Compilation

IUT Informatique

Université Paris-Saclay

CHAPITRE 0 : INTRODUCTION

Langages….

Langages de programmation

Langages de description

Quels langages ?

- En informatique, la notion de langage est centrale. On distingue notamment
  - des **langages de programmation** (C, Caml, Java…)
  - des **langages de description** (HTML, XML, PDF…)
- Souvent, l'informaticien est amené à écrire des programmes pour **analyser, interpréter** des données écrites dans un langage donné, par exemple pour les traduire en un autre langage ou les mettre en entrée d'un logiciel existant.

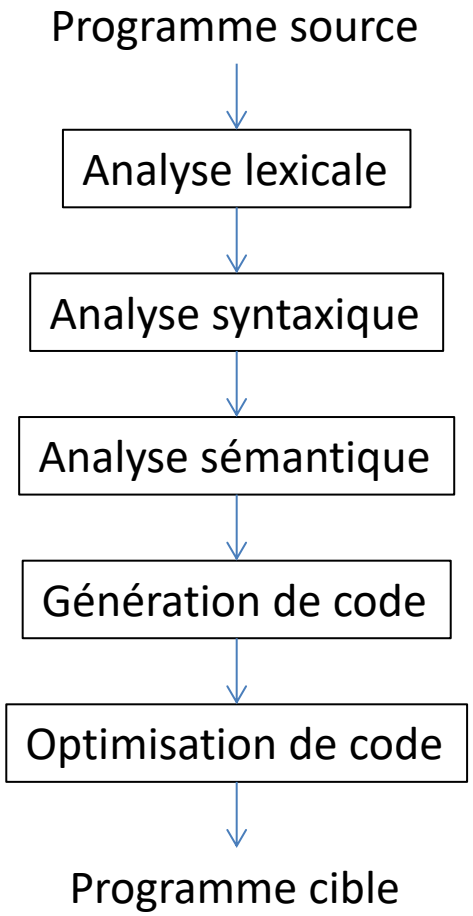
Un exemple emblématique : la compilation des programmes

- Un compilateur prend en entrée un programme écrit dans un langage de haut niveau et le traduit
  - soit en langage machine
  - Soit en langage intermédiaire (cas des langages « semi-compilés », comme Java)
- Nous allons étudier (en partie) le fonctionnement d'un compilateur.

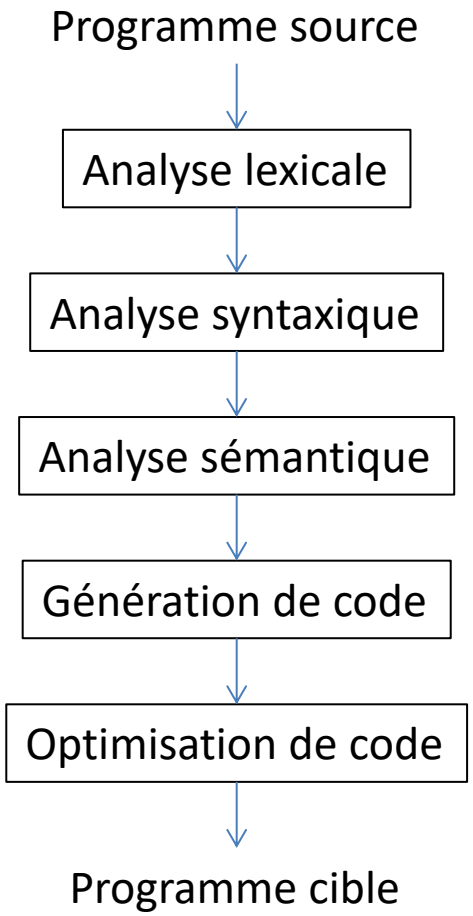
Ce que nous verrons

- Des concepts théoriques :
  - bases de **la théorie des langages formels**
  - un peu de **sémantique** des langages de programmation.
- Des outils pratiques :
  - Lex pour l'analyse lexicale
  - Yacc pour l'analyse syntaxique

