Structure d'un compilateur

L. Sais

Compilation

5 septembre 2016

Structure d'un compilateur

2 phases :

- **Une phase d'analyse** : reconnaître les variables, les instructions, les opérateurs et élaborer la structure syntaxique du programme ainsi que certaines propriétés sémantiques
- **Une phase de synthèse (production)** : produire le code cible

Phase d'analyse

But :

- Décider si le programme source est correct ou pas par rapport aux spécifications du langage source.
- Préparer la phase synthèse en produisant des structures de données adaptées (table de symboles, arbre syntaxique, etc...)

3 phases successives :

- analyse lexicale
- analyse syntaxique
- analyse sémantique

Phase d'analyse

Analyse lexicale (analyse linéaire)

But : Reconnaître les "types" des "mots" du langage source, pour cela les caractères sont regroupés en unités lexicales (token).

L'analyse lexicale

- élimine les caractères superflus (commentaires, espace, passage la ligne, ...)
- identifie différents types d'unités lexicales comme les identificateurs, séparateurs (" ;" , ",", ...), mots clés du langage, opérateurs (affectation, addition, ...), constantes (réelles, entières, chaînes de caractères, ...)

Phase d'analyse

Analyse lexicale (analyse linéaire)

Exemple

$if\,(i < a + b)\; x = 2 * x;$

L'analyseur lexicale déterminera la suite de token : mot_clé, séparateur, ident, op_rel, ident, op_arithm, ident, séparateur, ident, affectation, cste, op_arithm, ident.

Exemple

Règles :

- Un *identificateur* commence obligatoirement par une lettre ou " _" suivi d'une suite de lettres et de chiffres
- ...

Phase d'analyse

Analyse lexicale (analyse linéaire)

Les programmes tel que LEX permettent de générer automatiquement des programmes qui une fois compilés constituent des analyseurs lexicaux.

Phase d'analyse

Analyse syntaxique (analyse hiérarchique ou grammaticale)

But : Vérifier que les unités lexicales sont dans le bon ordre défini par le langage : regrouper les unités lexicales en structures grammaticales et découvrir la structure du programme. L'analyseur syntaxique sait comment sont construites les expressions, les instructions, les déclarations de variables, les appels des fonctions, etc.

Exemple

En C, une instruction conditionnelle doit se présenter sous forme :
if (*expression*) *instruction*
Si l'analyseur syntaxique recoit la suite d'unités léxicales : MC_IF, IDENT, OPREL, ENTIER...
Il doit signaler une erreur (pas de parenthèse après le **if**).

Phase d'analyse

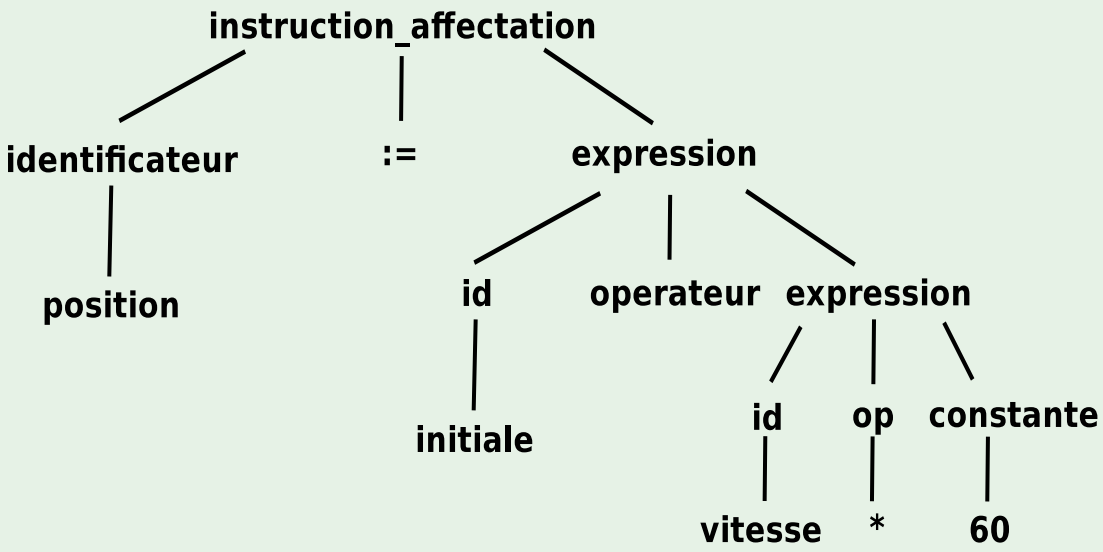
Analyse syntaxique (analyse hiérarchique ou grammaticale)

