# Truth and provability

Herman Ruge Jervell

May 24, 2009

# Contents

# 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

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 Truth

The basic situation is that we have given a universe U and a language L over it. The universe is given by

- the elements
- relations between elements
- functions from some number of elements to an element

We usually require that the relations and functions have some fixed arity and that the functions are total. We assume first that we have names for each element. Then we build the language

- We have symbols for the relations, the functions and some of the elements.
- We get atomic sentences.
- The literals are either the atomic sentences or the negation of them.
- We combine literals with conjunctions and disjunctions.
- We have $\forall$- and $\exists$-quantifiers ranging over the elements. We may also have higher order quantifiers ranging over say predicates.

All this is standard. We can imagine that the truth of a sentence is given by a (large) wellfounded AND-OR tree with sentences at each node and ending up with literals at the leaves. We call this tree for the semantic tree of the sentence.

When we come to provability there are two essential changes

- We try to make things finite
- Our trees have only AND-nodes

Some of this can be obtained in a very general setting — changing the semantic trees into a sequential calculus. But we may also have to consider the structure of the universe — either as given by general axiomatic properties or as generated from constructors like a datastructure.

## 0.3 Thinking from assumptions

One naive semantics is given by the "'picture theory of meaning"'. There we assume we have a correspondence between words and pictures. This seems to work for proper nouns. Imagine that we have such a full fledged picture theory and consider a proof by contradiction — say the ordinary proof that there are infinitely many primes. Then we start with a picture of the world with only a finite number of primes. Then as the argument proceeds we get more and more details in that picture. But then a contradiction appears and our picture

vanishes. What we thought was a picture turned out not to be a picture after all. But — what was it then?

We explain the situation by introducing the notion of thinking from assumption. And it is important that those assumption may even be wrong. In logic and other formal sciences we think from assumptions. The assumptions could be that something is finite, something is a syntactical object, ..... Then after a number of steps we may conclude that something else is finite, syntactical object, ...

In general we cannot decide whether something is finite or not. But for the reasoning we only have to assume that we know that something is finite and then can transform that knowledge into that something else is finite. Similarly with a number of other types of assumptions made.

To do this reasoning we had to know how to handle assumptions of finiteness, infiniteness, ...

Our minds are finite, but we can nevertheless imagine how an infinite mind would work. We do this in formal reasoning all the time even when we work with finite objects. In the typical argument we assume that some finite thing is given and can then conclude that also something else is finite. But to operate on finite objects in this way — and be sure that the results are finite — we usually have to use the infinite mind. Both intuitionistic and classical logic use the infinite mind.

We have chosen to use classical logic for most of our theory. To us this is mostly a matter of convenience. In a number of situations we can go between classical and intuitionistic logic by for example a double-negation interpretation. But in these cases the choice between the logics is not fundamental — it is only a matter of what we would like to bring forth in our analyses and is similar to the choice of one particular syntax of the formal system over another. There are cases where we do not have such an easy connection between the classical and the intuitionistic logic. We have for example problems when we come to iterated inductive definitions and stronger systems. There the choice of logic may matter.

## 0.4 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)
- Wilhelm Ackermann (1896 - 1962)
- Emil Post (1897 - 1954)

- Kurt Gödel (1906 - 1978)
- Gerhard Gentzen (1909 - 1945)
- Stephen Kleene (1909 - 1994)
- Kurt Schütte (1909 - 1998)
- Alan Turing (1912 - 1954)
- Gaisi Takeuti (1926 - )
- Jean Yves Girard (1947 - )

Look up some of their papers — they are easily available — and read them.

# 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*. 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.

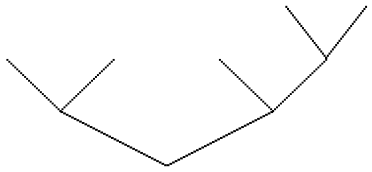
The tension between logic as a language and logic as a calculus has followed logic through history. With Skolems paper we have the means to approach the problem in an interesting way — a way leading through the works of Kurt Gödel, Gerhard Gentzen and Alan Turing in the 1930 up to present work in proof 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**$(x)$ **:** $\neg x = \mathbf{nil}$

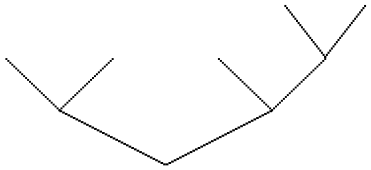
**Head**$(x, y)$ **:** $\exists z \prec x.x = (y \cdot z)$

**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* $\Delta_0$*-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 $\Delta_0$-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.

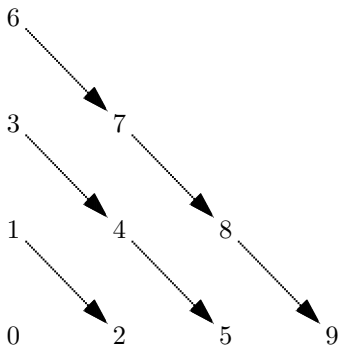
Let us start with unary numbers, addition and multiplication. A standard way to represent a pair of two numbers is

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

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 $\beta$-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 $\beta$-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 $\Delta_0$-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 $\Pi_1$-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 $\Delta_0$-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 inconsistent if there is a sentence which is not provable

We then get to the real incompleteness theorem by Gödel.

## 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.

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 $\Sigma_1$-induction in Skolem theory.

We shall not show this in detail. To do it we need to know more about the actual system — and we would like to eliminate quantifier-cuts which is something which we shall treat later. Let us however note the following detail. Assume that we have primitive recursive realizers, we can then prove $\mathbf{\Sigma_1}$-induction in Skolems theory. For assume that we have proved

- $\exists y.R0yz$
- $\forall x.(\exists y.Rxyz \rightarrow \exists y.Rsxyz)$

We can then find primitive recursive functions $f$ and $g$ and proofs in Skolem theory of

- $R0f(z)z$
- $\forall x.\forall y.(Rxyz \rightarrow Rsxg(y,z)z)$

From $f$ and $g$ we can construct a primitive recursive $h$ and a proof in Skolem theory of

$$Rxh(x,z)z$$

and therefore

$$\forall x.\exists y.Rxyz$$

In fact with a little more work we can derive induction for formulas which are disjunctions of $\mathbf{\Sigma}_1$ and $\mathbf{\Pi}_1$ formulas.

The property about primitive recursive realizers give bounds on what we can prove in Skolems theory. The formula $\forall x \exists y Axy$ expressing that the Ackermann function is total is true but not provable here.

# 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 $\Delta_0$-formulas
- The axiom system is strong enough to derive all true $\Delta_0$-sentences

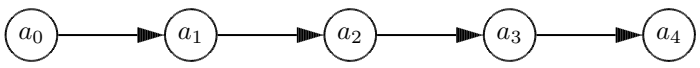
For some systems we can derive decidability. But then either the language is not strong enough or the system is not strong enough. Let us look a little closer at this phenomenon.

## 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

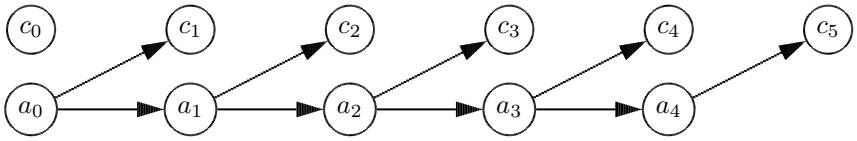
- strings over $\mathcal{A}$
- (binary) trees over $\mathcal{A}$

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

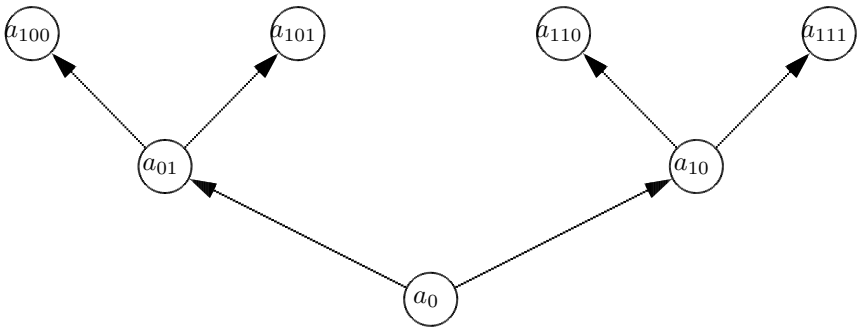



An automaton over $\mathcal{A}$ is a coloring scheme. There is given a finite set of colors $\mathcal{C}$ and an association $\mathcal{A} \times \mathcal{C} \to \mathcal{C}$. If this association is a function, then

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 these questions about acceptance are 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.

There are four logical theories connected with automata. In the theories we have natural numbers at the bottom and above them monadic relations

**WS1S:** Weak second order monadic logic with one successor. Natural numbers with one successor, the second order quantifiers is over finite subsets.

**WS2S:** Weak second order monadic logic with two successors. Natural numbers with two sucessors — i.e. like the binary numbers.

**S1S:** Second order monadic logic with one successor. The second order quantifiers is over arbitrary subsets.

**S2S:** Second order theory with two successors.

There is a close connections between definability in the logics and acceptance by automata

**WS1S:** Automata with finite strings as input

**WS2S:** Automata with finite binary trees as input

**S1S:** Automata with infinite strings as input

**S2S:** Automata with infinite binary trees as input

The trees are binary. But the same theory works for trees with finite, and even countable, branchings. Sentences in these theories can be taken as true or false. It turns out that we can decide this using operations on automata. The crucial thing is to show that we can define the complements of automata. For finite inputs we use the usual transformation of the automata from non-deterministic to deterministic automata. For infinite inputs there is more work to be done. This is treated in the last two sections in this chapter.

## 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.

## 3.7 Infinite input

We now consider the behavior of finite automata where we have infinite input — either infinite strings or infinite trees. The mechanisms for coloring is the same as for finite automata with finite input, but the acceptance criteria are different. We need three different kinds of criteria — Büchi, Muller and Rabin. Here are the acceptance criteria for strings

**Finite string:** There is a subset $F \subseteq \mathcal{F}$ — the accepted colors. A finite string is accepted if there is a coloring which ends up with an accepted color.

**Büchi automaton:** There is a subset $F \subseteq \mathcal{F}$ — the accepted colors. An infinite string is accepted if there is a coloring where one of the accepted colors is used infinitely often.

**Muller automaton:** There is a set $G$ of subsets of $\mathcal{F}$. An infinite string is accepted if there is a coloring where the set of colors used infinitely often is exactly one of the elements of $G$.

**Rabin automaton:** There is a level function which to each color gives a natural number. An infinite string is accepted if there is a coloring where the colors of lowest level and used infinitely often is of even level.

For automata over infinite trees we have acceptance if all the branches of the tree are accepted as infinite strings. We consider here only infinite trees with binary branching, but this is no restriction on the generality — we may have arbitrary finite branching and even infinite denumerable branching.

Let us now consider some examples of what we can get accepted. As alphabet we have $\{\mathbf{a}, \mathbf{b}\}$.

*String automata for infinitely many* $\boldsymbol{a}$*'s.* We have two colors

**Red:** Last seen $\mathbf{a}$

**Green:** Last seen $\mathbf{b}$

and transitions

$$\begin{aligned} \mathbf{red} \times \mathbf{a} &\mapsto \mathbf{red} \\ \mathbf{red} \times \mathbf{b} &\mapsto \mathbf{green} \\ \mathbf{green} \times \mathbf{a} &\mapsto \mathbf{red} \\ \mathbf{green} \times \mathbf{b} &\mapsto \mathbf{green} \end{aligned}$$

We have deterministic automata with the following acceptance criteria

**Büchi:** $\{\mathbf{red}\}$

**Muller:** $\{\{\mathbf{red}\}, \{\mathbf{red}, \mathbf{green}\}\}$

**Rabin:** $\mathbf{red} \mapsto 0$ and $\mathbf{green} \mapsto 1$

*String automaton for infinitely many* $\boldsymbol{a}$*'s.* This is the complement of the strings above. We use the same automaton but with the following acceptance criterion

**Muller:** $\{\{\mathbf{green}\}\}$

There is no deterministic Büchi automaton for these strings. For assume we had one $\mathcal{B}$ with accepting colors $F$. We feed the automaton with only $\mathbf{b}$'s until we meet a color in $F$. Let us say we use $\mathbf{b}^{n_0}$. Then we add a $\mathbf{a}$ and continues with $\mathbf{b}$'s until we meet a color in $F$. Add one $\mathbf{a}$ and continue. We get finite strings

$$\mathbf{b}^{n_0}\mathbf{a}\mathbf{b}^{n_1}\mathbf{a}\cdots\mathbf{a}\mathbf{b}^{n_k}$$

all ending up with a color from $F$. One of the colors must be repeated and we construct an infinite string

$$\cdots(\cdots\mathbf{a}\cdots)^\infty$$

which is accepted. But this string contains infinitely many **a**'s.

On the other hand we can make a non-deterministic Büchi automaton accepting the strings. We have the colors **red** and **green** and **brown** which stands for

**Brown:** Not decided yet

**Red:** Seen **a** after having decided

**Green:** only seen **b**'s after having decided

with the transitions

$$\begin{aligned}
\mathbf{brown}\times\mathbf{a} &\mapsto \mathbf{brown}\\
\mathbf{brown}\times\mathbf{b} &\mapsto \mathbf{brown}\\
\mathbf{brown}\times\mathbf{b} &\mapsto \mathbf{green}\\
\mathbf{red}\times\mathbf{a} &\mapsto \mathbf{red}\\
\mathbf{red}\times\mathbf{b} &\mapsto \mathbf{red}\\
\mathbf{green}\times\mathbf{a} &\mapsto \mathbf{red}\\
\mathbf{green}\times\mathbf{b} &\mapsto \mathbf{green}
\end{aligned}$$

and acceptance

**Büchi:** {**green**}

**Muller:** {{**green**}}

**Rabin:** **brown** $\mapsto 1$, **red** $\mapsto 1$ and **green** $\mapsto 0$

We can also make a deterministic Rabin automaton for the strings. We then use more colors to distinguish between the last two symbols seen

**Red:** Seen last **a** and then **a**

**Yellow:** Seen last **a** and then **b**

**Green:** Seen last **b** and then **a**

**Blue:** Seen last **b** and then **b**

with transitions

$$\begin{aligned}
\textbf{red} \times \textbf{a} &\mapsto \textbf{red} \\
\textbf{red} \times \textbf{b} &\mapsto \textbf{yellow} \\
\textbf{yellow} \times \textbf{a} &\mapsto \textbf{green} \\
\textbf{yellow} \times \textbf{b} &\mapsto \textbf{blue} \\
\textbf{green} \times \textbf{a} &\mapsto \textbf{red} \\
\textbf{green} \times \textbf{b} &\mapsto \textbf{yellow} \\
\textbf{blue} \times \textbf{a} &\mapsto \textbf{green} \\
\textbf{blue} \times \textbf{b} &\mapsto \textbf{blue}
\end{aligned}$$

and acceptance

**Muller:** $\{\{\textbf{blue}\}\}$

**Rabin:** **red** $\mapsto 1$, **yellow** $\mapsto 1$, **green** $\mapsto 1$, **blue** $\mapsto 2$

*Tree with a branch with infinitely many* ***a****'s.* We have three colors

**Brown:** Outside chosen branch

**Red:** Lastly seen **a** in chosen branch

**Green:** Lastly seen **b** in chosen branch

For acceptance we have

**Büchi:** $\{\textbf{brown}, \textbf{red}\}$

**Muller:** $\{\{\textbf{brown}\}, \{\textbf{red}\}\}$

**Rabin:** **brown** $\mapsto 2$,**red** $\mapsto 2$, **green** $\mapsto 1$

*Trees with no branch with infinitely many* ***a****'s.* We have two colors in a deterministic automaton

**Red:** Last seen **a**

**Green:** Last seen **b**

and as acceptance

**Muller:** $\{\{\textbf{green}\}\}$

There is no Büchi automaton accepting these trees. Suppose we had one $\mathcal{B}$ with accepting colors $F$. Let the tree $T$ consist of all branches with $< n$ **a**'s. This should be accepted and there is a coloring $\mathbb{T}$ of the tree such that all branches are accepted by $\mathcal{B}$. We then find a number $k_i$ med

- $\mathbf{b}^{k_0}$ has color in $F$
- $\mathbf{b}^{k_0}\mathbf{a}\mathbf{b}^{k_1}$ has color in $F$
- ...
- $\mathbf{b}^{k_0}\mathbf{a}\mathbf{b}^{k_1}\mathbf{a}\cdots\mathbf{a}\mathbf{b}^{k_n}$ has color in $F$

One of the colors must be repeated. But then we can make a new tree $U$ with coloring $\mathbb{U}$ such that it contains a branch

$$\cdots(\cdots\mathbf{a}\cdots)^{\infty}$$

such that for all $k$

$$\cdots(\cdots\mathbf{a}\cdots)^{k} \text{ have accepted color}$$

and all the branches in $\mathbb{U}$ is also accepted. But then $\mathcal{B}$ will accept a tree with infinitely many $\mathbf{a}$'s.

**Lemma 1** *For all automata:*

$$\textit{BÜCHI} \subseteq \textit{RABIN} \subseteq \textit{MULLER}$$

Given a Büchi automaton $\mathcal{B}$. We make a Rabin automaton by letting all accepting colors have level 0 and the not-accepting colors level 1.

