Compilation

Pr. Youness Tabii

Plan

- Introduction et rappel
- Analyseur lexical
- Analyseur syntaxique
- Analyseur sémantique
- Représentation intermédiaire
- Génération de code
- Optimisation

Introduction

- Les micro-processeurs deviennent indispensables et sont embarqués dans tous les appareils que nous utilisons dans la vie quotidienne
  - Transport
    - Véhicules, Systèmes de navigation par satellites (GPS), Avions, …
  - Télécom
    - Téléphones portables, Smart Phones, …
  - Électroménager
    - Machine à laver, Micro-ondes, Lave vaisselles, …
  - Loisir
    - e-book, PDA, Jeux vidéo, Récepteurs, Télévision, TNT, Home Cinéma…
  - Espace
    - Satellites, Navettes, Robots d'exploration, …
- Pour fonctionner, ces micro-processeurs nécessitent des programmes spécifiques à leur architecture matérielle

Introduction

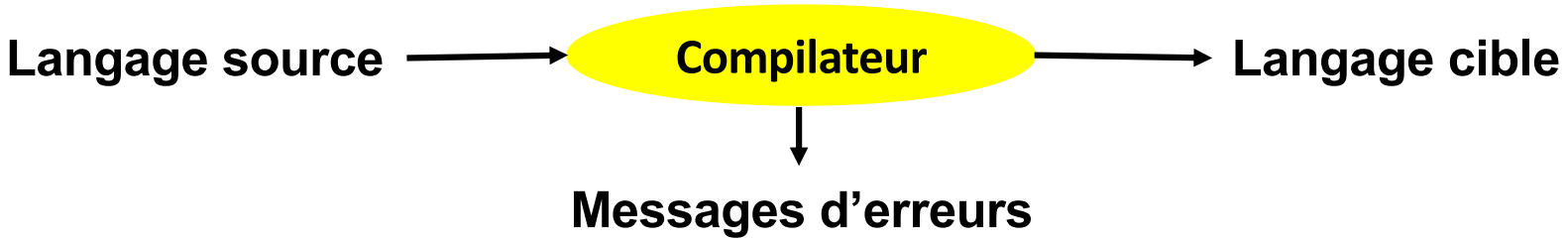
- Programmation en langages de bas niveau (proches de la machine)
  - **très difficile** (complexité)
    - Courbe d'apprentissage très lente
    - Débuggage fastidieux
  - **très coûteuse en temps** (perte de temps)
    - Tâches récurrentes
    - Tâches automatisables
  - **très coûteuse en ressource humaine** (budget énorme)
    - Tâches manuelles
    - Maintenance
  - **très ingrate** (artisanat)
    - Centrée sur les détails techniques et non sur les modèles conceptuels
  - **n'est pas à 100% bug-free** (fiabilité)
    - Le programmeur humain n'est pas parfait

Introduction

- Besoin continu de
  - Langages de haut niveau
    - avec des structures de données, de contrôles et des modèles conceptuels proches de l'humain
  - Logiciels de **génération** des **langages bas niveau** (propres aux microprocesseurs) à partir de **langages haut niveau** et vice versa
    - **Compilateurs**
    - **Interpréteurs**
    - **Pré-processeurs**
    - **Dé-compilateur**
    - **etc.**

Compilateur - Définition

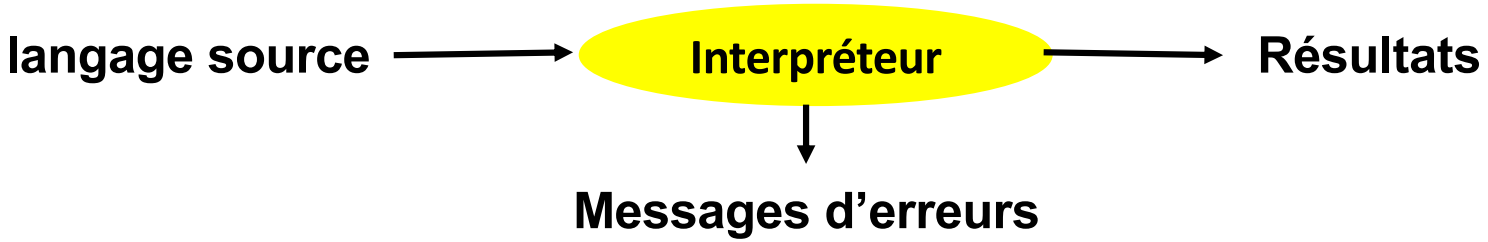
- Un compilateur est un logiciel (une *fonction*) qui prend en entrée un programme **P1** dans un langage source L1 et produit en sortie programme équivalent **P2** dans un langage L2



- Exemple de compilateurs :
  - C pour Motorola, Ada pour Intel, C++ pour Sun, doc vers pdf, ppt vers Postscript, pptx vers ppt, Latex vers Postscript, etc.

