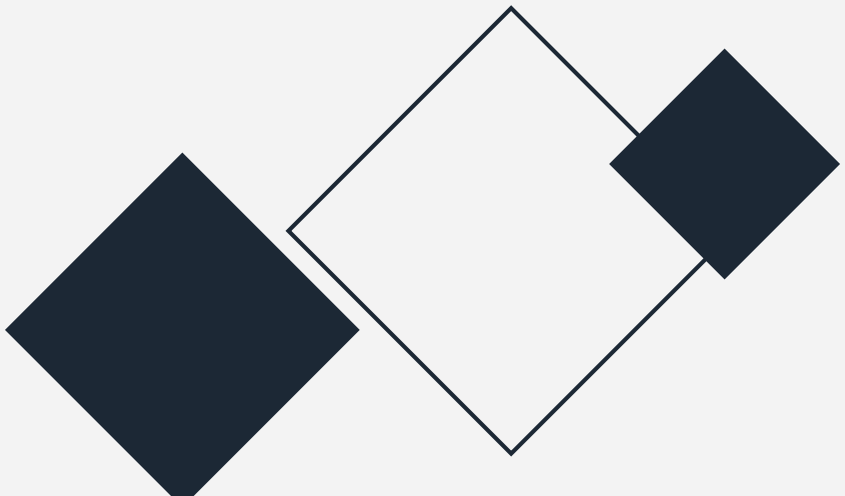
Finite automata of lexical analysis for C programming language in prolog

(Only some key words, variable names and boundary symbols are recognized.)

Reporter：Gao Xiangyu
School of Computer Science and Technology, Hangzhou Dianzi University



The origin of the project

- This language is suitable for the analysis of natural language.
- Natural language is much more complicated than programming language because it has too many combinations.
- Just now, I need to make a compiler demo recently, so I want to try to use prolog to make lexical analysis and syntax analysis.

about

Its functions and features

NFA —

At present, I have only made a preliminary model of lexical analysis.

It can identify some keywords (if, else, for, int, etc.), variables, numbers and delimiters (spaces, operation symbols, brackets, etc.).

1. Using predicate ‘start/1’ to enter a string, For example, start(‘if (i < 20)’).
2. NFA matches and distinguishes the parts of the string.
3. Output process and result of analysis

