Outils Logiques : Travaux Dirigés*

12 décembre 2006

Contrôle des connaissances

Le contrôle continu en TD est organisé comme suit : chaque semaine une liste d'exercices est proposée et un certain nombre d'étudiants sont invités à rédiger le corrigé et à le présenter la semaine suivante. Des épreuves écrites sont aussi possibles.

La note finale N est déterminée par la formule :

$$N = (1/2)E + (1/4)max(TD, E) + (1/4)\,TP$$

où E est la note de l'examen écrit, TD est la note de contrôle continu des travaux dirigés et TP est la note des travaux pratiques. A la deuxième session on garde les notes TD et TP mais dans le calcul de la note finale, la note E est remplacée par la note obtenue à l'examen écrit de la deuxième session.

Préliminaires

- Soit $V = \{x_1, x_2, \ldots\}$ un ensemble dénombrable de *variables propositionnelles*.
- L'ensemble *Form* des *formules* est le plus petit ensemble tel que $Form \supseteq V$ et si $A, B \in Form$ alors

  $$\begin{array}{ll} \neg A & \text{(négation)}, \\ (A \wedge B) & \text{(conjonction)} \\ (A \vee B) & \text{(disjonction)} \end{array}$$

  sont des formules.
- Si $A \in V$ on dit que A est une *formule atomique*.
- Si $A \in V$ ou $A = \neg B$ et $B \in V$ on dit que A est un littéral. Dans le premier cas on dit que le littéral est positif et dans le deuxième qu'il est négatif. On dénote un littéral avec $\ell, \ell', \ldots$
- L'ensemble $Var(A)$ des variables présentes dans la formule A est défini par :

  $$Var(x) = \{x\}, \quad Var(\neg A) = Var(A), \quad Var(A \wedge B) = Var(A \vee B) = Var(A) \cup Var(B).$$

- $\mathbf{2} = \{0, 1\}$ est l'ensemble des *valeurs booléennes*, d'après George Boole. De façon équivalente on peut utiliser $B = \{\mathsf{faux}, \mathsf{vrai}\}$ avec la convention que faux correspond à 0 et vrai à 1.
- Une *affectation* est une fonction *partielle*

  $$v : V \rightharpoonup \mathbf{2}$$

  avec domaine de définition $dom(v)$.
- Si v est une affectation, x est une variable propositionnelle et b une valeur booléenne alors $v[b/x]$ est l'affectation définie par

  $$v[b/x](y) = \begin{cases} b & \text{si } y = x \\ v(y) & \text{autrement} \end{cases}$$

*Les exercices marqués par une * sont plus difficiles et peuvent demander un certain temps de réflexion.

– L'interprétation $[\![A]\!]v$ d'une formule A par rapport à l'affectation v est définie par récurrence sur la structure de A en supposant que $Var(A) \subseteq dom(v)$ (autrement l'interprétation n'est pas définie) :

$$[\![x]\!]v = v(x) \qquad [\![\neg A]\!]v = NOT([\![A]\!]v)$$
$$[\![A \wedge B]\!]v = AND([\![A]\!]v, [\![B]\!]v) \qquad [\![A \vee B]\!]v = OR([\![A]\!]v, [\![B]\!]v).$$

où les fonctions NOT, AND, OR sont définies par :

x	$NOT(x)$
0	1
1	0

x	y	$AND(x,y)$
0	0	0
0	1	0
1	0	0
1	1	1

x	y	$OR(x,y)$
0	0	0
0	1	1
1	0	1
1	1	1

Parfois, il est préférable d'utiliser une notation plus compacte, à savoir : $\overline{x} = NOT(x)$, $x + y = OR(x, y)$ et $x \cdot y = AND(x, y)$.

– On écrit $v \models A$ si $[\![A]\!]v = 1$.
– On dit que A est *satisfiable* s'il existe une affectation v telle que $v \models A$.
– On dit que A est *valide* (ou une *tautologie*) si pour toute affectation v, $v \models A$.
– On appelle *clause* une disjonction de littéraux. Une formule est en forme normale conjonctive (CNF pour *Conjuctive Normal Form*) si elle est une conjonction de clauses. Dans le cours on montre que toute formule est 'équivalente' à une formule en CNF.

Exercice 0.1 *Montrez que A est valide si et seulement si $\neg A$ n'est pas satisfiable.*

Exercice 0.2 *Si X est un ensemble de variables et v est une affectation alors $v_{|X}$ est la restriction de v à X. Soit A une formule et $X \supseteq Var(A)$. Montrez que si $v_{|X} = v'_{|X}$ alors $[\![A]\!]v = [\![A]\!]v'$. Donc l'interprétation $[\![A]\!]v$ est indépendante des valeurs de l'affectation v sur les variables propositionnelles qui ne sont pas présentes dans A.*

Calcul propositionnel 1 (méthode de Davis Putnam)

La méthode de Davis Putnam permet de décider si une formule en forme normale conjonctive est satisfiable. On représente une formule A en CNF comme un ensemble (éventuellement vide) de clauses $\{C_1, \ldots, C_n\}$ et une clause C comme un ensemble (éventuellement vide) de littéraux. Dans cette représentation, on définit la substitution $[b/x]A$ d'une valeur booléenne $b \in \{0, 1\}$ dans A comme suit :

$$[b/x]A = \{[b/x]C \mid C \in A \text{ et } [b/x]C \neq 1\}$$

$$[b/x]C = \begin{cases} 1 & \text{si } (b = 1 \text{ et } x \in C) \text{ ou } (b = 0 \text{ et } \neg x \in C) \\ C\backslash\{\ell\} & \text{si } (b = 1 \text{ et } \ell = \neg x \in C) \text{ ou } (b = 0 \text{ et } \ell = x \in C) \\ C & \text{autrement} \end{cases}$$

