Nondeterminism

CSCI 2210

2023-09-18 and 2023-09-20

Union Language Revisited

Last time we constructed a DFA for the union of regular languages.

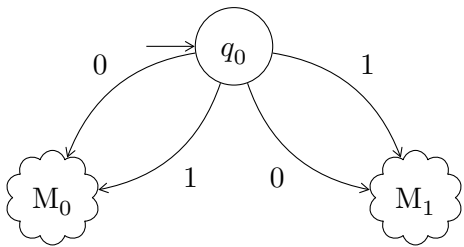
It took some of work to encode pairs of states as states,
and pairs of transitions as transitions.

What if we could just *guess* whether $w \in \mathrm{L}(\mathrm{M}_0)$ or $w \in \mathrm{L}(\mathrm{M}_1)$ and then run the right machine.

Determinism

The behavior of a DFA is *deterministic* because the state transition function $\delta : Q \times \Sigma \to Q$ prescribes *exactly one* next state for each current state and input symbol.

To solve the union language problem, it would suffice to have two possible transitions for each symbol out of the start state.



We will ask for such *nondeterministic transition* not just for the start state, but for all states.

deterministic Finite Automata

A **deterministic finite automaton** ("DFA") M has the following components:

input alphabet: a nonempty finite set of symbols Σ,

state set: a nonempty finite set of states Q,

start state: a chosen state $q_0 \in \mathrm{Q}$,

accept state set: a chosen subset of states $\mathrm{F} \subseteq \mathrm{Q}$,

state transition function: a function $\delta : \mathrm{Q} \times \Sigma \to \mathrm{Q}$.

Nondeterministic Finite Automata

A **nondeterministic finite automaton** ("NFA") M has the following components:

input alphabet: a nonempty finite set of symbols Σ,

state set: a nonempty finite set of states Q,

start state: a chosen state $q_0 \in \mathrm{Q}$,

accept state set: a chosen subset of states $\mathrm{F} \subseteq \mathrm{Q}$,

state transition relation: a relation $\Delta : \mathrm{Q} \times \Sigma \nrightarrow \mathrm{Q}$.

NFA State Transition Graph

We can represent an NFA M using a state transition graph.

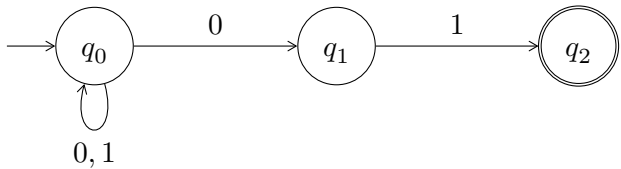
The vertex set is Q.

For vertices $q_a, q_b \in \mathrm{Q}$ and alphabet symbol $s \in \Sigma$ there is an edge $s : q_a \rightarrow q_b$ just in case $\Delta\,((q_a\,,\,s) \dashrightarrow q_b)$.

In a diagram for this graph we annotate the vertex q_0 and those of F.

Example: NFA State Transition Graph Diagram

Let $\mathrm{M}_{\_01}$ be the NFA specified by:



This means:

- $\Sigma = \{0, 1\}$,
- $\mathrm{Q} = \{q_0, q_1, q_2\}$,
- $q_0 = q_0$,
- $\mathrm{F} = \{q_2\}$,
- $\Delta =$

	q_0	q_1	q_2
q_0	$\{0, 1\}$	$\{0\}$	$\{\,\}$
q_1	$\{\,\}$	$\{\,\}$	$\{1\}$
q_2	$\{\,\}$	$\{\,\}$	$\{\,\}$

Language of an NFA: Operational Semantics

Each NFA M over alphabet Σ determines a **language** $\mathrm{L}(\mathrm{M})$ according to the following **operational semantics**.

Given an input string $w = [w_0, w_1, \cdots, w_n] \in \Sigma^*$:

- M begins in the start state q_0,
- M inspects the symbols of w left-to-right one at a time,
- If M is in state q_a inspecting symbol w_i then M *may* transition to state q_b if $\Delta\left((q_a\ , w_i) \dashrightarrow q_b\right)$,
- Upon exhausting w, there is a *set of states* Q_f that M may be in,
- We say $w \in \mathrm{L}(\mathrm{M})$ just in case $\mathrm{Q}_f \cap \mathrm{F} \neq \emptyset$.

Language of an NFA: State Transition Graph Interpretation

For a NFA M, each word $w \in \Sigma^*$ determines a *set of paths* $\{p_k : q_0 \to q_{f_k}\}$ in the state transition graph $\mathrm{G}(\mathrm{M})$.

