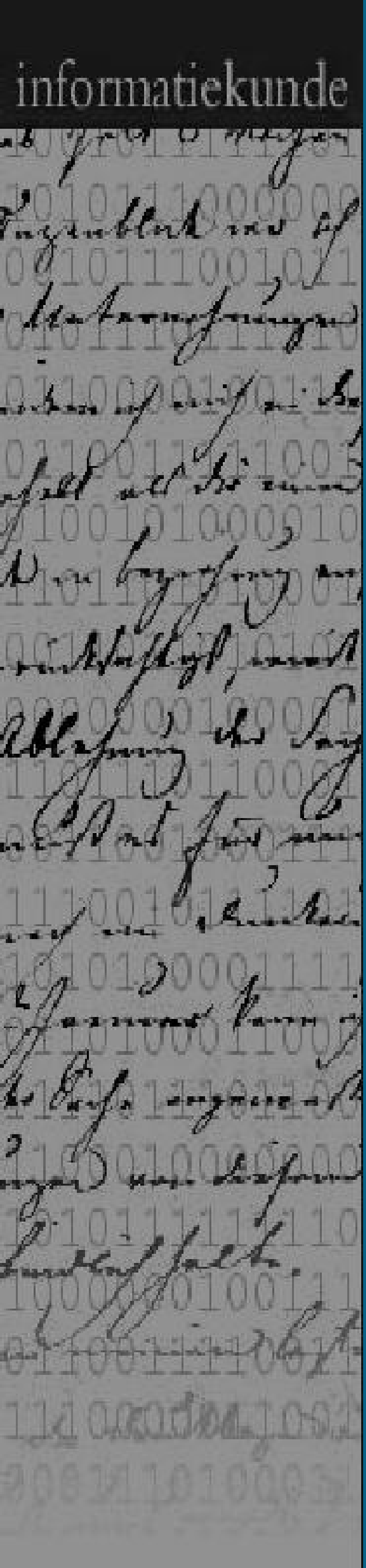
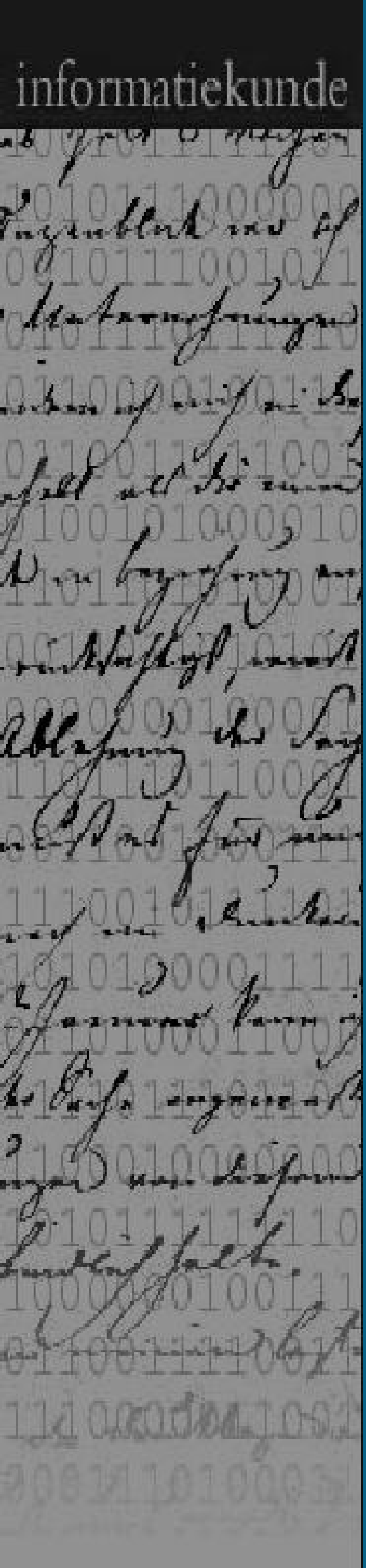


