

# Approche pratique de la théorie des langages formels.

## Rapport de TIPE

Charlotte THOMAS

12 mars 2022

### Résumé

L'informatique théorique et principalement dans l'analyse théorique des langages formels m'a toujours intéressé, l'idée de créer un langage et d'en implémenter moi-même sa grammaire, son lexer, parser, interpreter et en calculer les propriétés théoriques m'a vraiment intéressée et motivée.

Les erreurs de compilation peuvent coûter cher, et parfois se compter en vies humaines, il est donc nécessaire de développer des outils et des théories dans le domaine de la théorie des langages formels pour prévenir les bugs en amont, et protéger des vies et la société.

Pour illustrer ceci ce rapport va parler de la création du *Baguette#* (à prononcer "Baguette Sharp") un langage de programmation exotique avec une syntaxe proche d'un BASIC, basé sur des pâtisseries. Le REPL est disponible sur OPAM *opam install baguette-sharp*, le code source est sur [GitHub](https://github.com/CharlotteThomas/baguette-sharp) sous licence MIT, enfin le site web est disponible à cette adresse <https://www.baguettesharp.fr>

Finalement, ce rapport traitera essentiellement de l'implémentation d'une Binary Turing Machine en B# pour plus de renseignements généraux merci de voir les liens au ci dessus.

## Table des matières

<b>1</b>	<b>Le Baguette#</b>	<b>2</b>
1.1	Syntaxe vu par un exemple . . . . .	2
1.2	Librairie Standard . . . . .	2
<b>2</b>	<b>Machine de Turing Binaire en B#</b>	<b>3</b>

# 1 Le Baguette#

## 1.1 Syntaxe vu par un exemple

La syntaxe du Baguette# est très inspiré d'un BASIC, la plus grande différence est l'absence complète d'opérateur INFIX, là où un langage "classique" sera comme ça

```
int a = 1 + 2;
```

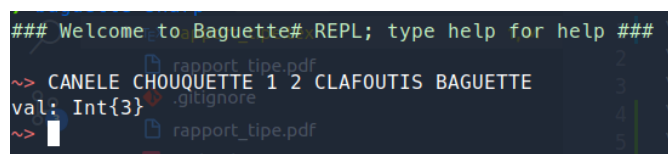
afin de calculer la somme de 1 et 2 (l'affectation sera discuté plus tard) le B# utilise une instruction comme ceci

```
ADD ( 1 2 );
```

la différence étant qu'en B# l'instruction *ADD* se nomme *CANELE*, et que les parenthèse ouvrante et fermantes sont respectivement *CHOUQUETTE* et *CLAFOUTIS* finalement le point-virgule est *BAGUETTE* donc la somme est notée

```
CANELE CHOUQUETTE 1 2 CLAFOUTIS BAGUETTE
```

Le résultat est visualisé ici sur le REPL



```
### Welcome to Baguette# REPL; type help for help ###
~> CANELE CHOUQUETTE 1 2 CLAFOUTIS BAGUETTE
val: Int{3}
~>
```

FIGURE 1 – Résultat dans le REPL de l'opération  $1 + 2$

La liste des mots clé de définitions est assez courte, il s'agit de *IF*, *LABEL*, *GOTO* leur utilisation est expliquée [ici](#) à l'exception de ces mots clé, toutes les instructions doivent avoir leur paramètre entre parenthèse, les paramètres peuvent être séparés par une virgule le *lexer* ne les vois pas. Mais ils doivent être séparés par un espace au minimum (y compris avec une virgule)

```
CANELE CHOUQUETTE 1 , 2 CLAFOUTIS BAGUETTE
```

est équivalent à

```
CANELE CHOUQUETTE 1 2 CLAFOUTIS BAGUETTE
```

Une explication complète de la syntaxe est disponible sur le [wiki-syntaxe](#)

## 1.2 Librairie Standard

Une liste complète des instructions est visible sur le [wiki](#) et un tutoriel sur l'utilisation basique est disponible [ici](#). La librairie standard du langage implémente les usages suivants :

- Opération Mathématiques sur les entiers et flottant
- Algèbre booléenne

- Lecture de l'entrée standard, écriture sur la sortie standard
- Manipulation de tableaux
- Manipulation de chaînes de caractère

Le langage est typé faiblement, et avec inférence de type, les types primitifs suivants sont implémentés

- Nombres entiers, flottants (NB : aucune distinction n'est faite entre les deux en général)
- Chaînes de caractères (NB : les caractères sont des chaînes de caractère de longueur 1)
- Booléens (*CUPCAKE* est *true* et *POPCAKE* est *false*)
- Null (le *unit* de OCaml)

Enfin le langage implémente les tableaux, à savoir qu'ils peuvent être *non-homogène* donc vous pouvez faire un tableau contenant deux entiers, trois flottants et quatre chaînes de caractère cependant comme ce sont des tableaux ils sont de *taille fixe* et ils héritent du caractère *mutable* des tableaux de OCaml

## 2 Machine de Turing Binaire en B#