Théorie des Langages - EISTI - ING 1

Yannick Le Nir

Ecole Internationale des Sciences du Traitement de l'Information

yannick.lenir@eisti.fr

Introduction

Historique

- Tentative de modélisation des langues naturelles (optimisme IA)
- Relatif échec (fin de l'optimisme)
- Utilisation pour la description des langages de programmation
- ALGOL (langage de programmation hors contexte)
- "French School" : Marcel Schützenberger et Maurice Nivat
- Analogie langage de programmation, langage naturel (langages, alphabet, mot, grammaire)

Introduction

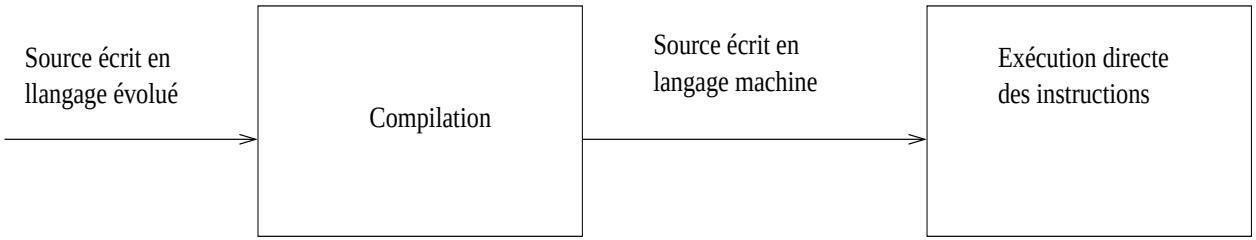
Description de la théorie

- Définir les concepts et théorèmes permettant d'établir qu'un source est bien écrit dans un langage donné :
  - Concepts de langage et de grammaire
  - Classification des langages et grammaires
  - Propriétés des différentes classes de langages
  - Concepts algorithmiques permettant de chercher et/ou valider les mots et phrases d'un langage (automates)
- Application aux langages de programmation
  - Analyse lexicale
  - Analyse syntaxique
  - Analyse sémantique

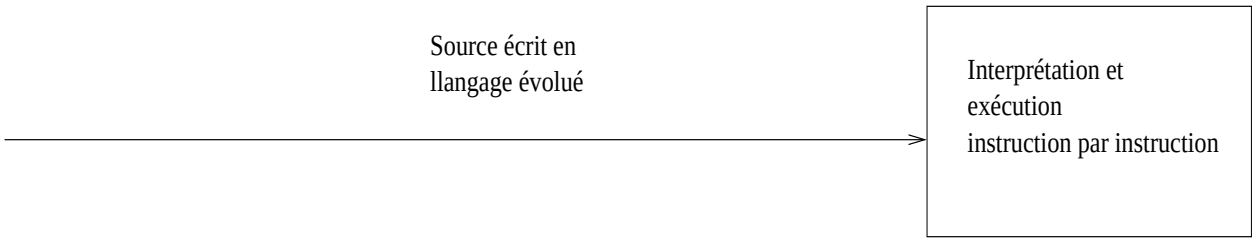
Analyse de sources

Description

- Exécution de programmes en langage machine
- Langages plus évolués utilisés par les programmeurs
- Processus de transformation :
  1. Compilation puis exécution



  2. Interprétation lors de l'exécution



Analyse des instructions

3 étapes principales

1. Analyse lexicale :
   Décomposition de l'instruction en mots du langage de programmation (lexème ou token). Attribution de catégories syntaxiques aux mots.
2. Analyse syntaxique :
   Vérification de la validité de la succession des catégories syntaxiques (exemple : Déterminant Nom Verbe Déterminant Nom)
3. Analyse sémantique :
   Signification d'une phrase valide, sens des lexèmes et de leur association (exemple `"A:=B  AND C"` n'a de sens que si A,B et C sont de type BOOLEAN)

Concepts de langage

Définitions

- Enumération des mots sur un alphabet (sans intérêt)
- Règles de productions pour définir les mots valides de ce langage. Notion de grammaire et de langage décidable.

