Ordinals

Remi Pallen, Thomas Lamiaux, Cyril Pujol, Quentin Chuet, Pranay Agrawal, Marius Belly-Le Guilloux

September 2021 - December 2021

Contents

Introduction

When one begins in research, some advice can be useful. The first one is that it is important to use public tools to speak about your works, because if you use private tools like gmail for exemple, Google can take over your work. We already know that Google and Microsoft know 6 months in advance on which subject new papers will be published. The goal is to keep in hand your project.

One more thing is that you should work with people with the same motivations as yours. Four motivations are recurrent for searchers : the curiosity and will to discover more deeply a field of science, the challenge to prove some centenary conjecture, enjoying doing science or the usefulness of some discoveries. These four aspirations can be summarized in a sort of "compass":

Your priorities will lead you to prefer some subject or fields to others. It is particularly important to choose a PhD supervisor with similar expectations.

Chapter 1

Formulate a Problem

We consider the following program (Prog1.java):

```java
int x = Isn.readInt();
while (x > 0) x = x - 2;
System.out.println("Done");
```

We would like to prove that this program terminates for all x.
It would be troublesome to study the behaviour of this program without a layer of abstraction.
Therefore, we translate the problem to simpler models of computation.

1.1 Prove the termination with rewriting

In order to prove that the program terminates we have chosen to modelise it by a rewriting system.
A rewritting system is a relation on terms, stable by susbtitution and context.

We define our terms as $n := 0 \mid S\ n$ and only one rewriting relation as $R := S\ S\ x \to x$.

Theorem 1.1.1. *R terminates.*

Proof. To prove this theorem, we need a semantic. We take $[\![\_]\!] : T \to \mathbb{N} : [\![0]\!] = 0$ and $[\![S\ x]\!] = [\![x]\!] + 1$. Hence $[\![S\ S\ x]\!] = [\![x]\!] + 2$ and then $[\![S\ S\ x \to x]\!] = -2$. Has $(\mathbb{N}, <)$ is well-founded, we can't have an infinite succession of this rewriting rule. Yet as it is the only one, R terminates. □

1.2 Prove the termination with sequences

To prove the termination of the program, we use a sequence where every element correspond to one state of the system. In our exemple, the state is given by the value of the variable x.

The sequence is defined as followed :

$$u_0 = x_0$$
$$u_{n+1} = u_n - 2 \text{ if } u_n > 0$$

To show that the program terminates, we have to show that the sequence is finite.

This is true because u_n is strictly decreasing, has its values in $\mathbb{Z}$ and is lower-bounded by $\min(x_0, -1)$. Therefore it cannot be infinite.

1.3 Proving the termination using state transition system

We define a state transition system $T = (Q, I, R)$ from a while loop Π as follows :

- $Q = [\![0, len(\Pi) - 1]\!] \times \prod\limits_{y \in Var} dom(y)$ the set of states
- $I \subseteq \{0\} \times \prod\limits_{y \in Var} dom(y)$ the set of initial states
- $R \subseteq Q \times Q$ the set of transitions
- $((i, \rho), (j, \rho')) \in R \iff$ after execution of line i in environment ρ, the new environment and the new line to execute may be respectively ρ' and j

The state transition system associated with our while loop is the following ($I = Q$)

$$\begin{tikzcd}
\bullet\ \bullet\ \bullet & (0,(-1,)) & (0,(0,)) & (0,(1,)) \arrow[ll, bend left] & (0,(2,)) \arrow[ll, bend right] & (0,(3,)) \arrow[ll, bend left] & \bullet\ \bullet\ \bullet
\end{tikzcd}$$

Lemma 1.3.1. *(i) The previous state transition system has no infinite path.*
(ii) Its initial states are exactly the possible valuations at the beginning of Π

Proof. (i) $((i, m), (j, n)) \in R \implies n < m$ and $m \in \mathbb{N}$ which proves the result because $\mathbb{N}$ is well founded.
(ii) Any integer may have been chose by the user. □

