Computational Linguistics
Lecture 2 – Finite State Automata

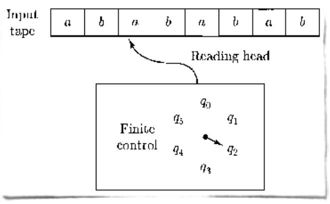
Dietrich Klakow & Stefan Thater
FR 4.7 Allgemeine Linguistik (Computerlinguistik)
Universität des Saarlandes

Summer 2012

Some basic definitions

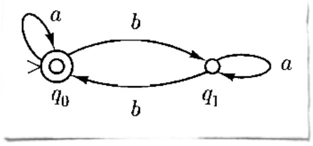
- **An alphabet Σ** is a finite set of symbols
- **A string over Σ** is a sequence of symbols from Σ
  - ε stands for the empty string
- **The length |w|** is the number of symbols in w
- **Σ^*** denotes this set of all strings over Σ
- **A (formal) language** is a subset of Σ^* for some alphabet Σ

Deterministic Finite Automata



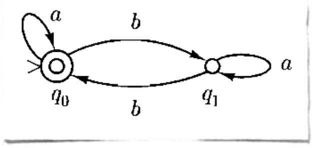
- **M = ⟨K, Σ, δ, s, F⟩**
  - K is a finite set of states
  - Σ is an input alphabet
  - δ is a transition function
  - s ∈ K is the start state
  - F ⊆ K is the set of final (accepting) states
- **Transition function**
  - δ(q, a) = q'
  - when M is in state q and reads input a, it goes into state q'

Automata as Graphs



- Nodes = states
- Edges = transition function
  - an edge from state q to state q' labeled by a ⇔ δ(q, a) = q'
- > marks the start state
- Double circles = final states

Automata as Graphs

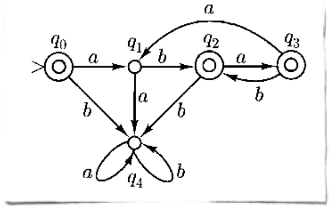


- **M = ⟨K, Σ, δ, s, F⟩**
  - K = {q_0, q_1}
  - Σ = {a, b}
  - s = q_0
  - F = {q_0}
  - δ(q_0, a) = q_0
  - δ(q_0, b) = q_1
  - δ(q_1, a) = q_0
  - δ(q_0, b) = q_1

More definitions

- **A configuration** is a pair ⟨q, w⟩ ∈ K × Σ*
  - q = the current state
  - w = the unread part of the string being processed
- **Yields in one step**
  - ⟨q, w⟩ $\vdash_M$ ⟨q', w'⟩
  - iff w = aw' for some a ∈ Σ, w' ∈ Σ* and δ(q, a) = q'
- **Yields**
  - $\vdash_M^*$ is the reflexive, transitive closure of $\vdash_M$
- **The language accepted** by a DFA M = ⟨K, Σ, δ, s, F⟩
  - L(M) = { w | ⟨s, w⟩ $\vdash_M^*$ ⟨q, ε⟩ for some q ∈ F }

An Example



$\langle q_0, ababa\rangle \vdash_M \langle q_1, baba\rangle$

$\vdash_M \langle q_2, aba\rangle$

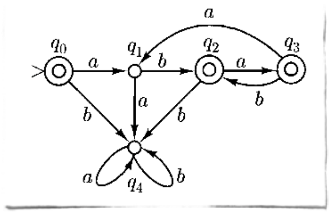
$\vdash_M \langle q_3, ba\rangle$

$\vdash_M \langle q_2, a\rangle$

$\vdash_M \langle q_3, \varepsilon\rangle$

$\Rightarrow ababa \in L(M)$

An Example



$\langle q_0, abaa\rangle \vdash_M \langle q_1, baa\rangle$

$\vdash_M \langle q_2, ba\rangle$

$\vdash_M \langle q_3, a\rangle$

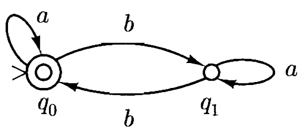
$\vdash_M \langle q_1, \varepsilon\rangle$