Théorie

- Définition de grammaires engendrant des langages
- Classification de ces grammaires
- Propriétes des différentes classes de grammaires
- Etude des automates servant de support algorithmique aux grammaires
- Problèmes d'indécidabilité de certains langages

Définitions

1. Alphabet et vocabulaire
   Un alphabet A est un ensemble de symboles. Si les symboles sont déjà des suites de caractères, on peut alors parler aussi de vocabulaire
2. Mot et vocabulaire
   Un mot associé à un alphabet A est une suite éventuellement vide de symboles de cet alphabet. Le mot vide sera noté ε. Dans le cas où A est un vocabulaire, on ne parlera pas de mots mais de phrase.

Définitions

Définitions(suite)

1. Longueur d'un mot
   Soit A un alphabet et w un mot associé à cet alphabet : on définit la longueur de w comme le nombre de symboles contenus dans w. On note cette longueur $|w|$.
2. Ensembles A^+ et A^*
   Soit A un alphabet : on définit A^+ comme l'ensemble des mots associés à A tels que $|w| > 0$ et A^* comme l'ensemble des mots associés à A tels que $|w| \geq 0$
3. Langage
   Soit A un alphabet : un langage est un sous ensemble de A^* .

Concept de Grammaire

Présentation

Une grammaire permet de générer les mots d'un langage à l'aide de règles de production

Définition

Une grammaire est un quadruplet `G = <T, N, S, P>` tel que

- T est un ensemble dit de terminaux
- N est un ensemble dit de non terminaux,
- S élément de N dit axiome de la grammaire
- P est un ensemble de règles de production.
  Une règle de production est de la forme :
  $X \rightarrow Y$ où $X \in (T \cup N)^+$ et $Y \in (T \cup N)^*$

Langage déduit de la grammaire

Construction

Combinaison successives des règles dites de production

- L'ensemble T représente l'alphabet de notre langage.
- Les éléments de N sont appelés non terminaux car ils représentent des mots intermédiaires qui ne font pas partie des mots du langage mais qui servent à leur construction par combinaisons successives des règles de production.
- Dans chaque combinaison de règles pour former un mot, S représente le membre gauche de la première règle de la combinaison.

Exemple de grammaire

Grammaire

G = <T, N, S, P>

- T = {identifiant, opérateur, nombre}
- N = {opérande, expression}
- S = expression
- P = { opérande -> identifiant | nombre,
  expression -> opérande
  | expression opérateur expression }

Langage

La suite "identifiant opérateur nombre" est un mot (plutôt une phrase du langage) défini par cette grammaire car :

1. "expression" est l'axiome de la grammaire
2. on part de la règle "expression $\longrightarrow$ expression opérateur expression"
3. on applique la règle "expression $\longrightarrow$ opérande" et on en déduit : "expression opérateur expression $\longrightarrow$ opérande opérateur expression"
4. on applique la règle "expression $\longrightarrow$ nombre" et on en déduit : "opérande opérateur expression $\longrightarrow$ opérande opérateur opérande"

Génération de mots par dérivation

Règle terminale

Soit `G = <T, N, S, P>` une grammaire et $R \in P$. R est une règle terminale si son membre à droite est un élément de T^*.

Dérivation directe

Soit `G = <T, N, S, P>` une grammaire. $u \in (T \cup N)^*$ se dérive directement en $v \in (T \cup N)^*$ selon G, si et seulement si $u = u1Mu2$, $v = u1Nu2$ et $M \longrightarrow N$ est une règle de production. On notera la dérivation directe de u en v par l'expression $u \longrightarrow v$.

Dérivation

Soit `G = <T, N, S, P>` une grammaire. $u \in (T \cup N)^*$ se dérive en $v \in (T \cup N)^*$ selon G, si et seulement
$\exists k \in N\ k > 0$ et $\exists u = u1, u2, ..., u_k = v$ et
$\forall i < k\ u_i \longrightarrow u_{i+1}$.