L'analyseur syntaxique produit un arbre syntaxique.

Exemple

position := *initiale* + *vitesse* ∗ 60;

arbre syntaxique :



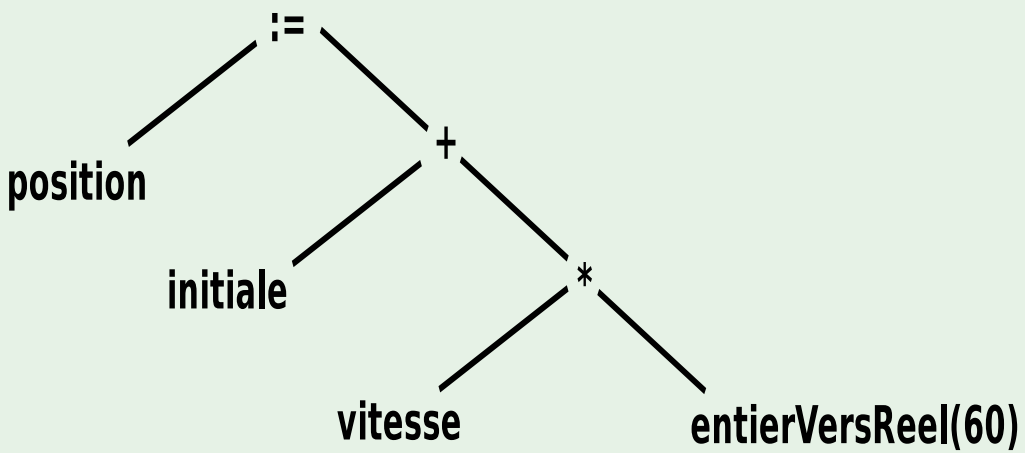
Phase d'analyse

Analyse sémantique (analyse contextuelle)

But : Dans cette phase on opère certains contrôles (contrôles de type, par exemple) afin de vérifier que l'assemblage des constituants a un sens. On ne peut pas, par exemple, additionner un réel avec une chaîne de caractères, ou affecter une variables à un nombre ... (erreurs sémantiques).

Exemple

Dans notre exemple, si vitesse est définie comme un réel et si 60 est reconnu comme un entier, 60 va etre converti en réel.



Phase synthèse

But : Production du code cible à partir de l'arbre syntaxique et de la table des symboles.

3 phases :

- génération du code intermédiaire
- optimisation du code intermédiaire
- production du code cible

Phase synthèse

Génération du code intermédiaire

But : Production d'instructions faciles à traduire en langage cible.

- Le langage cible est défini par le type de processeur utilisé
  - Problèmes de portabilités
  - Introduction de machines dites abstraites (virtuelle) qui font abstraction des architectures réelles
  - Production d'instructions pour une machine virtuelle
  - Traduction en instructions directement exécutables par la machine réelle.

