Sujet n°1

Ce sujet est composé de deux problèmes indépendants. Le premier est de type « écrit 1 » et le second de type « écrit 2 ». Les programmes de SNT en seconde et NSI en première et terminale sont fournis en annexe. L'écriture « `machine à écrire` *» représente du code Python.*

Problème 1. Circuits booléens

1.1 Portes logiques, circuits

On rappelle que les circuits booléens sont réalisés avec des circuits élémentaires appelés *portes logiques*. On en compte usuellement six, illustrées sur la figure 1 et dont les comportements sont donnés par les tables 1 appelées *tables de vérité* :

— la porte OU qui prend deux entrées x et y et donne en sortie la valeur $x \vee y$;
— la porte ET qui prend deux entrées x et y et donne en sortie la valeur $x \wedge y$;
— la porte NON qui prend une entrée x et donne en sortie la valeur $\neg x$;
— la porte OU-EXCLUSIF qui prend deux entrées x et y et donne en sortie la valeur $x \oplus y$;
— la porte NON-OU qui prend deux entrées x et y et donne en sortie la valeur $\neg(x \vee y)$;
— la porte NON-ET qui prend deux entrées x et y et donne en sortie la valeur $\neg(x \wedge y)$.

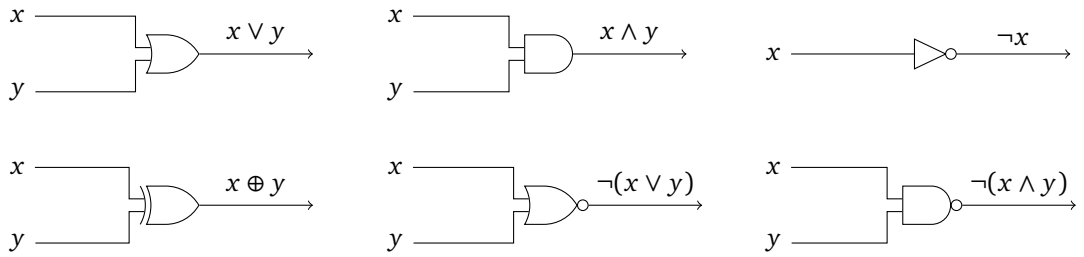

FIGURE 1 – Portes logiques usuelles

x	y	$x \vee y$	$x \wedge y$	$\neg x$	$x \oplus y$	$\neg(x \vee y)$	$\neg(x \wedge y)$
0	0	0	0	1	0	1	1
0	1	1	0	1	1	0	1
1	0	1	0	0	1	0	1
1	1	1	1	0	0	0	0

TABLE 1 – Tables de vérité des portes classiques

Un circuit booléen est un assemblage de plusieurs portes qui prend un nombre quelconque d'entrées et produit un nombre quelconque de sorties. Dans ce problème nous ne considérons que des circuits produisant une seule sortie. La figure 2 montre un circuit booléen à trois entrées x_0, x_1 et x_2, et une sortie $M(x_0, x_1, x_2)$.
Toute sortie d'un circuit peut être traduite en une expression algébrique utilisant les symboles $\vee$ pour le OU, $\wedge$ pour le ET, $\neg$ pour le NON, $\oplus$ pour le OU-EXCLUSIF, les variables d'entrées et les parenthèses. La sortie du circuit de la figure 2 est

$$M(x_0, x_1, x_2) = (\neg x_0 \wedge x_1 \wedge x_2) \vee (x_0 \wedge \neg x_1 \wedge x_2) \vee (x_0 \wedge x_1 \wedge \neg x_2) \vee (x_0 \wedge x_1 \wedge x_2).$$

