ALE2
Automata in professional practice

Michael Franssen
Joris Geurts

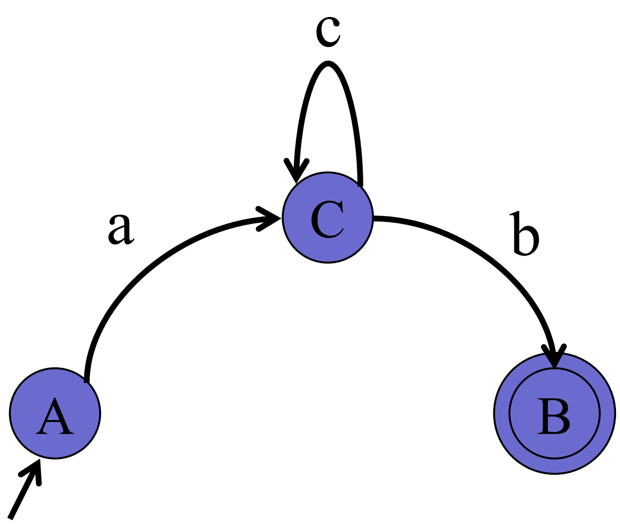
week overview

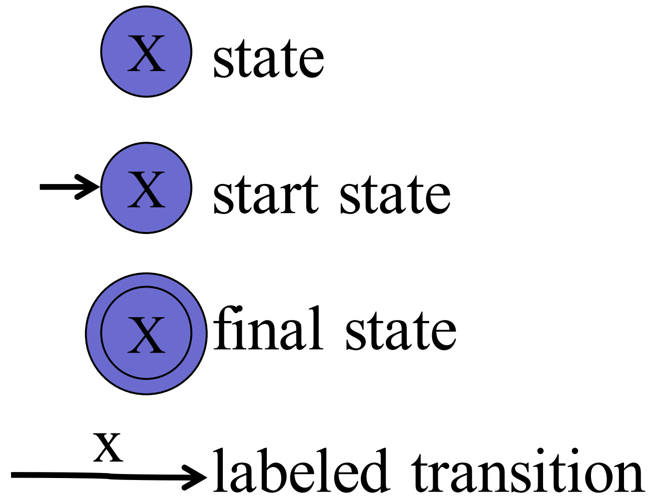
1. NDFA
2. regular expression
3. powerset construction for DFA
4. PDA
5. parallel automata

week 1: Finite State Automata

- finite number of states (set S)
- a starting state ($s \in S$)
- several final states ($F \subseteq S$)
- alphabet Σ of labels
- labeled transitions between states (transition relation δ: $(S x \Sigma) \hookrightarrow S$)

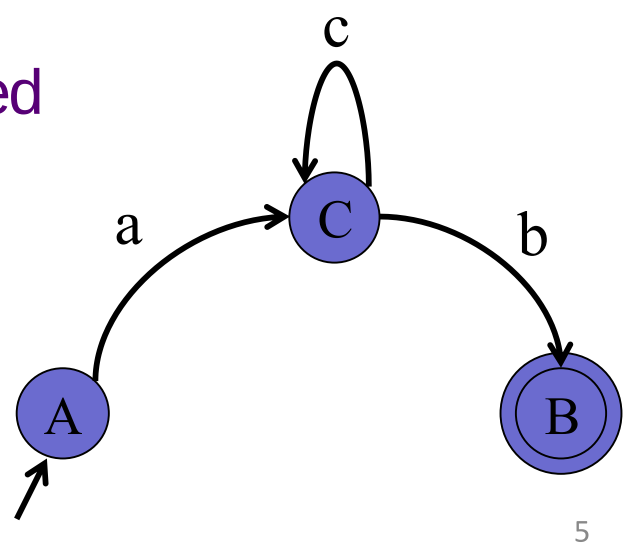
finite automata (II)





operational model

- an automaton is a machine that generates strings
  - every path from start state to final state produces a string being the concatenation of the labels of the transitions
    - path ACCB produces string acb
    - path ACB produces string ab
    - string acbb cannot be produced
- or: automaton *accepts* a given string
- language $\mathscr{L}$(A):
  set of all strings that can be produced by automaton A