Regular expressions and Finite State Automata

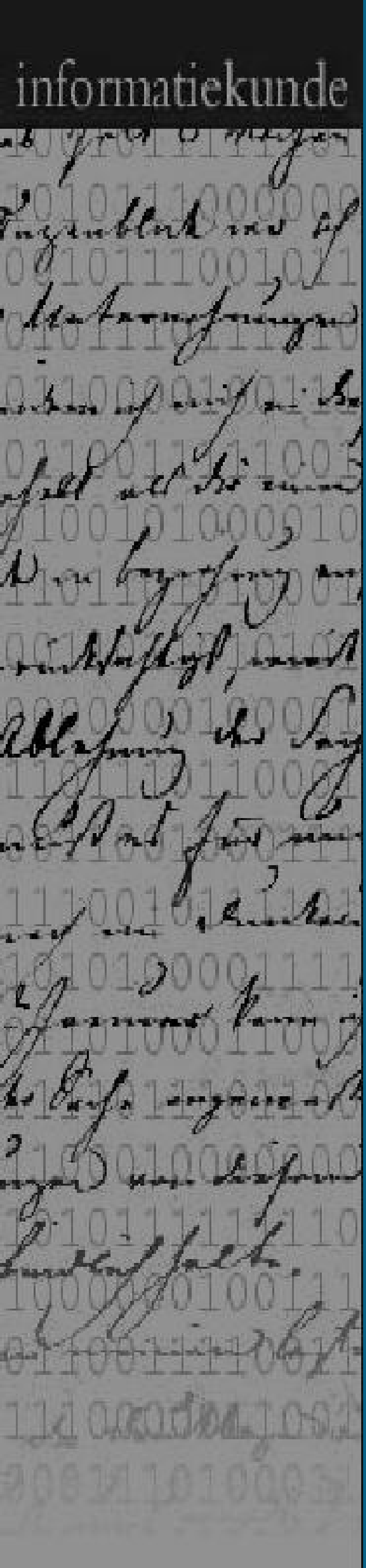
NLP Lecture 1

March 25, 2003



Overview

- **regular expressions**
  - uses
  - operators and patterns
- **finite state automata** (FSA)
  - relation between regexp and fsa
  - definition
  - uses
- Van Noord's **FSA Tools**
  - regular expression
  - writing macros



Regular expressions

- a formula for specifying text search strings

- a string is
  - ⋆ sequence of alphanumeric characters (letters, numbers, spaces, tabs and punctuation)
  - ⋆ /ath/ matches maths, path, Catherine Athenas
  - ⋆ /(r|m|s|l)am/ matches mambo samba lambada Partisam

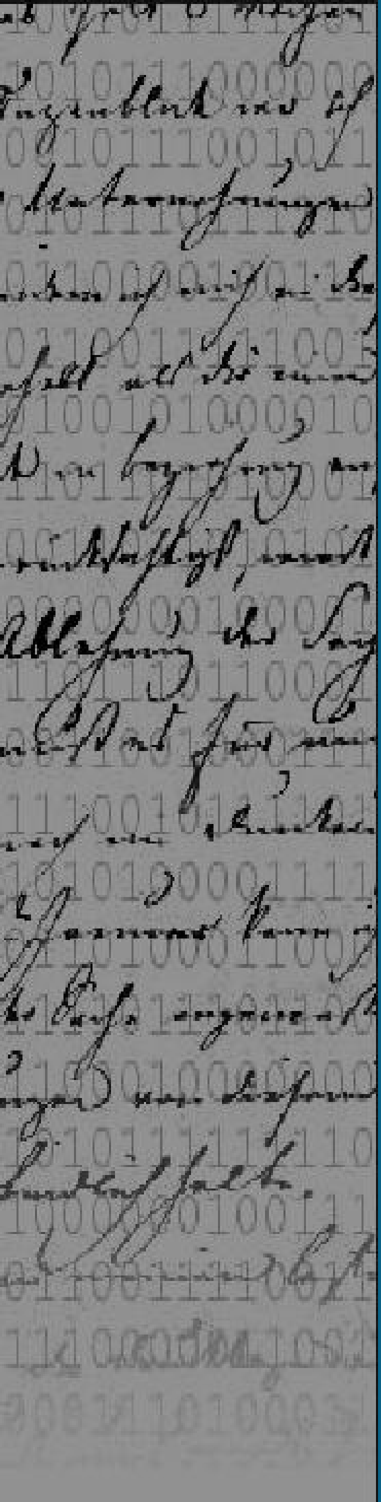
- used by many tools and applications
  - ⋆ UNIX, Text editors and numerous Web search engines