**Lemma 2** *For non-deterministic string automata:*

$$\textit{BÜCHI} = \textit{RABIN} = \textit{MULLER}$$

Given a Muller string automaton $\mathcal{M}$. We make an equivalent non-deterministic Büchi automaton by guessing a point in the string and guessing a set of colors and then checking whether we visit those colors infinitely often after the point in the string.

**Lemma 3** *For all automata:*

$$\textit{RABIN} = \textit{MULLER}$$

Given a Muller automaton $\mathcal{M}$ with accepting sets $M$ and assume we have $k$ colors $\mathcal{F}$. We now keep a record over the colors we visit and which colors we visited last. The record contains the following information

- the colors $\langle f_0, f_1, \ldots, f_{k-1}\rangle$ given in their order of visit
- a number $< k$

Assume that an input change color $f_{k-1}$ into color $f_n$ in $\mathcal{M}$. Then we change the record from

$$\langle\langle f_0, \ldots, f_{n-1}, f_n, f_{n+1}, \ldots, f_{k-1}\rangle r\rangle$$

to record

$$\langle\langle f_0, \ldots, f_{n-1}, f_{n+1}, \ldots, f_{k-1}, f_n\rangle n\rangle$$

The colors of the Rabin automaton is the set of all records. The level for the record

$$\langle\langle f_0, \ldots, f_m, f_{m+1}, \ldots, f_{k-1}\rangle m\rangle$$

is $2m$ if the set $\{f_m, f_{m+1} \ldots, f_{k-1}\}$ is an accepting set of colors, else it is $2m+1$.

## 3.8 Complement of infinite strings

The sets of infinite strings defined as accepting sets from finite automata are closed under complementations. There are two ingredients to the proof of this

- a normal form of strings defined by finite automata using congruence relations
- Ramsey theorem — which we are going to prove later

Given a finite automaton $\mathcal{T}$ over alphabet $\mathcal{A}$. Two finite words $\sigma$ and $\tau$ over are congruent — written $\sigma \sim \tau$ — if for any two colors $p$ and $q$ we can come from $p$ to $q$ using $\sigma$ if and only if we can do it using $\tau$. Given a finite subset $F$ of the colors they are congruent relative to $F$ — written $\sigma \sim_F \tau$ — if we in addition visit at least one of the colors of $F$.

**Lemma 4** *The equivalence relations $\sim$ and $\sim_F$ have only a finite number of equivalence classes. These are constructed from the finite automaton. Given two words we can decide whether they belong to the same equivalence class. The equivalence relations are congruence relations with respect to concatenation of words.*

Now enter Ramseys theorem. Assume we have a congruence relation on finite words like the above with a finite number of equivalence classes. Then any infinite string can be written as

$$\sigma \circ \tau_1 \circ \tau_2 \circ \tau_3 \circ \cdots$$

where all the $\tau_i$ belong to the same equivalence class. We can write this as $[\sigma] \circ [\tau]^\infty$.

Back to the complementation of strings accepted by a Büchi automaton with accepting colors $F$. We then use the equivalence classes of $\sim_F$ to get the complement.

## 3.9 Complement of infinite trees

We use the argument given by Leo Harrington and Yuri Gurevich. The main points are

- The run of a tree in a treeautomaton is considered as a game between a PATHFINDER and an AUTOMATON. In this way we represent the non-determinism of the automaton and the guessing of the path through the tree.
- acceptance/non-acceptance in the tree automaton correspond to AUTOMATON/PATHFINDER having a winning strategy in the game
- the winning strategy can be assumed to be history less
- the complementary automaton can be described as a game between winning strategies for AUTOMATON and PATHFINDER
- playing between history less strategies can be reduced to acceptance/non-acceptance of infinite strings in a string automaton
- lastly we treat decidability

Given an infinite tree $\alpha$ as input in a Rabin automaton $\mathcal{R}$. We consider this as a game between AUTOMATON $\mathbb{A}$ and PATHFINDER $\mathbb{P}$. $\mathbb{A}$ chooses one of the possible moves of $\mathcal{R}$ and $\mathbb{P}$ chooses a path through the tree $\alpha$. Through the game we shall have chosen a branch decorated by colors and we are in a winning situation if the branch is accepted by $\mathcal{R}$. The tree $\alpha$ is accepted by $\mathcal{R}$ if and only if the automaton $\mathbb{A}$ have a winning strategy for $(\mathcal{R}, \alpha)$. Player $\mathbb{A}$ tries to get an even color, while player $\mathbb{P}$ tries to get an odd color.

We first show that a winning strategy can be assumed to be history less. This is proved by induction over the finite number of levels and considering a slightly more general type of game:

- The GAME has two players ODD and EVEN
- The arena is a countable directed graph with nodes $V$ partitioned into $V_{ODD}$ and $V_{EVEN}$
- From each node there is an edge going out
- To each node in $V_{ODD}$ all the edges go to $V_{EVEN}$ — and conversely
- Each node is colored by a color from the finite set $F$
- To each color there is assigned a level — a natural number
- a run is an infinite sequence of moves and produces a path through the graph
- ODD wins if the colors met infinitely often of least level has an odd level

- EVEN wins if the colors met infinitely often of least level has an even level

If we have only one level,we have trivially a history less strategy. Consider now the case where the lowest level is even. If the lowest level is odd interchange ODD and EVEN in the argument below. We partition the nodes up into

**A:** nodes where ODD has a history less winning strategy

**B:** nodes where ODD has not a history less winning strategy

EVEN can with a start in **B** use strategies which keeps the game within **B**. Let $E$ be the nodes from **B** of lowest level — they are all of even level. We then partition **B** into

**B0:** nodes where EVEN can force in a history less way a visit in $E$ — keeping within **B**

**B1:** nodes where EVEN cannot force in a history less way a visit in $E$ — keeping within **B**

The nodes **B1** gives a game $\mathcal{H}$ where the lowest level is avoided and we can use the induction assumption to partition it into

**B10:** nodes where ODD has a history less winning strategy for $\mathcal{H}$

**B11:** nodes where EVEN has a history less winning strategy for $\mathcal{H}$

The game $\mathcal{H}$ has all the moves from $V_{EVEN}$, but some and not necessarily all from $V_{ODD}$. So a winning strategy for ODD in **B10** is also a winning strategy in the original game — and therefore **B10** is empty. Hence

$$\mathbf{B} = \mathbf{B0} \cup \mathbf{B11}$$

We then get a strategy for EVEN, and consider a game using this strategy. If EVEN starts with a node in **B**, then he can force to remain within **B** and ODD cannot move to **A**. If the game is within **B0** infinitely many times, then a node from $E$ is visited infinitely often and the game is won. Else he comes to **B11** and again EVEN has a winning strategy. If EVEN starts with a node in **A**, then ODD has a winning strategy.

Now we return to the tree automaton.

**Lemma 5** *Given a Rabin tree automaton $\mathcal{R}$. There is a Muller automaton $\mathcal{M}$ accepting exactly the trees which $\mathcal{R}$ do not accept.*

The Rabin automaton $\mathcal{R}$ is given by

- states $\mathcal{Q}$ with initial state $q \in \mathcal{Q}$
- alphabet $\mathcal{A}$

- transition relation $\mathcal{T}$ in $\mathcal{Q} \times \mathcal{A} \times \mathcal{A} \times \mathcal{Q} \times \mathcal{Q}$
- level function $\mathcal{Q} \to \mathcal{N}$

Then note

- not acceptance of a tree $T$ in $\mathcal{R}$ means that the pathfinder $\mathbb{S}$ has a strategy to show that the tree is not accepted
- this strategy can be assumed to be history less — it is a function which to a node in $T$, a state and a symbol gives a direction
- the strategy can be seen as a tree $S$ in the extended alphabet $\mathcal{Q} \times \mathcal{A} \to \{\mathbf{left}, \mathbf{right}\}$ — and is called the strategy tree for the pathfinder
- the strategy tree $S$ gives a path $\pi_S$ through $T$
- there is a Muller string automaton $\mathcal{A}$ which checks if $T|\pi_S$ is not accepted
- whether there is such a strategy tree $S$ can be treated — using non-determinism — by a Muller tree automaton $\mathcal{M}$ accepting exactly those trees which $\mathcal{R}$ do not accept

We now show that it is decidable whether there is a tree accepted by an automaton. Consider first the special case where the alphabet has only one symbol. Then the tree is unique — and equal to any subtree in it. The automaton $\mathcal{R}$ may have many states and may be non-deterministic. Now all nodes are equal and the history less strategies are for the automaton

$$\mathbf{state} \to \mathbf{state} \times \mathbf{state}$$

and for the pathfinder

$$\mathbf{state} \to \mathbf{direction}$$

There are only a finite number of such strategies and we can play the strategies against each other to decide acceptance of $\mathcal{R}$.

For a general automaton $\mathcal{B}$ we use non-determinism to get a large automaton with only one symbol and answering the question whether there is a tree accepted by $\mathcal{B}$.

# Chapter 4

# Diagonalization

## 4.1 Cantors argument

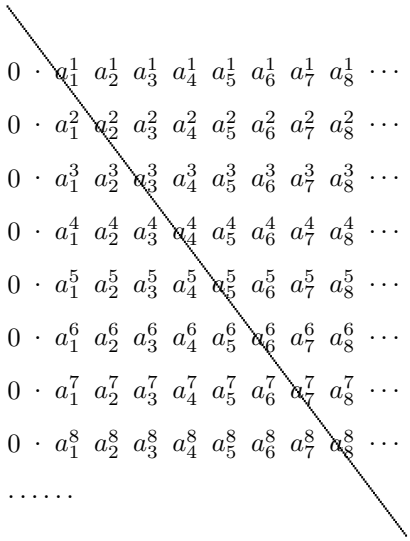
We all know Cantors diagonal argument. In this chapter we spell it out in a little more detail than usual to make clear the underlying technique.

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

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

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 $\Phi$. 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 $\Phi$ 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 $\Phi x$ be a formula with free variable $x$. Then put up the two dimensional schema obtained by composition of formulae and then apply $\Phi x$ on the diagonal. We then get a formula $\psi x$ such that

$$\psi x \leftrightarrow \Phi\ulcorner \psi x \urcorner$$

A closer look reveals the following

**Lemma 6 (Fix point)** *Let $x$ be a variable and $Ax$ a formula with free variable $x$. There is then a sentence $B$ with*

$$B \leftrightarrow A\ulcorner B \urcorner$$

**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

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

And

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

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. 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 Craig Smoryński's and George Boolos work.

## 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 $\Sigma_1$-induction and we have a fair coding of equality and inequality. Then for all $\Sigma_1$ 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 $\omega$-*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 $\omega$-consistency implies consistency. Instead of $\omega$-consistency it may be more perspicuous to use the following consequence

**Theorem 6** *If the theory is $\omega$-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 $\omega$-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 $\nvdash G$*

**$\omega$-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 $\omega$-consistent**. Assume $\vdash \neg G$. Then $\vdash \Box G$ and from the above by $\omega$-consistency $\vdash G$. This contradicts the consistency of the theory. ■

Under the weak assumption of $\omega$-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 $\Gamma$ is falsifiable if there is a falsification of each of the formulas contained in it. We have the usual shorthand $\Gamma, F$ and $F, G$ and so on for sequents. We write $\mathfrak{F} : \Gamma$ for $\Gamma$ 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 $\Lambda$ 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:** $\Lambda$ is not falsifiable by containing both an atomic formula and its negation. If $\bot$ and $\top$ is part of the language, then $\Lambda$ is not falsifable if it contains $\top$

**Falsifiable:** $\Lambda$ is falsifable and $\Gamma$ is empty.

Given a sequent $\Gamma$ we can write down the analysis tree over $\Gamma$ 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 7 (Analysis)** *Assume that the sequent $\Gamma$ is falsifiable with a Kripke model on a finite tree as frame, then any analysis tree $\mathbb{T}$ over $\Gamma$ 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 $\Theta$ 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:** $\Lambda$ is falsifiable, and $\Gamma$ is empty. Then take this as a top node in the witnessing subtree. Observe that we there have $\Diamond\Delta$ trivially falsifiable.

  **Continue:** $\Lambda$ is falsifiable and $\Gamma$ 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. ■

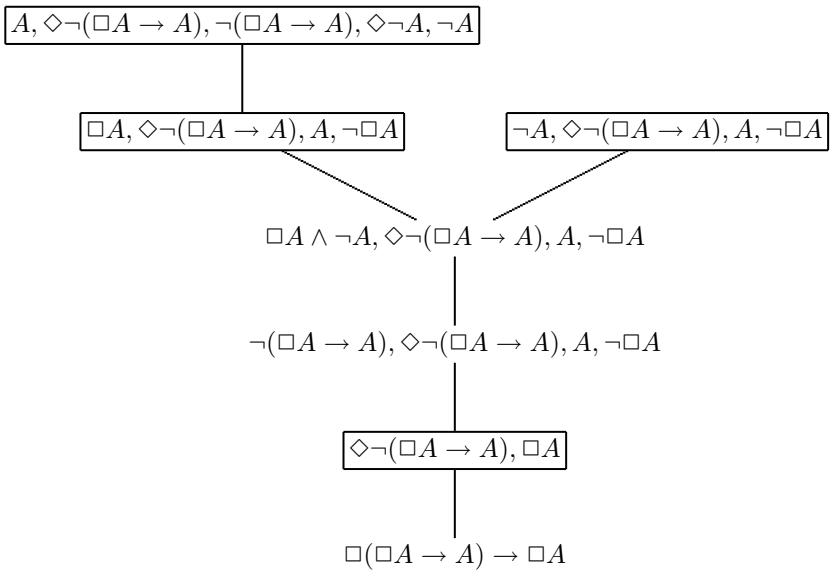
**Lemma 8 (Synthesis)** *Assume that the analysis tree $\mathbb{T}$ over $\Gamma$ is falsified. Then we can find a Kripke model over a finite tree which falsifies $\Gamma$*

**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. ■

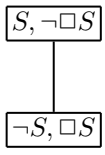
From these two lemmas we get

**Theorem 9 (Completeness of GL)** *A sequent $\Gamma$ 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).*

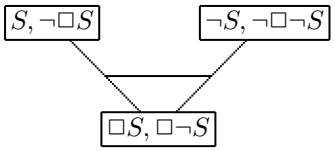
We look at an analysis tree for GL2:




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 — 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

$$\begin{aligned} h(0) &= 0 \\ h(x+1) &= \begin{cases} j & \text{, where } j \succ h(x) \text{ and } x \text{ proves } \exists y > x \cdot h(y) \succ j \\ h(x) & \text{, otherwise} \end{cases} \end{aligned}$$

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

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

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 9**

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

**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 $F$

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

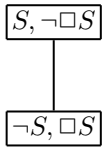
We use induction over $B$. If $B$ is atomic, then $B^\star$ is a disjunction of $S_i$s. Assume $1 \models B$. Then by construction of 0 we have $0 \models B$ and $S_0$ is a disjunct of $B^\star$. So $\vdash S_0 \rightarrow B^\star$. Now assume $1 \not\models B$. By construction of 0 we have $0 \not\models B$ 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.

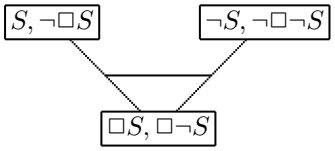
What is the complexity of the interpreted sentences? We observe that for a topmost node $j$ we can write $S_j = \exists x \cdot h(x) = j$ since we cannot climb further, and we get $S_j$ a $\mathbf{\Sigma}_1$-sentence. For a node $i$ lower down we can write

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

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. 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 first order 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

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

**$\omega$-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.

We note that the connectives are supposed to be countable. The reason for this is that the (cut-free) calculus is simply not complete for uncountable connectives. Consider the valid formula

$$\bigwedge_{i\in\omega} \bigvee_{j\in 2} Aij \to \bigvee_{f\in 2^\omega} \bigwedge_{i\in\omega} Aifi$$

Assume that we have a cut-free derivation of it. This derivation is a well-founded tree with countable branching. It only involves a countable number of instances of the rule $\bigwedge$. That means that there is a countable subset $F$ of $2^\omega$ such that

$$\bigwedge_{i\in\omega} \bigvee_{j\in 2} Aij \to \bigvee_{f\in F} \bigwedge_{i\in\omega} Aifi$$

is derivable. But this formula is not valid as a simple diagonalization argument shows.

To get the sequential calculus version of our elementary systems we use predicate logic with equality, extra axioms we have for true $\Delta_0$ 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 $\Gamma$.

**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:

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

**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 (as long as we have only countable disjunctions). 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 $\Gamma$ being falsifiable. Then we have

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

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 $\Gamma$ that we want to falsify. We write down a falsification tree above $\Gamma$ as follows:

- start with the sequent $\Gamma$ 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 10 (Synthesis)** *Suppose we have a fair falsification tree over* $\Gamma$ *with a branch not containing an axiom, then there is a falsification of* $\Gamma$.

**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 $\Gamma$ are false. ■

**Lemma 11 (Analysis)** *Assume we have a falsification of* $\Gamma$. *Then every falsification tree over* $\Gamma$ *contains a branch without any axiom.*

**Proof:** Assume we have a falsification $I$ of $\Gamma$. 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* $\Gamma$ *is valid if and only if there is a falsification tree over* $\Gamma$ *contain axioms in all branches if and only if any fair falsification tree over* $\Gamma$ *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.

