

**Prosit Aller**

Compilation

**ABALO A. Blessing**

**SOMMAIL**

Mots de clés …………………………………………………………….............1

Contexte…………………………………………….…………..…………………...1

Contrainte ……………………………………….……………………………….….1

Besoins………………………………………………….……………………………..1

Problématiques…………………………………………………………………….1

Généralisation……………………………………………………………………...1

Hypothèses…………………………………………………………………………...1

Pistes……………………………………………………………………….………..….1

**Plan d’action**

**Etude…………………………………………………………………..….I**

Définition…………………………………………………….………………………..1

Ressources…………………………………………………………………………….2

Construction des langages (lexique et grammaire) ………….……3

Les automates……………………………………………………………………….4

Compilateurs (a générateur / b Arduino) ……………………………..5

**Vérification…………………………………………………………….II**

Etude du schéma …………………………………………………………………..1

Vérification du code……………………………………………………………….2

Vérification des mots …………………………………………………………….3

CONCLUSION…………………………………………………………….III

Étude des résultats ………………………………………………………………..1

Validation des hypothèses ……………………………………………………..2

Réponse aux problématiques …………………………………………………3

**Mots clés :**

-Compilateur

-Assembleur / Langage assembleur

- Programme d’exécution

- Langage machine

- Analyse sémantique

- Téléverser un programme

- Circuit

- Instruction

- Analyse syntaxique

- Arduino

- Automate

**Contexte :**

- Arnold et Vincent cherchent un moyen de communiquer avec l’arduino

**Besoins / Contraintes** :

-Carte arduino

-Langage C et C++

- Temps

**Problématiques :**

-Comment communiquer avec la machine ?

- Comment fonctionne un compilateur ?

-Comment téléverser un programme

**Généralisation** :

Conversion d’un langage à un autre

**Hypothèses :**

- La compilation sert à convertir le langage humain en langage machine

- Il existe plusieurs étapes dans une compilation

- Le schéma sert à vérifier l’existence des mots T-X, T-100, …

- Le schéma est un compilateur

**Plan d’action :**

**I. ETUDE**

1. Définition des mots clés

2. A voir/Personnel (Ex : Etude des ressources, etc…)

3. Construction des langages (lexique et grammaire)

4. Les automates

5. Compilateurs (a générateur / b Arduino)

**II. Vérification**

1. Etude du schéma

2. Vérification du code

3. Vérification des mots

**III. CONCLUSION**

1. étude des résultats

2. validation des hypothèses

3. réponse aux problématiques

**Plan d’action :**

**I. ETUDE**

**1. Définition des mots clés**

- **Compilateur** est un programme qui lit un autre programme rédigé dans un langage de programmation, appelé **langage source** et qui le traduit dans un autre langage, **le langage cible**.

-**Assembleur / Langage assembleur** est un langage de bas niveau qui représente le langage machine sous une forme lisible par un humain. Il convertit ces mnémoniques en langage machine en vue de créer par exemple un fichier objet ou un fichier exécutable.

- **Programme d’exécution** est une succession d'instructions exécutable par l'ordinateur.

- **Langage machine** est la suite de [bits](https://fr.wikipedia.org/wiki/Bit_(informatique)) qui est interprétée par le [processeur](https://fr.wikipedia.org/wiki/Processeur) d'un [ordinateur](https://fr.wikipedia.org/wiki/Ordinateur) exécutant un [programme informatique](https://fr.wikipedia.org/wiki/Programme_informatique).

- **Analyse sémantique** tente de découvrir de façon plus générale le sens des phrases ou de textes entiers ainsi que la table de symboles aﬁn d’effectuer un certain nombre de contrôles sémantiques : vériﬁer que les variables utilisées ont bien été déclarées.

- **Téléverser un programme** transférer des données d’un ordinateur local vers un ordinateur distant par un réseau informatique.

- **Circuit** est un ensemble des composants électroniques interconnectés.

