Lecture #33: NFA’s and the Subset Construction
David Mix Barrington
14 April 2014
Nondeterministic Finite Automata

- Kleene’s Theorem: What and Why?
- Nondeterministic Finite Automata
- The Language of an NFA
- The Model of $\lambda$-NFA’s
- The Subset Construction: NFA’s to DFA’s
- Applying the Construction to No-aba
- The Validity of the Construction
Kleene’s Theorem: What and Why?

- We have now defined two classes of formal languages -- **regular** languages that are denoted by regular expressions, and what we will call **recognizable** languages that are decided by a DFA.
- Kleene’s Theorem, the subject of the next several lectures, says that these two classes are the same.
Kleene’s Theorem

• Mathematically, it’s interesting that two classes with such different definitions should turn out to coincide -- it suggests that the class is important.

• But the theorem also has practical consequences.

• A class of languages is closed under an operation if applying the operation to elements of the class results in another element.
Kleene’s Theorem

• It’s easy to see that the regular languages are closed under union, concatenation, and star, and that the recognizable languages are closed under complement and intersection.

• The theorem tells us that both classes have all these closure properties.

• The efficient way to test whether a string is in a regular language is to create the DFA for the language and run it on the string.
Nondeterminism

- DFA’s are **deterministic** in that the same input always leads to the same output.

- Some algorithms are not deterministic because they are randomized, but here we will consider “algorithms” that are not deterministic because they are **underdefined** -- given a single input, more than one output is possible.

- We had an example of such an algorithm with our generic search, which didn’t say **which** element came off the open list when we needed a new one.
Nondeterministic Finite Automata

- Formally, a **nondeterministic finite automaton** or **NFA** has an alphabet, state set, start state, and final state just like a DFA.

- But instead of the transition function $\delta$, it has a **transition relation** $\Delta \subseteq Q \times \Sigma \times Q$. If $(p, a, q) \in \Delta$, the NFA may move to state $q$ if it sees the letter $a$ while in state $p$. 
Drawing an NFA

- We draw an NFA like a DFA, with an a-arrow from p to q whenever \((p, a, q) \in \Delta\).
- The NFA no longer has the rule that there must be exactly one arrow for each letter out of each state -- there may be more than one, or none.
The Language of an NFA

• We can no longer say what the NFA will do when reading a string, only what it might do. The language of an NFA \( N \) is defined to be the set \( \{ w : w \text{ might be accepted by } N \} \).

• More formally, we define a relation \( \Delta^* \subseteq Q \times \Sigma^* \times Q \) so that the triple \( (p, w, q) \) is in \( \Delta^* \) if and only if \( N \) might go from \( p \) to \( q \) while reading \( w \).

• Then \( w \in L(N) \iff (i, w, f) \in \Delta^* \) for some final state \( f \in F \).
Clicker Question #1

• A string $w$ is in the language of this NFA if it is possible to follow a path with the letters of $w$ from the start state to a final state. Which string is not in $L(N)$?

• (a) abaa
• (b) baab
• (c) bbba
• (d) bbaa
A string $w$ is in the language of this NFA if it is possible to follow a path with the letters of $w$ from the start state to a final state. Which string is not in $L(N)$?

- (a) $abaa$
- (b) $baab$
- (c) $bbaa$
- (d) $bbba$ (bbb can only go one place)
An NFA Example

• Consider the NFA N with state set \{i, p, q\}, start state i, final state set \{i\}, alphabet \{a, b, c\}, and \(\Delta = \{(i, a, i), (i, a, p), (p, b, i), (i, b, q), (q, c, i)\} \).

• This is nondeterministic because there are two a-moves out of i, and several situations with no move at all.
An NFA Example

- Here $L(N)$ is the regular language $(a + ab + bc)^*$, because any path from $i$ to itself must consist of pieces labeled $a$, $ab$, or $bc$.

- It is not immediately clear how, for a larger NFA, we could determine whether a particular string was in $L(N)$. Our method will be to turn $N$ into a DFA.
Interpretations of Nondeterminism

- Because we can’t speak clearly of “what happens when we run N on w”, we need other ways to think of the action of an NFA.

- In our proofs, we will just replace “w ∈ L(N)” by “∃f: (i, w, f) ∈ Δ*” and argue about the possible w-paths in the graph of N.
Interpretations of Nondeterminism

- Suppose the NFA makes a choice uniformly at random whenever it has more than one option. This makes it a **Markov process** in the language of CMPSCI 240.

