I. Introduction

Le projet que nous avons choisi de développer sera un système centré sur l'électronique logique, et qui permet l'entraînement à la réduction d'expressions booléennes en leur plus petite forme, avec leurs méthodes et autres propriétés associées (ie. circuits).

Ce projet a pour but d'accompagner les élèves dans l'apprentissage et l'acquisition de ces notions qui pourront s'en servir en tant qu'outil de compréhension et de correction, et/ou les professeurs comme outil pédagogique.

L'idée d'un tel projet nous est venu par notre cursus, puisque nous avons eu un module d'électronique au S1 dans lequel une partie du contenu était basé sur les expressions logiques, et que ces bases nous sont toujours nécessaire au S2. Un service pareil aurait donc pu être une aide bienvenue pour certains étudiants.

II. Description générale du projet

Le système à développer sera sous forme de site web, interactif pour la saisie des expressions booléenne et des potentiels choix de l'utilisateur.

Le site web sera découpé en trois parties (hormis la page d'accueil) dont **deux** principales (ie. en gras), une première partie sur les notions de l'électronique logique sous forme de cours, une **deuxième** partie sur qui proposera la réduction d'expressions booléennes saisies par l'utilisateur, et une **troisième** partie qui permettra à l'utilisateur de s'entraîner à en résoudre par lui-même.

Les fonctionnalités principales et prioritaires étant si dessus, l'intérêt de notre projet est avant tout d'être un outil pédagogique, c'est-à-dire, que les utilisateurs puissent comprendre et apprendre de leurs erreurs. Donc si possible, nous souhaiterions rajouter un procédé (pour la deuxième partie) et/ou une correction (pour la troisième partie), selon le souhait de l'utilisateur, qui afficherait les étapes à suivre. De plus, les circuits étant aussi une grande partie du programme de l'UE, nous pourrions les afficher, encore une fois sous le souhait de l'utilisateur.

Les utilisateurs ciblés par notre projet seront principalement les étudiants. Les enseignants pourront également s'en servir, tout comme un utilisateur lambda qui souhaite s'initier à l'électronique (d'où l'intérêt d'une partie cours).

III. Exigences fonctionnelles

Pour la première partie, comprenant les notions d'électronique logique, nous aurons un sommaire avec chaque points du cours et le contenu de ces points. Il n'y a aucune récupération de données, c'est une partie à titre informative.

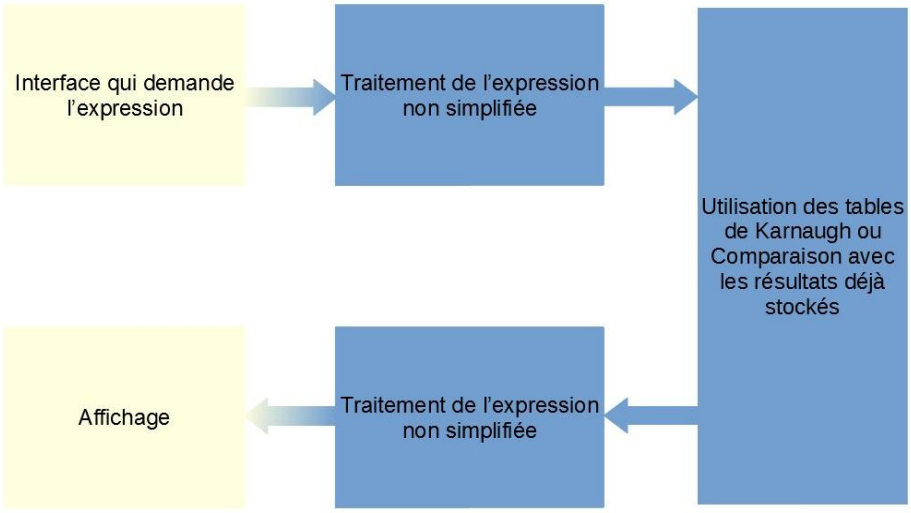
Pour la deuxième partie, qui correspond à la réduction d'expressions de Boole par le système, nous aurons un champ permettant de saisir une expression selon une syntaxe précise, avec un bouton qui lancera le procédé de réduction. L'expression réduite sera ensuite affichée. Dans la mesure du possible, nous rajouterons l'option d'afficher les étapes d'un ou de plusieurs procédés de réduction, ceux-ci étant les tables de Karnaugh et les propriétés algébriques de Boole, et le circuit associé.

