch in the falsification tree which gives a model M for T. This model is defined by a $\Delta_2 = \Pi_2 \cap \Sigma_2$ predicate as long as the provability is Σ_1. Truth / falsity in the model is then also Δ_2. This gives another proof of Gödels first incompleteness theorem:

- Assume T is complete and has true axioms.
- Since consistency is true we can prove consistency in T.
- Then we can find a model M for T with a Δ_2 truth predicate for M.
- Since T is complete with true axioms, then true / false in M corresponds to true / false in the real world.
- This is impossible by Tarskis theorem.

Here we needed strong enough theory to get the existence of the falsification branch, and to get the fix point theorem used in the proof of Tarskis theorem.

7.4 Cut elimination

There are two important measures for a proof in sequent calculus:

Height: The height of a proof.

Cut rank: The maximum size of a cut formula in the proof

Lemma 7 *We have syntactical operations as below which neither increase height nor rank:*

$$\begin{array}{rcl} \vdash \Gamma & \Rightarrow & \vdash \Gamma, \Delta \\ \vdash \Gamma, F \wedge G & \Rightarrow & \vdash \Gamma, F \\ \vdash \Gamma, F \wedge G & \Rightarrow & \vdash \Gamma, G \\ \vdash \Gamma, F \vee G & \Rightarrow & \vdash \Gamma, F, G \\ \vdash \Gamma, \forall x.Fx & \Rightarrow & \Gamma, Ft \\ \vdash \Gamma(a) & \Rightarrow & \vdash \Gamma(t) \end{array}$$

We talk about:

Connected: Look at the rules of sequential calculus. An occurrence of a part of Γ (or for rule $\exists$ of $\Gamma, \exists x.Fx$) in the conclusion is connected to the occurrences of the same part in the premisses.

Thread: Reflexive, symmetric, transitive closure of "connected to". So a proof is built up by many threads. Each thread gives a set of occurrences of the same syntactical piece, and uppermost parts of a thread belongs either to an axiom or to a main formula in one of the rules.

Extra eigen parameter condition: Given a set of terms T and a proof of Γ. We can always assume that the eigen parameters of the proof are not included in T. This is done by renaming the eigen parameters of the proof in an appropriate way. The renaming does not affect the height or the cut rank of the proof.

Now to the syntactical transformations. This is all done by changing all occurrences in a thread. For the different cases we do:

- Assume that none of the eigen parameters in the proof of Γ occurs in Δ. Then add Δ to all sequents in the proof.
- Replace each $F \wedge G$ with F in the thread of $F \wedge G$. Applications of the rule $\wedge$ with main formula $F \wedge G$ will then be transformed to an application of the trivial rule of the one premiss and the other premiss is chopped away. All other applications of rules or axioms are untouched by this transformation. Similar for replacing $F \wedge G$ with G.
- Replace each $F \vee G$ with F, G in the thread of $F \vee G$. Applications of the rule $\vee$ with main formula $F \vee G$ will then be transformed to an application of the trivial rule. All other applications of rules or axioms are untouched by this transformation.

- Assume that none of the eigen parameters in the proof occurs in t. Replace each $\forall x.Fx$ with Ft in the thread of $\forall x.Fx$. Applications of the rule $\forall$ with main formula $\forall x.Fx$ will then be transformed to an application of the trivial rule. All other applications of rules or axioms are untouched by this transformation.

- Assume that none of the eigen parameters in the proof occurs in t. Replace each a with t in the proof.

There is no similar syntactic transformation of $\exists x.Fx$. The best we can do is the following:

Lemma 8 (Herbrand) *Assume we have a proof of $\Gamma, \exists x.Fx$ without eigen parameters. Then there are terms $s, t, \ldots, u$ and a syntactical operation which neither increase height nor cut rank with*

$$\vdash \Gamma, \exists x.Fx \Rightarrow \vdash \Gamma, Fs, Ft, \ldots, Fu$$

This is proved as above — note that we must assume that there are no eigen parameters. The problematic case is when we have eigen parameters in the proof. We then have the situation:

$$\dfrac{\dfrac{\Gamma_1, Fs}{\Gamma_1, \exists x.Fx}\ \vdots}{\dfrac{\dfrac{\dfrac{\Gamma_3, Fu}{\Gamma_3, \exists x.Fx}\ \vdots}{\Gamma_2, \exists x.Fx, Ft}}{\Gamma_2, \exists x.Fx}\ \vdots}{\Gamma, \exists x.Fx}$$

There is in general no way that we replace $\exists x.Fx$ with $Fs, Ft, \ldots, Fu$ without violating the eigen value conditions. The terms $s, t, \ldots, u$ may contain terms which are used as eigen variables.

But we are able to eliminate uses of the cut rule. The key lemma is the following

Lemma 9 (Main lemma for cut elimination) *Assume that we have proofs*

$$\vdash \Gamma, H \text{ and } \vdash \Delta, \neg H$$

with cut rank less than rank of F. Then there is a proof

$$\vdash \Gamma, \Delta$$

with cut rank less than rank of H. For the resulting height we have cases depending on the formula H

Literal: *Sum of heights.*

Connectives: *Maximum of heights* $+1$

Quantifiers: *Sum of heights*

Proof:

Literals: We start with the proof of $\vdash \Gamma, L$. Then first replace each L with Δ. This transformation may replace axioms $\Theta, L, \neg L$ with $\Theta, \Delta, \neg L$ but we have already proofs of $\vdash \Delta, \neg L$.

Connectives: Using the lemma of syntactical transformation we have for $H = F \wedge G$

$$\vdash \Gamma, F \text{ and } \vdash \Gamma, G \text{ and } \vdash \Delta, \neg F, \neg G$$

without increasing height or cut rank. Then by using cuts on F and G we get

$$\vdash \Gamma, \Delta$$

with cut rank less than rank of H and height increased by 2.

Quantifiers: We investigate $H = \exists x.Fx$. By the syntactical transformation we have for any term t

$$\vdash \Delta, \neg Ft$$

without increasing height or cut rank. Now look at the proof of $\Gamma, \exists x.Fx$:

$$\dfrac{\dfrac{\Gamma_1, Fs}{\Gamma_1, \exists x.Fx}\ \vdots}{\Gamma, \exists x.Fx} \qquad \dfrac{\dfrac{\dfrac{\Gamma_3, Fu}{\Gamma_3, \exists x.Fx}\ \vdots}{\Gamma_2, \exists x.Fx, Ft}}{\Gamma_2, \exists x.Fx}\ \vdots$$

We now replace the $\exists x.Fx$ with Δ and inferences of $\exists x.Fx$ with some cuts with cut rank the rank of Ft

$$\dfrac{\dfrac{\Gamma_1, Fs \quad \Delta, \neg Fs}{\Gamma_1, \Delta}\ \vdots}{\Gamma, \Delta} \qquad \dfrac{\dfrac{\dfrac{\Gamma_3, Fu \quad \Delta, \neg Fu}{\Gamma_3, \Delta}\ \vdots}{\Gamma_2, \Delta, Ft \qquad \Delta, \neg Ft}}{\Gamma_2, \Delta}\ \vdots$$

The cut rank is not increased, but as height we may get the sum of the two heights. ■

We can measure the cut complexity of a proof by the pair

- the maximal rank of a cut
- the number of cuts of maximal rank

By picking out a topmost cut of maximal rank we can decrease this measure by a proof transformation as long as there are cuts. Furthermore the procedure gives some perspicuous syntactical transformations. To measure the height we introduce

$$\begin{aligned} 2_0^x &= x \\ 2_{n+1}^x &= 2^{(2_n^x)} \end{aligned}$$

Then

Theorem 13 (Cut elimination) *Assume we have a proof of* Γ *with height* h *and cut rank* r*. Then we can transform the proof into a cut free proof of height*

$$2_r^h$$

It may be useful to sum up why we get the super exponential growth of height when we eliminate cuts:

- the cuts are eliminated systematically by using cuts with smaller cut formulas
- we eliminate cuts of connectives anywhere in the proof without really increasing heights
- for the quantifier cuts we eliminate one of those by at most doubling the height
- in the cut elimination we may copy parts of the proof and therefore may multiply the number of cuts
- we are therefore careful that we eliminate cuts in such a way that only smaller cuts are copied
- we can eliminate all cuts of highest rank by working from the top of the proof downwards — no new cuts of highest rank will be introduced in the process
- the estimate of height is given by a recursion over the height of the original proof — at each step we at most double the height of the subtree, and therefore get at most an exponential growth of the original height
- we decrease the cut rank, and get an exponential growth of the original height. By repeating this process and lowering the cut ranks to get rid of all cuts we get a super exponential growth of the original height

Note the following

- the cuts are eliminated in a specific order — taking the topmost cut of maximal rank
- we need extra arguments to conclude that we could have eliminated cuts in a different order

7.5 Notations

A cut free proof can be thought of as a direct proof. An example will make it clearer. Imagine we want to make a notation system for natural numbers. As framework we have

- constant 0
- successor s
- function symbols
- predicate Nx — x is a number
- predicate logic with equality
- axiom NUM — $N0 \wedge \forall x(Nx \to Nsx)$

From ordinary praxis we know that the unary numbers $0\ s0\ ss0\ldots$ are almost impossible to use. Instead we have notations — function symbols defined by recursion equations like

$$\begin{aligned} +0y &= y \\ +sxy &= s + xy \\ *0y &= 0 \\ *sxy &= + * xyy \end{aligned}$$

Here we have used a Polish notation for addition and multiplication. A notation system is a set of equations. It is sound if it can be interpreted in the natural numbers. It is complete if we can derive all true atomic formulas.

A particularly interesting notation system is got from the binary function symbol e and the equations $\star$

$$\begin{aligned} e0y &= sy \\ esxy &= exexy \end{aligned}$$

We interpret exy as $2^x + y$ and observe that $\star$ is sound and complete.

Using terms built up from e, s and 0 we can make short notations for some very large numbers

$$eeeeee0000000 = 2^{2^{16}}$$

is a number much larger than the number of atoms in the universe.

Let us now consider a notation system which is sound and complete. So we have

- axiom NUM — $N0 \wedge \forall x(Nx \to Nsx)$
- recursion equations REC

A cut free proof of NUM,REC $\vdash Nt$ must be of height at least the value of t.

Definition 1 *A term t is inaccessible in a logic if the shortest proof is larger than the value of t.*

Theorem 14 *In cut free predicate logic all terms are inaccessible.*

An important point of a notation t is that we have a short proof of NUM,REC $\vdash Nt$. So without cuts there are no good ways to use a notation system. Consider now the notation system $\star$ for exy written above.

Abbreviate

$$\begin{aligned} N_0x &= Nx \\ N_{n+1}x &= \forall y : N_n \cdot exy : N_n \end{aligned}$$

Lemma 10 *For all n NUM, $\star \vdash N_n0$*

This is immediate for $n = 0$ and $n = 1$. We want to prove

$$\forall y : N_{n+1} \cdot sy : N_{n+1}$$

So assume we have y with $N_{n+1}y$. Then

$$\forall z : N_n \cdot eyz : N_n$$

and

$$\forall z : N_n \cdot eyeyz : N_n$$

specializing z to eyz. Hence $N_{n+1}sy$ and we have proved $N_{n+2}0$ and the lemma.

Using the lemma we get short proofs of NUM, $\star \vdash Neeeeee0000000$. We start with NUM, $\star \vdash N_60$. Then using NUM, $\star \vdash N_50$ we get NUM, $\star \vdash N_5e00$. Going on we get NUM, $\star \vdash N_4ee000$ and in the end NUM, $\star \vdash Neeeeee0000000$. The proof is much shorter than the value of the term. An essential ingredient in the proof is the use of the lemma. This is where the cuts come in.

7.6 Infinitary proofs

So far the proofs are finite syntactic objects. They are used to describe provability. But as we have seen from Gödels incompleteness theorem there is a gap between provability and truth if the universe is one of the common data structures like pairs. By relaxing on the definition of proofs we are able to define truth in a "proof like" way. So assume we have a fixed universe $\mathcal{U}$. Then truth for sentences are defined by an AND-OR tree where at each node we have a sentence in negation normal form (negations innermost)

Conjunction: analyzed by a binary AND-branching

Disjunction: analyzed by a binary OR-branching

For all: analyzed by an AND-branching over each element in $\mathcal{U}$

Exists: analyzed by an OR-branching over each element in $\mathcal{U}$

This is just another way to write down the semantics of sentences — and gives exactly the same. Using sequents we can get rid of the OR-branchings provided

- The sequents are interpreted as finite disjunctions of sentences
- We analyze "exists" within a sequent using a fair procedure

The analysis of exists is done by:

$$\frac{\Gamma, \exists x.Fx, Fu}{\Gamma, \exists x.Fx}$$

The fairness of the procedure consists in that to every element $v \in \mathcal{U}$ sooner or later $\exists x.Fx$ must be analyzed as $\exists x.Fx, Fv$. This is always possible if the universe $\mathcal{U}$ is countable. For uncountable universes the following valid sentence can not be analyzed in a fair way:

$$\forall x : \mathcal{N}.\exists y : \mathcal{N}.F(x, y) \to \exists f : \mathcal{N} \to \mathcal{N}.\forall x : \mathcal{N}.F(x, fx)$$

and the sentence is not derivable. For assume it was derivable. Then the derivation has only countable branchings and involves only countably many inferences. In particular the existential quantifier $\exists f : \mathcal{N} \to \mathcal{N}$ is only analyzed countably many times. But then there must be a countable subset $\mathcal{F} \subseteq \mathcal{F} \to \mathcal{N}$ which contain all instances which are used in the derivation. Therefore the sentence

$$\forall x : \mathcal{N}.\exists y : \mathcal{N}.F(x, y) \to \exists f : \mathcal{F}.\forall x : \mathcal{N}.F(x, fx)$$

is also derivable. But this sentence is not valid as easily proved by a diagonal argument.

For countable universes we have sequential calculi with completeness and cut elimination. The argument for the finitary case is transferred in a direct way to the infinitary case. We shall later see that these calculi is of considerable interest in our analyses of what can be proved in a number of cases. This is a theme which will be taken up after having introduced ordinal notations.

7.7 Direct proofs in syntactic theories

Our proofs in syntactic theories are finite objects. We usually use some form of induction rule. For the theory of pairs we have

$$\frac{\Gamma, \neg Fa, \neg Fb, F(a \cdot b)}{\Gamma, \neg F(nil), Ft}\ \mathbf{ind}$$

The induction rule destroys a number of the useful properties we needed in our cut elimination procedure.

