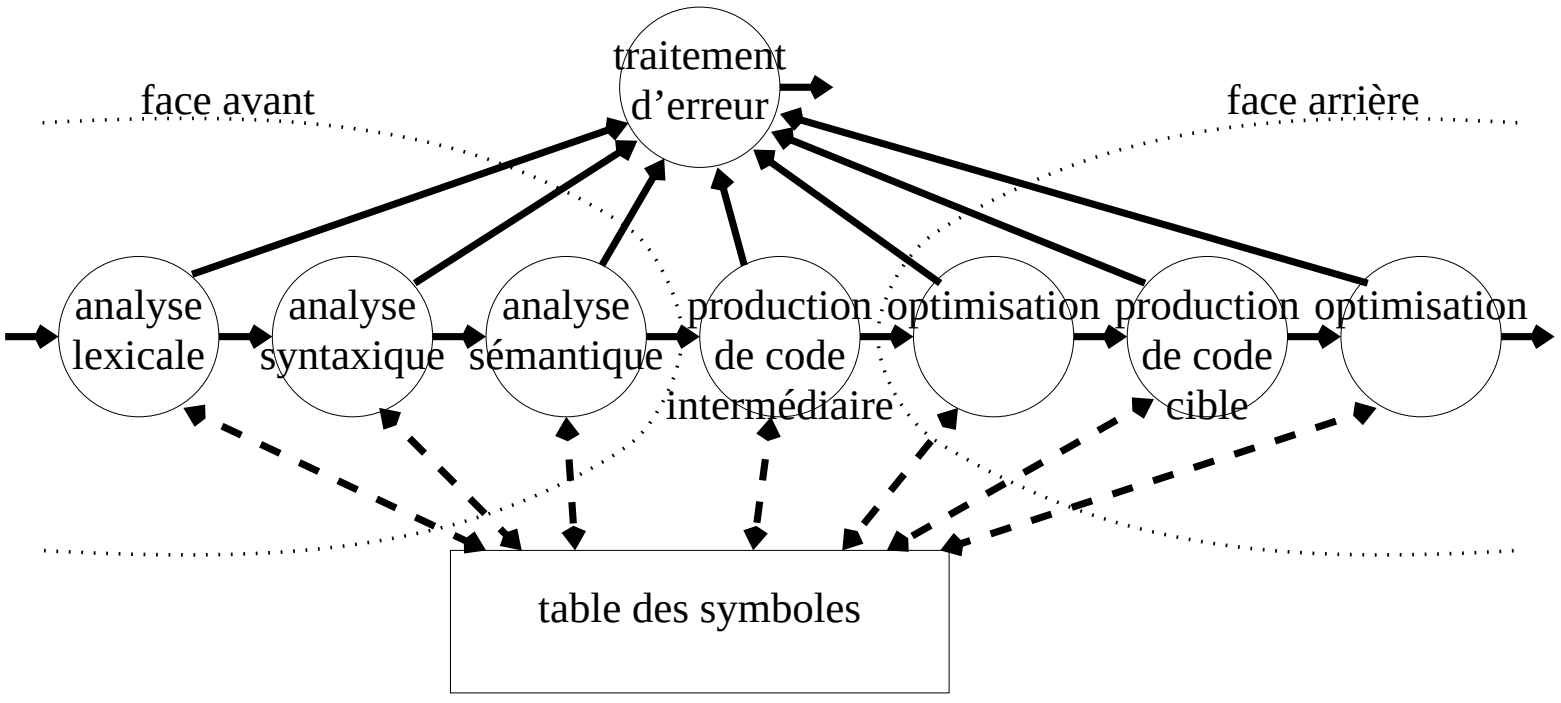
Compilation

CM3 - Analyse sémantique & Grammaires attribuées

ISTIC, Université de Rennes 1
Sebastien.Ferre@irisa.fr

COMP, M1 info

Les phases de la compilation



Plan

Plan

Motivation 1

Se limiter à l'analyse syntaxique pure revient à répondre «oui» à une question comme «Pouvez-vous me donner l'heure ?»