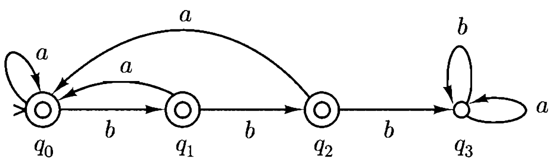
$\Rightarrow abaa \notin L(M)$

Recognition Algorithm

```
function RECOGNIZE(DFA M, STRING input)
  index ← 0
  state ← start state of M
  while index < length(input) do
      state ← trans[state, input[index]]
      index ← index + 1
  end
  if state is a final state of M
  then return accept
  else return reject
end
```

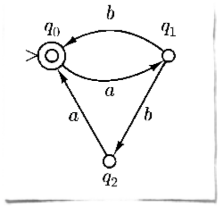
Exercise: L(M) = ?

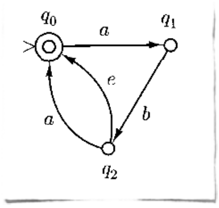




Nondeterministic Automata

- Nondeterministic finite automata:
  - several symbols can be read at once, or none at all
  - several possible next states

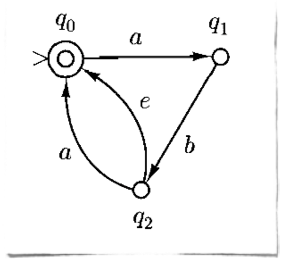




Nondeterministic Automata

- **M = ⟨K, Σ, Δ, s, F⟩**
  - K   is a finite set of states
  - Σ   is an input alphabet
  - Δ $\subseteq K \times \Sigma^* \times K$ is a finite transition relation
  - s $\in$ K is the start state
  - F $\subseteq$ K is the set of final (accepting) states
- **Transition relation** $\Delta \subseteq K \times \Sigma^* \times K$
  - ⟨q, w, q'⟩ $\in \Delta$ = when the automaton is in state q and reads input w, it can go into state q'
  - Note: here we restrict ourself to NFA where $|w| \leq 1$

An Example

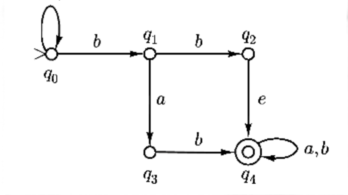


- **M = ⟨K, Σ, Δ, s, F⟩**
  - K = {q_0, q_1, q_2}
  - Σ = {a, b}
  - s = q_0
  - F = {q_0}
  - Δ = {⟨q_0, a, q_1⟩, ⟨q_1, b, q_2⟩, ⟨q_2, a, q_0⟩, ⟨q_2, ε, q_0⟩}

Configurations

- **Configurations**
  - are elements from K × Σ* (as before)
- **Yields in one step**
  - ⟨q, w⟩ $\vdash_M$ ⟨q', w'⟩
  - iff w = uw' for some u, w ∈ Σ* and ⟨q, u, q'⟩ ∈ Δ
- **The language accepted** by an NFA
  - L(M) = { w | ⟨s, w⟩ $\vdash_M^*$ ⟨q, ε⟩ for some q ∈ F }

An Example

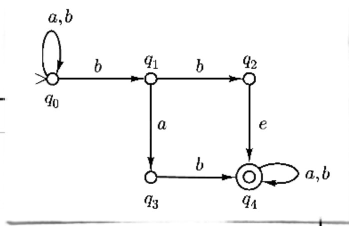


⟨q_0, babba⟩	⟨q_0, babba⟩	⟨q_0, babba⟩
$\vdash_M$ ⟨q_0, abba⟩	$\vdash_M$ ⟨q_1, abba⟩	$\vdash_M$ ⟨q_0, abba⟩
$\vdash_M$ ⟨q_0, bba⟩	$\vdash_M$ ⟨q_3, bba⟩	$\vdash_M$ ⟨q_0, bba⟩
$\vdash_M$ ⟨q_0, ba⟩	$\vdash_M$ ⟨q_4, ba⟩	$\vdash_M$ ⟨q_1, ba⟩
$\vdash_M$ ⟨q_1, a⟩	$\vdash_M$ ⟨q_4, a⟩	$\vdash_M$ ⟨q_2, a⟩
$\vdash_M$ ⟨q_3, ε⟩	$\vdash_M$ ⟨q_4, ε⟩	$\vdash_M$ ⟨q_4, a⟩
		$\vdash_M$ ⟨q_4, ε⟩

