Non-Deterministic Finite-State Automata

Robert M. Keller
Harvey Mudd College
March 2010

What is Non-Determinism?

- Can an automaton have choice or free-will?
- Non-determinism supposes it can.

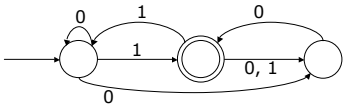
Uses of Non-Determinism?

- Modeling asynchronous events, wherein we don't know the order in which things happen.
- Modeling certain kinds of parallel computing.
- As a mathematical construction device.

NFA's

- NFA = Non-deterministic finite-state automaton
- From any given state, we can have **more than one** transition for the same letter.
- From any given state, we can have **no** transitions for a given letter.
- We can have free transitions that occur without using any letter. (These are labeled with ε.)
- We can have multiple starting states, freely chosen.

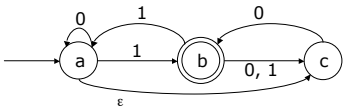
Random Example of an NFA



Paths in an NFA

- Let x be a **string** over the alphabet of N.
- Let q and q' be states of N.
- We say there is a "**path from q to q' with label x**" provided that there is a series of arcs connecting q to q' and the labels on that series, when concatenated, form x.
- Note that ε-transitions can be concatenated and do not change the string to which they are concatenated.

Paths in an NFA (not identical to previous)

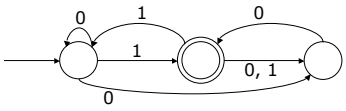


Some Paths
1 from a to b
ε from a to c
0 from a to c
00 from a to b
11 from a to a
1 from b to c
010 from a to a
. . .

Acceptance by an NFA

- An NFA accepts a string x if there is **a** path from **some** start state to **some** accepting state with label x.
- If there is no such path, then the NFA does not accept the string.
- The **language** accepted by an NFA is the set of all strings accepted by the NFA.

Original Example of an NFA



Strings accepted:
1
00
01
000
001
100
110
111
. . .

Strings not accepted:
ε
0
10
11
010
011
101
. . .

Observation

- A DFA can be viewed as a special case of an NFA.
- If L is a language accepted by a DFA, then L is also accepted by an NFA.

The NFA Theorem

- If L is accepted by an NFA, there is also a DFA that accepts L.

Proof of the NFA Theorem (called the "Subset Construction")

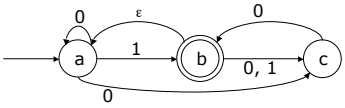
- Suppose we have an NFA N. We want to construct a DFA D that accepts the same language exactly.
- Let S be the set of states of N. Choose as the states of D the set P(S) = the **set of all subsets** of the states of N. Since S is finite, so is P(S).
- (In practice, we won't usually need **all subsets**.)

Transitions of D

- Let R be a state of D. So R ⊆ S.
- We must define a transition from R for each symbol σ in our alphabet.
- To designate the **next state** from R given symbol σ, use:

$R_\sigma = \{q' \in S \mid (\exists q \in R)\ \exists \text{ path from } q \text{ to } q' \text{ with label } \sigma\}$

Examples of DFA Transitions



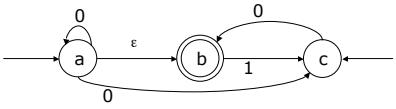
$\{a, b\}_0 = \{a, c\}$
$\{a, b\}_1 = \{a, b, c\}$
$\{b\}_0 = \{a, c\}$
$\{b\}_1 = \{c\}$
$\{c\}_0 = \{a, b\}$
$\{c\}_1 = \varnothing$

Initial State of D

- Letting S_0 be the set of initial states of N, the initial state of D is defined to be the subset

$S_0 \cup \{q' \in S \mid (\exists q \in S_0)\ \exists \text{ path from } q \text{ to } q' \text{ with label } \varepsilon\}$

Examples of Initial State



The initial state of D is {a, b, c}.

Conclusion of the NFA Theorem

- The preceding construction shows how we capture all paths of the original NFA N in a DFA D.
- The only part remaining is defining the **accepting states** of D. These will be the subsets of S that **contain at least one accepting state** of N.

Interpretation of the DFA Construction

- The DFA constructed in the proof of the theorem can be interpreted as an automaton that simulates all possible choices of the NFA **in parallel**.
- This idea will have at least one other application (to be seen).

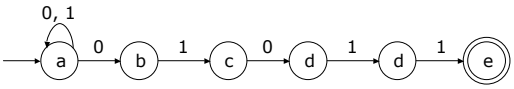
A Computational Shortcut

- By beginning with the initial state of the DFA and working from there, we can eliminate the need to generate all states in P(S) in most cases.
- In other words, we need only generate the states **reachable from** the initial state of D.

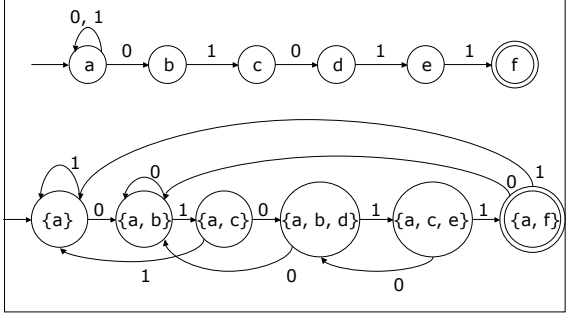
Application of the Construction

- Suppose we want to construct a DFA that will accept the language of **strings ending with a given string**, for example: 01011.
- The tricky part here is that if we get as input, for example, 01010, while this is not accepted, the last part 010 can possibly be used as the initial part of a string that is, for example 0101011.
- By constructing an appropriate NFA and converting it, it is easy to get the DFA right.

An NFA for {0, 1}* 01011



Constructing a DFA for {0, 1}* 01011



Notes on the Previous Example

- Not all subsets were reachable. Those that weren't were not generated.
- The number of states is the same. This is a coincidence; it may be more or fewer.
- The transition structure is more complex in the DFA. This is typical.

Algorithmic Application

- The principle underlying the illustrated method for text matching was the insight and basis for two text searching algorithms:
  - **Knuth-Morris-Pratt**: Search for a single string
  - **Aho-Corasick**: Search for a finite set of strings

Another Possibility for Search

- Rather than go through the DFA construction and simulate the result to do search a text, it is possible to simulate the NFA directly.
- In this case there would be a **set** of "current states" rather than a single one.
- Just use **multiple state pointers** for the simulation rather than a single one.

More applications of the subset construction are forthcoming.

- Closure properties
- DFA from regular expressions