On définit une fonction DP qui agit récursivement sur une formule A en CNF dans la représentation décrite ci-dessus :

function $DP(A) =$ case	
(1) $A = \emptyset$:	true
(2) $\emptyset \in A$	false
(3) $\{x, \neg x\} \subseteq C \in A$:	$DP(A\backslash\{C\})$
(4) $\{x\} \in A$:	$DP([1/x]A)$
(5) $\{\neg x\} \in A$:	$DP([0/x]A)$
(6) *else* :	choisir x dans A;
	$DP([0/x]A)$ or $DP([1/x]A)$

Dans (1), nous avons une conjonction du vide qui par convention est équivalente à true. Dans (2), A contient une clause vide. La disjonction du vide étant équivalente à false, la formule A est aussi équivalente à false. Dans (3), une clause contient un littéral et sa négation et elle est donc équivalente à true. Dans (4) et (5), A contient une clause qui est constituée uniquement d'une variable ou de sa négation. Ceci permet de connaître la valeur de la variable dans toute affectation susceptible de satisfaire la formule. Dans (6), nous considérons les deux valeurs possibles d'une affectation d'une variable.

Exercice 0.3 *Appliquez DP aux formules* $\{\{x, \neg y\}, \{\neg x, y\}\}$ *et* $\{\{x, y\}, \{\neg x, y\}, \{x, \neg y\}, \{\neg x, \neg y\}\}$.

Exercice 0.4 (1) *Montrez que si A est une fonction en CNF alors la fonction DP termine.*
(2) *Montrez que $DP(A)$ retourne* true *(*false*) si et seulement si A est satisfiable (ne l'est pas).*

Exercice 0.5 *Fait : toute formule peut être transformée en CNF. Expliquez comment utiliser la méthode de Davis-Putnam pour décider la validité d'une formule.*

Exercice 0.6 *Modifiez la fonction DP pour que, si la formule A est satisfiable, elle retourne une affectation v qui satisfait A.*

Exercice* 0.7 *Réfléchissez aux structures de données et aux opérations nécessaires à la mise en oeuvre de l'algorithme en Java.*

Calcul Propositionnel 2 (équivalence et définissabilité)

Exercice 0.8 *Montrez les équivalences logiques :*

$$(A \vee \mathbf{0}) \equiv A, \quad (A \vee \mathbf{1}) \equiv \mathbf{1}, \quad (A \vee B) \equiv (B \vee A),$$
$$((A \vee B) \vee C) \equiv (A \vee (B \vee C)), \quad (A \vee A) \equiv A$$
$$(A \wedge \mathbf{0}) \equiv \mathbf{0}, \quad (A \wedge \mathbf{1}) \equiv A, \quad (A \wedge B) \equiv (B \wedge A),$$
$$((A \wedge B) \wedge C) \equiv (A \wedge (B \wedge C)), \quad (A \wedge A) \equiv A,$$
$$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C), \quad (A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C),$$
$$\neg\neg A \equiv A, \quad \neg(A \vee B) \equiv ((\neg A) \wedge (\neg B)), \quad \neg(A \wedge B) \equiv ((\neg A) \vee (\neg B)) \ .$$

Exercice 0.9 (1) *Montrez l'équivalence logique :*

$$(A \wedge B) \vee (\neg A \wedge B) \equiv B \tag{1}$$

(2) *On peut appliquer cette équivalence logique pour simplifier une forme normale disjonctive. Par exemple, considérez la fonction $f(x, y, z)$ définie par le tableau de vérité :*

$x \backslash yz$	00	01	11	10
0	0	1	1	0
1	1	1	1	1

Calculez la forme normale disjonctive de f et essayez de la simplifier en utilisant l'équivalence logique (1).

(3) *La présentation du tableau de vérité n'est pas arbitraire... Proposez une méthode graphique pour calculer une forme normale disjonctive simplifiée.*

Exercice 0.10 *Soit f une fonction sur les nombres naturels. Dire qu'un problème est décidé en $O(f)$, signifie qu'on dispose d'un algorithme A et de n_0, k nombres naturels tels que pour toute entrée dont la taille n est supérieure à n_0, le temps de calcul de A sur l'entrée en question est inférieure à $k \cdot f(n)$.*

(1) *Montrez que la satisfaction d'une formule en DNF et la validité d'une formule en CNF peuvent être décidées en $O(n)$.*

(2) *Soit $pair(x_1, \ldots, x_n) = (\Sigma_{i=1,\ldots,n} x_i) \bmod 2$ la fonction qui calcule la parité d'un vecteur de bits. Montrez que la représentation en DNF ou CNF de cette fonction est en $O(2^n)$. Peut-on appliquer l'équivalence logique (1) pour simplifier la représentation ?*