Recognition Algorithm

```
function RECOGNIZE(NFA M, STRING input)
  agenda = list of configurations, initially containing only
           the configuration (start state of M, input)
  while agenda is not empty do
    conf ← POP(agenda)
    if conf is an accepting configuration then
      return accept
    else
      for all conf' such that conf ⊢ conf' do
        PUSH(agenda, conf')
      end
  end
  return reject
end
```


An Example



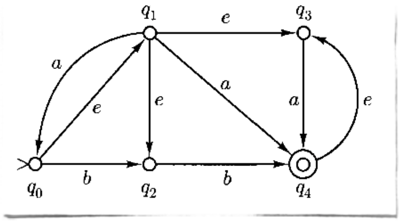
conf	agenda
–	⟨q_0, babba⟩
⟨q_0, babba⟩	**⟨q_0, abba⟩, ⟨q_1, abba⟩**
⟨q_0, abba⟩	**⟨q_0, bba⟩**, ⟨q_1, abba⟩
⟨q_0, bba⟩	**⟨q_0, ba⟩, ⟨q_1, ba⟩**, ⟨q_1, abba⟩
⟨q_0, ba⟩	**⟨q_0, a⟩, ⟨q_1, a⟩**, ⟨q_1, ba⟩, ⟨q_1, abba⟩
⟨q_0, a⟩	**⟨q_0, ε⟩**, ⟨q_1, a⟩, ⟨q_1, ba⟩, ⟨q_1, abba⟩
⟨q_0, ε⟩	⟨q_1, a⟩, ⟨q_1, ba⟩, ⟨q_1, abba⟩
⟨q_1, a⟩	**⟨q_3, ε⟩**, ⟨q_1, ba⟩, ⟨q_1, abba⟩
⟨q_3, ε⟩	⟨q_1, ba⟩, ⟨q_1, abba⟩
⟨q_1, ba⟩	**⟨q_2, a⟩**, ⟨q_1, abba⟩
⟨q_2, a⟩	**⟨q_4, a⟩**, ⟨q_1, abba⟩
⟨q_4, a⟩	**⟨q_4, ε⟩**, ⟨q_1, abba⟩
⟨q_4, ε⟩	⟨q_1, abba⟩

NFA = DFA (preliminary)

- **Theorem:** for every NFA M = ⟨K, Σ, Δ, s, F⟩ there is an equivalent DFA M' such that L(M) = L(M')
- Let us first consider the special case where for all elements ⟨q, w, q'⟩ ∈ Δ, w is a string of length 1
- We construct the DFA M' = ⟨K', Σ, Δ', s', F'⟩ as follows:
  - $K' = 2^K$
  - s' = {s}
  - δ(Q, a) = { k ∈ K | ⟨q, a, k⟩ ∈ Δ for some q ∈ Q }
    - for all Q ⊆ K
  - F' = { Q ⊆ K | Q ∩ F ≠ ∅ }

ε-Closure

- **ε-Closure**
  - $E(q) = \{ k \mid \langle q, \varepsilon \rangle \vdash^* \langle k, \varepsilon \rangle \}$
- Examples:
  - $E(q_0) = \{ q_0, q_1, q_2, q_3 \}$
  - $E(q_1) = \{ q_1, q_2, q_3 \}$
  - $E(q_2) = \{ q_2 \}$
- Note:
  - For all q, $q \in E(q)$



NFA = DFA

- **Theorem:** for every NFA $M = \langle K, \Sigma, \Delta, s, F \rangle$ there is an equivalent DFA M' such that L(M) = L(M')
- We construct the DFA $M' = \langle K', \Sigma, \delta', s', F' \rangle$ as follows:
  - $K' = 2^K$
  - $s' = E(s)$
  - $\delta(Q, a) = \cup\{ E(k) \in K \mid \langle q, a, k \rangle \in \Delta \text{ for some } q \in Q \}$,
    - for all $Q \subseteq K$
  - $F' = \{ Q \subseteq K \mid Q \cap F \neq \varnothing \}$

