CORRIGÉ DE L'ÉPREUVE D'INFORMATIQUE DES MINES - 2002

1 Exercice de logique des propositions

1 - i) F_1 est satisfiable pour $p_1 = p_2 = faux$ par exemple.
ii) F_2 n'est pas satisfiable car p_2 et $\neg p_2$ ne sont jamais vraies simultanément.
iii) Pour satisfaire F_3, il faut avoir $p_2 = vrai$, puis $(\neg p_1 \vee \neg p_2) = vrai$ ce qui impose $p_1 = faux$, mais alors $p_1 \vee \neg p_2 = faux$, donc F_3 n'est pas satisfiable.

2 - H est satisfiable lorsque toutes les variables propositionnelles p_k figurant dans H prennent toutes la valeur $faux$.

3 - On suppose $\exists\, j \in [\![1;n]\!]\ /\ C_j = (p_k)$.
Pour satisfaire H, il faut obligatoirement attribuer la valeur $vrai$ à p_k. Alors, pour $i \neq j$, on note C'_i la clause (non vide) obtenue en supprimant dans C_i le littéral éventuel $\neg p_k$.
On remarque que C'_i contient au plus un littéral positif. En posant $H' = C'_1 \wedge \ldots C'_{j-1} \wedge C'_{j+1} \ldots C'_m$, alors H' est encore une clause de Horn, qui, si elle n'est pas vide (ce qui arrive si $m = 1$), est telle que $\mathcal{V}_{H'} = \mathcal{V}_H \setminus \{p_k\}$.
Lorsque $p_k = vrai$, les clauses C_i et C'_i prennent les mêmes valeurs pour chacune des valuations des autres variables.
En résumé, H est satisfiable si et seulement si H' est satisfiable (résultat valable même si $H' = T$ puisque T est satisfiable par convention).

4 - Soit H une formule de Horn élément de $\mathcal{F}_\mathcal{V}$.
- Si $H = T$, alors H est satisfiable.
- Sinon H est de la forme $H = C_1 \wedge \ldots \wedge C_m$.
  a) Si chaque C_i contient au moins un littéral négatif, alors H est satisfiable.
  b) Sinon, il existe au moins une clause C_j ne contenant pas de littéral négatif.
  H étant une formule de Horn, C_j est nécessairement de la forme $C_j = (p_k)$.
  α) Si une des C_i est égale à $(\neg p_k)$, alors H n'est pas satisfiable.
  β) Sinon, on remplace H par H' comme décrit à la question **3**.

On est alors ramené à étudier la satisfiabilité de H' qui est soit vide, soit formée de $m-1$ clauses et comportant $n-1$ variables propositionnelles.

Cet algorithme s'arrête donc en au plus m étapes.

Notons $C(m,n)$ la complexité de la détermination de la satisfiabilité lorsque qu'il y a m clauses et n variables.
. Au a), on teste pour chaque clause la présence d'un littéral négatif, d'où au plus mn comparaisons.
. En cas d'échec, au b).α), en 2 comparaisons au plus, on peut tester si chacune des clauses C_i est ou non égale à $(\neg p_k)$, d'où au plus $2\,(m-1)$ comparaisons.
. En b).β) on recommence, d'où une complexité en $C(m-1, n-1)$.
Ainsi $\underline{C(m,n) \leq \alpha\, m\, n + \beta\, m + C(m-1,n-1)}$ avec $C(0,n) = C(m,0) = 0$.

$*$ $\underline{\text{Si } n \leq m}$, alors $C(m,n) = \displaystyle\sum_{\ell=1}^{n} \big[C(m-n+\ell,\ell) - C(m-n+\ell-1,\ell-1)\big] \leq \sum_{\ell=1}^{n} (\beta + \alpha\,\ell)\,(m-n+\ell)$,

donc $C(m,n) \leq \displaystyle\sum_{\ell=1}^{n} (\beta+\alpha\,\ell)\,(m+\ell) \leq \beta\, m\, n + (\beta+\alpha\, m) \sum_{\ell=1}^{n} \ell \ +\ \alpha \sum_{\ell=1}^{n} \ell^2 \leq m\,O(n) + m\,O(n^2) + O(n^3) = \underline{O(m\,n^2)}$.