Exercice 0.11 (if-then-else) *La fonction ternaire ITE est définie par $ITE(1, x, y) = x$ et $ITE(0, x, y) = y$. Montrez que toute fonction $f : \mathbf{2}^n \to \mathbf{2}$, $n \geq 0$ s'exprime par composition de la fonction ITE et des (fonctions) constantes 0 et 1.*

Exercice 0.12 *L'*or exclusif $\oplus$ *(xor) est défini par $A \oplus B \equiv (A \wedge \neg B) \vee (\neg A \wedge B)$. Montrez que :*

(1) *$\oplus$ est associatif et commutatif.*

(2) *$x \oplus 0 \equiv x$ et $x \oplus x \equiv 0$.*

(3) *Toute fonction booléenne $f : \mathbf{2}^n \to \mathbf{2}$ peut être représentée à partir de $\mathbf{1}$, $\wedge$ et $\oplus$.*

Exercice 0.13 (nand,nor) *Les fonctions binaires NAND et NOR sont définies par $NAND(x, y) = NOT(AND(x, y))$ et $NOR(x, y) = NOT(OR(x, y))$. Montrez que toute fonction $f : \mathbf{2}^n \to \mathbf{2}$, $n \geq 0$, s'exprime comme composition de la fonction NAND (ou de la fonction NOR). Montrez que les 4 fonctions unaires possibles n'ont pas cette propriété et que que parmi les 16 fonctions binaires possibles il n'y en a pas d'autres qui ont cette propriété.*

Calcul Propositionnel 3 (clauses de Horn et circuits combinatoires)

Exercice 0.14 *Une clause de (Alfred) Horn est une clause (c'est-à-dire une disjonction de littéraux) qui contient au plus un littéral positif. Une formule de Horn est une formule en CNF dont les clauses sont des clauses de Horn.*

(1) *Montrez que toute formule de Horn est équivalente à la conjonction (éventuellement vide) de clauses de Horn de la forme :*

$$\begin{array}{ll}(1) & x\\ (2) & \neg x_1 \vee \cdots \vee \neg x_n\\ (3) & \neg x_1 \vee \cdots \vee \neg x_n \vee x_{n+1}\end{array}$$

où $n \geq 1$ et $x_i \neq x_j$ si $i \neq j$. Dans ce cas on dit que la formule de Horn est réduite.

(2) *Montrez qu'une formule de Horn réduite qui ne contient pas de clauses de la forme (1) ou qui ne contient pas de clauses de la forme (2) est satisfiable.*

(3) *Donnez une méthode efficace (temps polynomial) pour déterminer si une formule de Horn est satisfiable.*

Exercice 0.15 *Un décodeur est un circuit avec n entrées $x_{n-1}, \ldots, x_0$ et 2^n sorties $y_{2^n-1}, \ldots, y_0$ tel que*

$$y_i = 1 \text{ ssi } i = (x_{n-1} \cdots x_0)_2$$

Réalisez un tel circuit.

Exercice 0.16 *Un additionneur est un circuit booléen avec $2n$ entrées $x_{n-1}, y_{n-1}, \ldots, x_0, y_0$ et $n+1$ sorties $r_n, s_{n-1}, \ldots, s_0$ tel que*

$$(x_{n-1} \cdots x_0)_2 + (y_{n-1} \cdots y_0)_2 = (r_n s_{n-1} \cdots s_0)_2$$

On peut réaliser un additionneur en utilisant l'algorithme standard qui propage la retenue de droite à gauche.

(1) *Réalisez un circuit A avec 3 entrées x, y, r et deux sorties s, r' tel que*

$$(r's)_2 = (x)_2 + (y)_2 + (r)_2$$

(2) *Expliquez comment inter-connecter n circuits A pour obtenir un additionneur sur n bits.*

(3) *Montrez que dans le circuit en question le nombre de portes et la longueur du chemin le plus long sont proportionnels à n.*

Exercice* 0.17 *Le but de cet exercice est de réaliser un additionneur dont le nombre de portes est encore polynomiale en n mais dont la longueur du chemin le plus long est proportionnelle à $lg(n)$. Pour éviter que la retenue se propage à travers tout le circuit, l'idée est d'anticiper sa valeur. Ainsi pour additionner 2 vecteurs de longueur n, on additionne les premiers $n/2$ bits (ceux de poids faible) et en même temps on additionne les derniers $n/2$ bits (ceux de poids fort) deux fois (en parallèle) une fois avec retenue initiale 0 et une fois avec retenue initiale 1. On applique cette méthode récursivement sur les sous-vecteurs de longueur $n/4, n/8, \ldots$ selon le principe* diviser pour régner.

(1) *Construisez explicitement un tel circuit pour $n = 4$.*

(2) *Déterminez en fonction de n le nombre de portes et la longueur du chemin le plus long du circuit obtenu.*

Langages formels et automates finis

Exercice 0.18 *Montrez que pour tout langage L, $L^* = (L^*)^*$.*

Exercice 0.19 *Montrez qu'il existe des langages L_1 et L_2 tels que $(L_1 \cup L_2)^* \neq L_1^* \cup L_2^*$.*

Exercice 0.20 *Montrez qu'il existe des langages L_1 et L_2 tels que $(L_1 \cdot L_2)^* \neq L_1^* \cdot L_2^*$.*

Exercice 0.21 *Pour chacun des langages suivants, construire un automate fini non déterministe qui l'accepte :*

1. *Les représentations binaires des nombres pairs.*
2. *Le langage des mots sur l'alphabet $\{a, b\}$ contenant ou bien la chaîne aab ou bien la chaîne aaab.*
3. *Le langage des mots sur l'alphabet $\{0, 1\}$ dont le troisième caractère de droite existe et est égale à 1.*

Construire des automates déterministes pour les langages décrits ci-dessus.