Lemma 1.3.2. *Assuming that the initial states are the possible valuations at the beginning of* Π*,*

$$\Pi \textit{ always terminates } \iff T \textit{ has no infinite path}$$

Proof. $(s_i)_{i \in \mathbb{N}}$ is an infinite path in T.
$\iff s_0$ is a possible state at the beginning of Π [by hypothesis] and $\forall i \in \mathbb{N}, (s_i, s_{i+1}) \in R$ [by definition]
$\iff s_0$ is a possible state at the beginning of Π and the execution of Π in the state s_i may lead to the state s_{i+1}.
$\iff (s_i)_{i \in \mathbb{N}}$ is an infinite execution of Π. □

Lemma 1 and 2 proves the termination of the while loop, hence the termination of the algorithm.□

1.4 Generalization

We can notice that the only element in a programming language that can cause uncertainty regarding termination is the while loop (if and for loops are easy to study).
Let's consider a generalization of Prog1.java:

```java
int x = Isn.readInt();
while (x > C) x = f(x);
System.out.println("Done");
```

Where C is a constant and f is a computable function over integers.
In the case of program 1, $C = 0$ and $f(x) = x - 2$.

The question is, given the initial value of x, does this program end?

This problem is more succinctly summarized in the following sequence:

$$u_0 = x_0$$
$$u_{n+1} = f(u_n) \text{ if } u_n > C$$

Definition 1.4.1 (Sequence termination)**.** The sequence $(u_i)_{i\geq 0}$ *terminates* if $\exists k \in \mathbb{N}$ such that $u_k \leq C$.
In that case, the sequence has a finite number of terms.

The associated decision problem is `P(f,C)` : "Given x_0 as input, does the sequence $(u_i)_{i\geq 0}$ terminate ?"
This problem is equivalent to deciding if the program described above terminates depending on the input.

1.4.1 Necessary and sufficient condition for termination

Let $(u_i)_{i\geq 0}$ be the sequence defined previously by x_0, C and f.

Definition 1.4.2 (Bounded sequence)**.** $(u_i)_{i\geq 0}$ is *bounded* if $\exists K \in \mathbb{Z}$ such that $\forall n \in \mathbb{N}, u_n \leq K$.

Definition 1.4.3 (Repeating sequence)**.** $(u_i)_{i\geq 0}$ is *repeating* if $\exists n \in \mathbb{N}, m \in \mathbb{N}, n \neq m$ such that $u_n = u_m$.

Theorem 1.4.4. *The sequence $(u_i)_{i\geq 0}$ terminates iff it is bounded and nonrepeating.*

Proof.
($\implies$) Suppose $(u_i)_{i\geq 0}$ terminates, i.e. $\exists k \in \mathbb{N}$ such that $u_k \leq C$.
Let $K = \max_{0\leq i\leq k}(u_i)$. K is an upper bound on the sequence.
By contradiction, suppose $(u_i)_{i\geq 0}$ is repeating, i.e. $\exists n < m < k$ such that $u_n = u_m$.
Then, by recurrence, we would have $u_{n+1} = u_{m+1} \implies , \ldots , \implies u_{k-m+n} = u_k$.
However, $u_{k-m+n} > C$ by construction of the sequence, yet $u_k \leq C$. We have a contradiction.

($\impliedby$) Suppose $(u_i)_{i\geq 0}$ is bounded by $K \in \mathbb{N}$ and nonrepeating.
Then the sequence takes a finite number of values between C and K, and all terms are distinct.
Therefore, the sequence contains a finite number of terms, and hence terminates.

□

Corollary 1.4.4.1. *Let $f : \mathbb{Z} \to \mathbb{Z}$ be a computable function over integers, and $C \in \mathbb{Z}$ a constant.*
If $\forall x_0 \in \mathbb{Z}$ the associated sequence $(u_i)_{i\geq 0}$ is bounded, then the decision problem $P(f, C)$ is decidable.

Proof. Either the sequence terminates, or it is repeating. In the first case, the answer to the problem is Yes. Otherwise, we detect that the sequence is repeating (by comparing the current term of the sequence with all previous terms), and the answer to the problem is No. □

Application

The previous theorem is hard to use because it is generally difficult to bound a recursive sequence.
In the following, we give a criterion for decidability based on f rather than the sequence itself.

