Licence Informatique ISTIC
Module LOG *2015-2016*

LOG TD 3

Ce TD est une préparation au problème de la satisfaisabilité des formules du calcul propositionnel, et en particulier à l'étude de la résolution que nous verrons plus en détail au TD suivant.

Exercice 1 (* préliminaire à l'Exercice 2)
On considère le résultat suivant :

Théorème 1 (Complétude fonctionnelle) *Supposons* $Prop = \{p_1, p_2, \ldots, p_n\}$ *fini. Soit* Val *l'ensemble des valuations sur Prop. Toute fonction* f *de* Val *dans* $\{0,1\}$ *est la valeur de vérité d'une formule* φ_f *sur Prop.*

On remarque que la réciproque est évidente : toute formule φ sur un ensemble de variables propsitionnelles $Prop = \{p_1, p_2, \ldots, p_n\}$ correspond à une fonction de f_φ de Val^n dans $\{0,1\}$ définit par $f_\varphi(\nu) = \nu(\varphi)$, pour tout $\nu \in Val$.

Démontrer le Théorème 1 par récurrence sur le nombre n de variables dans Prop.

Exercice 2 (*)

2.1 *Rappeler la définition d'une formule forme normale conjonctive (FNC).*

2.2 *Démontrer par récurrence sur le nombre* n *de variables propositionnelles dans la formule le théorème vu en cours :*

Théorème 2 *Pour toute* $\varphi \in \mathcal{F}_{cp}$*, il existe* $\tilde{\varphi} \in \mathcal{F}_{cp}$ *en FNC telle que* $\varphi \equiv \tilde{\varphi}$*.*

On pourra noter $\varphi(p_1, \ldots, p_n)$ *pour indiquer que les variables propositionnelles de* φ *sont* $p_1, \ldots, p_n$*.*

Exercice 3
* Trouver deux formules (vraiment) différentes en FNC qui sont équivalentes.

Mise en FNC

Exercice 4 (Un algorithme récursif)
On utilise "CNF" pour "conjunctive normal form".

Algorithm 1 CNF(φ)

```
1: φ ← φ
2: φ ← RemoveImpl(φ)                          ▷ on supprime les implications
3: φ ← PushNeg(φ)                  ▷ on pousse les négations vers les propositions
4: switch φ do
5:     case φ est un litéral
6:         return φ
7:     case φ = ψ1 ∧ ψ2
8:         return CNF(ψ1) ∧ CNF(ψ2)
9:     case φ = ψ1 ∨ ψ2
10:        return Distrib(CNF(ψ1), CNF(ψ2))
11: return
```

où

Algorithm 2 Distrib(ϕ_1, ϕ_2)

Require: ϕ_1 et ϕ_2 sont en CNF

```
1: switch φ1, φ2 do
2:     case φ1 = φ11 ∧ φ12                          ▷ φ1 a au moins 2 clauses
3:         return Distrib(φ11, φ2) ∧ Distrib(φ12, φ2)
4:     case φ2 = φ21 ∧ φ22                          ▷ φ2 a au moins 2 clauses
5:         return Distrib(φ1, φ21) ∧ Distrib(φ1, φ22)
6:     case autre                          ▷ ni φ1 ni φ2 ne sont des conjonctions
7:         return φ1 ∨ φ2
```

4.1 *Décrire le principe d'un algorithme pour* RemoveImpl(ϕ) *et d'un algorithme pour* PushNeg(ϕ) *(sans détail).*

4.2 *Montrer que* CNF(φ) *termine.*

4.3 *Montrer que* CNF(φ) *est correct.*

Le problème SAT

Exercice 5

5.1 *Énoncer le problème de satisfaisabilité d'une formule du calcul propositionel.*

5.2 *Proposer un algorithme naïf pour résoudre ce problème.*

Pour le problème SAT, on se place maintenant dans le cas particulier des clauses de Horn.

Définition 1 (Clauses de Horn) *Une* clause de Horn *est une clause contenant au plus un litéral positif. Les clauses de Horn sans littéraux positifs sont appelées* négative pure *et les autres des* implication.

5.3 *Expliquer la terminologie "négative pure" et "implication" de la Définition 1.*

5.4 *On considère l'algorithme suivant, écrit en pseudo code de très haut niveau :*

Algorithm 3 Fonction SAT-HORN-GLOUTON($\mathcal{C}$)

```
1: Initialiser toutes les variables de l'ensemble des clauses de C à faux
2: while une implication de C n'est pas satisfaite do
3:     mettre la variable droite de cette implication à vrai
4: end while
5: if toutes les clauses négatives pures de C sont satisfaites par la valuation then
6:     return cette valuation
7: else
8:     return "la formule n'est pas satisfaisable"
9: end if
```

Analyser cet algorithme (terminaison et correction). Qu'en conclure ?