CPT411: AUTOMATA THEORY & FORMAL LANGUAGES

Semester 2, Academic Session 2023/2024

GROUP ASSIGNMENT 1

Title: L6. English Conjunctions/Adverb/Adjectives Finder

Group Number: 34

Name	Matric Number
Ching Jia Ying	153463
Yeong Yen Ting	152664

Date of Submission: 10th May 2024

1.0 Introduction

Conjunctions serve as a fundamental pillar in English syntax that binds words, phrases, and clauses together, facilitating coherence and cohesion within sentences. Their significance lies in their ability to establish relationships between various components of a sentence, whether it be adding information, indicating contrast, showing cause and effect, or demonstrating sequence. It is crucial for effective communication and comprehension in both spoken and written English to understand the role and function of conjunctions.

In this report, we delve into the design and analysis of a Deterministic Finite Automaton (DFA) for English conjunctions, aiming to explain the patterns and structures inherent in their usage. The DFA recognizer constructed is designed to identify English conjunctions within the provided text. The scope of the language is a defined set of 20 English conjunctions as listed below. There are a total of 41 states, starting from q0 to q40, with the start state (q0), accepting state (q3), and the trap state (q21).

The alphabet of this language is defined as

$\Sigma = \{ a, \ldots, z, 0, \ldots, 9 \}$

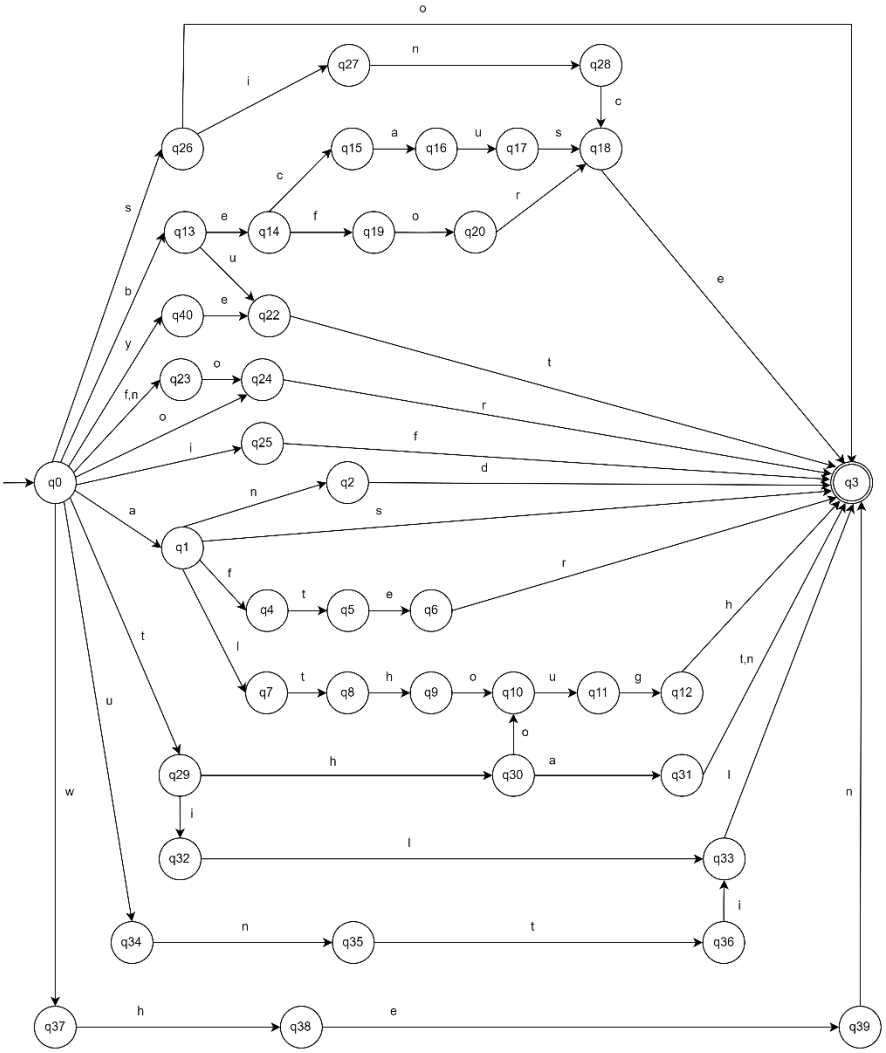
The language that is accepted is defined as

$L = \{w \in \Sigma * \mid w$ contains "after", "although", "and", "as", "because", "before", "but", "for", "if", "nor", "or", "since", "so", "than", "that", "though", "till", "until", "when", "yet"}.

