Exercice 1.

On commence par simplifier la formule φ_2 :

$$\begin{aligned}
\varphi_2 &= x_1 \Rightarrow (\neg(x_2 \texttt{ NAND } \overline{x_1}) \vee (x_1 \Leftrightarrow x_3)) \\
&\equiv \overline{x_1} \vee (\neg\neg(x_2 \wedge \overline{x_1}) \vee (x_1 \wedge x_3) \vee (\overline{x_1} \wedge \overline{x_3})) \\
&\equiv \overline{x_1} \vee (x_2 \wedge \overline{x_1}) \vee (x_1 \wedge x_3) \vee (\overline{x_1} \wedge \overline{x_3}) \\
&\equiv \overline{x_1} \vee (x_1 \wedge x_3) \\
&\equiv \overline{x_1} \vee x_3.
\end{aligned}$$

On peut maintenant établir les tables de vérité de ces formules :

x_1	x_2	x_3	φ_1	φ_2	φ_3
0	0	0	0	1	0
0	0	1	1	1	0
0	1	0	1	1	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	0	1	0
1	1	0	0	0	0
1	1	1	1	1	0

Question 1 – Voici les formes normales disjonctives canoniques :

$$\begin{aligned}
\varphi_1 &\equiv (\overline{x_1} \wedge \overline{x_2} \wedge x_3) \vee (\overline{x_1} \wedge x_2 \wedge \overline{x_3}) \vee (x_1 \wedge \overline{x_2} \wedge \overline{x_3}) \vee (x_1 \wedge x_2 \wedge x_3) \\
\varphi_2 &\equiv (\overline{x_1} \wedge \overline{x_2} \wedge \overline{x_3}) \vee (\overline{x_1} \wedge \overline{x_2} \wedge x_3) \vee (\overline{x_1} \wedge x_2 \wedge \overline{x_3}) \vee (\overline{x_1} \wedge x_2 \wedge x_3) \\
&\qquad \vee (x_1 \wedge \overline{x_2} \wedge x_3) \vee (x_1 \wedge x_2 \wedge x_3) \\
\varphi_3 &\equiv \perp
\end{aligned}$$

Notez que pour φ_3, on a pris la convention que $\perp$ est une disjonction vide, donc c'est bien une forme normale disjonctive (avec 0 clause).

Question 2 – Voici les formes normales conjonctives canoniques :

$$\begin{aligned}
\varphi_1 &\equiv (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \\
\varphi_2 &\equiv (\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \\
\varphi_3 &\equiv (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \\
&\qquad \wedge (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_2 \vee \overline{x_3}) \wedge (x_1 \vee x_2 \vee x_3)
\end{aligned}$$

Exercice 2. Formules logiques en OCaml

Question 1 –

```
let tab_formules = [|
    Et (Var 2, Non (Var 1));
    Non (Et (Var 1, Var 2));
    Ou (Non (Var 2), Et (Var 2, Non (Var 1)));
    Et (Ou (Var 1, Var 2), Ou (Non (Var 1), Non (Var 2)));
    Ou (Et (Non (Var 1), Var 2), Et (Var 0, Non (Var 2)))
  |];;
```

Question 2.a –

```
let rec eval d f = match f with
  | V -> 1
  | F -> 0
  | Var i -> d.(i)
  | Non f -> 1 - eval d f
  | Ou (f1, f2) -> max (eval d f1) (eval d f2)
  | Et (f1, f2) -> min (eval d f1) (eval d f2);;
```

Question 2.b –

Solution 1. L'arbre des appels récursifs a la même forme que l'arbre donné en entrée (en particulier il a le même nombre de noeuds). Sans compter les appels récursifs, chaque appel à `eval` s'exécute en temps constant. Finalement :

$$\boxed{\text{Le temps d'exécution est en } \Theta(n_A) \text{ avec } n_A \text{ le nombre de noeuds dans l'arbre.}}$$

Solution 2. On note $T(A)$ le temps d'exécution pour un arbre A et n_A son nombre de noeuds. Chaque appel à la fonction s'exécute en temps constant plus le temps d'exécution de zéro, un ou deux appels récursifs. On a trois cas :

- Cas 1 : l'arbre est réduit à une feuille. Il n'y a pas d'appel récursif.
- Cas 2 : la racine est étiquetée par $\neg$. Si on note A' l'arbre privé de sa racine, alors $n_A = n_{A'} + 1$.
- Cas 3 : la racine est étiquetée par $\vee$ ou $\wedge$. Si on note G et D les sous-arbres gauche et droit, alors $n_A = 1 + n_G + n_D$.

