Compilation (#1a) – Aperçu

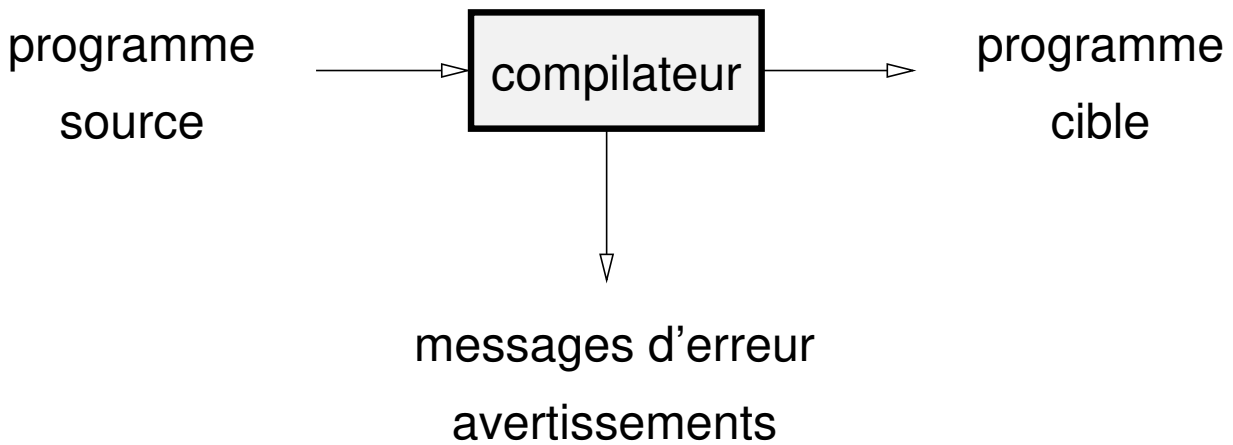
C. Deleuze & L. Gonnord

Grenoble INP/Esisar

2022-2023

Plan

Compiler = ?


programme
source
compilateur
programme
cible
messages d'erreur
avertissements

Exemple

```
int fact(int n)
{
  if (n==0) return 1;
  else return(n*fact(n-1));
}
```

```
$ gcc -m32 -S fact.c
```

```
fact:
        pushl %ebp
        movl %esp, %ebp
        subl $24, %esp
        cmpl $0, 8(%ebp)
        jne .L2
        movl $1, %eax
        jmp .L3
.L2:
        movl 8(%ebp), %eax
        subl $1, %eax
        movl %eax, (%esp)
        call fact
        imull 8(%ebp), %eax
.L3:
        leave
        ret
```

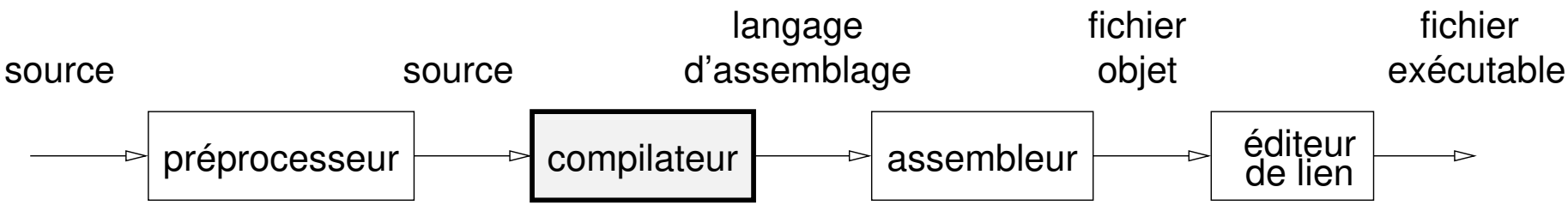
Pourquoi étudier la compilation ?

- pb compliqué, bien résolu
  - structuration du pb
  - utilisation judicieuse de formalismes
  - utilisation d'outils
- touche à presque tout
  - de la manipulation de texte,
  - aux architectures de machines,
  - en passant par les concepts des langages de programmation

Pourquoi étudier la compilation ?