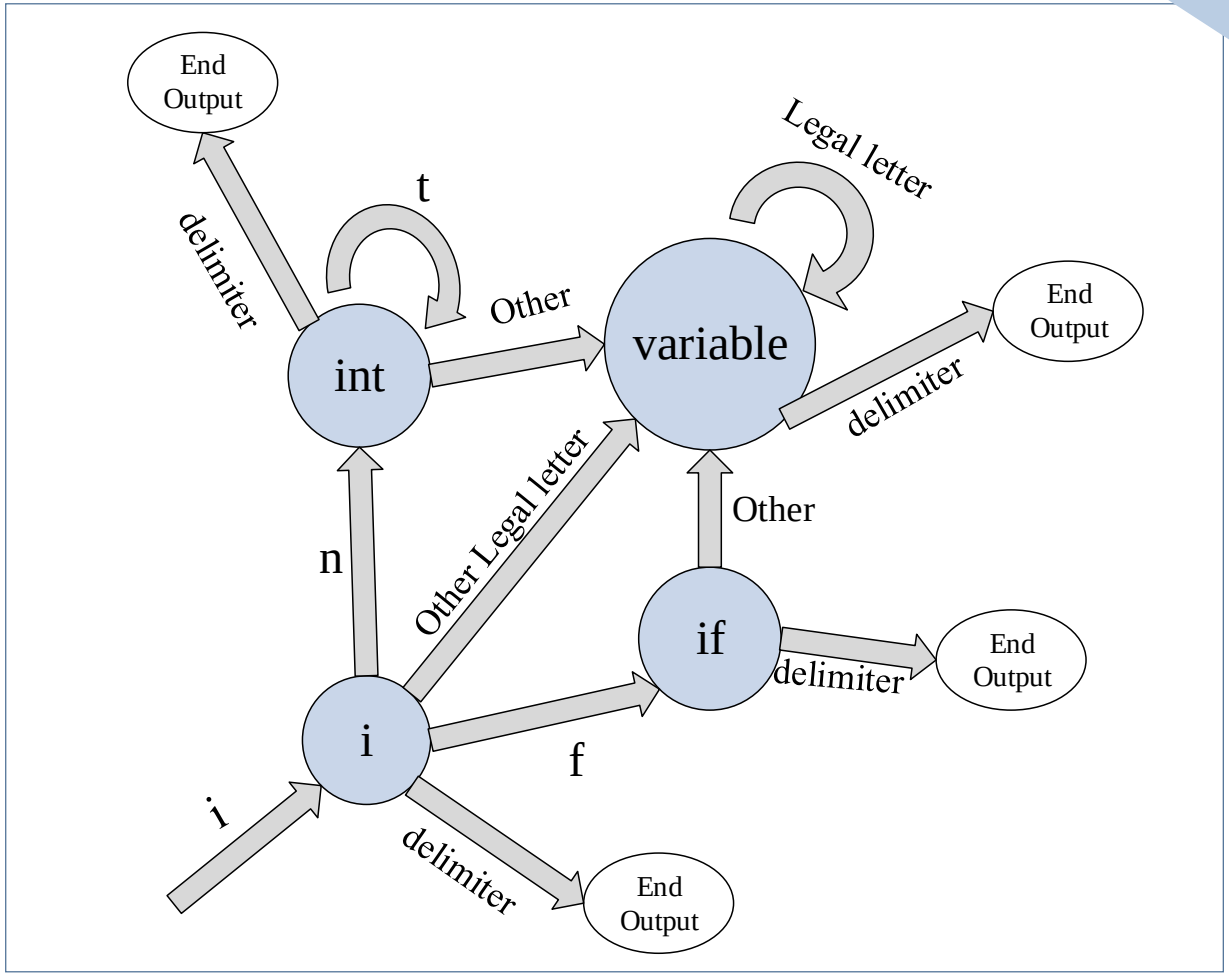
Nondeterministic Finite Automata(NFA)

lexical analysis for C programming language

This is a part of NFA structure.

