DATA FORMATS AND FORMAL LANGUAGES (1/2) - REGULAR GRAMMAR

Matthieu Tixier – #4

Outline

- Data format specification
  - Database
  - Data structure (struct | objet)
  - Formal languages
- Formal language theory
  - Language vs Grammar
  - Chomsky hierarchy
  - Regular grammar and finite states automata
- Application to software engineering
  - Pattern matching
- Discussion

Data format specification

- Formal modeling for specification
  - The quest for a balance between ambiguity and expressivity
    ⇒ here: no ambiguity but limited meaning
  - Formal model as more compact notations (than Natural language)

- Data storage
  - Databases
    - Tables with recordings (rows)
    - A set of fields (columns)
      - With constraints on stored values types (INT, DOUBLE, DATE…)
  - Files containing structures or objects
    - A set of properties
    - Data stream

Data format specification

- Formal languages
  - Several description levels
    - Syntax: The rules that allows to combine symbols
      - E.g. : 03 51 59 12 61
      - There are **symbols** (number, space)
      - and **rules** :

        Two numbers must be followed by a space
        A maximum of 5 groups of two numbers is allowed

    - Semantic : the meaning of the expression as a symbol combination
      - The lottery winning numbers?
      - A chest security code?
      - A phone number (in France)?
      - ...

Formal languages theory

- Differences with « natural language » (NL)
  - Principle of compositionality of meaning is assumed
  - Ambiguity at several levels
    - "Time flies like an arrow"
  - Human languages as subject to evolution

- Relevance for software engineering
  - Programming languages
    - Syntactic validation for interpreters and compilers
  - Data format and protocols definition
    - Standard (e.g. IETF, ECMA)
  - Natural langage processing
    - Automated translation, spell checking, grammar checking…

Formal languages theory

- Formal language vs Formal grammar
  - « The two sides of a same coin »
  - **Language**: Let L be a set of expressions defined on a finite symbol alphabet Σ.
    - L is a part of Σ^*
    - $*$: « Kleene closure » unary operator
      - Σ^*, all the expressions that combines by concatenation any symbols of Σ (ε included).
      - with ε, the empty symbol
  - **Grammar**: the set of rules that defines the combination of symbols of Σ that are allowed for a langage L.
    - A concise definition for a formal language
    - Enable to check whether an expression belongs to a specific formal langage or not (ie, syntax checking)
    - Generation of well formed expressions

Formal languages theory

- For instance:
  - L1 a language defined on $\Sigma = \{a, b, c\}$

    with the following sequences:
    - $bacc, babacc, babababacc, \ldots$
  - L2 a language defined on

    with the following sequences:
    - $abc, aabcc, aaabccc, \ldots$

➔ As NL, a formal language comprises an infinite set of expression

Formal languages theory

- Formal grammar definition

L1, L2 ➔ share the same symbols but have a different structure

➔ How can we provide a minimal description of this difference?

Expressions en L1 : $bacc, babacc, babababacc$ ...

Expressions en L2 : $abc, aabcc, aaabccc$ ...

Formal languages theory

Formal grammar definition

L1, L2 → share the same symbols but have a different structure

- $G = (N, \Sigma, S, R)$
  - N : a finite alphabet of **non-terminal** symbols
  - Σ : a finite alphabet of **terminal** symbols
  - S : a particular element of N as start symbol
  - R : A finite set of production rules defined on $(N \cup \Sigma)^*$

→ How can we provide a minimal description of this difference?

L1 : $S \rightarrow baS$
$S \rightarrow cc$

L2 : $S \rightarrow aSc$
$S \rightarrow b$

Expressions en L1 : $bacc, babacc, babababacc$ ...

Expressions en L2 : $abc, aabcc, aaabccc$ ...