- If we can compute $K > C$ such that $\forall x \geq K, f(x) \leq x$:
  - The sequence is bounded by $max(x_0, f(C+1), f(C+2), \ldots, f(K))$, therefore we can apply the previous corollary to prove that $P(f, C)$ is decidable.
- If we can compute $K > C$ such that $\forall x \geq K, f(x) \geq x$:
  - The sequence is infinite for $x_0 \geq K$) (increasing sequence), therefore it does not terminate.
  - For $x_0 < K$, either the sequence is greater than K at some point (therefore it is infinite), or it bounded by K, in which case we can detect whether the sequence is repeating and conclude.
  - Therefore $P(f, C)$ is decidable.

Note: It is not sufficient to prove that some arbitrary K fulfills the property, we must also compute such K in order to have an actual algorithm.

Proposition 1.4.5. *$P(f, C)$ is decidable if f is a polynomial function of x with integer coefficients.*

Proof. Let n be the degree of the polynomial $p(x) := f(x) - x$ $(a_n \neq 0)$.
Let's assume $x > 1$. We have two cases to consider.

If $a_n > 0$: Let $I = \{i \mid 0 \leq i < n \text{ and } a_i < 0\}$. Let $K = -\sum_{i \in I} a_i$. Consider $x \geq K$.
We have $p(x) = \sum_{i=0}^{n} a_i x^i \ \geq\ a_n x^n + \sum_{i \in I} a_i x^i \ \geq\ a_n x^n + \sum_{i \in I} a_i x^{n-1}$
$\geq\ a_n x^n + (-K)x^{n-1} \ \geq\ a_n x^n - x^n \ \geq\ (a_n - 1)x^n \ \geq\ 0$
We have computed K such that $\forall x \geq K,\ p(x) \geq 0$ (i.e. $f(x) \geq x$). Therefore $P(f, C)$ is decidable.

If $a_n < 0$: Let $I = \{i \mid 0 \leq i < n \text{ and } a_i > 0\}$. Let $K = \sum_{i \in I} a_i$ Consider $x \geq K$..
We have $p(x) = \sum_{i=0}^{n} a_i x^i \ \leq\ a_n x^n + \sum_{i \in I} a_i x^i \ \leq\ a_n x^n + \sum_{i \in I} a_i x^{n-1}$
$\leq\ a_n x^n + K x^{n-1} \ \leq\ a_n x^n + x^n \ \leq\ (a_n + 1)x^n \ \leq\ 0$
We have computed K such that $\forall x \geq K,\ p(x) \leq 0$ (i.e. $f(x) \leq x$). Therefore $P(f, C)$ is decidable.

□

Proposition 1.4.6. *There exists a computable function f, and $C \in \mathbb{Z}$, such that $P(f, C)$ is undecidable.*

Proof. The idea is to run an universal Turing machine step-by-step (which itself simulates an arbitrary Turing machine on an arbitrary input).
Each term of the sequence $(u_i)_{i \geq 0}$ is a binary encoding decribing a Turing machine and its current configuration (we assume the first bit of an encoding is 1, so that it always corresponds to a strictly positive integer).

We consider the following function:

```
f(x):
    if x is not a proper encoding, return 0.
    decode x to get the TM and its configuration.
    if the configuration is in a halting state, return 0
    execute the next action of the TM on the configuration.
    return the encoding of the TM and the modified configuration.
```

Decoding, simulating, and encoding are computable operations, therefore f is computable.
We reduce the Halting Problem to P(f,0) :
A Turing machine M with initial state q_0 halts on input x if and only if the sequence $(u_i)_{i \geq 0}$ starting at $encode(M, "q_0 x")$ terminates (the sequence literally simulates M on input x).

Since the Halting Problem is undecidable, we conclude that $P(f, 0)$ is undecidable. □

Chapter 2

To formulate a conjecture

Let us come back to our example (Prog1.java). Another way to prove the termination would be to map the state of the program to a well-founded order. It would be convenient to suppose without loss of generality that such a mapping always exists in $(\mathbb{N}, <)$. However, is it truly always the case ?