$*$ $\underline{\text{Si } m \leq n}$, alors $C(m,n) = \displaystyle\sum_{k=1}^{m} \big[C(k,n-m+k) - C(k-1,n-m+k-1)\big] \leq \sum_{k=1}^{m} \big(\beta + \alpha\,(n-m+k)\big)\,k$,

donc $C(m,n) \leq \displaystyle\sum_{k=1}^{m} \big(\beta+\alpha\,(n+k)\big)\,k \leq \big(\beta+\alpha\,n\big) \sum_{k=1}^{m} k \ +\ \alpha \sum_{k=1}^{m} k^2 \leq n\,O(m^2) + O(m^3) = \underline{O(n\,m^2)}$.

Ainsi $\boxed{C(m,n) = O\big(\max(m,n) \times \min(m,n)^2\big).}$

5 - $H_3 = (p_2) \wedge (\neg p_1 \vee \neg p_2) \wedge (p_1 \vee \neg p_2) \wedge (p_1 \vee \neg p_2 \vee \neg p_3)$ est une clause de Horn sur $\mathcal{V} = \{p_1, p_2, p_3\}$ puisque chaque clause contient au plus un littéral positif.
On est dans le cas où $C_1 = (p_2)$.
H_3 est donc satisfiable si et seulement si $H'_3 = (\neg p_1) \wedge (p_1) \wedge (p_1 \vee \neg p_3)$ l'est.
Les deux clauses $C'_2 = (\neg p_1)$ et $C'_3 = (p_1)$ ne pouvant pas être satisfaites en même temps, H'_3 n'est pas satisfiable, donc H_3 non plus.

2 Problème d'algorithmique

Première partie : *Sac à dos fractionnaire*

1 - a - Remarque : il n'y a pas unicité de solution maximale lorsqu'on a des objets de même poids et de même utilité.

Si $\sum_{i=0}^{n-1} p_i \leq Q$, alors la solution maximale est obtenue en prenant la totalité des objets, donc pour $x_i = 1$ pour tout $i \in [\![0; n-1]\!]$, ce qui correspond bien dans ce cas à la solution proposée par l'énoncé (cas où $i^* = n$).

On suppose maintenant que $\sum_{i=0}^{n-1} p_i > Q$, donc que $i^* \neq n$.

Vérifions tout d'abord que le choix proposé par l'énoncé est une solution au problème.
En effet, si $i \neq i^*$, alors $x_i = 0$ ou 1, donc $x_i \in [0,1]$.

De plus $\sum_{i=0}^{i^*-1} p_i < Q \leq \sum_{i=0}^{i^*} p_i = \sum_{i=0}^{i^*-1} p_i + p_{i^*}$, donc $0 < Q - \sum_{i=0}^{i^*-1} p_i \leq p_{i^*}$, donc $x_{i^*} \in [0,1]$.

Enfin $\sum_{i=0}^{n-1} p_i\, x_i = \sum_{i=0}^{i^*} p_i + p_{i^*}\, x_{i^*} = Q$. Alors $z = \sum_{i=0}^{i^*-1} u_i + u_{i^*}\, x_{i^*}$.

Considérons des $y_j \in [0,1]$ vérifiant : $\sum_{j=0}^{n-1} p_j\, y_j \leq Q = \sum_{i=0}^{n-1} p_i\, x_i$ et montrons que $\sum_{j=0}^{n-1} u_j\, y_j \leq z = \sum_{i=0}^{n-1} u_i\, x_i$.

On a $\sum_{j=0}^{n-1} p_j\, y_j \leq Q = \sum_{i=0}^{i^*-1} p_i + p_{i^*}\, x_{i^*}$, donc $\sum_{j=i^*+1}^{n-1} p_j\, y_j \leq \sum_{i=0}^{i^*-1} p_i\,(1-y_i) + p_{i^*}\,(x_{i^*} - y_{i^*})$.

