Dr. Bui

CSC 212 – Theory of Computing

Shyang Kao

April 25, 2022

# *TEXT PARSING MACHINE – FINITE AUTOMATA*

## Abstract

Automata Theory is a fascinating subject of computer science that focuses on theory. Mathematicians began constructing - both conceptually and practically - computers that emulated certain qualities of man, completing computations more rapidly and consistently, during the twentieth century. The word automaton, which is closely connected to the word "automation," refers to automated processes that carry out specified operations. Simply expressed, automata theory is concerned with the logic of computation in the context of basic machines known as automata. Computer scientists may learn how computers compute functions and solve issues using automata, as well as what it means when a function is specified as computable or a question is represented as decidable. Automata theory is concerned with the design of robots from basic automaton building pieces in computer science. An electronic digital computer is the best example of a generic automaton. Automata networks could be programmed to replicate human behavior. Furthermore, Vending machines, traffic lights, video games, speech recognition, regular expression matching, and other industries have all benefited from automata theory. As a result of these advantages, various machines based on automata theory for logictis labor processes have been

developed. This paper will describe a basic nondeterministic finite automaton (NFA) that can recognize and accept the given words from a sentence by parsing method**.**

## **Introduction**

The study of abstract machines and automata, as well as the computational problems that can be solved with them, is known as automata theory. Computer scientists can learn how machines compute functions, solve issues, and what it means for a function to be defined as computable by using automata. Scientists may apply finite automata theory to a variety of real-world machines and potentially improve them as a result of this understanding. States and transitions make up this automaton. We can utilize this strategy to address a real-world problem like text parsing because of the features of this sort of automata. In addition, the inputs (words we wish to search) must satisfy all of the states and transitions in order for finite automata to accept them and search the words. Keyword search is one of the most significant features in reading books, searching text, and condensing articles, among other things. Let's pretend we're on the Internet and need to look up some key words. We'll want to look for any lines that include one or more of these words.

## **Body**

- ### Methodology

- Non-deterministic Finite Automata (NDFA), is particularly useful for parsing text using regular expressions. You can easily convert a regular expression into an NDFA and then use the NDFA to parse the text input**.** This is the reason why we start with nfa.
- A non-deterministic finite accepter or nfa is defined by the quintuple

**M = (Q, Σ, δ, q0, F),**

where

Q is a finite set of internal states,

Σ is a finite set of symbols called the input alphabet,

δ : Q × Σ → Q is a total function called the transition function,

q0 ∈ Q is the initial state,

F ⊆ Q is a set of final states.