- important theoretical tool in computer science and linguistics

Definition

Formally

- algebraic notation for characterizing a set of strings
- useful to specify search strings
- and to define a language in a formal way



Regexp patterns

Pattern	Example
a single letter	/a/
sequence of characters	/bar /
ranges	[Bb], [A–Z], [a–z],[0–9]
negating	/[^ABC]/
Kleene $\star$	/a*/, /[ab]*/
Kleene $^+$	/a$^+$/
wildcards	/?/, /.*/

Disjunction and Grouping

- disjunction operator |
  - /terug|af|mee|aan/
- grouping (. . .)
  - / mee(kom|nem|breng|blijv)en /
  - / (doe|doet|doen|deed|deden|gedaan) /
  - / (burgemees|hop-|reggae-|supermini|tv-| schaats-|mega-)ster /
- Use: find all Dutch verbs beginning with **ver** or **voor** in a corpus
  - / (ver|voor).* / retrieves
  - vergeven, vervangen, verhuizen, verwijderen, vertellen,
  - voordragen, voorzien, voorkomen, voorbehouden, vooruitlopen

Finite State Automata

Regexp implemented by finite state automata. A regexp serves to define the set of strings (language) recognized by the finite state automaton.

Finite State Automata commonly used in NLP for

- grapheme to phoneme conversion
- breaking words into syllables
- stemming
- building dictionaries

What is an FSA?

[From Clocksin and Mellish: laughing machine]