- **Instruction** est une forme d'information communiquée qui est à la fois une commande et une explication pour décrire l'action, le comportement, la méthode ou la tâche qui devra commencer, se terminer, être conduit, ou exécuté.

- **Analyse syntaxique** est l’étude de la structure de la phrase, dans le but de définir comment les lexèmes sont organisés et quelles fonctions ont les mots qui servent à les mettre en relation.

- **Arduino** est une carte matériellement libre sur laquelle se trouve un microcontrôleur.

- **Automate** est un appareil renfermant de dispositif mécanique ou électrique lui permettant d'exécuter à la commande un ensemble d'action déterminées.

**2-Etude des ressources**

**3. Construction des langages (lexique et grammaire)**

**La grammaire du langage**

**Un programme** est une suite de déﬁnitions de fonction. Une déﬁnition de fonction est composée du nom de la fonction suivie de ses arguments suivie de la déclaration de ses variables internes suivie d’un bloc d’instructions.

**Une instruction** est formelle :

Programme → listeDecFonc

listeDecFonc → decFonc listeDecFonc

listeDecFonc →

decFonc → ID FCT listeParam listeDecVar ’;’ instrBloc

Les contraintes syntaxiques sont représentées sous la forme de règles de réécriture. La règle A→ BC nous dit que le symbole A peut se réécrire comme la suite des deux symboles B et C. L’ensemble des règles de réécriture constitue la grammaire du langage. La grammaire d’un langage L permet de générer tous les programmes corrects écrits en L et seulement ceux-ci

Dans la règle A→ α A est appelée partie gauche de la règle. α est appelée partie droite de la règle. Lorsque plusieurs règles partagent la même partie gauche : A→ α1 , A→ α2 ,..., A→ αn On les note : A→ α1 | α2 |...| αn

EXPRESSION → EXPRESSION OP2 EXPRESSION OP2 → +|-|\*|/EXPRESSION → NOMBREEXPRESSION → (EXPRESSION) NOMBRE → CHIFFRE|CHIFFRE NOMBRECHIFFRE → 0|1|2|3|4|5|6|7|8|9 Les symboles EXPRESSION, OP2, NOMBRE, CHIFFRE sont appelés symboles non terminaux de la grammaire Les symboles +, -, \*, /, (, ), 0, 1 , ..., 9 sont appelés symboles terminaux de la grammaire

**Construction des langages lexique**

Aﬁn de simpliﬁer la grammaire décrivant un langage, on omet de cette dernière la génération de certaines parties simples du langage. Ces dernières sont prises en charge par un analyseur lexical L’analyseur lexical traite le programme source et fournit le résultat de son traitement à l’analyseur syntaxique.

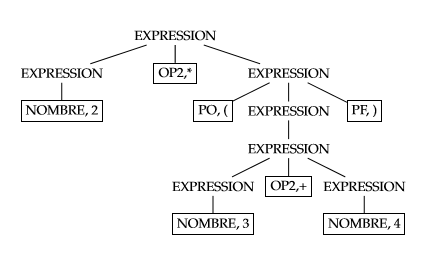


Lit le programme source Reconnait des séquences de caractères signiﬁcatives appelées lexèmes Pour chaque lexème, l’analyseur lexical ´émet un couple

(Type du lexème, valeur du lexème)

**Exemple**

(NOMBRE,123) Les types de lexèmes sont des symboles, ils constituent les symboles terminaux de la grammaire du langage. Les symboles terminaux de la grammaire (ou types de lexèmes) constituent l’interface entre l’analyseur lexical et l’analyseur syntaxique. Ils doivent ˆêtre connus des deux.