The state transition *relation* Δ determines where these paths go at each step.

At the ith symbol in w, i.e. w_i, we have a *set* of length i paths in $\mathrm{G}(\mathrm{M})$ all with source q_0 and edges labeled $[w_0, w_1, \cdots, w_{i-1}]$.

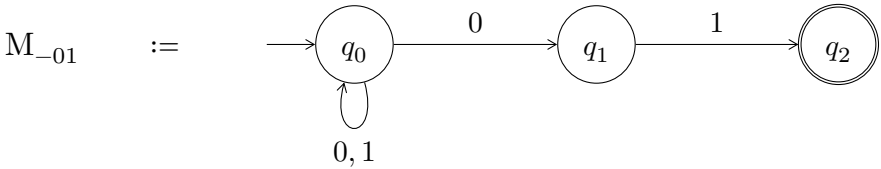
Let $p_k : q_0 \to q_k$ be such a path. For each edge $w_i : q_k \to q_l$ we make the length $i+1$ path $p_k +\!\!+ [w_i]$.

We take the *union of all paths* obtained by extending each path p_k by each w_i-labeled edge.

A path $q_0 \to q_f$ constructed in this way is called a **run** of M.

NFA M accepts word w just in case there is an accepting run of M labeled by w.

Example: Language of M_{-01}



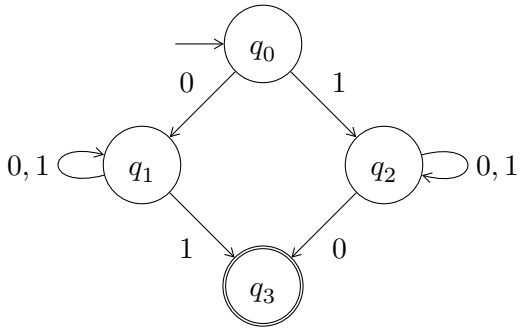
determines the language of words with suffix 01.

for example:

- word 01 is accepted by run $[(q_0), 0, (q_1), 1, (q_2)]$
- word 0101 is accepted by run $[(q_0), 0, (q_0), 1, (q_0), 0, (q_1), 1, (q_2)]$

Activity: Language of an NFA

Verify that the following state transition graph represents an NFA over alphabet $\Sigma := \{0, 1\}$ and give a description of its language.



NFA Variations

There are several common variations of NFAs.

Fortunately, they are all equivalent in that they can be translated into one another.

Variation: NFA with Transition Function

Instead of a *state transition relation*:

$$\Delta : \mathrm{Q} \times \Sigma \nrightarrow \mathrm{Q}$$

we can specify a set-valued *state transition function*:

$$\delta : \mathrm{Q} \times \Sigma \to \wp(\mathrm{Q})$$

This is just the “powerset trick” we learned in week 1:

$$\Delta\,((q\,,\,s) \dashrightarrow q') \quad \text{just in case} \quad q' \in \delta(q\,,\,s)$$

Variation: NFA with Start State Set

Instead of a single *start state* q_0

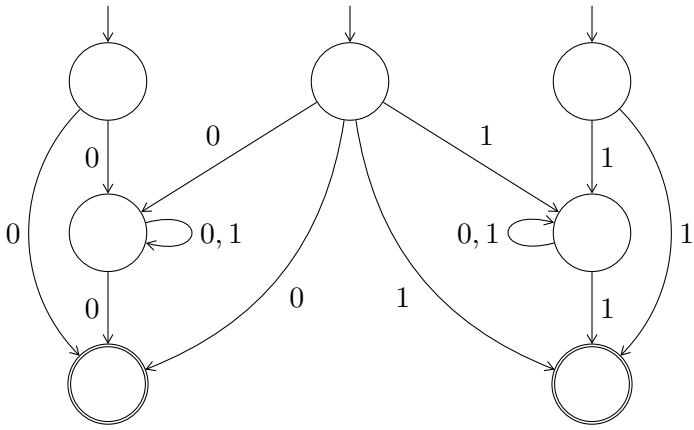
we can specify a *nonempty set of start states* I

Such a machine accepts a word w just in case there is a run $q_i \to q_f$
with $q_i \in \mathrm{I}$ and $q_f \in \mathrm{F}$.

We can translate such a machine back to our original setting by
forgetting that the states in I are start states and adding a new start state q_0
with an edge $s : q_0 \to q_j$ just in case $\exists\, q_i \in \mathrm{I}\ .\ s : q_i \to q_j$.

The new start state is an accept state if any of the original start states were.