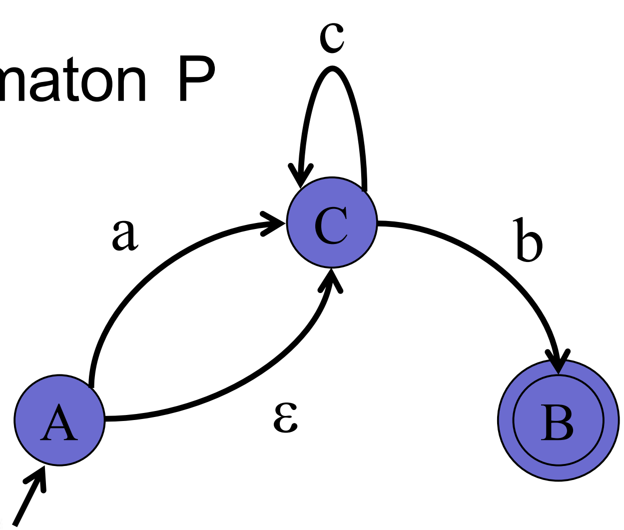
deterministic automata (DFA)

- exactly 1 transition for each state and each symbol of the alphabet
- therefor it's simpel to check if string s belongs to $\mathscr{L}(A)$
- other finite automata: non deterministic finite automata (NDFA)
- both types are equally powerful:
  - a DFA is een special case of an NDFA
  - for each NDFA you can construct a DFA that produces the same language (but this DFA has (theoretically) exponentially more states compared to the NDFA)

ε ('epsilon') transitions

- an NDFA may have transition with label ε (the empty string)
- if an automaton is in state `X` and `X` can go to `Y` via an ε transition, and `Y` goes to `Z` via label `a`, then `X` goes to `Z` via `a` as well

automaton P



- A goes via a to C
- A goes via b to B
- cccb $\in \mathscr{L}$(P)
- b $\in \mathscr{L}$(P)

week 2: regular expressions

You can use RE's to generate words:

- ❑ ε is a RE
- ❑ each letter a from alphabet Σ is a RE
- ❑ if x and y are REs, then x.y (concatenation), x|y (choice) en x* (repetition) are RE's as well.
- ❑ sometimes we leave the . for concatenation
- ❑ the language of a RE:
  - $\mathscr{L}(\varepsilon) = \emptyset$
  - $\mathscr{L}(a) = \{ a \}$
  - $\mathscr{L}(x.y) = \{ ab \mid a \in \mathscr{L}(x) \text{ and } b \in \mathscr{L}(y) \}$
  - $\mathscr{L}(x^*) = \{ a_1 a_2 a_3 \ldots a_n \mid a_i \in \mathscr{L}(x) \text{ for } 1 \leq i \leq n \text{ and } 0 \leq n \}$
  - $\mathscr{L}(x|y) = \{ a \mid a \in \mathscr{L}(x) \text{ or } a \in \mathscr{L}(y) \}$

regular expressions

examples:

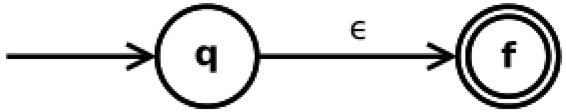
$\mathfrak{L}$(a.b*|c*) = {a, ab, abb, abbb, abbbb, …
ε, c, cc, ccc, cccc, … }

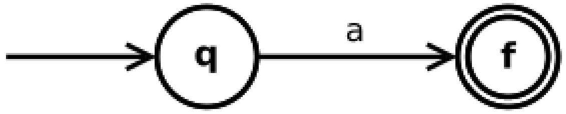
(note: ε is the empty string!)