Or si $i < i^*$ et $j > i^*$, on a $\frac{u_i}{p_i} \geq \frac{u_{i^*}}{p_{i^*}} \geq \frac{u_j}{p_j}$, d'où en posant $k = \frac{p_{i^*}}{u_{i^*}}$, alors $k\, u_j \leq p_j$ et $p_i \leq k\, u_i$.

Ainsi $\sum_{j=i^*+1}^{n-1} k\, u_j\, y_j \leq \sum_{i=0}^{i^*-1} k\, u_i\,(1-y_i) + k\, u_{i^*}\,(x_{i^*} - y_{i^*})$, donc $\sum_{j=i^*+1}^{n-1} u_j\, y_j \leq \sum_{i=0}^{i^*-1} u_i\,(1-y_i) + u_{i^*}\,(x_{i^*} - y_{i^*})$,

d'où $\sum_{j=0}^{n-1} u_j\, y_j \leq \sum_{i=0}^{i^*-1} u_i + u_{i^*}\, x_{i^*} = \sum_{i=0}^{n-1} u_i\, x_i = z$, ce qui prouve que la solution proposée par l'énoncé est bien maximale.

b - On constate que $\frac{16}{15} > \frac{21}{22} > \frac{19}{20} > \frac{15}{17} > \frac{13}{15} > \frac{7}{9}$. Comme $15 + 22 < 51 < 15 + 22 + 20$, alors $i^* = 2$.

On obtient $x_0 = x_1 = 1$, $x_3 = x_4 = x_5 = 0$ et $x_2 = \frac{51 - 37}{20} = \frac{7}{10}$ et $z = 16 + 21 + \frac{7}{10} \times 19 = 50,3$.

c -

```
let test_in\'egalites_1 () =                      (* version recursive *)
   let rec aux i =
       if i = n-1 then true
                  else (u.(i+1) * p.(i)<= p.(i+1) * u.(i))  && aux (i+1)
   in aux 0 ;;

let test_inegalite_2 () =                        (* version iterative *)
  let ok = ref(true) and i = ref 1 in
    while !ok  && !i < n do
        if u.(!i-1) * p.(!i) < p.(!i-1) * u.(!i) then ok := false;
        i := !i + 1;
    done;
    !ok ;;
```

d -

```
let calcule_indice_1 () =                        (* version recursive *)
   let rec aux i s = if i = n then n
                              else if s >= Q then (i-1)
                                             else aux (i+1) (s + p.(i))
in aux 0 0;;

let calcule_indice_2 () =                        (* version iterative *)
  let s = ref 0 and i = ref 0 in
    while !s < Q && !i <n do
        s := !s + p.(!i);
        i := !i +1;
    done;
    if !s < Q then n else !i -1 ;;
```

2 - a - Appelons *partage* l'algorihme linéaire dont il est question dans l'énoncé.
L'algorithme demandé peut être formulé de la façon suivante :

Fonction récursive sépare(E, Q) =
- si E est un singleton i^*, alors retourner le triplet $(\emptyset, \emptyset, i^*)$;
- sinon $(E', E'') \longleftarrow$ partage(E) ;
  - . Calculer $P = \sum_{i \in E'} p_i$;
  - . si $P < Q$, alors $(F', F'', i^*) \longleftarrow$ sépare$(E'', Q - P)$ et retourner $(I' = E' \cup F', I'' = F'', i^*)$;
  - . si $P \geq Q$, alors $(F', F'', i^*) \longleftarrow$ sépare(E', Q) et retourner $(I' = F', I'' = F'' \cup E'', i^*)$;

fin.

