Theorie des Langages

- Numero: 1
- Prof: Fabrizio Jonathan
- Date: 9 Octobre 2017

Introduction

Site du prof

Lien du cours

On va parler de la partie de lecture de la compilation. Les algos sont relativement vieux (’70).

Quelques langages interprete: Python Dans un langage compile la traduction a deja eu lieu contrairement a un langage interprete.

Un langage interprete donne plus de flexibilie au dev. Un langage interprete a une plus grande portabilite.

Il y a des langage qui sont compile et interprete. Ex: Java

Definition

Un langage est un ensemble de **mot**. Un langage peut etre **infini**. (Exemple **N**: entier Naturel) Un langage n’a pas de relation d’ordre (On peut lui en ajouter un) Un langage peut etre **vide**.

Un mot est une sequence **fini** de **symboles** appartenant a un **alphabet**. Un mot peut etre vide (Note E: epsilon)

Un **alphabet** est un ensemble de symboles (Note Sigma). Un alphabet ne peut pas etre vide et infini. Ce qui **caracterise** un alphabet est son **cardinal** (le nombre de symboles).

Ex: $\sum = \{a, b\}$ $\sum^* = \{\varepsilon, a, b, aa, ab, ba, bb \ldots\}$

Il y a deux classes de langage: * Les langages recursivement enumerables: Il existe un algo pour lister tous les mots du langage. * Les langages recursif: Il existe un algo en tant fini pour tester si un mot appartient a un langage.

Donc un **langage recursif** est un **langage recursivement enumerables**.

Operations

Sur les mots

- $\sum^*$: L'ensemble des mots
- `.`: Concatenation (C'est un monoide)
  - `a.E = E.a = a` (`E` est l'element neutre)
  - `|a.b| = |a| + |b|`
  - `(a.b).c = a.(b.c)` (Associative)
  - `a.`` = { E si n = 0; a.a.a.a....a }`

Sur les langages

- $\cup$: Union
- $\cap$: Intersection
- $-$: Complement
- `.`: Concatenation
  - {E}.L = L.{E} = L (`{E}` est l'element vide)
  - $\emptyset$.L = ⊠
- $\star$: Etoile de Kleene