Construct a NDFA from a RE: use "Thompson's construction"

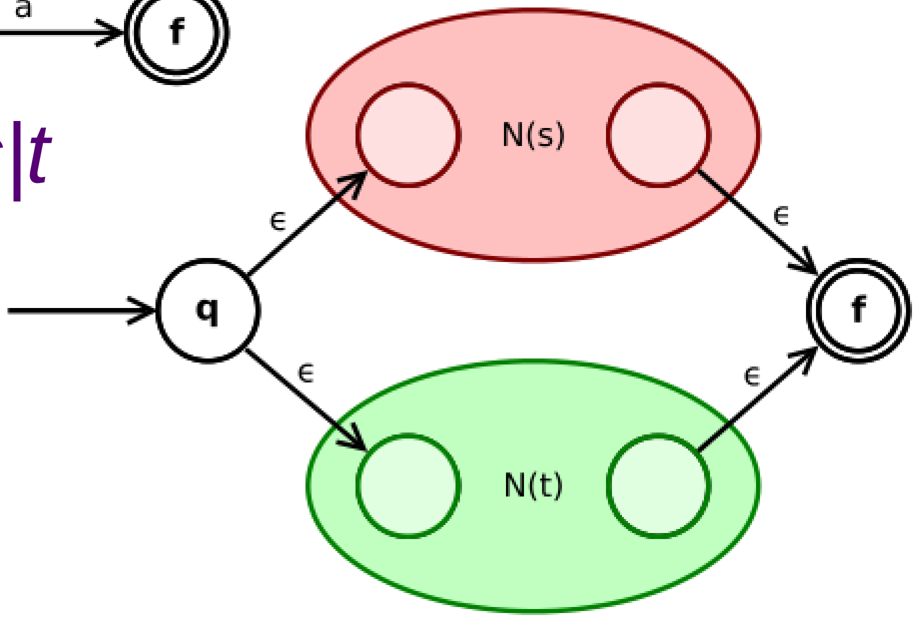
Thompson's construction (I)

- convert *ε* and *a*





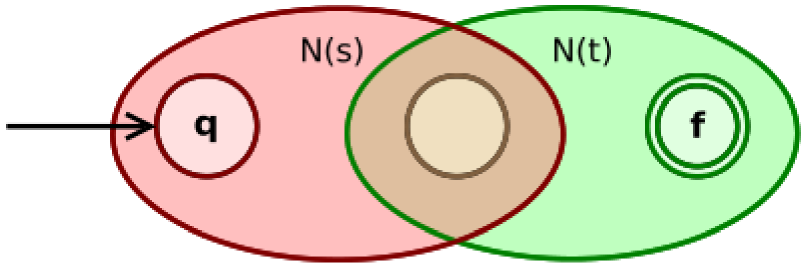
- convert *s*|*t*



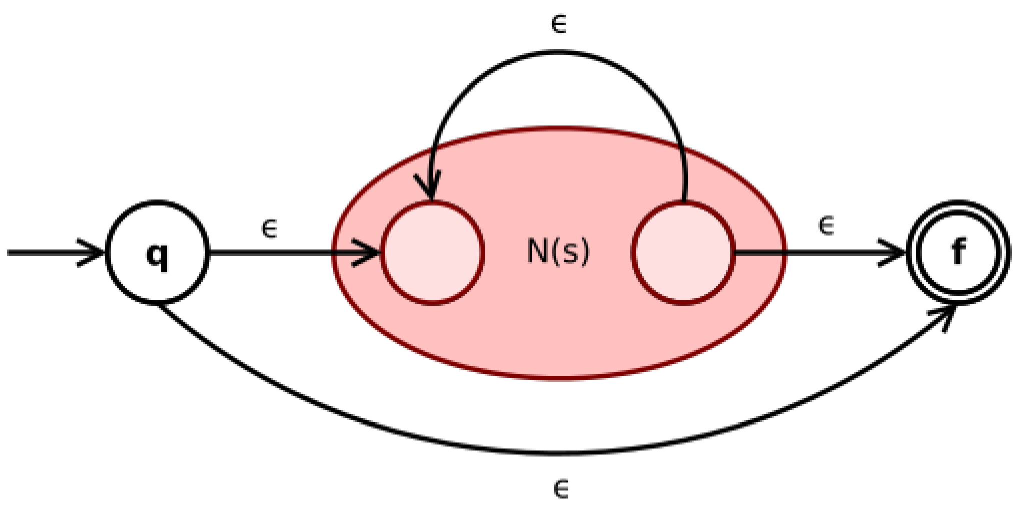
https://en.wikipedia.org/wiki/Thompson's_construction

Thompson's construction (II)

- convert *s·t* (concatenation)

