Foundations of Computer Science
Logic, models, and computations

Chapter: Solutions of some exercises

Course CSC_41012_EP

of l'Ecole Polytechnique

Olivier Bournez

bournez@lix.polytechnique.fr

Version of July 11, 2025

Solutions of some exercises

We will find in this chapter the correction of some of the exercises. The reader is invited to sent to bournez@lix.polytechnique.fr a redaction of the solution [1] of any exercise that is not corrected in these pages, or any more elegant solution to the presented solutions.

Chapter 1

Exercice 1.3 (page 11). We can write $A \cap B^c = (A \cap A^c) \cup (A \cap B^c) = A \cap (A^c \cup B^c) = A \cap (A \cap B)^c$. Similarly $A \cap C^c = A \cap (A \cap C)^c$. So $A \cap B = A \cap C$ implies $A \cap B^c = A \cap C^c$.

Since $(X^c)^c = X$ for any subset X of E, we deduce that $A \cap B^c = A \cap C^c$ implies $A \cap (B^c)^c = A \cap (C^c)^c$ implies $A \cap B = A \cap C$.

Exercice 1.5 (page 15). The function $f(x, y) = y + (0 + 1 + 2 + \cdots + (x + y))$ enumerates the elements of $\mathbb{N}^2$ as in the figure.

The function f is indeed bijective: Let $n \in \mathbb{N}$. There exists a unique $a \in \mathbb{N}$ such that $0 + 1 + \cdots + a \leq n < 0 + 1 + \cdots + (a + 1)$. The unique antecedent n by f is given by (x, y) with $y = n - (0 + 1 + \cdots + a)$ and $x = a - y$.

Another bijection between $\mathbb{N}^2$ and $\mathbb{N}$ is given by the function $g : \mathbb{N}^2 \to \mathbb{N}$ defined by $g(x, y) = 2^x(2y + 1) - 1$. The fact that this is indeed a bijection comes from the fact that any strictly positive integer is the produce of a power of two and some odd number.

Chapter 2

Exercice 2.3 (page 18). The proof is by contradiction. Consider $X = \{k \in \mathbb{N} | P(k) \text{ is false}\}$. If X is non empty, it admits a least element n.

We cannot have $n \neq 0$. Indeed, one knows that for $n = 0$, by supposing for any integer $k < 0$ the property $P(k)$ we deduce $P(0)$, since there is no $k < 0$, that means that we can deduce $P(0)$ without any hypothesis.

This states: $P(n - 1), P(n - 2), \ldots, P(0)$ must be true from definition of X. We obtain a contradiction with the property applied in n.

[1] If possible using LaTeX.

Exercice 2.4 (page 19). We must first be convinced that $L^*.M$ is solution of equation $X = L.X \cup M$.

For $M = \{\epsilon\}$, this is equivalent to state that $L^* = L.L^* \cup \{\epsilon\}$. This follows from

$$L^* = \bigcup_{n\in\mathbb{N}} L^n = L^0 \cup (\bigcup_{n\geq 1} L^n) = \{\epsilon\} \cup (\bigcup_{n\geq 0} L.L^n) = \{\epsilon\} \cup L.L^*.$$

We deduce for general M that $L^*.M = L.L^*.M \cup \{\epsilon\}.M$, and so that $L^*.M$ is indeed solution of equation $X = L.X \cup M$.

There remains to prove that this is the unique solution. Let $X \subset \Sigma^*$ satisfying $X = L.X \cup M$.

To prove that $L^*.M \subset X$, it suffices to prove that for any integer n we have $L^n.M \subset X$, since $L^*.M = \bigcup_{n\geq 0} L^n.M$. We prove by recurrence on n the property $P(n)$: $L^n.M \subset X$.

$P(0)$ is true since $L^0.M = \{\epsilon\}.M = M \subset M \cup L.X = X$.

Suppose $P(n)$ true. We have $L^{n+1}.M = L.L^n.M \subset L.X \subset L.X \cup M = X$. So $P(n+1)$ is true.

Conversely, we prove by recurrence the property $Q(n)$: Every word w of X of length n belongs to $L^*.M$. This clearly provides the other inclusion.

For this, we use the second induction principle. Suppose that for all $k < n$, $Q(k)$ is true. Let $w \in X$ be a word of length n. Since $X = L.X \cup M$, two cases must be considered. Let $w \in M$ and we have directly $w \in L^*.M$ since $M \subset L^*.M$.

Let $w \in L.X$ and we can write $w = u.v$ with $u \in L$, and $v \in X$. Since $\epsilon \not\in L$, the length of u is non-null and hence the length of v is strictly less than the one of w. By induction hypothesis, we have $v \in L^*.M$. So $w = u.v \in L.L^*.M \subset L^*.M$, which proves $Q(n)$, and terminates the demonstration.

Exercice 2.5 (page 22). The language L of the well parentheses expressions formed with identifiers taken in a set A and from the operators $+$ and $\times$ is the subset of $E = (A \cup \{+, \times\} \cup \{(,)\})^*$ defined inductively by

- (B) $A \subset L$;
- (I) $e, f \in L \Rightarrow (e + f) \in L$;
- (I) $e, f \in L \Rightarrow (e \times f) \in L$.

