Computer Science 351

Introduction to Nondeterministic Finite Automata

Instructor: Wayne Eberly

Department of Computer Science
University of Calgary

Lecture #5

Goals for Today

Goals for Today:

- Introduce ***nondeterministic finite automata*** — contrasting these with "deterministic finite automata".
- Present two ways to see how a nondeterministic finite automaton processes a string.

Note: The notion of ***nondeterminism*** will be extremely important later on in this course, and in CPSC 413.

Nondeterministic Finite Automata

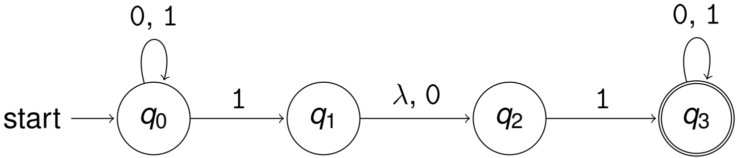
Suppose we "changed the rules" used to provide transition functions for finite automata:

- "λ-transitions" are introduced, allowing the machine to move from one state to another without processing any symbols in the input string at all, and
- the machine is allowed to move from a given state to *zero*, *one*, or *many* states when a symbol $\sigma \in \Sigma$ or λ is processed.

The resulting "finite-state machines" — now called ***nondeterministic finite automata*** — could look like the pictures shown on the next two slides.

First Example of an NFA

Input alphabet: $\Sigma_1 = \{0, 1\}$; NFA M_1 is as follows.



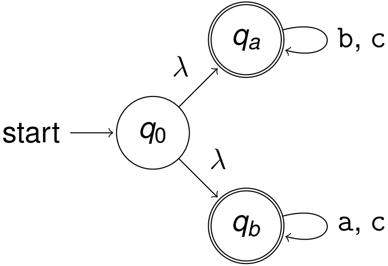
Note: When processing an initial 1 this machine can

- Follow a transition in order to stay in the start state, q_0; or...
- Follow a *different* transition in order to move to state q_1; or...
- Do the above and then use a λ-transition after that to move to state q_2.

So *three* states — q_0, q_1 and q_2 — can be reached from the start state by processing 1.

Second Example of an NFA

Input alphabet: $\Sigma_2 = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$; NFA M_2 is as follows.



Second Example of an NFA

Note: When processing an initial 'a', this machine...

- ...can try to follow a transition for 'a' out of the start state — but this doesn't work, because there is no such transition!... or
- ...can follow a λ transition to the state q_b and then follow a transition for 'a' to stay in state q_b... or
- ... can *try* to follow a λ transition to q_a and then follow a transition for 'a' out or q_a — but this doesn't work either, because no such transition exists!

So the *only* state that can be reached from the start state by processing 'a' is q_b.

Processing of Strings

In a way, nondeterministic finite automata are like

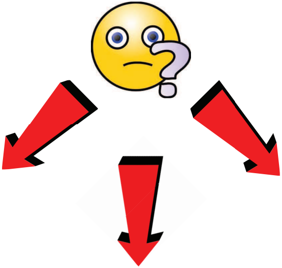
Magic!

Processing of Strings

When a ***deterministic finite automaton*** is used to process a string there is always ***exactly one*** transition that can be followed to process a symbol.

Processing of Strings

On the other hand, when a ***nondeterministic finite automaton*** is used to process a string there may be zero, one, or ***many*** transitions that you might choose.



Furthermore, if the nondeterministic finite automaton includes λ-transitions then it is possible to use one in order to change state, without processing any symbols at all!

Processing of Strings

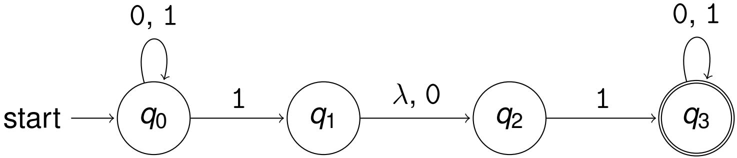
One can think about processing an input string by

guessing your way toward an accepting state....

because the string should be accepted as long as there is ***at least way*** to do this, so that an accepting state has been reached when the entire string has been processed.