Exercice* 0.22 *Soient M un AFD qui accepte un langage L et N_1, N_2 deux AFN qui acceptent les langages L_1, L_2, respectivement (sur un alphabet Σ fixé).*

1. *Montrez qu'on peut construire un AFD qui accepte le langage complémentaire $\Sigma^* \backslash L$.*
2. *Montrez qu'on peut construire un AFN qui accepte le langage $L_1 \cup L_2$ et le langage itéré $(L_1)^*$.*
3. *Conclure que la classe des langages acceptés par un AFD est stable par union, intersection, complémentaire et itération.*

Preuves par induction

Exercice 0.23 (transitivité) *Soit R une relation binaire sur un ensemble. Sa clôture réflexive et transitive R^* est la plus petite relation qui contient la relation identité, la relation R et telle que si $(x, y), (y, z) \in R^*$ alors $(x, z) \in R^*$. Montrez que R^* peut être vu comme un ensemble défini inductivement.*

Exercice 0.24 *Un graphe non-dirigé G est composé d'un ensemble fini non-vide de noeuds N et d'un ensemble A d'arêtes qui connectent les noeuds. Formellement, une arête est un ensemble $\{i, j\}$ de noeuds de cardinalité 2. Le* degré *d'un noeud i dans un graphe est le nombre d'arêtes qui le contiennent. Par exemple, un noeud isolé a degré 0. Démontrez en utilisant le principe de récurrence l'assertion suivante :*

Chaque graphe avec au moins 2 noeuds contient 2 noeuds avec le même degré.

Exercice 0.25 *Soit $(\mathbf{N} \cup \{\infty\}, \leq)$ l'ensemble des nombres naturels avec un élément maximum ∞, $0 < 1 < 2 < \ldots < \infty$. Montrez que toute fonction monotone f sur cet ordre admet un point fixe, c'est-à-dire un élément x tel que $f(x) = x$.*

Exercice 0.26 *On considère l'ensemble de symboles fonctionnels*

$$\Sigma = \{\epsilon, a, b\}$$

où $ar(\epsilon) = 0$ et $ar(a) = ar(b) = 1$. Calculez l'ensemble librement engendré associé à Σ. Cet ensemble est-il isomorphe à un ensemble déjà considéré dans le cours ?

Exercice 0.27 *Soient $\mathbf{N}$ l'ensemble des nombres naturels, $\mathbf{N}^k$ le produit cartésien $\mathbf{N} \times \cdots \times \mathbf{N}$ k fois et $A = \bigcup\{\mathbf{N}^k \mid k \geq 1\}$. Soit $<$ une relation binaire sur A telle que : $(x_1, \ldots, x_n) < (y_1, \ldots, y_m)$ ssi il existe $k \leq min(n, m)$ $(x_1 = y_1, \ldots, x_{k-1} = y_{k-1}, x_k < y_k)$ Est-il vrai que $<$ est un ordre bien fondé ? Donner soit une preuve soit un contre-exemple.*

Terminaison 1

Exercice 0.28 *On considère des programmes impératifs* while *dont les variables prennent comme valeurs des nombres naturels. Montrez que le programme suivant termine où l'on sait que le test Φ termine et n'a pas d'effet de bord (c'est-à-dire que l'évaluation du test n'affecte pas la valeur associée aux variables) :*

$$\begin{array}{l} \textit{while } u > l+1 \textit{ do} \\ (r := (u+l) \textit{ div } 2; \\ \textit{if } \Phi \textit{ then } u := r \textit{ else } l := r) \end{array}$$

Exercice 0.29 *Soit $(\{a,b,c,d\}, \rightarrow)$ un système de réécriture où $\rightarrow = \{(c,a),(c,d),(d,c),(d,b)\}$. Dire si le système termine, est localement confluent, est confluent.*

Exercice 0.30 (1) *Utilisez le principe d'induction pour démontrer la terminaison de la fonction récursive a telle que :*

$$\begin{array}{ll} a(0,n) & = n+1 \\ a(m+1,0) & = a(m,1) \\ a(m+1,n+1) & = a(m,a(m+1,n)) \end{array}$$

(2) *Calculez à l'aide d'un programme autant de valeurs $a(n,n)$ que possible.*

Exercice 0.31 *On étend l'ordre lexicographique à un produit $A = A_1 \times \cdots \times A_n$, $n \geq 3$, d'ordres bien fondés $(A_i, >_i)$:*

$$(x_1,\ldots,x_n) > (y_1,\ldots,y_n) \textit{ si } \exists\, k \leq n \ \ (x_1 = y_1,\ldots,x_{k-1} = y_{k-1} \textit{ et } x_k >_k y_k)$$

Montrez que $(A, >)$ est bien fondé.

