CS 420 - Compilers

Dr. Chen-Yeou (Charles) Yu

- **Finite Automata (3.6)**
  - **Nondeterministic Finite Automata (NFA) (3.6.1)**
  - **Transition Tables (3.6.2)**
  - **Acceptance of Input Strings by Automata (3.6.3)**
  - **Deterministic Finite Automata (DFA) (TBD. In Part7)**

Finite Automata

- There is a thing called ***"finite automata"*** which is in the heart of the translation for the tool of ***Lex*** we introduced in the last time.
- These are essentially graphs, like **transition diagrams**, with a few differences:
  - Finite automata are ***recognizers***; they simply say "**yes**" or "**no**" about each **possible input string**.

Finite Automata

- Finite automata come in **two** flavors:
  - ***Deterministic*** *finite automata (**DFA**)*
    - For each state, and for each symbol of its input alphabet, there is exactly **one edge** with **that symbol** **leaving** that state.
  - ***Nondeterministic*** *finite automata (**NFA**)*
    - It has **no restrictions** on the labels of their edges.
    - The **SAME** symbol can be used to **label several edges** out of the **same state**, and "**epsilon**", the empty string, is a possible label.
- Both deterministic and nondeterministic finite automata are capable of recognizing the **same languages**.
- In fact these languages are exactly the **same** languages, called the **regular languages**, that **regular expressions** can describe

NFA

This is saying the "Sigma" is a set, NOT containing the epsilon

In the transition function, that might involve "epsilon"

- NFA consists of:

1. A finite set of states S.
2. A set of input symbols Σ, the *input alphabet*. We assume that ϵ, which stands for the empty string, is never a member of Σ.
3. A *transition function* that gives, for each state, and for each symbol in $\Sigma \cup \{\epsilon\}$ a set of *next states*.
4. A state s_0 from S that is distinguished as the *start state* (or *initial state*).
5. A set of states F, a subset of S, that is distinguished as the *accepting states* (or *final states*).

So, F is a subset of S, stands for many double-circles

NFA

- We can represent either an NFA or DFA by a **transition graph**, where the **nodes are states** and the **labeled** edges represent the **transition** function
- There is an edge **labeled a** from **state s** to **state t**, if and only if **t** is one of the next states **for state s** and **input a**. (So, the edges are the inputs)

The **SAME** symbol can be used to **label several edges** out of the **same state**

a) The same symbol can label edges from one state to several different states, and

b) An edge may be labeled by ϵ, the empty string, instead of, or in addition to, symbols from the input alphabet.

NFA

- An Example
  - The language (a|b)*abb
  - It is similar to regular expressions that describe languages of real interest