Types of formal grammar

- Chomsky hierarchy (1959)
  - Gathers 4 classes of decreasing expressivity (from 0 to 3)
    - For $\gamma \in (N \cup \Sigma)^*$ and ε the empty symbol

Type *Type*	**Nom** *Name*	**Forme des règles** *Constraints on grammar rules*
0	Récursivement énumérable *Recursively enumerable*	$\alpha \to \gamma$ tel que $\gamma \neq \varepsilon$
1	Sensible au contexte *Context sensitive grammar*	$\alpha A \beta \to \alpha \gamma \beta$ tel que $\gamma \neq \varepsilon$
2	CFG (« Context Free Grammar ») Grammaire algébrique	$A \to \gamma$
3	**Grammaire régulière** *Regular grammar*	$A \to \gamma B$ **ou** $A \to \gamma$ **avec** $\gamma \in \Sigma^*$

Regular grammar

- An instance of regular grammar ($A \to \gamma B$ or $A \to \gamma$ with $\gamma \in \Sigma^*$)

  The grammar of Dates (dd/mm) with D as start symbol:

  $D \to 0J \mid 1J \mid 2J \mid 30/M \mid 31/M$

  $J \to 0/M \mid 1/M \mid 2/M \mid 3/M \mid 4/M \mid 5/M \mid 6/M \mid 7/M \mid 8/M \mid 9/M$

  $M \to 0N \mid 1O$

  $N \to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$ $\qquad O \to 0 \mid 1 \mid 2$

  - As shown above, several non-terminal symbols can be used inside a formal grammar (be it regular or not)
  - Alternative rewriting rules for a same non-terminal symbol can be written with the symbol " | ".

- Limits in term of expressivity
  - For instance:

$$\{a_n b_n\} \text{ (for } n > 0)$$

Regular grammar

- Lets try it
  - Define a regular grammar for the "flu" language that contains (at least) the following expressions

ah, aaaah, aaaaaah, aaaaaah,   *aaaaaaaah-tchoum!!, aaaaaaaah-tchoum!!!!!!!, (...)*

  - Notes:
    - A regular grammar has to respect the constraints:

$$(A \rightarrow \gamma B \text{ or } A \rightarrow \gamma \text{ with } \gamma \in \Sigma^*)$$

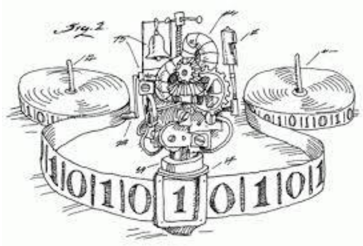
    - (…) is not part of the langage
and means there exist other similar expressions

Regular grammar - FSA

- Recognized by Finite-state machines (FSM or FSA)
  - A Finite-State Automaton (FSA) is equivalent to a regular grammar
  - Some intersting properties:
    - Determination : For all, non determinstic FSA, there exists a determinitic FSA that allows to recognize the same language
    - Operations over sets (intersection, union and complement), concat and closure *
- FSA as useful perspective when you feel limited within the formal grammar formalisms as it is equivalent

