

Structure syntaxique

Structure

Langage : lexique+syntaxe+sémantique

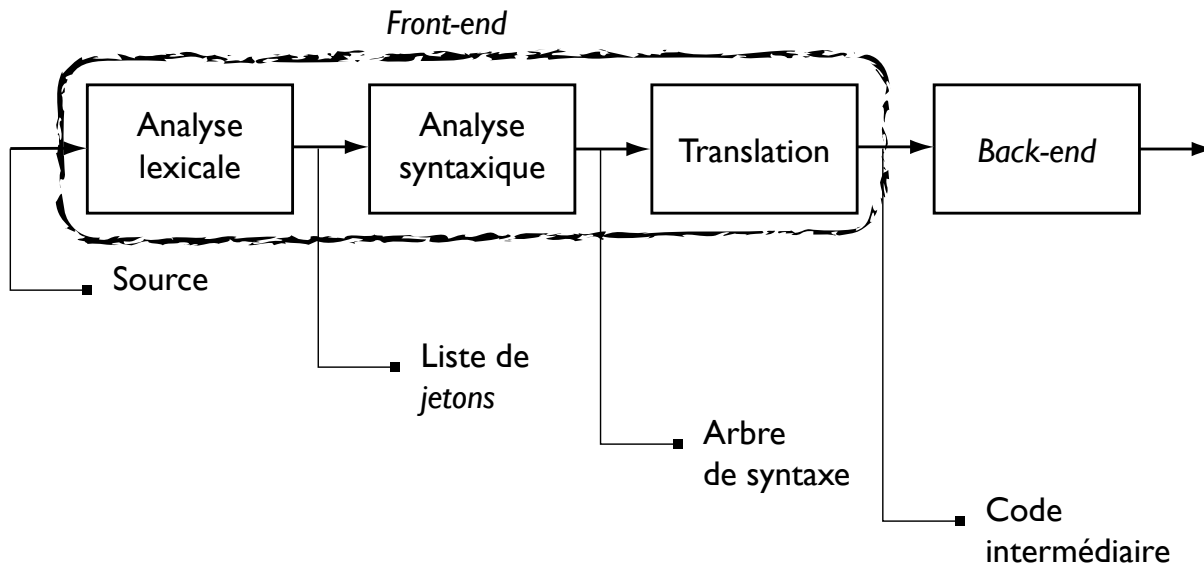
lexique : mots du langage

syntaxe : règles de combiner les mots (grammaire)

sémantique : tout ce qui n'est pas syntaxe mais est nécessaire pour phrases correctes (eg. ty-page)

(pragmatique : usage/goût/style)

Implantation



1. **Compilation** : back-end produit code assembleur puis code machine, code machine est lancée séparément

2. **Interprétation** : back-end exécute code intermédiaire

2 $\frac{1}{2}$. back-end produit code machine virtuelle (eg. Java)

Implantation II

- Plus efficace par compilation si programme exécuté plusieurs fois ou contient des boucles
- Interprète : plus prompt, compact, flexible et portable (eg, shells de commande : csh, perl)

Analyse lexicale

lit le programme, élimine les espaces et les remarques

jetons (*tokens*) : unités lexicales

- mots clés : `while`
- symboles d'opération : `+`
- littéraux/constantes : `2.34e-2`, `"Bonjour."`
- identificateurs : `x`, `Calcule_la_moyenne123`

Exemple :

```
if (x>3) {y=x+1;} else {y=x-1;};
```

`if` `(` `id` `>` `const` `)`...