- pb compliqué, bien résolu
  - structuration du pb
  - utilisation judicieuse de formalismes
  - utilisation d'outils
- touche à presque tout
  - de la manipulation de texte,
  - aux architectures de machines,
  - en passant par les concepts des langages de programmation
- la compilation c'est "cool" !

La chaine de compilation



Un programme est . . . une suite de caractères ?

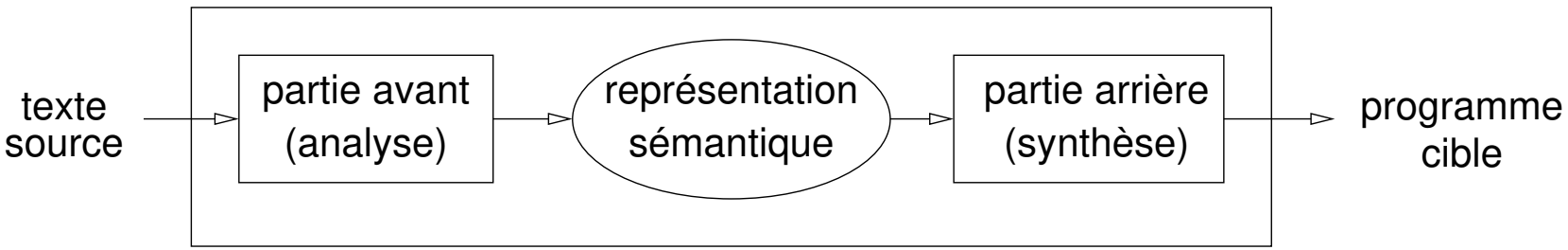
Un programme est . . . une suite de caractères ?

. . . organisés suivant une certaine structure

- structure lexicale
- structure syntaxique
- structure sémantique/contextuelle

```
position := initiale + vitesse * 60
```