Interpréteur - Définition

- Un interpréteur est logiciel qui prend en entrée un programme **P1** dans un langage source L1 et produit en sortie les **Résultats** de l'exécution de ce programme



- Exemples d'interpréteurs :
  - Batch DOS, Shell Unix, Prolog, PL/SQL, Lisp/Scheme, Basic, Calculatrice programmable, etc.

Dé-compilateur

- Dé-Compilateur est un compilateur dans le sens inverse d'un compilateur (depuis le langage bas niveau, vers un langage haut niveau)
- Applications :
  - Récupération de vieux logiciels
    - Portage de programmes dans de nouveaux langages
    - Recompilation de programmes vers de nouvelles architectures
  - Autres
    - Compréhension du code d'un algorithme
    - Compréhension des clefs d'un algorithme de sécurité

Questions

- Quels sont les constituants d'un langage naturel ?

- Si l'on veut programmer un traducteur de l'Anglais vers le Français que proposeriez-vous comme algorithme ?

Quelques éléments de réponses

- Quelques constituants d'un langage naturel ?
  - Vocabulaire (lexique), Syntaxe (grammaire), Sémantique (sens selon le contexte)

- Un macro-algorithme pour traduire de l'Anglais vers le Français que proposeriez-vous comme ?
  1. Analyser les mots selon le dictionnaire Anglais
  2. Analyser la forme des phrases selon la grammaire de l'Anglais
  3. Analyser le sens des phrases selon le contexte des mots dans la phrase anglaise
  4. Traduire le sens des phrases dans la grammaire et le vocabulaire du Français

Étapes et Architecture d'un compilateur

1. Compréhension des entrées (**Analyse**)
   - Analyse des éléments lexicaux (lexèmes : mots) du programme : **analyseur lexical**
   - Analyse de la forme (structure) des instructions (phrases) du programme : **analyseur syntaxique**
   - Analyse du sens (cohérence) des instructions du programme : **analyseur sémantique**

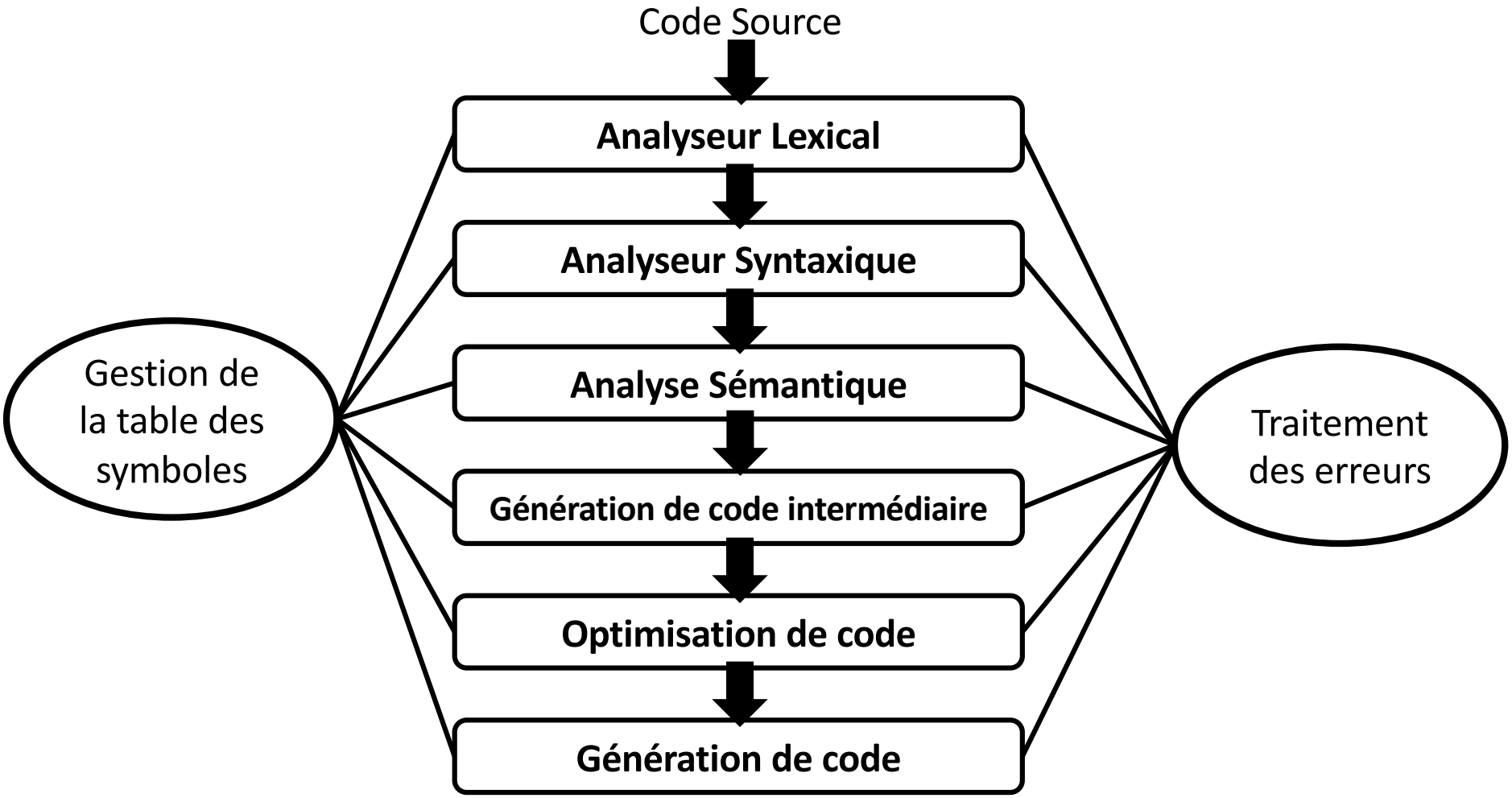
2. Préparation de la génération (**Synthèse**)
   - Génération d'un code intermédiaire (nécessitant des passes d'optimisation et de transcription en code machine) : **générateur de pseudo-code**
   - Optimisation du code intermédiaire : **optimisateur de code**

