Finite automata and formal languages (DIT322, TMV028)

Nils Anders Danielsson

2020-02-03

Today

- NFAs with ε-transitions.
- Exponential blowup.

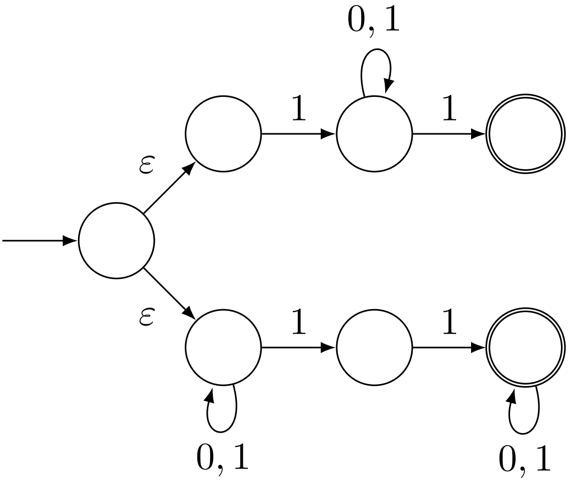
ε-NFAs

ε-NFAs

- Like NFAs, but with ε-transitions:
  The automaton can "spontaneously" make a transition from one state to another.
- Can be used to convert regular expressions to finite automata.

ε-NFAs

Strings over $\{ 0, 1 \}$ that start and end with a one, or that contain two consecutive ones:



ε-NFAs

An ε-NFA can be given by a 5-tuple $(Q, \Sigma, \delta, q_0, F)$:

- A finite set of states (Q).
- An alphabet (Σ with $\varepsilon \notin \Sigma$).
- A transition function ($\delta \in Q \times (\Sigma \cup \{ \varepsilon \}) \to \wp(Q)$).
- A start state ($q_0 \in Q$).
- A set of accepting states ($F \subseteq Q$).

Transition diagrams

As for NFAs, but arrows can be labelled with ε.

Transition tables

As for NFAs, but with one column for ε.

ε-closure

ε-closure

The ε-closure of a state q consists of those states that one can reach from q by following zero or more ε-transitions.

ε-closure

Given an ε-NFA $A = (Q, \Sigma, \delta, q_0, F)$ one can, for each state $q \in Q$, define the ε-closure of q (a subset of Q) inductively in the following way:

$$\frac{}{q \in \varepsilon\text{-}closure(q)}$$

$$\frac{q' \in \varepsilon\text{-}closure(q) \qquad q'' \in \delta(q', \varepsilon)}{q'' \in \varepsilon\text{-}closure(q)}$$

Some notation

The ε-closure of a set $S \subseteq Q$:

$$\varepsilon\text{-}closure(S) = \bigcup_{s \in S} \varepsilon\text{-}closure(s)$$

Transition functions applied to a set $S \subseteq Q$:

$$\delta(S, a) = \bigcup_{s \in S} \delta(s, a)$$

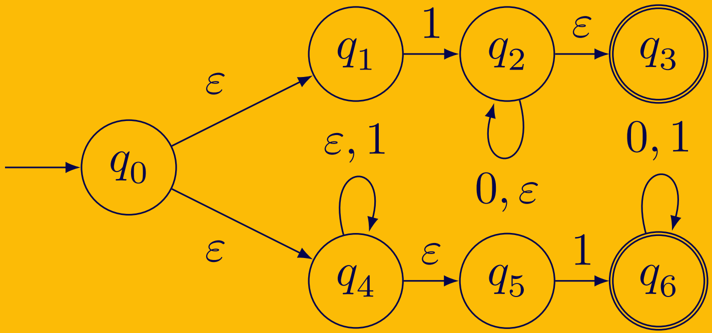
$$\hat{\delta}(S, w) = \bigcup_{s \in S} \hat{\delta}(s, w)$$

Computing the ε-closure

The ε-closure of q can be computed (perhaps not very efficiently) in the following way:

- Initialise C to $\{ q \}$.
- Repeat until $\delta(C, \varepsilon) \subseteq C$:
  - Set C to $C \cup \delta(C, \varepsilon)$.
- Return C.

