CSEN 1003: Compilers

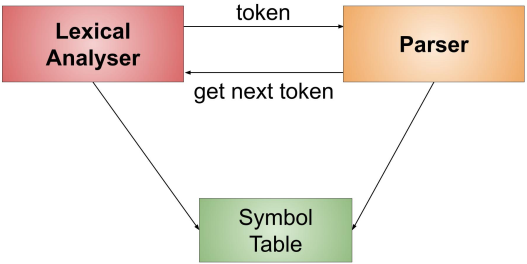
Tutorial 2 - Lexical Analysis

9/2/2020 - 12/2/2020

Today's Plan

1. Regular Definitions
2. Fallback DFA with Actions
3. Recap

Function of a Lexical Analyzer



1. Partition the input stream into lexemes.
2. Generate a token for each lexeme. There are two types of tokens.
    - $\langle L, A\rangle$ where L is a lexical category and A is an attribute.
    - $\langle L\rangle$ where L is a lexical category.
3. Auxiliary function: ignore irrelevant substrings.

Today's Plan

1. Regular Definitions
2. Fallback DFA with Actions
3. Recap

Regular Languages

- A set of lexemes for a programming language makes up a regular language which can be represented by regular expressions.

Extensions to regular expressions:

1. $R_1 \mid R_2 \equiv R_1 \cup R_2$.
2. $R? \equiv R \mid \varepsilon$
3. $[a_1 a_2 ... a_n] \equiv a_1 \mid a_2 \mid ... \mid a_n$ where $a_i \in \Sigma$.
4. $[a_1 - a_n] \equiv [a_1, a_2, ..., a_n]$ where $a_1 ... a_n$ is a natural order.

Regular Definitions

Definition

A regular definition is a finite sequence of pairs $P_i = \langle D_i, R_i \rangle$:

$$D_1 \longrightarrow R_1$$

$$..$$

$$D_i \longrightarrow R_i$$

where R_1 is a regular expression over Σ and R_i is a regular expression over Σ and $D_1, ..., D_{i-1}$.

Example (Java Identifiers)

Write a regular definition to represent Java Identifiers.

Action-Augmented Regular Definitions

Example

Give a regular definition for non-negative numbers without leading zeros.

Action-Augmented Regular Definitions

Example

Give a regular definition for non-negative numbers without leading zeros.

Example

Augment your regular definition with actions to generate tokens. For integers a token $\langle \mathit{int}, \mathit{num} \rangle$ should be generated, and for floats a token $\langle \mathit{float}, \mathit{num} \rangle$ where num is the recognized lexeme.

Today's Plan

1. Regular Definitions

2. Fallback DFA with Actions

3. Recap

How to Build a Lexical Analyser in 5 steps?

1. Write an action-augmented regular definition where for each category there is a pair $\langle D_i, R_i \rangle$.
2. For each D_i, compile a regular expression by unrolling.
3. Construct $R = R_1 \mid R_2 \mid \ldots \mid R_n$.
4. Construct an NFA N_i for each R_i. Make sure each N_i has a unique accept state labelled by D_i. Construct NFA N to be the union of all the NFAs.
5. Construct a fallback DFA with actions equivalent to N.

Operation of a Fallback DFA with Actions

Example

Consider the input string aaabaabb and the action-augmented regular definition:

$$\begin{array}{llll} 1 & \longrightarrow & \texttt{b}^{+} & \{\texttt{printf("1")}\} \\ 2 & \longrightarrow & \texttt{aab}^{*} & \{\texttt{printf("2")}\} \\ 3 & \longrightarrow & \texttt{a} & \{\texttt{printf("3")}\} \end{array}$$

Draw the state diagram of an equivalent fallback DFA with actions. What will be printed when the DFA is run of the provided input?

More Regular Definitions

Example

Write a regular definition to generate tokens for strings without the enclosing quotes or escape characters.

The following methods are predefined:

- `table.open()`: opens a new entry.
- `table.close()`: closes the last opened entry.
- `table.isOpen()`: returns true if there is a current open entry.
- `table.last()`: returns the content of the last closed entry.
- `table.add(char)`: appends a character to the last open entry.

Today's Plan

1. Regular Definitions
2. Fallback DFA with Actions
3. Recap

Points to Take Home

1. Action-Augmented Regular Definitions.
2. Fallback DFA with Actions.

Next Week: Context Free Grammars!