Chapitre 1

Introduction à la compilation

Yousra Hlaoui

Faculté des Sciences de Tunis
Département des Sciences Informatique

Sommaire

Introduction

Évolution des langages

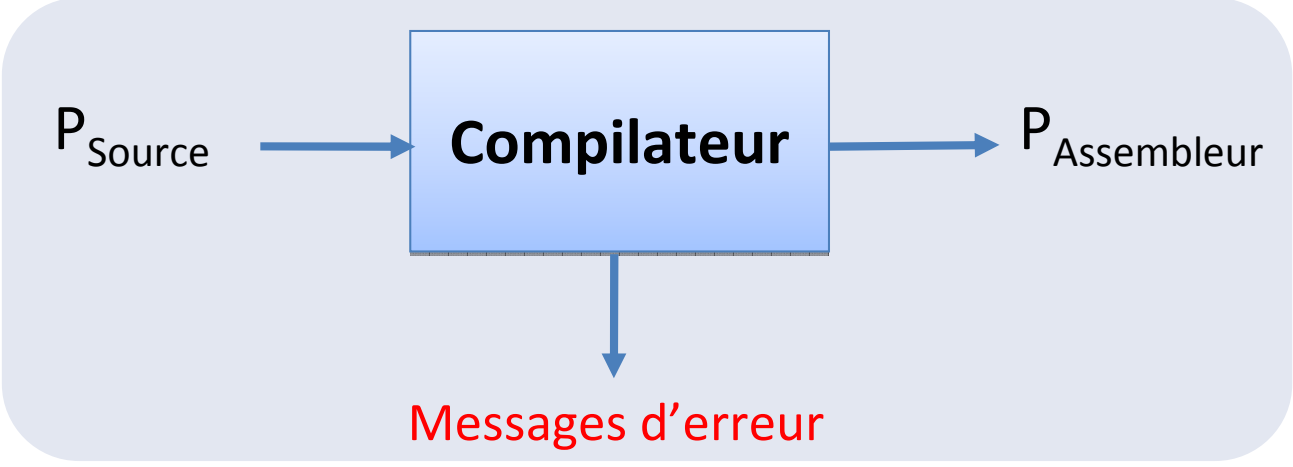
- Langage machine
- Langage assembleur
- Langages évolués :
  - Algorithmiques : C, PASCAL, ADA
  - Logiques : PROLOG
- Langages de $4^{ème}$ génération

Qu'est ce qu'un compilateur ?

- Un compilateur est un programme qui lit un programme écrit dans un premier langage (langage source) et le traduit en un programme équivalent écrit dans un autre langage (le langage cible).

Introduction

- Au cours de ce processus de traduction
  - Le compilateur signale à son utilisateur la présence d'erreurs dans le programme source.



1. Phases de compilation

- On distingue deux catégories :
  - Phases d'analyse
    - Partitionne le programme source en ses constituants et en crée une représentation intermédiaire.
  - Phases de synthèse ou de production
    - Construit le programme cible à partir de la représentation intermédiaire.

