L2/LD — TP 4 — Simplification de formules propositionnelles

Objectif : L'objectif de ce TP est d'implanter en OCaml des règles de simplification de formules de la logique des propositions.

Travail à rendre

Vous devez rendre dans un unique fichier source vos réponses à tous les exercices de ce sujet. Le nom du fichier doit être composé des 8 premières lettres de votre nom de famille, sans accents. Le fichier doit être complet et exécutable sans modification (lourdes pénalités sinon). Ce fichier doit contenir un commentaire indiquant vos nom, prénom et groupe de TP, le code OCaml demandé et des commentaires explicatifs.

L'évaluation de ce travail compte pour la note de TP du module. Le travail doit être strictement personnel.

Déposez votre fichier sur le site du module sur moodle :

`http://moodle.univ-fcomte.fr/course/view.php?id=2594`

dans le devoir intitulé "Rendu du TP 4", de préférence avant la fin de la séance, ou sinon avant 23 heures 30 la veille du jour de votre séance de TP de la semaine 42 (le 12 ou le 16 octobre).

Introduction (*rappels et compléments de cours*)

Une formule propositionnelle est une *tautologie* si elle est vraie pour toute interprétation (valuation) de ses variables. On peut détecter qu'une formule est une tautologie en construisant sa table de vérité. Cependant, cette méthode est souvent plus longue que nécessaire. En effet, on peut conclure plus vite, en observant la formule et en la simplifiant. Par exemple, si une formule est de la forme $F \vee \neg F$, alors c'est une tautologie, quelle que soit la sous-formule F. Il est donc inutile d'interpréter F. Ce principe est à la base de certaines méthodes syntaxiques. Dans ce TP, on va implanter des simplifications de formules issues d'équivalences entre formules propositionnelles.

1 Elimination des constantes

$\neg\textsc{Faux} \Leftrightarrow \textsc{Vrai}$	(1)	$\neg\textsc{Vrai} \Leftrightarrow \textsc{Faux}$	(2)
$\textsc{Faux} \vee F \Leftrightarrow F$	(3)	$\textsc{Vrai} \vee F \Leftrightarrow \textsc{Vrai}$	(4)
$\textsc{Faux} \wedge F \Leftrightarrow \textsc{Faux}$	(5)	$\textsc{Vrai} \wedge F \Leftrightarrow F$	(6)

Figure 1 – Quelques tautologies

Ecrire la fonction `elimVraiFaux : pf -> pf` qui applique toutes les équivalences de la figure 1 de gauche à droite pour éliminer toutes les constantes VRAI et FAUX dans une formule propositionnelle. Par exemple, l'application de cette fonction à la formule $f = P \wedge (\text{FAUX} \vee (Q \wedge \text{VRAI}))$ doit donner la formule $P \wedge Q$.

Supposons que les formules propositionnelles suivantes sont définies :

— $f = P \wedge (\text{FAUX} \vee (\text{VRAI} \wedge Q))$
— $f1 = (\text{VRAI} \wedge \text{FAUX}) \vee (\text{VRAI} \wedge (\text{FAUX} \wedge ((\text{VRAI} \wedge \text{VRAI}) \wedge \text{FAUX})))$
— $f2 = (((\text{P} \wedge \text{FAUX}) \wedge (\text{FAUX} \vee \text{VRAI})) \wedge ((\text{P} \vee \text{VRAI}) \vee (\text{VRAI} \wedge \text{Q}))) \wedge \text{VRAI}$
— $f3 = \text{FAUX} \vee ((\text{R} \wedge \text{VRAI}) \wedge \text{P})$

Vous devez obtenir :

À taper : `elimVraiFaux f;;`
Réponse : `- : pf = Et (Atome "P", Atome "Q")`
À taper : `elimVraiFaux f1;;`
Réponse : `- : pf = Faux`
À taper : `elimVraiFaux f2;;`
Réponse : `- : pf = Faux`
À taper : `elimVraiFaux f3;;`
Réponse : `- : pf = Et (Atome "R", Atome "P")`

2 Idempotence

Ecrire une fonction qui applique les équivalences

$$F \vee F \Leftrightarrow F \quad (7) \;\Big|\; F \wedge F \Leftrightarrow F \quad (8)$$

à une formule quelconque et à toutes ses sous-formules.

Tester cette fonction sur une formule qui contient au moins trois cas d'application de ces équivalences.