Exercice 2.6 (page 24). The proof is similar to the proof of Theorem 2.28 (which is a generalisation of this phenomenon).

Exercice 2.7 (page 27). Let $P(x)$ be the property: "x is non-empty and without any vertex with a unique non-empty son". Clearly $P(x)$ is true for $x = (\emptyset, a, \emptyset)$. If we suppose $P(g)$ and $P(d)$, for $g, d \in ABS$, clearly $P(x)$ is also true for $x = (g, a, d)$. The first property is hence satisfied.

Let $P(x)$ be the property $n(x) = 2f(x) - 1$. We have $P(x)$ for $x = (\emptyset, a, \emptyset)$, since $n(x) = 1$, and $f(x) = 1$, and $1 = 2 * 1 - 1$.

Suppose $P(g)$ and $P(d)$ for $g, d \in ABS$. Consider $x = (g, a, d)$. We have $n(x) = 1 + n(g) + n(d) = 1 + 2 * f(g) - 1 + 2 * f(d) - 1 = 2 * (f(g) + f(d)) - 1 = 2 * f(x) - 1$.

The property is hence true for every $x \in ABS$.

Chapter 3

Exercice 3.17 (page 47). It is clear that the second property implies the first: For any truth value distribution v, we have $\overline{v}(G) = 1$ if G is a tautology, and $\overline{v}(F) = 0$ if $\neg F$ is one. In both cases $\overline{v}((F \Rightarrow G)) = 1$.

Suppose now that the second property is wrong. We can choose some distribution of truth values v such that $\overline{v}(\neg F) = 0$, and some truth value distribution v' such that $\overline{v}'(G) = 0$.

We define a distribution of truth value v'', by letting for each propositional variable x, $v''(x) = v(x)$ if x has at least one occurrence in F and $v''(x) = v'(x)$ if x has no occurrence in F. By construction, this distribution of truth values coincide with v on F and v' over G. We deduce that $\overline{v}''(F) = \overline{v}(F) = 0$ and $\overline{v}''(G) = \overline{v}'(G) = 0$. And hence $\overline{v}''((F \Rightarrow G)) = 0$. The first property is hence necessarily wrong.

Exercice 3.18 (page 47). It is clear that if a graph is colorable with k colors, each of its subgraph is colorable with k colors (the same). The difficulty is in proving the converse direction.

We introduce for each pair $(u,i) \in V \times \{1,2,\ldots,k\}$ some propositional variable $A_{u,i}$. We construct a set Γ of formulas of propositional calculus over the set of variables $A_{u,i}$ that is satisfiable if and only if G is k-colorable. The idea is that $A_{u,i}$ encodes the fact that the vertex u is colored with the color i. The set Γ is defined as $\Gamma = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3$, where each of the Γ_i expresses a particular constraint.

- For Γ_1: Every vertex has a color:

$$\Gamma_1 = \{A_{u,1} \vee \cdots \vee A_{u,k} | u \in V\}.$$

- For Γ_2: Every vertex has at most one color:

$$\Gamma_2 = \{\neg(A_{u,i} \wedge A_{u,j}) | u \in V, 1 \leq i,j \leq k, i \neq j\}.$$

- For Γ_3: Each edge has not its extremities of the same color:

$$\Gamma_3 = \{\neg(A_{u,i} \wedge A_{v,i}) | u \in V, 1 \leq i \leq k, (u,v) \in E\}.$$

Doing so, a graph is colorable with k colors if and only if one can satisfy all the formulas of $\Gamma = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3$.

We will then use the compactness theorem: Let Γ_0 be some finite subset of Γ. Let $V_0 = \{u_1, \cdots, u_n\}$ be the vertices u such that $A_{u,i}$ appears among the formulas of Γ_0. Let $G_0 = (V_0, E_0)$ be the subgraph determined by V_0.

If we we suppose that we have a graph such all the subgraphs are colorable with k colors, in particular this must hold for G_0, and so Γ_0, that is a subset of the constraints expression the fact that Γ_0 is k-colorable, is satisfiable.

Since Γ has all its finite subsets satisfiable, by the compactness theorem, Γ is hence satisfiable. This means that G is hence satisfiable. This means that G is k-colorable, since G is k-colorable if and only if Γ is satisfiable.

Chapter 4

Exercice 4.1 (page 52). We write

- $F_1 : ((F \Rightarrow ((F \Rightarrow F) \Rightarrow F)) \Rightarrow (((F \Rightarrow (F \Rightarrow F)) \Rightarrow (F \Rightarrow F)))$ (instance of axiom 2.)
- $F_2 : ((F \Rightarrow ((F \Rightarrow F) \Rightarrow F))$ (instance of axiom 1.);
- $F_3 : ((F \Rightarrow (F \Rightarrow F)) \Rightarrow (F \Rightarrow F))$ (modus ponens from F_1 and F_2);
- $F_4 : (F \Rightarrow (F \Rightarrow F))$ (instance of axiom 1.);
- $F_5 : (F \Rightarrow F)$ (modus ponens from F_3 and F_4).