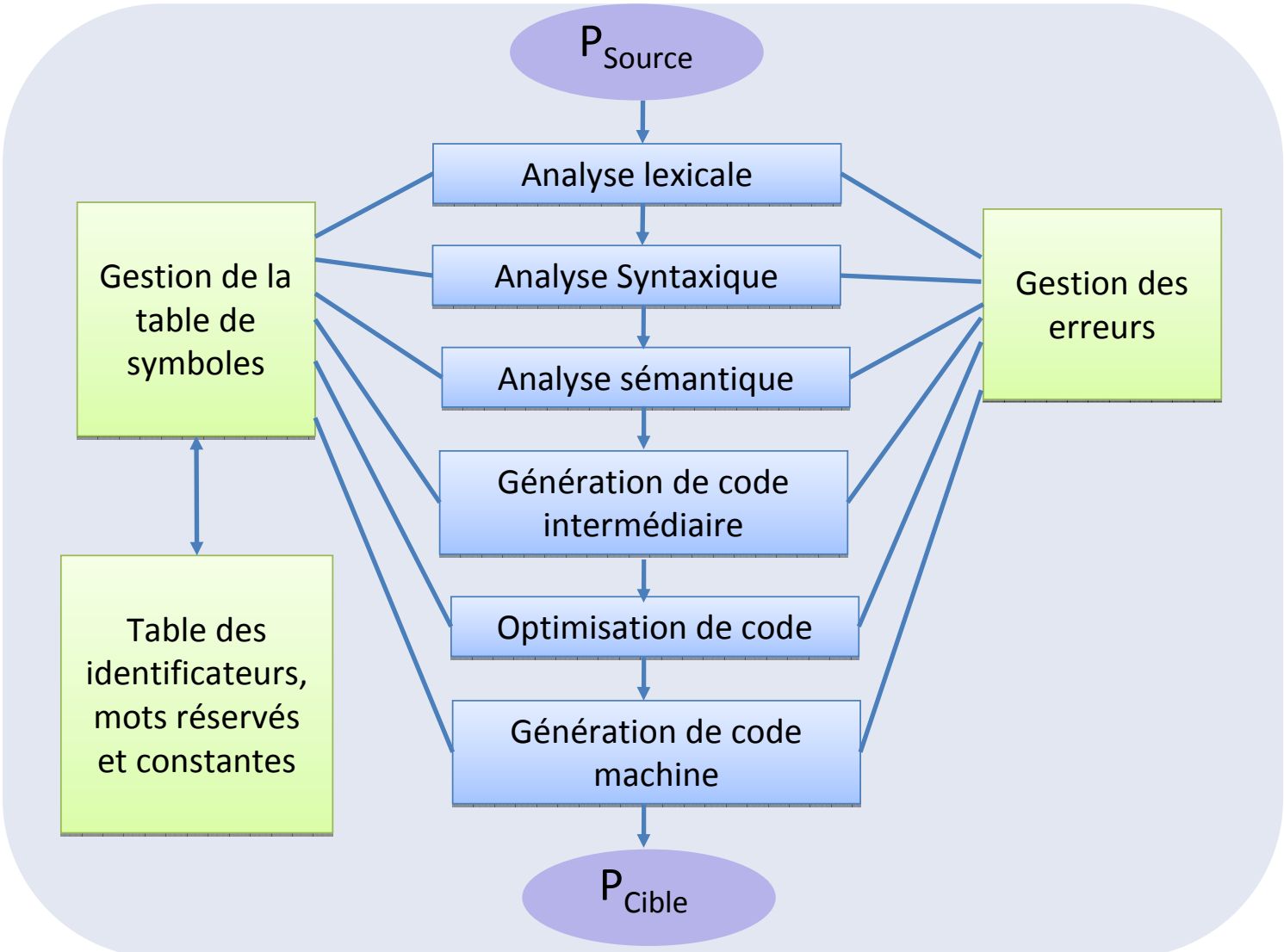
Phase d'analyse

- Analyse lexicale
- Analyse syntaxique
- Analyse sémantique

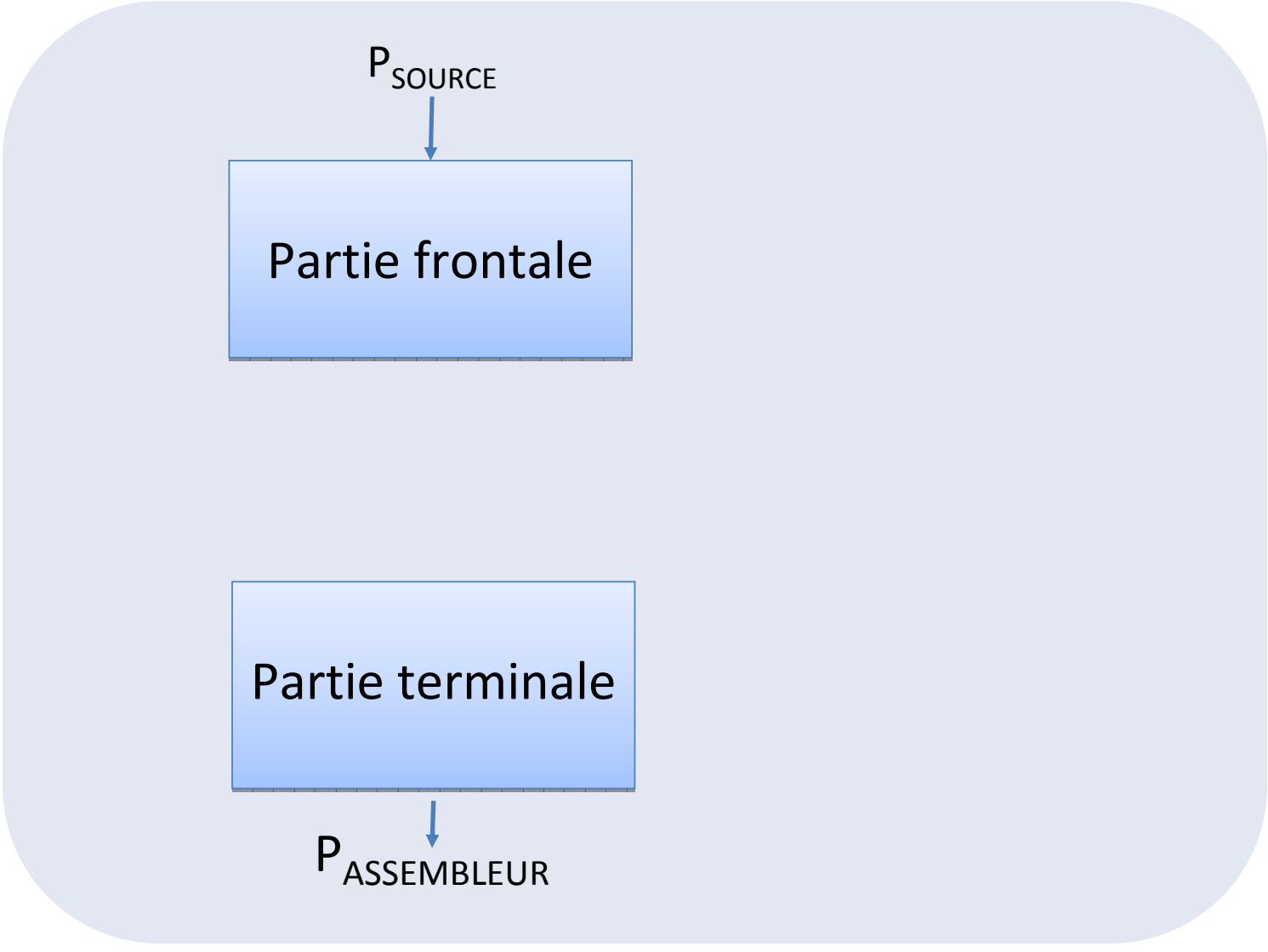
Phase de production

- Génération de code intermédiaire
- Optimisation de code
- Génération de code

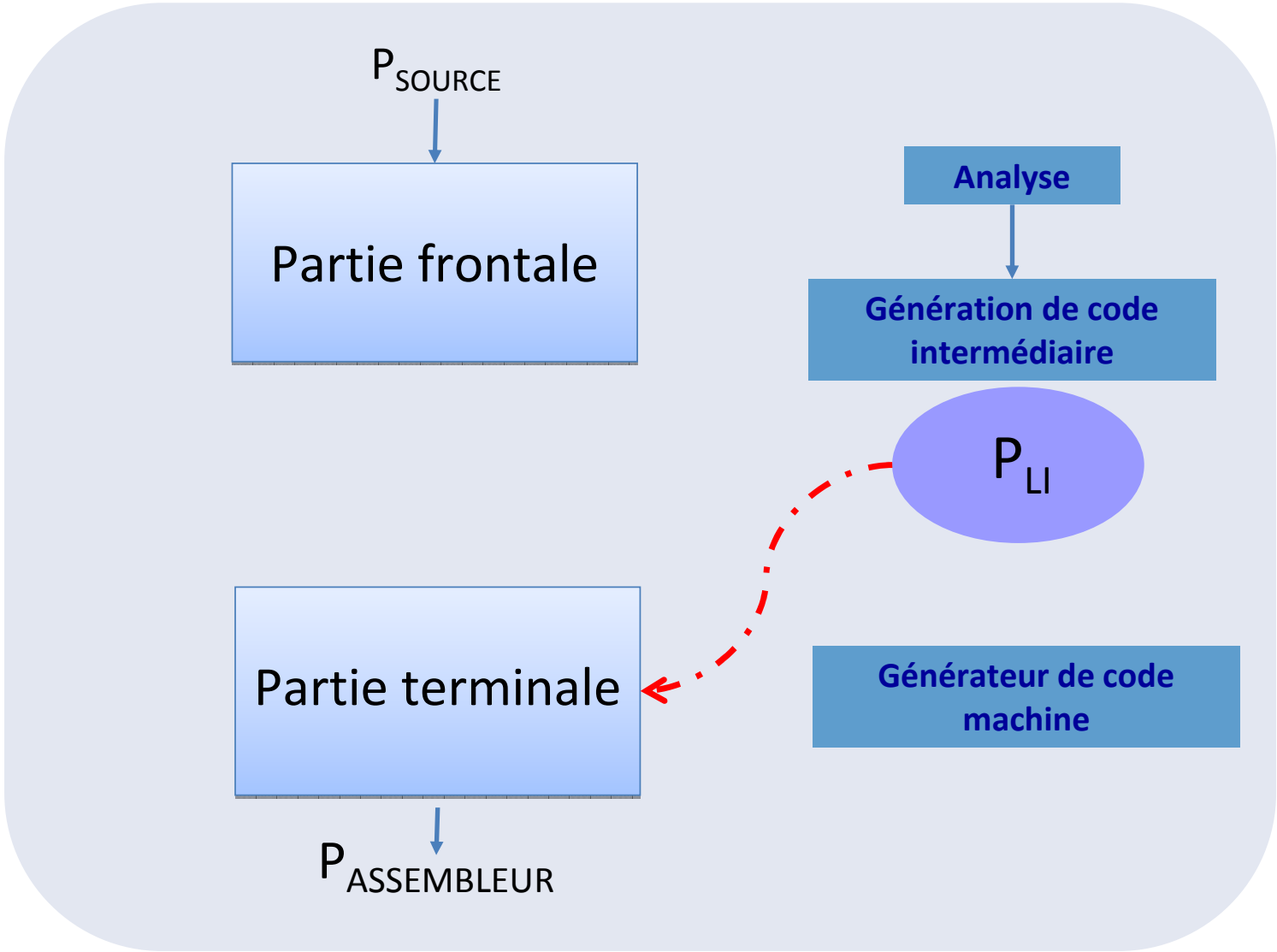
2. Structure d'un compilateur