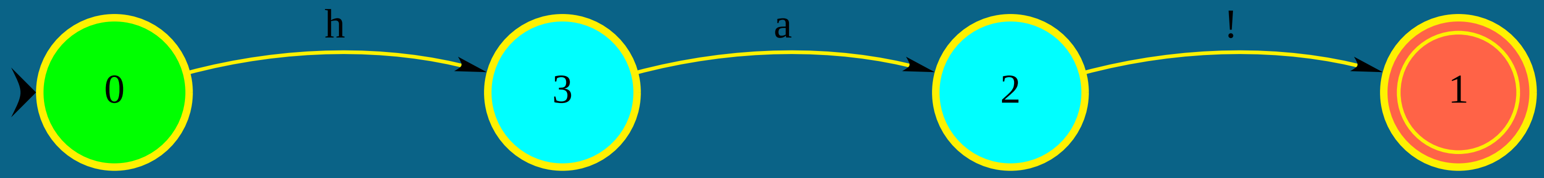
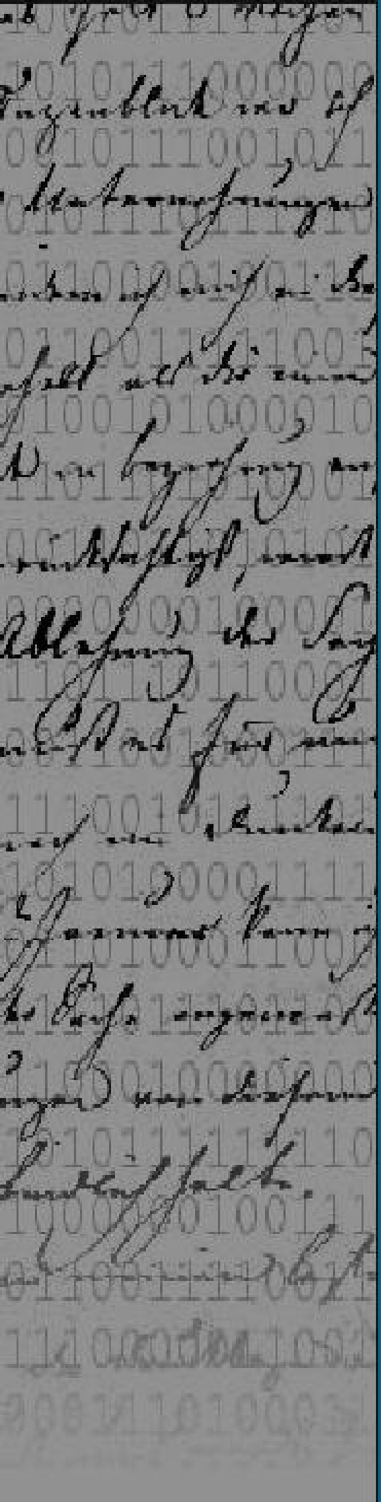
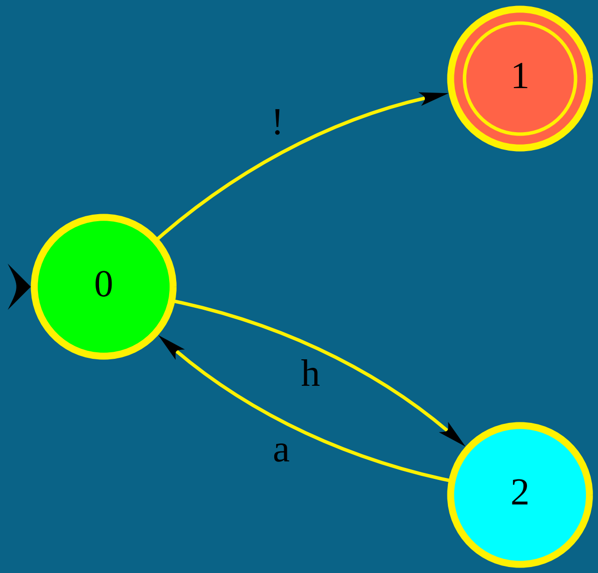
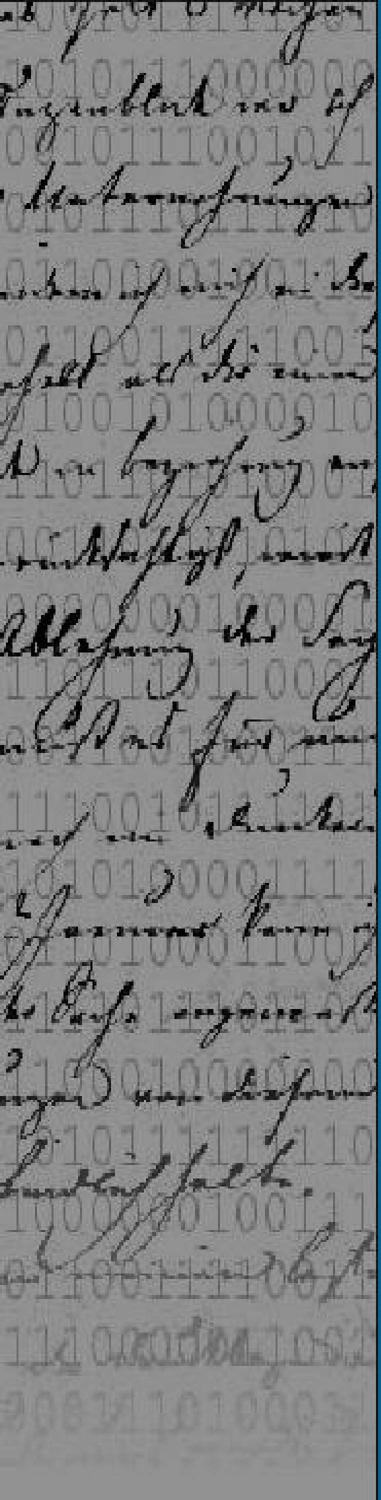