Exercice 4.2 (page 52). One direction is easy: If $F_1, F_2, \ldots, F_n$ is a proof of $F \Rightarrow G$ from T, then $F_1, F_2, \ldots, F_n, F, G$ is a proof from G from $T \cup \{F\}$.

Conversely, we prove by recurrence on n that if there exists a proof of length n of G from $T \cup \{F\}$, then there exists a proof of $(F \Rightarrow G)$ from T.

By recurrence hypothesis, there exists a proof from T of each of the formulas $(F \Rightarrow F_1)$, $\ldots$, $(F \Rightarrow F_n)$, this applying by default to the case $n = 1$. Consider the formula F_n that is to say G. Three cases are possible:

1. The formula G is an axiom or a formula of T. We then have $T \vdash G$. We know that $(G \Rightarrow (F \Rightarrow G))$ is an instance of axiom 1., hence we have $\vdash (G \Rightarrow (F \Rightarrow G))$, and hence also $T \vdash (G \Rightarrow (F \Rightarrow G))$. By modus ponens, we have $T \vdash (F \Rightarrow G)$.
2. G is the formula F. The correction of the previous exercise provides a proof of $(F \Rightarrow F)$.
3. There exists $i, j < n$ such that H_j is a formula $(H_i \Rightarrow G)$. By recurrence hypothesis, we have $T \vdash (F \Rightarrow H_i)$ and $T \vdash (F \Rightarrow (H_i \Rightarrow G))$. Then the sequence:
   - $((F \Rightarrow (H_i \Rightarrow G)) \Rightarrow ((F \Rightarrow H_i) \Rightarrow (F \Rightarrow G)))$ (instance of axiom 2.);
   - $((F \Rightarrow H_i) \Rightarrow (F \Rightarrow G))$ (modus ponens)
   - $(F \Rightarrow G)$ (modus ponens)

   is a proof of $(F \Rightarrow G)$ from $(F \Rightarrow H_i)$ and $(F \Rightarrow (H_i \Rightarrow G))$. By concatenating this proof of $(F \Rightarrow H_i)$ and of $(F \Rightarrow (H_i \Rightarrow G))$ from T, we obtain a proof of $F \Rightarrow G$ from T.

Exercice 4.3 (page 52). For the first assertion: Suppose that $T \cup \{F\} \vdash G$. By the deduction theorem (Exercise 4.2), we have $T \vdash (F \Rightarrow G)$. But the formula $((F \Rightarrow G) \Rightarrow (\neg G \Rightarrow \neg F))$ is an instance of axiom 5, so by modus ponens, we obtain $T \vdash (\neg G \Rightarrow \neg F)$, so by the deduction theorem again $T \cup \{\neg G\} \vdash \neg F$.

Suppose now $T \cup \{\neg G\} \vdash \neg F$. By what is above, we obtain $T \cup \{\neg\neg F\} \vdash \neg\neg G$. Since $\neg\neg G \Rightarrow G$ is an instance of axiom 4., by modus ponens, we deduce $T \cup \{\neg\neg F\} \vdash G$.

But $(F \Rightarrow \neg\neg F)$ is an instance of axiom 3. From this, we deduce from any proof of a formula from $T \cup \{\neg\neg F\}$ a proof of the same formula from $T \cup \{F\}$, and we finally obtain $T \cup \{F\} \vdash G$.

For the second assertion: We have $\{\neg F, \neg G\} \vdash \neg F$ by definition, so by the first assertion, $\{\neg F, F\} \vdash G$, and from there $T \vdash G$ if both F and $\neg F$ are provable from T.

Exercice 4.4 (page 52). We have $\{\neg G, \neg G \Rightarrow G\} \vdash G$. By using Exercice 4.3, item 1, this is equivalent to say $\{\neg G, \neg G\} \vdash \neg(\neg G \Rightarrow G)$. Hence $\{\neg G\} \vdash \neg(\neg G \Rightarrow G)$. By using Exercice 4.3, item 1 in the reverse direction, $\{\neg G \Rightarrow G\} \vdash G$.

Exercice 4.5 (page 52). Suppose $T \cup \{F\} \vdash G$ and $T \cup \{\neg F\} \vdash G$. By applying the first assertion of Exercise 4.3, we obtain $T \cup \{\neg G\} \vdash \neg F$ and $T \cup \{\neg G\} \vdash F$. By the second assertion of Exercise 4.3, we obtain that any formula is provable from $T \cup \{\neg G\}$, and in particular $T \cup \{\neg G\} \vdash G$. By the deduction theorem (Exercise 4.2), $T \vdash (\neg G \Rightarrow G)$. It suffices then to use Exercise 4.4 in order to deduce $T \vdash G$.

Chapter 5

Exercice 5.1 (page 70). Only the last word corresponds to a formula: In the first and in the second, the arity of R_2 is not respected. In the third the quantification $\exists R$ is not on a variable but on a relation symbol. This is what is called a second order formula, and this is not considered as valid formula in the previous definition (i.e. in this document).

Exercice 5.2 (page 71). There is no free variable nor free occurrence in the first formula. In the second formula, the variable x is free: Its first occurrence is bound, its second occurrence is free.