Principales phases de la compilation d'un programme

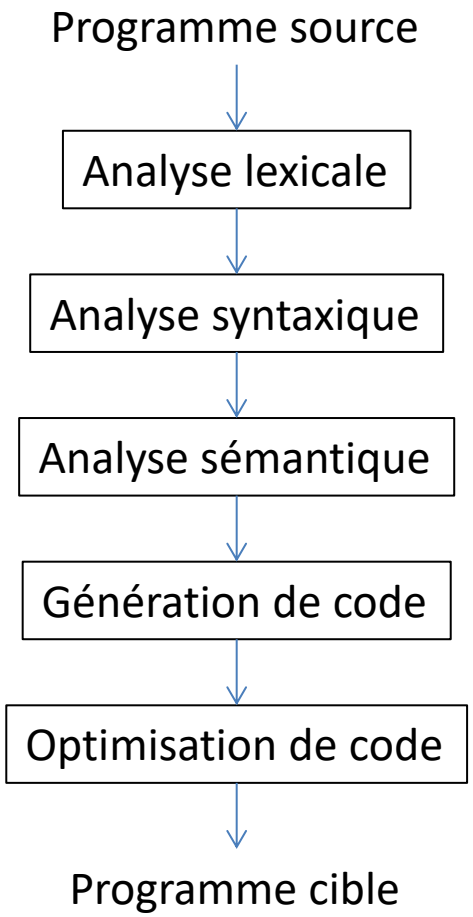


Principales phases de la compilation d'un programme



- Analyse lexicale :
  - Il s'agit de reconnaître le « statut » de chaque mot lu dans le programme.
  - L'analyse lexicale détermine la suite des **unités lexicales** (ou **tokens**) du programme.
  - L'outil `lex` permet de faire de l'analyse lexicale (mais pas seulement).

Principales phases de la compilation d'un programme



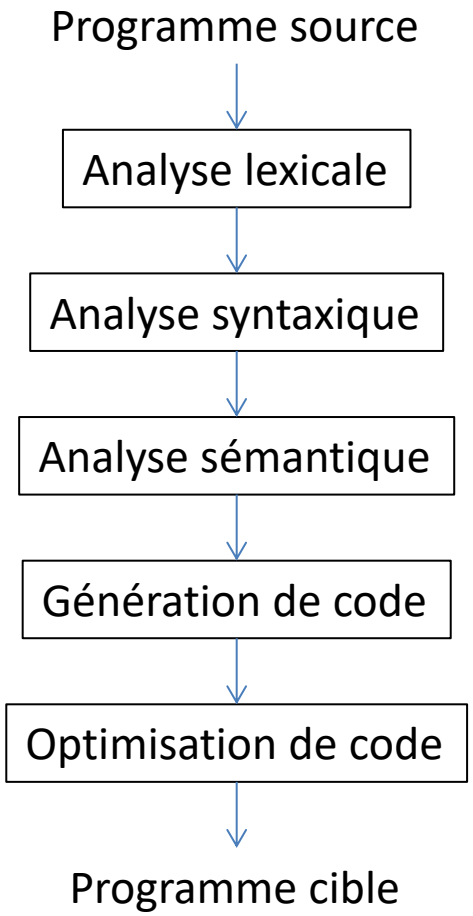
- Exemple :

```
if (i < a+b)    //test
   x = 2*x
```

donnera

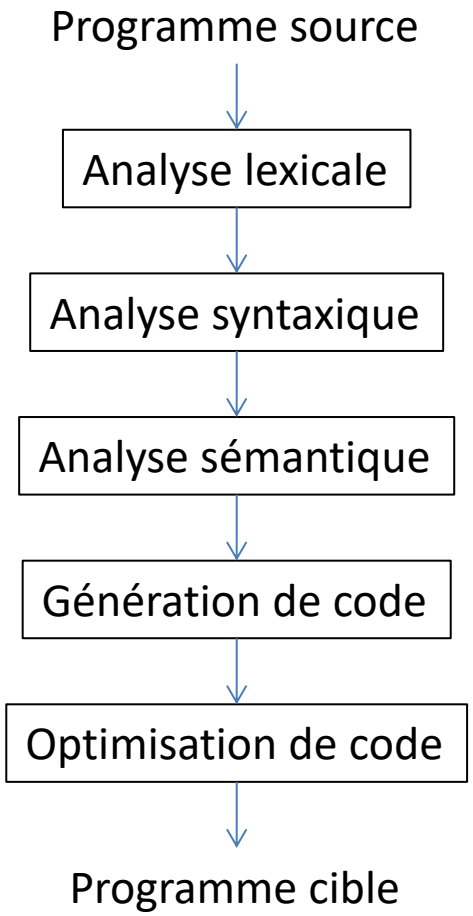
```
MC_IF PAR_O IDENT
OP_REL IDENT OP_ARITH
IDENT PAR_F IDENT
AFFECT CONST…
```

Principales phases de la compilation d'un programme



- Analyse syntaxique :
  - Il s'agit de vérifier que les unités lexicales sont dans le bon ordre.
  - Ce « bon ordre » est défini par la **grammaire** du langage.
  - Le résultat de l'analyse syntaxique (si elle se passe bien) est un **arbre syntaxique**.
  - L'outil `yacc` permet de faire de l'analyse syntaxique (mais pas seulement).