**$\omega$-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 $\Sigma_1$. Truth / falsity in the model is then also $\Delta_2$. 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 $\Delta_2$ 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 12** *We have syntactical operations as below which neither increase height nor rank:*

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

We talk about:

**Connected:** Look at the rules of sequential calculus. An occurrence of a part of $\Gamma$ (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 $\Gamma$. 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 $\Gamma$ occurs in $\Delta$. Then add $\Delta$ 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 13 (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:

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

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 14 (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 $\Delta$. 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 $\Gamma, \exists x.Fx$:

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

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

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

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* $\Gamma$ *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 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.

# Chapter 8

# Analysis and synthesis

## 8.1 Completeness

In the completeness theorem we had two main lemmas – analysis and synthesis. They reflect different ways of thinking:

**Analysis:** We start with a sentence $F$ and then systematically breaks down to smaller and smaller pieces in order to derive a falsification. The falsification is given by the term model constructed from an infinite branch and we prove by induction over the build up of sentences that the falsification does what it should do.

**Synthesis:** We start with a wellfounded tree with axioms at the leaves, and then show how we can derive the sentence at the root. Here we have an induction over the wellfounded tree.

Now we have the difference between direct and indirect derivations. The direct derivations do not use the cut rule. And we showed that the cut rule could be eliminated. In the proof we used in an essential way that the derivations were wellfounded trees.

Our cut elimination could be called a theorem of direct synthesis. We miss a theorem of direct analysis. This will be given in this chapter.

## 8.2 Direct analysis

We use a system of sequential calculus where we have the usual rules

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

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

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

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

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

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

In rule $\forall$ we have the usual eigen variable condition. We can do a number of syntactical transformations from the bottom upwards

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

Let us look at the second transformation. We start with a derivation with $\Gamma, F \wedge G$ at the root. Then look at the thread above $F \wedge G$ and replace all $F \wedge G$ in that thread with $F$ until we come to a place where the thread is introduced by an $\wedge$-rule. We chop off the right premiss and replace the $\wedge$-rule with **triv**-rule. Similarly with all the other transformation. To avoid clashes with eigen variables (as for example in the first transformation) we may have to change the name of them. This can also be done from the bottom upwards.

We have previously seen the complications when we analyze an instance of an $\exists$-sentence. The analysis starts a process where it try first one term, then later another one and so on. This process is simulated in our syntactic transformation. The key is to remember the derivations of $\forall$-sentences and use them whenever we come to an analysis of an $\exists$-sentence.

The analysis can be seen as a stream — we see the root of the derivation and a few steps above it but do not know whether the analysis breaks off or continues indefinitely. The goal here is to construct a transformation of this stream to a stream with the same root and not containing cuts. To do this we keep track of extra information:

- the threads of variables which we want to change
- the threads of conjunctions $F \wedge G$ which we want to change into one of the conjuncts

- the threads of disjunctions $F \vee G$ which we want to change into $F, G$
- the threads of existential $\exists x.Fx$ and analysis of $\Gamma, \forall x.\overline{Fx}$ where we do the following
  - the formula $\exists x.Fx$ is deleted
  - if $\exists x.Fx$ is introduced by $\Delta, Ft, \exists x.Fx$, then we replace it by a cut between $\Delta, Ft$ and $\Gamma, \overline{Ft}$
  - we continue to keep track of the thread $\exists x.Fx$ and the analysis of $\Gamma, \forall x.\overline{Fx}$ as we go further up the stream and may encounter further introductions of $\exists x.Fx$

Now we can define the transformation of an analysis using cut to a cutfree analysis

- we work with the analysis from the root upwards and keeping track of the appropriate threads
- if the downmost step is not a cut, then go further upwards
- if the downmost step is a cut then we see whether the cutformula is a connective-formula, a quantifier-formula or a litteral
  - connective: we get cuts with smaller formula around the root
  - quantifier: we get cuts with smaller formula but higher up in the analysis where we introduce the $\exists$-formula connected with the cut
  - litteral: we chose one of the literals – say the left – $L$, replace the cut with the trivial rule from the left premiss, in the thread above $L$ erase the formula until we come to an axiom involving $L, \neg L$. Then use the right premiss from there on.

## 8.3 Higher order logic

In the analysis part we rely heavily on the subformula property. When we break a formula up into parts, then the parts are smaller than the formula. This breaks down in second order logic. Say we start with the formula

$$\forall \phi.\phi$$

We would like to specialize the quantifier. Say we insert the proposition

$$\forall \phi.\phi$$

for $\phi$, we get the same formula. So substitution of a second order term does not necessarily make things shorter. This is different from the first order case.

On the other hand in second order logic we can express the induction in arithmetic in a direct way. The induction axiom

$$\forall\psi.(\psi 0 \wedge \forall x.(\psi x \rightarrow \psi sx) \rightarrow \forall y.\psi y)$$

has as a consequence all the instances of induction. Similarly with other datastructures.

In second order logic we can express validity in first order logic for a given sentence in a straightforward way. And we can express not validity. This shows that we cannot have a completeness theorem in second order logic. For else we could have a computation of whether a first order sentence is not valid.

We now — following work of William Tait, Dag Prawits and Moto-o Takahashi — look closer at the possibility of treating higher order logic with a sequential calculus. This calculus will not be complete, but it may nevertheless be useful. Our proof of the completeness theorem for predicate logic can be used to show that any formula which is derivable with the cut rule can also be derived without the cut rule. We do not need to speak about validity at all. The proof uses the following steps for a formula $F$

1. Construct a fair falsification tree over $F$
2. If the fair falsification tree is a derivation, then $F$ is provable without cut
3. If the fair falsification tree is not a derivation, then there is a falsification of $F$
4. If there is a falsification of $F$, then $F$ is not provable with cut

The crucial part is step 3 — what we have called the analysis part. Let us call a set of formulas $\mathcal{F}$ for an anlytic set if we have the following:

$$\text{for no literal } L \text{ not both } \mathcal{F} : L \text{ and } \mathcal{F} : \neg L$$

and

$$\begin{aligned}
&\mathcal{F} : G \wedge H \Rightarrow \mathcal{F} : G \text{ or } \mathcal{F} : H\\
&\mathcal{F} : G \vee H \Rightarrow \mathcal{F} : G \text{ and } \mathcal{F} : G\\
&\mathcal{F} : \forall x.Gx \Rightarrow \mathcal{F} : Ga \text{ for some free variable } a\\
&\mathcal{F} : \exists x.Gx \Rightarrow \mathcal{F} : Ga \text{ for all } a
\end{aligned}$$

We now want to prove that from an analytic set $\mathcal{F}$ we get a falsification of all formulas in $\mathcal{F}$. In predicate logic this is proved by

- Given an analytic set $\mathcal{F}$ in predicate logic
- We construct a falsification by
- The universe of the falsification is the set of terms in $\mathcal{F}$
- An atomic $A$ is **false** if it occurs in $\mathcal{F}$, else it is **true**

Then by induction over the lengths of formulas we prove that every formula in $\mathcal{F}$ has value **false**. This proof breaks down in higher order logic. We are not able to have the induction over formulas. In the falsification we must also give an interpretation of terms of higher order.

We use the fair branch to give a falsification of the formula in the branch. For the falsification we use as universe all terms in the branch and the literals occurring in the branch are supposed to be false. Then we note

- Not all formulas in the language are given a truthvalue
- We have been vague so far about what we mean by all terms

The first point does not matter. We are free to choose any truthvalue for atomic formulas not occurring in the branch. In predicate logic without equalit and with no function symbols, the terms are the variables occurring. If we have function symbols, then we must close under application of functions. With equality we must introduce new rules in the sequential calculus and the universe are going to be equivalence classes of terms.

Let us now see what happens in the higher order case. Then we must make some decision about what should be considered as a higher order term. Here are some possibilities

- the higher order terms are atomic — there are no operations on terms
- we get new terms by the usual logical constructions
- we have operations on terms like in set theory
- we have also some equality relation between terms

In the first case the higher order terms are just like new sorts in a many-sorted calculus. This does not give anything new beyond predicate calculus. In particular we would not be able to develop Peano arithmetic within such a system. The second possibility is the one which we want to investigate further. The fourth possibility can be combined with the other three — and we shall discuss it later.

So we have new terms for the usual logical constructions. This is how sequential calculus for higher order logic is usually formulated. An analytic set is a set of formulas $\mathcal{F}$ satisfying

$$\text{for no literal } L \text{ not both } \mathcal{F} : L \text{ and } \mathcal{F} : \neg L$$

and

$$\begin{aligned}
&\mathcal{F} : G \wedge H \Rightarrow \mathcal{F} : G \text{ or } \mathcal{F} : H \\
&\mathcal{F} : G \vee H \Rightarrow \mathcal{F} : G \text{ and } \mathcal{F} : G \\
&\mathcal{F} : \forall x.Gx \Rightarrow \mathcal{F} : Ga \text{ for some free variable } a \\
&\mathcal{F} : \exists x.Gx \Rightarrow \mathcal{F} : Ga \text{ for all } a
\end{aligned}$$

where in the quantifiers we have variables and terms of the appropriate types. So we have

- at ground type : terms denoting elements
- at level 1 : terms denoting propositions
- at level 1 : terms denoting relations between elements
- at level 1 : terms denoting functions between elements
- and so on at higher levels

The typical analytic set is from a branch of a fair falsification tree not containing an axiom. So assume we have an analytic set $\mathcal{A}$. We now want to define a falsification $\mathcal{F}$. We build it up in steps

- The universe of $\mathcal{F}$ is the set of all terms of ground type
- The literals occurring in $\mathcal{A}$ are given truthvalue **false**

To make the falsification total we let

- The atomic formulas not occurring in $\mathcal{A}$ are given truthvalue **true**

Now we must decide what to do with the second order notions. The second order terms are built up from free second order variables using the usual constructions of second order logic. Now we observe that we have already given an interpretation of the free second order variables in the falsification $\mathcal{F}$ — they occur in $\mathcal{A}$ as literals. Using ordinary second order comprehension we get interpetations of all second order terms. This gives the falsification $\mathcal{F}$. The construction is heavily non-constructive, but is no problem for an infinite mind. Now we have

- All formulas in $\mathcal{A}$ are given truthvalue **false** under $\mathcal{F}$

This construction shows that the cut rule can be eliminated from second order logic. For assume we have a derivation of a formula $F$ using cut

$$\mathbf{cut} \vdash F$$

Then construct a fair analysis over $F$. This analysis cannot contain an analytic branch. Else we would get a falsification of $F$. So it must contain a cut-free derivation

$$\mathbf{cut\text{-}free} \vdash F$$

The construction does not give completeness of second order logic. We have constructed our second order falsification to fit our job. We have taken as subsets in our universe those subsets given by the second order terms starting with the subsets giving interpretations of the free variables in our term model and then closing under second order constructions. This is not what a second order completeness would require.

## 8.4 Interpolation

As the most important arguments in predicate logic we take those given by

**Gottlob Frege:** The language and a calculus

**Thoralf Skolem:** The algorithmic treatment of falsification using terms giving both completeness and compactness

**Gerhard Gentzen:** The treatment of direct proofs

**William Craig:** Tracing the parts of a formula within a direct proof

Here we comes to Craigs interpolation theorem. Given a formula and some parts of it. The parts can be traced within a direct proof and gives information about the proof of parts from a proof of the formula. This can be expressed with the interpolation theorem proved below.

We write $\Gamma \circ \Delta$ for a partition of a sequent into two parts. We do not require that $\Gamma$ and $\Delta$ are disjoint.

**Definition 1** *Assume* $\vdash \Gamma \circ \Delta$. *A formula* $F$ *is a separating formula of the sequent relative to the partition into* $\Gamma$ *and* $\Delta$ *if*

- $\vdash \Gamma, F$
- $\vdash \Delta, \neg F$
- $F$ *is built up from* $\top$, $\bot$ *and symbols occurring in both* $\Gamma$ *and* $\Delta$ — *in particular the free variables of* $F$ *are free in both* $\Gamma$ *and* $\Delta$

$\Gamma$ *is said to be the negative part of the partition and* $\Delta$ *is the positive part for the separating formula* $F$.

**Theorem 14 (Interpolation)** *Assume* $\vdash \Gamma \circ \Delta$. *Then we can find a separating formula for it.*

We are going to show that the separating formulas propagate through the proofs starting with the axioms and going through the cut free rules from the premisses to the conclusion. We shall look at individual rules. Observe that for all the rules — except cut — a partition of the conclusion induces in a natural way partitions of the premisses. Take as an example the rule

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

Assume we have a partition of the conclusion into $\Delta \circ, E, F \wedge G$. Then the two premisses are partitioned into $\Delta \circ E, F$ and $\Delta \circ E, G$.

**Definition 2** *A rule is separable if for any partition of the conclusion and for any separating formulas of the induced partitions of the premisses, we can construct a separating formula of the conclusion. An axiom is separable if we can find a separating formula for any partition.*

We assume that we have a first order language without function symbols. And then we show that the axioms and the cut free rules are separable and for the propagation of the separating formulas we have the following table

| rule | negative part | positive part |
|---|---|---|
| $\wedge$ | $\vee$ | $\wedge$ |
| $\vee$ | – | – |
| $\forall$ | – | – |
| $\exists$ | – / $\forall$ | – / $\exists$ |
| $=$ | – | – |

There are many cases to consider, but they are all simple. We take some typical cases below. The remaining cases are similar to one of the cases we have written down.

*Axiom which is split by the partition* Assume that we have $\vdash \Gamma, L \circ \neg L, \Delta$. Then we can use $\neg L$ as separating formula.

*Axiom which is not split by the partition* Assume that we have $\vdash \Gamma, t = t \circ \Delta$. We can use $\bot$ as separating formula.

$\wedge$*-rule in the negative part* We have a partition $\vdash \Gamma, F \wedge G \circ \Delta$ and assume there are separating formulas $I$ and $J$ of the premisses with

$$\vdash \Gamma, F, I \qquad \vdash \Delta, \neg I$$
$$\vdash \Gamma, G, J \qquad \vdash \Delta, \neg J$$

Then by ordinary rules we get

$$\vdash \Gamma, F \wedge G, I \vee J \qquad \vdash \Delta, \neg(I \vee J)$$

and $I \vee J$ is a separating formula of the conclusion.

$\vee$*-rule or* $\forall$*-rule in the negative part* We can use the separating formula of the premiss as a separating formula of the conclusion. In the $\forall$-rule it is essential that the separating formula does not contain more free variables than are common to both parts.

$\exists$*-rule in the negative part* This is slightly complicated and we must use the assumption that we do not have function symbols. Then all terms are variables. Assume that we have a partition of the premisse $\vdash \Gamma, Ft \circ \Delta$ and there are separating formula $I$ with

$$\vdash \Gamma, Ft, I \qquad \vdash \Delta, \neg I$$

Here we must be careful with the free variable $t$. If $t$ does not occur in $I$ we can use $I$ as a separating formula of the conclusion. The same if $t$ occurs both in $\Gamma, \exists x Fx$ and $\Delta$. The case when $t$ occurs in $I$ (as $It$) and in $\Delta$ but not in $\Gamma, \exists x Fx$ remains. Then we use that $t$ is a variable and get

$$\vdash \Gamma, \exists x Fx, \forall y Iy \qquad \vdash \Delta, \neg \forall y Iy$$

*=-rule which is separated by the two parts* Assume we have partition $\vdash \Gamma, \neg s = t \circ Rt, \Delta$ with separating formula $It$. If $t$ does occur in $Rs, \Delta$, then we can use $It$ as separating formula of the conclusion. Else we have

$$\vdash \Gamma, \neg s = t, It \qquad\qquad \vdash \Delta, Rt, \neg It$$

By =-rule $\vdash \Gamma, \neg s = t, Is$. Since $t$ does not occur in $\Delta$ and is a free variable since there are no function symbols we can substitute $s$ for $t$ and get $\vdash \Delta, Rs, \neg Is$

*=-rule which is one of the parts* The separating formula of the premisse can also be used in the conclusion.

*The remaining cases* Similar to one of the cases above.

□

What to do for when there are terms with function symbols? There are two options

- relax the assumptions about free variables in the definition of separating formula. Then we get into trouble with the $\forall$-rule, but the $\exists$-rule and =-rule go through in full generality
- say we have unary function symbol $f$. We get rid of it by using binary relation symbol $F$, rewriting the sequents using $F$ and add extra axiom $\forall x \exists y \forall z \cdot Fxz \leftrightarrow y = z$

The second option gives the interpolation theorem in full generality. The first option allows one to treat special cases in a sharper way.

## 8.5 Applying interpolation

Now let us give some applications. The first is Beths definability theorem – the standard application.

**Definition 3** *Assume we have a propositional expression $\mathcal{F}(P)$ where $P$ is an atomic formula. We say that it defines $P$ implicitly if for a new atomic formula $Q$ we have $\vdash (\mathcal{F}(P) \leftrightarrow \mathcal{F}(Q)) \rightarrow (P \leftrightarrow Q)$.*

**Theorem 15 (Beth)** *Assume that $\mathcal{F}(P)$ defines $P$ implicitly. Then there is a formula $R$ in the language of $\mathcal{F}$ with no occurrence of $P$ with $\vdash \mathcal{F}(P) \rightarrow (P \leftrightarrow R)$. We say that $R$ defines $P$ explicitly.*

By assumption

$$\vdash \mathcal{F}(P) \wedge P \rightarrow (\mathcal{F}(Q) \rightarrow Q)$$

There is a separating formula $R$ in the language of $\mathcal{F}$ alone with

$$\vdash (\mathcal{F}(P) \wedge P) \rightarrow R \qquad\qquad \vdash R \rightarrow (\mathcal{F}(Q) \rightarrow Q)$$

Substituting $P$ for $Q$ and regrouping gives the result. □

Beth definability theorem shows that interpolation is not true in arithmetical theories. We know from recursion theory that we can define the recursive functions implicitly in the language of $= 0\ s\ +$ and $\times$, but there are no explicit definition of all recursive functions in the language.