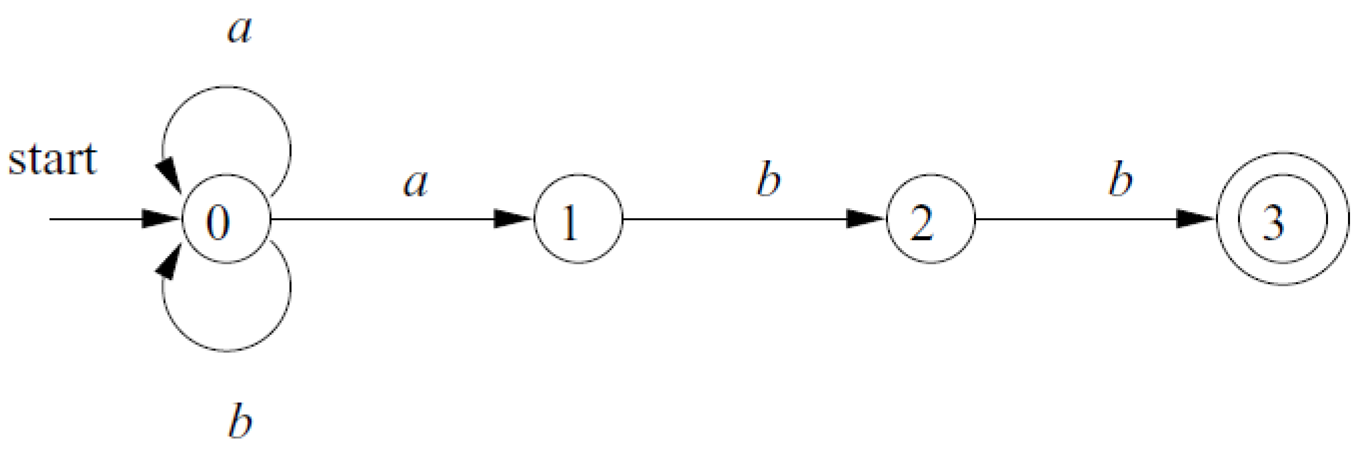

Figure 3.24: A nondeterministic finite automaton

Transition Tables (for NFA)

- We can also represent an NFA by a **transition table**, whose **rows** correspond to **states**, and whose **columns** correspond to the **input** symbols and epsilon.
- The entry for a given state and input is the **value** of the transition function applied to those arguments.
- If the transition function has **no information** about that state-input pair, we put Φ in the table for the pair.
- The advantage that we can easily find the transitions on a given state and input.
- Its **disadvantage** is that it takes **a lot of space**, when the input alphabet is large
- See the next page for detail

Transition Tables (for NFA)

(a|b)*abb

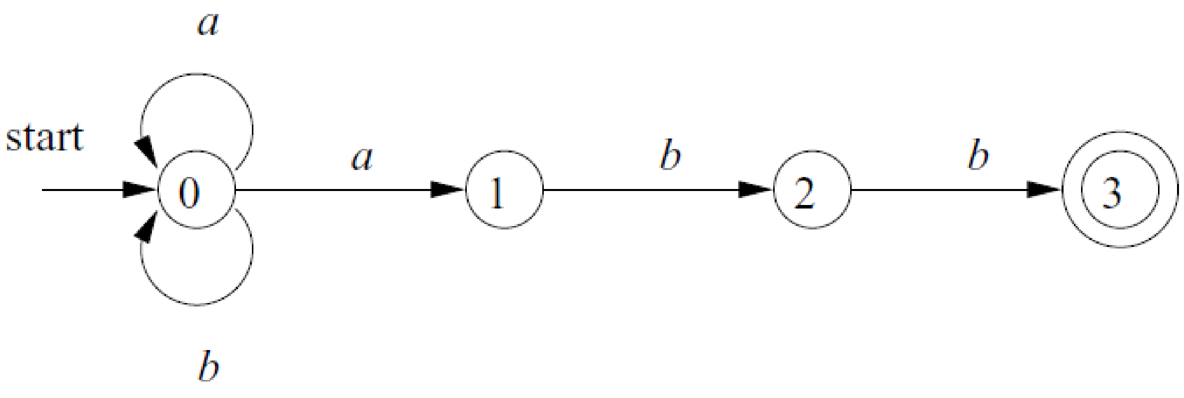

Figure 3.24: A nondeterministic finite automaton

STATE	a	b	ϵ
0	$\{0, 1\}$	$\{0\}$	$\emptyset$
1	$\emptyset$	$\{2\}$	$\emptyset$
2	$\emptyset$	$\{3\}$	$\emptyset$
3	$\emptyset$	$\emptyset$	$\emptyset$

Figure 3.25: Transition table for the NFA of Fig. 3.24

Acceptance of Input Strings by Automata

- An NFA accepts input string x (for example, in our grammar, “abb”) if and only if there is **some path** in the **transition** graph **from the start state** to **one of the accepting states**
- Note that “epsilon” labels along the path are effectively **ignored**, since **the empty string does not contribute to the string constructed along the path.**

Acceptance of Input Strings by Automata

- The **same** input string **might lead to different paths**. Considering the same **grammar**: (all of them below are “**aabb**”)

