COMP4141 Theory of Computation

Lecture 2 Regular Languages, Finite Automata

Ron van der Meyden

CSE, UNSW

Revision: 2013/03/07

(Credits: David Dill, Thomas Wilke, Kai Engelhardt, Peter Höfner, Rob van Glabbeek)

Motivation

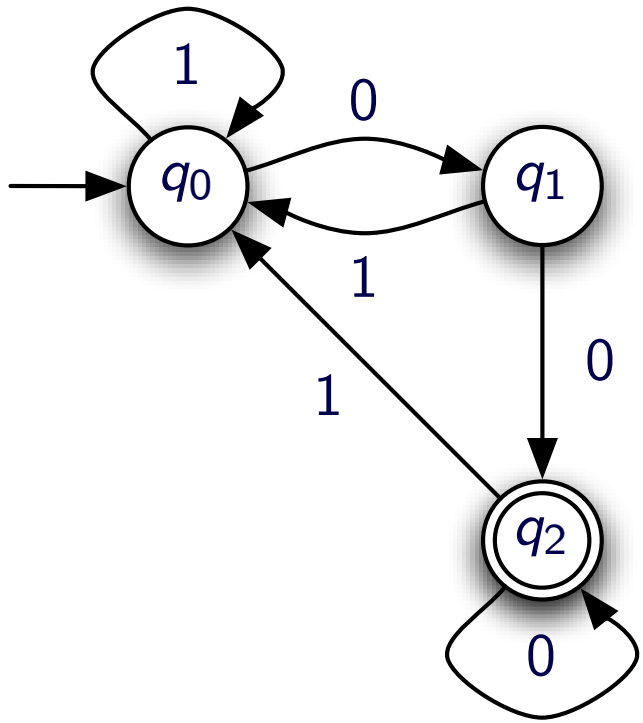
The next few lectures will be about *regular languages*.
Regular languages can be infinite, and can be described in several ways: by deterministic or non-deterministic finite automata, or by regular expressions. Regular languages are of great practical as well as theoretical importance.

- Widely used in applications
  (compilers, pattern matching, specification and formal verification of systems, TCP/IP, game AI).
- Constructions from automata theory are used in these applications.
- Robust — many representations are equivalent; they are closed under many operations.
- Basis for more sophisticated representations (automata on infinite strings, tree automata, timed automata).
- Answers to many questions about finite automata can be computed. (membership, emptiness, universality, subset, etc.)

Deterministic Finite Automata

A finite automaton is a mathematical model of a machine that reads strings and says "yes" or "no" for each string.

A *deterministic finite automaton* (*DFA*) defines a potentially infinite language: the set of strings for which it says "yes".



Deterministic Finite Automata

A DFA $D = (Q, \Sigma, q_0, \delta, F)$ consist of

- finite set Q of *states*,
- an *alphabet* Σ,
- a *start state* $q_0 \in Q$,
- a *next-state function* $\delta : Q \times \Sigma \longrightarrow Q$, and
- a set $F \subseteq Q$ of *final states*.

DFA-Example

The tuple for the example DFA above is:

$$Q = \{q_0, q_1, q_2\}$$

$$\Sigma = \{0, 1\}$$

$$q_0 = q_0$$

δ :

δ	0	1
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_0

$$F = \{q_2\}$$

We extend the next state function to work on whole strings.

Definition

Base:

$$\hat{\delta}(q, \epsilon) = q$$

Induction:

$$\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a)$$

Example

For our running example,

$$\hat{\delta}(q_1, \epsilon) = q_1$$
$$\hat{\delta}(q_0, 010101000) = q_2$$
$$\ldots$$

Language of a DFA

Definition

A DFA *accepts* a string $w \in \Sigma^*$ iff $\hat{\delta}(q_0, w) \in F$.

Definition

The *language* of a DFA A (written $L(A)$) is the set of strings accepted by A.

$$L(A) = \left\{ w \in \Sigma^* \;\middle|\; \hat{\delta}(q_0, w) \in F \right\}$$

Question: What is the language of the example DFA?