In the proof above the interpolation formula was computed from the derivation of $\vdash \Gamma \circ \Delta$ and not only from the partition itself. A short argument shows that this is necessary. Assume we have a total computable function $\phi(\Gamma, \Delta)$ which gives an interpolation formula if such a formula exists. Then we can use it to separate between $\vdash \Gamma$ and $\vdash \Delta$. For write $\Delta$ in a new alphabet distinct from the alphabet of $\Gamma$. An interpolation formula must be built up from $\top$ and $\bot$ alone using propositional connectives — and hence equivalent to either $\top$ or $\bot$. In case it is $\top$ we get $\vdash \Delta$, and in case it is $\bot$ we get $\vdash \Gamma$. This is impossible and we cannot have a total computable function giving the interpolant as a function of the partition of the sequent.

The interpolation theorem is obviously true for many sorted predicate logic. Using two sorts we can express that a formula is closed under extensions. Let us have a new sort $\star$ and let for any formula $F$ in the old sort $F^\star$ be the same formula written entirely in the new sort. Then we can express that $F$ is closed under extensions relative to axioms $A$ by

$$\vdash A \wedge A^\star \wedge F \wedge \forall x \exists x^\star \cdot x = x^\star \rightarrow F^\star$$

We then get an interpolant $I$ with

$$\vdash A \wedge F \wedge \forall x \exists x^\star \cdot x = x^\star \rightarrow I$$
$$\vdash A^\star \wedge I \rightarrow F^\star$$

Now the crucial observation is that the interpolant can only contain variables of the $\star$-sort and that the negative part contains only one $\star$-quantifier — $\exists x^\star$ — which becomes $\forall x^\star$ after having written it in the usual form. But then there are no $\exists$ in the negative part to produce $\forall$ in the interpolant. The interpolant contains only $\exists$-quantifiers. Now regrouping and setting the two sorts to be the same give

$$\vdash A \rightarrow (F \leftrightarrow I)$$

In this argument we have assumed that there are only relation symbols in the language. But the axioms $A$ may contain statements saying that some of the relation symbols correspond to functions and this gives the result also for languages containing function symbols.

Let us sidestep a little to propositional modal logic. There we have two new operators $\Box$ and $\Diamond$ which can be interpreted as

- $\Box A$ — for all later worlds $A$ true
- $\Diamond A$ — there exists a later world with $A$ true

Propositional logic is easily extended to the two new operators using de Morgans laws

$$\neg\Box F = \Diamond\neg F \qquad\qquad \neg\Diamond F = \Box\neg F$$

For sequents $\Gamma, \Delta$ we use the notation $\Box\Gamma, \Diamond\Delta$ for $\Box G_1, \ldots, \Box G_m, \Diamond D_1, \ldots, \Diamond D_n$ etc

Now we can make sequential calculi of various modal logics. We shall look at the interpolation theorems for the logics. Then we have to prove that the appropriate rules are separable. We first take the thinning rule

$$\frac{\Gamma}{\Gamma, \Delta}$$

This is obviously a separable rule. Here are some rules which are also separable

$$\frac{\Gamma, F}{\Box\Gamma, \Diamond F}$$

$$\frac{\Gamma, F, \Diamond F}{\Gamma, \Diamond F}$$

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

Here $\Diamond F$ stands for a single formula, not a sequence of formulas. For the partition of the conclusion of the inference the formula must be on one or the other side of the partition That is all that is required to make the rules separable.

**Theorem 16** *The interpolation theorem holds for the modal logics* $K, K4, GL, S4$.

# Chapter 9

# $\Sigma_1^0$-proof theory

## 9.1 Assumptions

Logic tries to theorize about the art of thinking from assumptions. Assumptions are of many kinds. We can roughly distinguish between

**$\Sigma_1^0$-assumptions:** We assume that an object is given as something finite

**$\Pi_1^1$-assumptions:** We assume something is true for all finite objects

**$\Pi_2^1$ -assumptions:** We assume something true for all countable ordinals

We shall return to the $\Pi_1^1$- and the $\Pi_2^1$-assumptions and the proof theory we get from them. Now we look at $\Sigma_1^0$-assumptions. As assumption we may have a concrete finite object — it could be a number or a derivation. Our application here is that there is a connection between the size of a finite object and the size of the derivation that the object have a certain property.

## 9.2 Predicative versus impredicative

The distinction between the different proof theories is from Jean Yves Girard. Others have used the distinction between predicative and impredicative (non-predicative) theories for something similar:

V.N. Grishin and A.G. Dragalin writes in the Encyclopaedia of Mathematics about predicativity

> A special way of forming concepts, characterized by the absence of a "vicious circle" in the definitions: the object to be defined should not participate in its own definition. If the language in which the definitions are stated is formalized, then predicativity means, as a rule, that the defining formula should not contain a bound variable whose domain of variation includes the object to be defined.

> A non-predicative definition, on the other hand, is distinguished by the presence of a "vicious circle" within it. The phenomenon of non-predicativity can also be encountered in certain reasoning wherein the process of substantiating a certain part of the reasoning under consideration is itself considered as an object of reasoning. It is exactly the use of this kind of reasoning that is a ground for the appearance of semantic antinomies (cf. Antinomy). A typical example is the contradiction in the liar's paradox: If someone states "I am lying" , then this statement can be neither true nor false.

But the predicative/impredicative distinction are used for different purposes

**$\Sigma_1^0$:** Induction over natural numbers is accepted by reflecting over how the natural numbers is built up. This argument holds only for induction over predicates where we do not assume that the totality of natural numbers as something given — we only use predicates where there are no $\forall$-quantifier over the numbers themselves. This critique is a background for Edward Nelsons predicative arithmetic.

**$\Pi_1^1$:** We use an impredicative argument in constructing real numbers from say Dedekind cuts. An alternative way goes through ramified type theory and their analysis by Kurt Schütte and Sol Feferman — and in particular the ordinal $\Gamma_0$ as the limit of predicative reasoning.

We shall not talk more about predicative/impredicative distinction, but prefer to use the logical complexity of the assumptions and in particular whether they are $\Sigma_1^0$, $\Pi_1^1$ and $\Pi_2^1$.

## 9.3 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

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

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

**Theorem 17** *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 15** *For all $n$ NUM, $\star \vdash N_n 0$*

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, $\star \vdash N_6 0$. Then using NUM, $\star \vdash N_5 0$ we get NUM, $\star \vdash N_5 e00$. Going on we get NUM, $\star \vdash N_4 ee000$ 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.

## 9.4 Short proofs

The (Kalmar) elementary functions are built up from

- 0 — zero
- $s$ — successor
- $-$ — modified subtraction

using

- composition
- diagonalization — e.g. from $Fxy$ to $Fxx$
- bounded sum
- bounded product

If we consider the corresponding predicates they are closed under

- negation
- conjunction
- disjunction
- bounded quantifiers

- bounded search

So the Kalmar elementary functions and predicates are what we need to define syntax and syntactical properties. In Skolems 1923 paper he developed number theory and the development was done within the elementary functions and predicates. Furthermore our cut elimination for predicate logic could also have been developed within the elementary functions. This suggests that cuts in predicate logic are useful only for elementary functions and predicates. In this section we make this suggestion more precise.

In the language we have enough for the elementary functions

$$0 \qquad s \qquad -$$

and possibly more function symbols, the unary predicate symbol $\mathcal{N}x$ (for $x$ is a natural number), equality $=$. The logical system has as axioms

**Basic axioms:**
- $0 : \mathcal{N}$
- $\forall x : \mathcal{N}.sx : \mathcal{N}$
- $-sxy = 0 \vee -sxy = s - xy$

**Recursion equations:**

$$\begin{aligned} p0 &= 0 \\ psx &= x \\ -x0 &= 0 \\ -xsy &= p - xy \\ +0y &= y \\ +sxy &= s + xy \\ *0y &= 0 \\ *sxy &= +y * xy \end{aligned}$$

We can also have recursion equations which goes beyond the elementary functions. Here are the equation for exponential tower and for the Ackermann-function

$$\begin{aligned} t0 &= 0 \\ tsx &= etx0 \\ aoy &= sy \\ asx0 &= axs0 \\ asxsy &= axasxy \end{aligned}$$

These functions are not elementary, and we have for example that the lengths of the proofs of $tn : \mathcal{N}$ is not elementary as a function of the numeral $n$.

Assume that for function symbol $f$ that we have an elementary function $P$ which to numeral $n$ gives a proof $Pn$ of $\vdash fn : \mathcal{N}$, then $f$ is elementary. For then we can in an elementary way find a cut free proof of $fn : \mathcal{N}$ and from that proof we can read off the value of $fn$. Hence $f$ must then be elementary. Here we shall show the converse. Given an elementary function $gx$ we can from its definition read off in a uniform way an elementary proof $Px$ of $\vdash gx : \mathcal{N}$.

In our analysis there are two important notions

**Progressive predicate:** A predicate $\mathcal{P}$ in one free variable is progressive if it behaves like $\mathcal{N}$ — we can prove $0 : \mathcal{P}$ and $\forall x : \mathcal{P}.sx : \mathcal{P}$.

**Accessible term:** A term $txy$ is accessible if there are progressive predicates $\mathcal{K}$ and $\mathcal{L}$ such that $\forall x : \mathcal{K}.\forall y : \mathcal{L}.txy : \mathcal{N}$ is provable. Similar for terms with more arguments.

We have two operations on progressive predicates

**Conjunction:** The conjunction of two progressive predicates is progressive. Note that we may have a conjunction progressive without being able to prove that the conjuncts are.

**Composition:** The composition $\mathcal{P}[\mathcal{Q}]$ of two progressive predicates $\mathcal{P}$ and $\mathcal{Q}$ is the predicate we get from $\mathcal{P}$ by substituting $\mathcal{Q}$ for $\mathcal{N}$.

The predicates connected with addition, multiplication, exponentiation and subtraction are all progressive

$$\begin{aligned}\mathcal{A}x &= \forall y : \mathcal{N}. + xy : \mathcal{N}\\ \mathcal{M}x &= \forall y : \mathcal{A}. * xy : \mathcal{N}\\ \mathcal{E}x &= \forall y : \mathcal{N}.exy : \mathcal{N}\\ \mathcal{D}x &= \forall y : \mathcal{N}. - xy : \mathcal{N}\end{aligned}$$

For subtraction we need the extra axiom.

The arithmetical operations are accessible. We can prove

$$\begin{aligned}\forall x : \mathcal{A}.\forall y : \mathcal{N}. + xy : \mathcal{N}\\ \forall x : \mathcal{M}.\forall y : \mathcal{A}. * xy : \mathcal{N}\\ \forall x : \mathcal{E}.\forall y : \mathcal{N}.exy : \mathcal{N}\\ \forall x : \mathcal{D}.\forall y : \mathcal{N}. - xy : \mathcal{N}\end{aligned}$$

**Theorem 18** *All Kalmar elementary functions are accessible.*

**Startfunctions**

We have already shown that the startfunctions — successor and subtraction — are accessible.

**Composition**

Assume that $fxy$ and $guv$ are accessible with the following progressive predicates

$$\forall x : \mathcal{J}.\forall y : \mathcal{K}.fxy : \mathcal{N}$$
$$\forall u : \mathcal{L}.\forall v : \mathcal{M}.guv : \mathcal{N}$$

But then we get

$$\forall x : \mathcal{J}[\mathcal{L}].\forall y : \mathcal{K}[\mathcal{L}].\forall v : \mathcal{M}.gfxyv : \mathcal{N}$$

**Diagonalization**

We have

$$\forall x : \mathcal{J} \wedge \mathcal{K}.fxx : \mathcal{N}$$

**Bounded sums and products**

We first write down the recursion equations

$$\Sigma_{i<0}fiy = 0$$
$$\Sigma_{i<sx}fiy = +fxy\Sigma_{i<x}fiy$$
$$\Pi_{i<0}fiy = s0$$
$$\Pi_{i<sx}fiy = *fxy\Pi_{i<x}fiy$$

where $fxy$ is accessible and the following is provable

$$\forall x : \mathcal{K}.\forall y : \mathcal{L}.fxy : \mathcal{N}$$

for progressive predicates $\mathcal{K}$ and $\mathcal{L}$.

For the bounded sum we let $y : \mathcal{L}[\mathcal{A}]$ and take the conjunction $\mathcal{S}$ of the following predicates in $x$

- $x : \mathcal{K}[\mathcal{A}]$
- $fxy : \mathcal{A}$
- $\Sigma_{i<x}fiy : \mathcal{N}$

This is progressive and we prove that bounded sum is accessible

$$\forall y : \mathcal{L}[\mathcal{A}].\forall x : \mathcal{S}.\Sigma_{i<x}fiy : \mathcal{N}$$

For the bounded product we let $y : \mathcal{L}[\mathcal{M}]$ and take the conjunction $\mathcal{P}$ of the following predicates in $x$

- $x : \mathcal{K}[\mathcal{M}]$
- $fxy : \mathcal{M}$
- $\Pi_{i<x} fiy : \mathcal{A}$
- $\Pi_{i<px} fiy : \mathcal{N}$

This is progressive and we prove that the bounded product is accessible. Note that we needed both the last two items. We get that the bounded sum is in $\mathcal{N}$ from $\mathcal{A}$ by adding 0.

## 9.5 Control versus value

In the proof above we have used natural numbers in two ways

**Value:** As output — or conclusion — in the derivations we get some $n : \mathcal{N}$. This $N$ has only use as a value.

**Control:** As input — or assumption — we use some $k : \mathcal{K}$ for a progressive predicate$\mathcal{K}$. This means that we must declare the iteration we want to use $k$ in before the calculations begin.

Often these concepts go together. We often say that programs and data are of the same kind — and refer to Gödel or Turing. Here we split them apart and get some new analyses using $\Sigma_1^0$-proof theory.

# Chapter 10

# Quantifiers

## 10.1 Skolem and quantifiers

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

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

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 $\phi a$ .

Skolem worked both in number theory and logic. One of his first insights in logic comes from number theory. In Diophantine equations we can reduce the degree to 2 by adding new variables and equations. Say we have the equation

$$y = x^5$$

Then we can introduce new variables $u, v$ and equations

$$u = x^2$$
$$v = u^2$$
$$y = vx$$

And we can get down to a single equation in degree 4 by introducing the sum of squares

$$(u - x^2)^2 + (v - u^2)^2 + (y - vx)^2 = 0$$

Skolem used a similar trick to reduce the quantifier complexity by introducing new relation symbols. Say we have a sentence

$$\forall x.\exists y.\forall u.\exists v.Rxyuv$$

We can replace it by the conjunctions of universal quantifications using new relation symbols $S, T, U$

$$\begin{aligned} & (Sxyu \leftrightarrow \exists v.Rxyuv) \\ \wedge\ & (Txy \leftrightarrow \forall u.Sxyu) \\ \wedge\ & (Ux \leftrightarrow \exists y.Txy) \\ \wedge\ & \forall x.Ux \end{aligned}$$

In this way we can reduce sentences to $\forall\exists$-form. This simplified form was used by Skolem in giving procedures for falsifying sentences and getting the mathematical core of the completeness theorem for predicate calculus.

## 10.2 Gödels interpretation

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.

We go through Joseph Shoenfields variant of 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 as Shoenfield showed. 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

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

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 $\Gamma$ *can be interpreted in* $\mathcal{T}$ if there are terms $\phi$ and $\psi$ such that

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

Note that as arguments in both $\phi$ and $\psi$ 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.

## 10.3 The proof

**Definition 5** *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 19** *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 20 (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 21 (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 22 (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 23 (Disjunction)** $\vdash \Gamma, F \Rightarrow \vdash \Gamma, F \vee G$ *is interpretable.*

Similar to negation-conjunction.

**Theorem 24 (Negation-disjunction)** $\vdash \Gamma, \neg F \,\&\, \vdash \Gamma, \neg G \;\Rightarrow \vdash \Gamma, \neg(F \vee G)$ *is interpretable.*

Similar to conjunction.

