T-(538|725)-MALV, Natural Language Processing
Finite-state automata

Hrafn Loftsson[1] Hannes Högni Vilhjálmsson[1]

[1]School of Computer Science, Reykjavik University

September 2009

Outline

1. Usage and definition
2. Types of automata
3. Operations

Outline

Finite-state automata

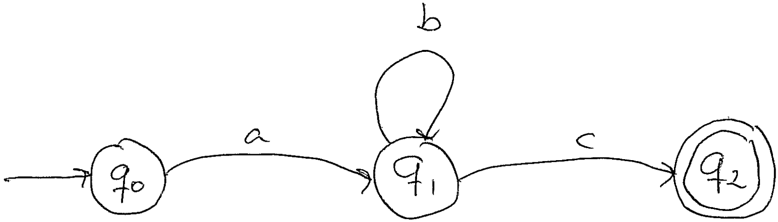
Usage

- We need to be able to search corpora for words or phrases.
- Search must extend beyond fixed strings.
  - For example, a word or its plural form, uppercase or lowercase letters
- This is where finite-state automata (FSA) (and regular expressions) help
- FSA are flexible tools for processing and searching texts.

Finite-state automaton

- A device which accepts or rejects an input stream of tokens (i.e. strings).
- Often called a "recognizer".
- Can also be used as a "generator", i.e. a device which generates strings.
- Very efficient in terms of speed and memory usage.
- Very suitable for text searching.

An example of a FSA



Finite-state automaton (FSA)

Mathematical definition

An FSA consists of five components $(Q, \Sigma, q_0, F, \delta)$:

1. Q is a finite set of states, q_0, $q_1 \ldots q_n$.
2. Σ is a finite set of input symbols.
3. q_0 is the start state, $q_0 \in Q$.
4. F is the set of final states, $F \subseteq Q$.
5. δ is the transition function $Q \times \Sigma \to Q$. $\delta(q, i)$ returns the state to which the automaton moves when it is in state q and consumes the input symbol i.

Example: $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{a, b, c\}$, $F = \{q_2\}$,
$\delta = \{\delta(q_0, a) = q_1, \delta(q_1, b) = q_1, \delta(q_1, c) = q_2\}$

Finite-state automaton (FSA)

Mathematical definition

An FSA consists of five components $(Q, \Sigma, q_0, F, \delta)$:

1. Q is a finite set of states, q_0, $q_1 \ldots q_n$.
2. Σ is a finite set of input symbols.
3. q_0 is the start state, $q_0 \in Q$.
4. F is the set of final states, $F \subseteq Q$.
5. δ is the transition function $Q \times \Sigma \rightarrow Q$. $\delta(q, i)$ returns the state to which the automaton moves when it is in state q and consumes the input symbol i.

Example: $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{a, b, c\}$, $F = \{q_2\}$,
$\delta = \{\delta(q_0, a) = q_1, \delta(q_1, b) = q_1, \delta(q_1, c) = q_2\}$

Outline

Finite-state automata: Two types

Deterministic Finite Automaton – DFA (í. Löggeng stöðuvél)

Given a state and an input, there is one single possible destination state.

Non-deterministic Finite Automaton – NFA (í. Brigðgeng stöðuvél)

- More than one path is possible from a state for an input.
- The path is not **determined** in advance.
- ϵ (the empty string) is an accepted input symbol.
- Example: Fig. 2.3 page 31.

An NFA can be converted to an equivalent DFA automatically.

Algorithm to simulate a DFA

- Input: a string x ending with EOF. DFA, D, with start state s_o and a set, F, of final states.
- Output: The answer "yes" if D recognises x, otherwise "no".

```
s = s_o
c = nextchar();
while (c <> EOF) {
    s = move(s, c);      // returns the state to which the automaton moves to
from state s on input c
    c = nextchar();
}
if s ∈ F then return "yes"
else return "no";
```

Outline

Operations on Finite-State Automata

Main operations

- Union (í. Sammengi)
- Concatenation (í. Samtenging)
- Iteration; "Kleene Closure" (í. Endurtekning)