Exercice 5.4 (page 75). To avoid too heavy notations, we will write xRy pour $R(x, y)$, and we authorize ourselves not to write all the parentheses. It suffices to consider

$$(\forall x xRx) \wedge (\forall x \forall y (xRy \wedge yRx \Rightarrow x = y)) \wedge (\forall x \forall y \forall z (xRy \wedge yRz \Rightarrow xRz)).$$

Exercice 5.7 (page 78). We will only indicate here by some arrow the implications. For example $\Rightarrow$ means that the left member implies the right member.

1. $\Leftrightarrow$
2. $\Leftrightarrow$
3. $\Rightarrow$
4. $\Leftrightarrow$
5. $\Rightarrow$

6. $\Rightarrow$

Exercice 5.8 (page 80). Here are some equivalent prenex forms.

$$\exists x \forall x' \forall y (P(x) \wedge (Q(y) \Rightarrow R(x')))$$

$$\forall x \forall y \exists x' (P(x') \wedge (Q(y) \Rightarrow R(x)))$$

$$\forall x \exists x' \forall y (P(x') \wedge (Q(y) \Rightarrow R(x)))$$

Chapter 6

Exercice 6.1 (page 86). The third item of Definition 6.12 is true for any relation symbol R and in particular for the symbol $=$ of arity 2: In particular, we have $\forall x_1 \forall x_1' \forall x_2 (x_1 = x_1' \Rightarrow (x_1 = x_2 \Rightarrow x_1' = x_2)$. Since we have $x = x$ by the first item of Definition 6.12, if we have $x = y$ then we have $y = x$ by applying the case where x_1, x_2, x_1' are respectively x, x and y.

Exercice 6.5 (page 89). Clearly the last assertion follows from the second, since the first produces a model that cannot be the standard model of the integers that satisfies the axioms of Robinson. Indeed, in the standard model of the integers (in the integers) the addition is commutative.

For the first assertion, it suffices to prove the property by recurrence over n. It is true for $n = 0$ by axiom $\forall x\, \mathbf{0} + x = x$, applied in $x = s^m(\mathbf{0})$. Suppose the property true at rank $n-1 \geq 0$: We have $s^n(\mathbf{0}) + s^m(\mathbf{0}) = s(s^{n-1}(\mathbf{0})) + s^m(\mathbf{0})$, which according to axiom $\forall x \forall y\, s(x) + y = s(x+y)$ applied for $x = s^{n-1}(\mathbf{0})$ and $y = s^m(\mathbf{0})$ values $s(s^{n-1}(\mathbf{0}) + s^m(\mathbf{0}))$ so $s(s^{n+m-1}(\mathbf{0}))$ by recurrence hypothesis, in other words, $s^{n+m-1+1}(\mathbf{0}) = s^{n+m}(\mathbf{0})$.

For the second assertion, we must consequently construct a model whose base set contains something else than (only) the elements $s^{(n)}(\mathbf{0})$ for some integer n. Here is a way to proceed. One considers a set X with at least two elements.

We consider the structure $\mathfrak{M}$ whose base set is

$$M = \mathbb{N} \cup (X \times \mathbb{Z}),$$

and where the symbols $s, +, *, =$ are interpreted by the following conditions:

- $=$ is interpreted by equality. $s, +, *$ extends the corresponding functions over $\mathbb{N}$;
- for $a = (x, n)$:
  - $s(a) = (x, n+1)$;
  - $a + m = m + a = (x, n+m)$;
  - $a * m = (x, n * m)$ if $m \neq 0$, and $(x, n) * 0 = 0$;
  - $m * a = (x, m * n)$;

- for $a = (x, n)$ and $b = (y, m)$:
  - $(x, n) + (y, m) = (x, n + m)$;
  - $(x, n) * (y, m) = (x, n * m)$.

(in these definitions, $\mathbb{N}$ and $\mathbb{Z}$ denotes the usual (standard) sets). One checks that this structure satisfies all the axioms of Robinson, and that the addition is not commutative.

Exercice 6.8 (page 90). We will only provide here a scheme of the proof . We prove successively that

- $\forall v(v + \mathbf{0} = v)$
- $\forall v \forall v' v + s(v') = s(v + v')$
- $\forall v(v + \mathbf{1} = s(v))$ where 1 denotes $s(\mathbf{0})$
- $\forall v \forall v'(v + v' = v' + v)$

For example, $\forall v(v + \mathbf{0} = v)$ is proved by observing that $\mathbf{0} + \mathbf{0} = \mathbf{0}$ and that $\forall v((v + \mathbf{0} = v) \Rightarrow (s(v) + \mathbf{0} = s(v))$. We use the scheme of Peano axioms in the case where the formula F is the formula $v + \mathbf{0} = v$ to deduce $\forall v(v + \mathbf{0} = \mathbf{0})$.