**Theorem 25 (Double negation)** $\vdash \Gamma, F \Rightarrow \vdash \Gamma, \neg\neg F$ *is interpretable.*

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

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

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

**Theorem 26 (∀-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 27 (Negation-∀)** $\vdash \Gamma, \neg F\phi \Rightarrow \vdash \Gamma, \neg\forall x \cdot Fx$ *is interpretable.*

$$\begin{aligned} (Fa)^{\#} &= \forall b_0 \cdot \exists c \cdot R\, a\, b_0\, c \\ (\neg Fa)^{\#} &= \forall C \cdot \exists b_0 \cdot \neg R\, a\, b_0\, Cb_0 \\ (\forall x \cdot Fx)^{\#} &= \forall ab_1 \cdot \exists c \cdot R\, a\, b_1\, c \\ (\neg\forall x \cdot Fx)^{\#} &= \forall D \cdot \exists ab_1 \cdot \neg R\, a\, b_1\, Dab_1 \end{aligned}$$

We assume $\vdash_{\mathcal{T}} \Pi, \neg R\, \phi\, b_0^{\star}\, Cb_0^{\star}$. 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 28 (Cut)** $\vdash \Gamma, F$ & $\vdash \Gamma, \neg F \Rightarrow \Gamma$ *is interpretable.*

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

Assume $\vdash_{\mathcal{T}} \Pi, R\, a\, b^{\star}$ and $\vdash_{\mathcal{T}} \Pi, \neg R\, a^{\star}\, Ba^{\star}$ where $b^{\star}$ may contain $a$ and the free variables of $\Pi$ and $a^{\star}$ may contain $B$ and the free variables of $\Pi$. We define

$\phi = \lambda a \cdot b^\star$ and $\psi = \lambda B \cdot a^\star$. Both contain only free variables from $\Pi$. Substitute $\phi$ for $B$ and $\psi\phi$ for $a$. Then

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

**Theorem 29 (Induction)** $\vdash \Gamma, F0$ & $\vdash \Gamma, \neg Fa, Fsa \Rightarrow \vdash \Gamma, F\phi$ *is interpretable.*

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

Assume $\vdash_{\mathcal{T}} \Pi, R\,0\,b_0\,c_0^\star$ and $\vdash_{\mathcal{T}} \Pi, \neg R\,a\,b^\star\,Cb^\star, R\,sa\,b_1\,c_1^\star$. Define $\phi = \lambda b_0 \cdot c_0^\star$ , $\psi = \lambda aCb_1 \cdot b^\star$ and $\xi = \lambda aCb_1 \cdot c_1^\star$. Note that $a$ is of type natural numbers.

We use it as recursion variable in defining $\zeta$ by primitive recursion

$$\begin{array}{rcl} \zeta 0 & = & \phi \\ \zeta sa & = & \xi asa \end{array}$$

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

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

This concludes the proof of

**Theorem 30 (Dialectica interpretation)** *Every sequent derivable in Peano arithmetic can be interpreted in Gödels $\mathcal{T}$.*

## 10.4 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 31** *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 $\lambda$-calculus

- explicit definition
- definition by cases
- $\lambda$-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.

Why use Gödels complicated interpretation of the quantifiers? One could have tried to interprete $\forall a \cdot \exists \cdot F\, a\, b$ 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

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

Assume now that there is a term $\phi$ interpreting $c$. So

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

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 $\phi$ 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 $\vdash F \to F$. 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

$$\vdash F \to \exists y_1 \ldots y_n \cdot R f_1 \ldots f_m y_1 \ldots y_n$$

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

$$\vdash_{\mathcal{A}} \exists y_1 \ldots y_n \cdot R f_1 \ldots f_m y_1 \ldots y_n$$

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

$$\vdash_{\mathcal{T}} \cdot R f_1 \ldots f_m c_1^\star \ldots c_n^\star$$

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

$$\begin{array}{c} \exists xz \cdot R\, x\, f x\, z \\ \exists y \cdot S\, g\, y\, h y \end{array}$$

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

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

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 $h$.

## 10.5 Spectors interpretation

In 1960 Clifford Spector generalized Gödels interpretation to full second order arithmetic. Per Martin-Löf has given the following version.

**Theorem 32 (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

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

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

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

The $g^\star$ we want to find can then be seen as an infinite sequence of elements of type $\pi$. Let $o$ be a constant of type $\pi$. 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 $\theta$ 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

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

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

$$g^\star = \theta() = \theta(g_0) = \cdots = \theta(g_0, \ldots, g_a) = (g_0, \ldots, g_a)$$

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

$$C_0(\epsilon a C_0) = C_0 g_a = C_1\theta(g_0, \ldots, g_{a-1}) = C_1 g^\star$$

## 10.6 $\epsilon$-calculus

Another way to eliminate quantifiers in arithmetic is to use $\epsilon$-calculus. We have a language of arithmetic (which includes 0 and $<$) extended with $\epsilon$-terms. As new $\epsilon$-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 $\epsilon$-free term is equal to a numeral. As in predicate logic we have the $\epsilon$-theorems.

**Theorem 33 (First $\epsilon$-theorem for arithmetic)** *Any formula $A$ in arithmetic is equivalent to a formula $A^\epsilon$ with no quantifiers and using $\epsilon$-terms. $A$ is provable in Peano-arithmetic if and only if $A^\epsilon$ is provable in $\epsilon$-arithmetic.*

The only new thing is to prove that the induction rule is derivable from the $\epsilon$-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 34 (Second $\epsilon$-theorem in arithmetic)** *Let $F$ be a formula without quantifiers, free variables or $\epsilon$-terms. If $F$ is provable with the $\epsilon$-axioms, then it is provable without them.*

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

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

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

We assume that the $\epsilon$-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 $\epsilon$-terms

introduced further down in the list. The same with $Bx$ etc. There are of course no restriction on the $\epsilon$-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 $\epsilon$-terms, $(\epsilon x.A)(\bar{y}) \mapsto \alpha(\bar{y})$, $(\epsilon x.B)(\bar{y}) \mapsto \beta(\bar{y})$, ... making all $\epsilon$-axioms derivable. The substitution $\alpha$ 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 $\epsilon$-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 $\alpha$, we then solve $\beta$ 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 $\gamma$ etc until we have given substitutions making all the $\epsilon$-axioms derivable. We observe that $\alpha^\infty(\beta, \gamma)$ is continuous in $\beta$ and $\gamma$ and that $\beta^\infty(\gamma)$ is continuous in $\gamma$. This means that we can have the above as a simultaneous recursion of all the substitutions of the $\epsilon$-terms.

# Chapter 11

# Ordinals

## 11.1 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*

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 $\omega$ 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 $\omega+1 > \omega$

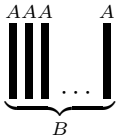
## 11.2 Arithmetical operations

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

$A + B$ : 

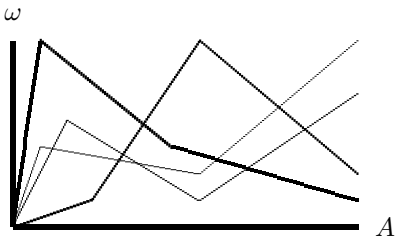
$A$ then $B$

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 $\omega^A$ 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 $\epsilon_0$ which is defined by

$$\begin{aligned} \alpha_1 &= \omega + 1 \\ \alpha_{n+1} &= 2^{\alpha_n} \\ \epsilon_0 &= \sup_n \alpha_n \end{aligned}$$

$\epsilon_0$ can be seen as the order type of finitely many times iterated lexicographical ordering.

## 11.3 Transfinite induction

We can obviously represent the ordinals — say below $\epsilon_0$ — 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 $\alpha$ by

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

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.

**Simple properties:**

- $\mathbf{TI}(\omega)$ — this is ordinary induction
- $\mathbf{TI}(\beta) \wedge \alpha < \beta \to \mathbf{TI}(\alpha)$ — we represent the ordinals in a primitive recursive way

**Addition.** $\mathbf{TI}(\alpha) \wedge \mathbf{TI}(\beta) \to \mathbf{TI}(\alpha + \beta)$. 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.** $\mathbf{TI}(\alpha) \wedge \mathbf{TI}(\beta) \to \mathbf{TI}(\alpha \times \beta)$.
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 $\mathbf{TI}(\alpha) \rightarrow \mathbf{TI}(2^\alpha)$
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 $\mathbf{PROG}(F)$, $\alpha$ and $\forall x < \alpha.F^*(x)$ be given. We want to prove $F^*(\alpha)$.
Let $\beta$ 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 three times 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.

**Note:** Observe that the proofs reflect the geometric structure we used in defining the arithmetical operations. The proof of addition can be broken into two pieces — first we prove induction up to $\alpha$, then the stretch from $\alpha$

to $\alpha + \beta$. In multiplication we have a nesting — an outer proof along $\alpha$ and then for each point inside a proof of $\beta$. Exponentiation must be explained by the more abstract property we also used in explaining what exponentiation of ordinals were.

**Exercise 11.3.1** $\boldsymbol{TI}(\alpha) \wedge \boldsymbol{TI}(\beta) \wedge \alpha > 0 \rightarrow \boldsymbol{TI}(\alpha^\beta)$

**Discussion.** We have proved that transfinite induction over ordinals are closed under the usual arithmetical operations. This gets us up to every ordinal less than $\epsilon_0$. For addition and multiplication we only need to have induction over some slightly more complicated predicates. The big step is with the exponentiation. Observe that we get transfinite induction over exponentiation by having extra quantifiers in the new predicates. Here there is a tradeoff between using larger ordinals and having more complicated induction formulas.

## 11.4 Ordinal bounds

We now want to do two different things

- in this section we show that the ordinal $\epsilon_0$ occur as a bound for heights when we eliminate cuts in infinitary calculus
- in the next section we show that $\epsilon_0$ 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 $\omega$-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 highest 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

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

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

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

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 exponential jump in the height. If the original derivation was of infinite height but less than $\epsilon_0$, then the cutfree derivation is of height $< \epsilon_0$.

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

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

We can replace it with an instance of the $\omega$-rule and many cuts

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

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.

## 11.5 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 $\alpha$

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

In this system $\mathbf{PROG}(F)$ is derivable. Furthermore the system with $\omega$-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 $\omega$-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 $\alpha$ 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 we can state Gentzens result.

**Theorem 35 (Gentzen)** $\vdash_{PA} \boldsymbol{TI}(\alpha) \Leftrightarrow \alpha < \epsilon_0$

The two sides combine different worlds

**Left:** Provability in first order arithmetic

**Right:** Ordertype of $\alpha$ is less than iterated lexicographical ordering

And the proof involves

$\Rightarrow$**:** The complexity of a direct proof of $\mathbf{TI}(\alpha)$

$\Leftarrow$**:** Provability of transfinite induction is closed under exponentiation

Gentzens result is an improvement over Gödels incompleteness. It analyzes provability in a system using the mathematical statement $\alpha < \epsilon_0$.

# Chapter 12

# Finite trees

## 12.1 Ordering trees

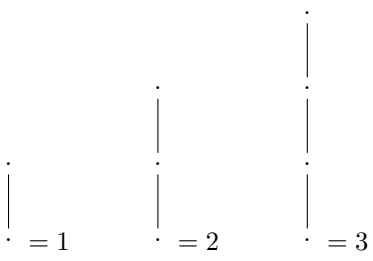
For the analysis of Gentzen we need to have a theory of ordinals with the ordinals less than $\epsilon_0$ as a natural part. We do this here with finite trees. The trees are finite with finite branchings. The ordinals less than $\epsilon_0$ turn out to be the trees with binary (and unary) branching.

Instead of working with ordinals as something given and having names for some of them, we work directly with finite trees and give them an ordering which we show in a constructive way to be a well ordering.

The smallest tree is the one with just a root

$= 0$

It corresponds to the ordinal 0. The natural numbers are the finite trees with unary branchings

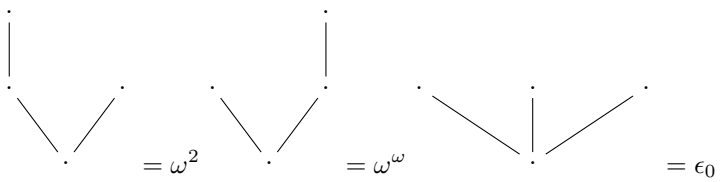


$= 1$ $= 2$ $= 3$

And so on through the natural numbers. The first infinite ordinal is the smallest tree that can be embedded in all finite trees except the unary ones. We have

$= \omega$

And some other infinite ordinals




Let us now define the ordering. We start with finite trees — the root is downmost and the branching is ordered from left to right.

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 define the ordering of trees by recursion over the immediate subtrees.

$$\boxed{\mathbf{A} < \mathbf{B} \;\Leftrightarrow\; \mathbf{A} \leq \langle \mathbf{B} \rangle \;\vee\; (\langle \mathbf{A} \rangle < \mathbf{B} \wedge \langle \mathbf{A} \rangle < \langle \mathbf{B} \rangle)}$$

All this is straightforward from a constructive point of view. It is a simple exercise to write a programs for deciding the ordering. Now to some simple properties proved by induction over heights of the trees

**Transitivity** $S < T$ and $T < U$ gives $S < U$

**Irreflexivity** For no $S$: $S < S$

**Totality** $S < T$ or $S = T$ or $T < S$ and these cases are mutually exclusive

**Equality** $S = T$ if they are equal as trees

We go through the proof that the ordering is transitive. So assume we have $S < T$ and $T < U$. We then prove by induction over the heights that $S < U$ and consider the following cases:

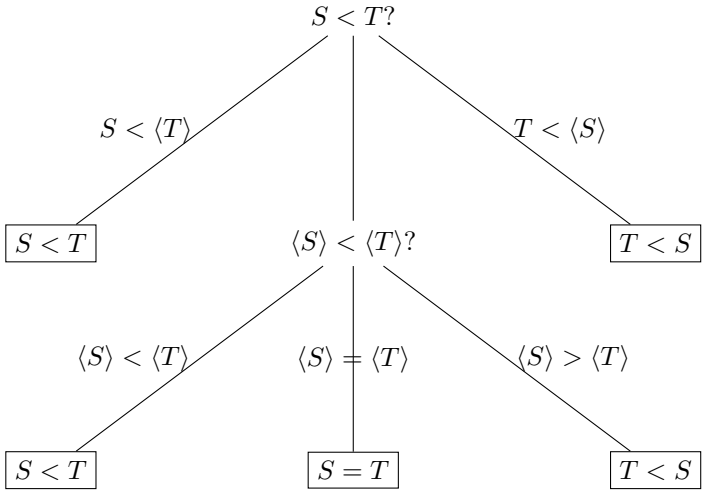
**$\mathbf{T} \leq \langle \mathbf{U} \rangle$:** By induction $S \leq \langle U \rangle$ and $S < U$

**$\langle \mathbf{T} \rangle < \mathbf{U}$ and $\langle \mathbf{T} \rangle < \langle \mathbf{U} \rangle$:** We then consider the cases of $S < T$

- **$\mathbf{S} \leq \langle \mathbf{T} \rangle$:** Using $S \leq \langle T \rangle < U$ we get by induction $S < U$
- **$\langle \mathbf{S} \rangle < \mathbf{T}$ and $\langle \mathbf{S} \rangle < \langle \mathbf{T} \rangle$:** By induction we first get $\langle S \rangle < U$. Then from $\langle S \rangle < \langle T \rangle < \langle U \rangle$ we get $\langle S \rangle < \langle U \rangle$ and can conclude that $S < U$.

## 12.2 Deciding the ordering

There is more work to calculate the ordinals of the trees above. We distinguish between properties which are given by ordinary induction over the height of the trees and those properties where we need more complicated methods of proofs. For the proofs it may be worthwhile to look at a decision tree for the ordering:




We start at the top and want to decide the ordering between $S$ and $T$. First we look at whether

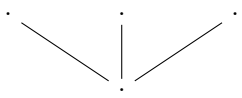
- $S < \langle T \rangle$
- $T < \langle S \rangle$

In the first case we go to the left and conclude $S < T$. In the second case we go to the right and conclude $T < S$. If neither we proceed down the middle and look at the lexicographical ordering between $\langle S \rangle$ and $\langle T \rangle$ and from the three possible outcomes we get $S < T$, $S = T$ or $T < S$. So to decide the ordering between $S$ and $T$ we only need to look at ordering between $S$ or $T$ and subtrees of them. This shows immediately that the ordering is decidable. Note that we get $S = T$ when the two trees are equal. It is a pleasant property that the equality in the ordering is simply ordinary equality of trees. Each ordinal which corresponds to a tree, corresponds to a unique tree. Below we shall often write down an ordinal instead of the corresponding tree.

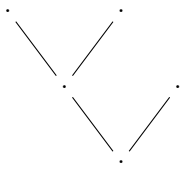
## 12.3 Embeddings

We have an ordinary embedding of tree $S$ into tree $T$ whenever we have an injection of the nodes of $S$ into the nodes of $T$ respecting the treeordering. If we have an ordinary embedding we also get an injection of paths in $S$ into paths in

$T$. Now we are here not interested in the general notion of embeddings, but in the socalled topological embedding. Then we require in addition of the injection of paths that the paths should not meet at interior nodes. An example will make it clearer. The tree



can be embedded into



We see that two of the paths will have an interior node in common. In fact the first tree cannot be topologically embedded into the second tree. Let us write $S \preceq T$ for $S$ can be topologically embedded into $T$. We then have the following cases

- $S = T$
- $S \preceq \langle T \rangle$
- $S$ is the empty tree
- Write $\langle S_i \rangle_{i \in I}$ and $\langle T_j \rangle_{j \in J}$ for the immediate subtrees of the $S$ and $T$. There is then an increasing function $f$ from $I$ into $J$ with $S_i \preceq T_{fi}$

We then have by induction over the build up of trees

**Theorem 36 (Embedding)** *If $S$ can be topologically embedded in $T$, then $S \leq T$.*

The first tree above corresponds to ordinal $\epsilon_0$ and the second one to $\omega^2$ as we shall see. This shows that the first tree cannot be topologically embedded in the second.

The ordering goes beyond topological embedding. The ordering is a linear order but the topological embedding is just a partial order. In particular the ordering has a left to right bias, while the topological embedding has no such bias.