Union

- The union of two automata A and B accepts (or generates) all strings of A and all strings of B.
- Denoted by $A \cup B$.
- Obtained by adding a new initial state with an ϵ-transition to both A and B

Operations on Finite-State Automata

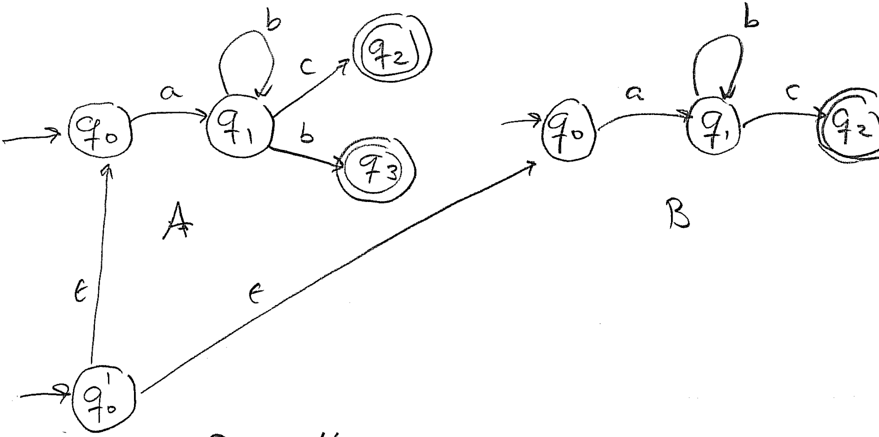
Main operations

- Union (í. Sammengi)
- Concatenation (í. Samtenging)
- Iteration; "Kleene Closure" (í. Endurtekning)

Union

- The union of two automata A and B accepts (or generates) all strings of A and all strings of B.
- Denoted by $A \cup B$.
- Obtained by adding a new initial state with an ϵ-transition to both A and B

An example of a union



$A \cup B$ = Union

Operations on Finite-State Automata

Concatenation

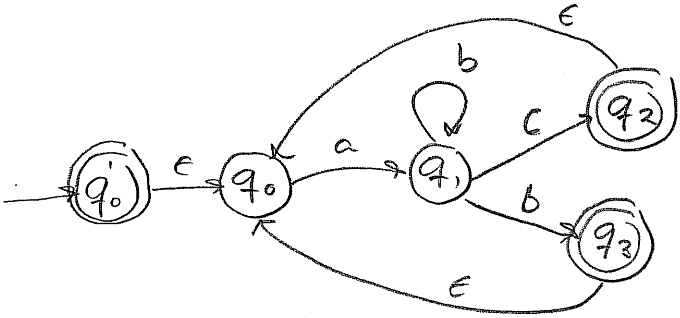
- The concatenation of two automata A and B accepts (or generates) all the strings that are concatenations of two strings, the first one being accepted by A and the second one by B.
- Denoted AB.
- Obtained by connecting all the final states of A to the initial state of B using an ϵ-transition (See Fig. 2.8 page 34).

Operations on Finite-State Automata

Iteration

- "Closure" af an automaton A accepts (or generates) the concatenations of any number of its strings and the empty string ϵ.
- Denoted A^*. $A^* = \{\epsilon\} \cup A \cup AA \cup AAA \cup \ldots$
- Obtained by linking the final state of A to its initial state using ϵ-transition and adding a new initial state.

An example of a closure



closure of A = A*

Operations on Finite-State Automata

Other common operations

- **Intersection** (í. Sniðmengi). The intersection of two automata $A \cap B$ accepts all the strings accepted both by A and B.
- **Difference** (í. Mismunur). The difference of two automata A - B accepts all the strings accepted by A but not by B.
- **Complementation** (í. Uppbót?).
  - Σ^* denotes the infinite set of all possible strings generated from the alphabet Σ.
  - The complementation of the automaton A in Σ^* accepts all the strings that are not accepted by A, i.e. $\hat{A} = \Sigma^* - A$.

Transformations to optimize speed and memory requirements

- ϵ-removal.
  - Transforms an initial automaton into an equivalent one without ϵ-transitions.
- Determination.
  - Transforms an NFA to a DFA.
- Minimisation.
  - Constructs an equivalent automaton with as few states as possible.