Rappels ou pas. . .

Un langage formel doit être intégralement spécifié.

Quelque soit la manière dont il a été spécifié, le langage obtenu doit permettre :

1. de produire des instances d'éléments du langage ;
2. d'analyser des instances existantes pour en valider leur appartenance.

Rappels ou pas. . .

Quelles sont les propriétés des grammaires formelles qui permettent cette spécification :

Des **grammaires formelles**, tel que les *grammaires hors-contexte* sont analysables par **des automates**.

Rappels ou pas. . .

Quelles sont les propriétés des grammaires formelles qui permettent cette spécification :

Des **grammaires formelles**, tel que les *grammaires hors-contexte* sont analysables par **des automates**.

les grammaires formelles : les mots sont produits par des règles en nombre fini, et se combinent dans des conditions précises ;

Rappels ou pas. . .

Quelles sont les propriétés des grammaires formelles qui permettent cette spécification :

Des **grammaires formelles**, tel que les *grammaires hors-contexte* sont analysables par **des automates**.

les grammaires formelles : les mots sont produits par des règles en nombre fini, et se combinent dans des conditions précises ;

les automates

Rappels ou pas. . .

Quelles sont les propriétés des grammaires formelles qui permettent cette spécification :

Des **grammaires formelles**, tel que les *grammaires hors-contexte* sont analysables par **des automates**.

les grammaires formelles

les automates : étant des mécanismes qui permettent de reconnaître les mots, mais également leurs enchaînements.

Rappels ou pas. . .

Quelles sont les propriétés des grammaires formelles qui permettent cette spécification :

Des **grammaires formelles**, tel que les *grammaires hors-contexte* sont analysables par **des automates**.

les grammaires formelles

les automates

Par exemple : les grammaires régulières : les mots sont décrits selon une symbolique qui permet de décrire des successions, des répétitions, des alternatives.

→ les expressions régulières sont un moyen très répandu pour la recherche de motifs dans du texte.

Rappels ou pas. . .

Quelles sont les propriétés des grammaires formelles qui permettent cette spécification :

Des **grammaires formelles**, tel que les *grammaires hors-contexte* sont analysables par **des automates**.

les grammaires formelles

les automates

Par exemple : les grammaires régulières : les mots sont décrits selon une symbolique qui permet de décrire des successions, des répétitions, des alternatives.

→ les expressions régulières sont un moyen très répandu pour la recherche de motifs dans du texte.

Question : chercher dans un texte exclusivement les séquences numériques sans séparateurs ?

Rappels ou pas. . .

Quelles sont les propriétés des grammaires formelles qui permettent cette spécification :

Des **grammaires formelles**, tel que les *grammaires hors-contexte* sont analysables par **des automates**.

les grammaires formelles

les automates

Par exemple : les grammaires régulières : les mots sont décrits selon une symbolique qui permet de décrire des successions, des répétitions, des alternatives.

→ les expressions régulières sont un moyen très répandu pour la recherche de motifs dans du texte.

Question : chercher dans un texte exclusivement les séquences numériques sans séparateurs ?

$/\hat{}([0-9]+)\$/$

Rappels ou pas. . .

Quelles sont les propriétés des grammaires formelles qui permettent cette spécification :

Des **grammaires formelles**, tel que les *grammaires hors-contexte* sont analysables par **des automates**.

les grammaires formelles
les automates
Par exemple : les grammaires régulières

→ les expressions régulières sont un moyen très répandu pour la recherche de motifs dans du texte.
Question : chercher dans un texte exclusivement les valeurs numériques entières et positives ?

Rappels ou pas. . .

Quelles sont les propriétés des grammaires formelles qui permettent cette spécification :

Des **grammaires formelles**, tel que les *grammaires hors-contexte* sont analysables par **des automates**.

les grammaires formelles
les automates
Par exemple : les grammaires régulières

→ les expressions régulières sont un moyen très répandu pour la recherche de motifs dans du texte.
Question : chercher dans un texte exclusivement les valeurs numériques entières et positives ?
/^([1 − 9][1 − 9]∗)$/

Rappels ou pas. . .

Quelles sont les propriétés des grammaires formelles qui permettent cette spécification :

Des **grammaires formelles**, tel que les *grammaires hors-contexte* sont analysables par **des automates**.

les grammaires formelles
les automates

Quelque soit la grammaire elle doit permettre de réaliser :
une analyse syntaxique

Rappels ou pas. . .