(a|b)*abb

Example 3.16 : The string *aabb* is accepted by the NFA of Fig. 3.24. The path labeled by *aabb* from state 0 to state 3 demonstrating this fact is:

$$0 \xrightarrow{a} 0 \xrightarrow{a} 1 \xrightarrow{b} 2 \xrightarrow{b} 3$$

Note that several paths labeled by the same string may lead to different states. For instance, path

$$0 \xrightarrow{a} 0 \xrightarrow{a} 0 \xrightarrow{b} 0 \xrightarrow{b} 0$$

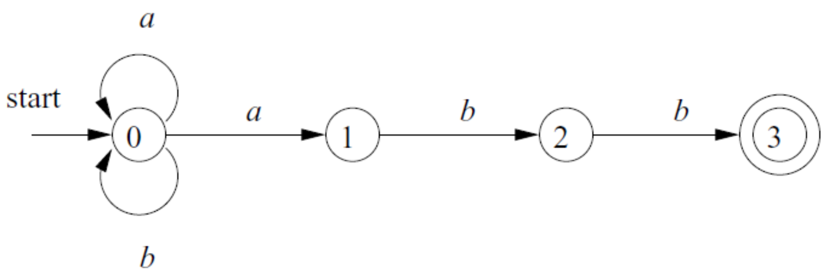

Figure 3.24: A nondeterministic finite automaton

- However, **the 2nd one** leads to the state “0” which is **NOT accepting**

Acceptance of Input Strings by Automata

- So, the **acceptance** of a string depends on what?
  - If a **path** is labeled by that string **leads from the starting state to an accepting state!**
  - The existence of other paths leading to a nonaccepting state is **irrelevant**.
- If a “language”, is said to be defined (or accepted) by an NFA, that means, it is the set of strings labeling some **path** from the **start** to an **accepting** state

Acceptance of Input Strings by Automata

- Here is an example from the book

Example 3.17: Figure 3.26 is an NFA accepting $L(\mathbf{aa}^*|\mathbf{bb}^*)$. String aaa is accepted because of the path

$$0 \xrightarrow{\varepsilon} 1 \xrightarrow{a} 2 \xrightarrow{a} 2 \xrightarrow{a} 2$$

Note that ϵ's "disappear" in a concatenation, so the label of the path is aaa.

This is NFA because it **violates** "exactly **one edge** with **that symbol** **leaving** that state." See? The **epsilon**!

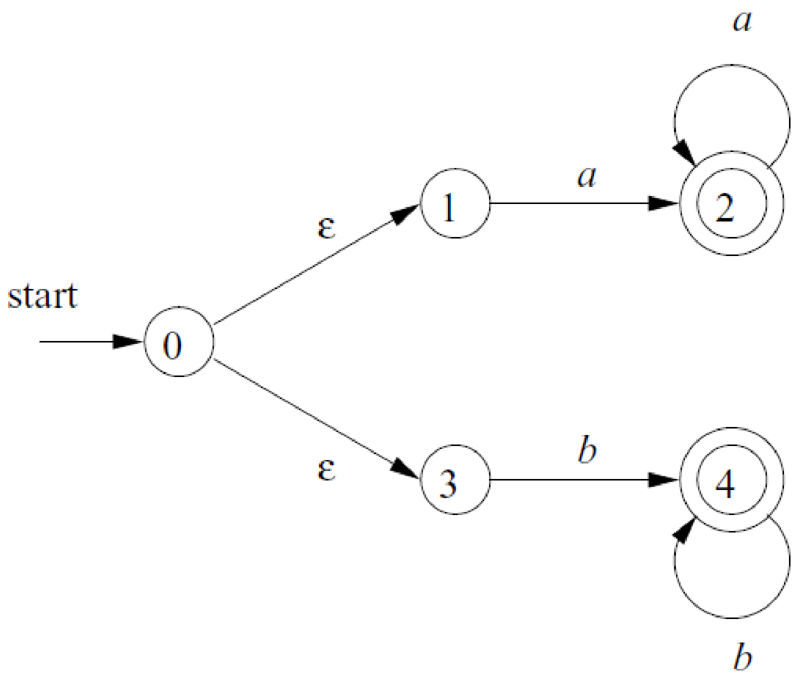

Figure 3.26: NFA accepting $\mathbf{aa}^*|\mathbf{bb}^*$

Deterministic Finite Automata

- A deterministic finite automaton (DFA) is a **special case** of an NFA
- We will talk about this in the next meeting!