Exercice 0.32 (ordre produit) *Soient $(A_i, >_i)$ pour $i = 1,\ldots,n$ des ordres bien fondés. On définit une relation $>$ sur le produit cartésien $A_1 \times \cdots \times A_n$ par*

$$(a_1,\ldots,a_n) > (a'_1,\ldots,a'_n) \textit{ si } a_i \geq_i a'_i, i = 1,\ldots,n \textit{ et } \exists\, i \in \{1,\ldots,n\} \ \ a_i >_i a'_i$$

(1) *La relation $>$ est-elle un ordre bien fondé ?*

(2) *Comparez la relation $>$ à l'ordre lexicographique sur le produit défini dans l'exercice 0.31.*

Terminaison 2

Exercice 0.33 *Considérons les programmes while :*

$$\begin{array}{l} \textit{while } m \neq n \textit{ do} \\ \quad \textit{if } m > n \textit{ then } m := m - n \textit{ else } n := n - m \end{array}$$

$$\begin{array}{l} \textit{while } m \neq n \textit{ do} \\ \quad \textit{if } m > n \textit{ then } m := m - n \\ \quad \textit{else } h := m; m := n; n := h \end{array}$$

Dire si les programmes terminent quand les variables varient sur les nombres naturels positifs.

Exercice 0.34 *Soit $A = \{a, b\}^*$ l'ensemble des mots finis sur l'alphabet $\{a, b\}$. Soit $\rightarrow$ une relation binaire sur A telle que :*

$$w \rightarrow w' \textit{ ssi } w = w_1 abw_2 \textit{ et } w' = w_1 bbaw_2$$

Donc w se réduit à w' si w' est obtenu de w en remplaçant un sous-mot ab avec bba.

Montrez ou invalidez les assertions suivantes (il est conseillé de s'appuyer sur les résultats démontrés dans le cours) :

1. *Le système de réduction $(A, \rightarrow)$ est à branchement fini.*
2. *Le système termine.*
3. *Le système est localement confluent.*
4. *Le système est confluent.*

Exercice* 0.35 *Soient $X, Y \in \mathcal{M}(A)$. On écrit $X >_1 Y$ si Y est obtenu de X en remplaçant un élément de X par un multi-ensemble d'éléments strictement plus petits. Montrez que la clôture transitive de $>_1$ est égale à $>_{\mathcal{M}(A)}$.*

Exercice 0.36 *Soit $\mathbf{N}^*$ les mots finis de nombres naturels. Montrez la terminaison du système :*

$$u(i+1)v \rightarrow iviui \textit{ pour } u, v \in \mathbf{N}^*$$

Calculabilité 1 (machines de Turing et énumérations)

Exercice 0.37 *Donnez la description formelle d'une MdT qui décide le langage* $\{w\sharp w \mid w \in \{0,1\}^*\}$.

Exercice 0.38 *Donnez la description formelle d'une MdT qui décide le langage des mots sur l'alphabet* $\{0\}$ *dont la longueur est une puissance de* 2 *:* 2^0, 2^1, $2^2, \ldots$

Exercice 0.39 *Décrivez informellement une MdT qui décide le langage :*

$$\{a^i b^j c^k \mid i \cdot j = k \text{ et } i,j,k \geq 1\} \ .$$

Exercice 0.40 *Soit* $\Sigma = \{0,1\}$ *et* $suc : \Sigma^* \to \Sigma^*$ *la fonction 'successeur' en base* 2 *telle que :*

$$(suc(w))_2 = (w)_2 + 1$$

Montrez que suc est récursive.

Exercice 0.41 *On peut énumérer les couples de nombres naturels en procédant 'par diagonales' :*

$$(0,0), \quad (1,0),(0,1), \quad (2,0),(1,1),(0,2), \quad (3,0)\ldots$$

Montrez que la fonction $\langle m,n\rangle = (m+n)(m+n+1)/2 + n$ *est une bijection entre* $\mathbf{N}\times\mathbf{N}$ *et* $\mathbf{N}$. *Décrire un algorithme pour calculer la fonction inverse.*

Exercice 0.42 *On définit les fonctions* $\langle\_\rangle_k : \mathbf{N}^k \to \mathbf{N}$ *pour* $k \geq 2$ *:*

$$\begin{array}{ll} \langle m,n\rangle_2 & = \langle m,n\rangle \\ \langle n_1,\ldots,n_k\rangle_k & = \langle\langle n_1,\ldots,n_{k-1}\rangle_{k-1}, n_k\rangle \text{ si } k \geq 3 \end{array}$$

Montrez que les fonctions $\langle\_\rangle_k$ *sont des bijections.*

Calculabilité 2 (énumérations et indécidabilité)

Exercice 0.43 *On considère l'ensemble* $\mathbf{N}^*$ *des mots finis de nombres naturels. Notez que* $\mathbf{N}^*$ *est en correspondance bijective avec* $\bigcup_{k\geq 0} \mathbf{N}^k$*. Définissez une bijection entre* $\mathbf{N}^*$ *et* $\mathbf{N}$*.*

Exercice 0.44 *Soit* $\Sigma = \{a, b, \ldots, z\}$ *un alphabet fini. On peut énumérer les éléments de* Σ^* *comme suit :*

$$\epsilon, \quad a, b, \ldots, z, \quad aa, \ldots, az, ba, \ldots, bz, za, \ldots, zz, \quad aaa, \ldots$$

Si Σ *contient* k *éléments on aura* k^0 *mots de longueur* 0*,* k *mots de longueur* 1*,* k^2 *mots de longueur* 2*, ... Définissez une bijection entre* Σ^* *et* $\mathbf{N}$*.*