2 - b - Le calcul de P est linéaire (en $n/2$) ainsi que l'appel de la fonction *partage*.
La complexité $C(n)$ vérifie donc une relation de récurrence de la forme : $C(n) = a.n + C(n/2)$ avec $C(1) = 0$.
En supposant que $n = 2^p$ et en posant $u_p = C(2^p)$, on obtient la relation de récurrence : $u_p = a.2^p + u_{p-1}$ avec $u_0 = 0$.

On trouve que : $C(n) = u_p = u_0 + a.\sum_{k=1}^{p} 2^k = 2a.(2^p - 1) \leq 2a.n$, donc l'algorithme est linéaire.

c - La connaissance de I', I'' et i^* dont l'obtention s'effectue en un temps majoré par une fonction linéaire de n permet alors d'obtenir immédiatement une solution maximale en prenant $x_i = 1$ si $i \in I'$, $x_i = 0$ si $i \in I''$ et $x_{i^*} = \dfrac{Q - \sum_{i \in I'} p_i}{p_{i^*}}$.

Seconde Partie : *Sac à dos en* $0 - 1$

3 - Puisque $\big(x_i \in \{0,1\} \Longrightarrow x_i \in [0,1]\big)$, il est évident que le maximum du problème du sac à dos en 0-1 est inférieur ou égal au maximum du problème du sac à dos en fractionnaire ayant les mêmes données numériques.

4 - **Méthode par séparation**
`i` est un indice de position dans les tableaux `p` et `u` qui prend donc les valeurs de 0 à $n - 1$.
La fonction récursive `aux i m` renvoie un couple (ℓ_{i+1}, z_{i+1}) où ℓ_{i+1} est une liste $(x_{i+1}, \ldots x_{n-1})$ d'éléments égaux à 0 ou 1 et correspondant à une solution maximale de valeur $z_{i+1} = \sum_{j=i+1}^{n-1} u_j\, x_j$ lorsqu'on considère uniquement les objets d'indice $j \geq i + 1$ avec la contrainte $\sum_{j=i+1}^{n-1} p_j\, x_j \leq m$.

Le premier appel récursif de `aux` correspondant au cas où $x_i = 0$.

Si $m \leq p_i$, le deuxième appel récursif correspondant à $x_i = 1$ n'a pas lieu.
On examine parmi les deux possibilités $x_i = 0$ ou $x_i = 1$ celle qui donne la meilleure valeur z_i et on renvoie le couple (ℓ_i, z_i) où $\ell_i = x_i :: \ell_{i+1}$.

```
let meilleur () =
   let rec aux  i m =
     if i = n-1 then if p.(i) <= m then ([1],u.(i)) else ([0],0)
     else
         begin
           let (l0,z0) = aux (i+1) m                    (* utile pour le cas o\'u x.(i)=0 *)
           in if m <= p.(i)
              then  (0::l0,z0)
              else let(l1,z1) = aux (i+1) (m-p.(i)) in  (* utile pour le cas o\'u x.(i)=1 *)
                   if z1 + u.(i) > z0
                   then (1::l1,z1+u.(i))
                   else (0::l0,z0)
         end
   in aux 0 Q ;;
```

5 - Le pire des cas est celui où tous les n-uplets de $\{0,1\}^n$ sont examinés, d'où une complexité exponentielle en $\boxed{2^n.}$

6 - Méthode par séparation et évaluation

a. Pour le sommet C ($x_0 = x_1 = 1$), on doit rendre maximum $z = 37 + 19x_2 + 15x_3 + 13x_4 + 7x_5$ avec la contrainte $20x_2 + 17x_3 + 15x_4 + 9x_5 \leq 51 - 37 = 14$.
La seule façon non triviale avec les x_i dans $\{0,1\}$ est $x_2 = x_3 = x_4 = 0$ et $x_5 = 1$.
On obtient donc $(1,1,0,0,0,1)$ et $z = 44$.

b. Pour le sommet E, on cherche le maximum de $z = 35 + 15x_3 + 13x_4 + 7x_5$ avec la contrainte $17x_3 + 15x_4 + 9x_5 \leq 16$.
Les deux possibilités non triviales sont $(1,0,1,0,0,1)$ et $z = 42$ ou $\underline{(1,0,1,0,1,0) \text{ et } z = 48}$ qui est la meilleure.