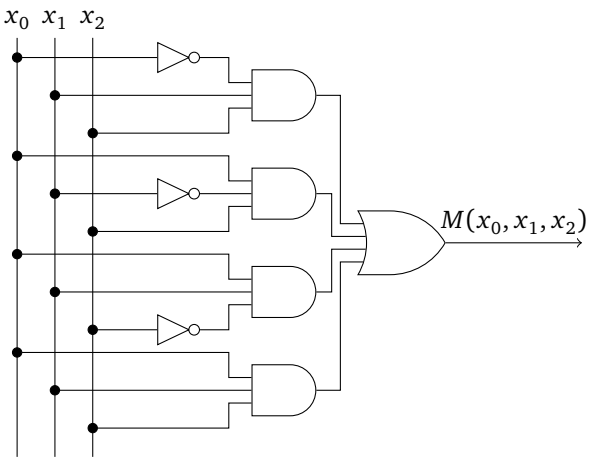

FIGURE 2 – Un circuit booléen à trois entrées

1.2 Fonctions booléennes

Les circuits à une seule sortie définissent des *fonctions booléennes*. Si le circuit a n entrées, $n \in \mathbb{N}$, la fonction booléenne a n variables. Une fonction booléenne peut être entièrement définie par une table qui donne pour chacune des 2^n valeurs de ses n variables la valeur de sortie associée. Une telle table est nommée *table de vérité* de la fonction booléenne.

Question 1.1.

(a) Établissez la table de vérité de la fonction booléenne $M(x_0, x_1, x_2)$ définie par le circuit de la figure 2.

Deux fonctions booléennes à n variables sont dites *distinctes* s'il existe une affectation des n variables pour lesquelles les valeurs de sortie associées aux deux fonctions sont différentes.

(b) Combien y a-t-il de fonctions booléennes distinctes à n variables ?

Question 1.2. Circuits équivalents Il est évident qu'on peut construire une infinité de circuits booléens différents à n entrées. On en déduit que plusieurs circuits différents peuvent définir la même fonction booléenne. Lorsque c'est le cas on dit que les circuits sont *équivalents*, et on utilise le symbole $\equiv$ pour signifier que deux circuits (ou deux expressions algébriques) sont équivalent(e)s. Les deux circuits à trois entrées de la figure 3 sont équivalents : cela résulte du fait que la porte OU est associative, ou exprimé algébriquement que $(x \vee y) \vee z \equiv x \vee (y \vee z)$.

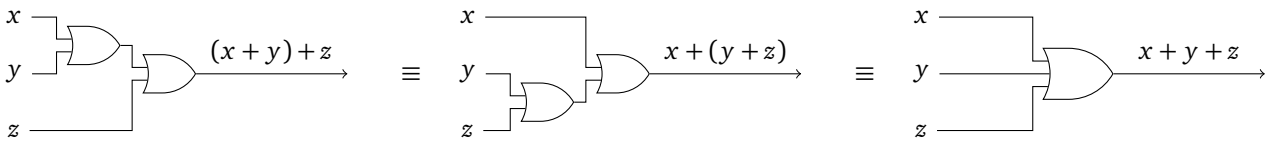

FIGURE 3 – Le OU est associatif

De nombreuses propriétés des portes OU, ET et NON permettent d'établir algébriquement l'équivalence de circuits. Elles sont rappelées dans la table 2.

(a) Prouvez la validité des lois de De Morgan.

Notons au passage que l'associativité du OU permet de généraliser les portes OU : une porte OU à n entrées peut toujours être construite à partir de $n-1$ portes OU. Il en est de même des portes ET et OU-EXCLUSIF. Le circuit de la figure 2 utilise des ET à trois entrées et un OU à quatre entrées.

(b) Trouvez un circuit pour la fonction $M(x_0, x_1, x_2)$ n'utilisant que quatre portes OU, ET et/ou NON.

identité	$1 \wedge x \equiv x$	$0 \vee x \equiv x$
absorption	$0 \wedge x \equiv 0$	$1 \vee x \equiv 1$
idempotence	$x \wedge x \equiv x$	$x \vee x \equiv x$
inversion	$x \wedge \neg x \equiv 0$	$x \vee \neg x \equiv 1$
commutativité	$x \wedge y \equiv y \wedge x$	$x \vee y \equiv y \vee x$
associativité	$(x \wedge y) \wedge z \equiv x \wedge (y \wedge z)$	$(x \vee y) \vee z \equiv x \vee (y \vee z)$
distributivité	$x \wedge (y \vee z) \equiv (x \wedge y) \vee (x \wedge z)$	$x \vee (y \wedge z) \equiv (x \vee y) \wedge (x \vee z)$
lois de De Morgan	$\neg(x \wedge y) \equiv \neg x \vee \neg y$	$\neg(x \vee y) \equiv \neg x \wedge \neg y$

TABLE 2 – Lois de l'algèbre de Boole

1.3 Complétude

Cette partie étudie la question suivante : peut-on réaliser toute fonction booléenne à l'aide d'un circuit ?