Chapter 8

Ordinals

8.1 Well Orderings

> It must be conceded that Cantor's set theory, and in particular his creation of ordinals, is a grandiose mathematical idea. *Thoralf Skolem [9].*

An ordinal is the order type of a wellordered set. The simplest ordinals are the natural numbers

$$0, 1, 2, 3, 4, 5, \ldots$$

Beyond that we have ω which we can visualize as follows



We observe that the effect of adding an element to the left gives an other order type than adding it to the right





In the first case we have $1+\omega = \omega$, while in the second case we have $\omega+1 > \omega$

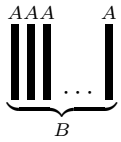
8.2 Arithmetical operations

We can obviously construct some ordinals. The ordinal ω is given by the natural numbers. Addition is performed by putting one ordering after the other:

$A + B$: 

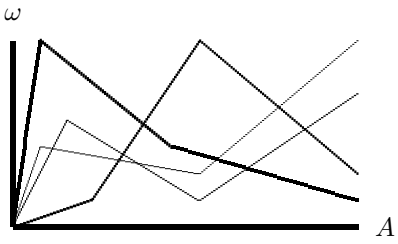
A then B

Multiplication is done by using the lexicographical ordering of pairs:

$A \times B$: 

A copied B times

Addition and multiplication is the same for cardinals and ordinals. Cardinal exponentiation is defined by using all functions, while ordinal exponentiation is defined by all functions with finite support. So to define ω^A we consider functions $f : A \to \omega$ with $fa = 0$ for all but a finite number of the a's. The graphs of such functions are ordered lexicographically — the values of the larger a's being more important.



Note that we define the arithmetical operation without invoking that the orderings are well ordered. For the exponentiation we could have defined B^A as long as B has a least element.

With the arithmetical operations on ordinals we get up to the ordinal ϵ_0 which is defined by

$$\begin{aligned} \omega_1 &= \omega \\ \omega_{n+1} &= \omega^{\omega_n} \\ \epsilon_0 &= \sup_n \omega_n \end{aligned}$$

8.3 Transfinite induction

We can obviously represent the ordinals — say below ϵ_0 — by some syntactical formulation. We suppose this is done in the language, and just write the ordinals instead of their representation. For our purposes in this chapter this is straightforward. We can then define transfinite induction for some formula F up to some ordinal α by

$$\begin{aligned} \mathbf{PROG}(F) &: \forall\beta(\forall\gamma < \beta F(\gamma) \to F(\beta)) \\ \mathbf{TI}(\alpha, F) &: \mathbf{PROG}(F) \to \forall\beta < \alpha F(\beta) \end{aligned}$$

The first property is called progressive. The property of being progressive subsumes the usual assumption in induction formulated as a course of values induction. $\mathbf{TI}(\omega, F)$ corresponds to induction over natural numbers and is provable in Peano arithmetic. We now want to prove that transfinite induction is closed under addition, multiplication and exponentiation. This is done by taking transfinite induction over some more complicated properties.

Addition. We want to prove $\mathbf{TI}(\alpha + \beta, F)$. So assume we have

$$\mathbf{PROG}(F)$$

By $\mathbf{TI}(\alpha, F)$ we get

$$\forall x < \alpha.F(x)$$

But then $\mathbf{PROG}(\lambda x.F(\alpha + x))$.
By $\mathbf{TI}(\beta, \lambda x.F(\alpha + x)$ we get

$$\forall x < \beta.F(\alpha + x)$$

and hence

$$\forall x < \alpha + \beta.F(x)$$

So we have proved $\mathbf{TI}(\alpha + \beta, F)$ from $\mathbf{TI}(\alpha, F)$ and $\mathbf{TI}(\beta, \lambda x.F(\alpha + x)$.

Multiplication. We want to prove $\mathbf{TI}(\alpha \times \beta, F)$.
So assume we have $\mathbf{PROG}(F)$, $\beta_0 < \beta$ and $\forall x < \alpha \times \beta_0.F(x)$.
Using the assumption we get $\mathbf{PROG}(\lambda x.F(\alpha \times \beta_0 + x))$
and by $\mathbf{TI}(\alpha, \lambda x.F(\alpha \times \beta_0 + x)$ we get

$$\forall x < \alpha.F(\alpha \times \beta_0 + x)$$

This gives $\mathbf{PROG}(\lambda y.F(\alpha \times y)$ and using

$$\mathbf{TI}(\beta, \lambda y.F(\alpha \times y))$$

we get $\forall z < \alpha \times \beta.F(z)$.
We prove $\mathbf{TI}(\alpha \times \beta, F)$ using $\mathbf{TI}(\alpha, \lambda x.F(\alpha \times \beta_0 + x)$ and $\mathbf{TI}(\beta, \lambda y.F(\alpha \times y))$.

2-Exponentiation. We want to prove $\mathbf{TI}(2^\alpha, F)$
For the proof we need the following easily proved fact about ordinals

$$\gamma + \beta < \gamma + 2^\alpha \Rightarrow \exists\alpha_0 < \alpha.\gamma + \beta < \gamma + 2^{\alpha_0} \times 3$$

Consider the more abstract property

$$F^*(x) : \forall y.(\forall z < y.F(z) \rightarrow \forall z < y + 2^x.F(z))$$

We first prove

$$\mathbf{PROG}(F) \rightarrow \mathbf{PROG}(F^*)$$

So let $\mathbf{PROG}(F)$, α and $\forall x < \alpha.F^*(x)$ be given. We want to prove $F^*(\alpha)$.
Let β and $\forall z < \beta.F(z)$ be given. We want to prove $\forall z < \beta + 2^\alpha.F(z)$.
Let $\gamma < \beta + 2^\alpha$ be given. Then there is $\alpha_0 < \alpha$ with $\gamma < \beta + 2^{\alpha_0} \times 3$.
Using $F^*(\alpha_0)$ we get $\forall y.(\forall z < y.F(z) \rightarrow \forall z < y + 2^{\alpha_0}.F(z))$.
Applying it twice we get

$$\forall y.(\forall z < y.F(z) \rightarrow \forall z < y + 2^{\alpha_0} + 2^{\alpha_0} + 2^{\alpha_0} = y + 2^{\alpha_0} \times 3.F(z))$$

Therefore we have proved $F(\gamma)$, $\forall z < \beta + 2^\alpha.F(z)$, $F^*(\alpha)$ and $\mathbf{PROG}(F^*)$.
Now we prove

$$\mathbf{TI}(\alpha, F^*) \rightarrow \mathbf{TI}(2^\alpha, F)$$

So assume $\mathbf{TI}(\alpha, F^*)$ and $\mathbf{PROG}(F)$. Then $\mathbf{PROG}(F^*)$ and we get

$$\forall x < \alpha.F^*(x)$$

and $F^*(\alpha)$ which is

$$\forall y.(\forall z < y.F(z) \rightarrow \forall z < y + 2^\alpha, F(z))$$

Substituting $y = 0$ we get

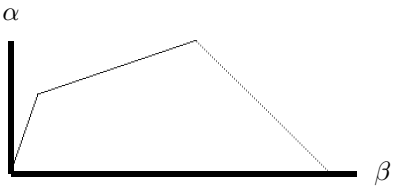
$$\forall z < 2^\alpha.F(z)$$

and the proof is finished.

Exponentiation. We want to prove $\mathbf{TI}(\alpha^\beta, F)$ for $\alpha > 0$.
We need another fact about ordinals:

$$\delta < \gamma + \alpha^\beta \rightarrow \exists \alpha_o < \alpha.\exists \beta_0 < \beta.\delta < \alpha^{\beta_0} \times (\alpha_0 + 1)$$

Take the case $\gamma = 0$ and look at the graph of δ



Let β_0 be the largest non-zero point in the graph and $\alpha_0 = \delta(\beta_0)$.

We now introduce a new concept

$$F^{\spadesuit}(x) : \forall y.(\forall z < y.F(z) \rightarrow \forall z < y + \alpha^x.F(z))$$

We first prove

$$\mathbf{PROG}(F) \rightarrow \mathbf{PROG}(F^{\spadesuit})$$

So let $\mathbf{PROG}(F)$, β and $\forall x < \beta.F^{\spadesuit}(x)$ be given. We want to prove $F^{\spadesuit}(\beta)$.
Let γ and $\forall z < \gamma.F(z)$ be given. We want to prove $\forall z < \gamma + \alpha^{\beta}.F(z)$.
Let $\delta < \gamma + \alpha^{\beta}$ be given.
Then there are $\beta_0 < \beta$ and $\alpha_0 < \alpha$ with $\delta < \gamma + \alpha^{\beta_0} \times (\alpha_0 + 1)$.
Using $F^{\spadesuit}(\beta_0)$ we get $\forall y.(\forall z < y.F(z) \rightarrow \forall z < y + \alpha^{\beta_0}.F(z))$.
We use this in $\mathbf{TI}(\alpha + 1, \lambda u.\forall z < \gamma + \alpha^{\beta_0} \times u).F(z)$ and prove $F(\delta)$.
Therefore $\forall z < \gamma + \alpha^{\beta}.F(z)$, $F^{\spadesuit}(\beta)$ and $\mathbf{PROG}(F^{\spadesuit})$.
Now we prove

$$\mathbf{TI}(\beta, F^{\spadesuit}) \rightarrow \mathbf{TI}(\alpha^{\beta}, F)$$

So assume $\mathbf{TI}(\beta, F^{\spadesuit})$ and $\mathbf{PROG}(F)$. Then $\mathbf{PROG}(F^{\spadesuit})$ and we get

$$\forall x < \beta.F^{\spadesuit}(x)$$

and $F^{\spadesuit}(\beta)$ which is

$$\forall y.(\forall z < y.F(z) \rightarrow \forall z < y + \alpha^{\beta}, F(z))$$

Substituting $y = 0$ we get

$$\forall z < \alpha^{\beta}.F(z)$$

and the proof is finished.

Discussion. We have proved that transfinite induction over ordinals are closed under the usual arithmetical operations. This get us up to every ordinal less than ϵ_0. For addition and multiplication we only need to have induction over some slightly more complicated predicates. The big step is with the exponentiation. We have included both 2^{α} and α^{β} above. But the last does not really bring anything new. Observe that we get transfinite induction over exponentiation by having extra quantifiers in the new predicates. Here there is a tradeoff and we leave to the reader to find good theorems expressing this.

8.4 Ordinal bounds

We now want to do two different things

- in this section we show that the ordinal ϵ_0 occur as a bound for heights when we eliminate cuts in infinitary calculus

- in the next section we show that ϵ_0 is a bound for the transfinite induction we can prove in arithmetic

We try to make the argument robust — the details involved can be done in many ways. We have a sequential calculus with an ω-rule

$$\frac{\Gamma, F0 \quad \Gamma, F1 \quad \Gamma, F2 \quad \ldots}{\Gamma, \forall x.Fx}$$

As usual we have a cut rule which we try to eliminate in derivations

$$\frac{\Gamma, F \quad \Gamma, \neg F}{\Gamma}$$

This elimination increases the height of the derivation. In the elimination procedure we look at

- the height of the derivation
- the cut ranks — that is the nesting of quantifiers in the cut formulas

The procedure works in many passes. In each pass we start from the top of the derivation and works downwards. In the first pass we eliminate cuts with maximal rank, in the next pass cuts with second heighest rank and so on. Now we have the following observations

- we can eliminate outermost conjunctions and disjunctions in a cutformula with just slightly increase of height
- we can eliminate outermost quantifier in a cutformula which is topmost and with maximal rank with at most a doubling of the height
- we can eliminate atomic cuts

The crucial elimination is the one for the quantifiers. Let us repeat how this is done. We start with a cut

$$\frac{\Gamma, \exists x.Fx \quad \Delta, \forall x.\neg Fx}{\Gamma, \Delta}$$

We first observe that we have the following transformation for universal quantifier

$$\vdash \Delta, \forall x.\neg Fx \Rightarrow \vdash \Delta, \neg Ft \text{ where } t \text{ is any term}$$

and that this transformation neither increase the height nor the cut rank of the derivation. This is simply proved by induction over the height. We now look at the existential quantifier. A typical derivation of $\Gamma, \exists x.Fx$ looks like

$$\dfrac{\dfrac{\Gamma_1, Fs}{\Gamma_1, \exists x.Fx}\ \vdots}{\Gamma, \exists x.Fx} \qquad \dfrac{\dfrac{\dfrac{\Gamma_3, Fu}{\Gamma_3, \exists x.Fx}\ \vdots}{\Gamma_2, \exists x.Fx, Ft}}{\Gamma_2, \exists x.Fx}\ \vdots$$

We now replace the $\exists x.Fx$ with Δ and inferences of $\exists x.Fx$ with some cuts with cut rank the rank of Ft

$$\dfrac{\dfrac{\Gamma_1, Fs \quad \Delta, \neg Fs}{\Gamma_1, \Delta}\ \vdots}{\Gamma, \Delta} \qquad \dfrac{\dfrac{\dfrac{\Gamma_3, Fu \quad \Delta, \neg Fu}{\Gamma_3, \Delta}\ \vdots}{\Gamma_2, \Delta, Ft \qquad \Delta, \neg Ft}}{\Gamma_2, \Delta}\ \vdots$$

The cut rank is not increased, but as height we may get the sum of the two heights. It was necessary here to consider topmost cut of maximal rank. We put in many copies of the inferences involving $\Delta, \forall x.\neg Fx$ and we must be sure that they do not contain cutformulas of higher rank. Here we eliminate a topmost cut of maximal rank.

We eliminate all cuts of maximal rank by increasing the height exponentially $\alpha \mapsto 2^\alpha$. Then we have to make new passes for the second highest rank, the third highest rank and so on. For each pass we get an exponetial jump in the height. If the original derivation was of infinite height but less than ϵ_0, then the cutfree derivation is of height $< \epsilon_0$.

We now come to arithmetic. We use ω-logic to handle induction. Let us say we have an arithmetical system with induction

$$\dfrac{F0 \quad \forall y.(\neg Fy \vee Fsy)}{\forall x.Fx}$$

We can replace it with an instance of the ω-rule and many cuts

$$\dfrac{F0 \qquad \dfrac{F0 \quad \neg F0, F1}{F1} \qquad \dfrac{\dfrac{F0 \quad \neg F0, F1}{F1} \qquad \neg F1, F2}{F2} \qquad \ldots}{\forall x.Fx}$$

This transforms a derivation in first order arithmetic of finite height and finite cut rank into a derivation of finite cut rank and height $< \omega^2$. After cutelimination we get a deivation of height $< \epsilon_0$. We leave to the reader to get more exact estimates.

8.5 Bounds for provable transfinite induction

Transfinite induction is given by

$$\mathbf{PROG}(F) : \forall x.(\forall y < x.Fy \to Fx)$$

$$\mathbf{TI}(\alpha, F) : \mathbf{PROG}(F) \to \forall x < \alpha.Fx$$

Let us now assume that we have a system for first order arithmetic with ordinals and one special predicate letter F. We have shown that we are able to prove

$$\mathbf{TI}(\alpha, F) \text{ where } \alpha < \epsilon_0$$

In this section we shall prove that we cannot do better. To do this we introduce a new system with the following infinitary rule, called progressionrule, for each ordinal α

$$\frac{F0 \quad F1 \quad \ldots \quad F\beta \quad \ldots}{F\alpha} \text{ where } \beta < \alpha$$

In this system $\mathbf{PROG}(F)$ is derivable. Furthermore the system with ω-rule and progressionrule admits cut elimination and the estimates of height are the same. The only place where the new rule can be involved in cuts are with atomic cuts and cutformula F.