Which of the following propositions hold for the following ε-NFA over $\{0, 1\}$?



1. $q_0 \in \varepsilon\text{-}closure(q_0)$.
2. $q_5 \in \varepsilon\text{-}closure(q_0)$.
3. $\varepsilon\text{-}closure(q_4) \subseteq \varepsilon\text{-}closure(q_0)$.
4. $q_6 \in \varepsilon\text{-}closure(q_0)$.
5. $q_3 \in \varepsilon\text{-}closure(q_1)$.
6. $\varepsilon\text{-}closure(q_4) \subseteq \varepsilon\text{-}closure(q_5)$.

Semantics

The language of an ε-NFA

The language $L(A)$ of an ε-NFA $A = (Q, \Sigma, \delta, q_0, F)$ is defined in the following way:

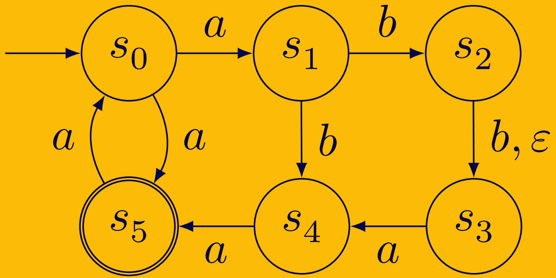
- A transition function for strings is defined by recursion:

$$\begin{aligned} &\hat{\delta} \in Q \times \Sigma^* \to \wp(Q) \\ &\hat{\delta}(q, \varepsilon) \quad = \varepsilon\text{-}closure(q) \\ &\hat{\delta}(q, aw) = \hat{\delta}(\delta(\varepsilon\text{-}closure(q), a), w) \end{aligned}$$

- The language is

$$\left\{\, w \in \Sigma^* \;\middle|\; \hat{\delta}(q_0, w) \cap F \neq \emptyset \,\right\}.$$

Which strings are members of the language of the following ε-NFA over $\{ a, b, c \}$?



1. $abba$.
2. $abbaca$.
3. $aaabaa$.
4. $aaabaaa$.
5. $aaaabaa$.
6. $abbaaaabaa$.

Which of the following propositions are valid?

1. $\varepsilon\text{-}closure(\varepsilon\text{-}closure(q)) = \varepsilon\text{-}closure(q)$.
2. $\hat{\delta}(q, w) = \hat{\delta}(\varepsilon\text{-}closure(q), w)$.
3. $\hat{\delta}(\delta(\varepsilon\text{-}closure(q), a), w) =$
   $\hat{\delta}(\varepsilon\text{-}closure(\delta(q, a)), w)$.

Constructions

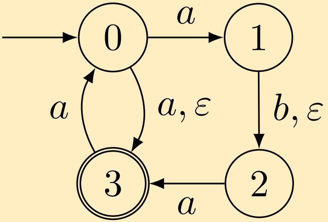
Subset construction

Given an ε-NFA $N = (Q, \Sigma, \delta, q_0, F)$ we can define a DFA D with the same alphabet in such a way that $L(N) = L(D)$:

$$D = (\wp(Q), \Sigma, \delta', \varepsilon\text{-}closure(q_0), F')$$
$$\delta'(S, a) = \varepsilon\text{-}closure(\delta(S, a))$$
$$F' = \{\, S \subseteq Q \mid S \cap F \neq \emptyset \,\}$$

Every accessible state S is *ε-closed* (i.e. $S = \varepsilon\text{-}closure(S)$).

If the subset construction is used to build a DFA corresponding to the following ε-NFA over $\{ a, b \}$, and inaccessible states are removed, how many states are there in the resulting DFA?