Quelles sont les propriétés des grammaires formelles qui permettent cette spécification :

Des **grammaires formelles**, tel que les *grammaires hors-contexte* sont analysables par **des automates**.

les grammaires formelles
les automates

Quelque soit la grammaire elle doit permettre de réaliser :
une analyse syntaxique

À retenir que, le résultat de cette analyse est :
une structure hiérarchique des mots du langage (syntagmes), représentable par un arbre syntaxique.

Rappels ou pas . . .

Alphabet, *Lettre*, *Mot*, *Langage*

- Un *Alphabet* A est un ensemble dont les éléments sont appelées des *Lettres*.
  ex : l'ensemble $\{1, 0\}$ c-a-d les chiffres binaires, l'ensemble des 26 lettres de l'alphabet français, etc.
- Un *Mot* est une suite finie $m_1 = a_1\ a_2\ \ldots\ a_n$ de *Lettres*, de longueur $= n$.
  ϵ le Mot vide.
  ex : 10110 mot créé à partir de l'alphabet $\{1, 0\}$ et de longueur$= 5$
- l'ensemble des mots de l'alphabet A est noté A^*
  ex : si $A = \{a, b\}$ alors $A^0 = \{\epsilon\},\ A^1 = \{a, b\},\ A^2 = \{aa, ab, ba, bb\}, \ldots$
- Tout sous-ensemble de A^* est un *Langage*
  y compris A^0 (langage composé du mot vide).
  ex : les mots binaires de parité paire :
  c-à-d que le nombre de bit à 1 est ? . . .

Rappels ou pas . . .

Alphabet, *Lettre*, *Mot*, *Langage*

- Un *Alphabet* A est un ensemble dont les éléments sont appelées des *Lettres*.
  ex : l'ensemble $\{1, 0\}$ c-a-d les chiffres binaires, l'ensemble des 26 lettres de l'alphabet français, etc.
- Un *Mot* est une suite finie $m_1 = a_1\ a_2\ \ldots\ a_n$ de *Lettres*, de longueur $= n$.
  ϵ le Mot vide.
  ex : 10110 mot créé à partir de l'alphabet $\{1, 0\}$ et de longueur$= 5$
- l'ensemble des mots de l'alphabet A est noté A^*
  ex : si $A = \{a, b\}$ alors $A^0 = \{\epsilon\},\ A^1 = \{a, b\},\ A^2 = \{aa, ab, ba, bb\}, \ldots$
- Tout sous-ensemble de A^* est un *Langage*
  y compris A^0 (langage composé du mot vide).
  ex : les mots binaires de parité paire :
  c-à-d que le nombre de bit à 1 est ? PAIRE : règle qui permet de définir les mots de ce *Langage*

Rappels ou pas. . .

Pour construire des ensembles de mots, on utilise la notion de *grammaire*

Grammaire

Une grammaire G comporte deux alphabets T et A et un ensemble P de production (règles de réécriture) .

- l'*Alphabet* T est dit alphabet *terminal*.Tous les mots construits par la grammaire sont constitués de lettres de T
- l'*Alphabet* A est dit alphabet *auxiliaire*, ses lettres servent de variables intermédiaires pour engendrer des mots.
- S est un symbole de plus haut niveau, appelé *symbole auxiliaire initiale* ou *axiome*
- P est l'ensemble fini des règles de *dérivation* ou de *production*

Excercice

Mais avant d'aller plus loin un petit test de vos connaissances. . .

Excercice

Mais avant d'aller plus loin un petit test de vos connaissances...

Catégorisation des langages informatiques :

- de spécification

Excercice

Mais avant d'aller plus loin un petit test de vos connaissances...

Catégorisation des langages informatiques :

- de spécification Z, B, UML, DSSSL, DTD, XSD...

Excercice

Mais avant d'aller plus loin un petit test de vos connaissances...

Catégorisation des langages informatiques :

- de spécification Z, B, UML, DSSSL, DTD, XSD...
- de programmation

Excercice

Mais avant d'aller plus loin un petit test de vos connaissances...

Catégorisation des langages informatiques :

- de spécification Z, B, UML, DSSSL, DTD, XSD...
- de programmation C_{++}, java, prolog, javascript, ada, python, PHP, XSLT ...

