Introduction: Fondements de la compilation

Martin Odersky

23 octobre 2006
version 1.2

Plan du cours

Pourquoi étudier la construction de compilateurs ?

Très peu de gens écrivent des compilateurs comme profession.

Alors pourquoi apprendre à construire des compilateurs ?

Ça vous rendra plus compétent :

- Un informaticien compétent comprend les langages de haut niveau ainsi que le matériel.
- Un compilateur relie les deux.
- Comprendre les techniques de compilation est essentiel pour comprendre comment les langages de programmation et les ordinateurs interagissent.

On compile ailleurs que dans les compilateurs :

- Beaucoup d'applications contiennent de petits langages pour leur configuration et pour flexibiliser leur contrôle :
  - les macros de Word, les scripts pour le graphisme et l'animation, les descriptions de structures de données.
- Les techniques de compilation sont nécessaires pour correctement implanter des langages d'extension.

- Les formats de données sont aussi des langages formels. De plus en plus de données en format interchangeable ressemblent à un texte d'un langage formel (p.ex. HTML, XML).
- Les techniques de compilation sont utiles pour lire, manipuler et écrire des données.

Ce sont des systèmes techniquement intéressants :

- Les compilateurs sont d'excellents exemples de grands systèmes complexes qui peuvent être :
  - spécifiés rigoureusement,
  - réalisés seulement en combinant théorie et pratique.

Le rôle d'un compilateur

Le rôle principal d'un compilateur est de traduire des programmes écrits dans un langage source donné en un langage objet.

- Souvent, le langage source est un langage de programmation et le langage objet est un langage machine.
- Quelques exceptions : traduction source-source, traduction de code machine, manipulation de données en XML.

Une partie du travail du compilateur est aussi de détecter si un programme donné est conforme aux règles du langage source.

Spécifier un compilateur

Une spécification d'un compilateur est constituée par

- une spécification de son langage source et de son langage objet,
- une spécification du processus de traduction de l'un en l'autre.

Langages

Formellement, un langage est un ensemble de chaînes de caractères que l'on appelle les phrases du langage.

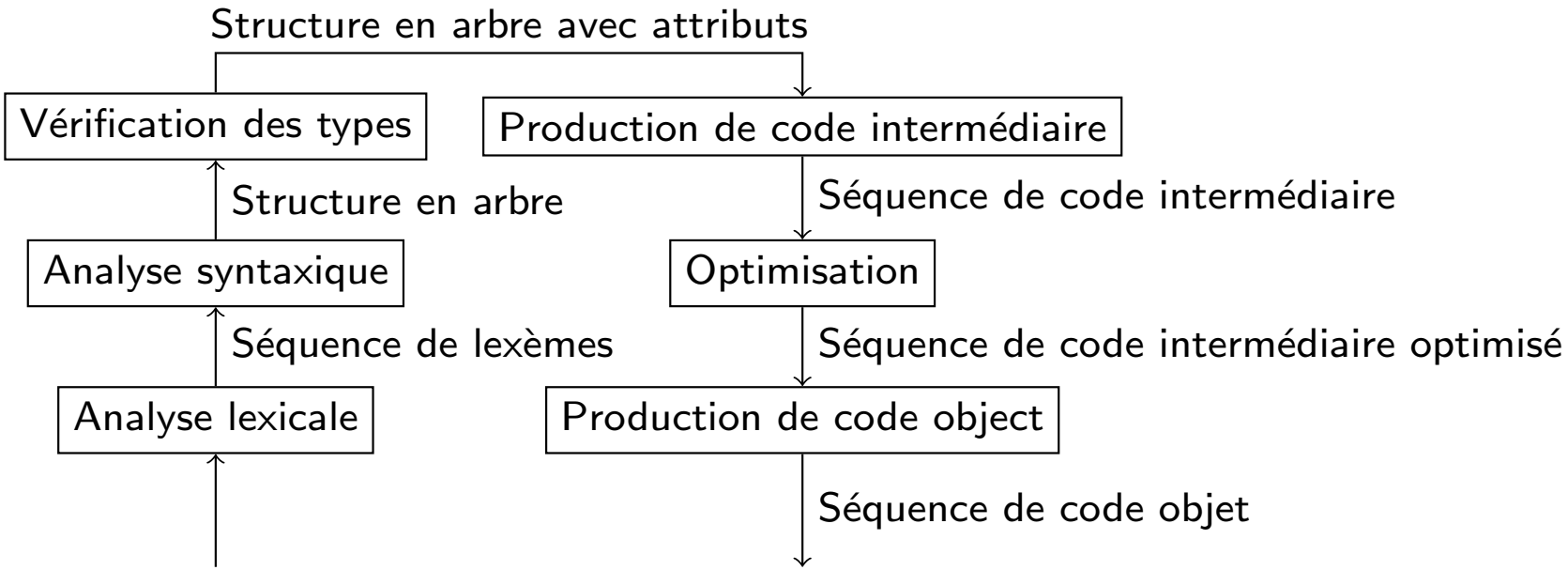
En pratique,

- chaque phrase possède une structure qui peut être décrite par un arbre,
- les règles régissant la structure des phrases sont définies par une grammaire.

