Foundations of Computer Science
Logic, models, and computations

Chapter: Incompleteness of arithmetic

Course CSC_41012_EP

of l'Ecole Polytechnique

Olivier Bournez

bournez@lix.polytechnique.fr

Version of August 20, 2024

Incompleteness of arithmetic

In 1930, Kurt Gödel proved a result whose philosophical consequences in science started a revolution: He proved that any sufficiently expressive theory to capture the arithmetic reasoning's is necessarily incomplete, that is to say that there exists some statements that cannot be proved, and whose negation can nor be proved.

This theorem is largely considered as one of the greatest achievements of the logic in the 20th century.

With all the previous ingredients, we are in position to understand this theorem, and to provide a full proof. This is the objective of this chapter. Actually, we will propose a proof due to Turing. We will only mention the proof from Gödel, that allows to say more.

1 Theory of Arithmetic

1.1 Peano axioms

The question we are focusing on now is to try to axiomatise the arithmetic, that is to say the properties of the inters.

We have already presented in Chapter 6, the axioms of the arithmetic from Robinson and the axioms from Peano: One expects that all these axioms are satisfied on the integers, that is to say in the *standard model of the integers* where the base set is the integers, and where $+$ is interpreted by addition, $*$ by multiplication and $s(x)$ by successor function $x \mapsto x+1$.

In other words, one expects that these axioms have at least one model: *The standard model of the integers*.

Given some closed formula F on the signature containing these symbols, F is either true or false on the integers (that is to say in the standard model of the integers). Call *theory!of the arithmetic* the set $Th(\mathbb{N})$ of closed formula F that are true over the integers.

1.2 Some concepts from arithmetic

It is possible to prove that numerous concepts from number theory can be defined perfectly from these axioms.

For example, we can express the following concepts:

- INTDIV(x, y, q, r) defined as "q is the quotient and r the remainder of the euclidean division of x by y".

  Indeed, this can be written as formula:

$$(x = q * y + r \wedge r < y).$$

- DIV(y, x) defined as "y divides x".

  Indeed, this can be written:

$$\exists q \ \text{INTDIV}(x, y, q, 0).$$

- EVEN(x) defined as "x is even". Indeed, this can be written:

$$\text{DIV}(2, x).$$

- ODD(x) defined as "x is odd. Indeed, this can be written:

$$\neg \, \text{EVEN}(x).$$

- PRIME(x) defined as "x is prime". Indeed, this can be written:

$$(x \geq 2 \wedge \forall y(\text{DIV}(y, x) \Rightarrow (y = 1 \vee y = x))).$$

- POWER$_2(x)$ defined as "x is a power of 2". Indeed, this can be written:

$$\forall y((\text{DIV}(y, x) \wedge \text{PRIME}(y)) \Rightarrow y = 2).$$

1.3 The possibility of talking of bits of an integer

One can also write formulas like BIT(x, y) that means that "y is a power of 2, say 2^k, and the kth bit of the binary representation of integer x is 1"'.

This is more subtle, but possible. Indeed, this can be written:

$$(\text{POWER}_2(y) \wedge \forall q \forall r(\text{INTDIV}(x, y, q, r) \Rightarrow \text{ODD}(q))).$$

The idea is that if y satisfies the formula, then y is a power of 2, and hence in binary is written 2^k for some integer k. By dividing x by y, the remainder of the division r will be the k less significant bits of x and the quotient q the other bits of x since we have $x = q * y + r$. By testing if q is odd, one "reads" the $k+1$th bit of x, hence the bit corresponding to the bit set to 1 in the integer y encoding this position.

1.4 Principle of the proof from Gödel

Kurt Gödel proved the incompleteness theorem by building in any reasonable proof system a formula ϕ from arithmetic that states its own non-provability in the system:

$$\phi \text{ is true } \Leftrightarrow \phi \text{ is not provable.} \tag{1}$$

Every reasonable proof system is valid, and hence one must have.

$$\psi \text{ provable } \Rightarrow \psi \text{ is true.} \tag{2}$$

Then ϕ must be true, since otherwise.

$$\begin{array}{lll} \phi \text{ is wrong} & \Rightarrow \phi \text{ is provable.} & \text{(par (1))} \\ & \Rightarrow \phi \text{ is true.} & \text{(by (2))} \end{array}$$

The construction of ϕ by itself is instructive, as it captures the notion of self-reference.

