# Documentation

## Vérification syntaxique

L'utilisateur introduit la formule sous la forme littérale ou numérique avec la syntaxe suivante:

- forme numérique : succession de valeurs séparées par des virgules (ex : 25,48,61,254)
- forme littérale :
    - Opérateurs : lettres alphabétique A,...Z
    - Opérandes :
        - "+" OU logique
        - ".": ET logique
        - "!" NON logique

- Pour une formule de plus de 26 variables, l'utilisateur est invité à introduire sa fonction en forme numérique.
- Les erreurs syntaxiques relatives à la formule sont traitées, et le type ainsi que la position de l'erreur sont affichés à l'utilisateur.

## Algorithme de McCluskey

La méthode de Quine-McCluskey est un algorithme permettant de minimiser les fonctions booléennes, développé par Willard V. Quine, puis étendu par Edward J. McCluskey. Elle comprends les étapes suivantes:

1. Exprimer la fonction sous forme canonique disjonctive puis exprimer les mintermes sous forme binaire.
2. Regrouper les mintermes selon leurs poids.
3. Constituer les nouveaux groupes résultants de l'adjacence des mintermes de deux groupes consécutifs.
4. Répéter l'étape autant de fois que nécessaire jusqu'à ne plus obtenir d'adjacences.
5. Identifier les impliquants premiers puis les impliquants premiers essentiels
6. Si la fonction est entièrement exprimée par ses impliquants premiers essentiels, Arrêter ;
7. Sinon, utiliser la méthode spéciale (décrite dans la section suivante).

## Méthode spéciale complémentaire à l'algorithme de McCluskey

On utilise une méthode spéciale dans le cas où les termes de la fonction ne sont pas tous exprimés par les impliquants essentiels. Cette méthode se chargera de traiter le reste des termes de la fonction, et ce en identifiant à partir des impliquants premiers non essentiels, la meilleure combinaison des termes permettant une couverture complète des mintermes non exprimés par les impliquants essentiels.

## Application du parallélisme dans l'algorithme de McCluskey

L'algorithme de Quine McCluskey est réputé pour l'optimalité de la simplification d'une part, mais d'un coût assez excessif en espace mémoire et en temps d'exécution d'autre part. Par conséquent, il est important de trouver le(s) moyen(s) de rendre l'algorithme plus performant et de lui permettre de dépasser le seuil d'arrêt de 10 variables. Afin d'atteindre cet objectif d'optimalité, nous avons procédé à deux types d'optimisation pour notre application :

### Optimisation algorithmique

Elle consiste en une bonne analyse du problème afin d'élaborer l'algorithme le plus performant possible, de telle sorte à diminuer au maximum les boucles, le nombre de variables utilisées, et éviter les traitements superflus. Les modules internes ainsi améliorés ont un moindre coût en espace mémoire et en temps d'exécution.

### Optimisation avec parallélisme

Afin d'accélérer la simplification, l'application fait usage du parallélisme. En effet, paralléliser les différents processus et les exécuter simultanément dans les différents cores du processeur offre une très grande performance en temps d'exécution. Ce procédé permet dans ce cas l'exécution parallèle d'un seul et même module à plusieurs reprises avec différentes entrées, en l'occurrence lorsque les performances de l'algorithme de Quine McCluskey diminuent dû à un très grand nombre de mintermes, celui-ci sera exécuté parallèlement sur des sous-ensembles constitutifs de la formule dont la taille assure une efficacité de l'algorithme. Cette alternative offre toutefois une solution approchée avoisinant l'optimale, sacrifiant ainsi un léger taux de précision pour un meilleur rendu en temps de minimisation.