CS 241 — Lecture 11

Bartosz Antczak *Instructor: Kevin Lanctot* *February 7, 2017*

11.1 Non-deterministic Finite Automata (NFA)

The key difference found in NFA's is that *two or more edges leaving the same state can have the same label and lead to different states.* This means that the next state is non-deterministic (i.e., there exists a state of possible states rather than a single state). Consider the input 010, running it through the NFA ℓ, we get:

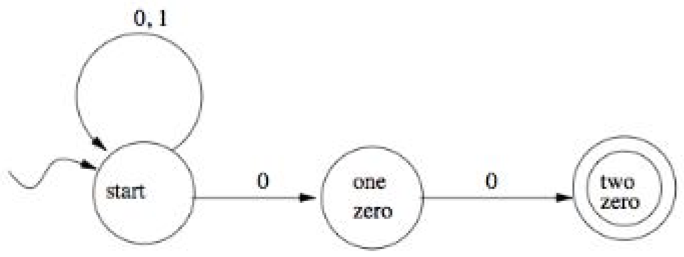

Figure 11.1: An example of an NFA. From Prof. Lanctot's slides.

- ε S
- 0 $\{S, 10\}$
- 1 $\{S\}$
- 0 $\{S, 10\} \cap \{20\} = \emptyset$

Ergo 101 $\notin \ell$.
When working with NFA's, we don't ask "what state are we in?". Instead, we ask

What set of states are we in?

One can consider a human as an NFA. We experience different states at the same time, for instance, we can be happy about finishing an assignment but sad at the same time because we missed another assignment.

11.1.1 Comparisons with a DFA

- A language is accepted in an NFA and DFA if at least one path leads to an accepting state
- DFA's are easier to implement
- NFA's are simpler because they tend to have less states than a corresponding DFA; however, NFA's are slower because they require set data types, which take longer to process

11.1.2 Transducers

It's an *extension*. It allows to provide an output on any transition:

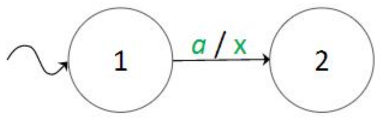

Figure 11.2: On transition a, output x. Courtesy of Prof. Lanctot's slides.

11.1.3 Epsilon-Non-Deterministic-Finite-Automata

An $\varepsilon-$NFA allows the use of $\varepsilon-$transitions (i.e., a transition that occurs without consuming any input:

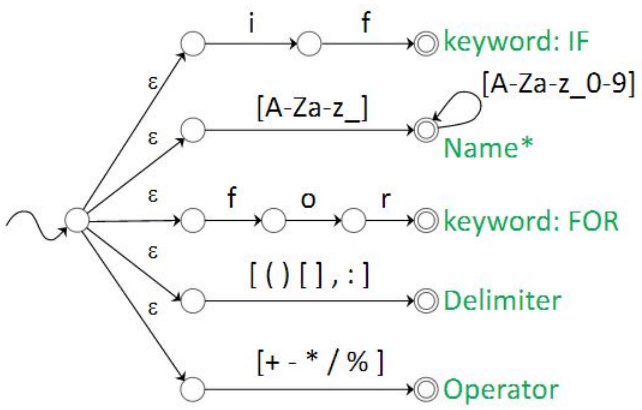

Figure 11.3: An $\varepsilon-$NFA. Courtesy of Prof. Lanctot's slides.