Assume now that we have proved in the ordinary first order theory

$$\mathbf{TI}(\alpha, F)$$

Then we can imbed this derivation into the system with ω-rule and progressionrules. There we can derive

$$F\alpha$$

This derivation is of height $< \omega^2$ and of finite cut rank. The cut elimination gives a cutfree derivation of height $< \epsilon_0$. Now look at such a cut free derivation of $F\alpha$. This derivation can only contain progressionrules and we see that α must be less or equal to its height. Therefore

$$\alpha < \epsilon_0$$

This is interesting. It gives us a concrete Gödel sentence. The sentence $\mathbf{TI}(\epsilon_0, F)$ is not provable in first order arithmetic — and it is true.

Chapter 9

Quantifiers

9.1 Skolem

With quantifiers we can express complicated properties with almost no effort. This is of course both good and bad news. The good news is that the language is very expressive. The bad news is that there are almost no new information involved.

Thoralf Skolem was especially concerned with the role of the quantifiers. He introduced (his variant of) the analyzing tree with new names introduced for the $\forall$-quantifiers and the $\exists$-quantifiers as source for terms. In this way he proved the mathematical core of the completeness theorem in 1922. He was also the first to prove that in some cases quantifiers could be eliminated. In 1919 he read Russell & Whiteheads "Principia Mathematica". He disagreed with the claims there for having made a foundation for mathematics. Instead he sketched in 1923 an alternative foundation. He introduced

- a rich system of terms — the primitive recursive functions
- a logic for proving properties of such terms — quantifierfree arithmetic

This was used to give a development of arithmetic. In more modern terms we could say that he introduced simultaneously a programming language and a programming logic. Both for the first time. The programming language was the language of primitive recursive functions. The logic is interesting here. He had free variable statements like we are used to in mathematics

$$a + b \quad = \quad b + a$$

and for the proofs he used the induction rule

$$\vdash F0 \ , \vdash Fa \rightarrow Fsa \Rightarrow \vdash F\phi$$

with the predicate Fa quantifierfree but could contain other free variables. This way of thinking is of course quite close to mathematical practice — and in my opinion Skolem made a very good point against Russell & Whitehead.

In a free variable theory we can interprete some quantifier combinations. The statement

$$\forall x \cdot Fx$$

corresponds to

$$Fa$$

with a a free variable. And

$$\forall x \cdot \exists y \cdot Gxy$$

corresponds to

$$Ga\phi a$$

for some free variable a and some term ϕa .

9.2 Gödel

In 1942 Kurt Gödel introduced an interesting way of interpreting theories of arithmetic into quantifier free theories. Gödel lectured about the results, but is was forgotten and revived with a publication in 1958 in the journal Dialectica. The plan now is to go through Gödels Dialectica interpretation. Then we'll go through some possible rules in arithmetic and see what kind of terms are needed for the corresponding quantifier free theory $\mathcal{T}$. The terms in $\mathcal{T}$ are functionals of finite type over the type of natural numbers $\mathcal{N}$. The theory $\mathcal{T}$ will be a free variable theory built on

- classical propositional calculus
- cutrule
- quantifier free induction
- equality in the ground type
- recursion equations for the terms introduced

Gödel used intuitionistic logic, but his interpretation works equally well in classical logic. It becomes particularly simple if we restrict the logical language to the symbols $\neg$, $\wedge$, $\vee$ and $\forall$. In the development below we will go through the rules of arithmetic and show how the interpretation into $\mathcal{T}$ propagates starting from interpretation of axioms and the go from interpretation of premisses of

rules to interpretation of their conclusion. As a result we get an interpretation for each sequent derivable in arithmetic.

First we must tell what an interpretation of a sequent in arithmetic is. To each formula F we assign a formula $F^{\#}$ of form

$$\forall a \cdot \exists b \cdot Rab$$

where Rab is a formula of $\mathcal{T}$ and a and b are functionals of higher type.

For atomic F

$$F^{\#} \quad = \quad F$$

Assume

$$\begin{aligned} F^{\#} &= \forall a \cdot \exists b \cdot Rab \\ G^{\#} &= \forall c \cdot \exists d \cdot Scd \\ Hx^{\#} &= \forall e \cdot \exists f \cdot Tefx \end{aligned}$$

Then

$$\begin{aligned} (\neg F)^{\#} &= \forall B \cdot \exists a \cdot \neg RaBa \\ (F \wedge G)^{\#} &= \forall ac \cdot \exists bd \cdot (Rab \wedge Scd) \\ (F \vee G)^{\#} &= \forall ac \cdot \exists bd \cdot (Rab \vee Scd) \\ (\forall x \cdot Hx)^{\#} &= \forall ex \cdot \exists f \cdot Tefx \end{aligned}$$

Observe the way negation is treated. Here we go up in the type hierarchy. This has of course consequences for the treatment of existential quantifiers. Using the above we translate

$$\begin{aligned} &\exists a \forall b \exists c \cdot R\ a\ b\ c \\ &\neg \forall a \neg \forall b \exists c \cdot R\ a\ b\ c \\ &\neg \forall a \forall C \exists b \cdot \neg R\ a\ b\ Cb \\ &\forall B \exists a \exists C \cdot R\ a; BaC\ C(BaC) \end{aligned}$$

In the above translation we delete all quantifiers which are not referred to. Using this we get that any formula of form $\forall\exists$ is translated as itself. Only with quantifier combination $\exists\forall$ do we get something new.

Now we show how to treat sequents. Consider — to simplify our explanation — the sequent $\Gamma = F, G$ with $F^{\#}$ and $G^{\#}$ as above. We say that Γ *can be interpreted in* $\mathcal{T}$ if there are terms ϕ and ψ such that

$$\vdash_{\mathcal{T}} Ra\phi ac\ ,\ Sc\psi ac$$

Note that as arguments in both ϕ and ψ we may have all free variables in the sequent. We will also use the more compact notation

$$\vdash_{\mathcal{T}} Rab^{\star}\ ,\ Scd^{\star}$$

where the $\star$ indicates that we have terms with variables from the free variables in the sequent. For sequents with more formulas we do the same as above.

9.3 Interpretability

Definition 2 *A rule is interpretable if from an interpretation of the premisses we get an interpretation of the conclusion. An axiom is interpretable if it can be interpreted.*

Theorem 15 *Axioms in arithmetic are interpretable.*

All literals — in fact all quantifierfree formulas — are interpreted by themselves. Axioms in arithmetic are transferred to axioms in $\mathcal{T}$. Now to the rules

Theorem 16 (Contraction) $\vdash_{\mathcal{T}} \Gamma, F, F \Rightarrow \vdash_{\mathcal{T}} \Gamma, F$ *is interpretable.*

Let $F^{\#} = \forall a \cdot \exists b \cdot Rab$. By assumption we have in $\mathcal{T}$

$$\vdash_{\mathcal{T}} \Pi, Rab^{\star}, Rcd^{\star}$$

where $b^{\star} = \phi ac$ and $d^{\star} = \psi ac$. Substitute c for a and define by cases

$$\xi a = \begin{cases} \phi aa & \text{if } Ra\phi aa \\ \psi aa & \text{if } \neg Ra\phi aa \end{cases}$$

Then

$$\vdash_{\mathcal{T}} \Pi, Ra\xi a$$

Theorem 17 (Conjunction) $\vdash \Gamma, F \,\&\, \vdash \Gamma, G \ \Rightarrow \vdash \Gamma, F \wedge G$ *is interpretable.*

Assume that in $\mathcal{T}$ we have $\vdash_{\mathcal{T}} \Pi, Rab^{\star}$ and $\vdash_{\mathcal{T}} \Pi, Scd^{\star}$. Then

$$\vdash_{\mathcal{T}} \Pi, Rab^{\star} \wedge Scd^{\star}$$

Theorem 18 (Negation-Conjunction) $\vdash\ \Gamma, \neg F \Rightarrow \vdash\ \Gamma, \neg(F \wedge G)$ *is interpretable.*

$$\begin{aligned} (\neg F)^{\star} &= \forall B \cdot \exists a \cdot \neg RaBa \\ (\neg(F \wedge G))^{\star} &= \forall BD \cdot \exists ac \cdot \neg(RaBac \wedge ScDac) \end{aligned}$$

We assume in $\mathcal{T}$ $\vdash_{\mathcal{T}} \Pi, \neg Ra^{\star}Ba^{\star}$. But then for a freely chosen term $c^{\star}$ $\vdash_{\mathcal{T}} \Pi, \neg(Ra^{\star}Ba^{\star} \wedge Sc^{\star}Dc^{\star})$. A similar argument for the case where we permute F and G.

Theorem 19 (Disjunction) $\vdash \Gamma, F \Rightarrow \vdash \Gamma, F \vee G$ *is interpretable.*

Similar to negation-conjunction.

Theorem 20 (Negation-disjunction) $\vdash \Gamma, \neg F \;\&\; \vdash \Gamma, \neg G \;\Rightarrow \vdash \Gamma, \neg(F \vee G)$ *is interpretable.*

Similar to conjunction.

Theorem 21 (Double negation) $\vdash \Gamma, F \Rightarrow \vdash \Gamma, \neg\neg F$ *is interpretable.*

$$\begin{aligned} F^{\#} &= \forall a \cdot \exists b \cdot R\, a\, b \\ (\neg F)^{\#} &= \forall B \cdot \exists a \cdot \neg R\, a\, B a \\ (\neg\neg F)^{\#} &= \forall A \cdot \exists B \cdot \neg\neg R\, AB\, B(AB) \end{aligned}$$

Assume $\vdash_{\mathcal{T}} \Pi, R\, a\, b^{\star}$ where $b^{\star} = \phi a$. Define $B^{\star} = \phi$ and substitute $A\phi$ for a. Then

$$\begin{array}{l} \vdash_{\mathcal{T}} \Pi, R\, A\phi\, \phi(A\phi) \\ \vdash_{\mathcal{T}} \Pi, R\, AB^{\star}\, B^{\star}(AB^{\star}) \\ \vdash_{\mathcal{T}} \Pi, \neg\neg R\, AB^{\star}\, B^{\star}(AB^{\star}) \end{array}$$

Theorem 22 ($\forall$-quantifier) $\vdash \Gamma, Fa \Rightarrow \vdash \Gamma, \forall x \cdot Fx$ *is interpretable.*

$$\begin{aligned} (Fa)^{\#} &= \forall b \cdot \exists c \cdot Rabc \\ (\forall x \cdot Fx)^{\#} &= \forall ab \cdot \exists c \cdot Rabc \end{aligned}$$

Both are interpreted in the same way.

Theorem 23 (Negation-$\forall$) $\vdash \Gamma, \neg F\phi \Rightarrow \vdash \Gamma, \neg\forall x \cdot Fx$ *is interpretable.*

$$\begin{aligned} (Fa)^{\#} &= \forall b_0 \cdot \exists c \cdot R\, a\, b_0\, c \\ (\neg Fa)^{\#} &= \forall C \cdot \exists b_0 \cdot \neg R\, a\, b_0\, C b_0 \\ (\forall x \cdot Fx)^{\#} &= \forall a b_1 \cdot \exists c \cdot R\, a\, b_1\, c \\ (\neg\forall x \cdot Fx)^{\#} &= \forall D \cdot \exists a b_1 \cdot \neg R\, a\, b_1\, D a b_1 \end{aligned}$$

We assume $\vdash_{\mathcal{T}} \Pi, \neg R\, \phi\, b_0^{\star}\, C b_0^{\star}$. Then define $a^{\star} = \phi$ and $b_1^{\star} = b_0^{\star}$. Then substitute $\lambda b \cdot Da^{\star} b$ for C to get $\vdash_{\mathcal{T}} \Pi, \neg R\, a^{\star}\, b_1^{\star}\, Da^{\star} b_1^{\star}$.

Theorem 24 (Cut) $\vdash \Gamma, F \;\&\; \vdash \Gamma, \neg F \Rightarrow \Gamma$ *is interpretable.*

$$\begin{aligned} F^{\#} &= \forall a \cdot \exists b \cdot R\, a\, b \\ (\neg F)^{\#} &= \forall B \cdot \exists a \cdot \neg R\, a\, B a \end{aligned}$$

Assume $\vdash_{\mathcal{T}} \Pi, R\, a\, b^{\star}$ and $\vdash_{\mathcal{T}} \Pi, \neg R\, a^{\star}\, B a^{\star}$ where $b^{\star}$ may contain a and the free variables of Π and $a^{\star}$ may contain B and the free variables of Π. We define $\phi = \lambda a \cdot b^{\star}$ and $\psi = \lambda B \cdot a^{\star}$. Both contain only free variables from Π. Substitute ϕ for B and $\psi\phi$ for a. Then

$$\begin{aligned} &\vdash_{\mathcal{T}} \Pi, R\, \psi\phi\, \phi(\psi\phi) \\ &\vdash_{\mathcal{T}} \Pi, \neg R\, \psi\phi\, \phi(\psi\phi) \\ &\vdash_{\mathcal{T}} \Pi \end{aligned}$$

Theorem 25 (Induction) $\vdash \Gamma, F0$ & $\vdash \Gamma, \neg Fa, Fsa \Rightarrow \vdash \Gamma, F\phi$ *is interpretable.*

$$\begin{aligned} (Fa)^{\#} &= \forall b \cdot \exists c \cdot R\, a\, b\, c \\ \neg Fa)^{\#} &= \forall C \cdot \exists c \cdot \neg R\, a\, b\, C b \end{aligned}$$

Assume $\vdash_{\mathcal{T}} \Pi, R\, 0\, b_0\, c_0^{\star}$ and $\vdash_{\mathcal{T}} \Pi, \neg R\, a\, b^{\star}\, C b^{\star}, R\, sa\, b_1\, c_1^{\star}$. Define $\phi = \lambda b_0 \cdot c_0^{\star}$, $\psi = \lambda a C b_1 \cdot b^{\star}$ and $\xi = \lambda a C b_1 \cdot c_1^{\star}$. Note that a is of type natural numbers.

We use it as recursion variable in defining ζ by primitive recursion

$$\begin{aligned} \zeta 0 &= \phi \\ \zeta sa &= \xi a sa \end{aligned}$$

Note that the typing is correct. Substitute ζa for c and define $b^{\star\star} = \psi a(\zeta a) b_1$. Then

$$\begin{aligned} &\vdash_{\mathcal{T}} \Pi, R\, 0\, b\, \zeta 0 b \\ &\vdash_{\mathcal{T}} \Pi, \neg R\, a\, b^{\star\star}\, \zeta a b^{\star\star}, R\, sa\, b\, \xi(\zeta a) b \\ &\vdash_{\mathcal{T}} \Pi, \neg R\, a\, b^{\star\star}\, \zeta a b^{\star\star}, R\, sa\, b\, \zeta sab \\ &\vdash_{\mathcal{T}} \Pi, R\, \phi\, b\, \zeta \phi b \end{aligned}$$

This concludes the proof of

Theorem 26 (Dialectica interpretation) *Every sequent derivable in Peano arithmetic can be interpreted in Gödels $\mathcal{T}$.*

9.4 Discussion

There are three ingredients in Gödels interpretation

- the elementary arithmetical system $\mathcal{P}$

- the free variable system $\mathcal{T}$
- the primitive recursive terms of finite type

The system $\mathcal{P}$ is ordinary Peano arithmetic. Complicated quantifier combinations in $\mathcal{P}$ are traded off for complicated terms. Note that quantifier free formulas in $\mathcal{P}$ are interpreted by themselves and we do not need any new terms. This gives

Theorem 27 *Consistency of $\mathcal{T}$ implies consistency of $\mathcal{P}$.*

Gödel called his 1958 paper "Über eine bisher noch nicht benutzte Erweiterung des finiten Standpunktes." He thought — and many constructivists agreed — that the consistency of $\mathcal{T}$ is evident from a finite point of view.