Processing of Strings

Consider, again, the first NFA M_1 shown above:



When processing the string 11, one can do the following:

- Use the transition for symbol 1 from state q_0 to state q_1, in order to process the first 1 and reach state q_1;
- follow the λ-transition from q_1 to q_2 in order to reach q_2;
- follow the transition for symbol 1 from q_2 to q_3 in order to process the final 1 in the input string and reach q_3.

Since q_3 is an accepting state it follows that this NFA ***accepts*** the string 11.

Processing of Strings

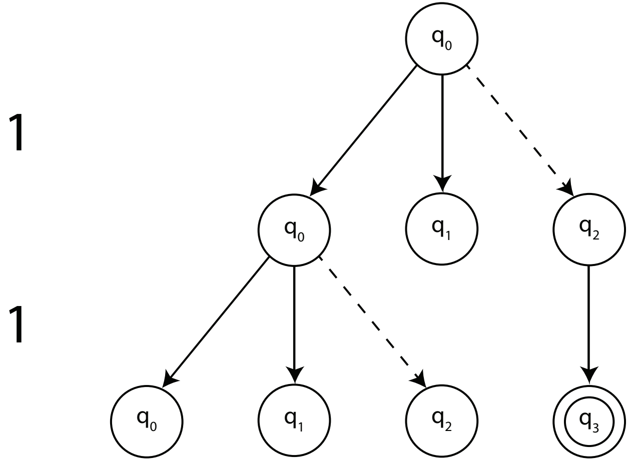
Another — more useful — way to think about how a nondeterministic finite automaton processes a string is to keep track of ***all*** the states that can be reached as symbols are processed.

One reference, *Introduction to the Theory of Computation*, uses "trees of possibilities" to show the states that can be reached when processing strings.

Examples are given — and explained — on the next few slides:

- The first of these displays the use of NFA M_1 to process the string `11`.
- The second of these displays the use of NFA M_2 to process the string `ca`.

Processing of Strings — Use of M_1 to process 11



Processing of Strings — Use of M_1 to process 11

Explanation of This Picture:

- q_0 is the start state, and there are no λ-transitions out of q_0, so q_0 is the only state that is reachable before any symbols are processed. Thus q_0 is shown, all by itself, at the top of the picture.

- As previously noted, a transition can be followed to move from q_0 to itself when processing a 1. A transition can also be followed to move from q_0 to q_1 when processing a 1. Finally, since there is a λ-transition from q_1 to q_2, you can get from q_0 to q_2 when processing a 1 by using the transition (for 1) from q_0 to q_1 and then following the λ-transition from q_1 to q_2.

  It is not possible to use λ-transitions to go any further, so the states reachable from q_0 when processing the first 1 are q_0, q_1, and q_2.

Processing of Strings — Use of M_1 to process 11

- Therefore, q_0, q_1 and q_2 are all shown at the next level of the picture (as "children" of q_0); a dashed line is being drawn between q_0 and q_2 to show that a λ-transition was also used in this case.
- A 1 is drawn to to the left of these levels, centred between them, to show that a 1 was processed.
- The next symbol to be processed was also a 1. Since q_0, q_1 and q_2 can all be reached from q_0 when processing *this* symbol as well, these are all shown (in the same way) at the next level, as children of q_0.

Processing of Strings — Use of M_1 to process `11`

- There are no transitions for `1` out of q_1 — and all the states that are reachable from q_1 by following λ-transitions are also included on the same level of the tree as q_1, so that we do not need to worry about them.

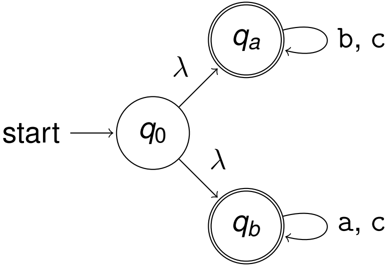
  Thus no children of q_1 are shown.