Conjecture 2.0.1. *Let $(Q, \leftarrow)$ be the states of the program reachable from initial states, quasi-ordered by the possibility for the program to go from a state to an other.*
If the program always terminates, there exists an order homomorphism (i.e an increasing function) from Q to $\mathbb{N}$.

How to tackle this conjecture ? First, it is always interesting to define the most precisely possible the objects that we use. For instance, one must agree on what is a program. Is it a theoretical algorihm operating on $\mathbb{N}$ and $\mathbb{R}$? An implemented algorithm working with a bounded memory on *int* and *float* objects ? Do we allow non-deterministic operations ?

Proposition 2.0.2. *The conjecture holds for a program working with a bounded memory.*

Proof. Q is finite. Therefore, the program terminates if and only if $\leftarrow$ is an actual order. In that case, $(Q, \leftarrow)$ is a DAG which means that it can be topologically ordered. We conclude by observing that an order homomorphism from Q to $\mathbb{N}$ is exactly the reverse of a topological order. □

Let us allow ourselves the use of a function $rand()$ that returns uniformely randomly a non-negative integer. Does the conjecture still holds ? To answer the question, it is useful to try to answer related ones. If $(\mathbb{N}, <)$ is not sufficient, then what is ? Are there countable ordered sets that cannot be mapped to $\mathbb{N}$ with an order homomorphism?

Lemma 2.0.3. *Let $E = \mathbb{N} \sqcup \{\omega\}$ ordered by the usual order relation on $\mathbb{N}$ extended with $n < \omega \ \forall n \in \mathbb{N}$. Then there is no order homomorphism from $(E, <)$ to $(\mathbb{N}, <)$*

Proof. Let us suppose that such a mapping ϕ exists. Then $\forall x \in \mathbb{N}, \phi(x) < \phi(\omega)$ which means that $\phi(E)$ is finite. By the infinite pigeonhole principle, there exists x and y such that $x \neq y$ and $\phi(x) = \phi(y)$. However, $<$ is total on E, hence the contradiction. □

Now, we may try to use this result to find a counter-example.

Proposition 2.0.4. *The following program (Prog2.java) terminates and is homomorphic to E but not to $\mathbb{N}$.*

```
int y = 0;
int x = rand();
while (x > 0) x = x - 2;
System.out.println("Done");
```

Proof. Let us suppose the existence of an homomorphism ϕ to $\mathbb{N}$. and let $n = \phi(q_0)$ where q_0 is the inital state. Then, an execution of the program takes at most n steps to terminate. However, the call to $rand()$ may return $2n+2$ which would need at least $n+1$ steps to terminate. This proves that $(\mathbb{N}, <)$ is not sufficient. On the other hand, let $\psi : Q \to E$ such that $\psi(q_0) = \omega$ and $\psi(q) = [\![x]\!]_q$ otherwise. Then the maximality of ω and q_0 as well as the proofs of termination used in Chapter 1 prove that ψ is an order homomorphism. □

This (quite simple) counter-example is particularly interesting. Indeed, it seems to show a way to incrementally build a counter-example to the sufficiency of an order. Thus, the following properties can be verified in the same way.

Proposition 2.0.5. *The following program (Prog3.java) terminates and is homomorphic to* $\mathbb{N} \sqcup \{\omega, \omega + 1\}$ *but not to* $\mathbb{N} \sqcup \{\omega\}$.

```
int y = 0;
y = 1;
int x = rand();
while (x > 0) x = x - 2;
System.out.println("Done");
```

Proposition 2.0.6. *The following program (Prog4.java) terminates and is homomorphic to* $\bigsqcup_{k \in \{0,1\}} (k\omega + \mathbb{N}) \sqcup \{2\omega\}$ *but not to* $\bigsqcup_{k \in \{0,1\}} (k\omega + \mathbb{N})$.

```
int y = 0;
int x = rand();
while (x > 0) x = x - 2;
int z = rand();
while (z > 0) z = z - 2;
System.out.println("Done");
```