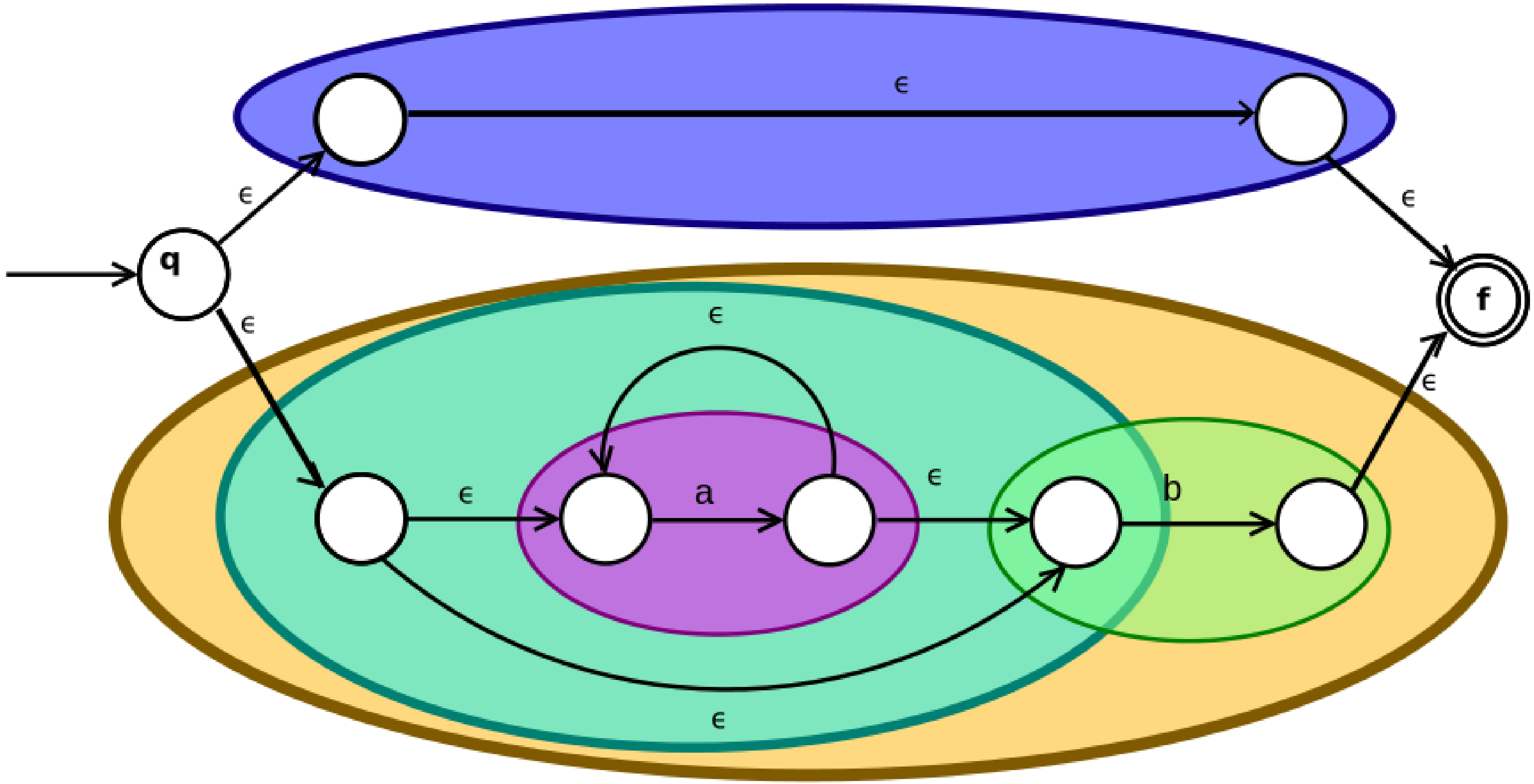


- convert Kleene star *s**



Thompson's construction (III)