Exercice* **0.45** (1) *Montrez qu'un langage est semi-décidable si et seulement si il est le domaine de définition d'une fonction partielle récursive.*

(2) *On dit qu'un langage* $L \subseteq \Sigma^*$ *est* récursivement énumérable *s'il est l'image d'une fonction partielle récursive. Montrez qu'un langage* L *est récursivement énumérable si et seulement si il est semi-décidable.*

Suggestion : Soit M *une MdT et* $w_0, w_1, w_2, \ldots$ *une suite d'entrées. On peut simuler* M *sur* w_0 *pour* 0 *pas, sur* w_0 *pour* 1 *pas, sur* w_1 *pour* 0 *pas, sur* w_0 *pour* 2 *pas, sur* w_1 *pour* 1 *pas, sur* w_2 *pour* 0 *pas,...*

Exercice* **0.46** (1) *Montrez que les langages acceptés par un AFN sont décidables.*

(2) *Montrez que la collection des langages décidables est stable par rapport aux opérations d'union, complémentaire, concaténation et itération.*

(3) *Montrez que la collection des langages semi-décidables est stable par rapport aux opérations d'union et concaténation.*

Suggestion : utilisez le non-déterminisme.

Exercice 0.47 *Montrez ou invalidez les assertions suivantes :*

1. *Il y a une MdT qui accepte les mots sur l'alphabet* $\{0, 1\}$ *qui contiennent autant de* 0 *que de* 1 *(si la MdT existe, il suffira d'en donner une description informelle).*
2. *Rappel : si* A *et* B *sont deux langages, on écrit* $A \leq B$ *s'il existe une réduction de* A *à* B*. Si* A *est semi-décidable et* $A \leq A^c$ *alors* A *est décidable.*
3. *L'ensemble des (codages de) MdT qui reconnaissent un langage fini est décidable.*

Exercice 0.48 *Montrez ou donnez un contre-exemple aux assertions suivantes :*

1. *L'ensemble des (codages de) MdT qui terminent sur le mot vide est décidable.*
2. *L'ensemble des (codages de) MdT qui divergent sur le mot vide est semi-décidable.*
3. *L'ensemble des (codages de) MdT qui terminent sur le mot vide en* 10^{100} *pas de calcul est décidable.*

Complexité (réductions polynomiales)

Exercice 0.49 *Un graphe (non-dirigé) G est composé d'un ensemble fini non-vide de noeuds N et d'un ensemble A d'arêtes qui connectent les noeuds. Formellement, une arête est un ensemble $\{i, j\}$ de noeuds de cardinalité 2. On dit que deux noeuds sont adjacents s'il y a une arête qui les connecte.*

Problème du coloriage *Étant donné un graphe $G = (N, A)$ et un nombre naturel $k \geq 2$ on détermine s'il existe une fonction $c : N \to \{1, \ldots, k\}$ telle que si i, j sont deux noeuds adjacents alors $c(i) \neq c(j)$.*[1]

Problème de l'emploi du temps *Étant donné (i) un ensemble d'étudiants $E = \{1, \ldots, n\}$ ($n \geq 2$), (ii) un ensemble de cours $C = \{1, \ldots, m\}$ ($m \geq 2$), (iii) un ensemble de plages horaires $P = \{1, \ldots, p\}$ ($p \geq 2$) et (iv) une relations binaire R telle que $(i, j) \in R$ si et seulement si l'étudiant i suit le cours j on détermine s'il existe une fonction emploi du temps $edt : C \to P$ telle que si un étudiant suit deux cours différents $j \neq j'$ alors $edt(j) \neq edt(j')$.*

Démontrez ou donnez un contre-exemple aux assertions suivantes :

1. *Le problème de l'emploi du temps se réduit au problème du coloriage.*
2. *Le problème de l'emploi du temps se réduit en temps polynomial au problème du coloriage.*
3. *Le problème du coloriage est dans NP.*

Exercice 0.50 *Soit G un graphe non-dirigé (cf. exercice 0.49). Un k-clique est un ensemble de k noeuds de G qui ont la propriété que chaque couple de noeuds est connectée par une arête.*

Le langage CLIQUE est composé de couples $\langle G, k\rangle$ où (i) G est le codage d'un graphe, (ii) k est un nombre naturel et (iii) G contient comme sous-graphe un k-clique.

Le langage 3-SAT est composé de formules en forme normale conjonctive où chaque clause contient 3 littéraux.

1. *Montrez que le langage CLIQUE est dans NP.*
2. *On souhaite construire une réduction polynomiale de 3-SAT à CLIQUE. Si la formule A contient k clauses alors le graphe associé G_A contient k groupes de noeuds où chaque groupe est composé de 3 noeuds et chaque noeud est étiqueté par un littéral. Par exemple, si la clause est $(x \vee \neg y \vee z)$ alors on aura un groupe de 3 noeuds étiquetés avec x, $\neg y$ et z.*

   (a) *Décrivez les arêtes de G_A de façon à ce que le graphe G_A contienne une k-clique si et seulement si la formule A est satisfiable et dessinez le graphe G_A dans le cas où*

   $$A = (x \vee y) \wedge (\neg x \vee \neg y) \wedge (\neg x \vee y) \wedge (x \vee \neg y)$$

   (la formule en question comporte seulement deux littéraux par clause mais la construction du graphe G_A s'applique aussi bien à ce cas).