Non-determinism

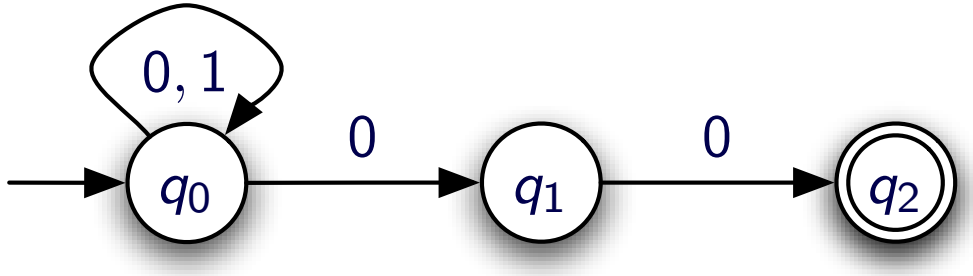
In a DFA, from each state

1. each input symbol leads to exactly one next state, and, therefore,
2. each input string leads to a unique state.

In a *non-deterministic finite automaton* (*NFA*),

1. there can be several possible next states for each input symbol (or none), and
2. an input string can lead to several states (or no states).

The string is accepted by an NFA if *at least one* of these states is accepting.



Essence of non-determinism

- Multiple alternative computations
  (often, huge numbers of them).
- Success if *any* succeed.

Different perspectives on this flavour of non-determinism:

- Massive parallel search.
- Search with "backtracking"
- Miraculously "guessing" the right option at each step.
- This is also known as *angelic* non-determinism in the literature.

Non-deterministic Finite Automata

An *NFA* is a 5-tuple $A = (Q, \Sigma, q_0, \delta, F)$.
Everything is the same as a DFA, except δ:

DFA: δ returns a unique state.

$$\delta : Q \times \Sigma \longrightarrow Q$$

NFA: δ returns a set of states.

$$\delta : Q \times \Sigma \longrightarrow 2^Q \ ,$$

where 2^Q is the powerset of Q, i.e. is the set $\{ S \mid S \subseteq Q \}$ of all subsets of Q.

NFA Acceptance

Definition (run of an NFA)

A *run* of an NFA A on a word $w = a_1 \dots a_k$ is a sequence $q_0 q_1 \dots q_k$ of states in Q such that

- q_0 is the initial state,
- for all $i = 1 \dots k$, we have $q_i \in \delta(q_{i-1}, a_i)$.

Such a run is *accepting* if the last state $q_k \in F$.

Definition (Language of an NFA)

$$L(A) = \{\ w \mid \text{ There exists an accepting run of } A \text{ on } w\ \}$$

Example

From the NFA A above, verify that the set of possible last states of runs of A on word 10100 is $\{q_0, q_1, q_2\}$

Is $10100 \in L(A)$? That is, from the start state, does the string lead to at least one final state?

NFAs vs. DFAs

Amazingly, an NFA is no more powerful than a DFA (although it can sometimes be a lot smaller).

(With other kinds of automata, the non-deterministic versions are sometimes more powerful than the deterministic ones.)