3. Génération finale de la sortie (**Synthèse suite**)
   - Génération du code cible à partir du code intermédiaire : **générateur de code**

Étapes et Architecture d'un compilateur

- ✓ Autres Fonctionnalités
  - ☐ gestion les erreurs et guide le programmeur pour corriger son programme : **gestionnaire d'erreurs**
  - ☐ gestion du dictionnaire des données du compilateur (symboles réservés, noms des variables, constantes) : **gestionnaire de la table des symboles**

Architecture d'un compilateur

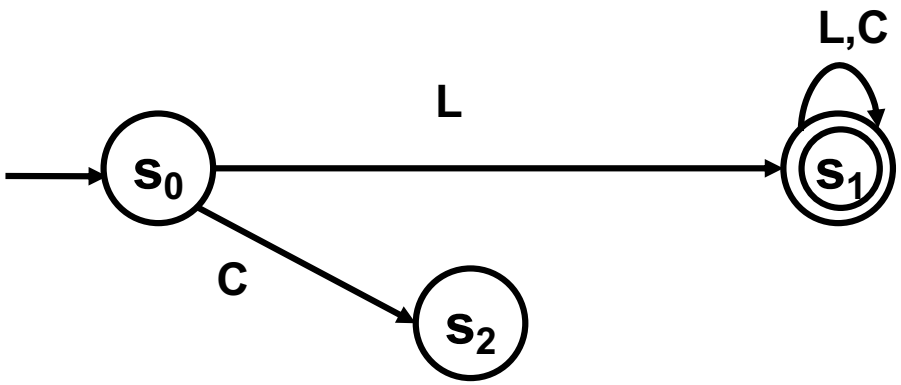


Transcription d'une ER

Comment Transcrire une expression régulière en langage C ?

Expression : **L(L+C)*** Avec **L=[a-zA-Z]** et **C=[0-9]**

Rappel
ER → DFA minimal → Programme en C



Etat mort/Trash

Rappel
ER → DFA minimal → Programme en C

```
typedef  enum {S0, S1, S2} State;

int main(int argc,char *argv[]){
    // 1- traitement intial
    State state = S0;

    printf("S0\n");

    automate(state);

    return 0;
}
```

```
void automate(State current_state){
    State state = current_state;
    char c = getchar();
    if (c != '$') {
    // 2- traitement récursif
    switch (state){
            case S0 :
                if (((c >= 'a') && (c <='z')) || ((c >= 'A') && (c <='Z'))){
                    state = S1; printf("S0 --> S1\n");
                }else {state = S2; printf("S0 --> S2\n");}
                break;
            case S1 :
                if (((c >= 'a') && (c <='z')) || ((c >= 'A') && (c <='Z')) || ((c
 >= '0') &&  (c <= '9')))
                 state = S1; printf("S1 --> S1\n");
                break;
            case S2 :
                state = S2;  printf("S2 --> S2\n");
                }
    automate(state);
    }else{
    // 3- traitement final
    if (state == S1) printf("mot accepté par l'automate des identificateurs\n");
    else printf("mot refusé par l'automate des identificateurs\n");}}
```

Rappel
ER → DFA minimal → Programme en C

./idAutomate

state0 **a123456**$

state0 --> state1

state1 --> state1

state1 --> state1

state1 --> state1

state1 --> state1

state1 --> state1

state1 --> state1

mot accepté par l'automate des identificateurs

./idAutomate

state0 **12345**$

state0 --> state2

state2 --> state2

state2 --> state2

state2 --> state2

state2 --> state2

mot refusé par l'automate des identificateurs