Finite state machines. Syllabification

NTV Lecture 2

April 1, 2003

Finite state and syllabification

- Relation between regular expressions and automata
- **Deterministic and non-deterministic automata**
- ϵ-**transitions**
- **Syllabification**
  - word splitting in order to justify paragraphs
- Part-of-speech tagging

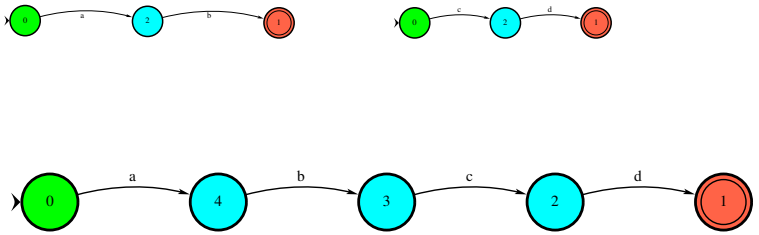
Last week

- **words**
  - fundamental building block of language
- **regular expressions**
  - specify text search strings
  - define the language recognized by an automaton
  - formally define a regular language
- **finite state automata** (FSA)
  - used to implement regular expressions
  - using finite alphabet may recognize infinite strings in a regular language

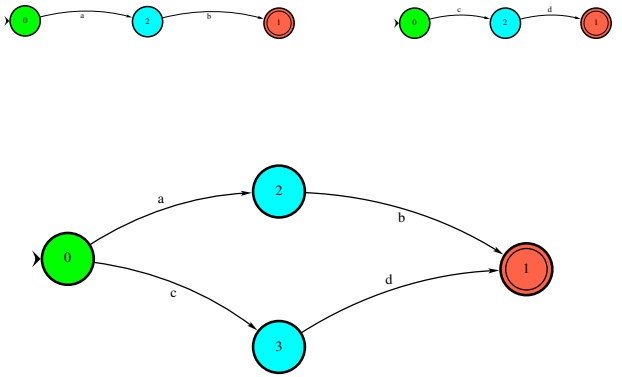
From regular expressions to finite automata

- Concatenation
  - $[A, B]$
  - Link final state(s) of automaton A with initial state of automaton B

Concatenation



Disjunction



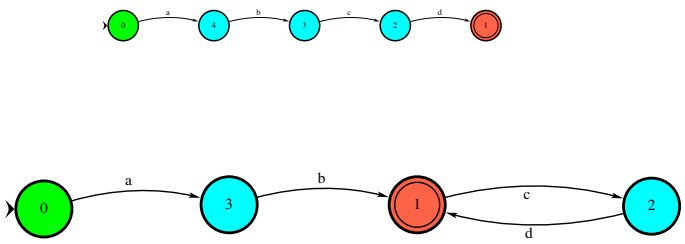
From regular expressions to finite automata

- Disjunction
  - ⋆ {A,B}
  - ⋆ Initial state A = Initial state B,
  - ⋆ Final state A = Final state B

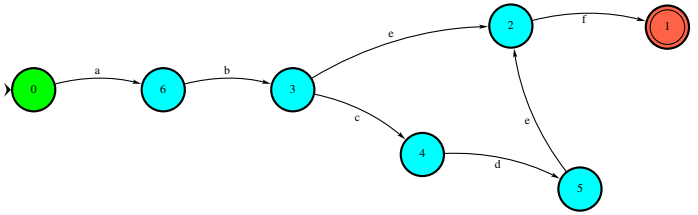
From regular expressions to finite automata

- Optional
  - ⋆ [A, B] → [A, B^]
  - ⋆ Add an epsilon-transition (jump) from the initial state of B to the final state(s) of B.

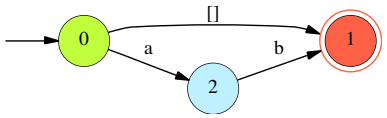
Optional



Epsilon-transitions in fsa



Epsilon-transitions (jumps)



From regular expressions to finite automata

- Kleene Plus: A^+
  - ⋆ Add an epsilon-transition from the final state of A to the initial state of A.
- Kleene Star: A^*
  - ⋆ Add an epsilon-transition from the final state of A to the inital state of A.
  - ⋆ Make the initial state of A also a final state.