Theorem

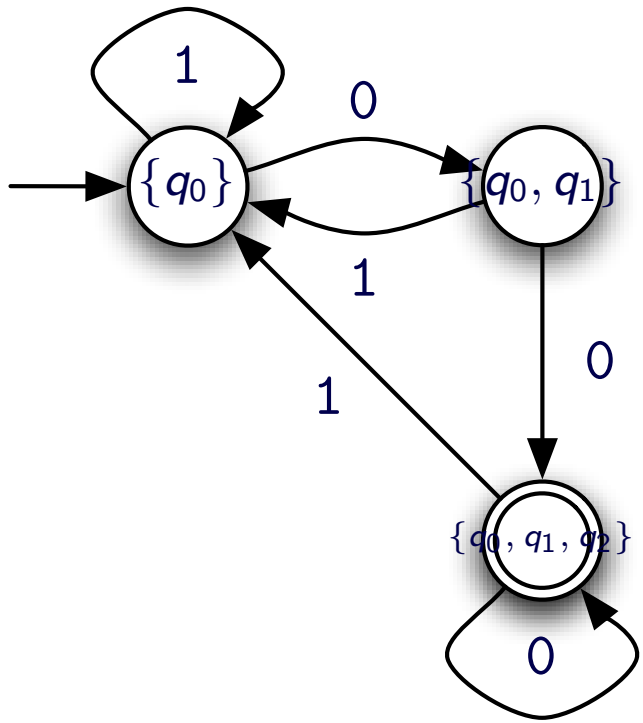
For every NFA N, there is a DFA D such that $L(N) = L(D)$.

The proof relies on the *subset construction*.

Subset Construction Example

The DFA produced by the subset construction *tracks* the set of states that the NFA can possibly be in (given the string read so far).
Hence, a state in the DFA = a set of NFA states.

Here is a tracking DFA for the above NFA:



The reasoning behind the construction was as follows:

. . . Suppose the original automaton was in one of the states in the set $\{q_0, q_1\}$, and we received a 0, what are all the states we can reach?

Clearly the answer is $\{q_0, q_1, q_2\}$.

Therefore there is an arrow in the DFA from the *DFA state* $\{q_0, q_1\}$ to the DFA state $\{q_0, q_1, q_2\}$.

Subset Construction

(Back to the proof of NFAs being no more powerful than DFAs.)
Given an NFA $N = (Q, \Sigma, q_0, \delta_N, F_N)$, construct a DFA

$$D = (2^Q, \Sigma, \{q_0\}, \delta_D, F_D)$$

such that $L(D) = L(N)$.

The key is in the definition of δ_D and F_D.

$$\delta_D(S, a) = \bigcup_{q \in S} \delta_N(q, a) \text{ , for each } S \subseteq Q.$$

The final states of D are the subsets that contain *at least one* final state of N:

$$F_D = \{ S \subseteq Q \mid S \cap F_N \neq \emptyset \} \quad .$$

Correctness of the subset construction

Theorem

If N is an NFA and D is the DFA obtained by applying subset construction to N then $L(N) = L(D)$.

But it's actually easier to prove a stronger claim:

Lemma

If N is an NFA and D is the DFA obtained by applying subset construction to N then $\hat{\delta}_D(\{q_0\}, w)$ is the set of all states q such that there exists a run $q_0 \ldots q$ of N on w, with last state q.

(I.e., the state you get to by running D on w is the same as the set of states you get to by running N on w.)
Reminder:
Base: $\hat{\delta}_D(q, \epsilon) = q$, **Induction:** $\hat{\delta}_D(q, wa) = \delta_D(\hat{\delta}_D(q, w), a)$

Correctness of the subset construction

Proof by induction on the length of w. Write $Last(w)$ for the set of last states of runs of N on w. We need to show $\hat{\delta}_D(\{q_0\}, w) = Last(w)$.

Base Case: $|w| = 0$, i.e. , $w = \epsilon$. Then the only run of N on w is q_0, so $Last(w) = \{q_0\}$. But $\hat{\delta}_D(\{q_0\}, \epsilon) = \{q_0\}$ by definition, so $\hat{\delta}_D(\{q_0\}, w) = Last(w)$.

Correctness of the subset construction (ctd)

Inductive case: Suppose that $\hat{\delta}_D(\{q_0\}, w) = \mathit{Last}(w)$. We show that, for $a \in \Sigma$, we have $\hat{\delta}_D(\{q_0\}, wa) = \mathit{Last}(wa)$. We prove this in two parts: $\hat{\delta}_D(\{q_0\}, wa) \subseteq \mathit{Last}(wa)$. and $\mathit{Last}(wa) \subseteq \hat{\delta}_D(\{q_0\}, wa)$.