Analyse et synthèse

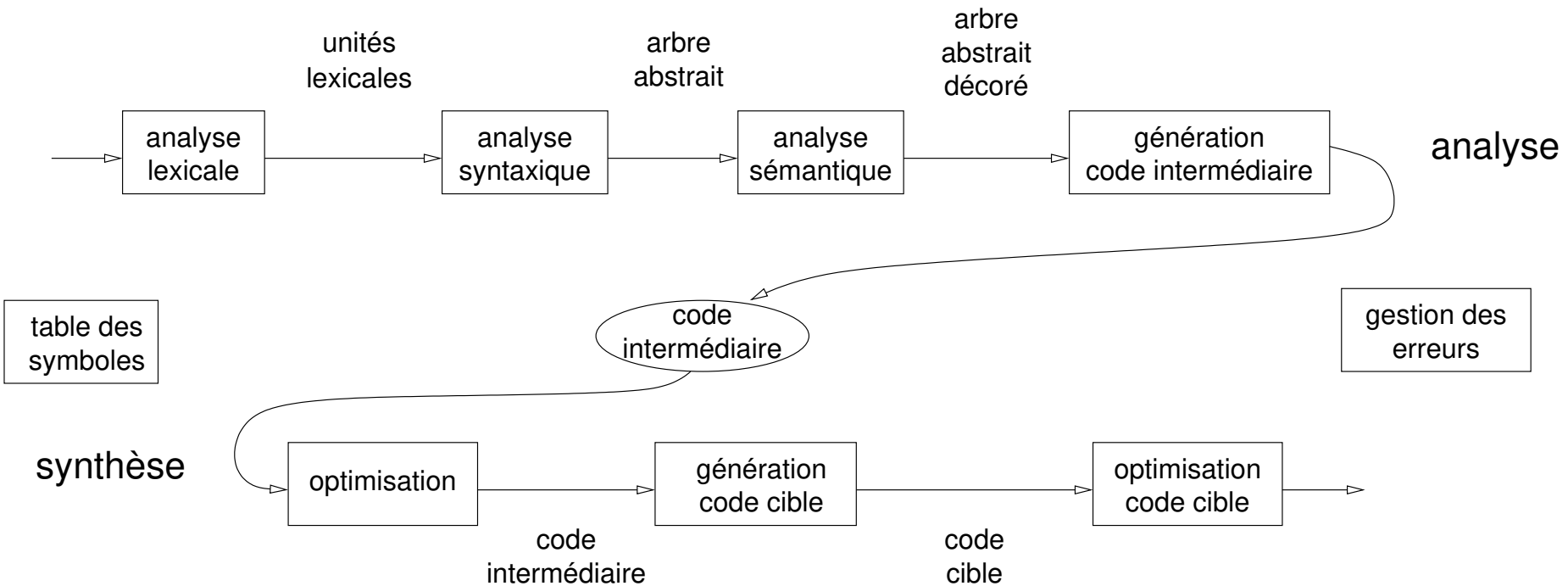


Analyse et synthèse

analyse à partir de la représentation concrète d'un pgm, produire une forme abstraite mettante en évidence sa structure et donc sa sémantique

synthèse à partir d'une représentation abstraite de la sémantique, produire une représentation dans la syntaxe concrète du langage cible

Phases


unités
lexicales
arbre
abstrait
arbre
abstrait
décoré
analyse
lexicale
analyse
syntaxique
analyse
sémantique
génération
code intermédiaire
analyse
table des
symboles
code
intermédiaire
gestion des
erreurs
synthèse
optimisation
génération
code cible
optimisation
code cible
code
intermédiaire
code
cible

Analyse lexicale

mise en oeuvre par le "lexer"

- entrée : flot de caractères
- sortie : flot d'unités lexicales
- élimine blancs et commentaires
- entre les identificateurs dans la table des symboles

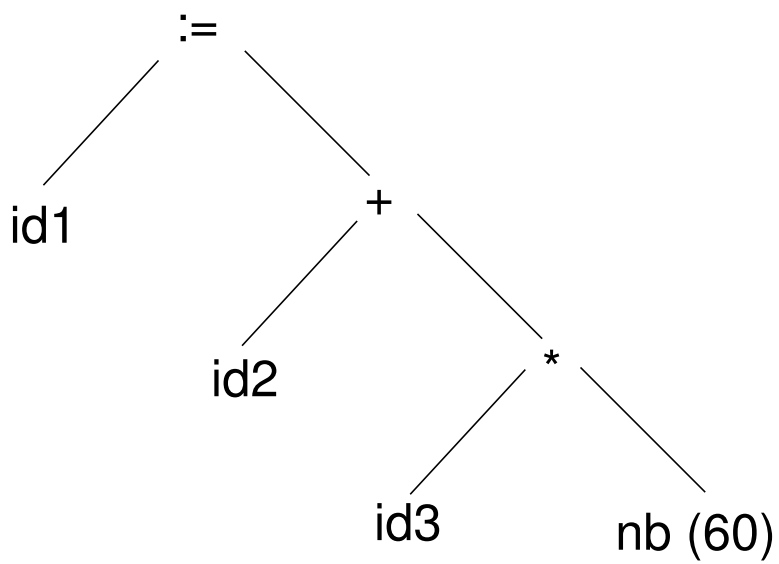
$$\text{position} := \text{initiale} + \text{vitesse} * 60$$