À taper :
```
let f4 = Et(Ou(Faux,Et(Vrai,Atome("Q"))),Et(Ou(Faux,Et(Vrai,Atome("Q"))),
  Ou(Faux,Et(Vrai,Atome("Q")))));;
```
Réponse :
```
val f4 : pf =
 Et (Ou (Faux, Et (Vrai, Atome "Q")),
  Et (Ou (Faux, Et (Vrai, Atome "Q")), Ou (Faux, Et (Vrai, Atome "Q"))))
```

À taper : `affichePF f4;;`
Réponse :
```
(Faux v (Vrai ^ Q)) ^ ((Faux v (Vrai ^ Q)) ^ (Faux
   v (Vrai ^ Q)))- : unit = ()
```
À taper : `elimIdempotence f4;;`
Réponse : `- : pf = Ou (Faux, Et (Vrai, Atome "Q"))`

Indication : En OCaml, un filtre ne peut pas contenir deux fois la même variable (on dit que le filtrage doit être *linéaire*). Par exemple, on ne peut pas écrire `| Et(f,f) -> f`. On peut contourner cette limitation en écrivant `| Et(f,g) -> if f = g then .. else ...`

3 Forme normale négative

Une formule est dite en *forme normale négative* lorsque toutes les négations s'appliquent à un atome.

1. Ecrire une fonction qui vérifie qu'une formule est en forme normale négative.

```
À taper :   verifFNN (Neg(Vrai));;
Réponse :   - : bool = false
À taper :   verifFNN (Et(Neg(Atome "D"), Faux));;
Réponse :   - : bool = true
À taper :   verifFNN (Neg(Neg(Atome "D")));;
Réponse :   - : bool = false
À taper :   verifFNN (Neg(Et(Atome "D", Atome "P")));;
Réponse :   - : bool = false
```

2. Utiliser les équivalences (1), (2) et

$$\begin{aligned} \neg\neg F &\Leftrightarrow F && (9)\\ \neg(F \wedge G) &\Leftrightarrow (\neg F) \vee (\neg G) && (10)\\ \neg(F \vee G) &\Leftrightarrow (\neg F) \wedge (\neg G) && (11) \end{aligned}$$

pour définir une fonction qui met toute formule en forme normale négative.

```
À taper :   formeNN(Neg(Vrai));;
Réponse :   - : pf = Faux
À taper :   # formeNN(Et(Neg(Atome "D"), Faux));;
Réponse :   - : pf = Et (Neg (Atome "D"), Faux)
À taper :   formeNN(Neg(Neg(Atome "D")));;
Réponse :   - : pf = Atome "D"
À taper :   formeNN(Neg(Et(Atome "D", Atome "P")));;
Réponse :   - : pf = Ou (Neg (Atome "D"), Neg (Atome "P"))
```

3. Tester ces deux fonctions sur la formule $P \wedge \neg(\textsc{Faux} \vee \neg(\neg Q \wedge \textsc{Vrai}))$.

4 Composition de simplifications

On peut composer les fonctions précédentes pour obtenir encore plus de simplifications.

1. Ecrire une fonction qui compose l'élimination des constantes, l'idempotence et la mise en forme normale négative de façon optimale.
2. Appliquer cette fonction à une formule pour laquelle le résultat est plus simple que si l'on n'applique qu'une seule des deux fonctions.

On considère désormais cette fonction pour mettre une formule en forme normale négative.

5 Forme normale conjonctive

Une formule est dite en forme normale conjonctive (CNF, pour *Conjunctive Normal Form*) quand elle est de la forme

$$D_1 \wedge \ldots \wedge D_n$$

où chaque D_i est une disjonction de littéraux. Un *littéral* est soit une proposition atomique, soit la négation d'une proposition atomique.

Ecrire une fonction `isCNF` qui retourne *vrai* si une formule est sous forme CNF.

Par exemple, pour la formule propositionnelle $f7 = (P \vee Q) \wedge ((P \vee \neg R) \wedge (P \vee (Q \vee R)))$, on doit avoir :

```
À taper :   affichePF f7;;
Réponse :   (P v Q) ^ ((P v Non(R)) ^ (P v (Q v R)))- : unit = ()
À taper :   isCNF f7;;
Réponse :   - : bool = true
```

6 Mise en forme normale conjonctive

La première étape pour mettre une formule en forme normale conjonctive consiste à mettre la formule sous forme normale négative.

Pour que les disjonctions ne portent plus que sur des littéraux, la deuxième étape consiste à appliquer systématiquement la règle de réécriture suivante :

$$A \vee (B \wedge C) \quad \rightarrow \quad (A \vee B) \wedge (A \vee C)$$

qui distribue les disjonctions ($\vee$) sur les conjonctions ($\wedge$).

1. Dans ce but, coder d'abord une fonction `distrib : pf -> pf` qui applique cette règle de réécriture chaque fois qu'un $\vee$ précède un $\wedge$.

   On considère la formule propositionnelle $f8 = (P \vee ((\neg R \vee (Q \wedge R)) \wedge Q))$.

   ```
   À taper :   affichePF f8;;
   Réponse :   P v ((Non(R) v (Q ^ R)) ^ Q)- : unit = ()
   À taper :   affichePF (distrib f8);;
   Réponse :   (P v ((Non(R) v Q) ^ (Non(R) v R))) ^ (P v Q)- : unit = ()
   ```

2. En utilisant la fonction `distrib`, écrire une fonction `formCNF` qui permet de faire la mise en forme CNF.
3. Reprendre les formules propositionnelles déjà définies et déterminer leur forme normale conjonctive.