EECS 510 Final Project

Lillian Brooks-Kanost and Eyassu Mongalo
Dr. Jennifer Lohoefener

EECS 510

December 2, 2025

1 Design a Formal Language

Our language of choice is the C programming language.

The alphabet for this language is all valid ASCII characters (ASCII 0-127).

A formal definition for our language is as follows:

Let

$$\Sigma_{\text{all}}$$

be the set of all ASCII characters, and let

$$\Sigma = \Sigma_{\text{all}} \setminus \{(,), [,]\}$$

be the set of all ASCII characters other than parentheses and brackets. Let

$$\Sigma_{\text{all}}^*$$

denote its Kleene closure.

The language L is defined as

$$\left\{ u \in \Sigma_{\text{all}}^* \;\middle|\; \begin{array}{l} \text{for every prefix } v \text{ of } u,\ \#_((v) \geq \#_)(v) \text{ and } \#_[(v) \geq \#_](v), \\ \text{and } \#_((u) = \#_)(u) \text{ and } \#_[(u) = \#_](u), \\ \text{excluding any (,), [,] that are enclosed in single or double quotes.} \end{array} \right\}.$$

For the purpose of this project, we are focusing on simple parsing of the C programming language standard library, restricted to 1) ensuring opening brackets are matched with closing brackets, and 2) ensuring opening parentheses are matched with closing parentheses. The intent of this is to build a simple C parser, focusing on a subset of rules that are easier to implement.

The C programming language has a variety of semantic rules. Since the scope of this project is fairly small, we are ignoring all but the following rules:

1. All opening brackets "{" must be paired with a closing bracket "}", unless declared as a string literal or character. There must not be unpaired brackets, or parsing fails. Brackets must also be correctly balanced/nested: an opening bracket must precede its matching closing bracket, or parsing fails.

   (a) Example success states:

   ```c
   #include <stdio.h>
   int main(){
       printf("Hello World!");
   }
   ```

   ```c
   "{"
   ```

   (b) Failing states:

   ```c
   #include <stdio.h>
   int main()
       printf("Hello World!");
   }//missing opening bracket
   ```

```
}{//brackets out of order
```

2. All opening parentheses “(” must be paired with a closing parenthesis “)”, unless declared as a string literal or character. There must not be unpaired parentheses, or parsing fails. Parentheses must also be correctly balanced/nested: an opening parenthesis must precede its matching closing parenthesis, or parsing fails.

   (a) Example success states:

   ```
   #include <stdio.h>
   int main(){
       printf("Hello World!");
   }
   ```

   ```
   '('
   ```

   (b) Failing states:

   ```
   #include <stdio.h>
   int main({ //missing closing parenthesis
       printf("Hello World!");
   }
   ```

   ```
   )(//parentheses out of order
   ```

2 Grammar

$S) \rightarrow P$
$S\} \rightarrow B$
$S" \rightarrow Q_d$
$S' \rightarrow Q_s$
S EOF $\rightarrow E$
$S\_ \rightarrow \_S$
$(P \rightarrow S$
BOF $P \rightarrow F$
$\_P \rightarrow P\_$
$\{B \rightarrow S$
BOF $B \rightarrow F$
$\_B \rightarrow B\_$
$Q_d' \rightarrow S$
Q_d EOF $\rightarrow F$
$Q_{d\_} \rightarrow$ (space)Q_d
$Q_s' \rightarrow S$
Q_s EOF $\rightarrow F$
$Q_{s\_} \rightarrow$ (space)Q_s
$\{E \rightarrow F$
$(E \rightarrow F$
BOF $E \rightarrow A$
$\_E \rightarrow_E$
$F \rightarrow \lambda$
$A \rightarrow \lambda$

In this case, _ is the wildcard operator

3 Automaton

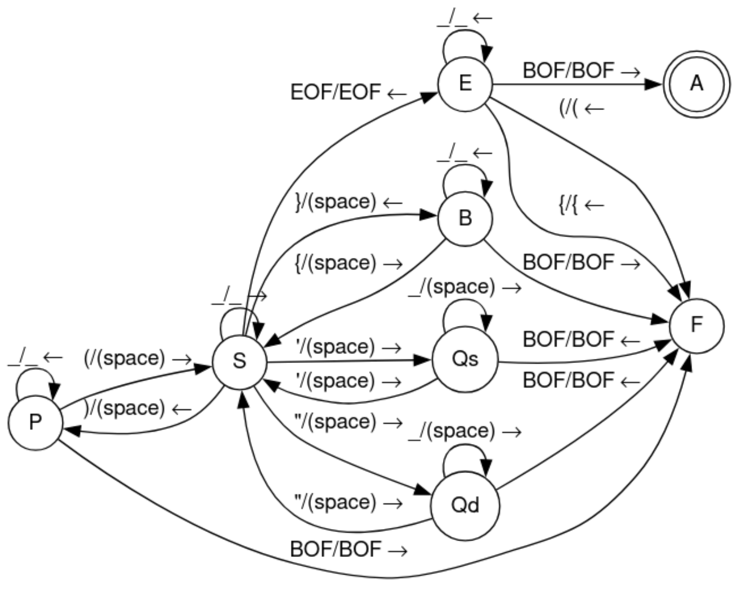

Figure 1: Turing Machine

4 Data Structure

S P B Qd Qs E A F

```
<NUL> <SOH> <STH> <EOT> <ENQ> <ACK> <BEL> <BS>
<HT> <LF> <VT> <FF> <CR> <SO> <SI> <DLE> <DC1>
<DC2> <DC3> <DC4> <NAK> <SYN> <ETB> <CAN> <EM>
<SUB> <ESC> <FS> <GS> <RS> <US>  (space) ! " #
$ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ;
< = > ? @ A B C D E F G H I J K L M N O P Q R S
T U V W X Y Z [ \ ] ^ _ ` a b c d e f g h i j k
l m n o p q r s t u v w x y z { | } ~ <DEL> EOF
```

S
A
_ = wildcard operator
BOF, EOF = beginning of file, end of file
$\Gamma = \Sigma_{all}$ + EOF + BOF
S _ _ → S
S) (space) ← P
S } (space) ← B
S ‘ (space) → Qs

S “ (space) → Qd
S EOF EOF ← E
P ((space) → S
P _ _ ← P
P BOF BOF → F
B { (space) → S
B _ _ ← B
B BOF BOF → F
Qs ‘ (space) → S
Qs _ (space) → Qs
Qs BOF BOF ← F
Qd “ (space) → S
Qd _ (space) → Qd
Qd BOF BOF ← F
E _ _ ← E
E ((← F
E { { ← F
E BOF BOF → A

5 Testing

Testing function found in main.rs