The trees are nicely stratified. The smallest tree is the empty tree. Then comes the ones with unary branching. If a tree has one binary branch, then we can topologically embed

into it. So the tree is the smallest tree among the trees with at least one binary branch. Similar results for trees with one ternary branch, and so on.

The following is also useful

**Theorem 37 (Monotonicity:)** *If we have $A < B$ and tree $C$ where we substitute in $A$ and $B$ at the same place, then $C[A] < C[B]$.*

## 12.4 Some calculations

So far we only know that each finite tree has an ordertype. In fact it turns out that the ordertypes are ordinals. We show this first by actually calculating the ordertypes. Later we shall show in a more direct way that the ordering is a well ordering.

The empty tree is the smallest tree and corresponds to the ordinal 0.

It is already some effort to show that we can define the successor function in a simple way. We have:

$$\overset{\alpha}{\overset{|}{\cdot}} = \alpha + 1$$

We do this by showing:

$$A < B \Leftrightarrow \overset{A}{\overset{|}{\cdot}} \leq B$$

or equivalently

$$B < \overset{A}{\overset{|}{\cdot}} \Leftrightarrow B \leq A$$

by induction over the heights. It is obviously true if one of the trees are empty. The one way of the equivalences follows from:

$$B \leq A < \overset{A}{\overset{|}{\cdot}}$$

So assume

$$B < \overset{A}{\overset{|}{\cdot}}$$

Then either $B$ is empty — and we get $B \leq A$ and are done — or we get if the branching in $B$ is unary:

$$B = \begin{array}{c} B_0 \\ \vdots \end{array}$$

But then by monotonicity:

$$B_0 < A$$

By induction:

$$B = \begin{array}{c} B_0 \\ \vdots \end{array} \leq A$$

If the branching in $B$ is more than unary, then we must use the first clause in the definition of the ordering and we get immediately:

$$B \leq A$$

To get order types for other trees we show how we can build up the order types from below. Note that this is done by induction over the height of the trees.

## 12.5 Building up from below

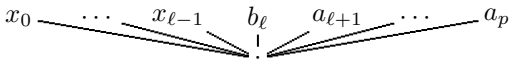
Given a tree $\mathbf{A}$ with immediate subtrees


*The immediate subtrees $a_i$ of $\mathbf{A}$ 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


This is interesting. Given a tree **A** we find both smaller trees and operations on trees which are closed under **A**

**Fundamental subtrees of A :** The immediate subtrees of **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.

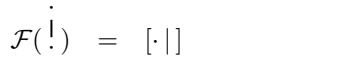
We denote the fundamental set of **A** with $\mathcal{F}(\mathbf{A})$ and we shall write it as

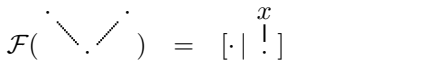
$$[S, \ldots, T|F, \ldots, G]$$

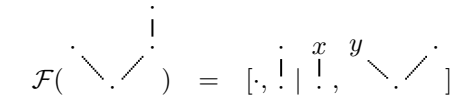
where we have displayed the fundamental subtrees $S, \ldots, T$ and the fundamental functions $F, \ldots, G$. Similarly for the elementary fundamental set $\mathcal{H}(\mathbf{A})$

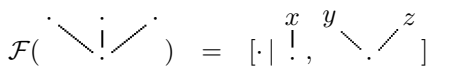
We have the following:

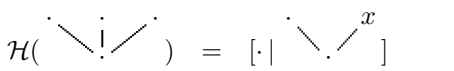
$$\mathcal{F}(\,.\,) = \emptyset$$



Here $x, y, z$ are variables used for describing fundamental functions.

Now we note that the fundamental sets give an approximation of trees from below. In fact we have for any tree $\mathbf{A}$:

$$\mathbf{B} < \mathbf{A} \Leftrightarrow \exists \mathbf{C} \in \mathcal{F}(\mathbf{A}).\mathbf{C} \geq \mathbf{B}$$

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 $\mathbf{A}$.

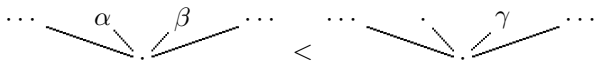
$\langle \mathbf{B} \rangle < \mathbf{A} \wedge \langle \mathbf{B} \rangle < \langle \mathbf{A} \rangle$ **:** By induction — to each immediate subtree $\mathbf{B}_i$ there is an $\mathbf{C}_i \in \mathcal{F}(\mathbf{A})$ with $\mathbf{C}_i \geq \mathbf{B}_i$. Depending on how we prove $\langle \mathbf{B} \rangle < \langle \mathbf{A} \rangle$ 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 it is proved. We can also use the elementary fundamental set

For any tree $\mathbf{A}$:

$$\mathbf{B} < \mathbf{A} \Leftrightarrow \exists \mathbf{C} \in \mathcal{H}(\mathbf{A}).\mathbf{C} \geq \mathbf{B}$$

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.

With some extra assumption we can conclude that the fundamental set is not only an approximation from below, but that it contains all smaller trees. The extra assumption is that all trees less than or equal to the fundamental subtrees of $\mathbf{A}$ is contained in $\mathcal{F}(\mathbf{A})$. We then have

$$\mathbf{B} < \mathbf{A} \Leftrightarrow \mathbf{B} \in \mathcal{F}(\mathbf{A})$$

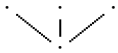
The proof follows the lines above. We have induction over the height of $\mathbf{B}$ and get to the cases

$\mathbf{B} \leq \langle \mathbf{A} \rangle$ **:** 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 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.

## 12.6 Normal functions and ordinal notations

The geometrical way of characterizing ordinals gives us the ordinals up to $\epsilon_0$ — but not much more. Oswald Veblen showed in 1908 how to get much further. So let us see how this can be done.

A unary function $F$ of ordinals into ordinals is *normal* if both

**Strictly monotone increasing:** $\alpha < \beta \Rightarrow F\alpha < F\beta$

**Continuous:** For an increasing sequence $\alpha_i$ we have $F \sup \alpha_i = \sup F\alpha_i$

The main property about normal functions is that they have fix points. Given an ordinal $\alpha$ the first fix points above $\alpha$ is given as the limit of

$$\alpha, F\alpha, FF\alpha, FFF\alpha, \ldots$$

If we have a countable family $\mathcal{F}$ of normal functions, we can find a common fix point by applying the functions from $\mathcal{F}$ in a fair way. The fix points is a closed unbounded set of ordinals. Conversely if we have a closed and unbounded set of ordinals its enumerating function is normal. A normal function $F$ is increasing — $x \leq Fx$. (For assume not and let $\alpha$ be the smallest ordinal with $F\alpha < \alpha$. But then $FF\alpha < F\alpha < \alpha$ and we have found an even smaller ordinal.)

Since Veblens paper from 1908 we know how to build a hierarchy of normal functions. There are two operations

**Unary to binary:** Given a normal function $Fx$. The binary function $Fxy$ of ordinals which is normal in $x$ and strictly monotone increasing in $y$ is given by

- $Fx0 = Fx$

- $Fxy =$ the $x$'th common fix point of $\lambda u.Fuz$ for all $z < y$

**Diagonalization:** Given a binary function $Fxy$ as constructed from a normal function $F$ as above. We define a unary function $Gy = F0y$. This unary function is seen to be normal.

We now start with a unary normal function $F$ and build a hierarchy of $n$-ary functions above it. We describe how it is done for quaternary functions and leave the general construction to the reader.

- $Fx0 = Fx$
- $Fxy0 = Fxy$
- $Fxyz0 = Fxyz$

$y > 0:$ $Fxyzu =$ the $x$'th common fix point of $\lambda v.Fvy'zu$ for all $y' < y$

$z > 0:$ $F0yzu =$ the $y$'th common fix point of $\lambda v.F0vz'u$ for all $z' < z$

$u > 0:$ $F00zu =$ the $z$'th common fix point of $\lambda v.F00vu'$ for all $u' < u$

There are no binary function of ordinal which is normal in one argument and strictly monotone increasing in the other. For assume $F\alpha\beta$ is such a binary function which is normal in the first argument. Let $\beta_1$ and $\beta_2$ be given. Then the functions $\lambda x.Fx\beta_1$ and $\lambda x.Fx\beta_2$ have a common fix point $Y$ and hence $FY\beta_1 = Y = FY\beta_2$ and since $F$ is increasing in the second argument we get $\beta_1 = \beta_2$ and the function is not strictly monotone increasing in the second argument.

In this way we get a system of notations for ordinals. It is common to start with the normal function $\lambda x.\omega^x$. We write this as $\phi x$. Then we have

- $\epsilon_0$ is the first fix point. $\epsilon_0 = \phi 01$
- $\Gamma_0$ is the first fix point of the diagonalization of the binary functions. $\Gamma_0 = \phi 001$
- The small Veblen ordinal is the supremum of the ordinals described by this hierarchy

Observe that we define the $n$-ary functions by just looking at a couple of the arguments. We can extend the hierarchy to functions with an infinite number of arguments — the arguments ordered by ordinals and only a finite number of them different from 0. If we do this we get ordinal notations up to the socalled large Veblen ordinal and a notation system for it is given by Kurt Schüttes Klammersymbole.

Later when we come to the order type of finite trees we need to consider a variant of the normal functions. Let $F$ be a normal function. We then have defined

- The range of $F$: $R_F$
- The fix points of $F$: $\mathrm{FIX}_F$

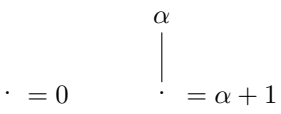
We let $F^*$ be the function enumerating $R_F - \mathrm{FIX}_F$. So $F*$ is like $F$ except that it jumps over fix points. We write

$$F^* \sim F$$

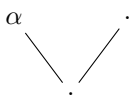
where $F$ is a normal function and $F^*$ is the same as $F$ except that it jumps over fix points.

## 12.7 Further calculations

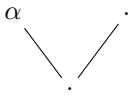
We have shown that the empty tree has order type 0 and that




We now try to characterize the function



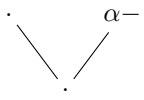

But this is the supremum of the set of $\alpha$ closed under the successor function. The function $\omega \cdot x$ enumerates the limit points. Now we observe that we therefore have after calibrating the start and observing that we jump over fix points



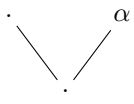

$$\sim (1 + \alpha) \cdot \omega$$

Now to the function


For $\alpha = 0$ it is $\omega$. For $\alpha > 0$ it is the supremum of the ordinals starting with $\alpha$ and closed under all

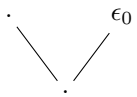



where $\alpha-$ runs over ordinals $< \alpha$. This gives for $\alpha < \epsilon_0$



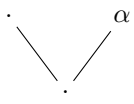

$= \omega^{\omega^\alpha}$

At $\epsilon_0$ we have a fix point of the function $\omega^{\omega^\alpha}$. In the treefunction we jump over the fix point and get



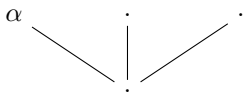

$= \omega^{\omega^{\epsilon_0+1}}$

For the full fuction we write



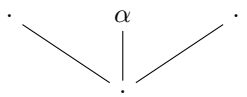

$\sim \omega^{\omega^\alpha}$

where the $\sim$ indicates that we jump over the fix points. We can then go on



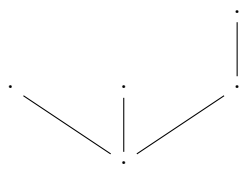

$\sim \epsilon_\alpha$

where $\epsilon_\alpha$ is the enumerating function of the $\epsilon$-numbers. The critical $\epsilon$-numbers are the numbers $\kappa$ with $\epsilon_\kappa = \kappa$. Let $\kappa_\alpha$ be the function enumerating them. Then




$\sim \kappa_\alpha$

The ordinal $\Gamma_0$ is the first fix point of the enumeration $\kappa_\alpha$ of the critical $\epsilon$-numbers. We have:



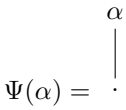
$= \Gamma_0$

Too sum up

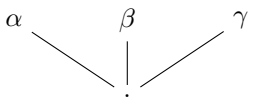
- We prove by direct calculation that the order types of the finite trees are in fact ordinals. Later we shall give other proofs of it.
- The tree functions correspond in a direct way to the Veblen hierarchy of functions where we jump over the fix points.

- The limit of the order type of the finite trees is the small Veblen ordinal.

Now let us introduce some new notation instead of the trees

$\Psi(\alpha) =$ 


$\Psi(\alpha, \beta) =$ 


$\Psi(\alpha, \beta, \gamma) =$ 


And so on. Let us write

$$\Psi_k(\alpha, \beta, \ldots) = \Psi(0, \ldots, 0, \alpha, \beta, \ldots)$$

where we have 0's in the first $k$ arguments. And we write

$$\Psi_k^n(\alpha, \beta, \ldots),$$

for $n$ times iteration of $\lambda x.\Psi_k(x, \beta, \ldots)$ starting with $\alpha$. We use this in describing the effect of elementary fundamental sets. With this notation we can sum up what can be proved here in Skolem arithmetic

- $\alpha < \Psi(\beta) \leftrightarrow \alpha \leq \beta$
- $\alpha < \Psi(\beta, 0) \leftrightarrow \exists n.\alpha \leq \Psi_0^n(\beta)$
- For $\beta > 0$: $\alpha < \Psi(0, \beta) \leftrightarrow \exists n.\alpha \leq \Psi_0^n(\beta, \beta-)$ for some $\beta- < \beta$
- $\alpha < \Psi(0, 0, 0) \leftrightarrow \exists n.\alpha \leq \Psi_1^n(0)$
- For $\beta > 0$: $\alpha < \Psi(\beta, 0, 0) \leftrightarrow \exists n.\alpha \leq \Psi_1^n(\Psi(\beta-, 0, 0))$ for some $\beta- < \beta$
- For $\beta > 0$: $\alpha < \Psi(0, \beta, 0) \leftrightarrow \exists n.\alpha \leq \Psi_0^n(\beta, \beta-, 0)$ for some $\beta- < \beta$
- For $\beta > 0$: $\alpha < \Psi(0, 0, \beta) \leftrightarrow \exists n.\alpha \leq \Psi_1^n(\beta, \beta-)$ for some $\beta- < \beta$
- $\alpha < \Psi(0, 0, 0, 0) \leftrightarrow \exists n.\alpha \leq \Psi_2^n(0)$
- And so on

## 12.8 Proof of wellordering

So far we have only used ordinary quantifier free induction over the heights of trees to prove properties, but more is needed to prove that the tree ordering is well founded. We give three proofs.

**Using Kruskals theorem:**

Kruskals theorem says that if we have an infinite sequence of finite trees $T_0, T_1, \ldots$, then there are numbers $m < n$ such that $T_m$ is topologically embedded in $T_n$. But then $T_m < T_n$ and no infinite sequence of finite trees can be infinitely descending.

**Using inductive definitions:**

We prove that if $\mathbf{T}_1, \ldots, \mathbf{T}_n$ are well founded then so is also all $\mathbf{A}$ with

$$\mathbf{A} < \mathbf{T} = \begin{array}{ccccc} \mathbf{T}_1 & & \cdots & & \mathbf{T}_n \\ & \diagdown & | & \diagup & \\ & & \cdot & & \end{array}$$

and hence $\mathbf{T}$ itself is well founded. This is done by induction over

- the height of $\mathbf{A}$
- the sequence $\mathbf{T}_1, \ldots, \mathbf{T}_n$ ordered by the inverse lexicographical ordering of tree ordering

So assume we have $\mathbf{T}_1, \ldots, \mathbf{T}_n$ well founded and $\mathbf{A} < \mathbf{T}$. We then have the following cases

$\mathbf{A} \leq \langle \mathbf{T} \rangle$ **:** Then $\mathbf{A} \leq \mathbf{T}_i$ for some $i$ and is therefore well founded.

$\langle \mathbf{A} \rangle < \mathbf{T} \wedge \langle \mathbf{A} \rangle < \langle \mathbf{T} \rangle$ **:** Then all immediate subtrees of $\mathbf{A}$ are less than $\mathbf{T}$ and by induction over height of $\mathbf{A}$ the immediate subtrees of $\mathbf{A}$ are well founded. But the immediate subtrees of $\mathbf{A}$ comes before the immediate subtrees of $\mathbf{T}$ in the inverse lexicographical ordering. We conclude that $\mathbf{A}$ is well founded.

We then get that all finite trees are well founded by some giant — and perhaps not so informative — step.

**Using minimal bad sequence:**

A bad sequence is an infinitely descending sequence $A_0, A_1, A_2, A_3, \ldots$. It is minimal bad if

- $A_0$ is of minimal height starting an infinite descending sequence with $A_0$

- $A_1$ is of minimal height starting an infinite descending sequence with $A_0, A_1$
- $A_2$ is of minimal height starting an infinite descending sequence with $A_0, A_1, A_2$
- and so on

Given a bad sequence we can find a minimal bad sequence. In such a minimal bad sequence none of the immediate subtrees of the elements are bad. Now assume we have a bad sequence and hence a minimal bad sequence $A_0, A_1, A_2, \ldots$. Then by minimality we can only use the second condition in the definition of the ordering

$$A_i > A_{i+1} \Leftrightarrow \langle A_i \rangle \geq A_{i+1} \vee (A_i > \langle A_{i+1} \rangle \wedge \langle A_i \rangle > \langle A_{i+1} \rangle)$$

Therefore we have a descendinng sequence in the lexicographical ordering

$$\langle A_0 \rangle > \langle A_1 \rangle > \langle A_2 \rangle > \langle A_3 \rangle > \cdots$$

Then after a while these sequences must have the same length and we can find an element in one of them which is bad. So we have an immediate subtree of one of the elements which is bad — contradicting the minimality of the original sequence. Therefore there are no bad sequences and the ordering is a well ordering.