2. Structure d'un compilateur



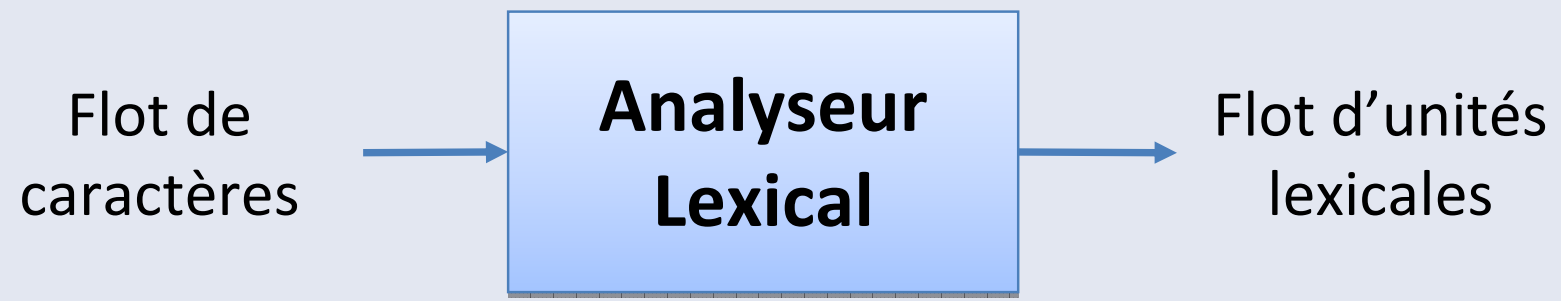
2. Structure d'un compilateur



3. Analyse lexicale

Analyse lexicale ou linéaire

- Au cours de laquelle le flot de caractères formant le programme source est lu de gauche à droite.
- Ce flot de caractères est groupé en unités lexicales qui sont