Il existe donc une constante c telle que :

$$\begin{cases} T(A) \leqslant c & \text{dans le cas 1} \\ T(A) \leqslant c + T(A') & \text{dans le cas 2} \\ T(A) \leqslant c + T(G) + T(D) & \text{dans le cas 3} \end{cases}$$

Par une récurrence immédiate sur n_A :

$$T(A) \leqslant n_A \times c$$

Donc $T(A) = \mathcal{O}(n_A)$. De même $T(A) = \Omega(n_A)$. Finalement le temps d'exécution est en $\Theta(n_A)$.

Question 3.a –

```
let sont_equiv f1 f2 n =
  let d = Array.make n 0 in
  let rec aux i =
    if i = n then eval d f1 = eval d f2
    else if not (aux (i+1)) then false
    else begin
        d.(i) <- 1;
        let b = aux (i+1) in
        d.(i) <- 0;
        b
      end in
  aux 0;;
```

Question 3.b – Soit $m = \max(m_1, m_2)$ où m_1 et m_2 sont les tailles des deux formules en argument.

Solution 1. Dans le pire cas (les deux formules sont sémantiquement équivalentes), l'arbre des appels récursifs est binaire complet de hauteur n. De plus, sans compter les appels récursifs un appel à `aux` s'exécute :

- En temps constant si `i < n`.
- En temps $\Theta(m)$ si `i = n`.

Dans l'arbre des appels récursifs, il y a 2^n noeuds à profondeur n et $\sum_{p=0}^{n-1} 2^p = 2^n - 1$ noeuds à profondeur $p < n$. Au total :

$$\boxed{\text{Le temps d'exécution est en } \mathcal{O}(2^n) + \mathcal{O}(m2^n) = \mathcal{O}(m2^n)}$$

Solution 2. On souhaite déterminer la complexité d'un appel à « `aux 0` ». On note $T(i)$ le temps d'exécution d'un appel à « `aux i` ». Alors, il existe une constante c telle que :

$$\begin{cases} T(n) \leqslant c \times m \\ T(i) \leqslant 2T(i+1) + c & \text{pour } 0 \leqslant i < n \end{cases}$$

On a donc :

$$\begin{aligned} T(0) &\leqslant cm2^n + \sum_{i=0}^{n} c \times 2^i \\ &= \mathcal{O}(m2^n) + \mathcal{O}(2^n) \\ &= \mathcal{O}(m2^n) \end{aligned}$$

Finalement le temps d'exécution est en $\mathcal{O}(m2^n)$.

Question 4 –

```
let est_tautologie f n = sont_equiv f V n;;
let est_antilogie f n = sont_equiv f F n;;
let est_satifiable f n = not (est_antilogie f n);;
```

Question 5 –

```
let rec subs phi_array f = match f with
  | V -> V
  | F -> F
  | Var i -> phi_array.(i)
  | Non f -> Non (subs phi_array f)
  | Ou (f1, f2) -> Ou (subs phi_array f1, subs phi_array f2)
  | Et (f1, f2) -> Et (subs phi_array f1, subs phi_array f2);;
```

Question 6 – Il s'agit de montrer que si pour toute distribution de vérité μ : $E_\mu(\varphi) = E_\mu(\psi)$ alors pour toute distribution de vérité μ : $E_\mu(S(\varphi)) = E_\mu(S(\psi))$.

Pour l'instant, on fixe μ et on définit μ' par $\mu' : x_i \mapsto E_\mu(\varphi_i)$ pour tout i. Montrons par induction sur φ que $E_\mu(S(\varphi)) = E_{\mu'}(\varphi)$:

- Si $\varphi = \top$ alors $S(\varphi) = \top$ et $E_\mu(S(\varphi)) = 1 = E_{\mu'}(\varphi)$.
- Si $\varphi = \bot$ alors $S(\varphi) = \bot$ et $E_\mu(S(\varphi)) = 0 = E_{\mu'}(\varphi)$.
- Si $\varphi = x_i$ pour un certain i alors $S(\varphi) = \varphi_i$ et $E_\mu(S(\varphi)) = E_{\mu'}(\varphi)$ par définition de μ'.
- Si $\varphi = \neg\psi$ alors $S(\varphi) = \neg S(\psi)$ et d'après l'hypothèse d'induction :

$$E_\mu(S(\varphi)) = 1 - E_\mu(S(\psi)) = 1 - E_{\mu'}(\psi) = E_{\mu'}(\varphi).$$