Figure 1: Laughing machine



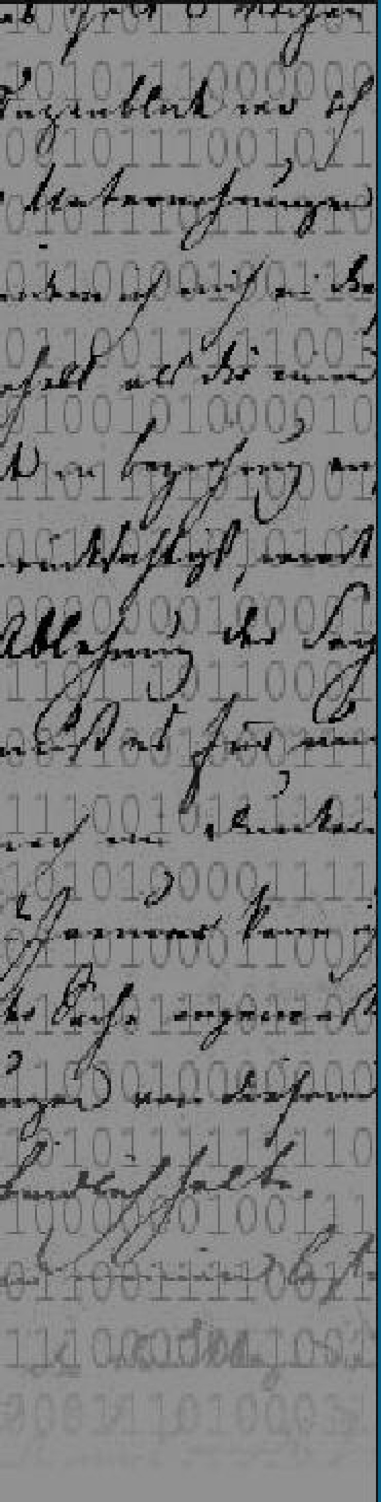
A reasonable laughing machine





What is an FSA?(2)

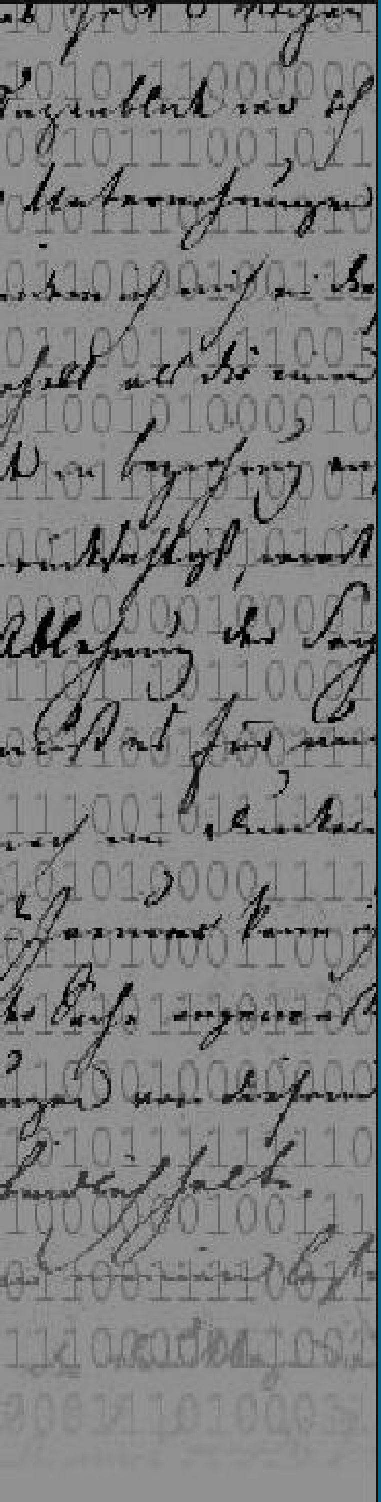
- a finite state laughing machine recognizes (or generates) strings of the form
  - ⋆ ha!
  - ⋆ haha!
  - ⋆ hahahaha!
- corresponding regular expression: $/(\text{ha})^{+}!/$
- a small vocabulary (h,a,!) and a finite state machine served us to formally define the laughing language
- NLP: recognition of verb paradigms, plural formation, word compounding, etc.



How to represent an FSA?

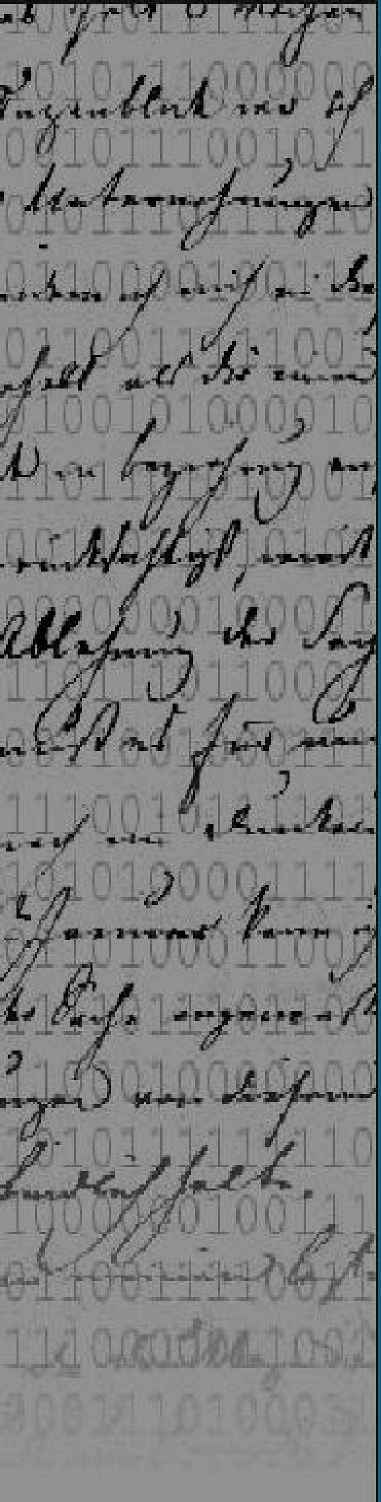
- as a directed graph with
  - ⋆ a finite set of vertices (states)
  - ⋆ vertices connected by links (archs)
  - ⋆ an alphabet (labels on archs)
  - ⋆ initial state and final state(s)

