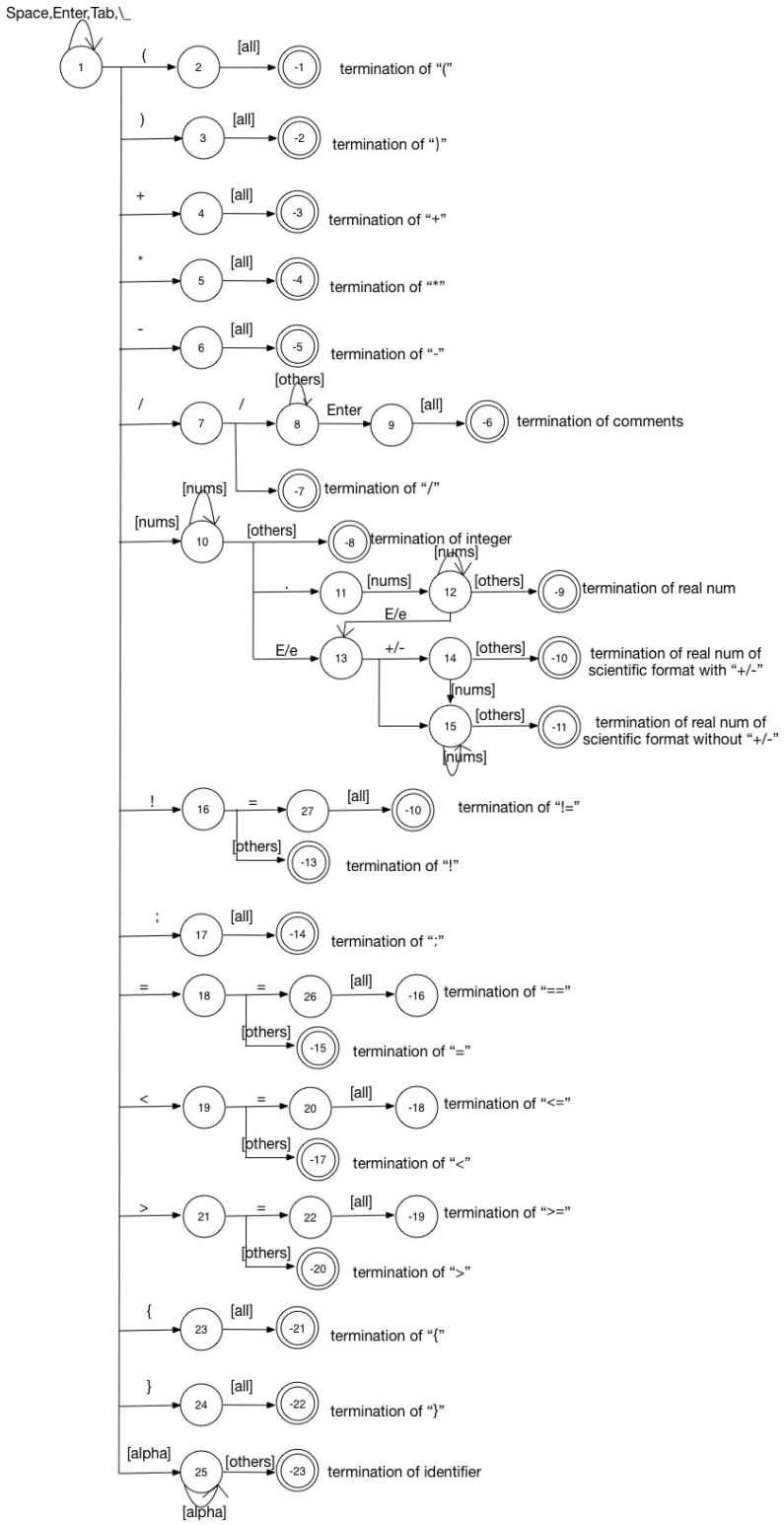
report for lexical analyzer

This is my report for lexical analyzer, I will show my design proposal here.

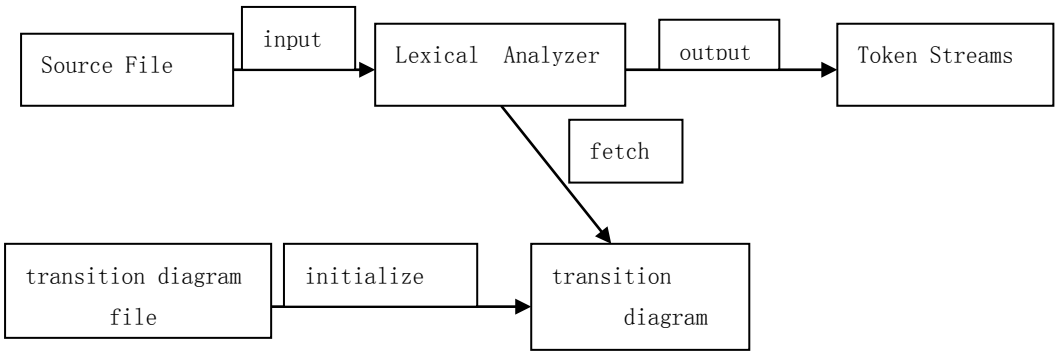
■First let me show my state transition graph



■I use a two-dimensional table to represent the state transition table, so I have to number the reserved words(some unused numbers can be used for expansion).

identifier	01															
Operators	+	-	/	*	=	==	<	<=	>	>=	!=	(	)	{	}	;
and others	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17
keywords	int		real		if		then		else		while					
	28		29		30		31		32		33					
constants	integer constants:34 normal real constants:35 real constants of scientific forms with +/-:36 real constants of scientific forms without +/-:37															

■My lexical analyzer model looks like following:



■Pseudocode

```
LexParse(string text)
{
   pos=0
   state=1
   while(pos<text.length())
   {
      buf=buf+text[pos];
      state=LexTbl[state][text[pos]];
      if(IsTerminal(state))
            submit the token
            buf=""
      else if (IsNonTerminal(state))
            continue
      else
            error handing
      pos++;
   }
}
```

■Answer for questions

- What are your token types? What are attribute values of tokens? What do they represent?
  As for the quadruple (tokentype,attributevalue,linenumber,lineposition), I let the ID explained above to represent tokentype, I almost didn't use attributevalue and linenumber and lineposion is just represent their literal meanings.
- What is the structure of the symbol table? What do you store and how?
  It is easy to build a simple symbol table in my program, which has some attributes like pointer,type and name of variable,but I think the whole work of building the symbol can be done until semantic analysis and besides symbol table is of no use to my final results,so I didn't implement symbol table.
- What kind of lexical errors can your lexical analyzer handle, and how it can recover from those errors?

  Lexemes which are not belong to our language is the errors.

  Compose an error queue in which I store the number of the line where error occurs. Then just skip until the sentence ends.