To show $\hat{\delta}_D(\{q_0\}, wa) \subseteq \mathit{Last}(wa)$, suppose that $q' \in \hat{\delta}_D(\{q_0\}, wa)$. Then there exists $q \in \hat{\delta}_D(\{q_0\}, w)$ such that $q' \in \delta_N(q, a)$. By induction $\hat{\delta}_D(\{q_0\}, w) = \mathit{Last}(w)$. So there exists a run $q_0 \ldots q$ of N on w, ending in q. But then $q_0 \ldots qq'$ is a run of N on wa. So $q' \in \mathit{Last}(wa)$.

Correctness of the subset construction (ctd)

To show the converse containment $Last(wa) \subseteq \hat{\delta}_D(\{q_0\}, wa)$, suppose that $q' \in Last(wa)$. Then there exists a run $q_0 \dots q q'$ of N on word wa. The prefix $q_0 \dots q$ is a run of N on w, and the last step corresponds to input a. Thus, $q \in Last(w)$ and $q' \in \delta_N(q, a)$. By induction $q \in \hat{\delta}_D(\{q_0\}, w)$. It now follows from the definition of δ_D that $q' \in \hat{\delta}_D(\{q_0\}, wa)$.

Equivalence of NFAs and DFAs

Theorem

A language L is $L(N)$ for some NFA N iff it is $L(D)$ for some DFA D.

Proof.

The subset construction was the hard part of this: Given any NFA, a DFA accepting the same language can be constructed. So, whenever a language is accepted by an NFA, it is also accepted by some DFA.

The other direction of the proof is obvious. Every DFA is essentially an NFA. The only change is that $\delta_N(q, a) = \{\delta_D(q, a)\}$. All other aspects of the FA remain the same, including F. □

Subset Construction Efficiency

In the worst case, the subset construction generates $2^{|Q|}$ states.

In practice, it is often unnecessary to generate them all, since few are actually reachable by any input string.

A more efficient procedure: Generate the subset *on the fly*. Create them as you trace through the NFA. That way, you only generate the *accessible* subsets.

Example

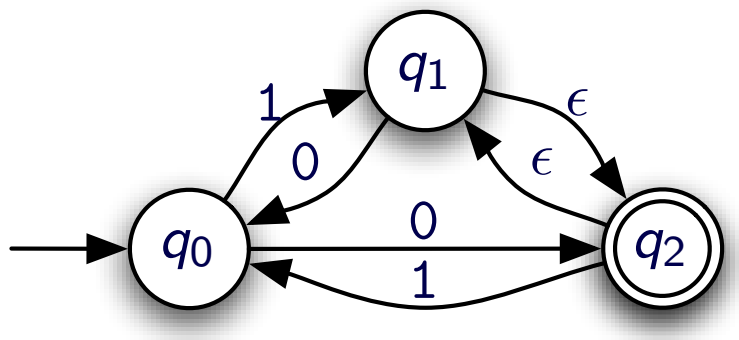
Notice that in our subset construction example the DFA states for the sets $\{q_0, q_2\}$ and the $\emptyset$ are absent in the final DFA obtained.

Clearly, we used the efficient procedure to compute it!

ϵ-NFA

Idea: Allow state changes that don't consume input symbols ("silent moves").

Choice of whether to take ϵ-transition is non-deterministic.



The input "001" is accepted. A possible run is $q_0 q_2 q_1 q_0 q_1 q_2$.

Definition of ϵ-NFA

The only difference in the mathematical definition is in δ:

$\delta : Q \times (\Sigma \cup \{\epsilon\}) \longrightarrow 2^Q$

NB

Sipser's notation sometimes varies from ours.

[Sipser06]	**COMP4141**
NFA	ϵ-*NFA*
$\mathcal{P}(Q)$	2^Q

Acceptance by an ϵ-NFA

Definition (run of an NFA)

A *run* of an NFA A on a word w is a sequence $q_0 q_1 \dots q_k$ of states in Q such that we can write $w = a_1 \dots a_k$ with each $a_i \in \Sigma \cup \{\epsilon\}$ and

- q_0 is the initial state,
- for all $i = 1 \dots k$, we have $q_i \in \delta(q_{i-1}, a_i)$.