Excercice

Mais avant d'aller plus loin un petit test de vos connaissances...

Catégorisation des langages informatiques :

- de spécification Z, B, UML, DSSSL, DTD, XSD...
- de programmation C++, java, prolog, javascript, ada, python, PHP, XSLT ...
- de définition de données ou de requête

Excercice

Mais avant d'aller plus loin un petit test de vos connaissances...

Catégorisation des langages informatiques :

- de spécification Z, B, UML, DSSSL, DTD, XSD...
- de programmation C++, java, prolog, javascript, ada, python, PHP, XSLT ...
- de définition de données ou de requête SQL, OWL, SPARQL, XPATH, XMI ...

Excercice

Mais avant d'aller plus loin un petit test de vos connaissances...

Catégorisation des langages informatiques :

- de spécification Z, B, UML, DSSSL, DTD, XSD...
- de programmation C++, java, prolog, javascript, ada, python, PHP, XSLT ...
- de définition de données ou de requête SQL, OWL, SPARQL, XPATH, XMI ...
- de structuration de contenu

Excercice

Mais avant d'aller plus loin un petit test de vos connaissances...

Catégorisation des langages informatiques :

- de spécification Z, B, UML, DSSSL, DTD, XSD...
- de programmation C++, java, prolog, javascript, ada, python, PHP, XSLT ...
- de définition de données ou de requête SQL, OWL, SPARQL, XPATH, XMI ...
- de structuration de contenu, souvent connus sous le nom *de langage de balisage*, LaTeX, JSON (X)HTML, DocBook, TEI ...

Excercice

Mais avant d'aller plus loin un petit test de vos connaissances. . .

Catégorisation des langages informatiques :

- de spécification Z, B, UML, DSSSL, DTD, XSD. . .
- de programmation C$_{++}$, java, prolog, javascript, ada, python, PHP, XSLT . . .
- de définition de données ou de requête SQL, OWL, SPARQL, XPATH, XMI . . .
- de structuration de contenu, souvent connus sous le nom *de langage de balisage*, LaTeX, JSON (X)HTML, DocBook, TEI . . .
- dédiés DSL (Domain Specific Language)

Excercice

Mais avant d'aller plus loin un petit test de vos connaissances. . .

Catégorisation des langages informatiques :

- de spécification Z, B, UML, DSSSL, DTD, XSD. . .
- de programmation C$_{++}$, java, prolog, javascript, ada, python, PHP, XSLT . . .
- de définition de données ou de requête SQL, OWL, SPARQL, XPATH, XMI . . .
- de structuration de contenu, souvent connus sous le nom *de langage de balisage*, LaTeX, JSON (X)HTML, DocBook, TEI . . .
- dédiés DSL (Domain Specific Language) Graphviz (Graph Visualization), Postscript, PDF, GeoJSON, SVG, KML. . .

Excercice

Mais avant d'aller plus loin un petit test de vos connaissances. . .

Catégorisation des langages informatiques :

- de spécification Z, B, UML, DSSSL, DTD, XSD. . .
- de programmation C$_{++}$, java, prolog, javascript, ada, python, PHP, XSLT . . .
- de définition de données ou de requête SQL, OWL, SPARQL, XPATH, XMI . . .
- de structuration de contenu, souvent connus sous le nom *de langage de balisage*, LaTeX, JSON (X)HTML, DocBook, TEI . . .
- dédiés DSL (Domain Specific Language) Graphviz (Graph Visualization), Postscript, PDF, GeoJSON, SVG, KML. . .

où situer XML ?

Excercice

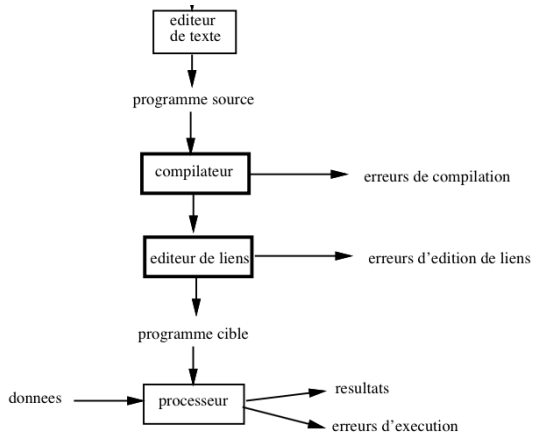
Mais avant d'aller plus loin un petit test de vos connaissances. . .