Question 1.3.

(a) Montrer que les quatre fonctions booléennes à une entrée possèdent chacune un circuit booléen. *Il est conseillé d'écrire les quatre tables de vérité.*

Soit $f(x_1, \ldots, x_n)$ une fonction booléenne à n variables. On note $f_0(x_1, \ldots, x_{n-1}) = f(x_1, \ldots, x_{n-1}, 0)$ et $f_1(x_1, \ldots, x_{n-1}) = f(x_1, \ldots, x_{n-1}, 1)$.

(b) Exprimer f en fonction de x_n, f_0 et f_1, à l'aide des portes logiques OU, ET et NON.

(c) En déduire que toute fonction booléenne admet un circuit booléen.

Question 1.4.

(a) Déterminer un circuit booléen équivalent au OU-EXCLUSIF et n'utilisant que les portes OU, ET et NON.

(b) Montrer que toute fonction booléenne peut être réalisée par un circuit ne contenant que les portes ET et NON. *On pourra montrer qu'il existe un circuit booléen équivalent à* OU *n'utilisant que les portes* ET *et* NON.

(c) Peut-on réaliser toute fonction booléenne uniquement avec la porte NON-ET ? Justifier.

Question 1.5.

(a) Montrer que la fonction booléenne représentée par un circuit n'ayant que des portes OU et ET vérifie $f(0, \ldots, 0) = 0$.

(b) En déduire que ces deux portes seules ne suffisent pas à représenter toute fonction booléenne.

(c) La porte NON-OU est-elle suffisante pour représenter toute fonction booléenne ? Justifier.

1.4 Fonctions particulières

Question 1.6. Bit de parité On définit la fonction *bit de parité* à n variables, notée P_n, par

$$P_n(x_0, x_1, \ldots, x_{n-1}) = \begin{cases} 0 & \text{si le nombre de variables non nulles est pair,} \\ 1 & \text{sinon.} \end{cases}$$

(a) Réaliser un circuit pour P_4.

(b) Prouver que pour toutes les valeurs des variables $x_0, x_1, \ldots, x_n$ on a

$$P_{n+1}(x_0, x_1, \ldots, x_n) \equiv (P_n(x_0, x_1, \ldots, x_{n-1}) \wedge \neg x_n) \vee (\neg P_n(x_0, x_1, \ldots, x_{n-1} \wedge x_n).$$

(c) En déduire un procédé de construction d'un circuit booléen réalisant la fonction P_{n+1} à partir de circuits réalisant P_n.

(d) On construit un circuit pour P_6 à partir du circuit que P_4 de la question **(a)**, par le procédé décrit dans la question précédente. Combien de portes de chaque type ce circuit utilise-t-il ?

Question 1.7. Multiplexeur Le multiplexeur est un circuit à $2^n + n$ entrées et une sortie. On note x_0, x_1, …, x_{2^n-1} les 2^n premières entrées, et y_0, y_1, …, y_{n-1} les n dernières, et on pose $k = \sum_{i=0}^{n-1} y_i 2^i$. Alors on définit la fonction booléenne MPX_n par

$$\text{MPX}_n(x_0, x_1, \ldots, x_{2^n-1}, y_0, \ldots, y_{n-1}) = x_k.$$

Autrement dit, MPX_n renvoie x_k où $y_0, \ldots, y_{n-1}$ est la représentation binaire de l'entier k. La figure 4 illustre un multiplexeur à $2^n + n$ entrées.

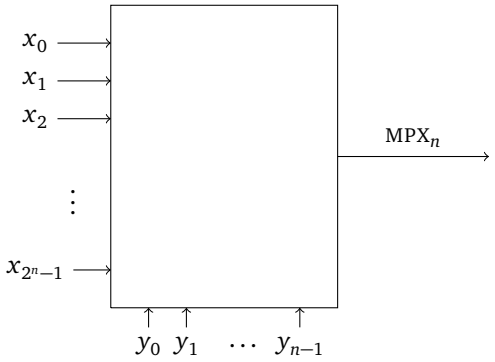

FIGURE 4 – Multiplexeur à $2^n + n$ entrées

(a) Dessiner les circuits correspondant à MPX_2 et MPX_3.