- In this case $w \in L(N)$ if and only if the probability that $N$ goes to a final state on $w$ is positive. If there is a path, there is a nonzero probability of $N$ taking it, and if there is no path, of course it cannot possibly reach a final state.
Interpretations of Nondeterminism

• Another interpretation has us **fork a process** whenever \( N \) is faced with a choice. One process takes each choice, and if **any** of the processes reaches a final state when it is done reading \( w \), then \( w \in L(N) \).

• “When you come to a fork in the road... take it.” (Y. Berra)
The Model of $\lambda$-NFA’s

- The main reason to use NFA’s is that they are easier to design in many situations when we have some other definition of the language.

- Often we will find it convenient to give the NFA the option to jump from one state to another without reading a letter.

- A $\lambda$-move is a transition $(p, \lambda, q)$ that allows a $\lambda$-NFA to do just that.
The Model of λ-NFA’s

- We need to redefine the type of $\Delta$, so that it is a subset of $Q \times (\Sigma \cup \{\lambda\}) \times Q$.
- In the diagram, this transition is represented by an arrow from $p$ to $q$ labeled with $\lambda$. 

$\begin{array}{c}

p \quad \lambda \quad q

\end{array}$
Paths in a $\lambda$-NFA

- Formally $\Delta^*$ is now more complicated to define. We say that $(p, \lambda, q) \in \Delta^*$ if there is a path of $\lambda$-moves from $p$ to $q$.
- Then we define $\Delta^*(p, wa, q)$ to be true if and only if there exist states $r, s,$ and $t$ such that $(p, w, r), (r, \lambda, s)$ and $(t, \lambda, q)$ are all in $\Delta^*$, and $(s, a, t)$ is in $\Delta$. 

Diagram:
- Node $p$ is connected to node $r$ with a $w$ edge.
- Node $s$ is connected to node $t$ with an $a$ edge.
- Node $t$ is connected to node $q$ with a $\lambda$ edge.
- Node $r$ is connected to node $t$ with a $\lambda$ edge.
Paths in a \( \lambda \)-NFA

- What this means is that \( \Delta^*(p, w, q) \) is true if and only if there exists a path from \( p \) to \( q \) such that the letters on the path, read in order, spell out \( w \).

- There may be any number of \( \lambda \)-moves in the path as well.

- (Thus we don’t even know how long the path from \( p \) to \( q \) might be.)
Clicker Question #2

• Which of these strings is not in the language of this λ-NFA?
• (a) \( \lambda \)
• (b) aab
• (c) bbabb
• (d) Trick question: all three are in the language.
• Which of these strings is not in the language of this \( \lambda \)-NFA?

• (a) \( \lambda \)

• (b) \( aab \) (can’t have two a’s in a row)

• (c) \( bbabb \)

• (d) Trick question: all three are in the language.
The Subset Construction

• Next lecture we’ll see how to convert \(\lambda\)-NFA’s to ordinary NFA’s.

• Now, though, we will convert ordinary NFA’s to DFA’s using the **Subset Construction**.

• Given an NFA \(N\) with state set \(Q\), we will build a DFA \(D\) whose states will be sets of states of \(N\) -- formally, \(D\)’s state set is the **power set** of \(Q\).
The Subset Construction

- Here’s an example of an NFA N for the language \((0 + 01)^*\), with two states i and p, start state i, final state set \(\{i\}\), and transitions \((i, 0, i)\), \((i, 0, p)\), and \((p, 1, i)\).

- At the start of its run, N must be in state i. If the first letter is 0, then it might be in either state i or p after reading the 0. If the first letter is 1, there is no run of N that reads that letter.

![Diagram of NFA N with states i and p and transitions (i, 0, i), (i, 0, p), and (p, 1, i).]
The Subset Construction

- Our DFA $D$ has states $\emptyset$, \{i\}, \{p\}, and \{i, p\}.

- Its start state is \{i\}, its final states are \{i\} and \{i, p\}, and we have $\delta(\{i\}, 0) = \{i, p\}$, $\delta(\{i\}, 1) = \emptyset$, $\delta(\{i, p\}, 0) = \{i, p\}$, $\delta(\{i, p\}, 1) = \{i\}$, $\delta(\{p\}, 0) = \emptyset$, $\delta(\{p\}, 1) = \{i\}$, and $\delta(\emptyset, a) = \emptyset$ for both letters.
Details of the Construction

• The general construction works just like this example.

• The start state of D is \{i\}, where i is the start state of N.

• The final state set of D is the set of all states of D that contain final states of N, since we want D to accept if and only if N can accept.
Details of the Construction

• In general, we need to define $\delta(S, a)$, where $S$ is a state of $D$, meaning that $S$ is a set of states of $N$.