   (b) *Quelle conclusion peut-on tirer de la construction précédente ? Motivez votre réponse :*

      i. *Si 3-SAT est un problème polynomiale déterministe alors CLIQUE est un problème polynomiale déterministe.*
      ii. *CLIQUE est un problème NP-complet.*

Exercice 0.51 *Soit A une matrice et b un vecteur à coefficients dans $\mathbf{Z}$. Le problème de programmation linéaire entière (ILP pour* integer linear programming*) consiste à déterminer s'il existe un vecteur $\vec{x}$ à coefficients dans $\mathbf{N}^m$ tel que $A\vec{x} = \vec{b}$.*[2] *Ce problème est dans NP. On utilise des notions*

[1] On peut voir les valeurs $\{1, \ldots, k\}$ comme des couleurs qu'on affecte aux noeuds, d'où le nom du problème.

[2] Comme pour le problème du voyageur de commerce, le problème ILP est souvent formulé comme un problème d'*optimisation*. Par exemple, il s'agit de minimiser une fonction linéaire $\vec{c}^T\vec{x}$ sous les contraintes $A\vec{x} = \vec{b}$ et $\vec{x} \geq 0$.

d'algèbre linéaire pour montrer que si le problème a une solution alors il en a une dont la taille est polynomiale dans la taille de la matrice A. Ensuite on peut appliquer la méthode standard qui consiste à deviner un vecteur $\vec{x}$ et à vérifier qu'il est une solution. A partir de ce fait, le but de l'exercice est de montrer que le problème est NP-complet par réduction du problème SAT. Il peut être utile de considérer d'abord les problèmes suivants.

- *Montrez qu'en introduisant des variables auxiliaires on peut exprimer la satisfaction d'une contrainte d'inégalité comme un problème d'ILP.*
- *Montrez qu'on peut exprimer la contrainte* $x \in \{0, 1\}$.
- *Montrez qu'on peut exprimer la contrainte* $x = \overline{y}$ *où* $x, y \in \{0, 1\}$, $\overline{0} = 1$ *et* $\overline{1} = 0$.
- *Montrez comment coder la validité d'une clause (disjonction de littéraux).*

Système de preuve de Gentzen et compacité

Rappel : voici le système de preuve de Gentzen.

$$(Ax) \quad \frac{}{A, \Gamma \vdash A, \Delta}$$

$$(\wedge \vdash) \quad \frac{A, B, \Gamma \vdash \Delta}{A \wedge B, \Gamma \vdash \Delta} \qquad (\vdash \wedge) \quad \frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta}$$

$$(\vee \vdash) \quad \frac{A, \Gamma \vdash \Delta \quad B, \Gamma \vdash \Delta}{A \vee B, \Gamma \vdash \Delta} \qquad (\vdash \vee) \quad \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \vee B, \Delta}$$

$$(\neg \vdash) \quad \frac{\Gamma \vdash A, \Delta}{\neg A, \Gamma \vdash \Delta} \qquad (\vdash \neg) \quad \frac{A, \Gamma \vdash \Delta}{\Gamma \vdash \neg A, \Delta}$$

Exercice 0.52 *Montrez que :*

(1) *Un séquent $A, \Gamma \vdash A, \Delta$ est valide.*

(2) *Pour chaque règle d'inférence la conclusion est valide si et seulement si les hypothèses sont valides.*

Exercice 0.53 (sous-formule) *Montrez que si un séquent est dérivable alors il y a une preuve du séquent qui contient seulement des sous formules de formules dans le séquent.*

Exercice* 0.54 (affaiblissement) *Montrez que si le séquent $\Gamma \vdash \Delta$ est dérivable alors le séquent $\Gamma \vdash A, \Delta$ l'est aussi.*

Exercice 0.55 (implication) *Dans le système de Gentzen on peut donner un traitement direct de l'implication :*

$$(\rightarrow \vdash) \quad \frac{\Gamma \vdash A, \Delta \quad B, \Gamma \vdash \Delta}{A \rightarrow B, \Gamma \vdash \Delta} \qquad (\vdash \rightarrow) \quad \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \rightarrow B, \Delta}$$

Démontrez la correction et complétude du système de Gentzen étendu avec ces règles.

Exercice 0.56 *Montrez que les règles pour la disjonction et l'implication sont dérivables des règles pour la conjonction et la négation en utilisant les équivalences : $A \vee B \equiv \neg(\neg A \wedge \neg B)$ et $A \rightarrow B \equiv \neg A \vee B$.*

Exercice 0.57 (coupure) *La règle de* coupure *(ou* cut*) est :*

$$(\textit{coupure}) \quad \frac{A, \Gamma \vdash \Delta \quad \Gamma \vdash A, \Delta}{\Gamma \vdash \Delta}$$

Montrez que le système de Gentzen étendu avec cette règle est toujours correct (et complet).