NFA = DFA

- **Lemma:** For any string $w \in \Sigma^*$ and any states $p, q \in K'$:
  - $\langle q, w \rangle \vdash^*_M \langle p, \varepsilon \rangle$ iff $\langle E(q), w \rangle \vdash^*_M \langle P, \varepsilon \rangle$ for some P containing p
- Using this lemma, it is easy to show that L(M) = L(M')
  - $w \in L(M)$
  - iff $\langle s, w \rangle \vdash^*_M \langle f, \varepsilon \rangle$, for some $f \in F$
  - iff $\langle E(s), w \rangle \vdash^*_{M'} \langle Q, \varepsilon \rangle$, for some Q containing f
  - iff $\langle E(s), w \rangle \vdash^*_{M'} \langle Q, \varepsilon \rangle$, for some $Q \in F'$
  - iff $w \in L(M')$

Subset construction algorithm

- **ε-closure(s)**
  returns the set of NFA states reachable from state s using ∈-transitions
- **ε-closure(T)**
  returns the set of NFA states reachable from some s in T using ∈-transition
- **move(T, a)**
  returns the set of NFA states to which there is transition for input a ∈ Σ from some state s ∈ T

Subset construction algorithm

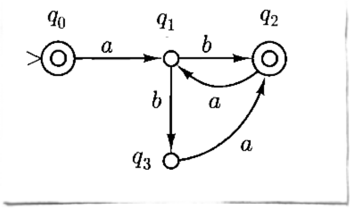
```
function DFA(K, Σ, Δ, s, F)
  K' ← list that contains only ε-closure(s), unmarked
  while there is an unmarked state T in K' do
    mark T
    for each symbol a ∈ Σ do
      U ← ε-closure(move(T, a))
      if U ∉ K' then
        add U as an unmarked state to K'
      δ[T, a] ← U
    end
  end
  return <the corresponding DFA>
end
```


Literature

- Jurafsky and Martin (2009). Chapter 2.
- Lewis and Papadimitriou (1981). Elements of the theroy of computation. Chapter 2.

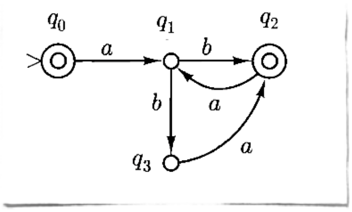
Exercise 1

- Apply (using pen and paper) the recognition algorithm on slide 16 for the nondeterministic automaton shown below to the input string "abaaba"
- There is a problem with this algorithm. Which one? How can the algorithm be improved?



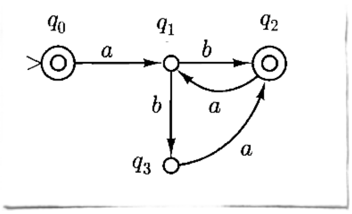
Exercise 2

- Construct a deterministic automaton for the nondeterministic automaton shown below, using the subset construction algorithm on slide 23.



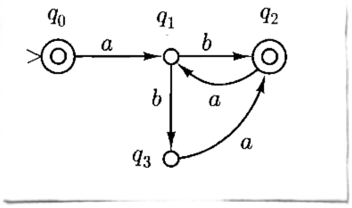
Exercise 3

- Implement the recognition algorithm for NFA on slide 16.
- Your submission should use the automaton shown below and the following inputs as test case.
  - ab ∈ L(M)
  - aba ∈ L(M)
  - abaab ∈ L(M)
  - abba ∉ L(M)
  - aabab ∉ L(M)
  - More test cases are welcome!



Exercise 4

- Implement the subset construction algorithm on slide 23.
- Your submission should use the automaton show below as a test case.



Exercises – Remarks

- Submit the source code by email to me (stth@...)
- The source code should contain a comment that tells how the code can be used.