Pour la troisième partie, qui correspond à la réduction d'expressions de Boole par l'utilisateur, une expression stockée au préalable et choisie de manière aléatoire et/ou par niveau de difficulté sera affichée à l'utilisateur, qui entrera sa réponse dans un champ. Un bouton lancera le procédé de vérification entre la réponse de l'utilisateur et l'expression réduite (qui sera probablement stockée au préalable) et affiche si sa proposition était correcte. Le cas échéant et selon la décision de l'utilisateur, une correction pourra être affichée, que ce soit par les tables de Karnaugh ou par les propriétés algébriques de Boole.

Les données entrées, sous forme d'une chaîne de caractère, seront contextualisées par des objets d'une classe qui représentera les différentes opérateurs ''non'', ''et'' et ''ou'', pour une simplification du traitement. Une idée de représentation est proposée dans la partie V. La table de Karnaugh sera elle représentée par une matrice, tableau de tableaux, de quatre lignes de quatre colonnes pour l'instant. Avec les données remises dans leur contexte et la table de Karnaugh, nous devrions pouvoir affecter les valeurs nécessaires au bon endroit dans la matrice en associant ses positions (i,j) avec les opérateurs utilisées, puis faire la traduction inverse pour récupérer l'expression réduite.
(explication précise de la table de karnaugh dans le glossaire des termes techniques ou dans références bibliographique je pense)

Pour une représentation d'un cas d'utilisation, voir la maquette de la page de réduction par le système (Annexe 3)

Voici un diagramme de flux du fonctionnement du système, pour les deuxième et troisième parties :



IV. Exigences non-fonctionnelles

Le projet, étant un site web, sera programmé en HTML et CSS, pour l'affichage de la page web, et en JavaScript (et/ou PHP) selon les tâches à réaliser et ce qui nous arrangera. Le site web sera déployé sur un navigateur web (ie. Chrome, Firefox,…) par

un localhost. Nous avons tout de même hésité avec Python, langage de programmation qui nous est beaucoup plus familier, avec lequel nous avons déjà abordé les notions de classes et qui a une récupération des données entrées légèrement plus simple. L'interface aurait alors été faite grâce à la bibliothèque d'interface graphique tkinter. Nous avons préféré développer nos compétences en programmation évènementielle et web.

Le temps de réponse devrait être moindre, si on considère les expressions manipulées lors des cours d'électronique, mais il faudrait éviter des expressions avec un nombre d'éléments trop important afin de ne pas se retrouver avec une infinité d'itérations.

Les données devront être aux normes de ce que nous avons décidé et de ce que le programme attend.

V. Exigences relatives aux données

Les seules données à traiter sous la contrainte d'une exigence seront les expressions booléennes saisies par l'utilisateur. Elles doivent être codifiées de manière à ce que le programme puisse reconnaître les opérateurs logiques et le nom des entrées utilisées. De plus, nous nous concentrerons sur les expressions sous forme canonique disjonctive, c'est-à-dire, les expressions en sommes de produits. Nous avons fait ce choix pour d'une part centraliser nos efforts, et d'une autre du fait que c'est la forme la plus commune utilisée dans les exercices. Cela dit, si la méthode de réduction avec la forme canonique disjonctive est opérationnelle, nous pourrons nous pencher sur le cas des expressions sous forme canonique conjonctive (ie. produits de sommes).

Encore une fois dans une optique de centrer nos efforts sur le nécessaire, nous exigerons un nombre de quatre entrées, ni plus, ni moins, qui auront pour nom ‘’A’’, ‘’B’’, ‘’C’’ et ‘’D’’, quoique leur nom importe peu. Des recherches pour un nombre d'entrées non défini, qui aura pour conséquence une résolution plus dynamique, pourra être pensé ultérieurement. Les opérateurs étant ‘’NON’’, ‘’ET’’ et ‘’OU’’ seront représentées respectivement par ‘’ !’’, ‘’.’’ et ‘’+’’.

Enfin à l'aide d'une classe et de ces expressions codifiées, nous les représenterons probablement sous forme de tableau de tableaux pour une exploitation plus simple, mais nous restons ouverts à d'autres alternatives.

Voici une visualisation des représentations :

$$\bar{A}.B.C + A.\bar{B}.C + A.B.\bar{C} + A.B.C \Leftrightarrow non(A)\,et\,B\,et\,C\;ou\;A\,et\,non(B)\,et\,C\;ou\;A\,et\,B\,et\,non(C)\;ou\;A\,et\,B\,et\,C$$

doit être représenté comme ci dessous

$$!A.B.C + A.!B.C + A.B.!C + A.B.C$$

que nous représenterons de cette manière (une possibilité) pour le système

[['NOT','A','.','B','.','C'],'+',['A','.','NOT','B','.','C'],'+',['A','.','B','.','NOT','C'],'+',['A','.','B','.'"C']]

VI. Interface utilisateur

Notre interface utilisateur sera un site web découpé en quatre pages dont la page d'accueil et les trois parties annoncées dans le **II**. Voir la représentation visuelle avec les maquettes **conceptuelles** (Annexes 1, 2, 3 et 4).

