YAFSM

CS3423-DSL
Group 6

Overview

The language YAFSM is designed to simplify working with FSMs such as DFAs, NFAs and PDAs.

It is syntactically similar to C with additional datatypes, operators and inbuilt functions.

The Lexer is built using flex, while Parser is built using bison.

Language Details

Implementation Details

- Datatypes:
  - Standard: int_x, u_int_x, float_y, bool, string, char ($x \in \{8,16,32,64\}$ & $y \in \{32,64\}$)
  - Non-Standard:
    - Ordered Set: o_set
    - Unordered Set: u_set
    - Regular Expression: regex
    - Context Free Grammar: cfg
    - Deterministic Finite Automata: dfa
    - Non Deterministic Finite Automata: nfa
    - PushDown Automata: pda

Implementation Details (contd.)

- Operators:
  - Over Automata:
    - Kleene: *
    - Concatenation: @
    - Union: +
    - Negation: !
  - Over Set and Automata:
    - Arithmetic Operators: +, -, *
    - Logical Operators: !, &&, ||
    - For sets, Power Set: ^2

Implementation Details (contd.)

- Standard
  - Arithmetic Operators: +, -, *, /, %
  - Logical Operators: !, &&, ||
  - Comparison Operators: ==, <=, >=, !=, <, >
  - Assignment Operators: =, +=, -=, *=, /=, %=, &=, |=
- Miscellaneous
  - Access Struct Member: .
  - Access Set Element: [ ]
  - Function Call: ( )

Implementation Details (contd.)

- Constants:
  - Integer Constants: Same as in C
  - String Constants: Same as in C
  - Character Constants: Same as in C
  - Floating Point Constants: Same as in C
  - Bool Constants: Same as in C
- Brackets: Parentheses, Brackets and Braces
- Punctuations: { “;” , “.” , “->” , “:” , “,” }
- Epsilon: \e
- Single Quote: ‘

Implementation Details (contd.)

- The keywords used in language are
  - Control keywords: if, elif, else, while, break
  - Return keyword: return
  - Struct keyword: struct
  - Comments: <!--, –!>

- Identifiers are same as in C-language
- White spaces and New line characters are ignored

Lexer

Overview

Most of the code written in the lex file is easy to understand, and does not contain any look aheads.

In the lex file, comments and regex tokens have been handled using states and lookaheads.

Handling of these tokens are explained in the next 2 slides.

Comments

Comments in YAFSM start with the <!-- token and end with --!>. There is no distinction between single line comments and multi line comments.

Comments can also be nested.

First the comment begin token is read, and the lexer shifts into the comment state. Here, it reads tokens, and considers it to be a comment, unless it is a comment begin/end token, where the comment level is incremented / decremented by 1 respectively. Once the comment level is 0, it shifts back to the initial state.

Regex

A regex is of the form r’REGEX’

Handling regex can either be the task of the lexer, or the translation layer. Here, the lexer handles regex, using multiple states.

First, the lexer checks that if a character r is read, then whether a string encapsulated by single quotes is present or not. If so, it recognizes the token as a regex token, and shifts to the regex state. Then using multiple states, rules of regex is recognized, and after encountering the ending single quote, it shifts back to the initial state.

Parser

Overview

In the Parser file, all tokens returned from the lex file are defined, along with the associativity of the tokens.

Considering the grammar, the language has 6 main instructions:

- Function Declarations
- Function Call Statements
- Struct Declarations
- Variable Declarations
- Assignment Statements
- Control Statements

Overview (contd.)

The grammar in the parser file reflects that the language is syntactically similar to C

The grammar also contains rules to detect regex pattern errors such as unbalanced paranthesis

It also contains grammar rules to define expressions for declaration of

- Context Free Grammars
- DFA/ NFA
- PDA

Up-Next (Semantic Analysis)

The following things are not handled by the parser yet and remain to be implemented as semantic checks

- Type checking for lhs and rhs
- Function parameter checks
- Return types
- Symbol Tables
- Declaration before using variables
- .....