Proposition 2.0.7. *The following program (Prog5.java) terminates and is homomorphic to* $\bigsqcup_{k \in \mathbb{N}} (k\omega + \mathbb{N}) \sqcup \{\omega^2\}$ *but not to* $\bigsqcup_{k \in \mathbb{N}} (k\omega + \mathbb{N})$.

```
int y = 0;
int x = rand();
while (x > 0) {
   int z = rand();
   while (z > 0) z = z - 2;
   indent x = x - 2; }
System.out.println("Done");
```

Chapter 3

Giving a Talk

Someone has remarked that 'An ideal math talk should have one proof and one joke and they should not be the same'

Ronald Graham

A difficult part of a research career is realizing no one is going to willingly read your papers. In order to attract attention to your hard work, the final step of writing a math paper is to prepare and give talks on it.

It's important to note that a talk is not a lecture: there shouldn't be too much of a focus on definitions and proofs, those who are interested will have the opportunity to read on them directly in the paper.

At the end of the talk, you should not expect more from the audience than to remember the important results of your paper, and to be convinced that they are indeed correct. If you manage to do that, then you should consider your talk to have been successful.

When preparing a talk, there are two important questions to ask before doing anything else:

- **Know your audience**
  One should know their audience before giving the talk, the content and the presentation may vary depending on the audience and their pre-requisites.
  For e.g., if you are giving a talk on *ordinals* to two different batches, say one batch of people who study maths and the other who study informatics. Then you might present the topic by using set theory to the former batch and using programming to the latter.
- **Duration and number of slides**
  One should not present an entire paper during a talk, rather they should present the key results, motivation and further improvements possible. Generally one can spend 1-4 minutes per slide and around 5-15 minutes in introduction.
- **Preparing a talk**
  One should remember 4 key points while preparing a talk :
  - *Motivation* :
    One should begin the talk with some motivation or questions which will serve as an ice breaker and engage the audience to formulate the problem and be an interactive part of the presentation
  - *Problem* :
    After giving enough motivation, one can start formulating the problem with the use of enough definitions so that the audience agrees and can move forward. A small pause after formulating the problem gives some time for audience to think.
  - *Solution* :
    Now, this is the main part of presentation is to present the solution, since your work is quite complicated the details of solutions can't fit in the small presentation, and hence you should present the key idea behind the solution, and take questions to make it clearer.

  - *Further improvements and questions* :
    This is also a very important part of the talk, it gets your work noticed as the quality and difficulty of possibility of improvements will make your work appear more interesting.

As a general rule, half of the talk should be dedicated to the motivation (say why is the problem important, answer it, and mention future work or open problems on the subject).

Also, plan the talk in such a way that you have time to answer questions or clarify some notions, the point of a talk is to have an interaction with the audience.

Example talks

- **Axiom of choice**

  - *Motivation* :
    One can talk about origin of numbers here, some history on infinities
  - *Problem* :
    Present the problem i.e. ”Can one choose something from infinity?” or ”Is every set can be well-ordered?”
  - *Solution* :
    Present a concise solution.
  - *Further improvements and questions* :
    If every set can be well-ordered. What will be the well order on $\mathbb{R}$?

Chapter 4

Vocabulary

Mathematics is the art of giving the same name to different things.

Henri Poincaré

4.1 Finding Definitions

Using the Pareto principle as a rough approximation, we can estimate that 80% of the words that exist are jargon: We may be able to communicate simple ideas using only 20% of words, but eventually we are limited by a lack of vocabulary and we need to learn the other 80% to express more complex ideas.

During our endeavour to learn the definition of new words, we may face a few difficulties:

- Homonyms: Multiple unrelated meanings for a single word.
  Researchers can be quite uncreative when they give a name to a concept. Perhaps some see the process of defining words to be the least interesting part of research since it does not involve proofs, but we should not neglect the importance of finding the simplest and most rigorous way to define an informal concept. Examples: *root* (analysis, graph theory), *rank* (linear algebra, set theory), *rational* (algebra, formal languages).