Exemple : langages des programmation

- Les phrases d'un langage de programmation sont des programmes (légaux).
- Un programme est une phrase constituée de mots (appelés aussi symboles ou lexèmes) ; sa structure est donnée par une grammaire.
- Un mot est lui-même une séquence de caractères dont la structure peut aussi être donnée par une grammaire.

Structure d'un compilateur



- Les phases ne sont pas nécessairement exécutées l'une après l'autre.
- Les structures de données intermédiaires n'existent parfois jamais dans leur intégralité.

Langage et syntaxe

Les phrases d'un langage ont une structure déterminée par une grammaire :

Exemple

« Une phrase correcte est formée par un sujet suivi d'un verbe.»

- Ceci peut être exprimé par la grammaire :
  - `Phrase = Sujet Verbe.`
- Complétons cela avec deux nouvelles productions :
  - ```
    Sujet = "Pierre" | "Claire".
    Verbe = "cours" | "marche".
    ```
- Ceci définit 4 phrases possibles :
  - ```
    Pierre cours | Pierre marche |
    Claire cours | Claire marche
    ```

Langage et syntaxe — langages infinis

Généralement, les langages contiennent un nombre infini de phrases.

- Un nombre infini de phrases peuvent être exprimées par un nombre fini de productions en définissant certains symboles récursivement.

Exemple

```
  Nombre   =  Chiffre | Chiffre Nombre.
 Chiffre   =  "0" | "1" | "2" | "3" | "4" | ... | "9".
```

va générer : 0, 12, 123, 1024, etc.

Langages formels

Une grammaire G est définie formellement par le n-uplet $\langle \Sigma, N, P, S \rangle$ où :

- Σ est l'ensemble des symboles terminaux ;
- N est l'ensemble des symboles non-terminaux ;
- P est l'ensemble des règles syntaxiques (ou productions) ;
- et S est le symbole initial.

Une grammaire définit un langage formé par l'ensemble des séquences finies de symboles terminaux qui peuvent être dérivées du symbole initial par des applications successives des productions.

Le langage des grammaires (non-contextuelles)

BNF

```
   grammar  =  production grammar | (empty).
production  =  ident "=" expression ".".
expression  =  term | expression "|" term.
      term  =  factor | term factor | "(empty)".
    factor  =  ident | string.
     ident  =  letter | ident letter | ident digit.
    string  =  "\"" stringchars "\"".
stringchars =  stringchars stringchar | (empty).
stringchar  =  escapechar | plainchar.
escapechar  =  "\\" char.
 plainchar  =  charNQNE.
      char  =  tout caractère imprimable
   charNQNE  =  tout caractère imprimable sauf «"» et «\»
```

Le langage des grammaires — notation Backus-Naur

Il a été développé à l'origine par J. Backus et P. Naur pour la définition du langage Algol 60.

C'est pourquoi on l'appelle la notation Backus-Naur ou BNF (de l'anglais *Backus-Naur form*).

Exercice

Déterminez le symbole initial, et les symboles terminaux et non-terminaux de cette grammaire.

Notation Backus-Naur étendue

Les grammaires peuvent souvent être simplifiées et raccourcies en utilisant deux constructions supplémentaires :

- `{x}` exprime la répétition : zéro, une ou plusieurs occurrences de x.
- `[x]` exprime l'option : zéro ou une occurrence de x.

Ce nouveau formalisme est appelé notation Backus-Naur étendue ou EBNF (de l'anglais *extended Backus-Naur form*).

Syntaxe EBNF

EBNF

```
    syntax  =  {production}.
production  =  identifier "=" expression ".".
expression  =  term {"|" term}.
      term  =  {factor}.
    factor  =  identifier
           |   string
           |   "(" expression ")"
           |   "[" expression "]"
           |   "{" expression "}".
identifier  =  letter {letter | digit}.
    string  =  "\"" {stringchar} "\"".
```

Identique à BNF pour le reste . . .

Exercice

Ecrivez la grammaire des nombres entiers (éventuellement signés) en BNF puis EBNF.