Principales phases de la compilation d'un programme

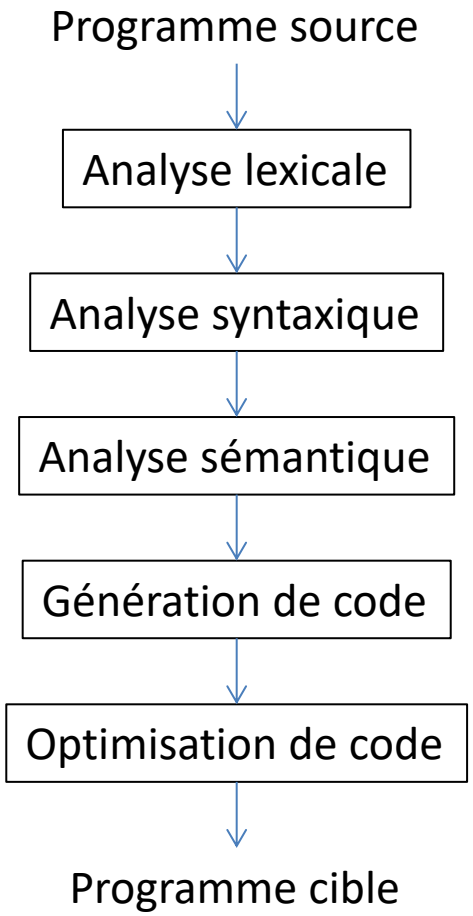


- Exemple :

```
MC_IF IDENT
```

aboutira à une erreur en C parce qu'on attend une parenthèse ouvrante après le mot-clé if.

Principales phases de la compilation d'un programme



- Analyse sémantique :
  - Vérifie si le programme (syntaxiquement correct) « a un sens ».
  - Notamment, on vérifie que les règles de typage sont vérifiées (ex : ne pas additionner un réel et une chaîne de caractères).

CHAPITRE 1 : NOTIONS DE BASE SUR LES LANGAGES (RAPPELS)

Avec des extraits d'un cours de Claude Marché

Généralités sur les langages

- Notions d'alphabet (fini), de mot, de langage
- Opérations sur les mots : concaténation, exponentiation. Le mot vide : epsilon
- Opérations sur les langages :
  - Opérations ensemblistes classiques : union, intersection, complémentaire.
  - Concaténation, exponentiation.
  - Etoile de Kleene.

Exercice 1

- Soit A = {a, b}
- Soient L_1={aa, ab}, L_2={a,aba}
- Que sont les langages suivants ?
  - L_1^2, $L_1.L_2$, L_2^2, $L_1^2 \cap L_2^2$?
  - $L_1 \cap \emptyset$, $L_1 \cup \emptyset$, $L_1.\emptyset$?
  - L_1^*

CHAPITRE 2: LANGAGES RATIONNELS (OU RECONNAISSABLES), EXPRESSIONS RATIONNELLES, EXPRESSIONS LEX.