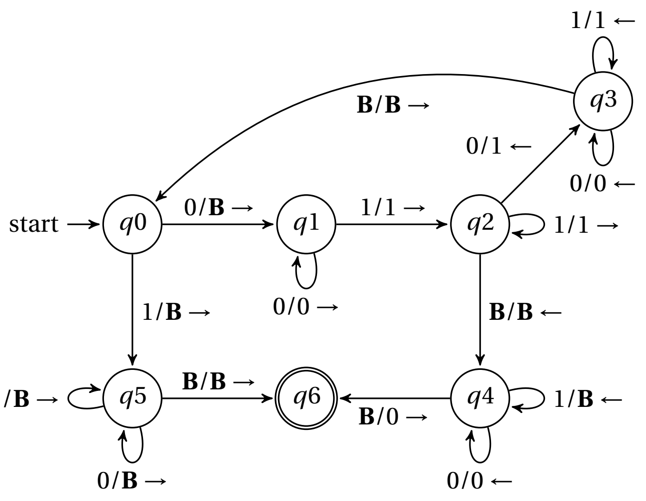
Exercice 6.12 (page 99). Consider a new constant symbol c. Add this constant symbol to the signature of Peano axioms. Consider $\mathcal{T}$ defined as the union of Peano axioms of the formulas $\neg c = s^n(\mathbf{0})$, for some integer n. Every finite subset of $\mathcal{T}$ has a model as it is included in the union of Peano axioms, and of the formulas $\neg c = s^n(\mathbf{0})$ for $1 \leq n \leq N$ for some integer N: It suffices to observe that if one interprets c by $N + 1$, one then obtain a model of this finite subset.

By compactness theorem, $\mathcal{T}$ has a model. This model must satisfy $\neg c = s^n(\mathbf{0})$ for every integer n. The interpretation of c is hence not a standard integer. The model is hence non-standard: It contains some "integers" that are not standard.

Exercice 6.13 (page 100). Suppose that $\mathcal{T}$ is a theory at most countable that has a model. It is coherent: Corollary 6.45 applies. Observe that the proof of Corollary 6.45 (actually the proof of completeness theorem) consists at the end in constructing a model $\mathfrak{M}$ of $\mathcal{T}$ whose domain is made of terms over the signature. Since the set of terms over a countable signature is countable, the model $\mathfrak{M}$ that is built is indeed a model of $\mathcal{T}$.

Chapter 7

Exercice 7.2 (page 110). Here is a solution: One considers a machine on the set of states $Q = \{q0, q1, q2, \cdots, q6\}$ with $\Gamma = \{0, 1, \mathbf{B}\}$.



The machine is built in order to do the following work: It seeks the 0 at the leftmost position, and it replaces it by a blank. It searches then on the right a 1, and when it finds one, it continues on the right until it finds a 0, that it replaces by a 1. The machine then returns left in order to find the 0 at the most left position that it identifies by finding the first blank on the left, and then moving one cell right from this blank.

The process is repeated until:

- either when searching on right a 0, one meets a blank. Then the n 0's in 0^m10^n have been changed into 1's and $n+1$ of the m symbols 0 have been changed into **B**. In that case, the machine replaces the $n+1$ symbols 1 by a 0 and n blanks, which remains $m-n$ symbols 0 on the tape. Since in this case, $m \geq n$, $m \ominus n = m - n$.

- or in restarting the cycle, the machine does not succeed to find a 0 to be changed in a blank, since the m first 0's have already been changed in **B**. Then $n \geq m$, and so $m \ominus n = 0$. The machine then replaces every 1 and 0 remaining by some blanks and halts with a tape completely blank.

Chapter 9

Exercice 9.1 (page 136). A is decidable.

Indeed: Suppose that s values 0. In that case, A is recognized by the Turing machine that compares the letter in front of the head to 0 and accepts if it values 0 and rejects otherwise.

Suppose that s values 1. In that case, A is recognized by the Turing machine that compares the letter in front of the head to 0 and accepts if it values 1 and rejects otherwise.

In any case, A is decided by a Turing machine.

Exercice 9.2 (page 148). This is a direct application of Rice theorem.

Exercice 9.3 (page 155). Suppose that $S \subset \mathbb{N}$ is decidable. The function χ is computable, since it is sufficient on input n to determine if $n \in S$ and to return 1 if this is the case, 0 otherwise.

Conversely, if χ is computable, then S is decidable: On some input n, one computes χ and one accepts (respectively: rejects) if $\chi(n) = 1$ (resp. $\chi(n) = 0$).

Suppose that $S \subset \mathbb{N}$ is semi-decidable. The function is computable, since it is sufficient on input n to simulate the machine that computes the function, and accepts if this simulation halts.

Conversely, if the function is computable, S is semi-decidable: On some input n, one simulates the computation of the function, and one accepts if the simulation accepts.

Exercice 9.1 (page 155). Let H be the computable function defined by: If t is some unary program (a Turing machine) of A, then $H(\langle t\rangle, n)$ provides the result of t on n, otherwise $H(\langle t\rangle, n) = 0$. By construction, H is some interpreter for all the unary programs of A. The function H is total.

We prove that the total computable function $H'(n) = H(n,n)+1$ is not in A: Indeed, otherwise there would be a t in A that computes H'. We would have for all n, $H(\langle t\rangle n)$ that would be the result of t on n, so $H'(n) = H(n,n)+1$. In particular, $H(\langle t\rangle, \langle t\rangle) = H(\langle t\rangle, \langle t\rangle)+1$. Contradiction.

This result implies the undecidability of the halting problem.