Now look at $\mathcal{T}$ together with the term system. To interpret the rules of $\mathcal{P}$ we need some properties of $\mathcal{T}$. First of all we needed the properties of typed λ-calculus

- explicit definition
- definition by cases
- λ-abstraction
- composition

Then to the induction rule $\vdash_{\mathcal{T}} \Gamma, F0$ & $\vdash_{\mathcal{T}} \Gamma, \neg Fa, Fsa \Rightarrow \vdash_{\mathcal{T}} \Gamma, F\phi$. The treatment of this depends on the quantifiers in Fa. There are three special cases

Quantifier free We do not need any special terms to interprete the rule.

Universal We do not need any special terms to interprete the rule.

Existential We need ordinary primitive recursive functions.

For the general case we need primitive recursive functionals of finite type.

9.5 Skolem functions

Why use Gödels complicated interpretation of the quantifiers? One could have tried to interprete $\forall a \cdot \exists b \cdot F\,a\,b$ by a term B and $\forall a \cdot F\,a\,Ba$ for all formulas F — and not only quantifier free formulas. From recursion theory we know that for quantifier free F we can assume that B is computable. But this is not longer true if F contains quantifiers.

Let Rab be quantifier free. Then

$$\vdash \forall a \cdot \exists c \cdot (\exists b\ Rab \rightarrow Rac)$$

Assume now that there is a term ϕ interpreting c. So

$$\vdash \forall a \cdot (\exists b\ Rab \rightarrow Ra\,\phi a)$$

But obviously

$$\vdash \forall a \cdot (\exists b\ Rab \leftarrow Ra\,\phi a)$$

and

$$\vdash \forall a \cdot (\exists b\ Rab \leftrightarrow Ra\,\phi a)$$

But then ϕ cannot in general be computable — we do not have a computable way to decide whether $\exists b\ Rab$.

On the other hand there is a conceptually simpler interpretation than the one that Gödel gave. Georg Kreisel introduced the no counter example interpretation. Let F be an arithmetical formula in prenex form. In ordinary predicate logic we have $\vdash F \rightarrow F$. But then we can trade the $\forall$-quantifiers in the last F with function symbols $f_1, \ldots, f_m$ to get in predicate logic

$$\vdash F \rightarrow \exists y_1 \ldots y_n \cdot Rf_1 \ldots f_m y_1 \ldots y_n$$

where R is quantifier free. Now assume in elementary arithmetic $\vdash_{\mathcal{A}} F$. Then

$$\vdash_{\mathcal{A}} \exists y_1 \ldots y_n \cdot Rf_1 \ldots f_m y_1 \ldots y_n$$

The Gödel interpretation works equally well with the new function symbols. So there are terms $c_1^\star, \ldots, c_n^\star$ with

$$\vdash_{\mathcal{T}} \cdot Rf_1 \ldots f_m c_1^\star \ldots c_n^\star$$

This is the no counter example interpretation. The new function symbols works as giving possible counterexamples as we noted in the previous chapter. Let us give two examples

$$\begin{aligned} F &= \exists x \forall y \exists z \cdot Rxyz \\ G &= \forall x \exists y \forall z \cdot Sxyz \end{aligned}$$

with R and S quantifier free. They are then interpreted as

$$\begin{gathered} \exists xz \cdot R\,x\,f x\,z \\ \exists y \cdot S\,g\,y\,hy \end{gathered}$$

And if $\vdash_{\mathcal{A}} F$ and $\vdash_{\mathcal{A}} G$, then

$$\begin{gathered} \vdash_{\mathcal{T}} R\,a^\star\,f a^\star\,c^\star \\ \vdash_{\mathcal{T}} R\,g\,b^\star\,hb^\star \end{gathered}$$

The interpreting terms $a^\star$ and $c^\star$ depend on the free variable f, and the interpreting term $b^\star$ depends on the free variables g and h.

9.6 Spector

In 1960 Clifford Spector generalized Gödels interpretation to full second order arithmetic. He showed

Theorem 28 (Spector) $\vdash \Gamma, \forall a \exists b \cdot R\,a\,b \Rightarrow \vdash \Gamma, \exists f \forall a \cdot R\,a\,f\,a$ *where* a *is of type natural number is interpretable.*

$$\begin{aligned}
(Rab)^{\#} &= \forall c \cdot \exists d \cdot S\,a\,b\,c\,d \\
(\forall a \exists b \cdot Rab)^{\#} &= \forall a C_0 \cdot \exists b D_0 \cdot S\ a\ b\ C_0 b D_0\ D_0(C_0 b D_0) \\
(\exists f \forall a \cdot Rab)^{\#} &= \forall A C_1 \cdot \exists f D_1 \cdot S\ A f D_1\ f(A f D_1)\ C_1 f D_1\ D_1(A f D_1)(C_1 f D_1)
\end{aligned}$$

There are terms $b^\star$ and $D_0^\star$ such that

$$\vdash_{\mathcal{T}} S\ a\ b^\star\ C_0 b^\star D_0^\star\ D_0^\star(C_0 b^\star D_0^\star)$$

Must find terms $f^\star$ and $D_1^\star$ such that

$$\vdash_{\mathcal{T}} S\ A f^\star D_1^\star\ f^\star(A f^\star D_1^\star)\ C_1 f^\star D_1^\star\ D_1^\star(A f^\star D_1^\star)(C_1 f^\star D_1^\star)$$

using $b^\star$, $D_0^\star$ and appropriate substitutions of the free variables a and C_0. We must solve the equations

$$\begin{aligned}
a &= A f^\star D_1^\star \\
b^\star &= f^\star(A f^\star D_1^\star) \\
C_0 b^\star D_0^\star &= C_1 f^\star D_1^\star \\
D_0^\star(C_0 b^\star D_0^\star) &= D_1^\star(A f^\star D_1^\star)(C_1 f^\star D_1^\star)
\end{aligned}$$

It is sufficient to solve

$$\begin{aligned}
a &= A f^\star D_1^\star \\
b^\star &= f^\star(A f^\star D_1^\star) \\
C_0 b^\star D_0^\star &= C_1 f^\star D_1^\star \\
D_0^\star &= D_1^\star(A f^\star D_1^\star)
\end{aligned}$$

Introduce the pairs $e^\star = (b^\star, D_0^\star)$ and $g^\star = (f^\star, D_1^\star)$. Must solve

$$\begin{aligned}
a &= A g^\star \\
e^\star &= g^\star(A g^\star) \\
C_0 e^\star &= C_1 g^\star
\end{aligned}$$

Here a is of type natural number. We have given the term $e^\star$ with free variables a and C_0. Must find term $g^\star$ depending on A and C_1 and appropriate substitutions for a and C_0 such that the equations hold. We define $\epsilon = \lambda a C_0 \cdot e^\star$ and get

$$\begin{aligned} a &= Ag^\star \\ \epsilon a C_0 &= g^\star(Ag^\star) \\ C_0 e^\star &= C_1 g^\star \end{aligned}$$

The first equation have type $\mathcal{N}$. Say that the second have type π and the third have type σ. We then have the following types for the individual terms in the equations

$$\begin{aligned} a &: \mathcal{N} \\ C_0 &: \pi \to \sigma \\ \epsilon &: \mathcal{N} \to ((\pi \to \sigma) \to \pi) \\ g^\star &: \mathcal{N} \to \pi \\ A &: (\mathcal{N} \to \pi) \to \mathcal{N} \\ C_1 &: (\mathcal{N} \to \pi) \to \sigma \end{aligned}$$

The $g^\star$ we want to find can then be seen as an infinite sequence of elements of type π. Let o be a constant of type π. We represent a finite sequence $(g_0, \ldots, g_{k-1})$ as the infinite sequence $(g_0, \ldots, g_{k-1}, o, o, o, \ldots)$. To solve the equations Spector defined the following auxiliary functional by bar recursion

$$\theta(g_0, \ldots, g_{a-1}) = \begin{cases} (g_0, \ldots, g_{a-1}) \text{ if } A(g_0, \ldots, g_{a-1}) < a \\ \theta(g_0, \ldots, g_{a-1}, \epsilon a(\lambda x \cdot C_1 \theta(g_0, \ldots, g_{a-1}, x))) \text{ otherwise} \end{cases}$$

Here θ is defined by the free variables A and C_1. These free variables are supposed to run over computable functionals. So sooner or later — by continuity of the computable functionals — the definition must terminate.

We then define

$$\begin{aligned} g^\star &= \theta() \\ a &= Ag^\star \\ C_0 &= \lambda x \cdot C_1 \theta(g_0, \ldots, g_{a-1}, x) \end{aligned}$$

and get a to be the number where the recursive definition of θ switches cases. We can now check that we have solutions of the equations

$$g^\star = \theta() = \theta(g_0) = \cdots = \theta(g_0, \ldots, g_a) = (g_0, \ldots, g_a)$$
$$g^\star(Ag^\star) = g^\star a = g_a = \epsilon a(\lambda x \cdot C_1 \theta(e_0, \ldots, e_{a-1}, x) = \epsilon a C_0$$
$$C_0(\epsilon a C_0) = C_0 g_a = C_1 \theta(g_0, \ldots, g_{a-1}) = C_1 g^\star$$

9.7 ϵ-calculus

Another way to eliminate quantifiers in arithmetic is to use ϵ-calculus. We have a language of arithmetic (which includes 0 and $<$) extended with ϵ-terms. As new ϵ-axioms we have

$$Ar \to A\epsilon x.A \wedge \epsilon x.A \leq r$$

for any formula A. The quantifier free axioms are such that any closed ϵ-free term is equal to a numeral. We have the ϵ-theorems.

Theorem 29 (First ϵ-theorem for arithmetic) *Any formula A in arithmetic is equivalent to a formula A^ϵ with no quantifiers and using ϵ-terms. A is provable in Peano-arithmetic if and only if A^ϵ is provable in ϵ-arithmetic.*

We prove that the induction rule is derivable from the ϵ-axioms. So assume that we have proved

$$\vdash F0 \wedge \forall x.(Fx \to Fsx)$$

Consider the term $t = \epsilon x.\neg Fx$. Then by the elementary axioms

$$\vdash t = 0 \vee \exists y.t = sy$$

We have

$$\vdash \epsilon x.\neg Fx = 0 \to F\epsilon x.\neg Fx$$

$$\vdash \epsilon x.\neg Fx = sy \wedge \neg Fy \to \epsilon x.\neg Fx \leq sy$$

$$\vdash \epsilon x.\neg Fx = sy \to Fy$$

$$\vdash \epsilon x.\neg Fx = sy \to F\epsilon x.\neg Fx$$

And then $\vdash F\epsilon x.\neg Fx$ — which corresponds to $\vdash \forall x.Fx$.

Theorem 30 (Second ϵ-theorem in arithmetic) *Let F be a formula without quantifiers, free variables or ϵ-terms. If F is provable with the ϵ-axioms, then it is provable without them.*

So assume we have a derivation of F not involving induction, but possibly involving ϵ-axioms. Let us list the ϵ-axioms involved

$$\begin{array}{lcl}
Ar_1\bar{y} & \to & A(\epsilon x.A)(\bar{y})\bar{y} \wedge (\epsilon x.A)(\bar{y}) \leq r_1 \\
 & \dots & \\
Ar_m\bar{y} & \to & A(\epsilon x.A)(\bar{y})\bar{y} \wedge (\epsilon x.A)(\bar{y}) \leq r_m \\
Bs_1\bar{y} & \to & B(\epsilon x.B)(\bar{y})\bar{y} \wedge (\epsilon x.B)(\bar{y}) \leq s_1 \\
 & \dots & \\
Bs_n\bar{y} & \to & B(\epsilon x.B)(\bar{y})\bar{y} \wedge (\epsilon x.B)(\bar{y}) \leq s_n \\
 & \dots &
\end{array}$$

In the derivation we get rid of all non-critical ϵ-terms (i e those not introduced by ϵ-axioms) and all free variables by substituting the constant 0 for them.

We assume that the ϵ-axioms are listed such that $\epsilon x.A$ is of highest rank, and then $\epsilon x.B$ etc. We observe then that in formula Ax there only occurs ϵ-terms introduced further down in the list. The same with Bx etc. There are of course no restriction on the ϵ-terms occuring in the terms $r_1, \ldots, r_m, s_1, \ldots, s_n, \ldots$.

We now want to find substitutions of numerals for the ϵ-terms, $(\epsilon x.A)(\bar{y}) \mapsto \alpha(\bar{y})$, $(\epsilon x.B)(\bar{y}) \mapsto \beta(\bar{y})$, ... making all ϵ-axioms derivable. The substitution α can be regarded as a functional of the substitutions further down on the list. We put $\alpha(\bar{y}) = \alpha(\beta, \gamma, \ldots)(\bar{y})$ and $\beta(\bar{y}) = \beta(\gamma, \ldots)(\bar{y})$ etc.

Each $\alpha(\beta, \gamma, \ldots)(\bar{y})$ has two possible values — either 0 or the least x such that $Ax\bar{y}$. The one that is chosen depends on $Ax\bar{y}$ and on the finite list of ϵ-axioms used.

We first solve $\alpha(\beta, \gamma, \ldots)$ by successive approximations. To be pedantic we write $r\bar{y}\alpha\beta\gamma$ and $Ax\bar{y}\beta\gamma$ to indicate variables in the term r and the formula A. We then define

$$\alpha^0(\beta, \gamma)(\bar{y}) = 0$$
$$\alpha^{i+1}(\beta, \gamma)(\bar{y}) = \max(\alpha^i(\beta, \gamma)(\bar{y}), \mu x \leq \{r_1\bar{y}\alpha^i\beta\gamma, \ldots, r_m\bar{y}\alpha^i\beta\gamma\}Ax\bar{y}\beta\gamma)$$
$$\alpha^\infty = \lim \alpha^i$$

After having solved α, we then solve β by

$$\beta^0(\gamma)(\bar{y}) = 0$$
$$\beta^{i+1}(\gamma)(\bar{y}) = \max(\beta^i(\gamma)(\bar{y}), \mu x \leq \{r_1\bar{y}\alpha^\infty\beta^i\gamma, \ldots, r_m\bar{y}\alpha^\infty\beta^i\gamma\}Bx\bar{y}\gamma)$$
$$\beta^\infty = \lim \beta^i$$

and γ etc until we have given substitutions making all the ϵ-axioms derivable. We observe that $\alpha^\infty(\beta, \gamma)$ is continuous in β and γ and that $\beta^\infty(\gamma)$ is continuous in γ. This means that we can have the above as a simultaneous recursion of all the substitutions of the ϵ-terms.