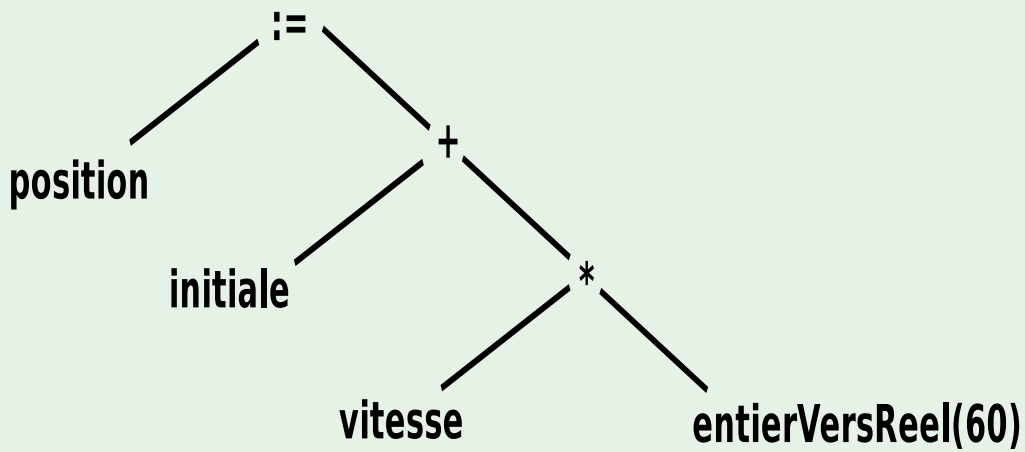
Exemple

Code intermédiaire :
code à 3 adresses $\Rightarrow$ instructions d'affectations dont la partie gauche est un nom temporaire avec au plus 3 opérandes (:= est une opération en plus)

Phase synthèse

Génération du code intermédiaire

Exemple



$temp1 := entierVersReel(60)$
$temp2 := vitesse * temp1$
$temp3 := initiale + temp2$
$position := temp3$

Phase synthèse

Optimisation du code intermédiaire

But : production du code intermédiaire plus optimisé par rapport à certains critères (temps, espace, ...)

- Elimination de calculs inutiles (fait en double)
- Propagation de constantes
- Il y'a des optimisation qui dépendent de la machine cible :
  - Remplacer des instructions générales par des instructions plus efficaces
  - Utilisation optimale des registres
- ...

Phase synthèse

Production du code machine

But : produire du code machine (par exemple en langage assembleur)

Exemple

$$temp1 := vitesse * 60$$

$$position := initiale + temp1$$

Avec 2 registres R1 et R2

MOVF vitesse, R2 (F $\Rightarrow$ virgule flottante)
MULF 60.00, R2 ($\Rightarrow$ constante)
MOVF initiale, R1
ADDF R2, R1
MOVF R1, position

Phases parallèles

Gestion de la table des symboles

- Construction de la table des symboles.
- Gestion d'erreurs

Phases parallèles

Gestion de la table des symboles

But : Stocker les informations sur les identificateurs, leurs types, leurs emplacements en mémoire, leurs portés, ...
Si c'est un nom de procédure : nombre et type des arguments, mode de passage, type de retour

Table des symboles :
Structure servant à stocker les informations sur les identificateurs

Le remplissage de cette table a lieu lors de la phases d'analyse.
Les informations contenues dans la table sont nécessaires lors des analyses syntaxiques et sémantiques, ainsi que lors de la génération du code.

