Compilers – Fall 2022

HW4: Lexical Analysis **Total: 20 points**

Q1. Regular Expression (2 pts)

We know for most of the programming languages are case sensitive, so the keywards can only be written in a way. In C++, it is "if" and is not possible to be an "IF".

For example, we now are going to deal with SQL's SELECT.

We just want to have this SELECT can be select, sELect, selecT, …, etc, any of mixed cases. Write a language by using regular expression to deal with such a kind of SELECT.

Note, we only need to deal with SELECT.

Q2. The structure of token (4 pts)

For a most commonly seen token is like such kind of structure:

<token-name, attribute-value>

Token-name are the elements corresponding to RHS of a grammar and the attribure value are just a value or a pointer to a sumbol table entry (if it is a variable)

Consider the following C-like grammar, an assignment statement:

asst-stmt → id = expr;

expr → expr * expr

expr → id * number

expr → number

expr → id

Now, tell me, how to describe a token in the following formula's coefficient?

A = B * 2

1. **In other words, how to describe "2", a number, as a token? (2 pts)**

2. **How to describe the operator "*" as a token. (The answer is hidden in my slides) (2 pts)**

Q3. Transition-Diagram based Lexical Analyzer (8 pts)

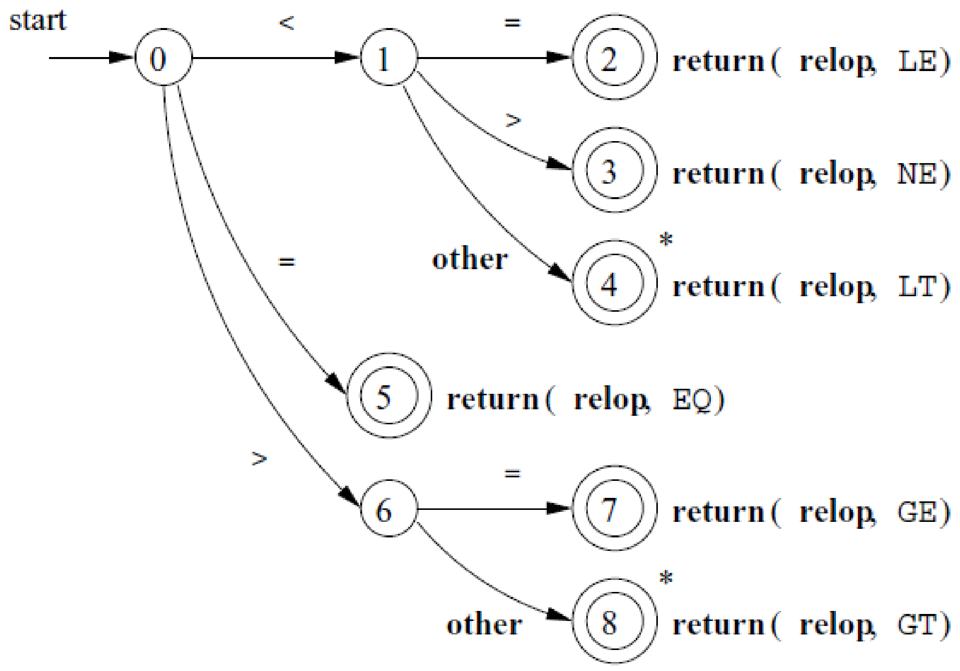
In the Lect3_Ch3 Lecical Analysis (Part5),

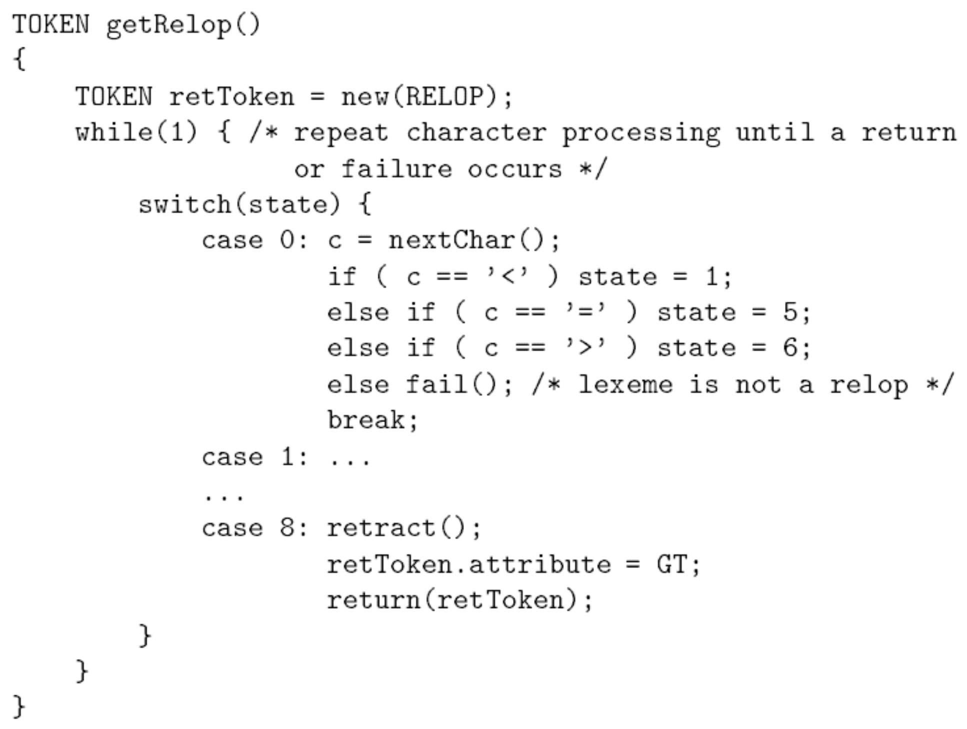
1. **What is the double circle means in the transition diagram? (2 pts)**

2. **What is the directed edge labeled by "other" means? (2 pts)**

3. **When is the fail() get called? (2 pts)**

4. **In the state 8, when we see that "*" in the diagram, what are we supposed to do? (2 pts)**





```
TOKEN getRelop()
{
    TOKEN retToken = new(RELOP);
    while(1) { /* repeat character processing until a return
                  or failure occurs */
        switch(state) {
            case 0: c = nextChar();
                    if ( c == '<' ) state = 1;
                    else if ( c == '=' ) state = 5;
                    else if ( c == '>' ) state = 6;
                    else fail(); /* lexeme is not a relop */
                    break;
            case 1: ...
            ...
            case 8: retract();
                    retToken.attribute = GT;
                    return(retToken);
        }
    }
}
```

Figure 3.18: Sketch of implementation of **relop** transition diagram

Q4. Tell me what is the different between NFA and DFA. (4 pts)

You need to point out at lease 2 different viewpoints.

Q5. Transition Table (2 pt)

Transition table can be used as a kind of tool to describe NFA or DFA.

Tell me, why sometimes, it is not very ideal for NFA to adopt such kind of tool?

[Hint] You need to understand the structure of the Transition Table.