- Si $\varphi = \psi_1 \wedge \psi_2$ alors $S(\varphi) = S(\psi_1) \wedge S(\psi_2)$ et d'après l'hypothèse d'induction :

$$E_\mu(S(\varphi)) = \min(E_\mu(S(\psi_1)), E_\mu(S(\psi_2))) = \min(E_{\mu'}(\psi_1), E_{\mu'}(\psi_2)) = E_{\mu'}(\varphi).$$

- Si $\varphi = \psi_1 \vee \psi_2$ alors $S(\varphi) = S(\psi_1) \vee S(\psi_2)$ et d'après l'hypothèse d'induction :

$$E_\mu(S(\varphi)) = \max(E_\mu(S(\psi_1)), E_\mu(S(\psi_2))) = \max(E_{\mu'}(\psi_1), E_{\mu'}(\psi_2)) = E_{\mu'}(\varphi).$$

On peut maintenant conclure. On suppose que pour toute distribution de vérité μ : $E_\mu(\varphi) = E_\mu(\psi)$. Soit μ une distribution de vérité alors :

$$E_\mu(S(\varphi)) = E_{\mu'}(\varphi) = E_{\mu'}(\psi) = E_\mu(S(\psi)).$$

Exercice 3.

Question 1 – La 3-clause ψ est de la forme $\psi = \ell_1 \vee \ell_2 \vee \ell_3$. Pour qu'une distribution de vérité μ vérifie $E_\mu(\psi) = 0$, il faut que $E_\mu(\ell_1) = E_\mu(\ell_2) = E_\mu(\ell_3) = 0$. Si on note x_1, x_2 et x_3 les variables présentent dans ψ, on en déduit que $\mu(x_1)$, $\mu(x_2)$ et $\mu(x_3)$ sont fixées et que $\mu(v)$ peut être quelconque pour toute autre variable $v \in V \setminus \{x_1, x_2, x_3\}$.

Puisque le nombre de variables dans $V \setminus \{x_1, x_2, x_3\}$ est $n-3$, le nombre de distributions de vérités qui ne satisfont pas ψ est 2^{n-3}. Finalement, le nombre de distributions de vérités qui satisfont ψ est $2^n - 2^{n-3} = \frac{7}{8}2^n$.

Question 2 – On note $\psi_1, \dots, \psi_m$ les 3-clauses présentes dans φ et :

$$\begin{aligned} M &= \Big\{\mu : E_\mu(\varphi) = 0\Big\}, \\ M_j &= \Big\{\mu : E_\mu(\psi_j) = 0\Big\} \qquad \text{pour tout } j \in [\![1, m]\!]. \end{aligned}$$

Pour qu'une distribution de vérité μ vérifie $E_\mu(\varphi) = 0$, il faut et il suffit que $E_\mu(\psi_j) = 0$ pour au moins l'un des ψ_j. Ainsi :

$$M = \bigcup_{j=1}^{m} M_j.$$

Si φ est une antilogie alors M contient toutes les distributions de vérité, c'est à dire $|M| = 2^n$. En utilisant la question précédente :

$$2^n = |M| = \left|\bigcup_{j=1}^{m} M_j\right| \leqslant \sum_{j=1}^{m} |M_j| = \sum_{j=1}^{m} \frac{1}{8}2^n = \frac{m}{8}2^n$$

D'où $m \geqslant 8$.

Exercice 4. Construction d'une FND à partir d'une FNC

Question 1 – En utilisant la distributivité :

$$y \wedge G = y \wedge \left[\bigvee_{i=1}^{a} \left(\bigwedge_{j=1}^{b_i} x_{i,j}\right)\right] \equiv \bigvee_{i=1}^{a} \left(y \wedge \bigwedge_{j=1}^{b_i} x_{i,j}\right).$$

Question 2 – En utilisant la distributivité et la question précédente :

$$
\begin{aligned}
D \wedge G = \left[\bigvee_{k=1}^{c} y_k\right] \wedge \left[\bigvee_{i=1}^{a}\left(\bigwedge_{j=1}^{b_i} x_{i,j}\right)\right] &\equiv \bigvee_{k=1}^{c}\left[y_k \wedge \left[\bigvee_{i=1}^{a}\left(\bigwedge_{j=1}^{b_i} x_{i,j}\right)\right]\right] \\
&\equiv \bigvee_{k=1}^{c}\bigvee_{i=1}^{a}\left(y_k \wedge \bigwedge_{j=1}^{b_i} x_{i,j}\right)
\end{aligned}
$$