(b) Justifier que

$$\begin{aligned}&\text{MPX}_{n+1}(x_0, \ldots, x_{2^{n+1}-1}, y_0, \ldots, y_n)\\ &\qquad = (\neg y_n \wedge \text{MPX}_n(x_0, \ldots, x_{2^n-1}, y_0, \ldots, y_{n-1})) \vee (y_n \wedge \text{MPX}_n(x_{2^n}, \ldots, x_{2^{n+1}-1}, y_0, \ldots, y_{n-1}))\end{aligned}$$

Dessiner un circuit pour MPX_{n+1} à l'aide de deux circuits pour MPX_n.

(c) En partant du circuit pour MPX_2 à la question **(a)**, on construit un circuit pour MPX_4 à l'aide de la question précédente. Combien de portes de chaque type ce circuit utilise-t-il ?

Problème 2. Didactique

2.1 Enseignement du réseau

Question 2.1. Le programme de l'enseignement SNT de seconde comporte une partie sur « Internet ». Des contenus sur la « transmission de données dans un réseau » et « les protocoles de communications » figurent également au programme de la spécialité NSI de première.
Quelles sont les différences et points communs entre les approches pédagogiques que vous adopteriez en SNT et en NSI sur ce thème des réseaux si vous aviez à l'enseigner ?

Question 2.2. Le document 5 est extrait d'un manuel scolaire de 1ère NSI.

(a) Que pensez-vous des explications fournies ? Est-ce compréhensible par les élèves ? Quels rectifications et compléments apporteriez-vous à vos élèves ?

(b) Quelle(s) activité(s) pédagogique(s) proposeriez-vous pour illustrer cette partie de cours ? Pour chaque activité, vous préciserez sa nature, sa durée, ses modalités et détaillerez son contenu en quelques lignes.

Question 2.3. Le document 6 présente deux exercices issus d'un manuel scolaire de 1ère NSI.

(a) Considérez-vous que ces deux exercices sont en adéquation avec le programme ? Argumentez votre réponse.

(b) Pour chaque exercice, proposez une correction pour vos élèves.

DNS, l'annuaire d'Internet

En conclusion de cette présentation des réseaux, nous décrivons le *système de résolution de noms* (ou *DNS* pour *Domain Name System*), qui est un protocole de la couche d'application. Ce dernier (qui utilise en général UDP sur le port 53, mais peut aussi utiliser TCP) permet de maintenir des annuaires pour faire correspondre des noms symboliques, facilement utilisables par les humains, à des adresses IP. En effet, comme on l'a vu dans les paragraphes précédents, le seul moyen d'identifier une machine sur le réseau est son adresse IP. Que se passe-t-il alors lorsque l'on indique au navigateur Web que l'on souhaite se connecter au site `www.nsi-premiere.fr` ? Les serveurs DNS sont un ensemble de serveurs hiérarchisés agissant comme des annuaires. Au sommet de la hiérarchie existent des serveurs dits « racines ». Ces derniers connaissent les serveurs à contacter pour les *domaines de premier niveau* (ou TLD pour l'anglais *Top Level Domain*). Ces derniers correspondent au `.fr`, `.com`, `.edu`, `.net` et toutes les autres extensions possibles. Les serveurs DNS des domaines de premier niveau connaissent l'adresse des serveurs contenant les adresses des *domaines de second niveau* (dans notre exemple `nsi-premiere` est le domaine de second niveau). Ces derniers permettent ensuite de contacter les serveurs connaissant les adresses des *sous-domaines* (dans notre exemple, `www` est le sous-domaine). En pratique, lorsque l'on configure une connexion IP sur un ordinateur, on définit aussi l'adresse de serveurs DNS (par exemple ceux du fournisseur d'accès à Internet). Ces derniers agissent comme un « cache » : lorsqu'une machine leur demande l'IP pour un nom de domaine, ils la renvoient directement s'ils la connaissent déjà. Sinon, ils procèdent à la requête récursive décrite plus haut (en décomposant le nom demandé), puis une fois la réponse obtenue, ils mémorisent l'IP du nom de domaine complet pour pouvoir répondre plus rapidement aux prochaines requêtes.