- Indeed, suppose that $H(A,n)$ decides of the halting of the Turing machine A on input n. For all unary program f, let f' be the unary program that always halts defined by $f'(n) = f(n)$ if $Ha(f,n)$, 0 otherwise.

- Let A defined as the set of coding of Turing machines of the form if $H(f,n)$ then $f(n)$ otherwise 0.

- For any unary program f that always halt, f' and f computes the same function.

- Any total unary function is represented by some Turing machine A. This is in contradiction with the previous result.

Exercice 9.4 (page 155). To test whether $x \in E$, one enumerates the elements of E until one finds x, in which case one accepts, or an element greater than x, in which case one rejects.

Exercice 9.5 (page 155). To extract an infinite decidable set from an infinite computably enumerable set, it is sufficient to extract a subsequence strictly increasing from the sequence of the $f(n)$: one starts with $y = max = 0$. For $n = 0, 1, \ldots,$

- One computes $y = f(n)$.
- If $y > max$ the one sets $max := y$, and one prints $f(n)$

Exercice 9.6 (page 155). For $n = 0, 1, \dots$, one tests if n is in the set, and if so, one prints it.

Exercice 9.7 (page 156). To test if $x \in \exists A$, it is sufficient to test for $y = 0, 1, \dots$ if $(x, y) \in A$. One halts as soon as one find such a y.

Every computably enumerable language is enumerated by some computable function f. It is then to the projection of the set $A = \{(f(n), n) | n \in \mathbb{N}\}$. This set A is indeed decidable.

Exercice 9.10 (page 157). Their uncomputability comes from Rice theorem.

The first is computably enumerable: One enumerates the triples (a, b, t) with a and b two distinct words, t some integer, and for each of them one tests if A accepts a and b in time t. If yest, one accepts.

The second is not computably enumerable since its complement is computably enumerable: One enumerates the pairs (a, t) and one tests if a is accepted by A in time t. If so, one accepts.

Chapter 10

Exercice 10.1 (page 162). Let $\mathbb{N}$ be the standard model of the integers. $Th(\mathbb{N})$ corresponds to the closed formula F true in $\mathbb{N}$.

The incompleteness theorem states that exist some closed formula true of $Th(\mathbb{N})$ which are not provable, nor their negation from Peano axioms, or from any "reasonable" axiomatisation of the integers.

Let F be such a closed formula. Suppose without loss of generality that F is satisfied on $\mathbb{N}$.

The incompleteness theorem states that non-provable formulas are exactly those that are true in all the models. This simply means that there must exist some other models than $\mathbb{N}$ to Peano axioms: In particular, there must exist a model where F is not satisfied.

Repeated in another way, there is a model of Peano axiom with F satisfied (for example $\mathbb{N}$) and another mode where F is not satisfied.

The completeness theorem remains compatible since F is not satisfied in all the models.

Exercice 10.1 (page 165). We consider the Turing machine S that does the following:

- on every input w
  - obtain by the recursion theorem its own description $\langle S \rangle$.

- construct the formula $\psi = \gamma_{S,\epsilon}$ (the formula γ of the proof of the course for S)
- enumerate the provable formulas as long as $\gamma_{S,\epsilon}$ is not produced.
- if previous steps eventual terminates, then accepts.

The formula $\psi = \gamma_{S,\epsilon}$ of the second step is not provable: Indeed ψ is true if an only if S does not accept the empty word, because:

- If S finds a proof of ψ, then S accepts the empty word, and hence formula ψ is wrong:
- (If arithmetic is coherent one cannot prove some wrong formula, and hence) This case cannot happen.

Now observe that if S does not find a proof of ψ, then S is not accepting the empty word: Consequently, ψ is true, but is not provable!

In short: ψ is true, but is not provable.

Chapter 11

Exercice 11.1 (page 176). We will use the fact that the limit exists and is positive to prove that $f(n) = \mathcal{O}\left(g(n)\right)$ and $f(n) = \Omega(g(n))$, according to the definition of Θ.

Since

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c > 0,$$

from the definition of a limit, there is a rank n_0 from which the ratio is between $\frac{1}{2}c$ and $2c$. So $f(n) \leq 2cg(n)$ and $f(n) \geq \frac{1}{2}cg(n)$ for all $n \geq n_0$, which proves exactly what is required.

Exercice 11.3 (page 177). The computation of the maximum of n numbers can be done in linear time (See course).

A sorting algorithm (one takes n numbers in input and must produce as output the same numbers but in increasing order) such as the *merge sort* works in time $\emptyset(n \log n)$: The merge sort consists, in order to sort n numbers, in splitting the set in two subsets of the same time (up to 1), sort recursively each subset, and then merge the two results. The merging of two sorted list can be solved in a time linear in the sum of the lengths of the two lists. Indeed, merging in that case consists in repeating the following operation as long as it is possible: Write the least element of the two lists,, and then delete this element. This gives a global complexity for merge sort that satisfies a recursive equation of type $T_n = \mathcal{O}(2 * T_{n/2}) + \mathcal{O}(n)$ which can be proved to providing a complexity of $On \log n$.