Question 3 – On souhaite définir une forme normale disjonctive G_i telle que pour tout $i \in [\![0, m]\!]$:

$$G_i \equiv \bigwedge_{\ell=1}^{i} D_\ell$$

C'est à dire :

$$\begin{cases} G_0 \equiv \top \\ G_{i+1} \equiv D_{i+1} \wedge G_i & \text{pour tout } i \in [\![0, m-1]\!] \end{cases}$$

On définit donc :

$$\begin{cases} G_0 = \bigvee_{i=1}^{1} T_i & \text{où } T_0 = \top \text{ est la conjonction vide.} \\ G_{i+1} = \phi(D_{i+1}, G_i) & \text{pour tout } i \in [\![0, m-1]\!]. \end{cases}$$

Par une récurrence sur $i \in [\![0, m]\!]$, la formule G_i est une forme normale disjonctive et :

$$G_i \equiv \bigwedge_{\ell=1}^{i} D_\ell$$

En particulier, la forme normale disjonctive G_m est sémantiquement équivalente à F.

Question 4.a – On utilise la notation G_i de la question précédente. Pour $i \in [\![0, m]\!]$, on note r_i le nombre de clauses conjonctives dans G_i et s_i le nombre de littéraux dans chacune de ces clauses. Alors :

$$\begin{cases} r_0 = 1 \\ r_{i+1} = |D_{i+1}| \times r_i \end{cases} \quad \text{et} \quad \begin{cases} s_0 = 0 \\ s_{i+1} = 1 + s_i \end{cases}$$

On en déduit que le nombre de clauses dans $F' = G_m$ est $|D_1| \times \ldots \times |D_m|$ et que chaque clause contient m littéraux. Donc :

$$|F'| = m \prod_{i=1}^{m} |D_i|.$$

Question 4.b – Soit $m \in \mathbb{N}^*$. On définit F une forme normale conjonctive contenant m clauses disjonctives contenant chacune 3 littéraux. Alors :

$$
\begin{aligned}
&|F| = 3m \\
&|F'| = m \times 3^m = \frac{|F|}{3} \times 3^{|F|/3} = \Theta\Big(|F| \times 3^{|F|/3}\Big)
\end{aligned}
$$

Question 5 – Fonction qui correspond à la question 1 :

```
(* Fait le et d'un litteral et d'une fnd *)
let rec lit_et_fnd (lit: lit) (f : fnd): fnd = match f with
  | [] -> []
  | c :: q -> (lit :: c) :: (lit_et_fnd lit q);;
```

Fonction qui correspond à la question 2 :

```
let rec disj_et_fnd (d: disj) (f: fnd): fnd = match d with
  | [] -> []
  | lit :: q -> (lit_et_fnd lit f) @ (disj_et_fnd q f);;
```

Fonction qui correspond à la question 3 :

```
let rec fnc_to_fnd (f : fnc): fnd = match f with
  | [] -> [[]]
  | d :: q -> disj_et_fnd d (fnc_to_fnd q);;
```