The formal definition of the DFA is defined as

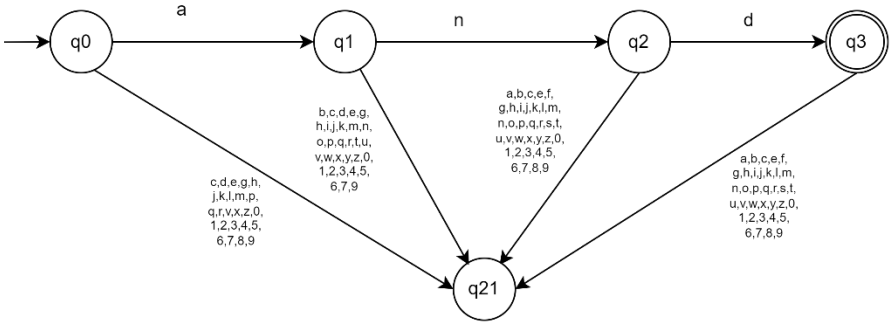
M = {{q0, q1, q2, q3, q4, q5, q6, q7, q8, q9, q10, q11, q12, q13, q14, q15, q16, q17, q18, q19, q20, q21, q22, q23, q24, q25, q26, q27, q28, q29, q30, q31, q32, q33, q34, q35, q36, q37, q38, q39, q40}, {a, … , z, 0, …, 9}, δ, q0, {q3}}.

The complete DFA is as follows:



Note: All alphabets that are not explicitly mentioned will lead to the trap state.

The diagram below shows the detailed version of the DFA that shows the trap state. The word taken is “and” hence if there is a character a from the start state, it will be directed to q1, while the other unrelated characters will be led into the q21 (trap state). When it is at q1, if “n” is fed, then it will proceed to q2, and if the machine detects a character “d” then it will proceed with q3. If there is any unrelated character in these states, then it will all be led to the trap state.



2.0 Implementation Information

2.1 Approach to Read and Process Strings

First, we accept a text file as an input to be further processed. There are two main functions to read and process the strings, which are the highlightAcceptedWords function and isAccepted function. The highlightAcceptedWords function will take the paragraph as an input and split it into individual words as strings. The punctuation is removed so that they will not be processed in the DFA. This function will call the isAccepted function for each string by looping and will highlight the string if the string is accepted. The isAccepted function takes a word as an input string and processed it character by character according to the DFA transitions starting from the start state q0. Once all characters have been processed, it checks if the final state is an accepted state. It will update the interface to say that the word is accepted or rejected.

2.2 Overview of Programming Constructs Used for the Program

The program is structured as an HTML document to provide a basic layout required for rendering content on a web page. CSS is used to style the colour for highlighting the currently processed character and the accepted strings. The program logic is mainly written in JavaScript which consists of variables to store the DFA, accepted states, and other data required for processing. There are two main functions to process the strings and update the interface.

3.0 Conclusion

In conclusion, the DFA for the selected 20 English conjunctions has been successfully constructed. By using this program, we can identify the position and occurrences of the selected conjunctions in a text file. This tool efficiently locates these conjunctions within text files, enhancing our understanding of syntactic structures. Moving forward, there is potential to expand the DFA's capabilities by incorporating additional conjunctions, thereby increasing its utility in linguistic analysis. Furthermore, ongoing efforts to optimize the DFA's performance will contribute to its effectiveness in handling larger datasets with improved efficiency. By leveraging these enhancements, we can further empower language researchers and practitioners with a valuable tool for comprehensive language analysis across various contexts.