Suppose that n sets S_1, S_2, ..., S_n are given, each of them being a subset of $\{1, 2, \ldots, n\}$, and that one wants to know if there is a disjoint pair among these sets. This can be solved in cubic time: It is sufficient to go through all the pairs S_i and S_j, and for each pair to scan the elements of S_i to know if there are in S_j.

The following chapters contain mainly problems for which no polynomial solutions is found: The presented algorithms are not polynomial.

Exercice 11.4 (page 177). The time is respectively for (a),

- multiplied by 4;
- multiplied by 8;
- multiplied by 400;
- multiplied by2 + 1;
- squared.

And for (b):

- increases of $2n+1$;
- increases of $3n^2+3n+1$;
- increases of $200n+100$;
- transformed into $(n+1)\log(n+1)$, hence essentially increased of $\mathcal{O}\left(n\log n\right)$;
- multiplied by 2.

Chapter 12

Exercice 12.4 (page 201). If P = NP then, since the complement of P is P (it is sufficient to inverse the accepting state and the rejecting state of a machine), we must have NP equal to its complement. Hence, we cannot have NP which is not equal to its complement.

Exercice ?? (page ??). This exercise is among the corrected exercises in [Kozen, 2006].

Concerning first item, this is incorrect because we have not shown how to produce x deterministically when it exists.

For second item: The data of x corresponds to a valid certificate that can be checked in polynomial time.

For the third item. Suppose that P = NP. Hence B is in P. Using this fact, given y of length n we can do a binary search on strings of length n to find x such that $f(x) = y$. Indeed, first ask whether $(y,\epsilon) \in B$?. If not, then no such x exists; halt an report failure. If so, ask, whether $(y,0) \in B$. If yes, there is an x with $f(x) = y$ whose first bit is 0, and if no, all such x have first bit 1. Now, depending on the previous answer, ask whether $(y,00) \in B$ or $(y,10) \in B$ as appropriate. The answer determines the second bit of x. Continue in this fashion until all the bits of some x with $f(x) = y$ have been determined.

For the fourth and fifth item: By definition.

Concerning sixth item: First determine whether the input is of the form $\phi\#t$, and if so, evaluate ϕ on t. If f is invertible, then P = NP, because ϕ is satisfiable if and only if there exists x such that $f(x) = \phi\#1^{|t|}$. We already proved the reverse direction.

The last assertion is then immediate.

Chapter 13

Exercice 13.3 (page 213).

- The problem is in NP since given some seating plan, one can check in polynomial time if every knight is not sitting close to one of its enemies.
- We will do the reduction from the problem HAMILTONIAN CIRCUIT.

  Let $\mathscr{I} =< G = (V,E) >$ be an instance of HAMILTONIAN CIRCUIT.

  We transform this instance into an instance $\mathscr{I}_2$ of the problem KNIGHTS OF THE ROUND TABLE in the following way:

  - Each vertex of the graph is a knight;
  - Two knights are enemies if and only there is no edge in G involving the two vertices represented by these two knights.

  This transformation can be done in polynomial time (we have constructed actually the complementary of graph G).

  It is easy to prove that:

  - If there exists some Hamiltonian cycle in G, then there is a seating plan. It suffices to see that an edge in G encodes the fact that two knights are not enemies. So the seating plan corresponds to a Hamiltonian cycle.
  - If there exists a seating plan, then there exists some Hamiltonian cycle in G.

  So the problem KNIGHTS OF THE ROUND TABLE is at least as hard as the Hamiltonian cycle problem.

Hence, the problem KNIGHTS OF THE ROUND TABLE is NP-complete.

Exercice 13.4 (page 213).

- The problem HAMILTONIAN CIRCUIT is in NP since given a path, one can check in in polynomial time if it crosses exactly once every vertex of the graph and if it has u and v as extremities.
- We are going now to do a reduction from the problem HAMILTONIAN CYCLE. Let $\mathscr{I} =< G = (V,E) >$ be an instance of problem HAMILTONIAN CYCLE. We are going now to transform this instance into some instance of the problem HAMILTONIAN PATH in the following way: We will construct a graph $G' = (V', E')$ such that.

- Let u be some arbitrary vertex of V;
- $V' := V \cup \{v\}$ such that v is a vertex not belonging to V;
- $E' := E \cup \{(v,\ell) : \ell$ is a neighborhood of u in $G\}$.

This transformation can be done in polynomial time (we just need to copy the graph G by adding a vertex and some edges).

It is easy to prove that:

- If there exists some Hamiltonian cycle in G, then there exists a Hamiltonian path in G'.

  Let $C = (u, \ell_1, \ldots, \ell_{n-1}, u)$ be a Hamiltonian cycle in G. We construct the path $\mathscr{P} = (u, \ell_1, \ldots, \ell_{n-1}, v)$ in the graph G'. This path is Hamiltonian: It goes exactly once through v and through each vertex of G since C is some Hamiltonian cycle.
- If there exists some Hamiltonian path in G', then there exists some Hamiltonian cycle in G.

  Let $\mathscr{P} = (u, \ell_1, \ldots, \ell_{n-1}, v)$ be a path in the graph G'. The cycle $C = (u, \ell_1, \ldots, \ell_{n-1}, u)$ is Hamiltonian for the graph G.