Regular languages

- Recall that a language $M \subseteq \Sigma^*$ is regular if there is some DFA (or NFA) A with alphabet Σ such that $L(A) = M$.
- For alphabets Σ with $\varepsilon \notin \Sigma$ a language $M \subseteq \Sigma^*$ is also regular if and only if there is some ε-*NFA* A with alphabet Σ such that $L(A) = M$.

Union

Recall:

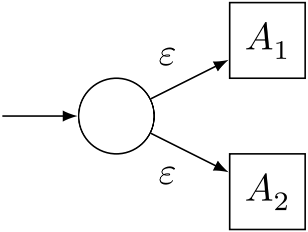
- One can use ε-NFAs to convert regular expressions to finite automata.

Union

Given two ε-NFAs A_1 and A_2 with the same alphabet we can construct an ε-NFA $A_1 \oplus A_2$ that satisfies the following property:

$$L(A_1 \oplus A_2) = L(A_1) \cup L(A_2).$$

Construction:



- The transitions go to the start states.
- States are renamed if the state sets overlap.

Can one do something similar for NFAs by "merging" the start states?

Given two NFAs $A_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ and $A_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$ satisfying $Q_1 \cap Q_2 = \emptyset$ and $q_0 \notin Q_1 \cup Q_2$, is the language of the NFA

$$(f(Q_1 \cup Q_2), \Sigma, f \circ \delta, q_0, f(F_1 \cup F_2)), \text{ where}$$

$$f(S) = (S \setminus \{ q_{01}, q_{02} \}) \cup \{ q_0 \mid q_{01} \in S \vee q_{02} \in S \},$$

$$\delta(s, a) = \begin{cases} \delta_1(q_{01}, a) \cup \delta_2(q_{02}, a), & \text{if } s = q_0, \\ \delta_1(s, a), & \text{if } s \in Q_1, \\ \delta_2(s, a), & \text{if } s \in Q_2 \end{cases}$$

equal to $L(A_1) \cup L(A_2)$?

1. Yes, always.
2. No, never.
3. No, not always, but sometimes.

Exponential blowup

Exponential blowup

Consider the following family of languages:

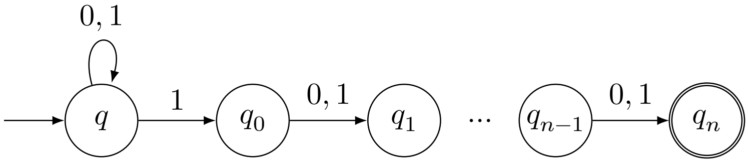
$$A \in \mathbb{N} \to \wp(\{\,0,1\,\}^*)$$
$$A(n) = \{\, u1v \mid u,v \in \{\,0,1\,\}^*, |v| = n \,\}$$

Exponential blowup

The family:

$$A(n) = \{\, u1v \mid u, v \in \{\, 0, 1 \,\}^*, |v| = n \,\}$$

For every $n \in \mathbb{N}$ the NFAs for $A(n)$ with the least number of states have at most $n + 2$ states:



Exponential blowup

Furthermore one can prove:

- For every $n \in \mathbb{N}$ the DFAs for $A(n)$ with the least number of states have at least 2^{n+1} states.

A key part of the proof in the course text book uses the pigeonhole principle:

- A DFA over $\{ 0, 1 \}$ with less than 2^k states has to end up in the same state for at least two distinct k-bit strings.

Exponential blowup

Thus it might be inefficient to check if a string belongs to a language represented by an NFA (or ε-NFA) by using the following method:

- Translate the NFA to a corresponding DFA.
- Use the DFA to check if the string belongs to the language.

Exponential blowup

- This method is used in practice by some tools.
- It seems to work fine in many practical cases.
- Exercise (optional):
  Make such a tool "blow up" by giving it a short piece of carefully crafted input.

Today

- ε-NFAs.
- ε-closure.
- Semantics.
- Constructions.
- Exponential blowup.

Next lecture

- Regular expressions.
- Translation from finite automata to regular expressions.