Chapter 10

Well orderings

10.1 β-logic

At the bottom of our calculations we have some fixed datastructure. It could be unary numbers, formulas, proofs etc. Gödel has learnt us how to represent work within these different datastructures as work within the datastructure unary numbers. It may be worthwhile to give a name to this level. For convenience we just call it

- elementary logic

All concrete calculations which are true can be proved at this level in some appropriate elementary system for the datastructures. We further know that we have usually no problems handling quantifierfree sentences. If such a sentence is true if and only if it is provable within the elementary system But we have problems with true universal — or false existential sentences — in elementary systems. This can be done at the next level

- ω-logic

Here we relax the requirement that all operations should be computable. We require instead that to be an element of the datastructure (i e unary numbers) is absolute. In ω-logic the proofs are wellfounded trees with countable branchings. Such trees can be understood using (countable) ordinals. The next level — and the level for this chapter — is

- β-logic

Here we shall require that the (countable) ordinals are absolute. A little imprecisely we could say: "We have climbed up levels. To get from a level to the one above we require that the means for analyzing the level should be absolute." So much for motivation. Now to the details.

10.2 β-completeness

As the logical system we take a many-sorted first-order sequential calculus with

- one sort of ordinals
- binary relation $<$ in the ordinal sort

A formula is valid in β-logic if it is true in all models where the ordinal sort is interpreted as (a subclass of) ordinals and the relation $<$ as the usual ordering on ordinals.

We want to analyze β-validity. To do this we make the following inessential assumptions about the sequential calculus

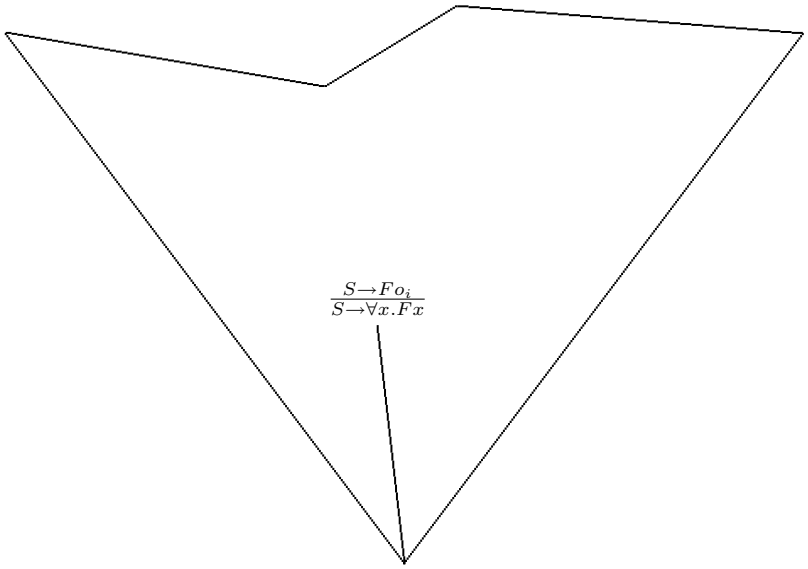
- there are no ordinal constants
- we have a fixed list $o_0, o_1, o_2, \ldots$ of ordinal variables
- no functionsymbols of ordinal sort

We start with the usual construction for the completeness of the first-order sequential calculus. So given a sequent $\Gamma \to \Delta$. We construct in a fair way a potential proof $\mathcal{P}$ over $\Gamma \to \Delta$. Since the construction is fair, we have in the first-order case either

- $\mathcal{P}$ gives a first-order proof of $\Gamma \to \Delta$
- there is a branch in $\mathcal{P}$ giving a counterexample to $\Gamma \to \Delta$

In the β-logic we have the same construction with the simple extra assumptions

- the ordinal variables in $\mathcal{P}$ are from the fixed list
- the ordinal variables are introduced in the same order as the list — specifically if o_i is introduced at one place in $\mathcal{P}$ as an eigenvariable, then only ordinal variables o_j with $j < i$ occurs below it



Here we have indicated a potential proof of $\Gamma \to \Delta$. In some places we introduce ordinal variables as e g o_i. Then on the line below there occurs no ordinal variables o_k with $k \geq i$.

Given a sequent $\Gamma \to \Delta$ we construct the potential first-order proof $\mathcal{P}(\Gamma \to \Delta)$ where we have the extra assumptions satisfied. It is clear that if we have a branch giving a counterexample the ordinal sort will be interpreted by the list of ordinal variables and $<$ a relation on these. There is of course no guarantee that they can be interpreted as a subclass of ordinals. The remainder of the construction in the β-completeness proof try to force such an interpretation.

Let $\langle \alpha_0, \alpha_1, \ldots, \alpha_{n-1} \rangle$ be a finite sequence of ordinals. We then write

$$\mathcal{P}(\Gamma \to \Delta)/\langle \alpha_0, \alpha_1, \ldots, \alpha_{n-1} \rangle$$

for the tree of sequents we get from $\mathcal{P}(\Gamma \to \Delta)$ by substituting α_0 for o_0, α_1 for o_1, $\ldots$, α_{n-1} for o_{n-1}. These substitutions may destroy the look of it as a potential proof. We may have an inference

$$R \to \forall x \, . \, SxR \to So_i$$

which is after substitutions transformed into

$$R^\star \to \forall x \, . \, S^\star x R^\star \to S^\star \alpha_i$$

and it is no longer an inference. Let us recapitulate briefly

- The tree of sequents $\mathcal{P}(\Gamma \to \Delta)$ can be seen as containing all possible first-order counterexamples to $\Gamma \to \Delta$ where the ordinal sort is interpreted as the list $o_0, o_1, \ldots$

- The substitution $\mathcal{P}(\Gamma \to \Delta)/\langle \alpha_0, \alpha_1, \ldots, \alpha_{n-1}\rangle$ gives the counterexamples where $o_0, o_1, \ldots, o_{n-1}$ is interpreted as $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$

Definition 3 *A sequence $\langle \alpha_0, \alpha_1, \ldots, \alpha_{n-1}\rangle$ is not secured (relative to $\Gamma \to \Delta$) if $\mathcal{P}(\Gamma \to \Delta)/\langle \alpha_0, \alpha_1, \ldots, \alpha_{n-1}\rangle$ contains a branch without an axiom.*

Definition 4 *$\mathcal{T}(\Gamma \to \Delta, \alpha)$ is the set of all finite sequences of ordinals $< \alpha$ which are not secured.*

This big set contains all possible counterexamples of $\Gamma \to \Delta$ where the ordinal sort is interpreted as contained in α. We need the following notations

- $\sigma \in \alpha^\star$ — σ is a finite sequence of ordinals $< \alpha$
- $\sigma \sim \tau$ — σ and τ are order isomorphic

The following lemmas are straightforward

Lemma 11 *$\mathcal{T}(\Gamma \to \Delta, \alpha)$ is a tree. Given a sequence σ in it, then all initial segments of σ are also contained in it.*

Lemma 12 *$\mathcal{T}(\Gamma \to \Delta, \alpha)$ is homogeneous. That means that given $\sigma \in \mathcal{T}(\Gamma \to \Delta, \alpha)$, and $\tau \in \alpha^\star$ and $\sigma \sim \tau$, then also $\tau \in \mathcal{T}(\Gamma \to \Delta, \alpha)$.*

Definition 5 *Given a set $\mathcal{T}$ of sequences of ordinals. Then*

$$\mathcal{T}[\alpha] = \{\tau \in \alpha^\star \mid \tau \sim \sigma \text{ for some } \sigma \in \mathcal{T}\}$$

We call it the α-extension of $\mathcal{T}$.

Lemma 13 $\mathcal{T}(\Gamma \to \Delta, \alpha) = \mathcal{T}(\Gamma \to \Delta, \omega)[\alpha]$

This shows that $\mathcal{T}(\Gamma \to \Delta, \alpha)$ is uniquely determined by $\mathcal{T}(\Gamma \to \Delta, \omega)$. We can use $\mathcal{T}(\Gamma \to \Delta, \omega)$ as a universal set coding all possible counterexamples of $\Gamma \to \Delta$ in all possible ordinals α. This is the mathematical core of the β-completeness theorem. Below we spell this out with more details.

Definition 6 *We write $\vdash_\alpha$ for derivability in the sequential calculus with α-rule like*

$$[\text{ for all } \beta < \alpha]R \to \forall x\,.\,Sx \cdots \quad R \to S\beta \quad \cdots$$

and as axioms all true sequents of literals.

Lemma 14 *If $\mathcal{T}(\Gamma \to \Delta, \alpha)$ is wellfounded, then $\vdash_\alpha \Gamma \to \Delta$.*

Proof:

Assume it is wellfounded.

We call a finite sequence of ordinals (in $\alpha^\star$) securable if it is either secured or all its immediate extensions are securable. That the tree is wellfounded gives that the empty sequence is securable.

The substitutions destroy some of the inferences where we substitute in ordinals for eigenvariables. From our construction of the tree of sequents these inferences are low down in the tree, the use of eigenvariable o_i will be below the use of o_j as eigenvariable for $j > i$. Given a finite sequence σ we say that a sequent in $\mathcal{T}(\Gamma \to \Delta, \alpha)/\sigma$ is not critical if it is above the conclusions of all eigenvariables substituted in for σ.

By induction over the securable sequences σ we prove that all not critical sequents in $\mathcal{T}(\Gamma \to \Delta, \alpha)/\sigma$ are derivable $\vdash_\alpha$.

This is obviously true for a secured sequence. Then we have axioms in all branches and using ordinary first order logic we can derive all not critical sequents.

Now consider the securable sequence $\langle \alpha_0, \alpha_1, \ldots, \alpha_{n-1} \rangle$. But then for all $\beta < \alpha$ we have $\langle \alpha_0, \alpha_1, \ldots, \alpha_{n-1}, \beta \rangle$ securable. Consider the inference in $\mathcal{S}(\Gamma \to \Delta)$

$$R \to \forall x \,.\, SxR \to So_n$$

By induction hypothesis we get for all $\beta < \alpha$

$$\vdash_\alpha R^\star \to S^\star \beta$$

and hence by the α-rule

$$\vdash_\alpha R^\star \to \forall x \,.\, S^\star x$$

But then we are able to derive in $\vdash_\alpha$ all sequents which are not critical with respect to $\langle \alpha_0, \alpha_1, \ldots, \alpha_{n-1} \rangle$. And the induction step is proved.

End of proof.

Lemma 15 *If $\mathcal{T}(\Gamma \to \Delta, \alpha)$ is not wellfounded, then there is a counterexample to $\Gamma \to \Delta$ where the ordinal are interpreted as a subset of α.*

Proof:

Assume it is not wellfounded. Then we can find an infinite sequence of ordinals $< \alpha$: $\alpha_0, \alpha_1, \ldots$ such that each finite initial sequence is not secured. There is then a branch in $\mathcal{S}(\Gamma \to \Delta)$ such that it contains no axioms even after having substituted in the α_i's. This branch gives in the ordinary way the counterexample with the ordinals interpreted as the α_i's. There is of course no guarantee that we get all the ordinals $< \alpha$.

End of proof.

In first order logic and in ω-logic we get a symmetry here. If it is not derivable in say ω-logic, we get a counterexample where the universe is ω. In β-logic we get an assymmetry. We can only assume that a counterexample is included in the α. This assymmetry is hidden in our formulation of β-completeness by quantifying over all ordinals. This is how this is done

Definition 7 *A tree of sequences $\mathcal{T}$ is strongly wellfounded if all its extensions $\mathcal{T}[\alpha]$ are wellfounded.*

Theorem 31 (β-completeness) *Given a sequent $\Gamma \to \Delta$: $\Gamma \to \Delta$ is β-valid if and only if $\mathcal{T}(\Gamma \to \Delta, \omega)$ is strongly wellfounded.*

10.3 Discussion

The notion of β-validity was introduced by Mostowski. He considered subsystems of analysis — i e second order arithmetic — with a specified linear ordering. He then asked for validity in all models where the linear ordering is interpreted as a well ordering. The use of β is to remind one of the French "bon ordre". This problem can be reduced to the β-logic of the previous section, and Mostowskis problem were answered by Jean-Yves Girard with his β-completeness.

Mostowski and his students observed that β-validity is a complete Π^1_2-notion, and used this as an argument that it would be hard to characterize a β-logic. In Girards solution the β-validity of a sequent $\Gamma \to \Delta$ is separated into two parts

Algebraic We construct in an elementary way the tree $\mathcal{T}(\Gamma \to \Delta, \omega)$

Wellfoundedness We ask whether the tree is strongly wellfounded

In ordinary predicate logic we have a similar separation

Algebraic We construct in an elementary way a tree

Wellfoundedness We ask whether the tree is finite

and in ω-logic

Algebraic We construct in an elementary way a tree

Wellfoundedness We ask whether the tree is wellfounded

We have separated into a constructive part containing the specific information in the problem and a not decidable question. The question is in general undecidable. This does not mean that the separation is of no value. Even if "to be finite" is undecidable, then we may use the separation in concrete cases. For the "finite" case we know of a number of operations that preserve "finiteness", and there are a number of results using such operations. We may view logic as the science of thinking from assumptions. Then in tracing assumptions through an argument it may be very useful to have operations preserving finiteness. Similarly for operations preserving well-foundedness and strongly well-foundedness.

Theorem 32 *To be strongly wellfounded is a complete* Π^1_2 *notion.*

It is obvious a Π^1_2 notion. That it is complete follows from the β-completeness. We observe that

Lemma 16 $\mathcal{T}$ *is strongly wellfounded if and only if* $\mathcal{T}[\Omega]$ *is wellfounded, where* Ω *is the first uncountable ordinal.*

Proof:
Assume $\mathcal{T}$ is not strongly wellfounded. There is then an α with $\mathcal{T}[\alpha]$ not wellfounded. We can then find a countable sequence of ordinals : $\alpha_0, \alpha_1, \ldots$ giving the infinite branch in $\mathcal{T}[\alpha]$. But then we have an order isomorphic sequence of countable ordinals : $\beta_0, \beta_1, \ldots$ giving an infinite branch in $\mathcal{T}[\Omega]$ — making it not wellfounded. The other way is trivial.
End of proof.

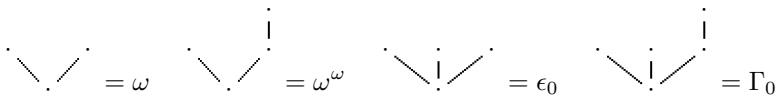
Given an ordinal $\alpha < \Omega$. So α is countable. There is then a sequence of ordinals $\alpha_0, \alpha_1, \ldots$ enumerating all ordinals $< \alpha$. We then construct the tree $\mathcal{T}$ of all finite sequences of natural numbers $n_0, n_1, \ldots, n_{k-1}$ order isomorphic to $\alpha_0, \alpha_1, \ldots, \alpha_{k-1}$. We obviously have that $\mathcal{T}$ is a homogeneous tree. Furthermore as is easily shown: $\mathcal{T}[\beta]$ is wellfounded if and only if $\beta < \alpha$. This shows that to get a tree strongly wellfounded it is necessary and sufficient to consider the wellfoundedness of all countable extensions.