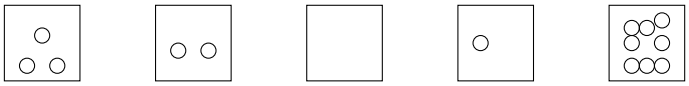
# Chapter 13

# $\Pi^1_1$-proof theory

## 13.1 Delayed decisions

Let us give one of the simplest examples where $\Pi^1_1$-assumptions enter. Consider the following one-person game. We have given

- some boxes — say five boxes
- the boxes are ordered from right to left
- in each box there is a finite number of balls
- a move in the game consists in taking away one ball from a box — and put a finite number of balls in some of the boxes to the right
- the game terminates if there are no balls left in any of the boxes



Obviously this game will always terminate. A way to prove this is to assign each configuration of boxes and balls an ordinal number less than $\omega^5$ — if there are 3, 2, 0, 1, 8 balls in the 5 boxes we assign the ordinal

$$\omega^4 \cdot 3 + \omega^3 \cdot 2 + \omega \cdot 1 + 8$$

and each move will lower the assigned ordinal. The indeterminacy of the moves does not matter. This gives a way to treat delayed decisions.

Many of our proof theoretic analyses are like this box-and-ball game. In particular we note the treatment of delayed decisions. When we start we do not know all the moves. We have not decided how many balls to put in the boxes after having one ball removed. This is not something we can calculate — we simply do not know. In the ball-and-box game there is no way we can calculate the number of moves the game will take. In applications to computations

we often distinguish between compile-time and run-time properties. Here the distinction is very clear.

We could analyze primitive recursive computations with a box-and-ball game with a finite number of boxes. Here we have a computation of for-loops nested within for-loops. The critical thing is that we first know when we enter a for-loop how many times we shall enter into it. We get a distinction between knowledge compile-time and knowledge run-time. The knowledge run-time is analyzed as if they were delayed decisions in a box-and-ball game.

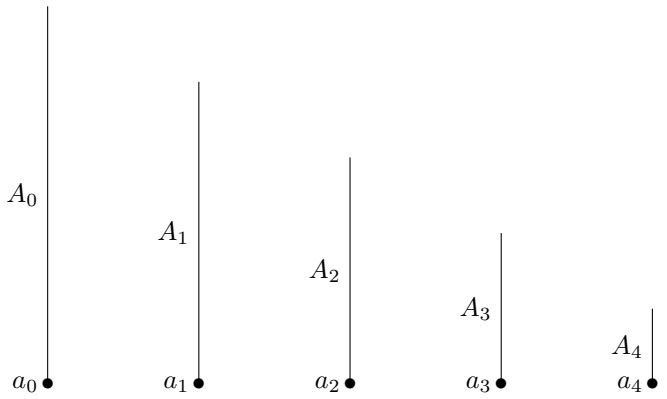
## 13.2 Ramseys theorem

Ramseys theorem can be seen as a generalization of the pigeon hole principle and does not properly belong to the $\Pi^1_1$-proof theory. But we treat it here as a preparation for Higmans lemma and Kruskals theorem where we use $\Pi^1_1$-assumptions in their analysis.

**Theorem 38 (Ramsey)** *We start with an infinite set $A$, a finite set of colors $C$ and a coloring of all pairs of elements i.e. a function $\phi : A^{(2)} \to C$. Then there is an infinite subset $B \subset A$ such that all pairs from $B$ are colored with the same color.*

The pairs are unordered with distinct elements. We do not count $(x, y)$ as different from $(y, x)$ and we have $x \neq y$.

We give Skolems proof of the Ramsey theorem. The following picture gives the idea




We do the construction from left to right as follows

- We start with the infinite set $A_0 = A$
- Pick an element $a_0$ from $A_0$
- We consider the colors of the pairs with $a_0$ and another element from $A_0$

- One of the colors, $c_0$, is repeated infinitely many times
- Let $A_1$ be an infinite subset of $A_0$ with elements $b$ such that $(a_0, b)$ are colored with $c_0$
- Repeat the above with the infinite set $A_1$ to get $a_1$, $c_1$ and $A_2$
- Continue
- For each $a_i$ all elements $b$ to the right are such that $(a_i, b)$ are colored with $c_i$
- One of the colors $c_j$ is repeated infinitely often
- The infinite set $a_{j_0}, a_{j_1}, a_{j_2}, \ldots$ is the sought subset $B$ of $A$

The same idea can be used to give other versions of Ramsey theorem

**Higher order tuples:** Given the Ramsey theorem for pairs we can use the same argument to give the theorem for triples. We then let $c_0$ and $A_1$ be such that all for all pairs $b, c$ from $A_1$ the triples $(a_0, b, c)$ is given color $c_0$. The infinite subset $A_1$ of $A_0$ is given by the Ramsey theorem for pairs. Then follow the rest of the argument above. Similarly for $n$-tuples.

**Finite version:** Given the wanted size of the monochromatic set and the number of colors, we can work backwards to find how large the original set could be to guarantee the size of a monochromatic set no matter how the coloring is.

## 13.3 Kruskals theorem

We consider the following ordering relations

**Quasi order:** Transitive and reflexive

**Partial order:** Quasi order which is also antisymmetric

**Linear order:** Partial order which is also total

And on these we have wellfoundedness properties

**Well quasi order (WQO):** Quasi order such that to every infinite sequence $a_0, a_1, a_2, \ldots$ there are $i < j$ with $a_i \leq a_j$

**Well partial order (WPO):** Well quasi order which is a partial order

**Well order (WO):** Well quasi order which is a linear order

The following is equivalent to being a well quasi order

- To every infinite sequence there is an infinite ascending subsequence $a_{i_0} \leq a_{i_1} \leq a_{i_2} \leq \cdots$
- There are no infinite strictly descending sequence and no infinite set of pairwise incomparable elements

Let us prove the first equivalence. So assume we have a well quasi order and an infinite sequence $a_0, a_1, a_2, \ldots$. We then give a coloring of the pairs $(a_i, a_j)$ with $i < j$ by mapping it to $\leq, >, |$ depending on the order relation between $a_i$ and $a_j$. By Ramsey theorem we have an infinite monochromatic subset. This must be the sought infinite ascending subsequence. The other way is trivial.

We have the following operations on orderings

**Products:** The product $A \times B$ is given by taking the ordering of each coordinate separately

**Strings:** The set of finite strings over $A$, $A*$, is ordered by letting $(a_0, \ldots, a_{m-1}) \leq (b_0, \ldots, b_{n-1})$ if there is a monotone strictly increasing $f : m \to n$ with $a_{fi} \leq b_i$ for all $i < m$

We then have

**Theorem 39** *The product of two WQO's is again WQO*

We call a pair of elements $(a, b)$ in an ordering for good if $a \leq b$. A sequence $a_0, a_1, \ldots$ of elements is good if it contains a good pair ($a_i \leq a_j$ for $i < j$). Else it is bad. So we have a WQO if every infinite sequence is good. Above we proved that if we have an infinite sequence in a WQO, we can find an infinite subsequence where all pairs are good.

Now assume $A$ and $B$ are WQO. Let $(a_0, b_0), (a_1, b_1), \ldots$ be an infinite sequence. Then the infinite sequence $a_0, a_1, \ldots$ have an infinite subsequence $a_{i_0}, a_{i_1}, a_{i_2}, \ldots$ where all the pairs are good. Then the infinite sequence in $B$, $b_{i_0}, b_{i_1}, b_{i_2}, \ldots$ have a good pair $b_{i_m}, b_{i_n}$ and the pair $(a_{i_m}, b_{i_m}), (a_{i_n}, b_{i_n})$ is good in $A \times B$ and $A \times B$ is a WQO.

**Theorem 40 (Higmans lemma)** *The finite strings over a WQO is again a WQO*

Let $A$ be a WQO and $A*$ the quasi ordering of finite strings. Assume we have an infinite bad sequence. We construct an infinite minimal bad sequence $\alpha_0, \alpha_1, \alpha_2, \ldots$ — minimal as follows

- $\alpha_0$ is of minimal length such that we have an infinite bad sequence starting with $\alpha_0$
- $\alpha_1$ is of minimal length such that we have an infinite bad sequence starting with $\alpha_0, \alpha_1$
- $\alpha_2$ is of minimal length such that we have an infinite bad sequence starting with $\alpha_0, \alpha_1, \alpha_2$

- and so on

We first observe that none of the elements in the sequence are empty. So we can write the sequence as

$$(a_0, \beta_0), (a_1, \beta_0), (a_2, \beta_2), \ldots$$

Let $B*$ be the set of $\beta_0, \beta_1, \beta_2, \ldots$. It is a subset of $A*$ and hence a quasi ordering. We show that it is a WQO. Assume we have an infinite sequence starting with say $\beta_i$. If this sequence involves an infinite number of times the $\beta_k$ with $k < i$, then we immediately get a good pair. Else we can assume that the sequence only involves $\beta_k$ with $k \geq i$. Consider now the sequence

$$\alpha_0, \alpha_1, \ldots, \alpha_{i-1}, \beta_i, \beta'_{i+1}, \beta'_{i+2}, \ldots$$

By minimality of $\alpha_i$ there must be a good pair. This good pair cannot involve any of the $\alpha$'s — neither as $\alpha_k, \alpha_l$ nor as $\alpha_k, \beta_m$. In the last case we would get $\alpha_k \leq \beta_m \leq \alpha_m$. So we must have a good pair among the $\beta$'s. Therefore $B*$ is a WQO. Furthermore $A \times B*$ is a WQO and we have a contradiction.

**Theorem 41 (Kruskal)** *The finite trees ordered by topological embedding is a WQO*

Let $\mathcal{T}$ be the set of finite trees ordered by topological embedding. Each such tree $T$ can be considered as a sequence of its immediate subtrees. Assume we have a bad sequence of finite trees. As above we construct a minimal bad sequence, $T_0, T_1, \ldots$ of trees where the minimality is by the height of the trees. Let the set of all immediate subtrees in the sequence be $\mathcal{F}$. We show that $\mathcal{F}$ is a WQO. Assume we have an infinite sequence $S_k, S_{k+1}, S_{k+2}, \ldots$ from $\mathcal{F}$ where $S_k$ is an immediate subtree of $T_k$. If there is an infinite number of elements from the sequence involving immediate subtrees of $T_i$ where $i < k$, then we immediately get a good pair. Else we can assume that the infinite sequence only involves immediate subtrees of the trees $T_i$ where $i \geq k$. Consider now the sequence

$$T_0, T_1, \ldots, T_{k-1}, S_k, S_{k+1}, \ldots$$

By minimality of $T_k$ it must have a good pair. This good pair cannot involve the $T$'s — neither as $T_i, T_j$ nor as $T_i, S_j$. So it must have a good pair among the $S$'s and we get that $\mathcal{F}$ is a WQO. By Higmans lemma we get that the sequences $\mathcal{F}*$ is also a WQO and hence so is $\mathcal{T}$.

The statement of Kruskals theorem involves topological embeddings. It is also true for embeddings, but the topological embedding formulation is stronger and it is reflected in the proof. We build our embeddings by using embeddings of the immediate subtrees and then glueing them together at the root.

## 13.4 Linearization

A quasi order $\leq$ can always be extended to a linear order. Look at the way to do it by using transitive closures. First assume that we have an order $\leq$ and an incomparable pair of elements $a|b$. We extend $\leq$ to $\leq^*$ where $a \leq^* b$ by using the transitive closure. Looking closer at the chains in the transitive closure and using the transitivity of $\leq$, we see that we have two possibilities for elements $c \leq^* d$

- $c \leq d$
- $c \leq a$ and $b \leq d$

We cannot have more than one occurrence of $a$ and $b$ in the chain going from $c$ to $d$. Else we would have a $b \leq a$ contradicting that they were incomparable. We also get that if $\leq$ is antisymmetric, then so is $\leq^*$. For in the chain going from $c$ to $d$ and then to $c$ we have at most one $a, b$ pair. Assume there is one. Then we get $b \leq c \leq a$ contradicting the incomparability of $a, b$.

These extensions behave nicely with respect to suborderings. Assume we have partial orderings $\leq \leq_1$ and $\leq_2$ where

- $\leq_1$ extends $\leq$ with domains $A_1 \supseteq A$
- $\leq_2$ extends $\leq$ with the same domain $A$

Then the transitive closure of $\leq_1$ and $\leq_2$ is a partial order. We must prove that it is antisymmetric. Consider the strict inequalities, $<_1$ and $<_2$, and look at the transitive chains from an element to itself of minimal length. Using the transitivity of the orderings we can assume that the chain alternates between $<_1$ and $<_2$. This means that all elements in the chain except possibly the last or the first must be from the smaller domain $A$ and they can all be considered as chained by $<_2$. So we have the following possibilities

- $a <_2 a$ — which contradicts the antisymmetry of $<_2$
- $a <_1 b <_2 c <_1 a$ — but then also $c <_1 a <_1 b$ and $c <_1 b$ — contradiction

We conclude

- Every partial order can be extended to a linear order
- Assume a subordering of the partial order can be extended to a linear order $<$, then the whole partial order can be extended to a linear order extending $<$
- A well partial order is a partial order where every linear extension is a well order

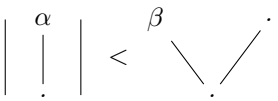
For a well partial order $\prec$ we define the maximal order type $|\prec|$ as the supremum of the order types of its linear extensions. We now consider linearizations of finite trees ordered by the topological embedding relation. So we consider orderings $\prec$ of finite trees with the following properties

**Linearity:** $\prec$ is a linear order

**Topological embedding:** For trees $\alpha$, $\beta$ where the branching at the root is larger in $\alpha$ than in $\beta$ then $\alpha \prec \beta \Leftrightarrow \alpha \preceq \langle\beta\rangle$ — that is $\alpha$ is $\prec$ or equal to one of the immediate subtrees of $\beta$.

We are now ready to give upper bounds for the orderings of finite trees. First the simple case where we have unary branching

**Theorem 42** *If $|\alpha| < \beta$, then*




Some explanation is in order. Here $|\alpha| < \beta$ is a relation between finite trees $\alpha$ and $\beta$. $|\alpha|$ is the maximal order type of a linearization of trees with topological embedding going up to $\alpha$. We compare this maximal order type with the finite tree $\beta$ in the ordering of them given in the last chapter. So implicitly we have assumed that the finite tree $\alpha$ have maximal order type an ordinal less than the tree $\beta$. The tree in the conclusion has $\beta$ on the top of the leftmost branch and 0 on the top of the other. The importance of that ordinal is that it is additively closed. For such an ordinal $\gamma$ we have

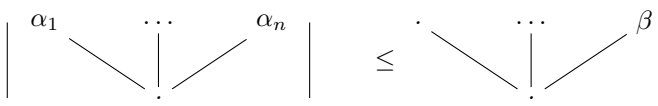
**Lemma 16** *Assume we have an ordering $L$ with $L = M \cup N$ and $M, N < \gamma$, then $L < \gamma$*

We are now ready to prove the theorem. Assume we have a sequence **A** ordered by $\prec$ and ending up with 

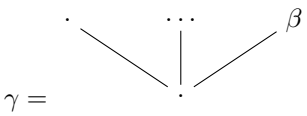

We then write **A** as the union of the empty tree and of the sequence **B** where all elements are of branching 1 at root, and the sequence **C** where all elements are of branching at root $> 1$. Observe that because of the topological embedding all elements of C are $\prec \alpha$. So the order types of both **B** and **C** are less than $\alpha$ and we get the wanted estimate for the order type of **A**.

Now to the general case

**Theorem 43** *Assume $n \geq 2$: If $|\alpha_1|, \ldots, |\alpha_n| < \beta$, then*



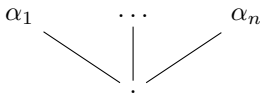
The ordinal




with at least two branches and all but the rightmost trivial, is both additively and multiplicatively closed

**Lemma 17** *Assume* $|K|, |L| < \gamma$*, then* $|K \cup L|, |K \times L| < \gamma$*.*

If we use the fact (not proved here) that there is always a maximal linear extension, then we can replace the $\leq$ with $<$ in the theorem. This is not done here. Now to the proof of the theorem. Assume we have a sequence **A** ending up with




We can assume that the for each element in the sequence we also have its subtrees. By induction over the heights we prove that all elements in **A** are $< \gamma$.

We write **A** as the union of the following sequences

**B1:** Trees with branching at root $\leq n$ and branch $1 < \beta$

**. . .**

**Bn:** Trees with branching at root $\leq n$ and branch $n < \beta$

**C:** Trees with branching at root $> n$

First note that since we have a topological embedding $\mathbf{C} < \beta$. Now look at **Bn**. Take an element $K$. The rightmost branch is $< \beta$, and by induction all the other branches are $< \gamma$. But then $|K| < \gamma$ by our order relation on trees. The other **B**'s are treated in the same way.

Then **A** can be written as the finite union of sequences each have order type $< \gamma$ — and the order type of **A** is $< \gamma$.

Our estimates are similar to those given by Diana Schmidt in her unpublished Habilitationsschrift.