une suite de caractères ayant une signification collective
- Ces unités lexicales (tokens) : identificateurs, constantes, opérateurs, séparateurs (parenthèses ou points virgules) ou mots clefs du langage



3. Analyse lexicale

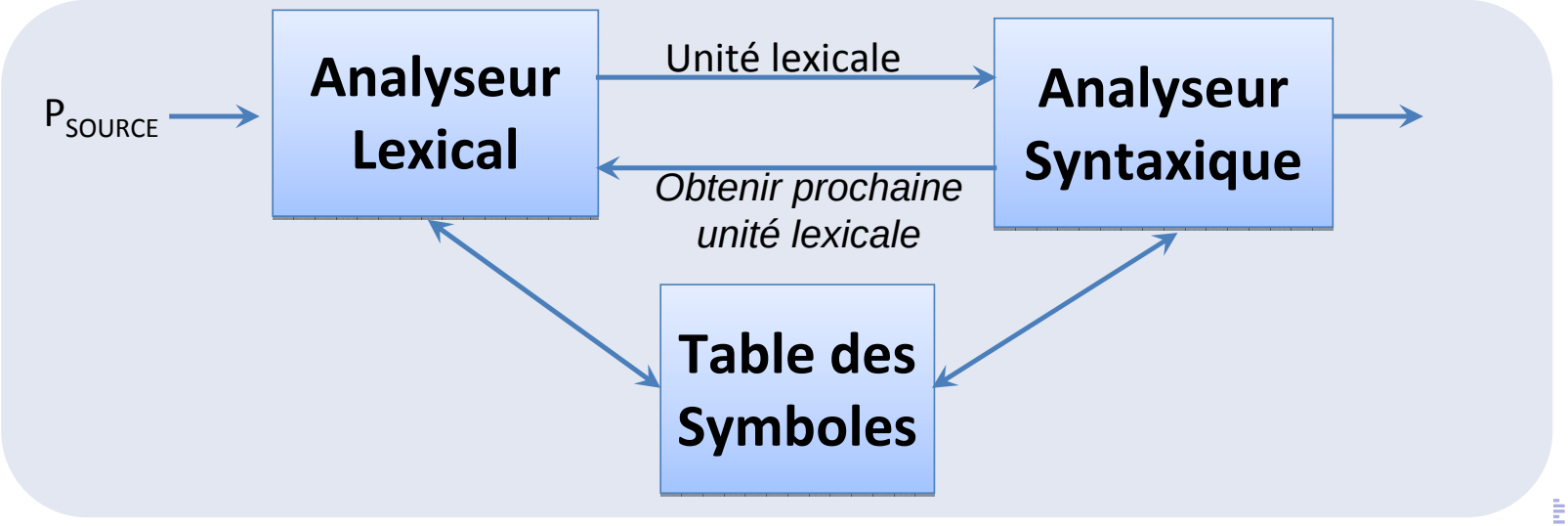
Exemple

- L'analyse lexicale de l'instruction d'affectation :
  - position := initiale + vitesse*60
- Regroupe les caractères formant cette instruction en des unités lexicales qui sont les suivantes :
  1. L'identificateur position ;
  2. Le symbole d'affectation := ;
  3. L'identificateur initiale ;
  4. Le signe + ;
  5. L'identificateur vitesse ;
  6. Le signe * ;
  7. Le nombre 60 ;