- A transition for `1` can be used to move from q_2 to q_3, and there are no λ-transitions that can be used to go farther, so q_3 is shown in the picture as the only child of q_2.

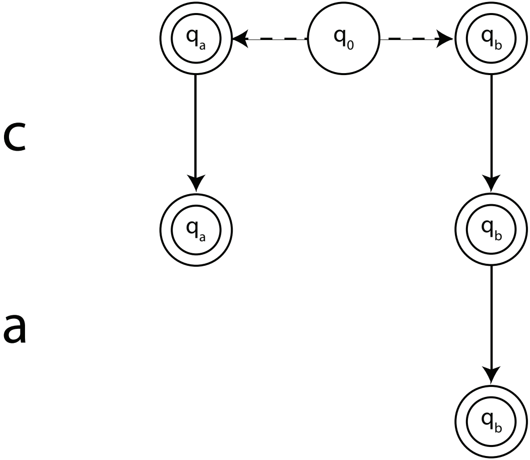
- The ***set*** of states that are reachable from the start state, q_0, by processing `11` are the ones shown (at least once) at the bottom level of picture — that is, $\{q_0, q_1, q_2, q_3\}$.

Processing of Strings — Use of M_2 to process `ca`

Consider the nondeterministic finite automaton M_2:



Processing of Strings — Use of M_2 to process `ca`



Processing of Strings — Use of M_2 to process `ca`

- There is only *one* significant difference between this example and the previous one: There are λ-transitions out of the start state, so that more than one state is reachable before any of the input symbols get processed.

  In particular, q_a and q_b are both reachable from the start state, q_0, by λ-transitions, so they are reachable before any symbols in an input string are processed.

  The rest of this possibility tree is created in the same way as the previous one is. Since q_b is the only state at its bottom level, we can conclude from this that the set of states that can be reached after processing `ca` is the set $\{q_b\}$.

Summary of a Process

To determine the set of states that are reachable by processing a string

$$\omega = \omega_1\omega_2 \ldots \omega_n \in \Sigma^\star$$

1. Create a set S_λ by including all the states reachable from the start state, q_0, by following zero or more λ transitions.
2. `for` $i = 1, 2, \ldots, n$
   - Initialize $S_{\omega_1\omega_2\ldots\omega_i}$ to be $\emptyset$
   - `for` every state $q \in S_{\omega_1\omega_2\ldots\omega_{i-1}}$, add, to $S_{\omega_1\omega_2\ldots\omega_i}$, every state r that is reachable from q by following a transition (from q) for the symbol $\omega_i \in \Sigma$, and then following zero or more λ-transitions after that.
3. The set of states that are reachable from q_0 by processing the above string ω is the set $S_\omega = S_{\omega_1\omega_2\ldots\omega_n}$.

Acceptance of a String

A nondeterministic finite automaton M ***accepts*** a string $\omega \in \Sigma^\star$ if and only if the set of states that are reachable from the start state, q_0, by processing the string ω includes *one or more* accepting states $q \in F$.

- Thus M_1 accepts the string `11` because q_3 is an accepting state and it can be reached from q_0 by processing this string.
- Thus M_2 accepts the string `ca` because q_b is an accepting state and it can be reached from q_0 by processing *this* string.

Formal Definition of an NFA

Definition: Suppose S is a finite set. Then the ***power set*** of S, $\mathcal{P}(S)$, is the set of all ***subsets*** of S. For example, if

$$S = \{\mathrm{x}, \mathrm{y}, \mathrm{z}\}$$

then $\mathcal{P}(S)$ includes the following eight sets:

- $\emptyset$ (the empty set);
- $\{\mathrm{x}\}$;
- $\{\mathrm{y}\}$;
- $\{\mathrm{z}\}$;
- $\{\mathrm{x}, \mathrm{y}\}$;
- $\{\mathrm{x}, \mathrm{z}\}$;
- $\{\mathrm{y}, \mathrm{z}\}$;
- $\{\mathrm{x}, \mathrm{y}, \mathrm{z}\}$.