Regular grammar - FSA

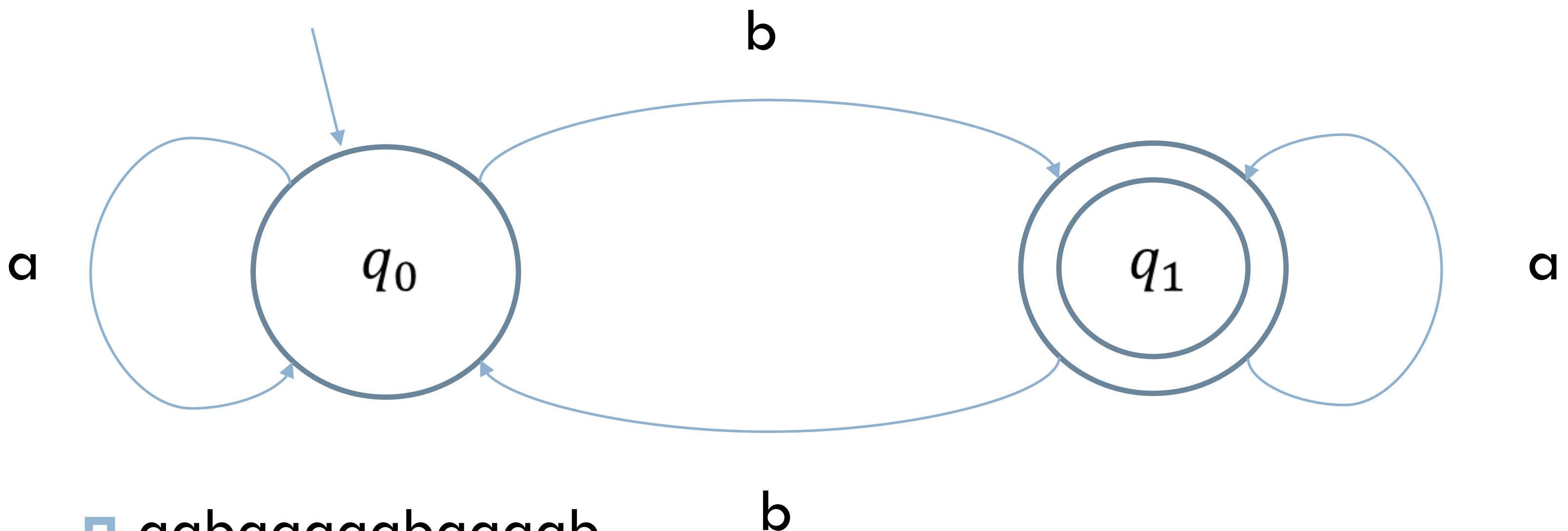
- Finite-state machine definition



  - An infinite tape of cells indexed by integer with a pointer
  - A control unit that controls the pointer moves over the tape given its state (the current cell index) and the read symbol
  - $M = (Q, \Sigma, q_0, F, \tau)$
    - Q : a finite set of unit control states
    - Σ : a finite alphabet of symbols that can be read on the tape
    - q_0 : a particular element from Q , the automaton initial state
    - F : a part of Q, the set of the automaton final states (états acceptants)
    - τ : a state-transition function that relates an input state $q \in Q$ and a read symbol $a \in \Sigma$ to an output state $\tau(q, a) : Q \times \Sigma \rightarrow Q$
  - An expression is recognized by the automaton if after n transitions the pointer is under one of the final states