- A finite state automaton is a machine defined by
  - ⋆ **Q**: a finite set of N states (q_0,q_1,q_2. . . ,q_N)
  - ⋆ $\sum$: a finite set of input symbols: *alphabet*
  - ⋆ $\mathbf{q}_0$: the start state
  - ⋆ **F**: set of final states
  - ⋆ $\delta(q, i)$: transition function between states



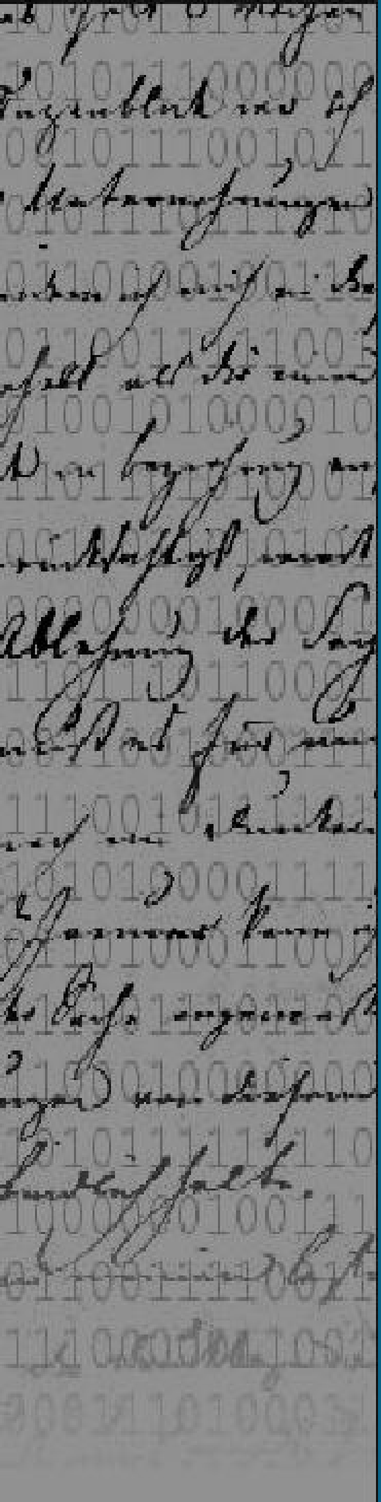
Transition function

Begin State	Input symbol	End State
0	h	2
2	a	0
0	!	1



Implementing FSA in Prolog

```
/* accept(L) succeeds if the list L belongs to the
   language defined by the FSA */

arc(0,h,2).          arc(2,a,0).           arc(0,!,1).

initial(0).          final(1).

accept(L) :-
      initial(P),
      accept0(L,P).

accept0([],F) :-
      final(F).

accept0([H|T],P) :-
      arc(P,H,Q),
      accept0(T,Q).
```



Gertjan Van Noord's FSA Toolkit

- Finite State Automata Utilities by Gerjan van Noord
  - ⋆ http://odur.let.rug.nl/~vannoord/Fsa/fsa.html
  - ⋆ sources and demo's
  - ⋆ manual describing regexp syntax and operators
- Tutorial
  - ⋆ http://odur.let.rug.nl/~gosse/tt/fsa.html
  - ⋆ Task 1: writing regular expressions, test strings accepted by the fsa, using patterns and operators

Finite State Automata

Next lecture we will explain

- difference between deterministic and non-deterministic automata
- epsilon transitions
- macros
- composition
- syllabification in Dutch
- assignment 1 of the course
- recommended readings: Syllabus chapters 2,3,4