Formal Definition of an NFA

Definition: If Σ is an alphabet, including *k* symbols, then Σ_λ is a set of size $k + 1$ including all the symbols in Σ as well as the empty string. For example, if $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ then Σ_λ includes the following:

- $\mathtt{a}$;
- $\mathtt{b}$;
- $\mathtt{c}$;
- λ.

Formal Definition of an NFA

Definition: A ***nondeterministic finite automaton*** is 5-tuple

$$(Q, \Sigma, \delta, q_0, F),$$

where

1. Q is a finite (and nonempty) set of ***states***,
2. Σ is a finite (and nonempty) ***alphabet***,
3. $\delta : Q \times \Sigma_\lambda \to \mathcal{P}(Q)$ is the ***transition function***,
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the set of ***accept states***.

For $q \in Q$ and $\sigma \in \Sigma_\lambda$, $\delta(q, \sigma)$ is the *set* of states that can be reached by following a ***single*** transition for σ out of q.

Formal Definition of an NFA

The NFA M_1 can be formally modelled as $M_1 = (Q, \Sigma, \delta, q_0, F)$ where

1. $Q = \{q_0, q_1, q_2, q_3\}$;
2. $\Sigma = \Sigma_1 = \{0, 1\}$;
3. The transition function $\delta : Q \times \Sigma_\lambda \to \mathcal{P}(Q)$ is shown in the table on the following slide;
4. q_0 is the start state;
5. $F = \{q_3\}$.

Formal Definition of an NFA

A table describing the transition function δ is as follows.

	0	1	λ
q_0	$\{q_0\}$	$\{q_0, q_1\}$	$\emptyset$
q_1	$\{q_2\}$	$\emptyset$	$\{q_2\}$
q_2	$\emptyset$	$\{q_3\}$	$\emptyset$
q_3	$\{q_3\}$	$\{q_3\}$	$\emptyset$

Formal Definition of an NFA

The NFA M_2 can be formally modelled as $M_2 = (Q, \Sigma, \delta, q_0, F)$ where

1. $Q = \{q_0, q_a, q_b\}$;
2. $\Sigma = \Sigma_2 = \{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$;
3. The transition function $\delta : Q \times \Sigma_\lambda \to \mathcal{P}(Q)$ is shown in the table on the following slide;
4. q_0 is the start state;
5. $F = \{q_a, q_b\}$.

Formal Definition of an NFA

A table describing the transition function δ is as follows.

	a	b	c	λ
q_0	$\emptyset$	$\emptyset$	$\emptyset$	$\{q_a, q_b\}$
q_a	$\emptyset$	$\{q_a\}$	$\{q_a\}$	$\emptyset$
q_b	$\{q_b\}$	$\emptyset$	$\{q_b\}$	$\emptyset$

Formal Definition of an NFA

Consider a function $Cl_\lambda : Q \to \mathcal{P}(Q)$: For $q \in Q$, $Cl_\lambda(q)$ is the set of states reachable from q by following zero or more λ-transitions.

- In Example #1
  - $Cl_\lambda(q_0) = \{q_0\}$;
  - $Cl_\lambda(q_1) = \{q_1, q_2\}$;
  - $Cl_\lambda(q_2) = \{q_2\}$;
  - $Cl_\lambda(q_3) = \{q_3\}$.
- In Example #2
  - $Cl_\lambda(q_0) = \{q_0, q_a, q_b\}$;
  - $Cl_\lambda(q_a) = \{q_a\}$;
  - $Cl_\lambda(q_b) = \{q_b\}$.

$Cl_\lambda(q)$ is sometimes called the ***λ-closure*** of the state q.

Formal Definition of an NFA

It is now possible to define an ***extended transition function*** $\delta^\star : Q \times \Sigma^\star \to \mathcal{P}(Q)$: For each state $q \in Q$ and each string $\omega \in \Sigma^\star$, $\delta^\star(q, \omega)$ is the set of states that can be reached from q by processing the string ω.