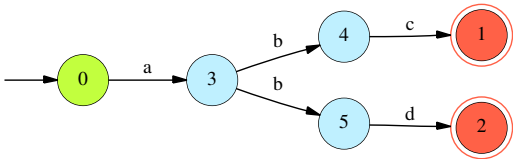
Determinism and Non-determinism

- An automaton is deterministic when being at any state Q looking at an input symbol S only one transition (move) is possible for the automaton.
- Automata with epsilon-transitions are non-deterministic.

Deterministic Recognizers

- For every recognizer with epsilon-transitions there is always an equivalent recognizer without jumps
- A non-deterministic recognizer can always be converted into a deterministic one.
- FSA produces deterministic recognizers

Non-deterministic recognizer



Syllabification (woorden afbreken)

- newspaper text fit into narrow columns
- long or complex words splitting
- hyphenation: (apparently) a simple typesetting problem
- in practice, not so simple (Volkskrant, 17-11-01)
  - Schaat-sunie
  - Bamboes-tok
  - Blessures-pook

Hyphenation rules

- respect word boundaries
  - Drugs-panden, drug-spanden
- Split syllables
  - Al-fa-bet, a-lfa-bet
- Split as early as possible (maximum onset rule)
  - Al-fa-bet, alf-a-bet, al-fab-et, alf-ab-et

What is a syllable? (lettergreep)

- A regular expression:
  - [onset^, nucleus, coda^]
  - Onset: {b, [b, r],[b, l], c,[c, h],. . . }
  - Nucleus: {a, [a, a], [a, a, i], e,. . . }
  - Coda: {b, c, [c, h], [c, h, t],...}

Simple syllabification program

- Set breaking points between syllables, as early as possible
- Gosse's algorithm evaluation :
  - 290.000 words (10,8 letters long, 2,5 hyphens per word)
  - 86% correct words
  - 94,5% correct hyphenation points
  - Errors are often compound words (samenstellingen)

A better syllabification program

- Machine learning algorithm helps to find hyphenation rules automatically
- Automatic syllabification of all words in Celex
- Comparison with correct syllabification
  - Rule i-st → is-t (li-stig → lis-tig) corrects 2900 errors (and introduces 300 new errors)
  - After learning 1400 rules 98,2% (words) and 99,2% (hyphens) correct

Regular expressions: macros

- Words with one syllable (monosyllable)
- Pattern:
  - consonants,vowels,consonants (medeklinkers,klinkers,medeklinkers)
  - macro(monosyllable,[cons$*$, vowel$_+$, cons$*$]).
  - macro(cons, { b,c,d,. . . ,z }).
  - macro(vowel, {,a,e,i,o,u,y}).

Macros 2

- In FSA macro is a label for a regular expression.
- macro(Name,RegExp).
- Macros can be used in the definition of other regular expressions
- To load macros in FSA use LoadAux.

Other applications: Part-of-speech tagging

- labelling of words with their word category
  - fiets $\rightarrow$ common noun, verb (1st sg present)
  - fietsen $\rightarrow$ common noun, verb (infinitive, 2nd–3rd pl present)
  - De fietsen staan in de schuur.
  - We fietsen naar school.
  - vliegen
- Typically this is the first step in syntactic analysis (description of sentence constituency)
- In a corpus with pos tags we can seek syntactic patterns
  - all sentences with 3 verbs, etc.
- POS-tagging : word recognition problem + word categorization problem

POS-tagging

- Word recognition problem:
  - Proper names : /[A. . . Z,a. . . z]* /
  - Verbs
    - /[a. . . z,{[e,n],[t],[de]}]/
    - /[g,e,a. . . z$^+$,{[e,n],[t],[d]}]/
- Usefulness of recognizers is limited because they only return a binary classification: 'yes' or 'no'
- Word categorization: more complex finite state machines are needed (finite state transducers)