Avec des extraits de cours de Claude Marché et de Paul Franchi

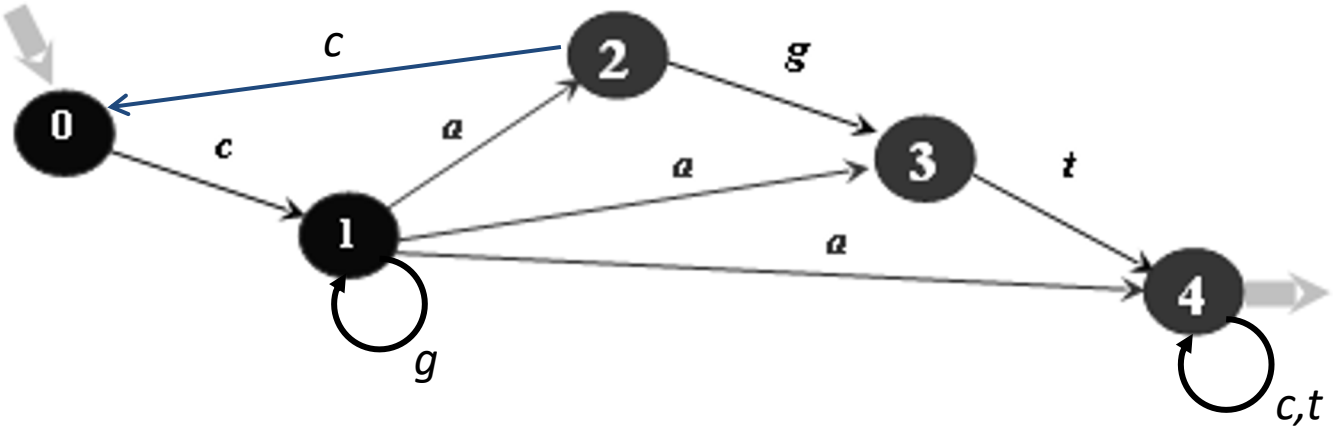
La classe des langages rationnels

- Définition : un langage est rationnel si et seulement si il existe une expression rationnelle qui le décrit.

- Définition : un langage est reconnaissable si et seulement si il existe un automate fini qui le reconnaît.

- Théorème de Kleene (1956) : La classe des langages rationnels et celle des langages reconnaissables n'en font qu'une.

Un automate fini



- A={a,c,g,t}
- Q={0,1,2,3,4}
- q_0={0}
- F={4}

δ

	a	c	g	t
0	-	1	-	-
1	2,3,4	-	1	-
2	-	0	3	-
3	-	-	-	4
4	-	4	-	4

Expressions rationnelles (ou régulières)

Soit A un alphabet fixé.

Définition 9 *Une* expression régulière *sur A est une notation qui décrit un langage sur A. Ces notations sont*

- ε *: décrit le langage* $\{\varepsilon\}$ *;*
- w *mot sur A : décrit le langage* $\{w\}$ *;*

et, si e, e_1 et e_2 sont déjà des expressions régulières désignant respectivement les langages L, L_1 et L_2 :

- $e_1 \mid e_2$ *: décrit le langage* $L_1 \cup L_2$ *;*
- $e_1 e_2$ *: décrit le langage* $L_1 \cdot L_2$ *;*
- $e\star$ *: décrit le langage* L^* *;*
- (e) *: décrit aussi le langage* L*.*

On dit aussi parfois *expression rationnelle* au lieu de expression règulière. Le parenthèsage sera utilisé pour éviter certaines ambiguïtés, comme $e_1 \mid e_2\star$: on écrira $(e_1 \mid e_2)\star$ ou bien $e_1 \mid (e_2\star)$ suivant ce que l'on souhaite décrire.

Expressions rationnelles (ou régulières)

Exemple : sur l'alphabet des caractères ASCII, l'expression

```
0|((1|2|3|4|5|6|7|8|9)
                ((0|1|2|3|4|5|6|7|8|9)*))
```

décrit le langage des constantes entières non signées en décimal.

sans zéro inutile

Exercice 2

- Donner une expression rationnelle et un automate pour chacun des langages suivants sur l'alphabet A={a,b,c}:
  - L'ensemble des mots qui commencent par « abc ».
  - L'ensemble des mots qui ne contiennent pas de « b ».
  - L'ensemble des mots qui commencent par « abc » ou par « ac ».
  - L'intersection des deux langages précédents.
  - L'ensemble des mots qui contiennent le facteur « abc ».

Expressions rationnelles étendues

Définition 11 *Les* expressions régulières étendues *sont obtenues en complétant les notations par*

- `e+` *: décrit le langage* $L \cdot L^* = \bigcup_{k>0} L^k$ *;*
- `e?` *: décrit le langage* $L \cup \{\varepsilon\}$ *;*
- `[`$a_1 \cdots a_n$`]` *où les* a_i *sont des symboles de* A *: décrit le langage des* n *mots à une lettre* $\{a_1, \ldots, a_n\}$ *;*
- `[^`$a_1 \cdots a_n$`]` *où les* a_i *sont des symboles de* A *: décrit le langage des mots à une lettre* $\{b \mid b \in A, b \neq a_i \text{ pour tout } i\}$.