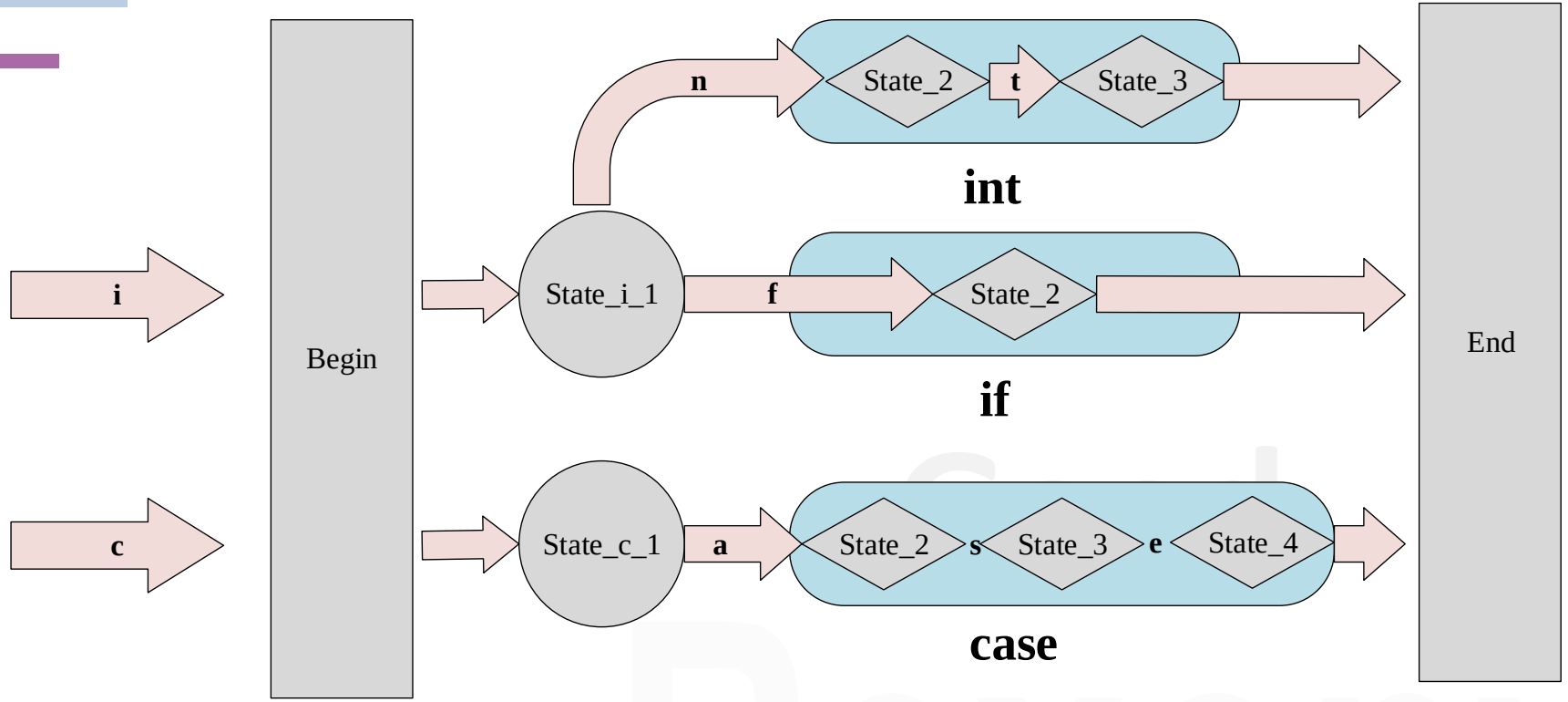
The first letter : ‘i’ ->Goto state ‘i’.
The second letter: ‘n’ ->Goto state ‘int’
The second letter: ‘f’ ->Goto state ‘if’
The second letter: other legal letter(a-z,A-Z,0-9,_)
->Goto state ‘variable’

Now,if into the state ‘int’:
The third letter: ‘t’ ->Goto the next state ‘int’
The third letter: other legal letter(a-z,A-Z,0-9,_)
->Goto state ‘variable’
The third letter: delimiter(+,*,-,/,(space),etc.)
->Goto End input ‘kw_int’



Structure design of NFA

and code implementation



Code In prolog

Path graph of call and recursion

```
/*Finite Automata*/
start(S):-
    name(S,L),
    write(L),nl,
    begin(L-_,_).  % Begin Loop
```

```
%begin:int if
begin([X|Y]-Y,S):-
        [X|Y]\=[],
        X is 105,
        write('begin_i'),nl,
        mover(sta_i_1,Y-_,S),
        write(S),nl,
        begin(S-_,_).
```

```
%i
mover(sta_i_1,[X|Y]-Y,S):-
    delimiter(X),
    write('This is a variable '),nl,
    append(Y,[],S).
```

```
mover(sta_i_1,[X|Y]-Y,S):-
        X is 102,
        write('sta_i_1 '),nl,
        mover(sta_if_2,Y-_,S).
```

```
mover(sta_i_1,[X|Y]-Y,S):-
        X \= 102,X \= 110,
        write('sta_i_1 '),nl,
        mover(sta_var,Y-_,S).
```

```
% End Loop
begin(S-_,_):- S ==[].
```

```
mover(sta_if_2,[X|Y]-Y,S):-
        name_legal(X),
        write('sta_if_2 '),nl,
        mover(sta_var,Y-_,S).
```

```
mover(sta_if_2,[X|Y]-Y,S):-
        delimiter(X),
        write('sta_kw_if'),nl,
        append(Y,[],S),
        write('This is a key word:if'),nl.
```

```
.............
```

SWISH

Rexony

NFA in prolog

Defects and development

1 Too much code

The total of more than 700 lines of code,
The rule ‘mover ()’ is repeated many times.

Reason: I'm not familiar with it. I can't use list to reduce the amount of code through recursion.

2 Development

Mixed with other languages to complete a lexical analyzer.

THANK YOU