c. On a $\dfrac{15}{17} > \dfrac{13}{15} > \dfrac{7}{9}$.
La méthode vue au **1-a** donne $i^* = 5$ car $17 + 15 < 36 < 17 + 15 + 9$. On obtient $x_3 = x_4 = 1$ et $x_5 = \dfrac{36 - 32}{9} = \dfrac{4}{9}$.
Une évaluation du sommet F est donc d'après la question **3** : $z \leq 16 + 15 + 13 + \dfrac{4}{9} \times 7 < 47,2$.

d. On doit maximiser $z = 21x_1 + 19x_2 + 15x_3 + 13x_4 + 7x_5$ avec les contraintes $22x_1 + 20x_2 + 17x_3 + 15x_4 + 9x_5 \leq 51$ et les x_i dans $[0,1]$.
On trouve $i^* = 3$, $x_1 = x_2 = 1$, $x_4 = x_5 = 0$ et $x_3 = \dfrac{51 - 42}{17} = \dfrac{9}{17}$.
Une évaluation du sommet G est donc d'après la question **3** : $z \leq 21 + 19 + \dfrac{9}{17} \times 15 < 47,95$.

e. La solution optimale du problème $(\mathcal{P})$ est donc $\boxed{(1,0,1,0,1,0) \text{ et } z = 48.}$

3 Exercice sur les automates finis

1 - Soit $\mathcal{A} = \langle Q, A, E, I, T\rangle$ un automate fini reconnaissant le langage L.
On construit l'automate normalisé $\mathcal{A}' = \langle Q', A, E', \{i\}, \{t\}\rangle$ de la façon suivante :
. on choisit deux éléments notés i et t n'appartenant pas à Q et on pose $Q' = Q \cup \{i, t\}$.
. E' est la réunion de E et de :
- l'ensemble des transitions $i \xrightarrow{c} q$ telles que $\exists\, p \in I \;/\; p \xrightarrow{c} q$
- l'ensemble des transitions $p \xrightarrow{c} t$ telles que $\exists\, q \in T \;/\; p \xrightarrow{c} q$
- l'ensemble des transitions $i \xrightarrow{c} t$ telles que $\exists\, (p, q) \in I \times T \;/\; p \xrightarrow{c} q$.

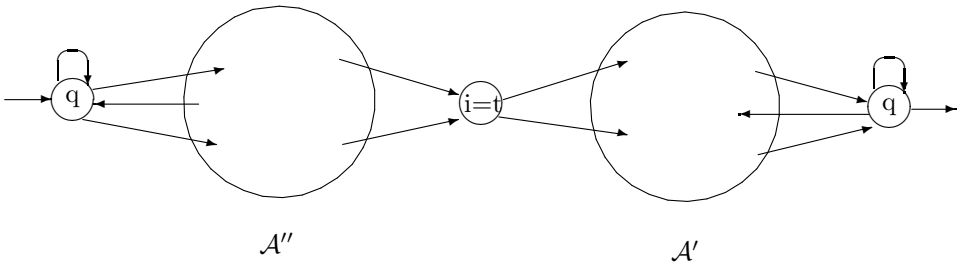
Montrons que $\underline{\mathrm{L}(\mathcal{A}) \setminus \{1_{A^*}\} = \mathrm{L}(\mathcal{A}')}$.