Regular grammar - FSA

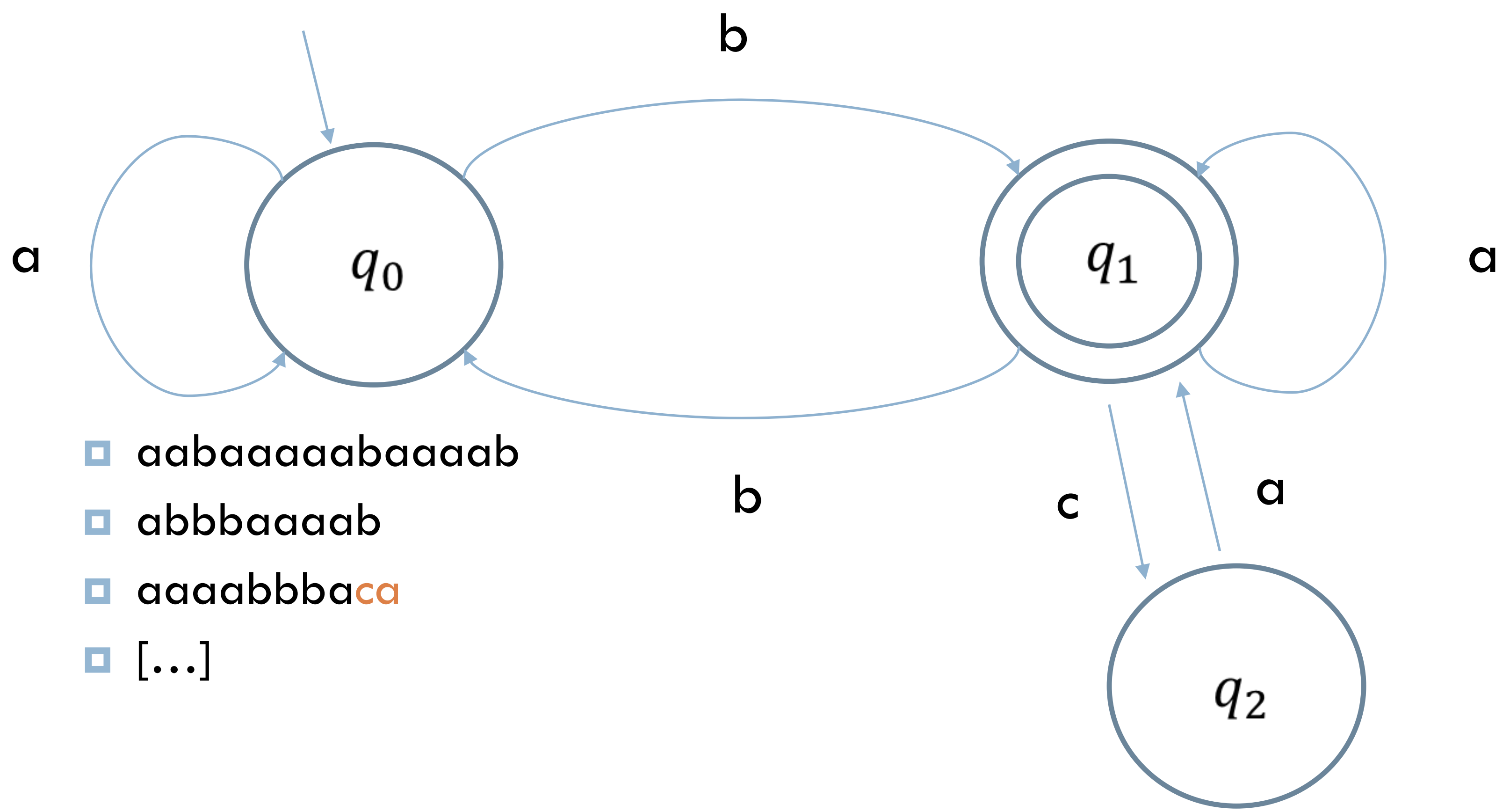
- Example



  - aabaaaaabaaaab
  - abbbaaaab
  - [...]

Regular grammar - FSA

- Example



  - aabaaaaabaaaab
  - abbbaaaab
  - aaaabbbaca
  - […]

Application for software engineering - RegEx

- Regular expressions
  - A compact syntax to express regular grammar
  - « pattern matching », useful to extract data from text

Termes	Exemples
Litterals	., [a-zA-Z], [^0-9], \w …
Quantifiers	+, ?, *, {3,}, {0,n}
Escaping character	\ (e.g. \\ , \. , \+)
Group(s) and backward reference	() (e.g. (a*)b\1)
Context mark	^, $

Application for software engineering - RegEx

- Regular expressions
  - A compact syntax to express regular grammar
  - « pattern matching », useful to extract data from text

Termes	Exemples
Litterals	., [a-zA-Z], [^0-9], \w …
Quantifiers	+, ?, *, {3,}, {0,n}
Escaping character	\ (e.g. \\ , \. , \+)
Group(s) and backward reference	() (e.g. (a*)b\1)
Context mark	^, $

Examples :

```
/^[\w\d\.-_]+@[\w\d\.-_]+\.\w+$/
/<img .*?src="([^"]+)"/
```

References

- Thanks for your attention
  - This week TD => M102
  - Question(s) ?

- Chomsky, N. (1959). "On certain formal properties of grammars". *Information and Control* 2 (2): 137–167
- Borges J. L. (1941) La bibliothèque de Babel (tr. « Fictions », Folio)
- Course based on « Introduction au Traitement Automatique du Langage » (G. Perrier, INRIA Lorraine)

Annexes