Langage engendré par une grammaire

Définition

Soit `G = <T, N, S, P>` une grammaire. On appelle langage engendré par G en partant de S, le langage défini par $L_G(S) = \{u \in T^* / S \longrightarrow u\}$

Langage de notre grammaire

$L_G(S) = \{u \in T^* \ / \ u = (identifiant|nombre)(operateur(identifiant|nombre)^*\}$

Exemple de Grammaire

Exemple pour affectation numérique

Une grammaire `G = <T, N, S, P>` peut être définie par :

- $T = lettre \cup chiffre \cup \{+, -, *, /, =, .\}$
- $N = \{mot, nombre, operateur, nombre, identifiant\}$
- $S = mot$
- $P =$

$$\left\{ \begin{array}{l} mot \longrightarrow operateur, \\ operateur \longrightarrow +|-|*|/|=, \\ mot \longrightarrow nombre, \\ nombre \longrightarrow (chiffre^+)|(chiffre^+).(chiffre^+), \\ mot \longrightarrow identifiant, \\ identifiant \longrightarrow lettre\ |identifiant(lettre\ |chiffre) \end{array} \right\}$$

Dans cet exemple $L_G(S) = \{+, -, *, /, =$
$, chiffre^+, chiffre^+.chiffre^+, lettre(lettre|chiffre)^*\}$

Génération de mots par arbre syntaxique

Définition

Une dérivation de règles de production pour générer un mot peut être représentée par un arbre :

- La racine contient l'axiome de la grammaire
- Chaque noeud qui n'est pas une feuille est le membre gauche d'une règle de production, ses fils sont les différents membres droits de cette même règle et ceux ci apparaissent dans le même ordre que dans la règle
- Chaque noeud feuille est un élément terminal de la grammaire.

Génération de mots par arbre syntaxique

Théorème

Un mot d'un langage est un arbre syntaxique.
Soit `G = <T, N, S, P>` une grammaire et Ar un arbre syntaxique déduit de cette grammaire. Le mot obtenu en prenant de gauche à droite les différentes feuilles de cet arbre Ar est un mot du langage associé à la grammaire G.

La classification de Chomsky

Langages décidables et hiérarchie de classes

- Les langages de type 3 : rationnels ou réguliers
- Les langages de type 2 : algébriques ou hors contexte
- Les langages de type 1 : sensibles au contexte
- Les langages de type 0 : tous les autres décidables

Grammaire de type 3

Définition

Une grammaire $G =< T, N, S, P >$ est de type 3 si les règles de production sont de la forme :
$A \longrightarrow a$ où $A \in N$ et $a \in T^*$ $A \longrightarrow B\ a$ (ou $a\ B$) où $A, B \in N$ et $a \in T^*$

Langage associé

Un langage est de type 3 s'il peut être engendré par une grammaire de type 3.

Utilisation des grammaires de type 3

Domaines

- Occurrence de motifs dans une chaîne (Recherche d'informations)
- Expressions régulières (Shell, C, emacs)
- Séquence de l'ADN (Génôme)
- Apprentissage de grammaires pour l'IA

Exemple de grammaire

Affectation numérique

La grammaire `G = <T, N, S, P>` du cours précédent pour l'analyse lexicale de l'affectation numérique est de type 3 :

- $T = lettre \cup chiffre \cup \{+, -, *, /, =, .\}$
- $N = \{mot, nombre, operateur, nombre, identifiant\}$
- $S = mot$
- $P =$

$$\left\{ \begin{array}{l} mot \longrightarrow operateur, \\ operateur \longrightarrow +|-|*|/|=, \\ mot \longrightarrow nombre, \\ nombre \longrightarrow (chiffre^{+})|(chiffre^{+}).(chiffre^{+}), \\ mot \longrightarrow identifiant, \\ identifiant \longrightarrow lettre\ |identifiant(lettre\ |chiffre) \end{array} \right\}$$

Grammaire de type 2

Définition

Une grammaire $G =< T, N, S, P >$ est de type 2 si les règles de production sont de la forme :
$A \rightarrow \alpha$ où $A \in \mathcal{N}$ et $\alpha \in (N \cup T)^*$

Langage associé

Un langage est de type 2 s'il peut être engendré par une grammaire de type 2.