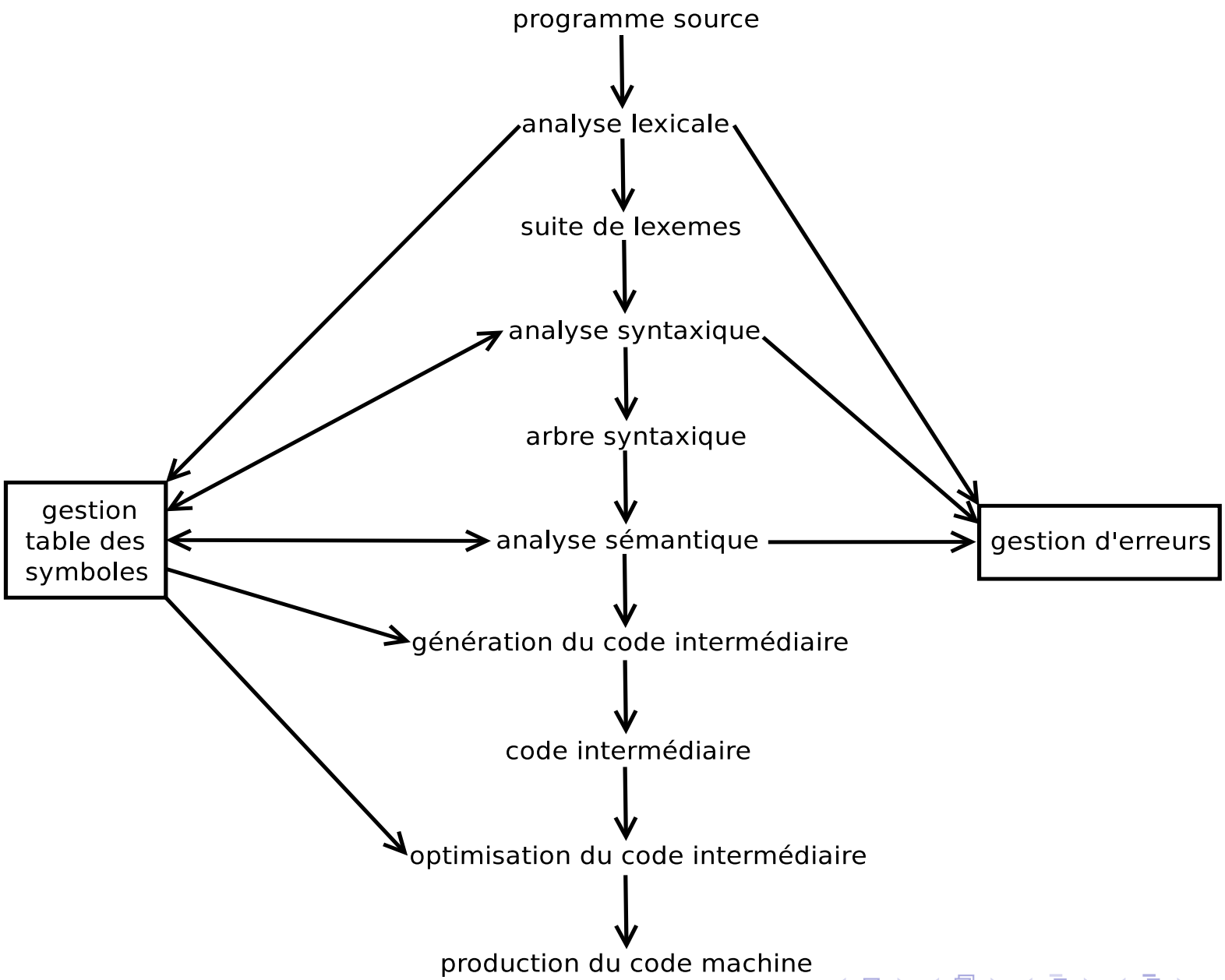
Phases parallèles

Gestion des erreurs

But : Détecter les erreurs et informer l'utilisateur le plus précisement possible.

- Erreurs de syntaxe, erreurs de sémantique, ...
- Traitement des erreurs (de manière à ce que la compilation continue). Ne pas s'arrêter à la première erreur.

Résumé


programme source
analyse lexicale
suite de lexemes
analyse syntaxique
arbre syntaxique
gestion
table des
symboles
analyse sémantique
gestion d'erreurs
génération du code intermédiaire
code intermédiaire
optimisation du code intermédiaire
production du code machine

Outils formels

Dans la phase d'analyse :

- analyse lexicale $\rightarrow$ expressions régulières, automates à états finis
- analyse syntaxique $\rightarrow$ grammaire, automates à pile
- analyse sémantique $\rightarrow$ traduction dirigée par la syntaxe.