Chapter 11

Finite trees as ordinals

11.1 Ordering trees

We work with ordinary finite trees with immediate subtrees ordered sequentially from left to right and with no labels at the nodes. Here we just call them trees. The smallest tree is $\cdot$ and we draw the trees with the root at the bottom. Here are four examples where we have indicated the corresponding ordinals in the ordering defined below



We write $\langle \mathbf{A} \rangle$ for the finite sequence of immediate subtrees of the tree $\mathbf{A}$, and $\langle \cdot \rangle$ is the empty sequence. Equality between trees is the usual equality. Given that we already know the ordering of some trees we let

$\mathbf{A} \leq \langle \mathbf{B} \rangle$: There is an immediate subtree $\mathbf{B}_i$ of $\mathbf{B}$ such that either $\mathbf{A} < \mathbf{B}_i$ or $\mathbf{A} = \mathbf{B}_i$

$\langle \mathbf{A} \rangle < \mathbf{B}$: For all immediate subtrees $\mathbf{A}_j$ of $\mathbf{A}$ we have $\mathbf{A}_j < \mathbf{B}$

$\langle \mathbf{A} \rangle < \langle \mathbf{B} \rangle$: The inverse lexicographical ordering of the immediate subtrees — we first check which sequence have smallest length, and if they have equal length we look at the rightmost immediate subtree where they differ

We are now ready to define the ordering of trees by recursion over the immediate subtrees.

$$\boxed{\mathbf{A} < \mathbf{B} \;\Leftrightarrow\; \mathbf{A} \leq \langle \mathbf{B} \rangle \;\vee\; (\langle \mathbf{A} \rangle < \mathbf{B} \wedge \langle \mathbf{A} \rangle < \langle \mathbf{B} \rangle)}$$

We must prove that this defines an ordering. We prove by induction over the height of the trees

- The ordering is always defined: $\mathbf{A} < \mathbf{B} \vee \mathbf{A} = \mathbf{B} \vee \mathbf{B} < \mathbf{A}$
- The ordering is transitive: $\mathbf{A} < \mathbf{B} \wedge \mathbf{B} < \mathbf{C} \rightarrow \mathbf{A} < \mathbf{C}$
- The ordering is not reflexive: $\neg \mathbf{A} < \mathbf{A}$
- The ordering is total

The ordering is always defined. It is a straightforward algorithm which to given finite trees $\mathbf{A}$ and $\mathbf{B}$ gives out as answer one of $\mathbf{A} < \mathbf{B}$, $\mathbf{A} = \mathbf{B}$, $\mathbf{B} < \mathbf{A}$.

Now to the transitivity. Given $\mathbf{A} < \mathbf{B} < \mathbf{C}$. We prove $\mathbf{A} < \mathbf{C}$ by induction over the sum of the heights of the trees. We have the following cases:

- $\mathbf{B} \leq \langle \mathbf{C} \rangle$: Then $\mathbf{A} < \mathbf{B} \leq \langle \mathbf{C} \rangle$, and by induction $\mathbf{A} \leq \langle \mathbf{C} \rangle$ and $\mathbf{A} <_j \mathbf{C}$
- $\langle \mathbf{B} \rangle < \langle \mathbf{C} \rangle$ and $\langle \mathbf{B} \rangle < \mathbf{C}$: Then
  - $\mathbf{A} \leq \langle \mathbf{B} \rangle$: Then $\mathbf{A} \leq \langle \mathbf{B} \rangle < \mathbf{C}$ and by induction $\mathbf{A} < \mathbf{C}$
  - $\langle \mathbf{A} \rangle < \langle \mathbf{B} \rangle$ and $\langle \mathbf{A} \rangle < \mathbf{B}$: Then we have $\mathbf{A} < \mathbf{C}$ from
    - $\langle \mathbf{A} \rangle < \langle \mathbf{B} \rangle < \langle \mathbf{C} \rangle$ which gives by induction $\langle \mathbf{A} \rangle < \langle \mathbf{C} \rangle$
    - $\langle \mathbf{A} \rangle < \mathbf{B} < \mathbf{C}$ which gives by induction $\langle \mathbf{A} \rangle < \mathbf{C}$

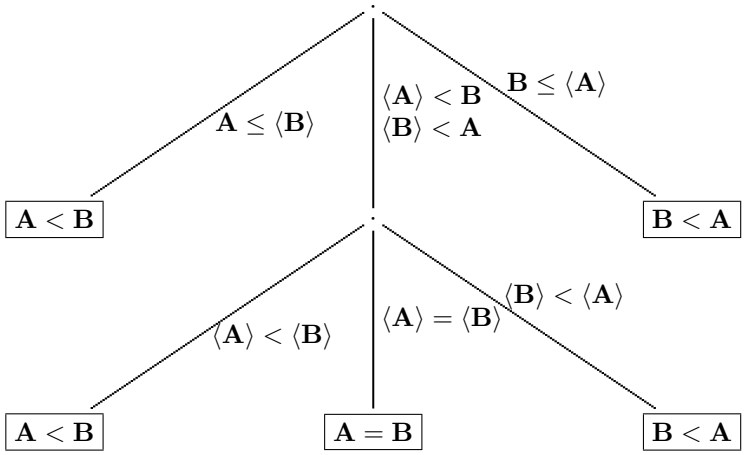
We prove by induction over the height of the trees that we cannot have $\mathbf{A} < \mathbf{A}$. This is immediately seen for the trees consisting of the root alone. Now assume $\mathbf{A} < \mathbf{A}$ with $\mathbf{A}$ of minimal height. Assume $\mathbf{A} \leq \langle \mathbf{A} \rangle$. So there is an immediate subtree $\mathbf{A}_0$ of $\mathbf{A}$ with $\mathbf{A} \leq \mathbf{A}_0$. Now we have $\mathbf{A}_0 < \mathbf{A}$. So by transitivity we have for the smaller tree $\mathbf{A}_0$ we have $\mathbf{A}_0 < \mathbf{A}_0$ which is impossible. So we must have $\langle \mathbf{A} \rangle < \langle \mathbf{A} \rangle$. Again we get an immediate subtree $\mathbf{A}_0$ of $\mathbf{A}$ with $\mathbf{A}_0 < \mathbf{A}_0$ which is impossible.

But then we cannot either have both $\mathbf{A} < \mathbf{B}$ and $\mathbf{B} < \mathbf{A}$. For by transitivity we would get $\mathbf{A} < \mathbf{A}$.

On the other hand for two trees $\mathbf{A}$ and $\mathbf{B}$ there are only the three possibilities $\mathbf{A} < \mathbf{B}$, $\mathbf{A} = \mathbf{B}$, $\mathbf{A} > \mathbf{B}$ and these are exclusive.

Theorem 33 *The ordering between finite trees is a total ordering where the equality is the usual equality between trees.*

We have the following decision tree for the ordering



To get that this total relation is a total ordering we prove that it is transitive. As before the argument is by induction over the heights of the trees.

By induction over the build up of trees we prove

Theorem 34 *Let $\boldsymbol{T}(x)$ be a tree where x indicates a place where we can substitute trees. Then*

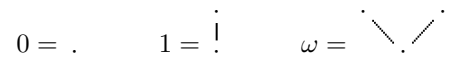
$$\boldsymbol{A} < \boldsymbol{B} \;\Leftrightarrow\; \boldsymbol{T}(\boldsymbol{A}) < \boldsymbol{T}(\boldsymbol{B})$$

Furthermore by induction over the build up

Theorem 35 *If tree $\boldsymbol{S}$ can be embedded in tree $\boldsymbol{T}$, then $\boldsymbol{S} \leq \boldsymbol{T}$.*

11.2 Approximating trees

We note here that we have the following:



But we need to develop more of the theory to come to these and other calculations. First we give approximations of a tree from below. Given a tree **A** with immediate subtrees



The immediate subtrees a_i of **A** are smaller.

Now assume

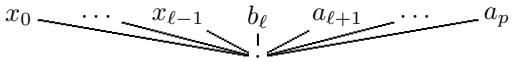
- $b_l < a_l$

- $c_i < \mathbf{A}$ for all $i < l$

Then the following tree is less than $\mathbf{A}$



This can be rephrased that for $b_l < a_l$ the function which to $x_0, \ldots, x_{\ell-1}$ gives



is closed under $\mathbf{A}$. We also get that for $s < p$ that $\mathbf{A}$ is closed under the function which to $x_0, \ldots, x_s$ gives



In the usual theory of ordinal notations we use fundamental sequences as a way to approach ordinals from below. [6] For a given tree $\mathbf{A}$ we call

Fundamental subtrees of A : The immediate subtrees of $\mathbf{A}$.

Fundamental functions of A : The two types of functions above.

Fundamental set of A : The set of trees generated by the fundamental functions starting with the fundamental subtrees.

Elementary fundamental function of A : We first get unary functions by letting all variables except the rightmost be 0. Then use all such unary functions of the first type. If there are no functions of the first type use the one of the second type with largest branching.

Elementary fundamental set of A : The set of trees generated by the elementary fundamental functions starting with the fundamental subtrees.

We denote the fundamental set of $\mathbf{A}$ with $\mathcal{F}(\mathbf{A})$ and we shall write it as

$$[S, \ldots, T | F, \ldots, G]$$

where we have displayed the fundamental subtrees $S, \ldots, T$ and the fundamental functions $F, \ldots, G$. Similarly for the elementary fundamental set $\mathcal{H}(\mathbf{A})$

We have the following:

$$\mathcal{F}(\,.\,) = \emptyset$$

$$\mathcal{F}(\;\vdots\;) = [\cdot\,|\,]$$

$$\mathcal{F}(\;\diagdown\!\diagup\;) = [\cdot\,|\;x\;]$$

$$\mathcal{F}(\;\diagdown\!\diagup\;) = [\cdot,\;\vdots\;|\;x\;,\;y\;,\;\diagdown\!\diagup\;]$$

$$\mathcal{F}(\;\diagdown\,\vdots\,\diagup\;) = [\cdot\,|\;x\;,\;y\;,\;z\;]$$

$$\mathcal{H}(\;\diagdown\,\vdots\,\diagup\;) = [\cdot\,|\;\diagdown\!\diagup\;]$$

Here x, y, z are variables used for describing fundamental functions.
The following theorem shows the importance of the fundamental sets.

Theorem 36 *For any tree* $\boldsymbol{A}$*:*

$$\boldsymbol{B} < \boldsymbol{A} \Leftrightarrow \exists \boldsymbol{C} \in \mathcal{F}(\boldsymbol{A}).\boldsymbol{C} \geq \boldsymbol{B}$$

We prove this by induction over the height of $\mathbf{B}$. It is trivial for height 0. So assume it proved for smaller heights than the height of $\mathbf{B}$. The direction $\Leftarrow$ is obvious. We assume $\mathbf{B} < \mathbf{A}$ and divide up into cases:

$\mathbf{B} \leq \langle \mathbf{A} \rangle$: But then $\mathbf{B}$ is less than or equal to one of the fundamental subtrees of $\mathbf{A}$.

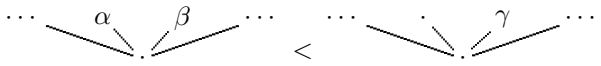
$\langle \mathbf{B} \rangle < \mathbf{A} \wedge \langle \mathbf{B} \rangle < \langle \mathbf{A} \rangle$: By induction — to each immediate subtree $\mathbf{B}_i$ there is an $\mathbf{C}_i \in \mathcal{F}(\mathbf{A})$ with $\mathbf{C}_i \geq \mathbf{B}_i$. Depending on how we prove $\langle \mathbf{B} \rangle < \langle \mathbf{A} \rangle$ we get a fundamental function which we can apply to some of the $\mathbf{C}_i$'s to get a $\mathbf{C} \in \mathcal{F}(\mathbf{A})$ with $\mathbf{C} \geq \mathbf{B}$

And the theorem is proved. We can also use the elementary fundamental set

Theorem 37 *For any tree* $\boldsymbol{A}$*:*

$$\boldsymbol{B} < \boldsymbol{A} \Leftrightarrow \exists \boldsymbol{C} \in \mathcal{H}(\boldsymbol{A}).\boldsymbol{C} \geq \boldsymbol{B}$$

We only need to note that



where $\gamma > \max(\alpha, \beta)$ and that the result of of an application of the second type of fundamental function can be embedded into an application of the first type.

We have also

Theorem 38 *Assume that all trees less than or equal to the fundamental subtrees of* $\boldsymbol{A}$ *is contained in* $\mathcal{F}(\boldsymbol{A})$*, then*

$$\boldsymbol{B} < \boldsymbol{A} \Leftrightarrow \boldsymbol{B} \in \mathcal{F}(\boldsymbol{A})$$

The proof follows the lines above. We have induction over the height of $\mathbf{B}$ and get to the cases

$\mathbf{B} \leq \langle \mathbf{A} \rangle$ **:** Then by assumption $\mathbf{B} \in \mathcal{F}(\mathbf{A})$.

$\langle \mathbf{B} \rangle < \mathbf{A} \wedge \langle \mathbf{B} \rangle < \langle \mathbf{A} \rangle$ **:** By induction — for each immediate subtree $\mathbf{B}_i$ we have $\mathbf{B}_i \in \mathcal{F}(\mathbf{A})$. Depending on how we prove $\langle \mathbf{B} \rangle < \langle \mathbf{A} \rangle$ we get a fundamental function which we can apply to the $\mathbf{B}_i$'s to get $\mathbf{B} \in \mathcal{F}(\mathbf{A})$.

We call a fundamental set which is an initial segment of the ordinals for *full*. The fundamental sets mentioned above are full.

Theorem 39 $\overset{\alpha}{\vdots} = \alpha + 1$

We can prove this by simple induction over trees. We prove

$$\beta < \overset{\alpha}{\vdots} \Leftrightarrow \beta \leq \alpha$$

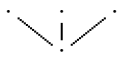
or we can use that the ordering is a wellordering and then induction over α noting that

$$\mathcal{F}(\overset{\alpha}{\vdots}) = [\alpha \,|\, \overset{\alpha-}{\vdots}\,]$$

We are now getting a clearer picture of the ordering. The trees can be divided into layers — we let $\mathcal{T}_i$ be the trees with at most i-branchings. We then get that $\mathcal{T}_1$ is majorised by



and this tree is the least in $\mathcal{T} - \mathcal{T}_1$. The $\mathcal{T}_2$ is majorised by



and this tree is the least in $\mathcal{T} - \mathcal{T}_2$. The $\mathcal{T}_3$ are majorised by



and this tree is the least in $\mathcal{T} - \mathcal{T}_3$. And so on.

11.3 Trees are well ordered

So far the properties could be proved by simple inductions over the heights. Below we shall prove that the trees are well ordered. This requires a stronger method of proof. Let $\mathcal{T}$ be the initial segment of trees which are well ordered. We then prove

Theorem 40 *Assume $\boldsymbol{S}_1$, ..., $\boldsymbol{S}_k$ are well ordered. Then so is $\boldsymbol{S}$ given by*



Let $\mathbf{A} < \mathbf{S}$ be as low as possible and not well ordered. We shall show that $\mathbf{A}$ is well ordered. This is done by the inverse lexicographical ordering of $\mathbf{S}_1$, ..., $\mathbf{S}_k$ where the field of the inverse lexicographical ordering is the set $\mathcal{T}$ of well ordered trees. We now have the following cases

$\mathbf{A} \leq \langle \mathbf{S} \rangle$ **:** Then $\mathbf{A} \leq \mathbf{S}_i$ and is therefore well ordered.