4. Analyse syntaxique

Analyse syntaxique, hiérarchique ou grammaticale

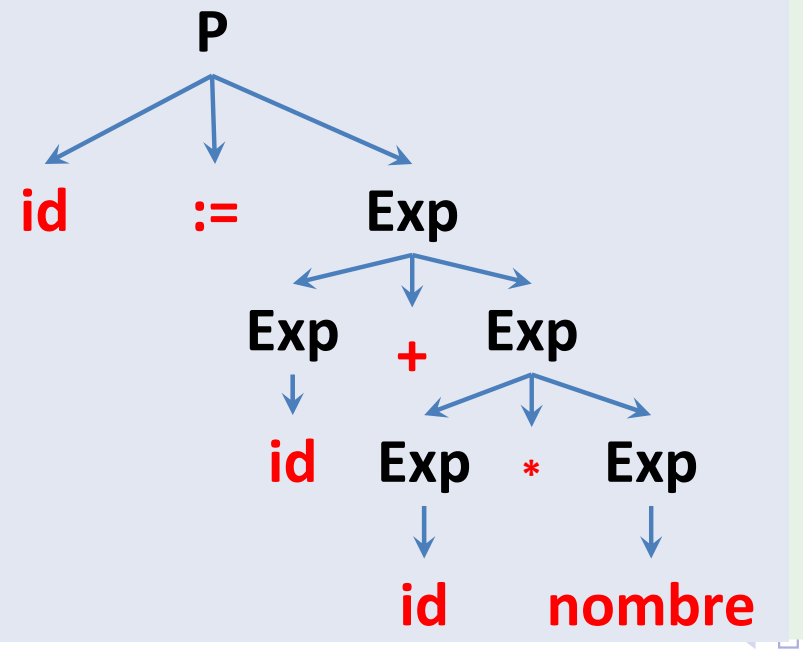
- Consiste à regrouper les unités lexicales du programme source en structures grammaticales qui seront utilisées par le compilateur pour synthétiser le résultat
- Les phrases grammaticales sont représentées par un arbre syntaxique dont les feuilles concordent avec la suite d'unités lexicales en les parcourant de gauche à droite.



4. Analyse syntaxique

Exemple

- L'instruction position := initiale + vitesse*60 est générée par la grammaire :
  - $P \rightarrow id := Exp;$
  - $Exp \rightarrow id|nb|(Exp)|Exp + Exp|Exp * Exp$
- L'analyse syntaxique génère l'arbre syntaxique suivant :



5. 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. On ne peut pas, par exemple, additionner un réel avec une chaîne de caractères, ou affecter une variable à un nombre.
- Dans le contrôle de type, l'analyseur sémantique peut insérer des opérations de conversion pour les expressions d'entier vers réel.
- Nous distinguons deux types d'erreurs sémantiques :
  - ***sémantiques statiques*** : contrôlées au moment de la ***compilation*** (variable non déclarée, incompatibilité de type, portée d'une variable)
  - ***sémantiques dynamiques*** : contrôlées au moment de l'***exécution*** (division par zéro, boucles infinies, débordement mémoire)