4.0 Appendix

4.1 Program Source Code

Complete source code can be found here:

https://github.com/JiaaaJiaa/English-Conjunction-Finder

Defining DFA from script.js

```
// Define the DFA
        // after, although, and, as, because, before, but,
        // for, if, nor, or, since, so, than, that, though,
        // till, until, when, yet
        var dfa = {
            'q0': {'a': 'q1', 'b':'q13', 'f':'q23', 'i':'q25', 'n':'q23','o':'q24','s':'q26', 't':'q29', 'u':'q34',
'w':'q37', 'y':'q40', 'default': 'q21'},
            'q1': {'n': 'q2', 'f': 'q4', 'l': 'q7', 's':'q3', 'default': 'q21'},
            'q2': {'d': 'q3', 'default': 'q21'},
            'q3': {'default': 'q21'},
            'q4': {'t': 'q5', 'default': 'q21'},
            'q5': {'e': 'q6', 'default': 'q21'},
            'q6': {'r': 'q3', 'default': 'q21'},
            'q7': {'t': 'q8', 'default': 'q21'},
            'q8': {'h': 'q9', 'default': 'q21'},
            'q9': {'o': 'q10', 'default': 'q21'},
            'q10': {'u': 'q11', 'default': 'q21'},
            'q11': {'g': 'q12', 'default': 'q21'},
            'q12': {'h': 'q3', 'default': 'q21'},
            'q13': {'e': 'q14', 'u':'q22', 'default': 'q21'},
            'q14': {'c': 'q15', 'f': 'q19', 'default': 'q21'},
            'q15': {'a': 'q16', 'default': 'q21'},
            'q16': {'u': 'q17', 'default': 'q21'},
            'q17': {'s': 'q18', 'default': 'q21'},
            'q18': {'e': 'q3', 'default': 'q21'},
            'q19': {'o': 'q20', 'default': 'q21'},
            'q20': {'r': 'q18', 'default': 'q21'},
            'q22': {'t': 'q3', 'default': 'q21'},
            'q23': {'o':'q24', 'default': 'q21'},
            'q24': {'r':'q3', 'default': 'q21'},
            'q25': {'f':'q3', 'default': 'q21'},
            'q26': {'i':'q27', 'o':'q3', 'default': 'q21'},
            'q27': {'n':'q28', 'default': 'q21'},
            'q28': {'c':'q18', 'default': 'q21'},
            'q29': {'h':'q30', 'i':'q32', 'default': 'q21'},
            'q30': {'a':'q31', 'o':'q10', 'default': 'q21'},
            'q31': {'n':'q3', 't':'q3', 'default': 'q21'},
```

```
            'q32': {'l':'q33', 'default': 'q21'},
            'q33': {'l':'q3', 'default': 'q21'},
            'q34': {'n':'q35', 'default': 'q21'},
            'q35': {'t':'q36', 'default': 'q21'},
            'q36': {'i':'q33', 'default': 'q21'},
            'q37': {'h':'q38', 'default': 'q21'},
            'q38': {'e':'q39', 'default': 'q21'},
            'q39': {'n':'q3', 'default': 'q21'},
            'q40': {'e':'q22', 'default': 'q21'},

            'q21': {
                'a': 'q21', 'b': 'q21', 'c': 'q21', 'd': 'q21', 'e': 'q21',
                'f': 'q21', 'g': 'q21', 'h': 'q21', 'i': 'q21', 'j': 'q21',
                'k': 'q21', 'l': 'q21', 'm': 'q21', 'n': 'q21', 'o': 'q21',
                'p': 'q21', 'q': 'q21', 'r': 'q21', 's': 'q21', 't': 'q21',
                'u': 'q21', 'v': 'q21', 'w': 'q21', 'x': 'q21', 'y': 'q21',
                'z': 'q21', '0': 'q21', '1': 'q21', '2': 'q21', '3': 'q21',
                '4': 'q21', '5': 'q21', '6': 'q21', '7': 'q21', '8': 'q21',
                '9': 'q21', 'default': 'q21'
            }
        };

        var acceptedStates = new Set(['q3']);
```