- example: *(ε|a*b)*



week 3: powerset construction

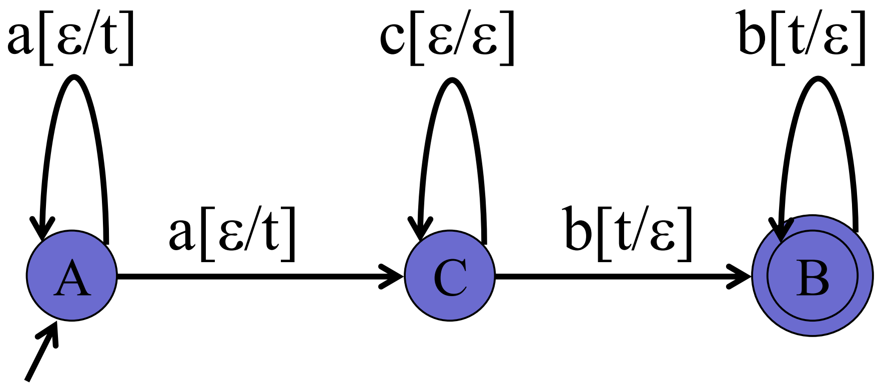
- see Math2 - week 6

week 4: Push Down Automaton

- contains a stack (with separate stack alphabet)
- in a state transition, element can be pushed onto and/or popped from the stack
- starts with empty stack
- a state transition can only be done when its stack symbol can indeed be popped from the stack
- ends when a final state is reached <u>and</u> when stack is empty

Fontys

example



- this automaton accepts all strings like $\mathtt{a}^n\mathtt{c}^m\mathtt{b}^n$, with n>0 and m≥0. So: `ab`, `acb`, `aaccccccbb`; but not: `aab`, `aaacbbbb`
- because: after n times reading an `a`, the stack contains n times a `t`. Now you can read as many `c`'s as you want. When reading the first `b`, the automaton is in state B, but it can only finish when all `t`'s are removed from the stack (so after reading n times `b`)

week 5: parallel automata (I)

- an automaton can be regarded as a machine who takes actions
- the generated string is the trace of action that this machine has done
- or: the given string represent the actions that this machine needs to execute
- in this way, we can observe the behaviour of a system with two (or more) parallel automata
- the state of this system is the combination of the states of the separate automata

parallel automata (II)

- suppose automaton A with alphabet Σ_A, states S_A, start state s_A, final states F_A and transition function δ_A
- and automaton B with Σ_B, S_B, s_B, F_B and δ_B
- automaton AxB models the behaviour of the parallel operating automata A and B
- AxB is defined by alphabet $\Sigma_A x \Sigma_B$ and states $S_A x S_B$. AxB starts in (s_A,s_B). The final states are those (s_a,s_b) with $s_a \in F_A$ and $s_b \in F_B$.
- transition function δ_{AxB} describes the parallel behaviour

Fontys

parallel automata (III)

- AxB can make a transition from (s_{a1},s_{b1}) with input (x,ε) to (s_{a2},s_{b1}) if $\delta_A(s_{a1},x)=s_{a2}$.
- AxB can make a transition from (s_{a1},s_{b1}) with input (ε,y) to (s_{a1},s_{b2}) if $\delta_B(s_{b1},y)=s_{b2}$.

- with those rules we have interleaving parallellisme: in fact the automaton does only one step every time
- real parallellism can be obtained by adding this rule: AxB can make a transition from (s_{a1},s_{b1}) with input (x,y) to (s_{a2},s_{b2}) if $\delta_A(s_{a1},x)=s_{a2}$ *and* $\delta_B(s_{b1},y)=s_{b2}$.

Fontys

parallel automata (IV)

- running two independent automata in parallel is not interesting
- therefor we add symbols R and T to both alphabets $\sum_A$ and $\sum_B$ for communication
- transitions for symbols R and T are only done when:
  - AxB makes a transition from (s_{a1},s_{b1}) with input (R,R) to (s_{a2},s_{b2}) when $\delta_A(s_{a1},R)=s_{a2}$ <u>and</u> $\delta_B(s_{b1},R)=s_{b2}$.
  - AxB makes a transition from (s_{a1},s_{b1}) with input (T,T) to (s_{a2},s_{b2}) when $\delta_A(s_{a1},T)=s_{a2}$ <u>and</u> $\delta_B(s_{b1},T)=s_{b2}$.
- so: communication steps are only done when both automata are at the synchronization point at the same time

Fontys