Grammaires de type 2

Exemple de grammaire

Le fameux langage $a^n b^n$ peut être engendré par la grammaire hors contexte suivante :

- $T = \{a, b\}$
- $N = \{S\}$
- $S = S$
- $P = \left\{ \begin{array}{l} S \rightarrow a\ S\ b, \\ S \rightarrow a\ b \end{array} \right\}$

Domaines d'application

- Langages de programmation
- La plupart des constructions des langues naturelles

Autres grammaires

Grammaires de type 1

Une grammaire $G =< T, N, S, P >$ est de type 1 si les règles de production sont de la forme :
$u\ A\ v \rightarrow u\ w\ v$ où $A \in N$, $u, v \in T^*$ et $w \in (N \cup T)^*$

Grammaires de type 0

Une grammaire $G =< T, N, S, P >$ est de type 0 si les règles de production sont quelconques.

Théorèmes

- Une grammaire n'engendre qu'un seul langage. La réciproque est fausse.
- Les différentes familles de langages sont incluses les unes dans les autres : $L_3 \subset L_2 \subset L_1 \subset L_0$

Forme normale de Chomsky

Définition

Une grammaire de type 2 (sans ε) est dite sous forme normale de Chomsky si et seulement si toutes les règles sont de la forme :

$$A \longrightarrow a \text{ (où } a \in T)$$
$$A \longrightarrow BC \text{ (où } B, C \in N)$$

Théorème

Tout langage hors-contexte sans ε peut être engendré par une grammaire en forme normale de Chomsky.

Transformation en forme normale de Chomsky

Algorithme

1. remplacer tous les terminaux x en partie droite des règles par des non-terminaux X en ajoutant les règles $X \longrightarrow x$
2. Toute règle $X \longrightarrow YZW$ est remplacée par $X \longrightarrow YV$ et $V \longrightarrow ZW$
3. remplacer les règles $X \longrightarrow Y$ par $X \longrightarrow WZ$ si $Y \longrightarrow WZ$

Exemple

Le langage $a^n b^n$ peut être engendré par la grammaire sous forme normale de Chomsky
suivante :$< \{a, b\}, \{A, B, S\}, S, P >$ avec

$$P = \left\{ \begin{array}{l} S \longrightarrow A\ B, S \longrightarrow A\ X, X \longrightarrow S\ B, \\ A \longrightarrow a, \\ B \longrightarrow b \end{array} \right\}$$

Analyse CKY

Forme normale de Chomsky

- La Forme Normale de Chomsky permet l'obtention d'un algorithme efficace pour reconnaître l'appartenance à un langage hors-contexte.
- Cet algorithme est du à Cocke, Kasami et Younger (CKY).

Idée de l'algorithme

- G : grammaire sous forme normale de Chomsky
- w : mot de longueur n
- Question : $w \in L(G)$?
- Chercher pour chaque sous-mot m de w, $X \in N$ tels que $X \rightarrow^* m$
- A la sortie, vérifier si le symbole axiome S fait partie des non-terminaux obtenus.

Analyse CKY

Construction

- Tableau triangulaire : colonnes numérotées par $i = 1, 2, ..., n$ (les positions de début de mot dans w) et lignes par $j = 1, 2, ..., n$ (les longueurs possibles).
- Remplissage de la ligne 1 puis 2 ...
- La $p$ième case de la ligne k correspond aux non-terminaux qui peuvent engendrer $w[p, k]$.
- On examine donc toutes les coupures du mot $w[p, k]$ en deux sous-mots : $w[p, l]$ et $w[p + l, k - l]$.
- Pour tous les Y possibles et tous les l compris entre 1 et k, on regarde toutes les expressions $X_p\ X_{p+l}$ en recherchant s'il existe Y dans la grammaire tel que $Y \rightarrow X_p X_{p+1}$.

Analyse CKY

Algorithme

```
Pour i = 1 à n faire :
  V[i, 1] <-- {A; A -> a est une règle
                et le ième symbole de w est a}
Pour j = 2 à n faire :
  pour i = 1 à n - j + 1 faire
   V[i, j] <-- ∅ ;
   pour k = 1 à j - 1 faire
    V[i, j] <--- V[i, j] union
    {A ; A -> BC est une règle, B appartient à V[i, k]
                          et C appartient à V[i+k, j-k]}
```

Analyse CKY

Complexité

- n^2 fois une action consistant à couper en deux un mot d'au plus n lettres
- n coupures à effectuer à chaque fois
- Complexité de l'algorithme de l'ordre de n^3 par rapport à la longueur n du mot d'entrée.
- Les langages hors-contexte peuvent être reconnus en un temps polynomial.