- Assume we have a nfa for the words: dice and iced.
- Make the nfa diagram

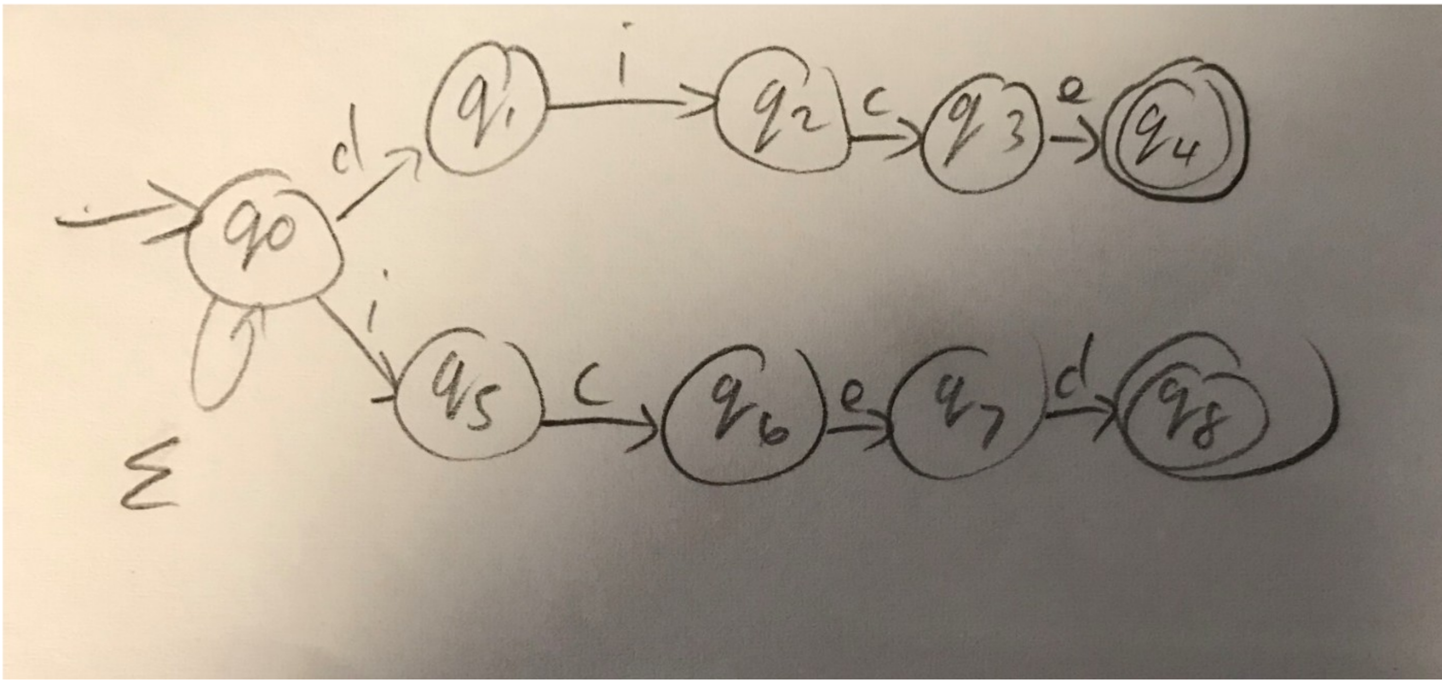


Figure 1: nfa for “dice” and “iced”

- In the figure 1 above, q0 is the initial state, and q4 and q8 are final states. If the character ends in either q4 or q8, the word is accepted and has been found. In addition, if the first character of the word is not start with either d or i, the transition will not be processed.
- Next, we convert nfa into dfa using the transition table below:

| States | d | i | c | e | d |
|---|---|---|---|---|---|
| q0 | q0,q1 | q0,q5 | q0 | q0 | q0 |
| q1 |  | q2 |  |  |  |
| q2 |  |  | q3 |  |  |
| q3 |  |  |  | q4 |  |
| q4 |  |  |  |  |  |
| q5 |  |  | q6 |  |  |
| q6 |  |  |  | q7 |  |
| q7 |  |  |  |  | q8 |
| q8 |  |  |  |  |  |

Transition table for nfa

| States | d | i | c | e | d | ∑-d-i | ∑-c d-i | ∑-e d-i | ∑-d d-i |
|---|---|---|---|---|---|---|---|---|---|
| q0 | {q0,q1} | {q0,q5} |  |  |  | q0 |  |  |  |
| {q0,q1 } | {q0,q1} | {q0,q2,q5 } |  |  |  | q0 |  |  |  |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| {q0,q5} | {q0,q1} | {q0,q5} | {q0,q6} | | | | q0 | | |
| {q0,q2, q5} | {q0,q1} | {q0,q5} | {q0,q3,q6} | | | | q0 | | |
| {q0,q3, q6} | {q0,q1} | {q0,q5} | | {q0,q7} | | | | q0 | |
| {q0,q3, q6} | {q0,q1} | {q0,q5} | | {q0,q4,q7} | | | | q0 | |
| {q0,q7} | {q0,q1} | {q0,q5} | | | {q0,q8} | | | | q0 |
| {q0,q4, q7} | {q0,q1} | {q0,q5} | | | {q0,q8} | | | | q0 |
| {q0,q8} | {q0,q1} | {q0,q5} | | | | q0 | | | |