• $S$ represents the possible places $N$ might be before reading the $a$. The set $T = \delta(S, a)$ will be the set of all states $q$ such that the transition $(s, a, q)$ is in $\Delta$ for some $s \in S$.

• In the graph, we take the set of destinations of all the $a$-arrows that start from a state of $S$. 
Details of the Construction

• The most common mistake in computing $\delta$ comes when one of the states in $S$ has no $a$-arrows out of it.

• Students often think that $\emptyset$ must now be part of $\delta(S, a)$. But in fact $\delta(S, a)$ is the union of the sets $\{q: \Delta(s, a, q)\}$ for each $s \in S$.

• So the empty set is part of the result, but doesn’t show up in the description of the result because unioning with $\emptyset$ is the identity operation on sets.
Applying This to No-aba

• The language Yes-aba has an easy regular expression \( \Sigma^*aba\Sigma^* \). From this expression we can build an NFA \( \mathcal{N} \) with state set \{1, 2, 3, 4\}, start state 1, final state set \{4\}, and \( \Delta = \{(1, a, 1), (1, b, 1), (1, a, 2), (2, b, 3), (3, a, 4), (4, a, 4), (4, b, 4)\} \).

• But what if we want a machine for No-aba? Switching the final and non-final states of \( \mathcal{N} \) will not do -- can you see why?
Clicker Question #3

• What is the language of this NFA?
• (a) \((a + b)^* + a + ab\)
• (b) \((a + b)^* + (a + b)^*a + (a + b)^*ab\)
• (c) \((a + b)^*\)
• (d) All three expressions are correct.
Clicker Question #3

- What is the language of this NFA?
- (a) \((a + b)^* + a + ab\)
- (b) \((a + b)^* + (a + b)^a + (a + b)^ab\)
- (c) \((a + b)^*\)
- (d) All three expressions are correct.
Applying This to No-aba

- The best way to get a DFA for No-aba is to first get one for Yes-aba.

- We begin with the start state \{1\} and compute \(\delta(\{1\}, a) = \{1, 2\}\) and \(\delta(\{1\}, b) = \{1\}\). Then we compute \(\delta(\{1, 2\}, a) = \{1, 2\}\) and \(\delta(\{1, 2\}, b) = \{1, 3\}\).
Applying This to No-aba

- Since \( \{1, 3\} \) is new, we must compute \( \delta(\{1, 3\}, a) = \{1, 2, 4\} \) and \( \delta(\{1, 3\}, b) = \{1\} \).

- Then we get \( \delta(\{1, 2, 4\}, a) = \{1, 2, 4\} \) and \( \delta(\{1, 2, 4\}, b) = \{1, 3, 4\} \).

  Not done yet!

- We have \( \delta(\{1, 3, 4\}, a) = \{1, 2, 4\} \) and \( \delta(\{1, 3, 4\}, b) = \{1, 4\} \).
Applying This to No-aba

- Finally, with $\delta(\{1, 4\}, a) = \{1, 2, 4\}$ and $\delta(\{1, 4\}, b) = \{1, 4\}$, we're done -- we have all reachable states.

- If we minimized this DFA, the three final states would merge into one. This gives us our four-state DFA for Yes-aba, from which we can get one for No-aba.
**Validity of the Construction**

- How can we prove that for any NFA $N$, the DFA $D$ that we construct in this way has $L(D) = L(N)$?

- The key property of $D$ is that for any string $w$, $\delta^*(\{i\}, w)$ is exactly the set of states $\{q: \Delta^*(i, w, q)\}$ that could be reached from $i$ on a $w$-path.

- We prove this property by induction -- it is clearly true for $\lambda$ (though if we had $\lambda$-moves it would not be).
Validity of the Construction

• If we assume that $\delta^*\{i\}, w) = \{q: \Delta^*(i, w, q)\}$, we can then prove $\delta^*\{i\}, wa) = \{r: \Delta^*(i, wa, r)\}$ for an arbitrary letter $a$, using the inductive definition of $\delta^*$ in terms of $\delta$, of $\delta$ in terms of $\Delta$, and of $\Delta^*$ in terms of $\Delta$.

• Once this is done, it is clear that $w \in L(D) \leftrightarrow \exists f: f \in \delta^*\{i\}, w) \leftrightarrow \exists f: \Delta^*(i, w, f) \leftrightarrow w \in L(N)$.

• Note that in general $D$ could have $2^k$ states when $N$ has $k$ states. But if we leave out unreachable states, $D$ could be much smaller.