De plus, dans ces deux dernières constructions on autorise la notation `a-b` *pour décrire l'intervalle des caractères entre* `a` *et* `b`*, ce qui a un sens quand* `a` *et* `b` *sont soit tous deux des chiffres, soit tous deux des lettres minuscules, soit tous deux des lettres majuscules.*

Exemple : sur l'alphabet des caractères ASCII, l'expression étendue `[0123456789]+` ou bien encore `[0-9]+` décrit le langage des constantes entières non signées en décimal, où l'on autorise les zéros superflus au début. L'expression étendue `"[^"]*"` décrit le langage des chaînes de caractères d'un langage de programmation (comme CAML, JAVA ou C).

Expressions rationnelles étendues

- Tout langage décrit par une expression rationnelle étendue peut être décrit aussi par une expression rationnelle « ordinaire ».

- Autrement dit : les expressions rationnelles étendues ne permettent pas de décrire davantage de langages ; elles offrent juste une plus grande souplesse d'utilisation.

Expressions régulières en lex

- Lex (dont on verra l'utilisation au prochain chapitre) utilise, comme d'autres outils ou langages, des expressions couramment appelées « expressions régulières »
- Toutefois certaines opérations vont au-delà des expressions rationnelles (classiques ou étendues)

Expressions régulières en lex

- Alphabets:
  - codes ISO, ASCII, etc

- Expressions régulières
  - forme de Kleene via des méta-opérateurs

    (concaténation, le • est omis)

    | *(alternative)*

    ✶ *(répétition)*

    ( )
  - exemple: les identificateurs C
    - *Id* = (a|b|..|z|A|..|Z|_) (a|b|..|z|A|..|Z|_|0|..|9)✶

Expressions régulières en lex

- Expressions régulières étendues
  - méta-opérateurs
    - [ ] - + ? • ^
    - " " \
  - exemples
    - *les entiers:* [0-9]*+*
    - *les identificateurs C:* [a-zA-Z_] [a-zA-Z0-9_]✶
    - *les chaînes de caractères sans " :* \" [^"]✶ \"
    - *les commentaires lignes Java:* "/ /"•✶

Expressions régulières en lex

- Les caractères terminaux
  - tous les caractères, sauf les spéciaux
  - l ’espace est significatif
  - les spéciaux doivent être protégés par ″ " ou \
- Les caractères spéciaux (méta-)

Les expressions autorisées

Opérations rationnelles	
e?	*0 ou 1 fois l'exp e*
*e**	*0 ou n fois l'exp e (n quelconque)*
e+	*1 ou n fois l'exp e (n quelconque)*
e f	*l'exp e f*
e | f	*l'exp e ou l'exp f*
e{n,m}	*l'exp e répétée entre n et m fois*
(e)	*l'exp e*
{D}	*l'exp obtenue par substitution de D (macro)*
Sensibilité au contexte	
e / f	*l'exp e si suivie de l'exp f*
^e	*exp e en début de ligne*
e$	*exp e en fin de ligne*
<E>e	*L'exp e si dans l'état E*
Caractères	
.	*tout caractère sauf \n*
\c	*le caractère c , même spécial*
"abc"	*la chaine de caractères abc*
[abc]	*le caractère a ou b ou c*
[^abc]	*un des caractères sauf a, b, c*
[a-c]	*le caractère a ou b ou c*

Solutions de l'exercice 1

L_1={aa, ab}, L_2={a,aba}

L_1^2 = { aaaa, aaab, abaa, abab }

$L_1.L_2$ = { aaa, aaaba, aba, ababa }

L_2^2 = { aa, aaba, abaa, abaaba }

$L_1^2 \cap L_2^2$ = { abaa }

$L_1 \cap \emptyset = \emptyset$

$L_1 \cup \emptyset = L_1$

$L_1.\emptyset = \emptyset$

L_1^* = { ε, aa, ab, aaaa, aaab, abaa, abab, aaaaaa, … }

= l'ensemble des mots sur l'alphabet {a,b} qui sont de longueur paire et dont tous les b (s'il y en a) sont à des positions paires.

Solutions de l'exercice 2

- L'ensemble des mots qui commencent par « abc ».

  abc(a|b|c)*

- L'ensemble des mots qui ne contiennent pas de « b ».

  (a|c)*

- L'ensemble des mots qui commencent par « abc » ou par « ac ».

  (abc|ac)(a|b|c)*

- L'intersection des deux langages précédents.

  ac(a|c)*