Example: NFA with Start State Set



Variation: NFA with String-Labeled Edges

Instead of transitions that act on symbols:

$$\Delta : Q \times \Sigma \nrightarrow Q$$

we can specify transitions that act on strings:

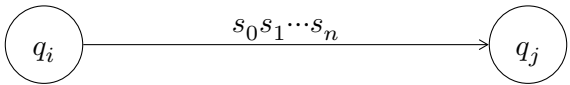
$$\Delta' : Q \times \Sigma^* \nrightarrow Q$$

We translate this back to our original setting as follows.

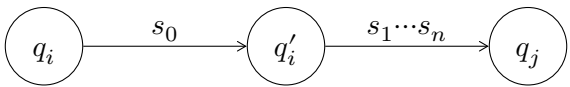
Variation: NFA with String-Labeled Edges of Length > 1

For $|s| > 1$,

replace edge:



with edges:

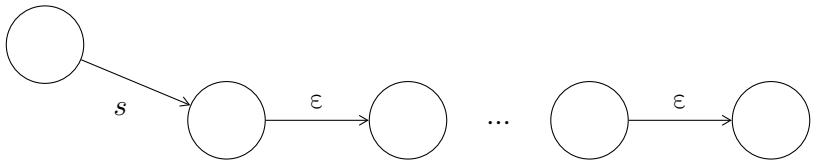


until all labels have $|s| = 1$.

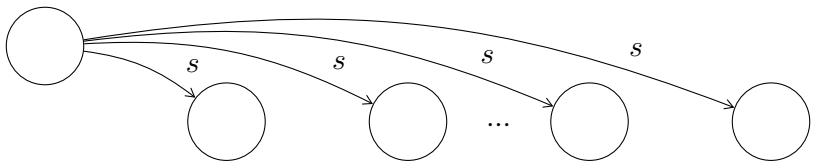
Variation: NFA with String-Labeled Edges of Length < 1

If $|s| = 0$ we call this an empty-string transition or ε**-transition**.

replace:



with:



and ε-transitions from q_0 with new start nodes, then eliminate multiple start nodes.

NFA Language Constructions

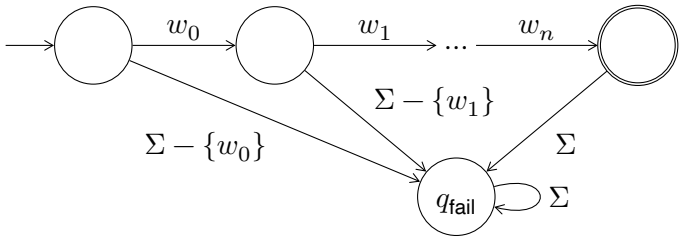
Clearly, every language decidable by a DFA is decidable by an NFA, because a DFA is just an NFA where the state transition relation is single-valued and total (i.e. there is always *exactly one* possible transition).

So the following languages are decidable by NFAs:

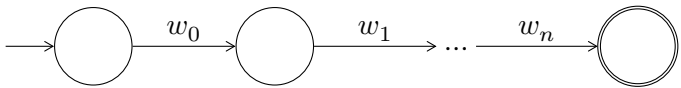
- the empty language
- all singleton languages
- the union of DFA-decidable languages
- the intersection of DFA-decidable languages (homework)

Singleton Languages Revisited

With DFAs we needed a “dump state” and lots of failure transitions to account for falling off of the “happy path”.



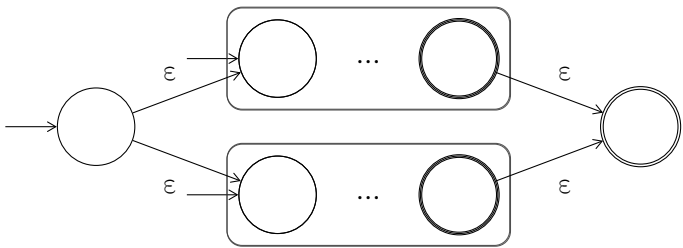
With NFAs we only need the happy path:



Union Languages Revisited

With DFAs we needed a **cartesian product construction** to build a DFA whose states and transitions were ordered pairs of those of the input machines.

With NFAs we can either use multiple start states, or else use ε-transitions to nondeterministically **fork** to the start states of the input machines and **join** from their accept states:



WLOG, we can assume an ε-NFA has one accept state (homework).

