CSC4251_4252 : Attributs Sémantiques et Table des Symboles

Pascal Hennequin, J. Paul Gibson, Denis Conan

Novembre 2024

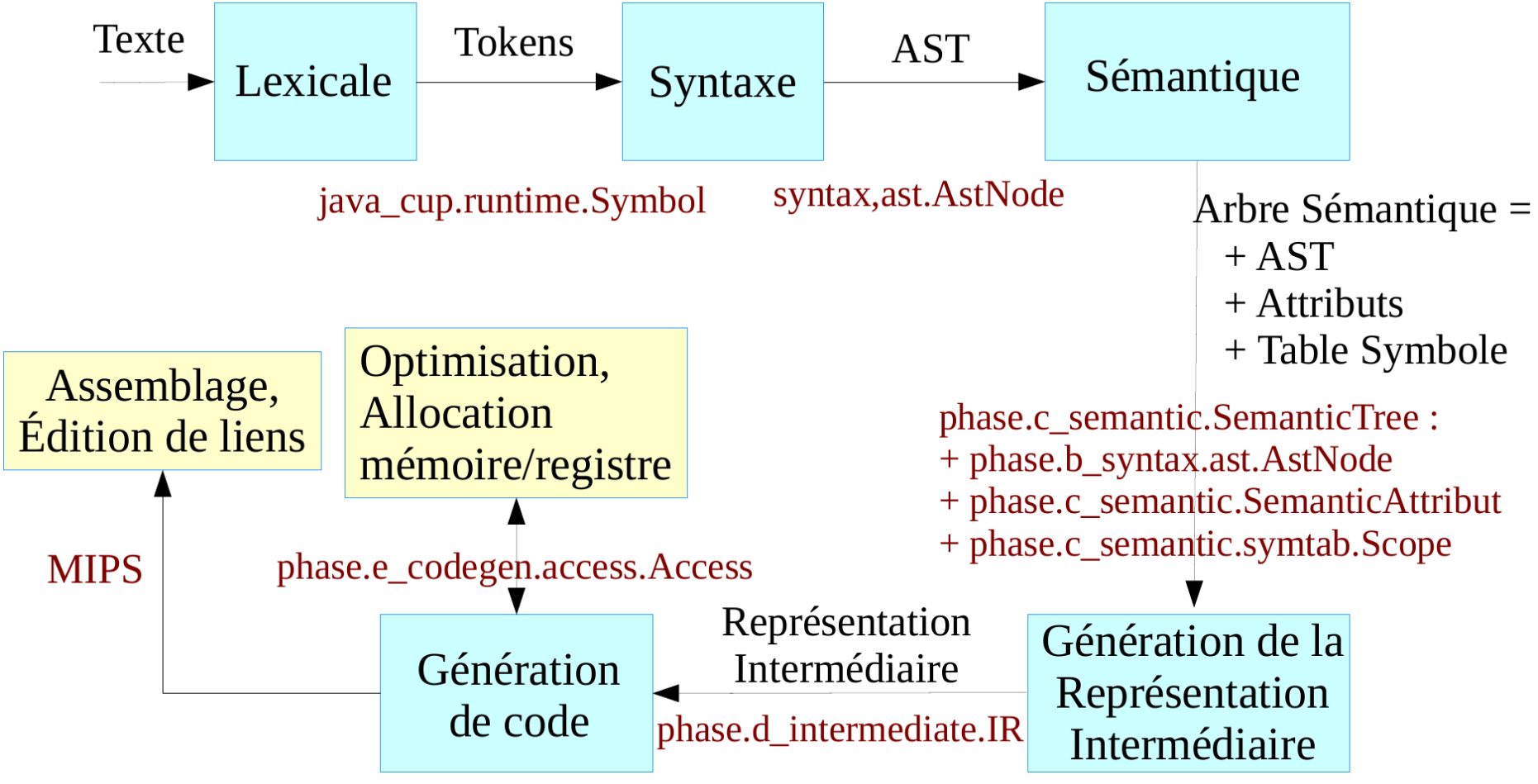
Préambule

- Jusqu'ici :
  - Plus de théorie sur l'analyse lexicale (automates) et syntaxique (grammaire, machine de calcul, et langage)
  - On construit un langage et son compilateur à partir de rien
- Et maintenant :
  - Plus de pratique pour aller jusqu'à la génération et l'exécution de code assembleur MIPS
  - On s'insère dans du code existant (beaucoup)
  - On ajoute des aspects génie logiciel pour construire le compilateur