We will come back to the construction from Gödel.

2 Incompleteness theorem

2.1 Principle of the proof from Turing

We will prove the incompleteness theorem by using an approach that allows to get the main consequences of the theorem, and which is due to Alan Turing.

This approach is simpler, and mainly, we have now all the ingredients to do full formal proof, by using the arguments from computability theory.

The idea is to convince ourselves that in Peano arithmetic, as well as in any "reasonable" proof system for the theory of arithmetic:

Theorem 1 *1. The set of theorems (closed formula that can be proved from the Peano axioms (or any "reasonable" axiomatisation of integers)) is computably enumerable.*

2. The set $Th(\mathbb{N})$ of closed formal F that are rue on the integers is not computably enumerable.

Consequently, the two sets cannot be the same, and the proof system cannot be complete. In other words:

Corollary 1 *There consequently exist some closed formula of $Th(\mathbb{N})$ that cannot be proved, or whose negation cannot be proved from Peano axioms or from any "reasonable" axiomatisation of the integers.*

This is the first incompleteness theorem from Kurt Gödel.

Exercise 1 *(solution on page 232) How to conciliate the previous incompleteness result (Gödel incompleteness theorem) with the completeness theorem (Gödel completeness theorem)?*

2.2 The easy direction

The set of theorems (closed formula provable from Peano axioms) is certainly computably enumerable: Whatever the proof method is (see for example those of Chapter 6), one can enumerate the theorems by enumeration the axioms and by applying in a systematic way all the reduction rules in all the possible manners, and produce as an output all the closed formula that can be derived.

This remains true as soon as we suppose that one can enumerate the axioms of the axiomatisation that one starts from. This is why, one can state that the set of theorems from any reasonable axiomatisation of the integers is recursively enumerable.

Remark 1 *In other words, if one wants a formal definition of "reasonable", one can take "computably enumerable".*

2.3 Crucial lemma

The crucial point is then to prove the following lemma.

Lemma 1 *The set $Th(\mathbb{N})$ is not computably enumerable.*

We prove this by reducing the complementary $\overline{\text{HP}}$ of the halting problem of Turing machines to this problem, i.e. by proving that $\overline{\text{HP}} \leq_m Th(\mathbb{N})$.

The theorem then follows from:

- $\overline{\text{HP}}$ is not recursively enumerable;
- and from the fact that $A \leq_m B$ and that A is not recursively enumerable, then consequently neither B.

Remember that the halting problem HP is the following problem: Given $\langle\langle M\rangle, w\rangle$, one must determine if the Turing machine M halts on input w.

Given $\langle\langle M\rangle, w\rangle$, we show how to produce a closed formula γ on the signature of arithmetic such that

$$\langle\langle M\rangle, w\rangle \in \overline{\text{HP}} \Leftrightarrow \gamma \in Th(\mathbb{N}).$$

In other words, given M and w, we must produce a closed formula γ on the signature of arithmetic that states that "the Turing machine M is not halting on input w".

This turns out to be possible since the language of arithmetic is sufficiently powerful to talk about Turing machines and the fact that they halt.

By using the principle of the previous formula BIT(y,x), we will construct a sequence of formula whose culminating point will be a formula VALCOMP$_{M,w}(y)$ that asserts that y is some integer that represents a sequence of configurations of M on input w: In other words, y represents a sequence of configurations $C_0, C_1, \cdots, C_t$ of M, encoded on a given alphabet Σ such that:

- C_0 is the initial configuration $C[w]$ of M on w;
- C_{i+1} is the successor configuration o f C_i, according to the transition function δ of the Turing machine M, for $i < t$;
- C_t is some accepting configuration

Once we will succeed to write the formula VALCOMP$_{M,w}(y)$, it will be easy to write that M is not halting on input x: The formula γ can be written as

$$\neg \exists y \text{ VALCOMP}_{M,w}(y).$$

This proves the reduction and will terminate the proof of previous lemma, and hence the proof of the theorem, recalling that $\overline{\text{HP}}$ is not recursively enumerable.