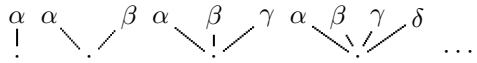
$\langle \mathbf{A} \rangle < \mathbf{S} \wedge \langle \mathbf{A} \rangle < \langle \mathbf{S} \rangle$ **:** Then all immediate subtrees of $\mathbf{A}$ are less than $\mathbf{S}$ and since $\mathbf{A}$ is lowest, then all the immediate subtrees of $\mathbf{A}$ are well ordered. But the immediate subtrees of $\mathbf{A}$ comes before the immediate subtrees of $\mathbf{S}$ in the inverse lexicographical ordering of well ordered sets and with field $\mathcal{T}$. We conclude that $\mathbf{A}$ is well ordered.

The proof is finished. Observe that this proof can be formalized and proved within the theory of inductive definitions. This means — in proof theoretical terms — that the trees give an initial segment of the ordinals, and this segment is below the Howard ordinal. Using that the ordering respects embedding we can lower this estimate considerably to the small Veblen ordinal. [7]

Theorem 41 *The ordering is a well order.*

For the rest of the paper when we talk about ordinals, we mean ordinals from the initial segment given by the (finite) trees. With our orderings on the trees we have obtained an easy translation between ordinals and trees. Note that this is an improvement over the usual theories of ordinal notations. Then we can have multiple representations of an ordinal and must worry how to pick a notation for an ordinal.

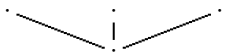
The function below are well defined:



where α, β, γ, δ, ... are ordinals corresponding to (finite) trees. The first function is the successor, but the other functions have not been characterized so far.

11.4 Binary trees

Here we want to describe the trees $\mathcal{T}_2$ with at most binary branching. We know that the supremum of it is



We shall prove that this is in fact the ordinal ϵ_0. We can describe $\mathcal{T}_2$ as the set of ordinals/trees given by

Start: ·

Successor: The function f_+ given by $x \mapsto$ 

α-jump: The function f_α given by $x \mapsto$ 

where α is in $\mathcal{T}_2$

Furthermore we have a 1-1 correspondence between the terms given above and the ordinals in $\mathcal{T}_2$. This can be used to describe the ordinals less than



The ordinals are given by terms built up from f_+ and f_0. For simplicity we just write down the sequence of indices. So we have a finite sequence of $+$ and 0. The empty sequence correspond to 0, and the finite ordinals are $+$,$++$, $+++$, $++++$, ... The tree ordering correspond exactly to the lexicographical ordering of the sequences where we let the 0's be signs separating the $+$'s. The sequence $0++0+00+++0++$ correspond to the sequence $\langle 0, 2, 1, 0, 3, 2\rangle$ and this is again given by the ordinal

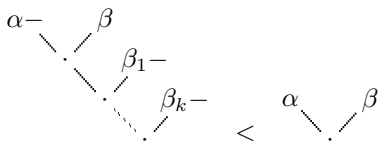
$$\omega^5 + \omega^4 \cdot 2 + \omega^3 \cdot 1 + \omega^2 \cdot 0 + \omega^1 \cdot 3 + 2$$

We get

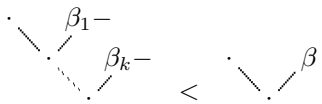
Theorem 42

$\omega^\omega =$ 

It is no surprise that we get connections to the lexicographical ordering in the ordering of binary trees. Let us note that we have



and



where $\alpha- < \alpha$ and $\beta_i- < \beta$. These are the crucial properties for the lexicographical ordering.

We now want to characterize the function

$$g(\alpha) = \quad \alpha$$

The fundamental set is

$$[\cdot, \alpha | x \mapsto x+1, x \mapsto \quad x \quad \alpha- \quad]$$

where $\alpha-$ runs over the ordinals $< \alpha$. Furthermore the fundamental set gives all the ordinals less than $g(\alpha)$. We get that the ordinals less than $g(\alpha)$ are those that are built up from f_+ and the $f_{\alpha-}$ where $\alpha- < \alpha$. We can write them as finite sequences of $+$ and the ordinals $\alpha-$. As before we have a lexicographical ordering where we have as separating sign the largest ordinal in the finite sequence and $+$ is the least element. Hence

$$g(\alpha+1) = g(\alpha)^\omega$$

We get a recursion equation for $g(\alpha)$. We have the same recursion equation in

$$\omega^{(\omega^{\alpha+1})} = (\omega^{\omega^\alpha})^\omega$$

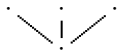
and both function behaves continuously at limits and have the same start. Therefore

Theorem 43 *For all* $\alpha \in \mathcal{T}_2$

$$\alpha \quad = \omega^{(\omega^\alpha)}$$

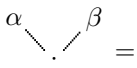
The ordinal $g(\alpha)$ majorises all ordinals built up from f_+ and $f_{\alpha-}$ where $\alpha- < \alpha$. Hence

Theorem 44

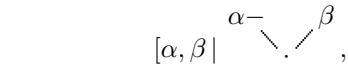
 $= \epsilon_0$

11.5 Finding ordinals of trees

Consider now the function

 $= \psi(\alpha, \beta)$

It has as fundamental set

$[\alpha, \beta \,|$  $]$

where $\alpha-$ runs over the ordinals $< \alpha$, $\beta-$ runs over the ordinals $< \beta$ and x and y are variables indicating functions. The fundamental set gives a recursion equation for the function $\psi(\alpha, \beta)$. We note that it is the least ordinal such that

- $\psi(\alpha, \beta) > \psi(\alpha-, \beta)$
- $\psi(\alpha, \beta) > \alpha$
- $\psi(\alpha, \beta) > \beta$
- $\psi(\alpha, \beta)$ is a limit ordinal
- $\psi(\alpha, \beta)$ is a fixpoint for all $x \mapsto \psi(x, \beta-)$

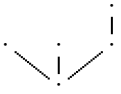
But this is the fix point free Veblen hierarchy starting with $x \mapsto \omega \cdot x$. This hierarchy does not grow so fast. The first critical point is ϵ_0.

Consider now the function

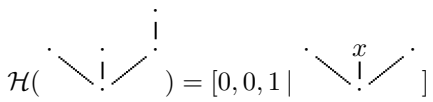
 $= \phi(\alpha, \beta)$

We first observe that $\phi(\alpha, 0)$ gives a fixpoint free enumeration of the ϵ-numbers. This is immediately seen from its fundamental set. But then we get $\phi(\alpha, \beta)$ is the fixpoint free Veblen hierarchy starting with the ϵ-numbers. This is almost the same as the usual Veblen hierarchy [6] where the start is the function ω^α enumerating the multiplicative principal numbers. The first critical number is of course the fixpoint of $x \mapsto \phi(0, x)$ which gives

Theorem 45

 $= \Gamma_0$

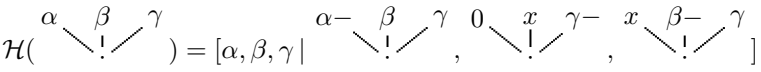
Its elementary fundamental set is



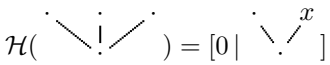
To go further along this line we need the Veblen hierarchy generalized to n-ary functions as defined by Kurt Schütte [8] based on work by Wilhelm Ackermann[1]. Assume we have the ordinary binary Veblen function $\phi(\alpha, \beta)$. We get the ternary Veblen function by

$$\begin{aligned} \phi(\alpha, \beta, 0) &= \phi(\alpha, \beta) \\ \gamma > 0 : \phi(\alpha, 0, \gamma) &= \text{ the } \alpha \text{ common fixpoint of all } \phi(0, x, \gamma-) \\ \beta, \gamma > 0 : \phi(\alpha, \beta, \gamma) &= \text{ the } \alpha \text{ common fixpoint of all } \phi(x, \beta-, \gamma) \end{aligned}$$

And we recognize these cases in the elementary fundamental set (for the case when one of the subtrees is different from 0)



and



So here we have a fixpoint free version of the ternary Veblen hierarchy.

Theorem 46 *The ordinal*  *is the ordinal of the ternary Veblen hierarchy.*

The same argument is lifted up.

Theorem 47 *The ordinal of the trees with n-ary branching where $n > 2$ is the ordinal of the n-ary Veblen hierarchy.*

Theorem 48 *The ordinal of all finite trees is the small Veblen ordinal $\phi_{\Omega^\omega} 0$.*

The small Veblen ordinal $\phi_{\Omega^\omega} 0$ is the ordinal of finitary Veblen functions. It is also the ordinal connected with Kruskals theorem.

Chapter 12

Labeled trees

12.1 Extending finite trees

We now extend our ordinal notations using finite trees. We use finite trees with labels at the nodes. The labels are from a well ordered set Λ. In addition we have an extra label ∞ which is larger than all the labels from Λ. Some examples of what we get:

- $\Lambda = \{0\}$ — this corresponds to our finite trees, the small Veblen ordinal
- $\Lambda = \{0, 1\}$ — to analyze ID_1, the Howard ordinal
- $\Lambda = \{0, 1, 2\}$ — to analyze ID_2
- $\Lambda = \{0, 1, 2, \ldots, \omega\}$ — to analyze ID_ω

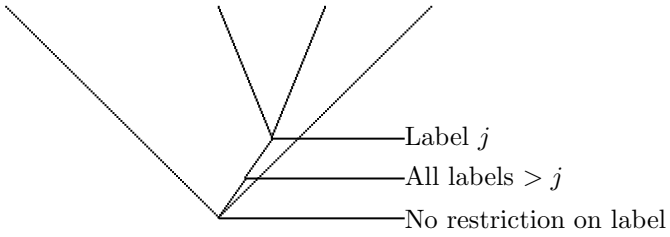
So we have given a well ordered set Λ and the extra label ∞. They are well ordered by $<$. The plan now is to introduce for each $j \in \Lambda \cup \{\infty\}$ the following in a finite labeled tree T

- the sequence $\langle T \rangle_j$ of j-subtrees of T
- an ordering $<_j$ for each j
- the ordering $<_0$ is the one we shall use to compare trees

For each label j: A subtree S of T is a j-subtree if

- the root node in S have label j
- all nodes between the root of T and the root of S have labels $> j$ (— there is no restriction on the label of the root node of T)

Let us use a figure to remind ourselves of the new notion, j-subtree



The sequence of j-subtrees are denoted by

$$\langle T \rangle_j$$

We can now define the orderings of trees

$$S <_j T \Leftrightarrow S \leq_j \langle T \rangle_j \vee (\langle S \rangle_j <_j T \wedge S <_{j+} T)$$
$$S <_\infty T \Leftrightarrow \textit{lexicographical ordering}$$

Here we use abbreviations like for the finite trees

- $\leq_j$ means either ordinary $=$ or $<_j$
- $S \leq_j \langle T \rangle_j$: There exists a j-subtree T_0 of T with $S \leq_j T$
- $\langle S \rangle_j < T$: For all j-subtrees S_0 of S , $S_0 <_j T$
- $j+$ is the smallest label in S and T larger than j if it exists, else it is ∞
- The lexicographical ordering is such that we compare in priority
  - The labels at the root of S and the root of T
  - For the same label i: The lengths of $\langle S \rangle_i$ and $\langle T \rangle_i$
  - The rightmost place where the two sequences differ in the $>_i$-ordering

Observe that if j does not occur in S, then S does not have any j-subtree and $\langle S \rangle_j$ is empty and conditions like $\langle S \rangle_j <_j T$ are trivially true and $T \leq_j \langle S \rangle_j$ are trivially false.

The relations $<_j$ are well defined. We define a relations by using either smaller trees or higher labels (among the finite number of them in the trees we compare).

We now show that the relations are all transitive. So assume we have

$$\mathbf{A} <_j \mathbf{B} <_j \mathbf{C}$$

and want to prove by induction that we get $\mathbf{A} <_j \mathbf{C}$. The induction uses either smaller trees or higher labels. First for $j \in \Lambda$

- $\mathbf{B} \leq_j \langle \mathbf{C} \rangle_j$: Then $\mathbf{A} <_j \mathbf{B} \leq_j \langle \mathbf{C} \rangle_j$, and by induction $\mathbf{A} \leq_j \langle \mathbf{C} \rangle_j$ and $\mathbf{A} <_j \mathbf{C}$
- $\mathbf{B} <_{j+} \mathbf{C}$ and $\langle \mathbf{B} \rangle_j <_j \mathbf{C}$: Then
  - $\mathbf{A} \leq_j \langle \mathbf{B} \rangle_j$: Then $\mathbf{A} \leq_j \langle \mathbf{B} \rangle_j <_j \mathbf{C}$ and by induction $\mathbf{A} <_j \mathbf{C}$
  - $\mathbf{A} <_{j+} \mathbf{B}$ and $\langle \mathbf{A} \rangle_j <_j \mathbf{B}$: Then we have $\mathbf{A} <_j \mathbf{C}$ from
    - $\mathbf{A} <_{j+} \mathbf{B} <_{j+} \mathbf{C}$ which gives by induction $\mathbf{A} <_{j+} \mathbf{C}$
    - $\langle \mathbf{A} \rangle_j <_j \mathbf{B} <_j \mathbf{C}$ which gives by induction $\langle \mathbf{A} \rangle_j <_j \mathbf{C}$

Then for $j = \infty$. Here we have

- The labels at (least two of) the roots are different. Then $\mathbf{A} <_\infty \mathbf{C}$
- The labels are the same. Then we compare immediate subtrees and by induction $\mathbf{A} <_\infty \mathbf{C}$

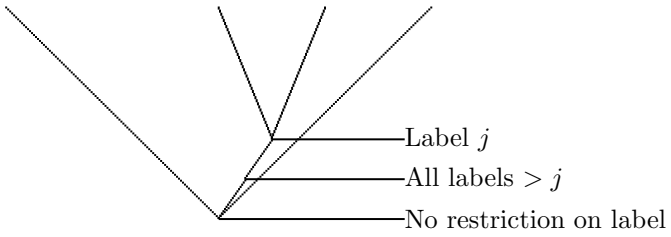
So the orderings are transitive. They are also total. We decide the ordering by a similar decision tree as for finite trees. To compare $\mathbf{A}$ and $\mathbf{B}$ we go through a number of tests and sequentially compare

- $A \leq_j \langle B \rangle_j$ — giving $A <_j B$
- $B \leq_j \langle A \rangle_j$ — giving $B <_j A$
- $A \leq_{j+} \langle B \rangle_{j+}$ — giving $A <_j B$
- $B \leq_{j+} \langle A \rangle_{j+}$ — giving $B <_j A$
- $A \leq_{j++} \langle B \rangle_{j++}$ — giving $A <_j B$
- $B \leq_{j++} \langle A \rangle_{j++}$ — giving $B <_j A$
- ...
- ...
- The root of A is less than the root of B — giving $A <_j B$
- The root of B is less than the root of A — giving $A <_j B$
- The roots have the same label i
- $\langle A \rangle <_i \langle B \rangle$ — giving $A <_j B$
- $\langle B \rangle <_i \langle A \rangle$ — giving $B <_j A$
- At last the roots must be the same and all immediate subtrees are the same — giving $A = B$

The test is read as follows: We start with line 1. If it succeds we are finished. Else we go to line 2. If it succeeds then we are finished. In this way we go through the tests line for line until we find a test which succeeds. If none succeds the two labeled trees must be equal.

