OPTION INFORMATIQUE
Devoir surveillé 3
Corrigé

1 Cherchez les Clés du Paradis (d'après CCP 2018)

2 Préparation

1. $A \vee \bot \equiv A$
   $A \vee \top \equiv \top$
   $A \wedge \bot \equiv \bot$
   $A \wedge \top \equiv A$
   $A \vee \overline{A} \equiv \top$
   $A \wedge \overline{A} \equiv \bot$
   $A \Rightarrow \bot \equiv \overline{A}$
   $A \Rightarrow \top \equiv \top$
   $\bot \Rightarrow A \equiv \top$
   $\top \Rightarrow A \equiv A$
2. $A \oplus B \equiv (A \vee B) \wedge (\overline{A} \vee \overline{B})$ qui est bien une forme normale conjonctive.
   $A \oplus B \equiv (A \wedge \overline{B}) \vee (\overline{A} \wedge B)$ qui est bien une forme normale disjonctive.

3 Première épreuve

3. $B_1 = P_1 \vee P_2$
   $B_2 = \overline{P_1}$
4. On note ϕ l'affirmation de l'animateur.

$$\begin{aligned} \phi &= (B_1 \wedge B_2) \vee (\overline{B_1} \wedge \overline{B_2}) \\ &\equiv ((P_1 \vee P_2) \wedge \overline{P_1}) \vee (\overline{P_1} \wedge \overline{P_2} \wedge P_1) \\ &\equiv (P_1 \wedge \overline{P_1}) \vee (P_2 \wedge \overline{P_1}) \\ &\equiv P_2 \wedge \overline{P_1} \end{aligned}$$

5. D'après le calcul précédent, il faut donc ouvrir la boîte numéro 2.

4 Deuxième épreuve

6. $B_1 = \overline{P_1} \oplus P_2$
   $B_2 = P_1$
7. On a $\phi = (B_1 \wedge B_2) \vee (\overline{B_1} \wedge \overline{B_2})$
   On dresse la table de vérité de ϕ :

P_1	P_2	B_1	B_2	ϕ
F	F	V	F	F
F	V	F	F	V
V	F	F	V	F
V	V	V	V	V

   Les valuations satisfaisant ϕ sont donc $\{P_1 \mapsto F, P_2 \mapsto V\}$ et $\{P_1 \mapsto V, P_2 \mapsto V\}$.
   Il faut donc ouvrir la boîte numéro 2.

5 Troisième épreuve

8. On remarque qu'on peut obtenir $\bot \equiv A \oplus A$
   On peut donc obtenir la négation : $\overline{A} \equiv A \Rightarrow \bot \equiv A \Rightarrow (A \oplus A)$.
   Enfin, on peut écrire une disjonction comme une implication :
   $A \vee B \equiv \overline{A} \Rightarrow B$
   On sait que $\{\vee, \neg\}$ est un système complet de connecteurs, il ne reste plus qu'à démontrer par induction structurelle que toute formule ϕ n'utilisant que les connecteurs $\vee$ et $\neg$ est équivalente à une formule n'utilisant que $\oplus$ et $\Rightarrow$:
   - Si ϕ est une variable, c'est immédiat ;
   - Si $\phi = \neg\phi'$.
     Supposons que ϕ' soit équivalente à ϕ'' composée de $\oplus$ et $\Rightarrow$.
     On a alors $\phi \equiv \phi'' \Rightarrow (\phi'' \oplus \phi'')$ composée de $\oplus$ et $\Rightarrow$.
   - Si $\phi = \phi_1 \vee \phi_2$.
     Supposons que ϕ_1, ϕ_2 sont équivalentes à ϕ'_1, ϕ'_2 composées de $\oplus$ et $\Rightarrow$.
     On a $\phi \equiv \overline{\phi'_1} \Rightarrow \phi'_2 \equiv (\phi'_1 \Rightarrow (\phi'_1 \oplus \phi'_1)) \Rightarrow \phi'_2$.composée de $\oplus$ et $\Rightarrow$.

   En conclusion, toute formule composée de $\vee$ et $\neg$ est équivalente à une formule composée de $\oplus$ et $\Rightarrow$, donc $\{\oplus, \Rightarrow\}$ est un système complet de connecteurs.

6 Implémentation en Caml

1. ```
   let phi = Imp(V 'A', Imp(Imp(V 'A', V 'B'), V 'B'));;
   ```
2. ```
   let rec interpretation f v =
       match f with
           V(a) -> v a
           |Neg(f') -> not (interpretation f' v)
           |Et(g,d) -> (interpretation g v) && (interpretation d v)
           |Ou(g,d) -> (interpretation g v) || (interpretation d v)
           |Imp(g,d) -> (interpretation g v) <= (interpretation d v)
           |Eq(g,d) -> (interpretation g v) = (interpretation d v);;
   ```
3. Une clause est une disjonction de littéraux, où un littéral est une variable ou sa négation.
   Une forme normale conjonctive est une conjonction de clauses.
4. ```
   let rec est_une_clause f =
       match f with
           V _ -> true
           |Neg (V _) -> true
           |Ou(f1,f2) -> est_une_clause f1 && est_une_clause f2
           |_ -> false;;
   ```
5. ```
   let rec est_une_fnc f =
       match f with
           V _ -> true
           |Neg (V _) -> true
           |Ou(f1,f2) -> est_une_clause f1 && est_une_clause f2
           |Et(f1,f2) -> est_une_fnc f1 && est_une_fnc f2
           |_ -> false;;
   ```
6. `est_une_clause` est une fonction récursive sur un arbre faisant dans le pire cas un appel récursif sur chaque fils et des calculs en $O(1)$, d'où une complexité totale linéaire en la taille de l'arbre, ie la longueur de la formule.
   On a la même relation de récurrence pour `est_une_fnc` appliquée sur une conjonction, et une complexité linéaire d'après le cas précédent si `est_une_fnc` est appliquée sur une conjonction (et du $O(1)$ dans les autres cas), d'où une complexité totale en $O(n)$.

7.
$$\begin{aligned} & (A \to B) \leftrightarrow (B \to C) \\ \equiv\ & ((A \wedge \neg B) \vee (\neg B \vee C)) \wedge ((B \wedge \neg C) \vee (\neg A \vee B)) \\ \equiv\ & (A \vee \neg B \vee C) \wedge ((\neg B \vee C) \wedge (B \vee \neg A) \wedge (\neg C \vee \neg A \vee B) \end{aligned}$$

qui est une FNC, mais qu'on peut encore simplifier en $(\neg B \vee C) \wedge (B \vee \neg A)$.

8.
```
let rec substitution phi a psi =
    match phi with
        V(b) -> if a=b then psi else phi
        |Neg(f') -> Neg(substitution f' a psi)
        |Et(g,d) -> Et(substitution g a psi , substitution d a psi)
        |Ou(g,d) -> Ou(substitution g a psi , substitution d a psi)
        |Imp(g,d) -> Imp(substitution g a psi , substitution d a psi)
        |Eq(g,d) -> Eq(substitution g a psi , substitution d a psi);;
```

On obtient la même relation de récurrence que précédemment, d'où une complexité linéaire en la taille de la formule `phi`.

9. (a) $V(s1) \Leftrightarrow \neg V(s2)$

(b)
```
let formule_graphe_biparti g =
        let n = Array.length g in
        let f = ref (V 0) in
        let rec aux u l =
                match l with
                    [] -> ()
                    |v::q -> if u<v then f := Et(!f, Eq(V u, Neg(V v))) ; aux u q
        in
        for s = 0 to n-1 do
            aux s g.(s)
        done;
        !f;;
```