- Synonyms: Several words for a single meaning.
  When several researchers work on the same topic, they may use different words for expressing the same idea. This is very unhelpful, because we have to learn all of them anyway. Examples: preorder and quasiorder (set theory), inverse and opposite (”multiplicative” groups vs ”additive” groups).

- Different yet equivalent meanings for a word:
  It may be the case that a concept can be expressed in multiple different ways. In that case, it is difficult to decide which of these definitions is more legitimate, but that doesn’t matter as long as we prove they are equivalent. Example: well-founded order.

- Different notations and conventions:
  Notations convey implicit information (n is often an integer, x a variable, E a set...). We may take these notations for granted, but it becomes apparent when reading older articles of research that these conventions are not always respected. A problem arises when researchers don’t settle on a single convention. For example, there isn’t a consensus on whether the set of natural numbers $\mathbb{N}$ should contain 0, since not everyone agrees on the definition of positive numbers.

- Lack of intuition or examples:
  Once you get a good grasp of a definition, it is easy to forget that even the most simple examples are not trivial to the uninitiated. If a definition does not provide a helpful intuition or example to illustrate its meaning, it is up to the reader to come up with those themself.

4.2 Adjectives

In science, it is often the case that researchers add adjectives to generalize a notion. This produces some pretty jarring instances of apparent contradictions: for example, Euclidean geometry is part of Non-Euclidean Geometry.
As a result, when there is a possible ambiguity, we resort to using adjectives to separate the notions, and the original word itself can never stand on its own. For example, we often have to mention whether a computing model is deterministic or nondeterministic, because there isn't a default case.

4.3 A few words of vocabulary concerning Set Theory

- Order = partial order = non-strict order : Reflexive, transitive, antisymetric relation.
- Strict order: Irreflexive transitive relation.
- Preorder = quasi-order : Reflexive transitive relation.
- Total order : partial order such that $\forall x \forall y, x \leq y$ or $x \geq y$.
- Well-founded order : $\forall S \subseteq E, S$ admits a minimal element.
  Equivalently, every decreasing sequence is finite.
- Well-order : total well-founded order.
  Equivalently, for any infinite sequence$(x_n)_{n\in\mathbb{N}}$, $\exists i < j$ such that $x_i \leq x_j$
- Ordinal : strictly well-ordered set (with respect to $\subset$) such that every element is also a subset.
- Chain of a partial order : a set $S \subseteq E$ such that $\leq_S$ is a total order on S.
- Length of a chain : $max(|S| : S \subseteq isachain)$.
- Rank of an element: $\rho(x) = \cup\{\rho(y) + 1 : y < x\}$.
  The rank is an ordinal. The notation $x + 1$ means the successor of x: $x \cup \{x\}$

4.3.1 Observations:

- We have three words to designate one concept (order). However, we might use the adjectives "partial" and "non-strict" to insist that an order is not total or strict.
- A "strict order" is not an "order".
- "well-founded order" and "well-order" sound confusingly similar, so we should be careful.

Chapter 5

How to read a paper

There are 3 ways to read a paper :

- **just to find solution**
  Look for main theorem, (maybe start reading backwards), try working with examples (for e.g. replace general sets S with $\mathbb{N}$). No need to understand everything, you only need to understand the notion that are used in the theorem you want to use. Depending on the situation, you may need(want) to understand the proof or not. Working with exemples allows you to apply the theorem in a context that you are familiar with, which may help you understand the gist of the theorems. It can also help you detect if you misunderstood something. Always start by reading the abstract, to know if you will find your answer in this paper.

- **reading from top to bottom**
  you might see terms/notion which don't make sense, then you can either guess them with context or look for references. This way of reading papers is rare, it can happen when you want to familiarise yourself with a subject, like at the beginning of a thesis. Keep in mind that you can also read books, they are in general more practical in order to discover a new subject. Wikipedia articles can generally be read that way.

- **reading like historian** In this case, you should pay attention to details like the notations and the notion used to prove the theorems. For now, this does not apply to us.