So the problem HAMILTONIAN CYCLE reduces to problem HAMILTONIAN PATH in polynomial time.

So the problem HAMILTONIAN PATH is NP-complete.

Exercice 13.5 (page 214).

- The problem PATH is in NP since given some path, on can check in polynomial time if it is of length $n/2$ and that it as u and v as extremities.

- We are going now to do a reduction from the problem HAMILTONIAN PATH. Let $\mathscr{I} =< G = (V,E), u, v >$ be an instance of problem HAMILTONIAN PATH.

  We transform this instance in an instance $\mathscr{I}'$ of problem CHAIN in the following way. We construct a graph $G' = (V', E')$ such that

  > G' is a copy of graph G plus a chain of $|V|$ vertices whose a unique vertex of this chain is neighbour of u.

  This transformation is done in polynomial time (we have just copied the graph G by adding a vertex and some edges).

  It is easy to prove that there exists some Hamiltonian chain in G if and only if there exists a Hamiltonian chain in G' of length $\frac{|V'|}{2}$.

So the problem HAMILTONIAN CHAIN reduces to the problem CHAIN; and the latter is hence NP-complete.

Exercice 13.6 (page 214). The problem TREE is NP-complete. It suffices to observe that an Hamiltonian path is a covering tree with two leaves.

Exercice 13.7 (page 215).

Point 1.: Let S be a vertex cover of graph G. One needs to prove that all the vertices of the graph are either neighbours of S or in S in the graph G'.

1. As all the edges of G have at least one of their extremities in S, all the vertices of G' corresponding to some edge of G are neighbours of S.
2. Let v be a vertex of V. If v does not belong to S, then v has at least an extremity of some edge (since G is connected). Since S is a vertex cover of graph G this implies that the other extremity of this edge is in S. So v is a neighbour of a vertex of S. Consequently, all the vertices of G are either in S or are neighbour of S.

So S is a dominating set of G'.

Point 2.: Suppose that $S' \subseteq V$. S' dominates the graph G'. This means that all the edges of G have one of its extremities in S'. So S' is a vertex cover of graph G.

Suppose that S' is not a subset of V. There exists a vertex s not belonging to V in S'. Then s is a vertex representing an edge (u, v) in graph G.

- If the two vertices u and v are in S', then $S' \backslash \{s\}$ is a dominant set of G' of cardinality smaller than S'.
- If one of the two vertices u and v are in S', then $S' \backslash \{s\} \cup \{u, v\}$is a dominant set of G' of same cardinality as S'.

We repeat the same reasoning on this set until all the vertices not belonging to V are suppressed.

Point 3: *Inputs* : a non-oriented graph G, and an integer k
Question : Does there exist a dominant set S of G such that $|S| \leq k$?

Point 4: One can check in polynomial time if a set of vertices is a dominant set and it is of cardinality less than k.

Point 5. It suffices to combine previous questions.

Exercice 13.8 (page 216). Point 1. Given a subset of vertices of K, one can check in polynomial time

1. if S of of cardinality less than k;
2. if, for every vertex v of V, its distance to one of the vertices of S is less than b.

Point 2. We will do the reduction from the problem DOMINANT. Let I be an instance of the minimum dominant: $G = (V, E)$ and some integer k'.

We construct the instance of problem k-CENTRES : $k' = k$ and the complete graph $K = (V, E')$ with the following weight function on the edges:

$$w(u, v) = \begin{cases} 1 & if \quad (u, v) \in E; \\ 2 & if \quad (u, v) \notin E. \end{cases}$$

This reduction can be done in polynomial time and it satisfies the following properties:

- If there exists a dominant set of size less than k in G then K admits a k-center of cost 1.
- If there exists a dominant set of size more than k in G then K admits a k-center of cost 2.

We can deduce that there exists a dominant set of size less than k in G if and only if K admits a k-center of cost 1.

Chapter 14

Exercice 14.1 (page 226). We now that NP $\subseteq$ PSPACE. We must prove the reverse inclusion. One considers SAT in PSPACE, that is NP-complete. It is hence NP-hard, and hence also PSPACE-hard. So for every language A PSPACE, A reduces to SAT, and since SAT $\in$ NP, we have A in NP.

1 Bibliographic notes

Several corrections are shamefully taken from [Arnold & Guessarian, 2005] for the first chapter. Some others are inspired from results proved in book [Cori & Lascar, 1993] or from book [Kleinberg & Tardos, 2006], or exercises from [Sipser, 1997].

Index

Bibliography

[Arnold & Guessarian, 2005] Arnold, A. & Guessarian, I. (2005). *Mathématiques pour l'informatique*. Ediscience International.

[Cori & Lascar, 1993] Cori, R. & Lascar, D. (1993). *Logique mathématique. Volume I.* Masson.

[Kleinberg & Tardos, 2006] Kleinberg, J. M. & Tardos, É. (2006). *Algorithm design*. Addison-Wesley.

[Kozen, 2006] Kozen, D. (2006). *Theory of computation*. Springer-Verlag New York Inc.

[Sipser, 1997] Sipser, M. (1997). *Introduction to the Theory of Computation*. PWS Publishing Company.