510 Final Project

Formal Language Description

The feeling of ordering food from your favorite restaurant has always felt satisfying. Personally, Chipotle is my go-to restaurant when I'm home. If I'm on campus, a hot and fresh meal from DeBruce will always cure my hunger. Because of these real-world situations that I experience in my week-to-week life, I have created a language to model customer orders at a restaurant. More specifically, the specific food orders one orders on campus or off. The language represents orders based on the rules that collectively capture the fully customizable meals. The goal of this language is to provide a structured way to utilize the learnings of lectures and connect them to real-world applications.

This language utilizes three alphabets: bases, the proteins, and the toppings. The bases include the basic options of a bowl or a burrito. Then, the type of rice. The proteins include chicken or steak. The toppings include beans, guacamole, lettuce, veggies, cheese, or queso.

Every order must follow these rules. Every base must start with exactly one base: B (Bowl) or R (Burrito). Rice is assumed as included with the base. Proteins must be either C (Chicken) or S (Steak). Toppings include N (Beans), G (Guacamole), L (Lettuce), V (Veggies), E (Cheese), Q (Queso).

Alphabet Breakdown:

$\Sigma_B = \{B, R\}$

$\Sigma_P = \{C, S\}$

$\Sigma_T = \{N, G, L, V, E, Q\}$

Every order follows the structure: Base Protein (Toppings)*. Toppings can also be repeated to represent and insinuate the customer has ordered extra of a topping. For example, if you wanted extra guacamole, you would be able to order that. Some examples of strings include BC (Bowl with chicken and no toppings), BSG (Bowl with steak and guacamole), RCN (Burrito with chicken and Beans), RCLGE (Burrito with chicken, lettuce, guacamole, and cheese), BSGG (Extra guacamole). Invalid strings are B (Burrito with no protein), RN (Burrito protein), and E (Missing base and protein).

This language is practical because it simulates a realistic system for ordering food and divides the alphabet into different categories to make the design easy to follow. It is a simple system but it provides variations for formal languages while also avoiding unnecessary complexity. The complete alphabet of the language:

$\Sigma = \{B, R, C, S, N, G, L, V, E, Q\}$

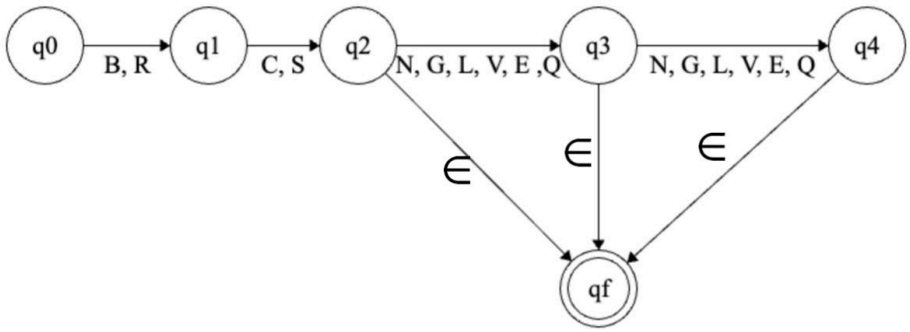
$S \rightarrow BP\ TExtra$

$B \rightarrow B \mid R$

$P \rightarrow C \mid S$

$TExtra \rightarrow T\ TExtra \mid \in$

$T \rightarrow N \mid G \mid L \mid V \mid E \mid Q$



Test Cases:

Test Case	String Tested	Accepted or Rejected	Explanation
1	B	Rejected	Bowl as the base. Every order must have a protein selected along with the base. Therefore, this is an invalid input.
2	R	Rejected	Burrito as the base. Every order must have a protein selected along with the base. Therefore, this is an invalid input.
3	BC	Accepted	Bowl with chicken. This is a valid string because it is a base + a protein.
4	RC	Accepted	Burrito with chicken. This is a valid string because it is a base + a protein.
5	BCG	Accepted	Bowl with chicken and guacamole. This is a valid string because it is a base + a protein + a topping.
6	RCG	Accepted	Burrito with chicken and guacamole. This is a valid string because it is a base + a protein + a topping.
7	BCGLV	Accepted	Bowl with chicken, guacamole, lettuce, and fajita veggies. This is a valid string because it is a base + a protein + multiple toppings.
8	RCGLV	Accepted	Burrito with chicken, guacamole, lettuce, and fajita veggies. This is a valid string because it is a base + a protein + multiple toppings.
9	BCBnL	Rejected	Bowl with chicken, a random input, and lettuce. This is not a valid string because ‘Bn’ is not an option the menu allows.
10	RR	Rejected	Burrito with a… burrito. You cannot have two bases. Therefore, this is an invalid string.

Test Case	String Tested	Accepted or Rejected	Explanation
11	BS	Accepted	Bowl with steak. This is a valid string because it is a base + a protein.
12	RS	Accepted	Burrito with steak. This is a valid string because it is a base + a protein.
13	RSGG	Accepted	Burrito with steak and double guacamoles. This is valid since it allows the customer to order double of a topping.
14		Rejected	This is an empty input, therefore it is an invalid string.

Data Structure

Our data structure represents a nondeterministic finite automaton in memory. It includes essential components like states, transitions, start states, accepting states, and a method for simulation input strings to determine whether or not they are accepted.

The code consists of multiple attributes that include the states, the alphabet, the initialState, and finalStates. The states include a list of all ‘State’ objects in the automaton. The alphabet includes a list of valid input symbols. The initialState represents the starting state of the automaton. The finalStates are a list of all of the accepting states.

The methods in the code consist of add_state, set_initial_state, add_final_state, parse_input_to_nfa, and run.

- add_state: Adds a state to the NFA
- set_initial_state: Sets the start state
- add_final_state: Adds a state to the list of accepting states
- parse_input_to_nfa: Parses the definition of the automaton from a formatting string to initialize the NFA
- run: Reads the input string and returns True if accepted and False if it isn’t.