$$id_1 := id_2 + id_3 * nb_{60}$$

Analyse syntaxique

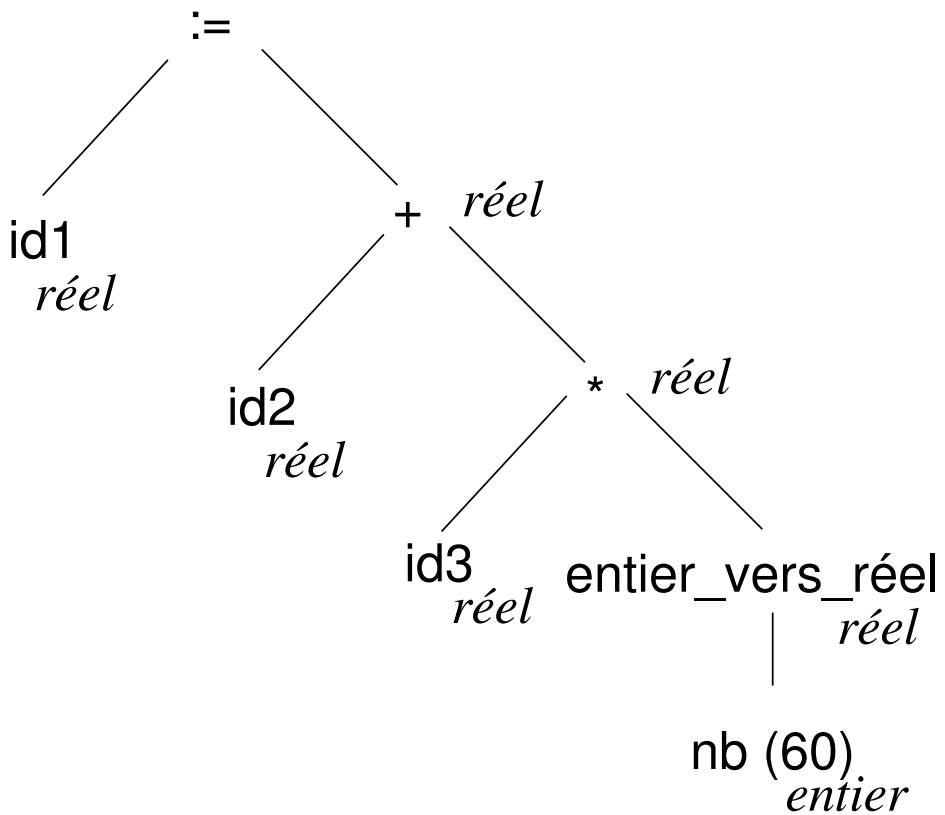
mise en oeuvre par le "parser"

- groupe les UL en structures syntaxiques
- produit un "arbre abstrait"



Analyse sémantique

- collecte les infos de type
  - pour recherche d'erreurs sémantiques
  - et pour usage par les phases suivantes
- "décore" l'arbre abstrait avec des infos de contexte



Génération de code intermédiaire

ex : code à trois adresses

langage d'assemblage pour une machine abstraite

- chaque instruction a trois opérandes (max)
- nombre infini de registres

```
t1 := entierVersRéel(60)
t2 := id3 *r t1
t3 := id2 +r t2
id1 := t3
```

Optimisation de code intermédiaire

optimiser = ?

```
t1 := id3 *r 60.0
id1 := id2 +r t1
```

Code cible

Génération de code cible

- sélection des emplacements mémoire
- traduction de chaque inst CI en séquence d'insts ASM
  - assignation des registres

Optimisation de code cible

- liées à l'architecture et au jeu d'intructions de la machine cible

Table des symboles

utilisée par toutes les phases

la table contient tous les identificateurs du pgm avec les infos les concernant :

- type
- portée
- classe
- adresse
- ...

Gestion des erreurs

Dans les phases d'analyse :

- erreur lexicale
- erreur syntaxique
- erreur sémantique

difficultés :

- donner un message approprié sur l'emplacement et la cause probable de l'erreur
- "récupération" : corriger l'erreur pour continuer l'analyse

Phases et passes

phase une étape conceptuelle dans le travail de compilation

passe parcours de l'ensemble du programme, effectuant un traitement

les trois phases d'analyse lexicale, syntaxique et sémantique sont souvent regroupées dans une seule passe

Plan du cours

séances :

1. aperçu, analyse lexicale
2. analyse syntaxique
3. analyse syntaxique
4. grammaires attribuées
5. arbre abstrait, typage
6. génération de code
7. représentations intermédiaires
8. allocation de registres
9. analyse de flot de données

Bilan