Catégorisation des langages informatiques :

- de spécification Z, B, UML, DSSSL, DTD, XSD. . .
- de programmation C$_{++}$, java, prolog, javascript, ada, python, PHP, XSLT . . .
- de définition de données ou de requête SQL, OWL, SPARQL, XPATH, XMI . . .
- de structuration de contenu, souvent connus sous le nom *de langage de balisage*, LaTeX, JSON (X)HTML, DocBook, TEI . . .
- dédiés DSL (Domain Specific Language) Graphviz (Graph Visualization), Postscript, PDF, GeoJSON, SVG, KML. . .

où situer XML ? Au dessus de tous les langages en bleu

Compilation
Principes



Compilation
Rôle de la compilation

- Tout code source doit être traduit en instructions élémentaires exécutables par le processeur : rôle du compilateur ;

  L'édition de liens (intégrée dans le compil.) résout les références à du code conservé dans des librairies ;
- Un interpréteur traduit directement chaque instructions du code source en instructions élémentaires exécutables ;
- Il existe des langages ou le compilateur traduit le code source en code plus compacte du *p-code (ou pseudo-code)*.

Compilation
Pourquoi la compilation ?

Mais les principes de "compilation" ne sont pas limités à la production de code exécutable.

Par exemple, ils s'appliquent également lors de la traduction :

- d'un code source vers un autre code source ;
- inverse d'un code exécutable vers un code source.

C-à-d, à la **traduction** d'un langage formel vers un autre langage formel.

Compilation
Étapes de base

La "traduction" se décompose en 2 phases :

1. **phase d'analyse :** reconnaissance du "vocabulaire" (variables, instructions et opérateurs), de la structure syntaxique ainsi que de certaines propriétés contextuelles ("sémantiques") ;
2. **phase de production :** traduction en langage cible.

Analyse
Première phase : Analyse Lexicale

Reconnaître le "vocabulaire" : les **unités lexicales (*token* ou symbole)**

Tout d'abord disposer :
- de l'alphabet *terminal* (V_T) du langage composé des symboles {le, chat} et
- de l'alphabet *auxiliaire(non-terminal)* (V_N) du langage composé des symboles {article, nom_commun, syntagme_Nom}
- S = *syntagme_Nom* : symbole auxiliaire initial ou axiome

Ensuite :
1. identifie tous les tokens qui appartiennent au langage ;
2. repère et isole toutes les autres parties du texte ;
3. fournit une suite plate de tokens reconnus.

Analyse
Première phase : Analyse Lexicale

Reconnaître le "vocabulaire" : les **unités lexicales (*token* ou symbole)**

Tout d'abord disposer :
- de l'alphabet *terminal* (V_T) du langage composé des symboles {le, chat} et
- de l'alphabet *auxiliaire(non-terminal)* (V_N) du langage composé des symboles {article, nom_commun, syntagme_Nom}
- S = *syntagme_Nom* : symbole auxiliaire initial ou axiome

Ensuite :
1. identifie tous les tokens qui appartiennent au langage ;
2. repère et isole toutes les autres parties du texte ;
3. fournit une suite plate de tokens reconnus.

Exercice : définir les alphabets, tokéniser et fournir la liste plate des tokens reconnus

$3 - x + y$ /* toto */

Analyse
Première phase : Analyse Lexicale

Reconnaître le "vocabulaire" : les **unités lexicales (*token* ou symbole)**

Tout d'abord disposer :
- de l'alphabet *terminal* (V_T) du langage composé des symboles {le, chat} et
- de l'alphabet *auxiliaire(non-terminal)* (V_N) du langage composé des symboles {article, nom_commun, syntagme_Nom}
- S = *syntagme_Nom* : symbole auxiliaire initial ou axiome

Ensuite :
1. identifie tous les tokens qui appartiennent au langage ;
2. repère et isole toutes les autres parties du texte ;
3. fournit une suite plate de tokens reconnus.

Exercice : définir les alphabets, tokéniser et fournir la liste plate des tokens reconnus

$3 - x + y$ /* toto */
/ toto / $y + x - 3$