Sommaire

1 Où en est-on ?

- **Retour sur les phases de compilation**



2 Objectifs de l'analyse sémantique

- Valider les règles du langage difficilement gérables au niveau syntaxique
- Établir la signification du programme et sa sémantique d'exécution
- Construire les éléments de contexte
  - Informations nécessaires pour l'exécution mais non explicites dans l'arbre de syntaxe, ou explicites dans une autre partie de l'arbre
- Vérifier ou Inférer des propriétés de l'algorithme
  - Terminaison, validité de l'algorithme, non débordement, ...
  - N.B. : Théorème de Rice $\Longrightarrow$ problèmes indécidables

3 Analyse Sémantique et AST

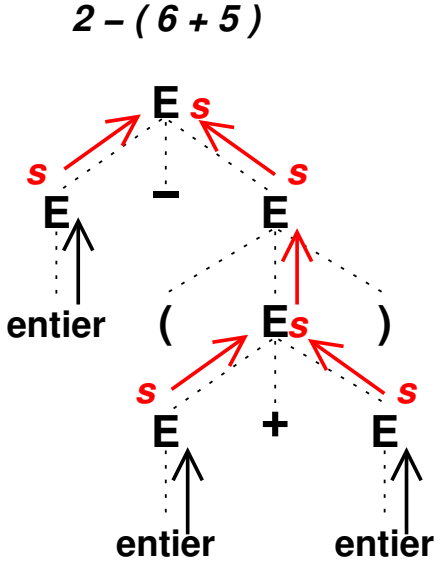
- **Analyse Sémantique = Décoration de l'AST**
  - Calcul d'attributs associés aux différents nœuds de l'arbre de syntaxe
    - Attributs sémantiques = informations ajoutées lors de la phase de l'analyse sémantique pour raisonner sur les programmes analysés
  - Pour des fonctions sémantiques (Cf. le prochain cours surtout)
    - Visibilité des identificateurs (dans ce cours)
    - Contrôle des types de données
    - Analyse de flot de données
    - ...

4 Attributs sémantiques

4.1 Attribut Synthétisé

- **Attribut Synthétisé** =
  - Tout nœud de l'AST possède potentiellement l'attribut
  - Valeur d'attribut au nœud N =
    f(valeurs d'attribut des enfants de N, N lui-même)
- P.ex., expressions arithmétiques
  - Attribut synthétisé « s » (E.s, E_1.s, E_2.s)
  - Notation : E différencié de E_1, E_2
    pour différencier les nœuds

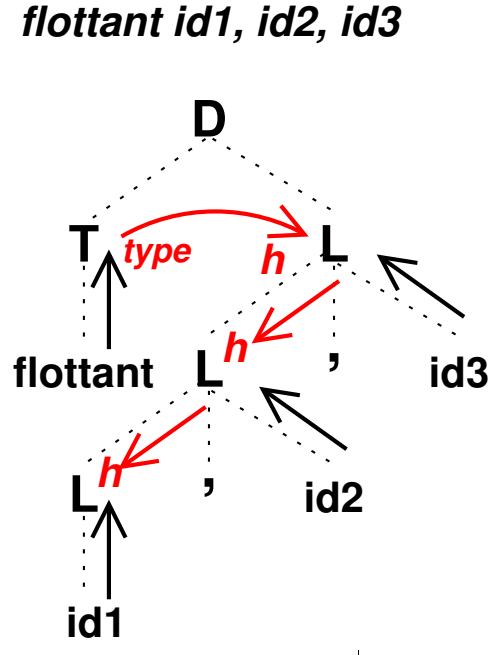
Grammaire	Règles sémantiques (f)
$\langle E \rangle := \langle E_1 \rangle + \langle E_2 \rangle$	$E.s = E_1.s + E_2.s$
$\langle E \rangle := \langle E_1 \rangle * \langle E_2 \rangle$	$E.s = E_1.s \times E_2.s$
$\langle E \rangle := (\langle E_1 \rangle)$	$E.s = E_1.s$
$\langle E \rangle :=$ entier	$E.s =$ entier



4.2 Attribut Hérité

- **Attribut Hérité =**
  - Tout nœud de l'AST possède potentiellement l'attribut
  - Valeur d'attribut au nœud N =
    f(valeur d'attribut du parent, N, valeurs d'attribut des frères)
- P.ex., déclaration de type dans un langage de programmation
  - Attribut hérité h (L.h)
  - Notation : L différencié de L_1 pour différencier les nœuds

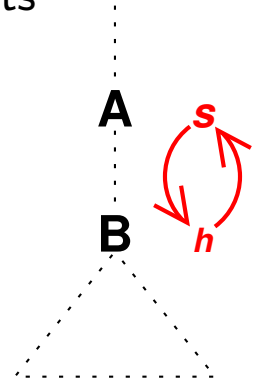
Grammaire	**Règles sémantiques (f)**
$\langle D \rangle$:= $\langle T \rangle$ $\langle L \rangle$	L.h = T.type
$\langle T \rangle$:= entier	T.type = entier
$\langle T \rangle$:= flottant	T.type = flottant
$\langle L \rangle$:= $\langle L_1 \rangle$ virgule id	L_1.h = L.h ajouterType(id, L.h)
$\langle L \rangle$:= id	ajouterType(id, L.h)