2.4 Construction of the formula

There only remain to provide the tedious details of the construction of the formula γ from M and w. Let us go.

Suppose that we encode the configurations of M on some finite alphabet Σ, that we will suppose without loss of generality of size p, with p some prime integer.

Every number has a unique representation in radix p: We will use this representation in radix p instead of the the binary representation to simplify the discussion.

Suppose that the initial configuration of M on $w = a_1 a_2 \cdots a_n$ is encoded by the integer whose digits in radix p are respectively $q_0 a_1 a_2 \cdots a_n$: We use the representation of the Definition 7.4 to represent configurations.

Consider that the blank symbol **B** is coded by digit k in radix p.

Let LEGAL the set of 6-tuples (a,b,c,d,e,f) of numbers in radix p that correspond to some legal windows for machine M: See the notion of legal window of Chapter 7. If one prefers, LEGAL is the set of 6-tuples (a,b,c,d,e,f) such that these three elements of Σ represented respectively by a,b and c appear consecutively in a configuration C_i, and if d,e,f appear consecutively in same locations in configuration C_{i+1}, then this is coherent with the transition function δ of Turing machine M.

We now define a few formulas:

- POWER$_p(x)$: "The number x is a power of p": Here p is a fixed primed number. This can be written:

$$\forall y((\text{DIV}(y,x) \wedge \text{PRIME}(y)) \Rightarrow y = p).$$

- LENGTH$_p(v,d)$: "The number d is a power of p that provides (an upper bound of) the length of v seen as a word on alphabet Σ with p letters. This can be written:
$$(\text{POWER}_p(d) \wedge v < d \wedge p * v \geq d).$$

- DIGIT$_p(v,K,b)$: "The 'kth digit of v written in radix p is b (where $K = p^k$)". This can be written:
$$\exists u \exists a (v = a + b * K + u * p * K \wedge a < K \wedge b < p).$$

- 3DIGIT$_p(v,K,b,c,d)$: "The 3 consecutive digits of v at position k are b, c and d (where $K = p^k$)". This can be written
$$\exists u \exists a (v = a + b * K + c * p * K + d * p * p * K + u * p * p * p * K \wedge a < K \wedge b < p \wedge c < p \wedge d < p).$$

- MATCH$_p(v,L,M)$: "The 3 digits of v at the position ℓ are respectively a, b and c and correspond to the 3 digits of v at the position m according to the transition function δ of the Turing machine (where $L = p^\ell$ and $M = p^m$). This can be written
$$\bigvee_{(a,b,c,d,e,f) \in \text{LEGAL}} 3\text{DIGIT}_p(v,L,a,b,c) \wedge 3\text{DIGIT}_p(v,M,d,e,f).$$

Remark 2 *We write obviously here,* $\bigwedge_{(a,b,c,d,e,f) \in \text{LEGAL}}$ *for the conjunction for each of the* 6*-tuples of* LEGAL.

- MOVE$_p(v,C,D)$: "The sequence v describe[1] a sequence of successive configurations of M of length c until d (where $C = p^c$ and $D = p^d$): All the pairs of sequences of 3-digits separated by exactly c positions in v are corresponding according to δ". This can be written as:
$$\forall y ((\text{POWER}_p(y) \wedge y * p * p * C < D) \Rightarrow \text{MATCH}_p(v, y, y * C)).$$

- START$_p(v,C)$: "The sequence v starts with the initial configuration of M on input $w = a_1 a_2 \cdots a_n$ with the addition of some blanks B until length c ($C = p^c$; $n, p^i, 0 \leq i \leq n$ are some fixed constants that are not depending of w)". This can be written:
$$\bigwedge_{i=0}^{n} \text{DIGIT}_p(v, p^i, a_i) \wedge p^n < C \wedge \forall y (\text{POWER}_p(y) \wedge p^n < y < C \Rightarrow \text{DIGIT}_p(v, y, B)).$$

- HALT$_p(v,D)$: "The sequence v has some accepting state somewhere". This can be written as:
$$\exists y (\text{POWER}_p(y) \wedge y < D \wedge \text{DIGIT}_p(v, y, q_a)).$$

[1] We see here a two-dimensional array as a unique word by putting the lines one after the other.