• Soit $f \in \mathrm{L}(\mathcal{A})$ et $f \neq 1_{A^*}$. f s'écrit $f = a_1 \dots a_n$.
. Si $n = 1$, alors $\exists\, (p, q) \in I \times T \;/\; p \xrightarrow{a_1} q$ dans $\mathcal{A}$. On a aussi $i \xrightarrow{a_1} t$ dans $\mathcal{A}'$, donc $f = a_1 \in \mathrm{L}(\mathcal{A}')$.
. Si $n \geq 2$, alors il existe un chemin réussi dans $\mathcal{A}$ étiqueté par f de la forme : $p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \dots p_{n-1} \xrightarrow{a_n} p_n$ avec $p_0 \in I$ et $p_n \in T$. On constate que $i \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \dots p_{n-1} \xrightarrow{a_n} t$ est un chemin réussi dans $\mathcal{A}'$ étiqueté par f, donc $f \in \mathrm{L}(\mathcal{A}')$.

• Réciproquement, soit $f \in \mathrm{L}(\mathcal{A}')$.
Alors $f \neq 1_{A^*}$ car $i \neq t$, donc f s'écrit $f = a_1 \dots a_n$.
. Si $n = 1$, alors $i \xrightarrow{a_1} t$ dans $\mathcal{A}'$, donc $\exists\, (p, q) \in I \times T \;/\; p \xrightarrow{a_1} q$ dans $\mathcal{A}$, donc $f \in \mathrm{L}(\mathcal{A}) \setminus \{1_{A^*}\}$.
. Si $n \geq 2$, alors il existe un chemin réussi dans $\mathcal{A}'$ étiqueté par f de la forme : $i \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \dots p_{n-1} \xrightarrow{a_n} t$.
Puisque $(i, a_1, p_1) \in E'$ et $(p_{n-1}, a_n, t) \in E'$, il existe $p_0 \in I$ et $p_n \in T$ tels que $(p_0, a_1, p_1) \in E$ et $(p_{n-1}, a_n, p_n) \in E$.
On constate que $p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \dots p_{n-1} \xrightarrow{a_n} p_n$ est un chemin réussi dans $\mathcal{A}$ étiqueté par f, donc $f \in \mathrm{L}(\mathcal{A}) \setminus \{1_{A^*}\}$.

2 - Notons $\mathcal{A}'$ le sous-graphe de $\mathcal{A}$ dont les arcs sont ceux figurant dans tous les chemins dans $\mathcal{A}$ joignant i à q (q pouvant figurer plusieurs fois dans un tel chemin).
Notons de même $\mathcal{A}''$ le sous-graphe de $\mathcal{A}$ dont les arcs sont ceux figurant dans tous les chemins dans $\mathcal{A}$ joignant q à t.

On considère le graphe $\mathcal{A}_q$ obtenu en mettant "bout à bout" $\mathcal{A}'$ et $\mathcal{A}''$ et en superposant i et t.



Remarque : $\mathcal{A}_q$ n'est pas normalisé en général.

Un chemin dans $\mathcal{A}_q$ est la concaténation $q \xrightarrow{a_1} \dots \xrightarrow{a_m} t = i \xrightarrow{b_1} \dots \xrightarrow{b_n} q$ d'un chemin $i \xrightarrow{b_1} \dots \xrightarrow{b_n} q$ dans $\mathcal{A}'$ et d'un chemin $i \xrightarrow{b_1} \dots \xrightarrow{b_n} q$ dans $\mathcal{A}''$.
On note ici l'importance que i soit non rentrant et t non sortant dans $\mathcal{A}$.
Il en résulte immédiatement que $\mathrm{L}(\mathcal{A}_q) = G_q$, donc G_q est reconnaissable.

3 - Si $1_{A^*} \in L$, alors $\mathrm{Conj}(1_{A^*}) = \{1_{A^*}\}$ est reconnaissable.
Si L contient des mots f de longueur 1, alors $\mathrm{Conj}(f) = \{f\}$ est reconnaissable.
Notons L' l'ensemble des mots de L de longueur ≥ 2 et $Q^* = Q \setminus \{i, t\}$. On remarque que $\mathrm{Conj}(L') = \bigcup_{q \in Q^*} G_q$.
Finalement $\underline{\mathrm{Conj}(L) \text{ apparaît comme réunion finie de langages reconnaissables}}$, donc est lui-même reconnaissable.

* + * + * + * + * + * + *