Chapter 6

From hydra to ordinals

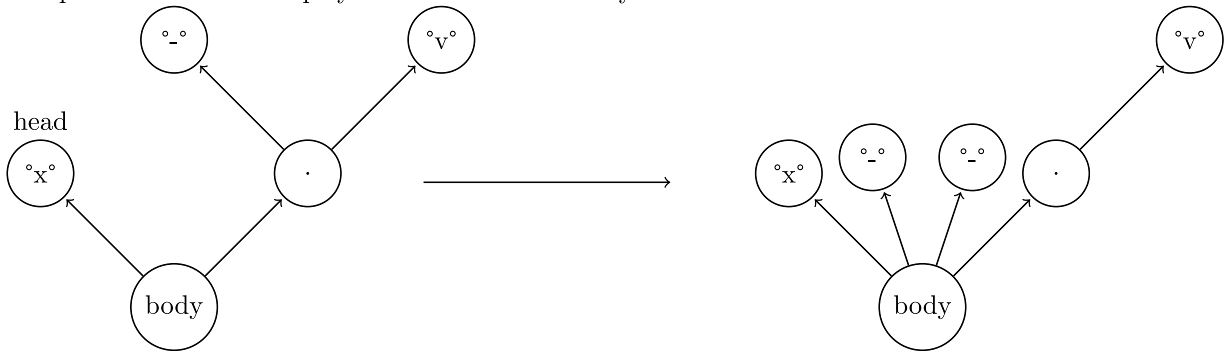
In this chapter, we will take a look at a famous termination problem, and analyse it with the point of view of ordinals.

6.1 The hydra problem

In the greek mythology, the hydra is a monster with many heads, which Hercules has to fight. This monster has the ability to instantaneously grow two heads for each one that is cut. The mathematical problem associated with it is the following:

Let a hydra be a rooted tree, where the root is the "body" of the hydra and every leaf is a "head". We denote the rank of a head as its distance in the tree to the body. We consider a game, were the initial state is a hydra, and at the i-th turn, the player can "cut" a head H, which remove this head, and if the rank r of the head is strictly positive, it add i heads of rank $r-1$ as sons of the grandfather of H. The player wins when they are no head left.

The problem is : can the player win ? Does she always win ?



6.2 The link with ordinals

To every hydra, one can assign the ordinal $\sum_i \omega^i(\#\text{ heads of rank } i)$. Then regardless of the number of turn and of the head chosen by the player, the move strictly decrease the ordinal associated with the hydra. This proves that the game always terminate, and thus that the player always win. However the number of steps might be enormous.

One can also prove the temination using a given strategy : If the player always cut a head of maximal rank, then the ordinal $\omega.(\text{maximal rank}) + (\#\text{ heads of maximal rank})$ is strictly decreasing with every move using the strategy. One can observe that this variant does not use powers of omega. This is only possible when we consider one given strategy.

The hydra problem can be reformulated as a rewriting problem, but this is actually not very practical.

Chapter 7

How far one should push an idea ?

7.1 Common idea ?

The relation $a^2 + b^2 = c^2$ in a square triangle is well kwnon in mathematics. So called the "Pythagor's theorem" in France, it was actually discoverd long before him. Actually it would seem that as soon that a civilisation find food and shelter - and so had time to thought - discoverd it soon after. This process can even be found in parallel civilisation such as the Maya.

A similar developement in different par of the world is actually a good thing because its allowed us to do math. Futhermore, new developement also happened in the same time such as with the phonogram. It this process can also be witness during the cold war. One downside is that one research must always check if a proof or an idea is really new or not. Especially when one is a beginner in a new topic research.

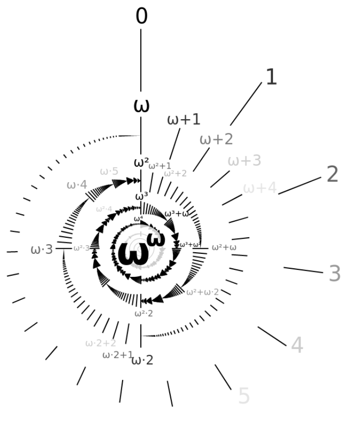
7.2 Why generalize further and when to stop ?

Some people are generalising for generalising. This can be an issue because you may start to talk about structures that have so few properties that it is impossible to prove anything on them. For instance you can extend vectorial space to module but you can't go much futher without having to loose keys properties such as the identity element.