**4. Les automates**

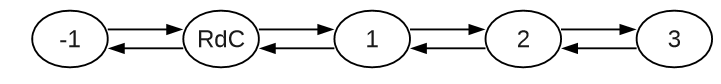
**Les principes de l’automate :**

**-Principe de la « séquence » :** suite d'actions**...**

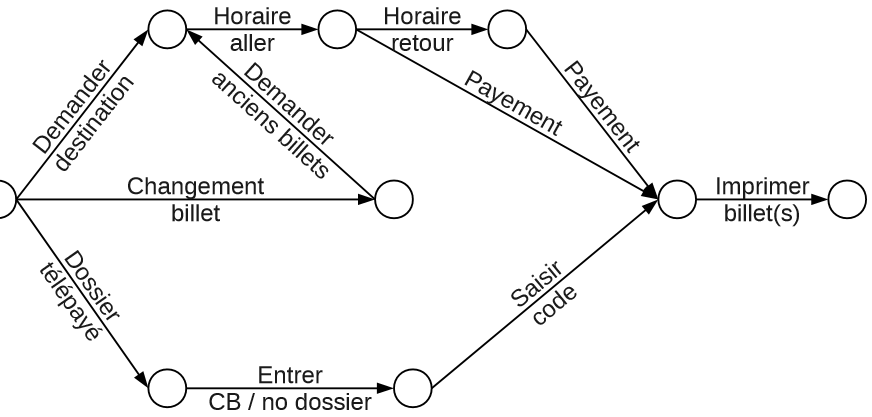
**-Principe de l'état :** change selon l'endroit où l'on est au sein de la séquence.

-**Principe de la transition** : selon la séquence et l'état, une décision est prise : actions à faire et nouvel état atteint.

Par exemple, l'ascenseur est un automate



Autre exemple : le distributeur de billet de train



**-Les entrées** : ce que l'automate « consomme »

**-Les sorties** : les actions / décisions de l'automate

**-L'état** interne parmi une liste d'états possibles

**- Les transitions** : l'automate bascule d'un état vers un autre

**5. Compilateurs (a générateur / b Arduino)**

**Le compilateur** signale de plus toute erreur contenue dans le programme source Lorsque le programme cible est un programme exécutable, en langage machine, l’utilisateur peut ensuite le faire exécuter aﬁn de traiter des données et de produire des résultats.

La compilation se décompose en deux phases :

- Une phase d'analyses, qui va reconnaître les variables, les instructions, les opérateurs et élaborer la structure syntaxique du programme ainsi que certaines propriétés sémantiques

- Une phase de synthèse et de production qui devra produire le code cible :

**Une phase d'analyses**

**Analyse lexicale**

Dans cette étape, il s'agit de reconnaître les "types" des "mots" lus. Pour cela, on lit le programme source de gauche à droite et les caractères sont regroupés en unités lexicales. L'analyse lexicale se charge de :

-éliminer les caractères superflus (commentaires, espaces, ...)  identifier les parties du texte qui ne font pas partie à proprement parler du programme mais sont des directives pour le compilateur

- identifier les symboles qui représentent des identificateurs, des constantes réelles, entière, chaînes de caractères, des opérateurs (affectation, addition, ...), des séparateurs (parenthèses, points virgules, ...), les mots clefs du langage, ... C'est cela que l'on appelle des unités lexicales.

**Analyse syntaxique**

Il s'agit de regrouper les unités lexicales en structures grammaticales, de découvrir la structure du programme. L'analyseur syntaxique sait comment doivent être construites les expressions, les instructions, les déclarations de variables, les appels de fonctions, ... Outils théoriques utilisés : grammaires et automates à pile.

**Analyse sémantique**

Dans cette phase, on opère certains contrôles (contrôles de type, par exemple) afin de vérifier que l'assemblage des constituants du programme a un sens. Outil théorique utilisé : schéma de traduction dirigée par la syntaxe.

- Les identificateurs apparaissant dans les expressions ont-ils été déclarés ?

- les opérandes ont-ils les types requis par les opérateurs ?

- les opérandes sont-ils compatibles ? n'y a-t-il pas des conversions à insérer ?

- les arguments des appels de fonctions ont-ils le nombre et le type requis ?