- VALCOMP$_{M,w}(v)$: "The sequence v is a valid computation of M on w". This can be written as:

  $$\exists c \exists d \ (\text{POWER}_p(c) \wedge c < d \wedge \text{LENGTH}_p(v,d) \wedge \text{START}_p(v,c) \wedge \text{MOVE}_p(v,c,d) \wedge \text{HALT}_p(v,d)).$$

- $\gamma_{M,w}$: "The machine M is not halting on w". This can be written as:

  $$\neg \exists v \ \text{VALCOMP}_{M,w}(v).$$

Our proof is over.

***Exercise 1** *(solution on page 232) The default of the previous constructions is that they allow to claim that there exists some true formulas which are not probable, but without providing any example of such a closed formula.*

Use the fix point theorem of computability (previous chapter) to provide explicitly a formula ψ which is not provable.

We will see later that the second theorem from Kurt Gödel allows to go further, and to prove that one can take ψ as the formula that asserts that the theory is not consistent.

(The solution of the previous formula produces a formula ψ whose practical interpretation is not clear).

3 The proof from Gödel

Kurt Gödel proved his incompleteness theorem in another manner, by constructing a closed formula that states its own non-provability. Write $\vdash$ for provable and $\models$ for true over the integers.

Suppose that we fix an encoding of formulas by the integers in any reasonable manner: If ϕ is a formula, then $\langle \phi \rangle$ denotes its encoding (an integer).

3.1 Fixpoint lemma

Here is a lemma that has been proved by Gödel, and that reads similar to the fixed point theorems already mentioned in previous chapter.

Lemma 2 (Gödel's fixpoint theorem) *For any formula $\psi(x)$ with free variable x, there is a closed formula τ such that*

$$\vdash \tau \Leftrightarrow \psi(\langle \tau \rangle),$$

i.e. the closed formula τ and $\psi(\langle \tau \rangle)$ are provably equivalent in Peano arithmetic.

Proof: Let x_0 be a fixed variable. One can certainly construct a formula SUBST(x,y,z) with free variables x,y,z which claims "the number z is the encoding of a formula

obtained by substituting the constant whose value is x in any occurrence of the free variable x_0 in the formula whose encoding is y".

For example, if $\phi(x_0)$ is a formula that contains a free occurrence of x_0, but no other free variable, the formula SUBST$(7, \langle\phi(x_0)\rangle, 312)$ is true if $312 = \langle\phi(7)\rangle$.

We will not provide the details of the construction of the formula SUBST, but the idea is to observe that this is indeed possible, by using for example the idea of relation BIT(x, y).

One considers now $\sigma(x)$ defined by $\forall y$ (SUBST$(x, x, y) \Rightarrow \psi(y)$), and τ defined by $\sigma(\langle\sigma(x_0)\rangle)$.

Then τ is the desired solution, since

$$\begin{array}{rcl} \tau & = & \sigma(\langle\sigma(x_0)\rangle) \\ & = & \forall y\ (\text{SUBST}(\langle\sigma(x_0)\rangle, \langle\sigma(x_0)\rangle, y) \Rightarrow \psi(y)) \\ & \Leftrightarrow & \forall y\ y = \langle\sigma(\langle\sigma(x_0)\rangle)\rangle \Rightarrow \psi(y) \\ & \Leftrightarrow & \forall y\ y = \langle\tau\rangle \Rightarrow \psi(y) \\ & \Leftrightarrow & \psi(\langle\tau\rangle) \end{array}$$

Of course, we have used here some informal equivalences, but the argument can indeed be fully formalized in Peano arithmetic. □

3.2 Arguments from Gödel

We observe now that the language of arithmetic is sufficiently expressive to talk about provability in Peano arithmetic. In particular, it is possible to code a sequence of formulas by an integer and to write a formula PROOF(x, y) that means that the sequence of formulas whose encoding is x is a proof of the formula whose encoding is y.