Such a run is *accepting* if the last state $q_k \in F$.

This is almost the same as the NFA definition, but note that since $\epsilon u = u\epsilon = u$, for $w = 01$, we can write $w = \epsilon\epsilon 0 \epsilon 1 \epsilon$, and there are (infinitely) many other ways to insert ϵ steps.

Definition (Language of an NFA)

$$L(A) = \{\ w \mid \text{There exists an accepting run of } A \text{ on } w\ \}$$

ϵ-closure

Preliminary: the *ϵ-closure* $E(q)$ of state q is the set of all states reachable from q by silent moves.

$$E(q_0) = \{q_0\}$$
$$E(q_1) = \{q_1, q_2\}$$
$$E(q_2) = \{q_1, q_2\}$$

Definition of E

[S is the *least* set satisfying property P if $P(S)$, and whenever $P(S')$ holds, $S \subseteq S'$.]

Definition

$E(q)$ is the least set satisfying:

- $q \in E(q)$
- $\delta(s, \epsilon) \subseteq E(q)$ for all $s \in E(q)$

Also, $E(S) = \bigcup \{ E(q) \mid q \in S \}$.

("Extend to sets" — this is overloading)

Converting an ϵ-NFA to a DFA

Theorem

For every ϵ-NFA N, there exists an DFA D such that $L(D) = L(N)$.

Proof.

The proof uses a modified subset construction, where every subset is ϵ-closed.

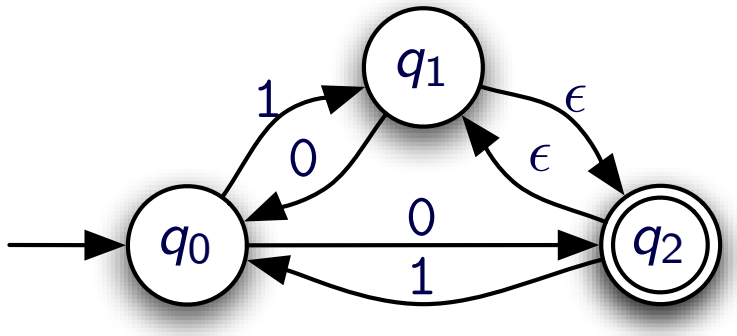
Given $N = (Q, \Sigma, q_0, \delta_N, F_N)$, we can define $D = (2^Q, \Sigma, E(q_0), \delta_D, F_D)$ as follows:

$$F_D = \{\ S \subseteq Q \mid\ S \cap F_N \neq \emptyset\ \}$$

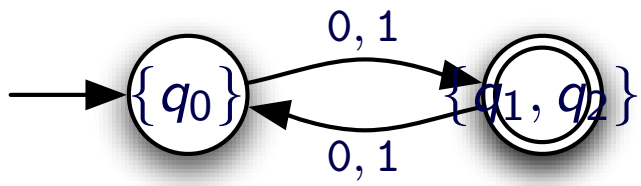
$$\delta_D(S, a) = E(\bigcup\nolimits_{s \in S} \delta_N(s, a))$$

□

ϵ-elimination example



converts to the DFA:



ϵ-elimination Proof

As with NFA's $Last(w)$ is the set of all q such that q is the last state of a run of N on w.

Lemma

If D is the DFA obtained from the ϵ-NFA N by the ϵ-elimination, then $\hat{\delta}_D(E(q_0), x) = Last(x)$ for all $x \in \Sigma^*$.

Theorem: $L(N) = L(D)$**.**

The proof is as in equivalence of NFAs and DFAs.

Summary

DFAs, NFAs, and ϵ-NFAs all have equal expressive power.

I.e., given a language L, if one kind of FA accepts L, then there are acceptors for L in the other kinds of FA, too.