Ces maquettes ont pour simple but de visualiser une idée générale du placement des différentes informations, et des services proposés. Le design avec le CSS sera pensé ultérieurement, lorsque les fonctionnalités principales seront testées et validées.

VII. Planification du projet

En ce qui concerne le planning prévisionnel du projet, voici notre répartition :

// mettre l'échéancier

VIII. Documentation

L'utilisateur doit simplement respecter les exigences relatives les données (ie. V). Une compréhension de base sur l'électronique pourrait donc être appréciée, pour être sur que la donnée soit traitée correctement avec un temps de réponse minimal, mais n'est pas nécessaire puisque nous développons un outil d'apprentissage. L'utilisateur a accès à trois fonctionnalités :

- Une page qui expose les notions d'électronique logique en lien avec les expressions de Boole.

- Une page qui permet de réduire des expressions de Boole données. Vous devez saisir l'expression selon la syntaxe demandée (ie. !A.B.C+A.!B.C+A.B.!C+A.B.C), puis appuyer sur le bouton qui affichera le résultat. Éventuellement mais à voir, vous pourrez cocher si vous souhaitez voir afficher une correction, si oui avec quelle méthode, et un circuit, automatiquement un composé de NAND (les circuits NOR n'étant pas notre priorité actuellement).

- Une page qui permet de vous exercer, en vous affichant une expression que vous devez réduire par vous même. Vous écrirez votre réponse dans le champ selon la syntaxe demandée (ie. !A.B.C+A.!B.C+A.B.!C+A.B.C) puis vous appuierez sur le bouton de correction. Selon l'exactitude de votre proposition et les options choisies, une correction pourra vous être affichée encore une fois selon la méthode et avec ou sans un circuit. Un niveau de difficulté sera peut-être mis en place.

La documentation technique pour les développeurs et les procédures de maintenances seront pensées durant et en fin de projet.

Les développeurs doivent tout de même avoir acquis ces notions d'électronique logique, notamment le fonctionnement des tables de Karnaugh pour comprendre en minimum le système que nous déciderons de mettre en place pour les modéliser, les remplir et extraire leur sortie.

IX. Validation et tests

Pour mener à bien les jeux de tests, nous allons au préalable dresser une liste conséquente d'expressions à difficulté de réduction variante, avec la méthode employée, en dehors du programme pour être sûrs d'avoir le résultat qui convient. Nous devrons valider que la donnée en entrée est bien reçue par le serveur, puis qu'elle est bien traitable. Ensuite, nous devrons vérifier que qu'elle a correctement été transformée dans notre format choisi (ie. tableau de tableaux), c'est-à-dire dans l'ordre qui permet de la traiter comme une somme de produits. Puis vérifier que les tables de Karnaugh s'initialisent correctement et que les valeurs affectées dedans sont bien en adéquation

avec l'expression saisie. Enfin qu'elle renvoie la bonne expression et que celle-ci s'affiche bien sur la page. Le système sera donc valider

Pour les jeux de tests, nous en avons imaginé différents types selon la difficulté de l'expression. Premièrement, un cas de base où l'expression est déjà sous sa forme réduite. Le système doit nous la renvoyer intacte. Deuxièmement, un cas intermédiaire qui n'est pas une expression réduite, mais qui reste assez basique et donc demande peu d'opération à réaliser. Troisièmement, un cas avec une expression qui ne respecte pas une ou plusieurs préconditions, tels que le nombre d'entrées (si celui-ci reste fixé), la forme de l'expression (produits de sommes au lieu de sommes de produits), ou les symboles représentants les opérateurs. Un message d'erreur sera donc renvoyé, et le donnée ne devrait pas traitée. Dernièrement, un cas avec une expression conséquente qui demanderait beaucoup de calculs à réaliser. Nous pourrons ici évaluer le temps d'attente si il y en a.

Le projet sera considéré comme acceptable si la partie de réduction faite par l'ordinateur est opérationnelle, c'est-à-dire si les tables de Karnaugh font bien leur travail, et si les trois premiers cas des jeux de tests sont validés. Le dernier n'est pas forcément nécessaire si on reste dans l'optique des exercices réalisés en cours d'électronique, qui proposaient ses expressions raisonnables en termes de longueur

X. Ressources

Nous aurons besoin de trois personnes avec à notre portée nos machines personnelles, pour travailler de chez nous, et les machines de l'établissement, avec une connexion internet si besoin pour faire des recherches. Nous auront besoin également éditeur tel que VSCodium pour programmer avec les langages adéquats, c'est-à-dire HTML, JavaScript ou PHP si besoin et CSS.