Question 6 –

- ⋆ Soit $G = \bigvee_{i=1}^{a} C_i$ une FND où les C_i sont des clauses conjonctives. Alors G est satisfiable si et seulement si l'un des C_i est satisfiable.
- ⋆ Soit $C = \bigwedge_{k=1}^{c} y_k$ une clause conjonctive, alors C est satisfiable si et seulement si il n'existe pas de variable $v \in V$ telle que v et $\overline{v}$ apparaissent dans C.
- ⋆ Pour tester si une clause conjonctive C est satisfiable, l'idée est de créer un tableau `apparait` de taille n tel que pour tout `i` $\in [\![$`0, n - 1`$]\!]$:
  - `apparait.(i)` vaut `1` si x_i apparaît dans C.
  - `apparait.(i)` vaut `-1` si $\overline{x_i}$ apparaît dans C.
  - `apparait.(i)` vaut `0` sinon.
- ⋆ Pour tester si l'une des clauses conjonctives $C_1, \dots C_a$ est satisfiable, on applique la méthode précédente à chacun des C_k. Pour éviter de créer le tableau `apparait` pour chaque k (ce qui prend du temps), on modifie ce que contient `apparait` ; pour tout `i` $\in [\![$`0, n - 1`$]\!]$:
  - `apparait.(i)` vaut `k` si x_i apparaît dans C_k.
  - `apparait.(i)` vaut `-k` si $\overline{x_i}$ apparaît dans C_k.
  - `apparait.(i)` a une autre valeur sinon (cette valeur vaut 0 ou bien a été définie lorsqu'on testait un autre C_k).

```
(* Renvoie l'indice maximum présent dans la formule *)
let rec ind_max_conj (f : conj): int = match f with
  | [] -> -1
  | Var i :: q | Neg i :: q -> max i (ind_max_conj q);;
```

```
let rec ind_max_fnd (f : fnd): int = match f with
  | [] -> -1
  | phi :: q -> max (ind_max_conj phi) (ind_max_fnd q);;
```

```ocaml
let est_sat_fnd phi0 =
  let n = ind_max_fnd phi0 + 1 in
  let apparait = Array.make n 0 in

  let rec est_sat_conj k f = match f with
    | [] -> true
    | (Var i) :: q when apparait.(i) = -k -> false
    | (Neg i) :: q when apparait.(i) = k  -> false
    | (Var i) :: q -> apparait.(i) <- k;  est_sat_conj k q
    | (Neg i) :: q -> apparait.(i) <- -k; est_sat_conj k q
  in

  let rec aux k phi = match phi with
    | [] -> false
    | psi :: q -> est_sat_conj k psi || aux (k+1) q
  in

  aux 1 phi0;;
```

Question 7.a –

```ocaml
let est_sat_fnc (f: fnc) = est_sat_fnd (fnc_to_fnd f);;
```

Question 7.b – D'après la question 4.b, la taille de la sortie de la fonction `fnc_to_fnd` peut être exponentielle en la taille de l'entrée. Ainsi, dans le pire cas, la fonction `est_sat_fnc` s'exécute en temps au moins exponentiel en la taille de son entrée.

Exercice 5. CCP 2017

Question 1 – Le comportement manichéen des interlocuteurs dans le premier sujet abordé est représenté par la formule φ_1 :

$$\varphi_1 = (X_1 \wedge Z_1) \vee (\overline{X_1} \wedge \overline{Z_1})$$

Question 2 – On a :

$$X_1 = V \vee C \qquad\qquad Z_1 = \overline{V}$$

Question 3 – On a :

$$\begin{aligned}\varphi_1 &= ((V \vee C) \wedge \overline{V}) \vee ((\overline{V \vee C}) \wedge \overline{\overline{V}})\\ &\equiv ((V \wedge \overline{V}) \vee (C \wedge \overline{V})) \vee (\overline{V} \wedge \overline{C} \wedge V)\\ &\equiv \bot \vee (C \wedge \overline{V}) \vee \bot\\ &\equiv C \wedge \overline{V}\end{aligned}$$

D'après l'énoncé, la distribution de vérité sur les variables V et C est telle que la formule φ_1 s'évalue en 1. Donc le village se trouve dans les collines et les membres du village disent la vérité.

Question 4 – Le comportement manichéen des interlocuteurs dans le second sujet abordé est représenté par la formule φ_2 :

$$\varphi_2 = (X_2 \wedge Y_2 \wedge Z_2) \vee (\overline{X_2} \wedge \overline{Y_2} \wedge \overline{Z_2})$$

Question 5 – On a :

$$X_2 = G \wedge D \qquad\qquad Y_2 = M \Rightarrow \overline{D} \qquad\qquad Z_2 = G \wedge \overline{M}$$

Question 6 – Voici la table de vérité :

G	M	D	X_2	Y_2	Z_2	φ_2
0	0	0	0	1	0	0
0	0	1	0	1	0	0
0	1	0	0	1	0	0
0	1	1	0	0	0	1
1	0	0	0	1	1	0
1	0	1	1	1	1	1
1	1	0	0	1	0	0
1	1	1	1	0	0	0

D'après l'énoncé, la distribution de vérité sur les variables G, M et D est telle que la formule φ_2 s'évalue en 1. On a donc deux possibilités :

- Soit le chemin du milieu et celui de droite mènent au village.
- Soit le chemin de gauche et celui de droite mènent au village.

Pour rejoindre le village à coup sûr, il faut donc suivre le chemin de droite.

Question 7 – Si les trois participants ont menti, alors la distribution de vérité sur les variables G, M et D est telle que les formules X_2, Y_2 et Z_2 s'évaluent en 0. Donc on aurait également pu prendre le chemin du milieu.