- automate : répond si «oui» ou «non» un mot appartient au langage
- compilateur : produit une traduction du programme source
  - la vérification syntaxique est un passage obligé
  - mais pas suffisant
  - vérification et traduction sont étroitement liés

Motivation 2

Il se peut que le langage source ne soit pas complètement caractérisé par une grammaire hors-contexte.

- constructions contextuelles (dépendences distantes)
  - ex. : variables déclarées avant d'être utilisées
  - cf. grammaires contextuelles (langages de type 1)
- vérifications sémantiques en plus de l'analyse syntaxique proprement dite

Analyse sémantique

Elle succède à l'analyse syntaxique et a 2 rôles :

1. vérification que le programme appartient bien au langage source
   - ex., déclarations avant utilisations, bon typage
2. traduction du programme source (supposé valide) en
   - un résultat final (interpréteur)
     - ex., calculatrice : expression $\rightarrow$ résultat
   - ou un programme cible (compilateur)
   - ou un code intermédiaire (face avant de compilateur)
   - ou un arbre de syntaxe abstraite
     - AST = *Abstract Syntax Tree*
     - différent de l'arbre de dérivation
     - représentation concise et complète du programme source

Analyse sémantique

Elle succède à l'analyse syntaxique et a 2 rôles :

1. vérification que le programme appartient bien au langage source
   - ex., déclarations avant utilisations, bon typage
2. traduction du programme source (supposé valide) en
   - un résultat final (interpréteur)
     - ex., calculatrice : expression $\rightarrow$ résultat
   - ou un programme cible (compilateur)
   - ou un code intermédiaire (face avant de compilateur)
   - ou un arbre de syntaxe abstraite
     - AST = *Abstract Syntax Tree*
     - différent de l'arbre de dérivation
     - représentation concise et complète du programme source

Analyse sémantique

Elle succède à l'analyse syntaxique et a 2 rôles :

1. vérification que le programme appartient bien au langage source
   - ex., déclarations avant utilisations, bon typage
2. traduction du programme source (supposé valide) en
   - un résultat final (interpréteur)
     - ex., calculatrice : expression $\rightarrow$ résultat
   - ou un programme cible (compilateur)
   - ou un code intermédiaire (face avant de compilateur)
   - ou un arbre de syntaxe abstraite
     - AST = *Abstract Syntax Tree*
     - différent de l'arbre de dérivation
     - représentation concise et complète du programme source