Concatenated Languages

For languages L_0 and L_1, the **concatenated language** $\mathrm{L}_0 +\!\!+ \mathrm{L}_1$ is defined as

$$\mathrm{L}_0 +\!\!+ \mathrm{L}_1 := \{w_0 +\!\!+ w_1 \mid w_0 \in \mathrm{L}_0 \text{ and } w_1 \in \mathrm{L}_1\}$$

E.g. if $\mathrm{L}_0 = \{\text{low}, \text{slow}\}$ and $\mathrm{L}_1 = \{\text{ly}, \text{-down}\}$ then

$\mathrm{L}_0 +\!\!+ \mathrm{L}_1 =$

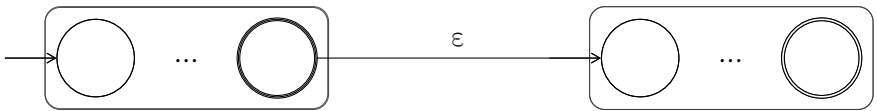
	ly	-down
low	lowly	low-down
slow	slowly	slow-down

Concatenation Closure

Let $\mathrm{L}_0 := \mathrm{L}(\mathrm{M}_0)$ and $\mathrm{L}_1 := \mathrm{L}(\mathrm{M}_1)$ be the languages decided by two NFAs.

Then $\mathrm{L}_0 +\!\!+ \mathrm{L}_1$ is decided by an ε-NFA.

We just add ε-transitions from the accept state(s) of M_0 to the start state(s) of M_1:



Iterated Languages

For language L, the **iterated language** L^* is given by the recursive definition,

$$\mathrm{L}^* := \mathrm{L}_\epsilon \cup (\mathrm{L} \shuffle \mathrm{L}^*)$$

Expanding this, we see

$$\mathrm{L}^* = \mathrm{L}_\epsilon \cup \mathrm{L} \cup (\mathrm{L} \shuffle \mathrm{L}) \cup (\mathrm{L} \shuffle \mathrm{L} \shuffle \mathrm{L}) \cup \cdots$$

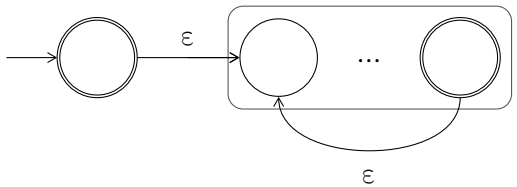
So the words of L^* are concatenations of all finite sequences of words of L (including the empty one).

Iteration Closure

If L be the languages decided an NFA M then L^* is decided by an ε-NFA.

We add:

- a new accepting start state (for sequences of <1 L-words),
- an ε-transition from the new to the old start state (for sequences of $=1$ L-word)
- ε-transitions from the old accept state(s) to the old start state (for sequences of >1 L-words)



Equivalence of NFAs and DFAs

NFAs are easier to work with than DFAs. Intuitively, they seem more powerful.

But in fact they are equivalent in power:

Theorem

A language is decidable by an NFA just in case it is decidable by a DFA.

One direction is easy: every DFA can be regarded as an NFA that happens to be deterministic.

The main idea for the other direction is the **powerset construction** (synonym "subset construction").

Cartesian Product Construction Revisited

Before describing this construction, let's recall how we built a DFA for the union of two languages.

We needed to build a DFA that could simulate *a pair of DFAs*.

We did this using the **cartesian product**:

states were ordered pairs of states,

transitions were ordered pairs of transitions.

In contrast, simulating an NFA is equivalent simulating *a set of DFAs*: one DFA for each nondeterministic choice made by the NFA.

This set of DFAs can't be infinitely large: its size is bounded by the number of possible subsets of states.

Powerset Construction

Let M be a ε-free NFA $(\Sigma\,,\,\mathrm{Q}\,,\,q_0\,,\,\mathrm{F}\,,\,\Delta)$. We make the DFA $\mathrm{D}(\mathrm{M})$ with:

input alphabet: Σ

state set: $\wp(\mathrm{Q})$

start state: $\{q_0\}$

accept state set: $\{\mathrm{S} \subseteq \mathrm{Q} \mid \mathrm{S} \cap \mathrm{F} \neq \emptyset\}$

state transition function: We construct $\delta : \wp(\mathrm{Q}) \times \Sigma \to \wp(\mathrm{Q})$
from $\Delta : \mathrm{Q} \times \Sigma \nrightarrow \mathrm{Q}$ as follows:
by the *powerset trick* the relation Δ is equivalent to a function
$\delta' : \mathrm{Q} \times \Sigma \to \wp(\mathrm{Q})$.
We define $\delta(\mathrm{T}\,,\,s) := \bigcup_{q \in \mathrm{T}} (\delta'(q\,,\,s))$.