Exercice 0.58 *Soit T un ensemble de formules. On écrit $T \models A$ si pour toute affectation v, si v satisfait T alors v satisfait A. Montrez que si $T \models A$ alors il existe T_0 sous-ensemble fini de T tel que $T_0 \models A$. Suggestion : utilisez le théorème de compacité.*

Résolution

Exercice 0.59 *Montrez que la règle d'inférence suivante est valide :*

$$\frac{A \vee \neg C \qquad B \vee C}{A \vee B} \tag{2}$$

Exercice 0.60 *Pour représenter les formules en CNF on adopte la même notation ensembliste utilisée pour décrire la méthode de Davis-Putnam.*

- *Une clause C est un ensemble de littéraux.*
- *Une formule A est un ensemble de clauses.*

Nous considérons la règle :

$$\frac{A \cup \{C \cup \{x\}\} \cup \{C' \cup \{\neg x\}\} \quad x \notin C \quad \neg x \notin C'}{A \cup \{C \cup \{x\}\} \cup \{C' \cup \{\neg x\}\} \cup \{C \cup C'\}} \tag{3}$$

Dans la suite on appelle (3) règle de résolution.[3] *L'effet de l'application de la règle consiste à ajouter une nouvelle clause $C \cup C'$ qu'on appelle* résolvant *des deux clauses $C \cup \{x\}$ et $C' \cup \{\neg x\}$.*

(1) *Montrez que l'hypothèse est logiquement équivalente à la conclusion.*

(2) *Conclure que si la conclusion n'est pas satisfiable alors l'hypothèse n'est pas satisfiable. En particulier, si la conclusion contient la clause vide alors l'hypothèse n'est pas satisfiable.*

Fait Si une formule A en CNF n'est pas satisfiable alors la règle de résolution permet de dériver une formule A' avec une clause vide. On dit que la règle de résolution est *complète pour la réfutation*, c'est-à-dire pour la dérivation de la clause vide. La méthode peut être implémentée itérativement. A chaque itération on ajoute toutes les clauses qui sont un résolvant de deux clauses. Cette itération termine forcement car le nombre de clauses qu'on peut construire est fini. Parfois, il convient de représenter la dérivation comme un graphe dirigé acyclique (ou DAG pour *directed acyclic graph*) dont les noeuds sont étiquetés par les clauses. Initialement on a autant de noeuds que de clauses et pas d'arêtes. Chaque fois qu'on applique la règle de résolution (3) on introduit un nouveau noeud qui est étiqueté avec la clause résolvant $C \cup C'$ et deux nouvelles arêtes qui vont des noeuds étiquetés avec les clauses $C \cup \{x\}$ et $C' \cup \{\neg x\}$ vers le noeud étiqueté avec la clause $C \cup C'$.

Exercice 0.61 *Soit A une formule en CNF et C une clause. Expliquez comment utiliser la méthode de résolution pour établir si l'implication $A \to C$ est valide.*

Exercice 0.62 *Construire la formule A en CNF qui correspond au principe du nid de pigeon avec 2 pigeons et 1 nid. Appliquez la règle de résolution. Même problème avec 2 pigeons et 2 nids.*

Exercice 0.63 *Soit A une formule en CNF avec m variables et n clauses. Montrez qu'il y a au plus $m \cdot (n \cdot (n-1)/2)$ façons d'appliquer la règle de résolution.*

Exercice 0.64 *Un exercice de révision. On considère les formules en CNF suivantes :*

1. $\neg x \vee (\neg y \vee x)$
2. $(x \vee y \vee \neg z) \wedge (x \vee y \vee z) \wedge (x \vee \neg y) \wedge \neg x$.
3. $(x \vee y) \wedge (z \vee w) \wedge (\neg x \vee \neg z) \wedge (\neg y \vee \neg w)$.

Pour chaque formule :

1. *Si la formule est valide calculez une preuve de la formule dans le système de Gentzen.*
2. *Si la formule est satisfiable mais pas valide calculez une affectation qui satisfait la formule en utilisant la méthode de Davis-Putnam.*
3. *Si la formule n'est pas satisfiable dérivez la clause vide en utilisant la méthode par résolution.*

[3] Sans les conditions $x \notin C$ et $\neg x \notin C'$ on pourrait par exemple 'simplifier' les clauses $\{x\}$ et $\{\neg x\}$ en $\{x, \neg x\}$.

Que sais-je ?

En vue de l'examen, voici une liste de *techniques de base* qui devraient être assimilées.

1. Appliquer la méthode de Davis-Putnam pour vérifier la satisfiabilité d'une formule du calcul propositionnel Voir planche TP1.
2. Construire un circuit combinatoire qui réalise une certaine fonction booléenne. Voir, par exemple, l'additionneur, exercice 0.16.
3. Appliquer le principe de preuve par induction (dont la principe de récurrence est un cas particulier). Voir, par exemple, l'exercice 0.24 pour une preuve par récurrence et l'exercice 0.30 pour une preuve par induction sur l'ordre lexicographique.
4. Montrer la terminaison d'un système de réduction à l'aide d'un plongement monotone dans un ordre bien fondé. Voir, par exemple, l'exercice 0.33 pour une preuve de terminaison d'une boucle while et l'exercice 0.34 pour une preuve de terminaison d'une relation de reduction sur les mots.
5. Construire une Machine de Turing qui décide un langage donné. Voir, par exemple, les exercices 0.37 et 0.40.
6. Réduction polynomiale d'un problème combinatoire à un autre (par exemple SAT). Voir, par exemple, le TP2.
7. Construire une preuve formelle de la validité d'une formule dans le système de Gentzen. Voir, par exemple, l'exercice 0.64.
8. Utiliser la méthode de résolution pour montrer qu'une formule du calcul propositionnel n'est pas satisfiable. Voir, par exemple, le TP3.