5. Analyse sémantique

Exemple

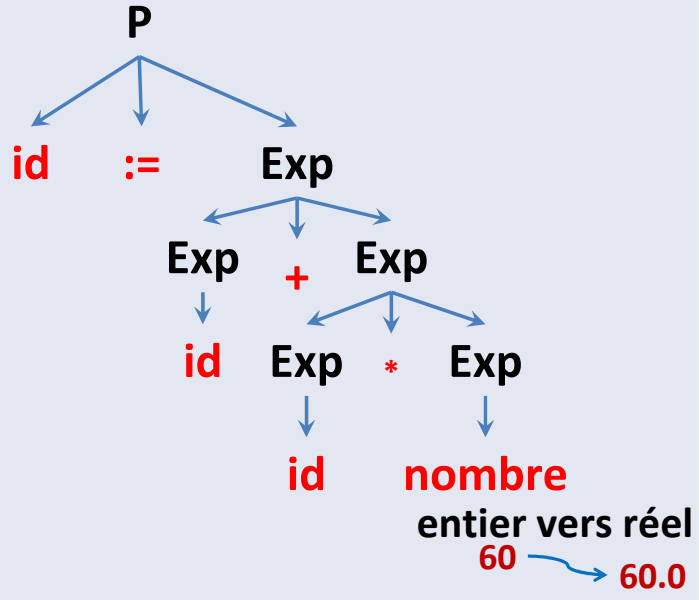
- Une Contrainte sémantique doit être vérifiée :
  - L'opérateur de multiplication doit être appliqué à deux opérandes de même type (entier, entier) ou (réel, réel)
  - Pour notre exemple, Vitesse est un réel et 60 est un nombre entier, il faut alors convertir 60 d'entier vers réel (60.0)

Var
position, Initiale, Vitesse : réel;

Analyseur Sémantique

Table des id de variables

N°	Lexème	Type
1	Position	Réel
2	Initiale	Réel
3	Vitesse	Réel



6. Phase de génération de code intermédiaire

Phase de génération de code intermédiaire

- Au cours de laquelle la séquence d'instructions du programme est traduite en une séquence d'instructions dans un langage intermédiaire (le langage pour machine à pile, le langage C,)
- La représentation intermédiaire doit avoir être :
  1. facile à produire
  2. facile à traduire en langage cible

Exemple : traduction en un code à trois adresses

- `Temp1 := EntierVersRéel(60);`
- `Temp2 := Temp1 * id3;`
- `Temp3 := id2 + Temp2;`
- `id1 := Temp3;`

7. Optimisation de code

Optimisation de code

- Cette phase tente d'améliorer le code produit de telle sorte que le programme résultant soit plus rapide : élimination du code d'une fonction jamais appelée, propagation des constantes, extraction des boucles des invariants de boucle.

Exemple

- $Temp1 := id3 * 60.0$;
- $id1 := id2 + Temp1$;

8. Génération de code cible

Génération de code cible

- Il s'agit de produire les instructions en langage cible qui est dans ce cas défini par le type de processeur utilisé.

Exemple

- Génération de code intermédiaire (Machine VON NEUMAN) Machine à registres
  - MOVF id3, R2
  - MULF 60.0, R2
  - MOVF id2, R1
  - ADDF R2, R1
  - MOVF R1, id1

9. Phases parallèles

1. Gestion de la table des symboles

- La table des symboles Contient des informations sur les différents symboles et les attributs associés.
- Par exemple les mots clés (ou réservés) et les identificateurs de variables en spécifiant l'unité lexicale **id** et leurs attributs : type, adresse, etc
- A chaque fois qu'une unité lexicale **id** est trouvée par l'AL, le lexème associé est inséré dans la table des symboles s'il n'a pas été déjà inséré et l'AL retourne **id** et un pointeur vers une entrée de la table des symboles.

7. Phases parallèles

2. Gestion des erreurs

- Chaque phase peut rencontrer des erreurs. Il s'agit de les détecter et d'informer l'utilisateur le plus précisément possible : erreur de syntaxe, erreur de sémantique...
- Un compilateur qui se contente d'afficher syntax error n'apporte pas beaucoup d'aide lors de la mise au point.

Qualité d'un compilateur

- Fournir le maximum d'erreurs en une seule compilation
- Rapidité

Outils pour la construction de compilateurs

- Écriture en langage évolué
- Constructeurs automatiques de compilateurs : LEX et YACC