Function highlightAcceptedWords from script.js

```
// Function to highlight the accepted words in a paragraph
        function highlightAcceptedWords(paragraph, callback) {
            var words =
paragraph.match(/\b[\w'-]+[.,\/#!$%\^&\;:{}=\_`~()]?|[\s][.,\/#!$%\^&\;:{}=\_`~()]/g);
            var highlightedWords = [];
            var acceptedWordsPositions = []; // Array to store the positions of the accepted words
            var acceptedWordsCount = {}; // Object to store the count of accepted words
            var i = 0;

            runButton.style.display = 'inline-block';

            // Use a self-invoking function to create a loop with delay
            (function loop() {
                if (i < words.length) {
                    var word = words[i].replace(/[.,\/#!$%\^&\*;:{}=\-_`~()]/g,"").replace(/<br>/g, "");
                    isAccepted(word, function(accepted) {
                        if (accepted) {
                            highlightedWords.push('<span class="highlight">' + words[i] + '</span>');
                            acceptedWordsPositions.push({word: word, position: i+1}); // Push the position to the
array

                            // If the word is already in the object, increment its count
```

```
                            // Otherwise, add it to the object with a count of 1
                            if (acceptedWordsCount[word]) {
                                acceptedWordsCount[word]++;
                            } else {
                                acceptedWordsCount[word] = 1;
                            }
                        } else {
                            highlightedWords.push(words[i]);
                        }
                        i++;
                        loop();
                    }, function(charIndex) {
                        // Update the paragraph to highlight the current character
                        var highlightedWord = words[i].substr(0, charIndex) +
                            '<span class="current">' + words[i][charIndex] + '</span>' +
                            words[i].substr(charIndex + 1);
                        document.getElementById('paragraph').innerHTML = highlightedWords.join(' ') + ' ' +
highlightedWord + ' ' + words.slice(i + 1).join(' ');
                    });
                } else {
                    callback(highlightedWords.join(' '), acceptedWordsPositions, acceptedWordsCount); // Pass the
positions and the count object to the callback
                }
            })();
        }
```

Function isAccepted from script.js

```
// Function to check if a word is accepted by the DFA
        function isAccepted(word, callback, highlightCallback) {
            var state = 'q0'; // Set the initial state
            word = word.toLowerCase(); // Convert the word to lowercase
            var i = 0; // Initialize counter

            // Use a self-invoking function to create a loop with delay
            (function loop() {
                if (!proceed) {
                    // If the Enter key hasn't been pressed, end this iteration
                    requestAnimationFrame(loop);
                    return;
                }

                if (i < word.length) {
                    var char = word[i];
                    if (state in dfa && char in dfa[state]) {
                        state = dfa[state][char];
                    } else if (state in dfa && 'default' in dfa[state]) {
                        state = dfa[state]['default'];
```

```
                    } else {
                        document.getElementById('result').innerHTML = 'Character: ' + char + ', No transition found.
Word rejected.';
                        callback(false, word); // Pass the word to the callback function
                        return;
                    }
                    // Update the interface with the current state
                    document.getElementById('status').innerHTML = 'Character: ' + char + ', New state: ' + state;
                    highlightCallback(i, acceptedStates.has(state)); // Highlight the current character
                    i++;
                    setTimeout(loop, 0); // Delay next iteration
                } else {
                    if (acceptedStates.has(state)) {
                        var resultElement = document.getElementById('result');
                        resultElement.innerHTML = 'The word "' + word + '" is accepted.';
                        resultElement.style.color = ' #23C552'; // Set the text color to green
                        callback(true, word); // Pass the word to the callback function
                    } else {
                        var resultElement = document.getElementById('result');
                        resultElement.innerHTML = 'No final state reached. The word "' + word + '" is rejected.';
                        resultElement.style.color = '#F84F31'; // Set the text color to red
                        callback(false, word); // Pass the word to the callback function
                    }
                }

                // Reset the proceed flag at the end of each iteration
                if(clickedButton){
                    proceed = true;
                }else{
                    proceed = false;
                }
            })();
        }
```