Transition table for dfa

- Next we make a dfa diagram based on the dfa transition table

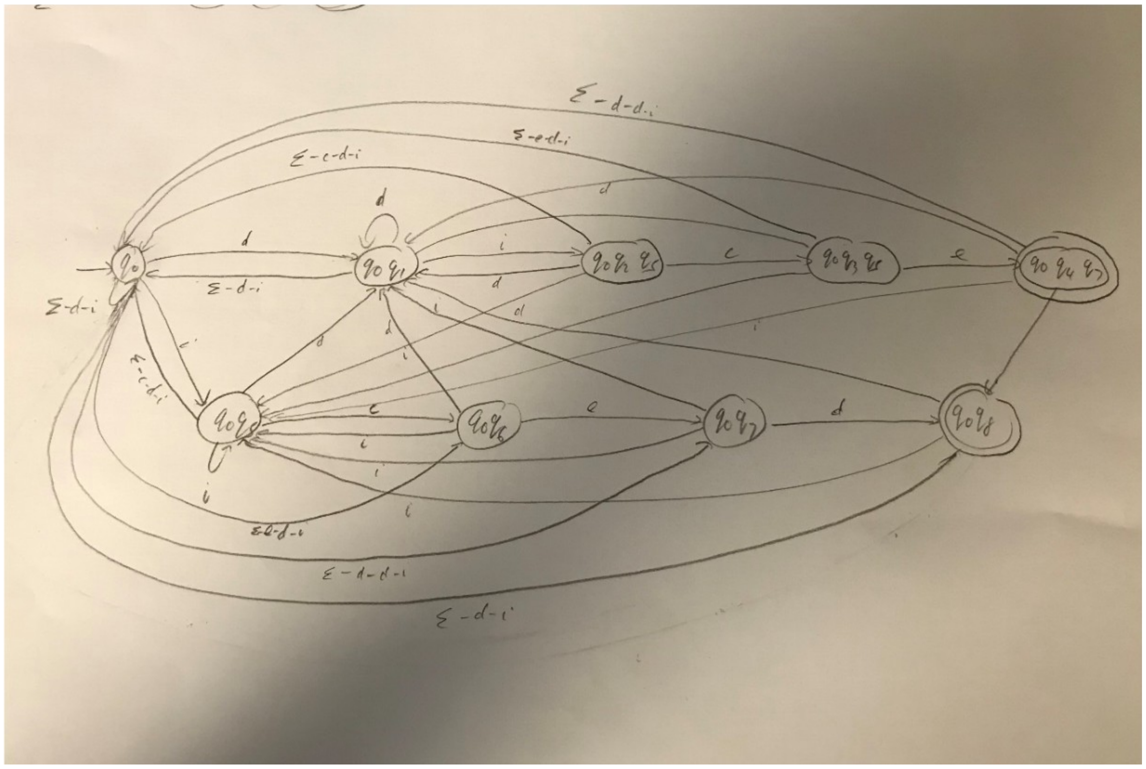
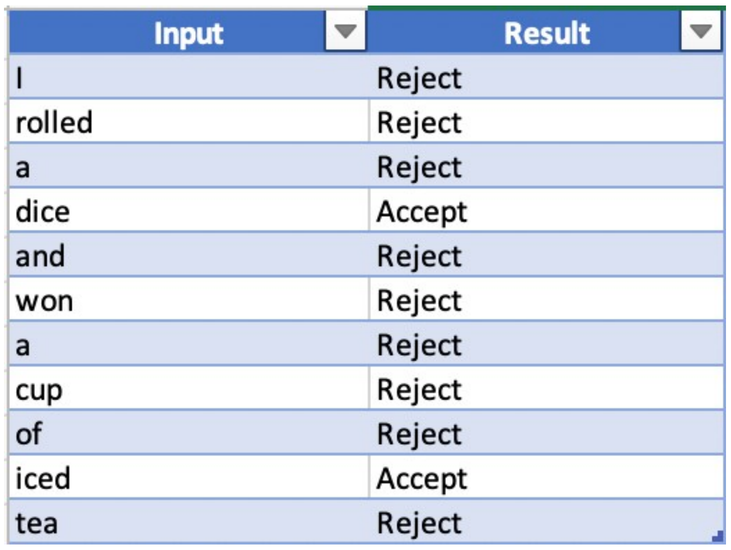


Figure 2: dfa for “dice” and “iced”

- **Result**

After finishing the nfa and dfa above, we understand how the text parsing works. For example, we have the sentence : “I rolled a dice and won a cup of iced tea” When we test the sentence, the result will be only the words “dice” and “iced” got accepted, and the rest will not.

Below is the image for the result



| Input | Result |
|---|---|
| I | Reject |
| rolled | Reject |
| a | Reject |
| dice | Accept |
| and | Reject |
| won | Reject |
| a | Reject |
| cup | Reject |
| of | Reject |
| iced | Accept |
| tea | Reject |

- **Discussion**

- In fact, using an NDFA often entails a lot of retracing. The algorithm must follow every potential transition from one state to the next, and if one fails, it must go back and try again. There are methods for converting an NDFA to a Deterministic Finite Automaton that are theoretically available (DFA). When your parser is made up of regular expressions, this translation can be rather difficult.
- Another problem will be time consuming when we convert from nfa to dfa.

## <u>Summary</u>

- The study of abstract machines and automata, as well as the computational problems that can be solved with them, is known as automata theory
- Automata theory can used to solve so many real-world problems not only text parsing machine. Even nowadays, finite automata was put into use such as vending machines, traffic lights, and especially video games.
- Since technology develops day by day, more and more machines can apply automata theory into it.

## References

[1] “High Performance Text Parsing Using Finite State Machines” from HackerNoon, by Spiros Dimopoulos. Link.

[2] “Basics of Automata Theory” from Automata Theory. Link

[3]”15 Differences DFA And NFA” from Vivadifferences, by Viva. Link

[4] “Formal Languages And Automata” from textbook, by Peter Linz.

[5] “Applications of Deterministic Finite Automata” by Eric Gribkof. Link