In other words, $\vdash$ PROOF$(\langle\pi\rangle, \langle\psi\rangle) \Leftrightarrow \pi$ is a proof of ψ in Peano arithmetic.

The provability in Peano arithmetic can hence be coded by the formula PROVABLE(y) defined by $\exists x$ PROOF(x, y).

Then for any closed formula ϕ,

$$\vdash \phi \Leftrightarrow \ \models PROVABLE(\langle\phi\rangle). \tag{3}$$

We then have

$$\vdash \phi \Leftrightarrow \ \vdash PROVABLE(\langle\phi\rangle). \tag{4}$$

The direction $\Rightarrow$ is true since if ϕ is provable then there is a proof π of ϕ. The arithmetic of Peano and the proof system allow to use this proof to prove ϕ (i.e. that PROOF$(\langle\pi\rangle, \langle\phi\rangle)$). The direction $\Leftarrow$ follows from 3 and from the validity of proof in Peano arithmetic.

Let us then use the point fix lemma to the closed formula $\neg$PROVABLE(x). We then obtain a closed formula ρ that states its own non-provability:

$$\vdash \rho \Leftrightarrow \neg \text{ PROVABLE}(\langle\rho\rangle),$$

in other words, ρ is true if and only if it is not provable in Peano arithmetic.

From the validity of proof in Peano arithmetic, we have

$$\models \rho \Leftrightarrow \neg \text{ PROVABLE}(\langle\rho\rangle). \tag{5}$$

Then formula ρ must be true, since otherwise then

$$\begin{array}{rcll} \models \neg\rho & \Rightarrow & \text{PROVABLE}(\langle\rho\rangle) & \text{(by 5)} \\ & \Rightarrow & \vdash \rho & \text{(by 3)} \\ & \Rightarrow & \models \rho & \text{(by validity of Peano arithmetic)} \end{array}$$

a contradiction.

So $\models \rho$. But now,

$$\begin{array}{rcll} \models \rho & \Rightarrow & \neg\,\text{PROVABLE}(\langle\rho\rangle) & \text{(by 5)} \\ & \Rightarrow & \not\models \rho & \text{(by definition of truth)} \\ & \Rightarrow & \not\vdash \rho & \text{(by 3)} \end{array}$$

Hence ρ is true, but cannot be proved.

3.3 Second incompleteness theorem from Kurt Gödel

The default of the previous proof is of course that it does not really make sense to formula ρ.

The second incompleteness theorem from Kurt Gödel provides an explicit example of a formula that cannot be proved.

One can express a formula CONSIST that expresses the fact that the theory is consistent. Basically, one writes that its is not possible to prove a formula F and its negations: It is "sufficient"' to write $\neg\exists x(\text{PROVABLE}(x) \wedge \text{PROVABLE}(y) \wedge NEG(x,y))$, where $NEG(x,y)$ means that y is the encoding of the negation of the formula encoded by x.

The second incompleteness theorem from Kurt Gödel allows to prove that this precise formula cannot be proved.

In other words:

Theorem 2 (Second incompleteness theorem from Kurt Gödel) *No deduction system can prove its own consistency.*

We will not go into further details.

4 Bibliographic notes

Suggested readings To go further with the notions of this chapter, we suggest the reading of the last chapters of the book [Kozen, 1997], which remain short and direct, or of the book [Cori & Lascar, 1993] for a complete proof.

Bibliography This chapter is taken from one of the last three chapters of the excellent book [Kozen, 1997].

Index

Bibliography

[Cori & Lascar, 1993] Cori, R. & Lascar, D. (1993). *Logique Mathématique, volume II*. Masson.

[Kozen, 1997] Kozen, D. (1997). *Automata and computability*. Springer Verlag. https://doi.org/10.1007/978-1-4612-1844-9