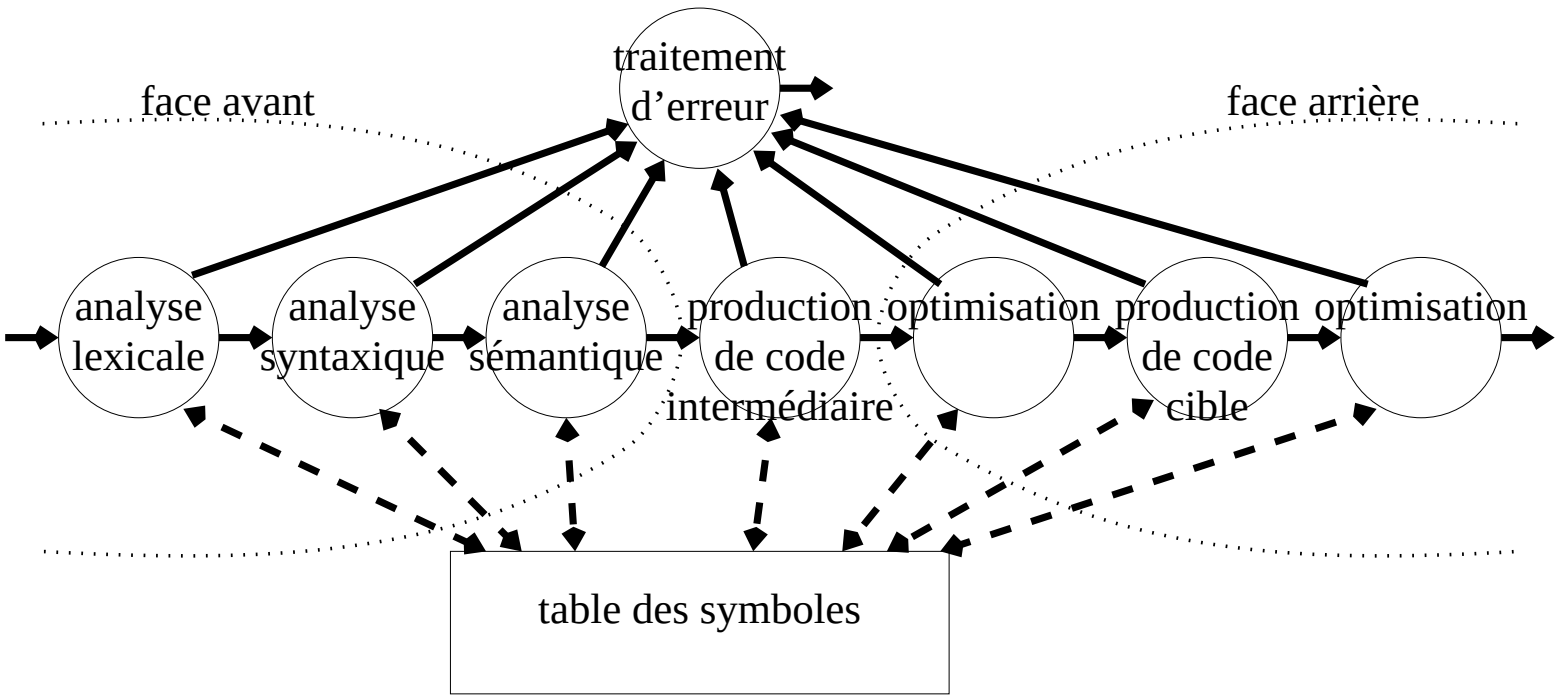
Analyse sémantique

Elle succède à l'analyse syntaxique et a 2 rôles :

1. vérification que le programme appartient bien au langage source
   - ex., déclarations avant utilisations, bon typage
2. traduction du programme source (supposé valide) en
   - un résultat final (interpréteur)
     - ex., calculatrice : expression $\rightarrow$ résultat
   - ou un programme cible (compilateur)
   - ou un code intermédiaire (face avant de compilateur)
   - ou un arbre de syntaxe abstraite
     - AST = *Abstract Syntax Tree*
     - différent de l'arbre de dérivation
     - représentation concise et complète du programme source