The theorem has as consequence both Higmans lemma (the case n=2) and Kruskals theorem (the general case).

## 13.5 Transfinite induction

We assume that we have some way of representing ordinals, and then define transfinite induction for some formula $F$ up to some ordinal $\alpha$ by

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

First observe that transfinite induction behaves nicely with respect to inequality

$$\mathbf{TI}(\beta) \wedge \alpha < \beta \to \mathbf{TI}(\alpha)$$

Then we go through the build-up of the finite trees. We have

$$\mathbf{TI}(0, F)$$

This is trivially true. We have $\forall x < 0.Gx$ for any formula $Gx$.

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

So assume $\mathbf{TI}(\alpha, F)$ and $\mathbf{PROG}(F)$. We then get $\forall x < \alpha.Fx$. From $\mathbf{PROG}(F)$ we also get $\forall x \leq \alpha.Fx$. But then $\forall x < \Psi(\alpha).Fx$ since $x < \Psi(\alpha) \leftrightarrow x \leq \alpha$ as we have previously proved.

Now to the next step we prove a similar statement

$$\mathbf{TI}(\alpha, F) \to \mathbf{TI}(\Psi(\alpha, 0), F)$$

Assume $\mathbf{PROG}(F)$ and $\mathbf{TI}(\alpha, F)$. We can prove in Skolem arithmetic

- $F\Psi_0^0(\alpha)$
- $F\Psi_0^n(\alpha) \to F\Psi_0^{n+1}(\alpha)$

Hence by quantifier free induction $F\Psi_0^n(\alpha)$ for all $n$, and $\mathbf{PROG}(F)$ gives $F\Psi(\alpha, 0)$ which should be proved.

Now we come to the next step. Here we no longer have the uniformity in the induction formula. Given formula $F$ we find a formula $F^*$ such that

$$\mathbf{TI}(\alpha, F^*) \to \mathbf{TI}(\Psi(0, \alpha), F)$$

Now observe that $\Psi(0, \alpha)$ is built up from below by $\Psi_0^n(\alpha, \alpha-)$ where $\alpha- < \alpha$. We define $F^*$ so that it takes $\Psi_0^n(\alpha, \alpha-)$ steps instead of the small steps that $F$ takes. We let

$$F^*\alpha = \forall x.(\forall z < x.Fz \to \forall z < \Psi_0(x, \alpha).Fz)$$

Given $F^*\beta$, we then have for each natural number $n$

$$\forall x.(\forall z < x.Fz \to \forall z < \Psi^n_0(x,\beta).Fz)$$

And by induction over the quantifier formula we get

$$\forall n.\forall x.(\forall z < x.Fz \to \forall z < \Psi^n_0(x,\beta).Fz)$$

These are the key steps in the proof. Now to the details. We first prove

$$\mathbf{Prog}(F) \to \mathbf{PROG}(F^*)$$

Let $\alpha$ be given and assume

- $\mathbf{PROG}(F)$
- $\forall y < \alpha.F^*y$
- $\alpha- < \alpha$
- An $x$

We then prove $\forall n.\forall z < \Psi^n_0(x,\alpha-).Fz$. And furthermore $\forall z < \Psi(x,\alpha).Fz$ which gives $\mathbf{PROG}(F^*)$. Now assume $\mathbf{PROG}(F)$ and $\mathbf{TI}(\alpha)$. Then $\mathbf{PROG}(F^*)$ and $\mathbf{TI}(\alpha, F^*)$ which gives

$$\forall x.(\forall z < x.Fz \to \forall z < \Psi(x,\alpha).Fz)$$

and inserting $x = 0$ gives

$$\forall z < \Psi(0,\alpha).Fz$$

and using $\mathbf{PROG}(F)$ we get $F\Psi(0,\alpha)$ as desired.

We can also see what is the obstruction to proving $\mathbf{TI}(\Psi(0,0,0)$ — or $\mathbf{TI}(\epsilon_0)$. Let us say that we want to prove $\mathbf{TI}(\Psi(0,0,0),F)$ for a $\Pi^0_m$ formula $F$. We must prove $\mathbf{TI}(\Psi^n_0(0,0),F)$. But this proof depends on an ordinary induction over a $\Pi^0_{m+n}$-formula — and the proof is different for different $n$. It is no longer uniform. This is where Peano arithmetic is no longer able to prove what is true.

# Chapter 14

# Labeled trees

## 14.1 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 $\Lambda$. In addition we have an extra label $\infty$ which is larger than all the labels from $\Lambda$. Some examples of what we get:

- $\Lambda = \{0\}$ — this corresponds to our finite trees, the small Veblen ordinal
- $\Lambda = \{0, 1\}$ — this gives the Howard ordinal
- $\Lambda = \{0, 1, 2\}$ — to analyze $ID_2$
- $\Lambda = \{0, 1, 2, \ldots, \omega\}$ — to analyze $ID_\omega$

We shall not calculate the ordinals here. The works of Hilbert Levitz and Helmuth Pfeiffer give the correspondence between Takeuti- and Schütte-style ordinal notations — and it is to be expected that this can be carried over into our framework of finite labeled trees.

So we have given a well ordered set $\Lambda$ and the extra label $\infty$. 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

So first we define the $j$-subtrees. For $j = \infty$ we get the immediate subtrees as defined for finite trees. We have

$$\langle T \rangle_\infty = \langle T \rangle$$

Consider now a $j \in \Lambda$. 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 $T$)

The sequence of $j$-subtrees are denoted by

$$\langle T \rangle_j$$

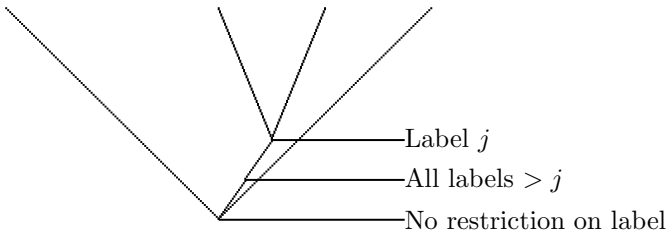
We can now define the orderings of trees

$$S <_j T \Leftrightarrow S \leq_j \langle T \rangle_j \vee (\langle S \rangle_j <_j T \wedge S <_{j+} T)$$
$$S <_\infty T \Leftrightarrow \textit{lexicographical ordering}$$

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 $\infty$
- 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

Let us use a figure to remind ourselves of the new notion, $j$-subtree




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).

Wr 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

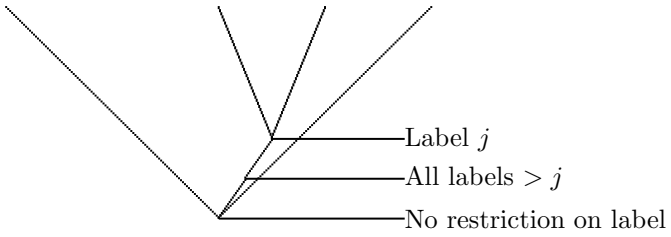
- $A \leq_j \langle B \rangle_j$ — giving $A <_j B$
- $B \leq_j \langle A \rangle_j$ — giving $B <_j A$
- $A \leq_{j+} \langle B \rangle_{j+}$ — giving $A <_j B$
- $B \leq_{j+} \langle A \rangle_{j+}$ — giving $B <_j A$
- $A \leq_{j++} \langle B \rangle_{j++}$ — giving $A <_j B$
- $B \leq_{j++} \langle A \rangle_{j++}$ — 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.

## 14.2 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 $i < j$.

## 14.3 Well ordering of finite labeled trees

The proof follows Gaisi Takeutis original proof that the the ordinal diagrams are well ordered. 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 minimal bad sequence

$$T_0^0 >_0 T_1^0 >_0 \ldots >_0 T_n^0$$

we must show how to extend it. By construction $T_n^0$ is minimal bad and can be continued

$$T_0^0 >_0 T_1^0 >_0 \ldots >_0 T_n^0 >_0 S_{n+1} >_0 S_{n+2} >_0 \ldots$$

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 observe

- we must decide whether certain trees are bad or not — i.e. a $\Pi_1^1$-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

$$\boxed{S <_j T \Leftrightarrow S \leq_j \langle T \rangle_j \vee (\langle S \rangle_j <_j T \wedge S <_{j+} T)}$$

Observe that the minimal badness excludes 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. In our case $0+ = \infty$. Therefore we have

$$T_0^0 >_\infty T_1^0 >_\infty T_2^0 >_\infty \ldots$$

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. We assume there is a bad sequence and construct a minimal bad sequence. The badness (and goodness) refers to the $<_i$-orderings, but the minimality is more complicated. We define a tree $T$ to be

**$i$-good:** No infinitely descending $<_i$-sequence

**$i$-minimal:** For every $j \leq i$ all elements of $\langle T \rangle_j$ are $j$-good

The proof starts with assuming we have a $<_0$-bad sequence. We construct an array of $i$-minimal $i$-bad sequences

$$\begin{array}{ccccccccccc}
T_0^0 & >_0 & T_1^0 & >_0 & T_2^0 & >_0 & T_3^0 & >_0 & T_4^0 & >_0 & \dots \\
T_0^1 & >_1 & T_1^1 & >_1 & T_2^1 & >_1 & T_3^1 & >_1 & T_4^1 & >_1 & \dots \\
T_0^2 & >_2 & T_1^2 & >_2 & T_2^2 & >_2 & T_3^2 & >_2 & T_4^2 & >_2 & \dots \\
T_0^3 & >_3 & T_1^3 & >_3 & T_2^3 & >_3 & T_3^3 & >_3 & T_4^3 & >_3 & \dots \\
\dots & & \dots & & \dots & & \dots & & \dots & & \dots
\end{array}$$

First we show how to proceed at successor stages — from row $i$ to row $i+1$. So assume we have an $i$-bad and $i$-minimal sequence

$$T_0^i >_i T_1^i >_i T_2^i >_i \dots$$

By $i$-minimality we cannot have $\langle T_i^k \rangle_i \geq_i T_i^{k+1}$ for any $k$. Hence

$$T_0^i >_{i+1} T_1^i >_{i+1} T_2^i >_{i+1} \dots$$

These trees are all $i$-minimal. We must construct a similar sequence which is $i+1$-minimal. This is done by using the sequence of $i+1$-subtrees as in the usual construction. But observe now that because of the gap-condition we do not create any new $j$-subtrees for $j \leq i$ — and hence the $i$-minimality is preserved under the construction.

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 $\infty$

$$T_0^\infty \quad >_\infty \quad T_1^\infty \quad >_\infty \quad T_2^\infty \quad >_\infty \quad T_3^\infty \quad >_\infty \quad T_4^\infty \quad >_\infty \quad \dots$$

by just taking the next line after taking one line for each label. The trees are $j$-minimal for each $j$.

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 the $i$-minimality.

# Chapter 15

# Well orderings

## 15.1 $\beta$-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

- $\omega$-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 $\omega$-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

- $\beta$-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.

## 15.2 System for $\beta$-logic

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 $\beta$-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 $\beta$-validity. To do this we make the following inessential assumptions about the sequential calculus

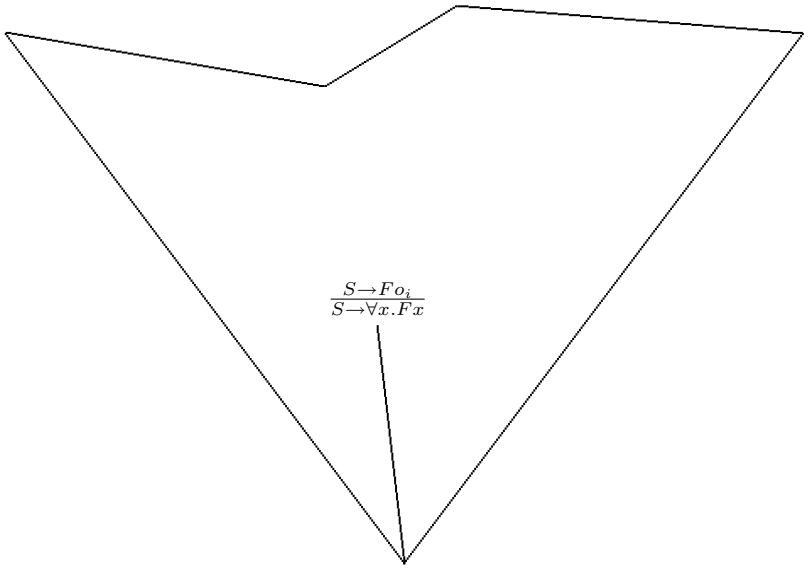
- there are no ordinal constants
- we have a fixed list $o_0, o_1, o_2, \ldots$ of ordinal variables
- no functionsymbols of ordinal sort

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 $\beta$-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 $\beta$-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 $\alpha_0$ for $o_0$, $\alpha_1$ for $o_1$, $\ldots$, $\alpha_{n-1}$ 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 6** *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 7** *$\mathcal{T}(\Gamma \to \Delta, \alpha)$ 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 $\alpha$. We need the following notations

- $\sigma \in \alpha^\star$ — $\sigma$ is a finite sequence of ordinals $< \alpha$
- $\sigma \sim \tau$ — $\sigma$ and $\tau$ are order isomorphic

The following lemmas are straightforward

**Lemma 18** *$\mathcal{T}(\Gamma \to \Delta, \alpha)$ is a tree. Given a sequence $\sigma$ in it, then all initial segments of $\sigma$ are also contained in it.*

**Lemma 19** *$\mathcal{T}(\Gamma \to \Delta, \alpha)$ 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 8** *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 $\alpha$-extension of $\mathcal{T}$.*

**Lemma 20** $\mathcal{T}(\Gamma \to \Delta, \alpha) = \mathcal{T}(\Gamma \to \Delta, \omega)[\alpha]$

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 $\alpha$. This is the mathematical core of the $\beta$-completeness theorem. Below we spell this out with more details.

**Definition 9** *We write $\vdash_\alpha$ for derivability in the sequential calculus with $\alpha$-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 21** *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 $\sigma$ 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 $\sigma$.

By induction over the securable sequences $\sigma$ 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 $\alpha$-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 22** *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 $\alpha$.*

**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 $\alpha_i$'s. This branch gives in the ordinary way the counterexample with the ordinals interpreted as the $\alpha_i$'s. There is of course no guarantee that we get all the ordinals $< \alpha$.

**End of proof.**

In first order logic and in $\omega$-logic we get a symmetry here. If it is not derivable in say $\omega$-logic, we get a counterexample where the universe is $\omega$. In $\beta$-logic we get an assymmetry. We can only assume that a counterexample is included in the $\alpha$. This assymmetry is hidden in our formulation of $\beta$-completeness by quantifying over all ordinals. This is how this is done

**Definition 10** *A tree of sequences $\mathcal{T}$ is strongly wellfounded if all its extensions $\mathcal{T}[\alpha]$ are wellfounded.*

**Theorem 44 ($\beta$-completeness)** *Given a sequent $\Gamma \to \Delta$: $\Gamma \to \Delta$ is $\beta$-valid if and only if $\mathcal{T}(\Gamma \to \Delta, \omega)$ is strongly wellfounded.*

## 15.3 $\beta$-completeness

The notion of $\beta$-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 $\beta$ is to remind one of the French "bon ordre". This problem can be reduced to the $\beta$-logic of the previous section, and Mostowskis problem were answered by Jean-Yves Girard with his $\beta$-completeness.

Mostowski and his students observed that $\beta$-validity is a complete $\Pi^1_2$-notion, and used this as an argument that it would be hard to characterize a $\beta$-logic. In Girards solution the $\beta$-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 $\omega$-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 45** *To be strongly wellfounded is a complete* $\Pi^1_2$ *notion.*

It is obvious a $\Pi^1_2$ notion. That it is complete follows from the $\beta$-completeness. We observe that

**Lemma 23** $\mathcal{T}$ *is strongly wellfounded if and only if* $\mathcal{T}[\Omega]$ *is wellfounded, where* $\Omega$ *is the first uncountable ordinal.*

**Proof:**
Assume $\mathcal{T}$ is not strongly wellfounded. There is then an $\alpha$ 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 $\alpha$ 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 16

# Conclusions

## 16.1 The way ahead

The explorations here are far from finished. Let us mentions some possibilities

**Fix point theories:** A good generalization of first order arithmetic are theories of fix points and iterated fix points. If we do not demand that the fix points are minimal, we get theories which seems to fit nicely in with our theory of finite trees. And then we have connections between those theories and versions of Kruskals theorem.

**Inductive theories:** These are fix point theories and iterated fix point theories where we also demand that the fix points are minimal. They should have been analyzed with labeled finite trees. This belongs also to the way ahead.

**Takeutis analysis:** The original aim for the whole project was to understand his proof of consistence for $\Pi^1_1$-analysis. This seems to be within our grasp — but lies also on our way ahead.

**Homogeneous trees:** We have — some time ago — developed a further development of the strongly wellfounded homogeneous trees used in the $\beta$-completeness theorem. In particular there is an induction principle on those trees which we would like to use here.

**Connections:** There should be a number of interesting connections between the themes opened up here. I would be especially interested in seeing connections between Rabins theory of automata and traditional proof theoretic themes.

I started the development by giving a list of pioneers. Let us end up with mentioning the four key pioneers

- Thoralf Skolem

- Gerhard Gentzen
- Gaisi Takeuti
- Jean Yves Girard

Their influence can be seen throughout the monograph.