**Une phase de synthèse**

Mais si l'on écrit un compilateur pour un processeur donné, il n'est alors pas évident de porter ce compilateur (ce programme) sur une autre machine cible. C'est pourquoi on introduit des machines dites abstraites qui font abstraction des architectures réelles existantes. Ainsi, on s'attache plus aux principes de traduction, aux concepts des langages, qu'à l'architecture des machines.

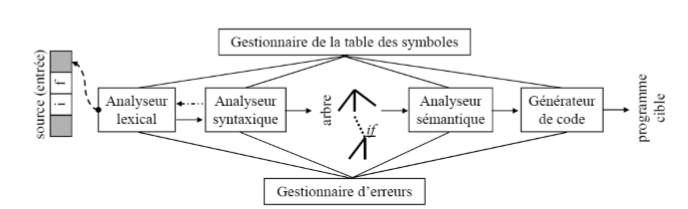
**Génération du code machine**

Cette phase nécessite la connaissance de la machine cible (réelle, virtuelle ou abstraite), et notamment de ses possibilités en matière de registres, piles, etc

**Gestion de la table des symboles**

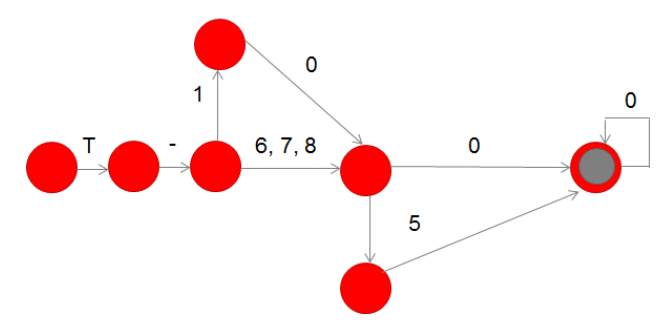
La table des symboles est la structure de données utilisée servant à stocker les informations qui concernent les identificateurs du programme source (par exemple leur type, leur emplacement mémoire, leur portée, visibilité, nombre et type et mode de passage des paramètres d'une fonction, ...)

**Gestion des erreurs**

Chaque phase peut rencontrer des erreurs. Cependant, après avoir détecté une erreur, une phase doit la traiter de telle façon que la compilation puisse continuer et que d'autres erreurs dans le programme source puissent être détectées. Un compilateur qui s'arrête à la première erreur n'est pas non plus très performant. Bien sûr, il y a des limites à ne pas dépasser et certaines erreurs (ou un trop grand nombre d'erreurs) peuvent entrainer l'arrêt de l'exécution du compilateur. 

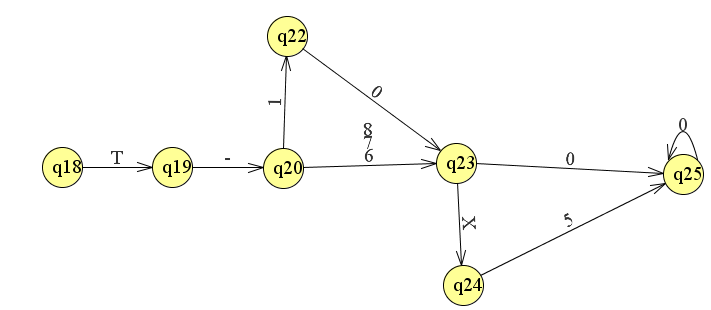
**II. Vérification**

1. **Etude du schéma**



Ce schéma représente à un automate déterministe dont les transitions à partir de chaque état sont déterminées de façon unique par le symbole d’entrée.

1. **Vérification du code**



Il manquant un X

1. **Vérification des mots**

**T-850, T-X, T-100, T-1000, T-10000, T-700, T-650**

Sont reconnus par l’automate déterministe

**III. CONCLUSION**

**1. étude des résultats**

**2. validation des hypothèses**

Oui

Oui

Oui

Non

**3. réponse aux problématiques**