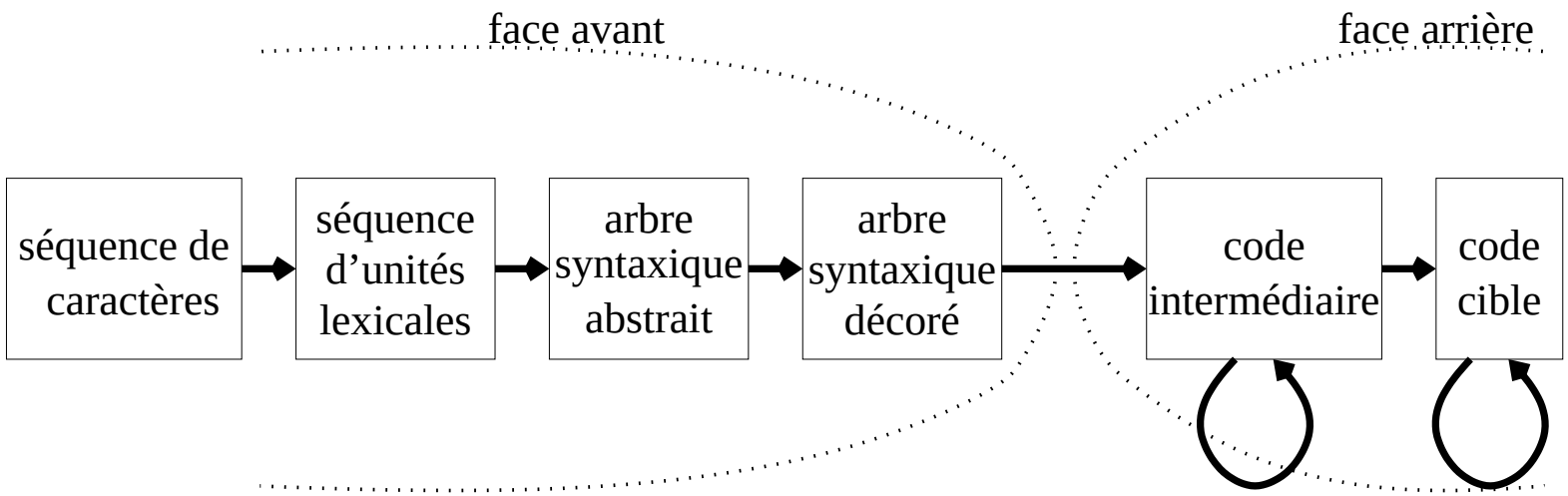
Les phases de la compilation

- elles forment un pipeline
- pas nécessairement en séquence stricte



Les représentations du programme

La représentation du programme évolue d'une phase à l'autre.



Plan

Grammaires attribuées

Comment spécifier les analyses sémantiques ?

- en les ancrant dans la syntaxe
  - garantit une certaine adéquation entre syntaxe et sémantique
- attributs attachés aux symboles de la grammaires
  - $X.a$ dénote l'attribut a attaché au symbole X
  - un attribut peut être de n'importe quel type
- expressions (calculs) attachés aux règles de la grammaire
  - $X_0.a_0 := f(X_1.a_1, \ldots, X_n.a_n)$, où les $X_i.a_i$ sont des attributs des symboles de la production
  - $X_0.a_0$ dépend des autres attributs

Grammaire attribuée = grammaire + attributs (par symbole) + expressions (par production)

Grammaires attribuées

Comment automatiser les analyses sémantiques ?

- en décorant l'arbre de dérivation
- chaque occurence de symbole X dans l'arbre (noeuds et feuilles) a ses propres instances d'attributs a
- chaque occurence de dérivation $\beta X \gamma \Rightarrow \beta \alpha \gamma$ (noeud de l'arbre) doit vérifier les expressions attachées à la règle correspondante $X \rightarrow \alpha$
  - on doit éviter les dépendences circulaires

Grammaires attribuées

Comment automatiser les analyses sémantiques ?

- en décorant l'arbre de dérivation
- chaque occurence de symbole X dans l'arbre (noeuds et feuilles) a ses propres instances d'attributs a
- chaque occurence de dérivation $\beta X \gamma \Rightarrow \beta \alpha \gamma$ (noeud de l'arbre) doit vérifier les expressions attachées à la règle correspondante $X \rightarrow \alpha$
  - on doit éviter les dépendences circulaires

Attributs hérités vs synthétisés

- Attribut synthétisé
  - correspond à un résultat (sortie) de l'analyse sémantique du sous-arbre
  - dépend uniquement des noeuds fils
  - permet de produire la traduction du programme source
  - pour les terminaux, la valeur est définie lors de l'analyse lexicale
    - attribut `text` dans ANTLR
- Attribut hérité
  - correspond à un paramètre (entrée) de l'analyse sémantique du sous-arbre
  - dépend des attributs des noeuds parents et des noeuds frères (gauches en général)
  - permet de prendre en compte le contexte (vérifications)
    - ex., table des symboles déclarés

Plan de la suite du CM3

Exemple d'un langage d'expressions arithmétiques

- calcul de la valeur d'une expression (interpreteur)
  - analyse ascendante avec attributs synthétisés uniquement
  - analyse descendante avec attributs hérités et synthétisés
- calcul de l'arbre de syntaxe abstraite (AST) (compilateur)
  - grammaire attribuée
  - grammaire avec actions (outils : ex., ANTLR)
    - intégration des calculs à un analyseur descendant (automate)
    - sans construction de l'arbre de dérivation
    - attributs hérités/synthétisés comme entrées/sorties des non-terminaux

. . . . . .