Even though definition can seem pointless at first, they can become usefull with time. For instance, reimannian manifold are now used to described relativity theory.

7.3 Counter futher then ω^ω

So far we have defined that :



However we can defined $\omega^\omega + 1$, $\omega^\omega 2$, $\omega^\omega \times \omega^\omega$, ω^{ω^ω} so what next number could be defined ?

Lets definine the sequence $(c_n)_{n\in\mathbb{N}}$ such that $c_0 = \omega$ and $c_{n+1} = \omega^{c_n}$. Then we can define ϵ_0 to be $\sup((c_n)_{n\in\mathbb{N}})$. This number is bigger than the previous one moreover it is big enough to be stable by some operation. For instance $\omega^{\epsilon_0} = \epsilon_0$. As seen before, Cantor normal form enables finite representation of ordinals. It is saying that for all ordinal defined so far can be written as $\omega^{\beta_1}c_1 + ... + \omega^{\beta_n}c_n$ with $c_1, ..., c_n < \omega$ with $\beta_1 \geq ... \geq \omega_n$.

One downside of this previous property is that for number bigger then ϵ_0 the representation is no longer working as the power collapses.

Chapter 8

Provability in PA

8.1 A termination problem

We were interested in the difference between these two properties:
Prop1: The sequence u_n terminates.
Prop2: The proposition "The sequence u_n terminates" is provable in PA.
To answer this question, we have first to express the property "The sequence u_n terminates" in PA.

How to define a sequence in PA?
$\rightarrow$ Thanks to a relation $A(x, y)$ with 2 free variables in which x represents n and y represents u_n. It is easy to see that we can enforce A to be a function. But which sequences can we define in a such way? In fact, according to the proof of the Church's theorem (undecidability of PA), all computable sequences are definable like this. So we restrict ourselves to computable sequences.

How to define termination of a sequence?
$\rightarrow$ Several choices are possible, we chose to say that u_n terminates if $\exists n \in \mathbb{N}$ such that $\forall m < n\ u_m$ is defined and $\forall m \geq n$, u_m is not. So if $A(x, y)$ defines a sequence u, it terminates if $\exists x, \neg(\exists y A(x, y))$.

Now, let's return to our 2 propositions. Does:
•"The sequence u_n terminates" $\Rightarrow$ The proposition "The sequence u_n terminates" is provable in PA? (So does Prop1 $\Rightarrow$ Prop2?).
•"The sequence u_n does not terminate" $\Rightarrow$ The proposition "The sequence u_n does not terminate" is provable in PA?

The Church's theorem give us that at least one of these two propositions are false. Indeed, knowing if a proposition is provable in PA is recursively enumerable (we can enumerate proofs). The first point give us that *Halts* is recursively enumerable and the second one that it is co-recursively enumerable. So if these two propositions are true, *Halts* is decidable, which is not the case. But is one of them true?

We can express in PA the sequence u such that $u_n = 1$ if $\perp$ is not provable in PA with a rule of size n, is not defined if not. Obviously, u terminates iff $\perp$ is provable iff PA is not consistent. So if the second point is true, PA is consistent $\Rightarrow$ The proposition "The sequence u_n does not terminate" is provable in PA. So if PA is consistent, we can prove in PA that PA is consistent. But, according to the second incompleteness theorem of Gödel, PA can't prove its own consistency. So the second property is false.

The idea to prove that the second point is false is the same. In sequent calculus, we can not derive $\vdash \perp$ without cut, and there exists an algorithm which takes in input a proof and return a proof of the same theorem without cuts. Therefore the existence of this algorithm implies the consistency of PA. So if we find a sequence which terminates if and only if a such algorithm exists, first proposition is false since PA can not prove its consistency.

8.2 Advice on research

There is at least two rules to respect when a group works on a problem:
•Keep using the same language, if not, not everyone will understand what it is said and it is damaging for the entire group.
•Don't do little groups, only one person speak at a time. You have to learn to listen your partners until the end and if it is you who are speaking, you have to be brief.