Théorie des Langages
Introduction

Yannick Le Nir Gaspard Férey

Contributions de :
Taisa Guidini Goncalves

CY TECH

yannick.lenir@eisti.fr gaspard.ferey@inria.fr

Introduction

Introduction

Langages et grammaires

Hiérarchie de Grammaires

Introduction

Historique

- La Théorie des Langages (TdL) est issue de la linguistique
- La TdL étudie les aspects syntaxiques des langages
- Objectif : Définir un langage formel
- Analogie : langage de programmation, langage naturel (langages, alphabet, mot, grammaire)

Yannick Le Nir,
Gaspard Férey

Contributions de :
Taisa Guidini Goncalves

Introduction
Introduction
Langages et grammaires
Hiérarchie de Grammaires

Introduction

Historique

- Tentative de modélisation des langues naturelles (optimisme IA) ... Relatif échec (fin de l'optimisme)
- Utilisation pour la description des langages de programmation
  - Lambda-Calcul (1930) : "un premier" langage de programmation (plutôt modélisation/algorithme)
  - Plankalk¨ul (1940) : "le premier" langage de programmation, pas connu
  - Fortran (début années 1950/IBM) : "le premier" langage de programmation de haut niveau, calcule numérique
  - Algol (fin années 1950/UNESCO) : langage de programmation "universel" (langage hors contexte)

Introduction

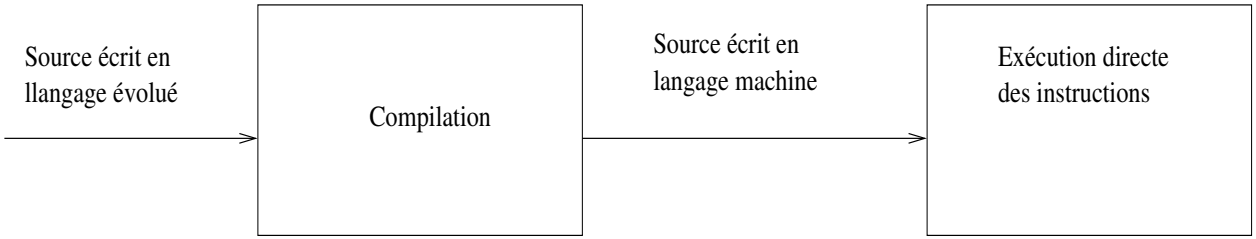
Description de la théorie

- 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

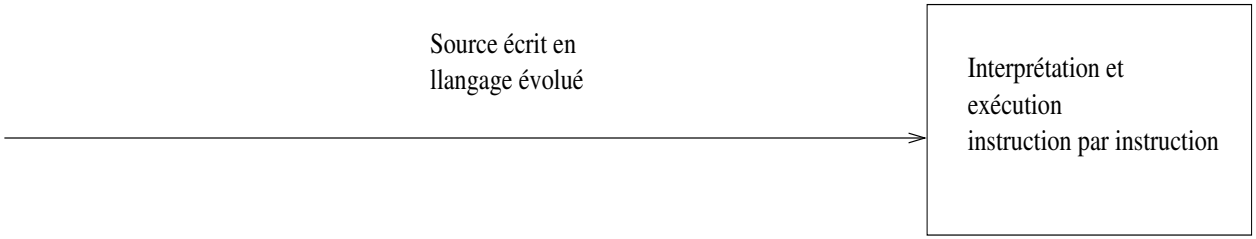
Analyse de sources

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
- 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

Définitions

- Alphabet ou vocabulaire
  Un alphabet A d'un langage formel est un ensemble fini non vide de symboles, lettres ou lexèmes. Si les symboles sont déjà des suites de caractères, on peut alors parler aussi de vocabulaire.
- Mots ou phrases
  Un mot associé à un alphabet A est une suite finie, éventuellement vide, de symboles de cet alphabet.
  Dans le cas où A est un vocabulaire, on ne parlera pas de **mots** mais de **phrase**.
- Mot vide
  On note ε ou parfois 1 (ou encore Λ) le **mot vide** : suite vide de symboles.
- Langage
  Un langage formel est un ensemble de **mots** de A.

Opérations sur les mots

Soit A un alphabet.

Définitions

- Longueur d'un mot
  On définit la longueur d'un mot w comme le nombre de symboles contenus dans w.
  On note cette longueur $|w|$ et on a $|w| \geq 0$.
  Note : seul ε, le mot vide, a pour longueur 0.
- Concaténation de deux mots
  Soient u et v deux mots de A.
  On définit la concaténation de u et v par la suite formée de u et v mis bout à bout.
  On la note uv, $u.v$, $u \cdot v$ ou encore $u\ v$.

Propriétés des mots

Soit A un alphabet. Soient u, v et w trois mots sur A.

Propriétés

- Associativité : $(uv)w = u(vw)$
  On note donc souvent ce mot uvw.
- Neutralité de ε : $\varepsilon u = u\varepsilon = u$
- Attention : $uv \neq vu$ en général.

- Longueur du mot vide : $|\varepsilon| = 0$
- Linéarité de la taille : $|uv| = |u| + |v|$
- Mots unitaires : $\forall a \in A, |a| = 1$.

Langages

Soit A un alphabet.