FIGURE 5 – Extrait de cours d'un manuel scolaire de 1ère NSI

Exercice 255 On souhaite pourvoir raccorder 1200 machines sur le même sous réseau IP. Donner le plus petit masque permettant de définir un tel sous-réseau.

Solution page 498 □

Exercice 256 On considère le masque `255.255.252.0`. Parmi les adresses suivantes, indiquer lesquelles dénotent des machines du même sous-réseau.

1. 129.175.127.1
2. 129.175.130.10
3. 129.175.128.17
4. 129.175.131.110
5. 129.175.132.8

Solution page 499 □

FIGURE 6 – Extrait d'exercices d'un manuel scolaire de 1ère NSI

2.2 Enseignement de la récursivité

Question 2.4. Quelle définition donnez-vous à des lycéens d'une fonction récursive ?

Question 2.5. Pair/Impair On a demandé à un élève de terminale d'écrire une fonction récursive prenant en paramètre un entier n et renvoyant `True` si n et pair et `False` dans le cas contraire. Un élève a proposé le couple de fonctions suivantes comme réponse.

```
def pair(n):
    if n == 0:
        return True
    else:
        return not impair(n)
```

```
def impair(n):
    if n == 0:
        return False
    else:
        return not pair(n)
```

(a) Quel type de récursivité est utilisée par cette réponse ?
(b) Que pensez-vous des expressions calculées par chacune des deux fonctions dans les deux cas de base et dans les appels récursifs ?
(c) Quelle règle nécessaire à la terminaison des algorithmes récursifs n'est pas vérifiée ?

Un autre élève a appris cette règle et formule la réponse suivante.

```
def pair(n):
    if n==0:
        return True
    else:
        return even(n-2)
```

(d) Que pensez-vous de cette fonction, la règle précédente est-elle suffisante ?

Question 2.6. Calcul Le cadre ci-dessous est le sujet d'un autre exercice portant sur la récursivité, suivi des réponses de deux élèves.

> Proposez un algorithme récursif de calcul du produit de deux entiers naturel a et b en supposant que les seules opérations de base dont vous disposez sont :
> — la somme de deux entiers ;
> — le retrait de 1 à un entier ;
> — la comparaison à 0 d'un entier a.

```
def produit1(a,b):
    if b == 0:
        return 0
    else:
        return produit1(a-1, b+a)
```

```
def produit2(a,b):
    p = 1
    if a == 0:
        return p
    else:
        p = p*b
        produit2(a-1,b)
```

(a) Pour chacune des deux réponses fournies, expliquer l'erreur commise par l'élève et proposer une remédiation pour que l'élève puisse prendre conscience de son erreur.
(b) Pour éviter ces erreurs, quelle(s) démarche(s) peut-on adopter lors de la construction d'un algorithme récursif ?

Question 2.7. Tri fusion

(a) Rappeler le principe du tri fusion, et le paradigme auquel il se rattache.

(b) On veut expliquer cet algorithme aux élèves, proposez une activité débranchée pour les sensibiliser à ce tri.

Question 2.8. Un élève a écrit la fonction annexe suivante dans le cadre du tri fusion. Corriger cette réponse.

```
def diviser(l):
    """
    Entrée : une liste d'entiers
    Sortie : un couple (l1, l2) de listes tel que l est l'union disjointe
        de l1 et l2, et |len(l1)-len(l2)| < 2
    """
    if len(l) == 0:
        return ([], [])
    else:
        t = diviser[2:]
        return ( l[0] + t[0], l[1] + t[1] )
```

Question 2.9.

(a) Rappeler la complexité du tri fusion (préciser l'opération d'importance et ce que représente la variable dans l'expression que vous donnerez).

(b) On souhaite sensibiliser des élèves à cette question en TP. Décrire une activité pouvant être menée.

(c) On souhaite maintenant aborder la démonstration en cours au tableau. Reproduire ce tableau dans votre copie en le commentant (étapes, commentaires oraux éventuels et questions).