jeton 1 : **if**, valeur `if`

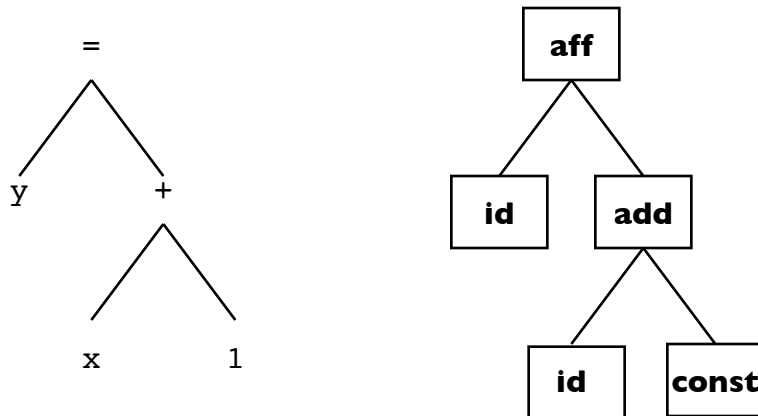
jeton 2 : `(`, valeur `(`

jeton 3 : **id**, valeur `x`

...

Analyse syntaxique

Arbre de syntaxe abstraite (ASA) : $y=x+1$



Traverser les noeuds :

infixe : $y = x + 1$ (FORTRAN, C, ...)

préfixe : $= y + x 1$ (Lisp, Scheme, ...)

postfixe : $y x 1 + =$ (Postscript, ...)

(trois syntaxes différentes)

ou même : $y \text{ soit } x 1 \text{ plus}$ (langage fictif)

Règles de priorité

notation infixe :

$a+b*c$: $a + (bc)$ ou $(a + b)c$?

$2-2-2$: $2 - (2 - 2)$ ou $(2 - 2) - 2$?

1. niveau de précedence : indique comment regrouper les sous-expressions (commence par plus haut niveau) :

typiquement, de haut en bas : $^$, $*/$, $+ -$

en C : 15 niveaux !

2. ordre d'associativité : indique comment regrouper les sous-expressions au même niveau

typiquement : $^$ — droite à gauche

$*/+ -$ — gauche à droite

3. parenthèses pour changer les règles 1 et 2

Définition de syntaxe

Définitions :

1. lexique/vocabulaire

(“alphabet” pour le théoricien) : ensemble des symboles dans le langage

$$\Sigma = \{0, 1\}$$

$$\Sigma = \{\text{if}, \text{ident}, \text{while}, \dots\}$$

2. phrase : séquence (possiblement vide) de symboles tirés de Σ

100100

3. langage : ensemble (possiblement infini) de phrases

Grammaire hors-contexte

(IFT2102 !)

Une GHC est définie par

1. Σ , ensemble fini de terminaux (jetons),
2. $\mathcal{V}$ ensemble de non-terminaux/variables (représentés par des catégories),
3. $\mathcal{R}$ ensemble de règles (productions),
4. S variable initiale (symbole de départ).

exemple simple : G_1

$$\Sigma = \{0, 1\}$$

$$\mathcal{V} = \{\langle \text{phrase} \rangle\}$$

$$\mathcal{R} = \left\{ \langle \text{phrase} \rangle ::= \varepsilon \mid 0 \langle \text{phrase} \rangle 1 \right\}$$

variable initiale : $\langle \text{phrase} \rangle$.

(ε : phrase vide, phrase de longueur 0)

Grammaire hors-contexte II

Dérivation : comment arriver à partir de la variable initiale à une phrase ?

Exemple G_1 :

$\langle \text{phrase} \rangle$

$\Rightarrow 0\langle \text{phrase} \rangle 1$

$\Rightarrow 00\langle \text{phrase} \rangle 11$

$\Rightarrow 0011.$

Langage défini par G : $L(G)$ — ensemble de toutes les phrases dérivables par G .

Grammaire hors-contexte III

Grammaire G_2 :

`<flottant> ::= <entier> . <entier>`

`<entier> ::= <chiffre>`

`<entier> ::= <chiffre> <entier>`

`<chiffre> ::= 0|1|2|3|4|5|6|7|8|9`

Dériver : 1.5, .5, 12.34