- Langage A^*
  On définit A^* comme l'ensemble des mots associés à A. **Mot vide inclus**. Tout langage sur A est un sous-ensemble de A^*.
- Langage A^+
  On définit A^+ comme l'ensemble des mots w de A^* tels que $|w| > 0$. C'est le langage des mots non vides.
- Langage A^n
  Pour $n \in \mathbb{N}$, on définit $A^n = \{w \in A^* | \ |w| = n\}$.
  C'est le langage des mots de taille n.
- Langage A
  On note A le langage des mots constitué d'un unique symbole de A.

Opérations sur les langages

Soit A un alphabet. Soient K et L deux langages sur A.

- Opérations ensemblistes
  $K \cup L$, $K \cap L$ et $\overline{K}$ sont des langages.
- Langage concaténé
  $K \cdot L = \{u \cdot v \mid u \in K \text{ et } v \in L\}$ est un langage.

Propriétés

- $A = \{\varnothing\}$ le langage vide, l'ensemble vide
- $A^0 = \{\varepsilon\}$ le langage contenant le seul mot vide
- $A^1 = A$
- $A^{n+1} = A \cdot A^n = A^n \cdot A$
-

$$A^+ = \bigcup_{n>0} A^n \qquad A^* = \bigcup_{n\geq 0} A^n$$

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 fini terminal.
- N est un ensemble fini non terminal.
- S $\in$ N est le symbole de départ ou 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 sur lequel est défini le 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. Catégorie syntaxique.
- 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 = {expression, opérande}
- S = expression
- P = {opérande -> identifiant | nombre,
  expression -> opérande
  | expression opérateur expression}

Notation

La notation | est utilisée pour simplifier l'écriture des règles.

A -> B | C est juste un raccourci syntaxique pour

A -> B et A -> C

Le symbole -> est une abréviation de "peut être composé de".

Langage

Langage

La suite "identifiant opérateur nombre" est un mot (plutôt une phrase du langage) défini par cette grammaire car on peut la produire à partir de l'axiome :

1. expression (Axiome S)
2. expression opérateur expression (par la dernière règle)
3. opérande opérateur expression (par la règle 3)
4. opérande opérateur opérande (par la règle 3)
5. opérande opérateur nombre (par la règle 2)
6. identifiant opérateur nombre (par la règle 1)

Génération de mots par dérivation

Soit `G = <T, N, S, P>` une grammaire

Règle terminale

$R \in P$ est une règle terminale si son membre à droite est un élément de T^*.

Dérivation directe

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

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 si :
$\exists k > 0$ et $(u_i)_{0 \leq i \leq k}$ tels que $u = u_1 \to u_2 \to ... \to u_k = v$.
(i.e. $u_1 = u$, $u_k = v$ et $\forall i < k, u_i \to u_{i+1}$).

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

$$L_G(S) = (identifiant|nombre)(operateur(identifiant|nombre))^*$$

Exemple de Grammaire

Exemple pour affectation numérique

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

- $T = \mathit{lettre} \cup \mathit{chiffre} \cup \{+, -, *, /, =, .\}$
- $N = \{\mathit{mot}, \mathit{nombre}, \mathit{entier}, \mathit{reel}, \mathit{operateur}, \mathit{suiteid}, \mathit{identifiant}\}$
- $S = \mathit{mot}$
- $P = \left\{ \begin{array}{l} \mathit{mot} \rightarrow \mathit{operateur} \mid \mathit{nombre} \mid \mathit{identifiant}, \\ \mathit{operateur} \rightarrow + \mid - \mid * \mid / \mid =, \\ \mathit{nombre} \rightarrow \mathit{entier} \mid \mathit{reel}, \\ \mathit{entier} \rightarrow \mathit{chiffre} \mid \mathit{entier}\ \mathit{chiffre}, \\ \mathit{reel} \rightarrow \mathit{entier}\ .\ \mathit{entier}, \\ \mathit{identifiant} \rightarrow \mathit{lettre} \mid \mathit{lettre}\ \mathit{suiteid}, \\ \mathit{suiteid} \rightarrow \mathit{lettre}\ \mathit{suiteid} \mid \mathit{chiffre}\ \mathit{suiteid} \mid \varepsilon \end{array} \right\}$

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

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 :
$A \rightarrow a$ où $A \in N$ et $a \in T$
$A \rightarrow \varepsilon$ où $A \in N$
$A \rightarrow B\ a$ (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>` de l'exemple précédent pour l'analyse lexicale de l'affectation numérique est de type 3 :

- $T = lettre \cup chiffre \cup \{+, -, *, /, =, .\}$
- $N = \{mot, suitereel, suitenb, suiteid\}$
- $S = mot$
- $P =$

$$\left\{ \begin{array}{l} mot \rightarrow +|-|*|/|=, \\ mot \rightarrow chiffre|lettre|chiffre\ suitenb|lettre\ suiteid, \\ suitenb \rightarrow chiffre|chiffre\ suitenb|.\ suitereel, \\ suitereel \rightarrow chiffre|chiffre\ suitereel, \\ suiteid \rightarrow lettre\ suiteid|chiffre\ suiteid \end{array} \right\}$$

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 :
$A \to \alpha$ où $A \in N$ 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 : $L_3 \subset L_2 \subset L_1 \subset L_0$