This can be "formally defined" as follows:

- For every state $q \in Q$, $\delta^\star(q, \lambda) = Cl_\lambda(q)$.
- For every state $q \in Q$, every string $\omega \in \Sigma^\star$, and every symbol $\sigma \in \Sigma$,

$$\delta^\star(q, \omega \cdot \sigma) = \bigcup_{r \in \delta^\star(q,\omega)} \left(\bigcup_{s \in \delta(r,\sigma)} Cl_\lambda(s) \right).$$

Application: Evaluation of $\delta^\star$

Suppose we wish to evaluate $\delta^\star(q_0, 11)$. Setting $\omega = 1$ and $\sigma = 1$ the second part of the above definition implies that

$$\delta^\star(q_0, 11) = \bigcup_{r \in \delta^\star(q_0, 1)} \left(\bigcup_{s \in \delta(r, 1)} Cl_\lambda(s) \right). \qquad (1)$$

Setting $\omega = \lambda$ and $\sigma = 1$, the second part of this definition implies that

$$\delta^\star(q_0, 1) = \bigcup_{r \in \delta^\star(q_0, \lambda)} \left(\bigcup_{s \in \delta(r, 1)} Cl_\lambda(s) \right). \qquad (2)$$

Application: Evaluation of $\delta^\star$

The first part of the definition implies that

$$\delta^\star(q_0, \lambda) = Cl_\lambda(q_0) = \{q_0\}. \tag{3}$$

Applying equations (2) and (3), it now follows that

$$\begin{aligned} \delta^\star(q_0, 1) &= \bigcup_{s \in \delta(q_0, 1)} Cl_\lambda(s) \\ &= Cl_\lambda(q_0) \cup Cl_\lambda(q_1) \\ &= \{q_0\} \cup \{q_1, q_2\} \\ &= \{q_0, q_1, q_2\}. \end{aligned}$$

Application: Evaluation of $\delta^\star$

Applying this along with equation (1), it now follows that

$$\begin{aligned}
\delta^\star(q_0, 11) &= \bigcup_{r \in \{q_0, q_1, q_2\}} \left(\bigcup_{s \in \delta(r,1)} Cl_\lambda(s) \right) \\
&= \bigcup_{s \in \delta(q_0,1)} Cl_\lambda(s) \;\cup\; \bigcup_{s \in \delta(q_1,1)} Cl_\lambda(s) \;\cup\; \bigcup_{s \in \delta(q_2,1)} Cl_\lambda(s) \\
&= (Cl_\lambda(q_0) \cup Cl_\lambda(q_1)) \;\cup\; \emptyset \;\cup\; Cl_\lambda(q_3) \\
&= Cl_\lambda(q_0) \cup Cl_\lambda(q_1) \cup Cl_\lambda(q_3) \\
&= \{q_0\} \cup \{q_1, q_2\} \cup \{q_3\} \\
&= \{q_0, q_1, q_2, q_3\}.
\end{aligned}$$

Formal Definition of an NFA

Finally: For every string $\omega \in \Sigma^\star$, M ***accepts*** ω if and only if

$$\delta^\star(q_0, \omega) \cap F \neq \emptyset.$$

M ***rejects*** ω otherwise.

The ***language*** of M, $L(M)$, is the set of strings $\omega \in \Sigma^\star$ such that M accepts ω.

Which States are Reachable?

The above formal definitions — including the definition of the "extended transition function" — can be used to write a program that can be used to decide whether a given NFA M accepts a given string $\omega \in \Sigma^{\star}$, ***provided that*** there is an algorithm (and program) that can be used to compute the set $Cl_{\lambda}(q)$ for any given state $q \in Q$.

An algorithm that can be used to do this will be described in a separate document.

What's Next?

NFA's are probably *not* very interesting, by themselves, as computational devices or machine models: The notion of "acceptance" is too complicated.

They ***are*** useful because of results that can be proved if you know about them. In particular, they help to show how every "regular expression" can be used to generate a DFA with the same language — so they help us to make use of regular expressions.

Next Time: A proof that every language $L \subseteq \Sigma^\star$ is the language of an NFA *if and only if* it is the language of a DFA as well.