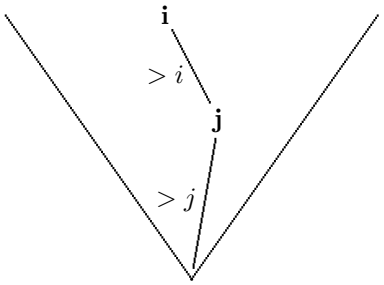
12.2 Gaps

In the finite labeled trees we do not just take immediate subtrees, we take j-subtrees for any label j. We have the following situation:



So to get to the j-subtrees we first pass over the downmost node and then pass over all nodes $> j$ until we come to a node with label j. This gives a layered structure for the j-subtrees. Furthest above the root are the 0-subtrees. Then come the 1-subtrees, the 2-subtrees and so on. To get to the j subtrees we would never pass through the i subtrees for $i < j$. And if we look at the j-subtrees and then the j-subtrees of those, the j-subtrees of those and so on we would never go through the i-subtrees where $i < j$.

Let us look a little closer at the i- and j-subtrees where $i < j$. We have the following situation



Observe that if S is an i-subtree of T and T is a j-subtree of U, then S is an i-subtree of U. This means that when we go to j-subtrees we do not get extra i-subtrees.

12.3 Well ordering of finite labeled trees

We start with assuming that we have an infinite $<_0$-descending sequence of trees.

$$T_0 >_0 T_1 >_0 T_2 >_0 \dots$$

Then we construct for each label i an infinite $<_i$-descending sequence satisfying some extra conditions. If we now look at the descending sequence for

$i = \infty$ we get a contradiction. Instead of going directly through the whole construction, we start with the case where the only label in the trees is 0. This is the same as looking at finite trees. We start with constructing from the T_n's a minimal bad sequence

$$T_0^0 >_0 T_1^0 >_0 T_2^0 >_0 \dots$$

A bad tree is one which is not well founded — there is an infinite $<_0$-descending sequence starting with it. A tree is minimal bad if it is bad but all of its immediate subtrees are not bad. Let us see how we construct a minimal bad tree from T_0. T_0 is bad but may not be minimal. If not then one of its immediate subtrees are bad. If this is not minimal, then we go to one of its immediate subtrees and so on. The topmost subtrees are not minimal bad. But either T_0 itself or one of its subtrees must be. Having now constructed a bad sequence starting with a minimal tree

$$T_0^0 >_0 T_1^0 >_0 \dots >_0 T_n^0$$

we must show how to extend it. By construction T_n^0 is minimal bad and can be continued

$$T_0^0 >_0 T_1^0 >_0 \dots >_0 T_n^0 >_0 S_{n+1} >_0 S_{n+2} >_0 \dots$$

but where S_{n+1} is bad but may not be minimal. But then by the same construction above we have a subtree U_{n+1} which is minimal bad. Now observe that $T_n^0 >_0 U_{n+1}$ since U_{n+1} is just a subtree of S_{n+1}.

In this construction

- we must decide whether certain trees are bad or not — i.e. a Π_1^1-notion
- in the first place where the original bad sequence and the constructed minimal bad sequence differ, the one tree is a subtree of the other

Now we go to the definition of the ordering

$$\boxed{S <_j T \Leftrightarrow S \leq_j \langle T \rangle_j \vee (\langle S \rangle_j <_j T \wedge S <_{j+} T)}$$

We have used the first disjunct in the (extension part of the) construction of minimal bad sequence. In a minimal bad sequence we cannot use the first disjunct

$$\langle T_n^0 \rangle \geq_0 T_{n+1}^0$$

For then we would have an immediate subtree of T_n^0 which is bad contradicting minimality. In our case $0+ = \infty$. Therefore we have

$$T_0^0 >_\infty T_1^0 >_\infty T_2^0 >_\infty \dots$$

Then from some stage in the sequence

- all trees have equally many immediate subtrees
- there is a number k such that immediate subtree k is the rightmost place where they all differ and this immediate subtree must be bad

And this contradicts that all the trees in the sequence where minimal.

We now go to the general case. There are some changes. The most important is that we go to a new notion of minimality.

$$S \text{ is } j-\text{minimal} \Leftrightarrow \forall i \leq j \neg \exists R \in \langle S \rangle_i \text{ is } i-\text{bad}$$

So 0-minimality coincides with the usual notion of minimality. We construct 0-minimal bad sequences using Π^1_1-tests. But this is more complex when we come to 1-minimal 1-bad sequences. We shall then go through the following steps

- We start with 0-bad sequence
- We then construct a 0-minimal 0-bad sequence
- This sequence is also a 1-bad sequence
- We then construct a sequence which is 1-minimal 1-bad

To construct the 0-minimal 0-bad sequence we quantify over all 0-bad sequences. Here we use Π^1_1. But to construct the 1-minimal 1-bad sequences we quantify over 1-bad sequences of 0-minimal trees. This goes beyond Π^1_1.

The proof starts with assuming we have a $<_0$-bad sequence. We construct an array of i-minimal i-bad sequences

$$\begin{array}{ccccccccccc}
T^0_0 & >_0 & T^0_1 & >_0 & T^0_2 & >_0 & T^0_3 & >_0 & T^0_4 & >_0 & \dots \\
T^1_0 & >_1 & T^1_1 & >_1 & T^1_2 & >_1 & T^1_3 & >_1 & T^1_4 & >_1 & \dots \\
T^2_0 & >_2 & T^2_1 & >_2 & T^2_2 & >_2 & T^2_3 & >_2 & T^2_4 & >_2 & \dots \\
T^3_0 & >_3 & T^3_1 & >_3 & T^3_2 & >_3 & T^3_3 & >_3 & T^3_4 & >_3 & \dots \\
\dots & & \dots & & \dots & & \dots & & \dots & & \dots
\end{array}$$

Line i is a i-bad sequence of i-minimal trees. Now we have to observe two things:

- How the construction of minimal badness works when we have labels
- How to proceed at limit ordinals

For the first assume we have an i-bad sequence of $i-1$-minimal trees.

$$T_0 >_i T_1 >_i T_2 >_i \dots$$

To get to an i-minimal i-bad sequence we go to i-subtrees. Observe now that this does not destroy the $i-1$-minimality. So we get an i-bad sequence of $i-1$-minimal trees where the first element is also i-minimal.

$$S_0 >_i S_1 >_i S_2 >_i \dots$$

Repeating this we get the whole sequence i-bad and i-minimal.

Now to the argument for how to proceed at limit ordinals. We use the following fact about the construction

- in the first place where the bad sequence at a line and the bad sequence at the line below differ, the one tree is a subtree of the other

We then show that if we look at the columns the trees there change only finite many times. For assume not. Call a column where we have infinitely many changes for critical. Then look at the leftmost critical column. From a certain line off no trees in the columns to the left of it are changing. This means that then in all changes in the column the change is only going from a tree to a subtree. But this cannot be done infinitely many times. So there are no critical columns. In each column there are only finite number of changes and we simply define the limit tree in the column to the tree which are not changing. As in the case for finite trees we get to line ∞

$$T_0^\infty \quad >_\infty \quad T_1^\infty \quad >_\infty \quad T_2^\infty \quad >_\infty \quad T_3^\infty \quad >_\infty \quad T_4^\infty \quad >_\infty \quad \dots$$

by just taking the next line after taking one line for each label. All the trees here are i-minimal for each i. We get a contradiction as for the simple case above. If we have an infinite $<_\infty$-descending sequence, then we get an i and an infinite $<_i$-descending sequence of immediate i-subtrees — contradicting i-minimality.

Chapter 13

Universes

13.1 Data structures

The incompleteness phenomena are connected with our attepts to make formal systems for data structures. In a data structure we have

- one or more start objects
- one or more constructors

And then we try to make a formal system for the data structure using

- one or more axioms
- one or more rules

We try to catch the properties of the data structure with something which looks like a data structure itself. The simplest data structure is the natural numbers

- one start: the zero
- one unary constructor: the successor

This is not complicated enough to get the incompleteness phenomena there. We are not able to code the formal system within it. But if we add the two binary functions — addition and multiplication — then we get the coding. This is also done for the slightly more complicated data structure of binary trees

- one start: the empty tree
- one binary constructor: the cons

We get incompleteness phenomena for sufficiently strong formal systems. We have the following levels

- MIN: all true syntactical sentences (Δ_0-sentences) are axioms
- the Robinson system: finite number of axioms
- Σ_1-IND: induction over Σ_1-formulas
- the Skolem system: names for all primitive recursive terms and quantifier free induction
- IND: induction over arbitrary formulas

We have the same levels for any data structure. Now to the analysis of transfinite induction. It is clear that — provided we can code syntax — we get transfinite induction up to the same ordinal. If we have full induction in the data structure, we can prove transfinite induction up to any ordinal $< \epsilon_0$. For the converse we only have to check that the formal system can be imbedded in a infinitary system where we can perform cut elimination. For any data structure we have a countable list of elements, and we can replace a finitary derivation involving induction with an infinitary derivation of height $< \omega^2$. And the whole argument with ϵ_0 as the 'limit' for the IND systems over the data structure.

It is crucial that we can imbed the finitary derivation into an infinitary derivation where we have at most countable branching. A short argument shows some troubles with uncountable branching. We observe first that cut elimination works for arbitrary infinitary derivations. Then crucially we do not have completeness of a sequential calculus with uncountable connectives. Consider the sentence in propositional calculus

$$\bigwedge m \in \mathbb{N} \bigvee n \in \mathbb{N}.A(m,n) \to \bigvee f \in \mathbb{N}^{\mathbb{N}} \bigwedge m \in \mathbb{N}.A(m,fm)$$

The sentence expresses an axiom of choice and is valid. Assume that the sentence is derivable in sequential calculus. Then it has a cut free derivation. In the cut free derivation there are only a countable number of instances of $f \in \mathbb{N}^{\mathbb{N}}$ used — say some countable $\mathcal{F} \subset \mathbb{N}^{\mathbb{N}}$ and with

$$\bigwedge m \in \mathbb{N} \bigvee n \in \mathbb{N}.A(m,n) \to \bigvee f \in \mathcal{F} \bigwedge m \in \mathbb{N}.A(m,fm)$$

derivable. But this sentence is no longer valid. So we do not have completeness for uncountable connectives.

13.2 Second order logic

13.3 Monotone operators

In the data structures each element is given by a finite tree with finite branching. We can extend this to wellfounded trees with countable branchings. Let us do this in a general setting:

- For each ordinal α we have variables x^α

- We consider formulas $F(x^\alpha)$ — called α-formulas — where
  - there are no β-variables for $\beta > \alpha$
  - the α-variables occur only positively

We then take comprehension of the α-formulas to get the elements of the universe.

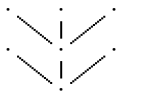
13.4 Fix point universes

We only demand that the universes satisfy the comprehension of the α-formulas. We then get a hierarchy of systems

$$\widehat{ID_\alpha}$$

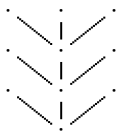
These systems are connected with the following ordinals

$$|\widehat{ID_0}| = \epsilon_0 =$$

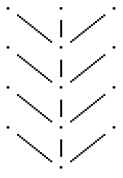


$$|\widehat{ID_1}| =$$

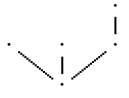
$$|\widehat{ID_2}| =$$



$$|\widehat{ID_3}| =$$



And so on. At stage ω we get to



$= \Gamma_0$

13.5 Inductive universes

Now we demand that not only α-formulas should have a fix point, but that this fix point is the least possible. Then we get the iterated inductive systems

$$ID_\alpha$$

As before we start with ϵ_0. We have $ID_0 = \widehat{ID_0}$. But the ordinals are then those constructed with labeled finite trees

$$|ID_n| = n + 1 \text{ for } n > 0$$

Index

Cantor, 23
Chinese remainder theorem, 13
Completeness, $\mathbf{\Sigma}_1$, 31
Consistency, 33
Curry, 28

Data structure, 9
Data structure, language, 10
Decision procedure, $\mathbf{\Delta}_0$ sentences, 11

Fair coding, 30
Fix point, 25

Gödel, 10, 20
Gödel-Löb, 30
Gentzen, 10

Incompleteness, theorem one, 34
Incompleteness, theorem two, 34
Incompleteness, theorem zero, 18
Induction, 21
Invariant, 13

Löbs rule, 32

Necessity, 30
Non standard model, 20

Pair, 10
Primitive recursive functions, 9
Provability, 13, 29

Russell, 24

Self reference, 26
Skolem, 9, 20
Specification, 13
Syntax, 13

Theories of pairs, 17
Theory, full induction, 17
Theory, minimal, 17
Theory, predicative induction, 17
Theory, Robinson, 17, 19
Theory, Skolem, 17, 21
Turing, 10

Unary numbers, 13
Undecidability, predicate logic, 19

Bibliography

[1] Wilhelm Ackermann. Konstruktiver aufbau eines abschnitts der zweiten cantorsche zahlenklasse. *Mathematische Zeitschrift*, 53:403–413, 1951.

[2] Jon Barwise. *Handbook of mathematical logic*. North-Holland, Amsterdam, 1977.

[3] George Boolos. *The logic of provability*. Cambridge University Press, Cambridge, 1993.

[4] Martin Davis. *The undecidable*. Raven Press, New York, 1965.

[5] Gerhard Gentzen. *The collected papers of Gerhard Gentzen*. North-Holland, Amsterdam, 1969.

[6] Wolfram Pohlers. *Proof Theory : An Introduction*. Number 1407 in Lecture Notes in Mathematics. Springer, Berlin, 1989.

[7] Michael Rathjen and Andreas Weiermann. Proof-theoretic investigations on kruskal's theorem. *Annals of Pure and Applied Logic*, 60:49–88, 1993.

[8] Kurt Schütte. Kennzeichnung von ordinalzahlen durch rekursiv erklärte funktionen. *Mathematische Annalen*, 127:15–32, 1954.

[9] Thoralf Skolem. *Abstract set theory*. Number 8 in Notre Dame Mathematical Lectures. Notre Dame University, 1962.

[10] Thoralf Skolem. *Selected works in logic*. Universitetsforlaget, 1970.

[11] Craig Smoryński. *Self-reference and modal logic*. Springer Verlag, New York, Berlin, Heidelberg, Tokyo, 1985.

[12] Raymond Smullyan. *Gödel's incompleteness theorems*. Oxford University Press, New York, Oxford, 1992.

[13] Robert Solovay. Provability interpretations of modal logic. *Israel Journal of Mathematics*, 25:287–304, 1976.

[14] Jean van Heijenoort. *From Frege to Gödel*. Harvard University Press, Cambridge Massachusetts, 1967.