XI. Conclusion

(je vois pas trop comment conclure honnêtement c'est un peu tôt dans le projet mais comme elle l'a précisé sur moodle juste avant l'onglet pour rendre le travail, si vous avez des idées de mini conclusions allez y)

XII. Annexes

table de Karnaugh : méthode graphique et simple pour trouver ou simplifier une fonction logique à partir de sa table de vérité.
Voir fonctionnement précis : https://fr.wikipedia.org/wiki/Table_de_Karnaugh

(puis si y'a d'autres termes a expliquer, on rajoutera)

(puis rajouter une bibliographie ? Pareil elle l'a précisé sur moodle avant l'onglet pour rendre le travail du coup un site qui explique le fonctionnement des classes en javascript ? Un autre site qui explique comment faire du graphique sur un site web ? jsp)

ELECTRONIQUE LOGIQUE

Message d'accueil sur le site avec explications brèves sur les différents services proposés. Message d'accueil sur le site avec explications brèves sur les différents services proposés. Message d'accueil sur le site avec explications brèves sur les différents services proposés. Message d'accueil sur le site avec explications brèves sur les différents services proposés. Message d'accueil sur le site avec explications brèves sur les différents services proposés. Message d'accueil sur le site avec explications brèves sur les différents services proposés. Message d'accueil sur le site avec explications brèves sur les différents services proposés.

NOTIONS | **RESOLUTION** | **ENTRAINEMENT**

Annexe 1

NOTIONS D'ELECTRONIQUE LOGIQUE

I. Fonctions booléennes
- Inverseur (NOT)
- Ou (OR)
- Et (AND)
- Ou exclusif (XOR)
- Non-Et (NAND)
- Non-Ou (NOR)

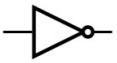
II. Propriétés algébriques
III. Les circuits et leurs composantes
IV. Formes d'expressions booléennes et leurs circuits associés
V. Résolution d'expressions booléennes

I . Fonctions Booléennes

- Inverseur (Not)

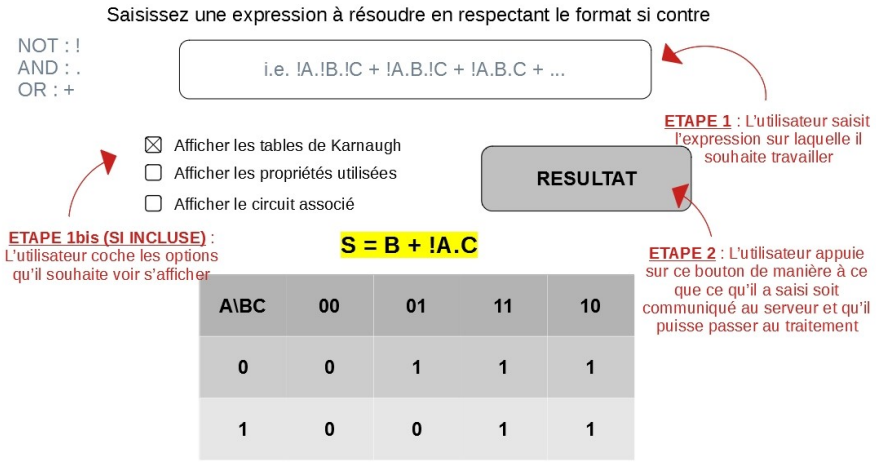
Informations en rapport de la sous-section

E	S
0	1
1	0



Annexe 2

REDUCTION D'EXPRESSIONS

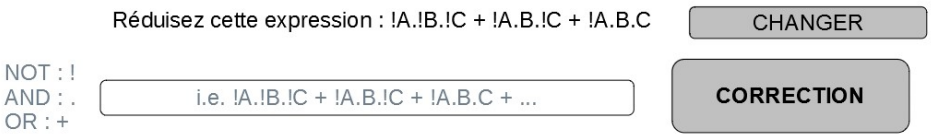


S = B + !A.C

A\BC	00	01	11	10
0	0	1	1	1
1	0	0	1	1

Annexe 3

ENTRAINEMENT



S = B + !A.C

Afficher la correction par tables de Karnaugh

Afficher la correction par les propriétés utilisées

A\BC	00	01	11	10
0	0	1	1	1
1	0	0	1	1

Annexe 4