4.3 Attribut Mixte

- Présences d'attributs hérités et d'attributs synthétisés avec des dépendances croisées
- Est-il possible de décomposer en (Hérité + Synthétisé) et de faire plusieurs « passes » (de calcul) ?
- En mono-passe, construire un arbre de dépendances pour déterminer un ordre d'évaluation pour les instances d'attributs
  - Risque de circuit : s'il y a un circuit, le calcul n'est pas possible
  - D'une manière générale, soit une grammaire et des règles d'analyses sémantiques, très difficile de dire s'il existe des circuits

Grammaire	**Règles sémantiques**
$\langle A \rangle := \langle B \rangle$	$A.\mathrm{s} = B.\mathrm{h}$ $B.\mathrm{h} = A.\mathrm{s} + 1$



4.4 Algorithmie

4.4.1 École Classique
4.4.2 École Moderne

4.4.1 École Classique I

- Approche dite « Définition Dirigée par la Syntaxe » [1]
  - Compilateur Mono-passe
    - Analyse syntaxique LR $\Longrightarrow$ synthétisé = facile
    - Analyse syntaxique LL $\Longrightarrow$ hérité = facile
    - N.B. : tout utilisation doit être précédée d'une définition
  - Difficile de dire s'il y a des circuits dans le graphe d'évaluation
    $\Longrightarrow$ Concepts de grammaires dites « S-Attribuée » ou « L-Attribuée »
    - Ajouter des contraintes sur l'écriture des grammaires pour qu'elles soient « S-Attribuée » ou « L-Attribuée »
    - Pour rappel : toute traduction qui peut être faite de manière descendante peut être faite de manière ascendante
- Grammaire dite « S-Attribuée » : uniquement des Attributs Synthétisés
  - Ordre d'évaluation ascendant, p.ex. un parcours postfixe avec évaluation au nœud N quand on quitte N pour la dernière fois

4.4.1 École Classique II

- Grammaire dite « L-Attribuée » avec ordre d'évaluation descendant
  - Un attribut peut être Synthétisé
  - Un attribut peut aussi être Hérité :
    soit l'attribut X_i.a avec la règle $\langle A \rangle ::= \langle X_1 \rangle \ \langle X_2 \rangle \ \ldots \ \langle X_n \rangle$, n'utiliser que :
    - des attributs hérités associés en partie gauche de A
    - des attributs (hérités ou synthétisés) associés aux $X_1 \ \ldots \ X_{i-1}$
    - des attributs (hérités ou synthétisés) associés à X_i, mais sans circuit

    P.ex. :

Grammaire	**Règles sémantiques**
$\langle T \rangle := \ \langle F \rangle \ \langle T' \rangle$	$T'.\mathrm{h} = F.\mathrm{s}$
$\langle T' \rangle := \ * \ \langle F \rangle \ \langle T'_1 \rangle$	$T'_1.\mathrm{h} = T'.\mathrm{h} \times F.\mathrm{s}$

1. Approche intéressante si on souhaite développer un compilateur qui traduit « simplement » des expressions selon la syntaxe : p.ex. passer d'une sérialisation JSON à une sérialisation XML.

4.4.2 École Moderne

- Approche « Patron de conception Visiteur » de l'École Moderne
  - Compilateur Multi-passe
  - Patron de conception Visiteur : Cf. Cours pratique et Mémento JAVA
- Structure de données
  - Dictionnaires : K = identificateur du nœud de l'AST, V = attribut
- Algorithmie
  - Attribut Synthétisé : Ordre d'évaluation ascendant, p.ex. un parcours postfixe avec visite des enfants avant calcul de l'attribut
  - Attribut Hérité : Ordre d'évaluation descendant, p.ex. un parcours préfixe avec calcul de l'attribut avant visite des enfants
  - Attribut Mixte

5 Table des symboles

- **Liaison entre déclarations et utilisation des identificateurs**
  - Et mise en œuvre de la surcharge, de la redéfinition, etc.
- **Structure spécifique : « Table des symboles »**
  - Regroupe l'ensemble des déclarations d'identificateurs, leurs portées (et aussi leurs visibilités) depuis chaque nœud de l'arbre
  - Propagée dans la suite de la compilation (y compris l'exécutable)
- **Dans le cas d'un langage orienté objet**
  - La visibilité est aussi héritée en suivant l'arbre d'héritage des classes qui est indépendant de l'AST (information de contexte)
    - Algorithmie : le calcul et la manipulation de la portée d'une classe enfant peut être facilitée si elle est incluse (au sens arbre des portées) dans la portée de la classe parente