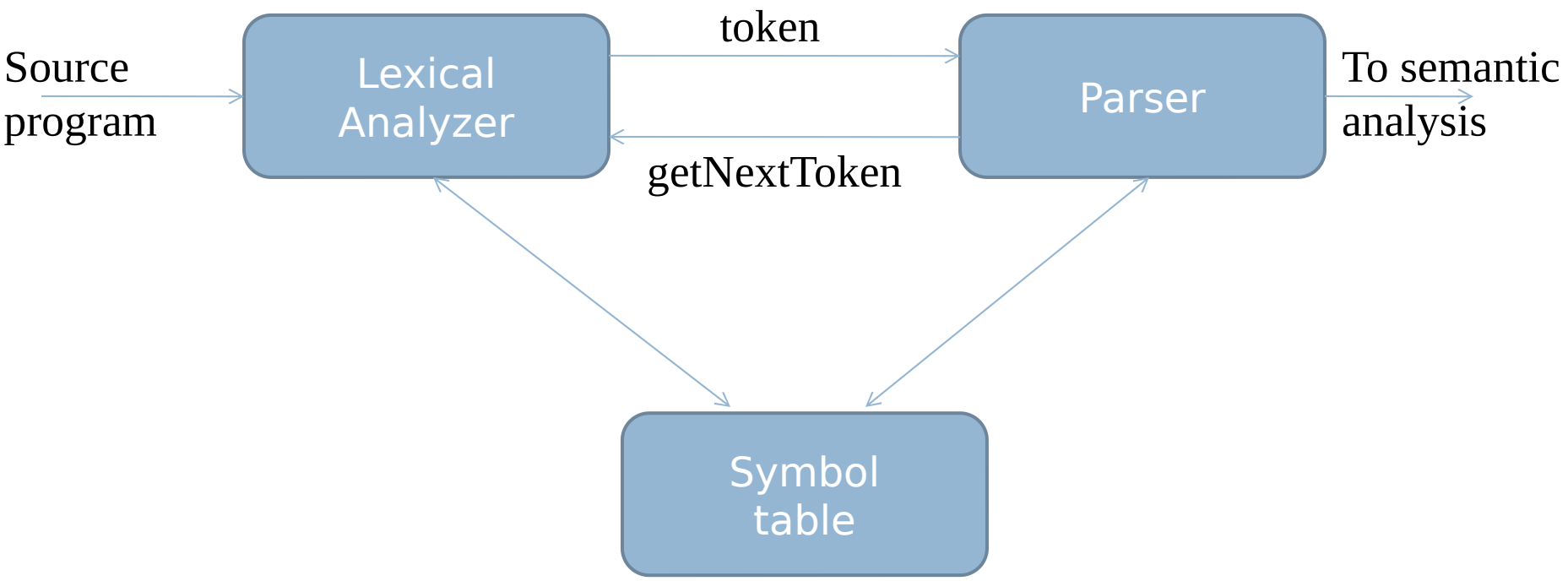
طراحی کامپایلرها

تحلیل لغوی (بخش 3)

قمرناز تدین تبریزی

یادآوری: نقش تحلیلگر لغوی



Example

Token	Informal description	Sample lexemes
if	Characters i, f	if
else	Characters e, l, s, e	else
comparison	< or > or <= or >= or == or !=	<=, !=
id	Letter followed by letter and digits	pi, score, D2
number	Any numeric constant	3.14159, 0, 6.02e23
literal	Anything but “ sorrounded by “	“core dumped”

printf(“total = %d\n”, score);

خصیصه های توکنها

- E = M * C ** 2
  - <id, pointer to symbol table entry for E>
  - <assign-op>
  - <id, pointer to symbol table entry for M>
  - <mult-op>
  - <id, pointer to symbol table entry for C>
  - <exp-op>
  - <number, integer value 2>

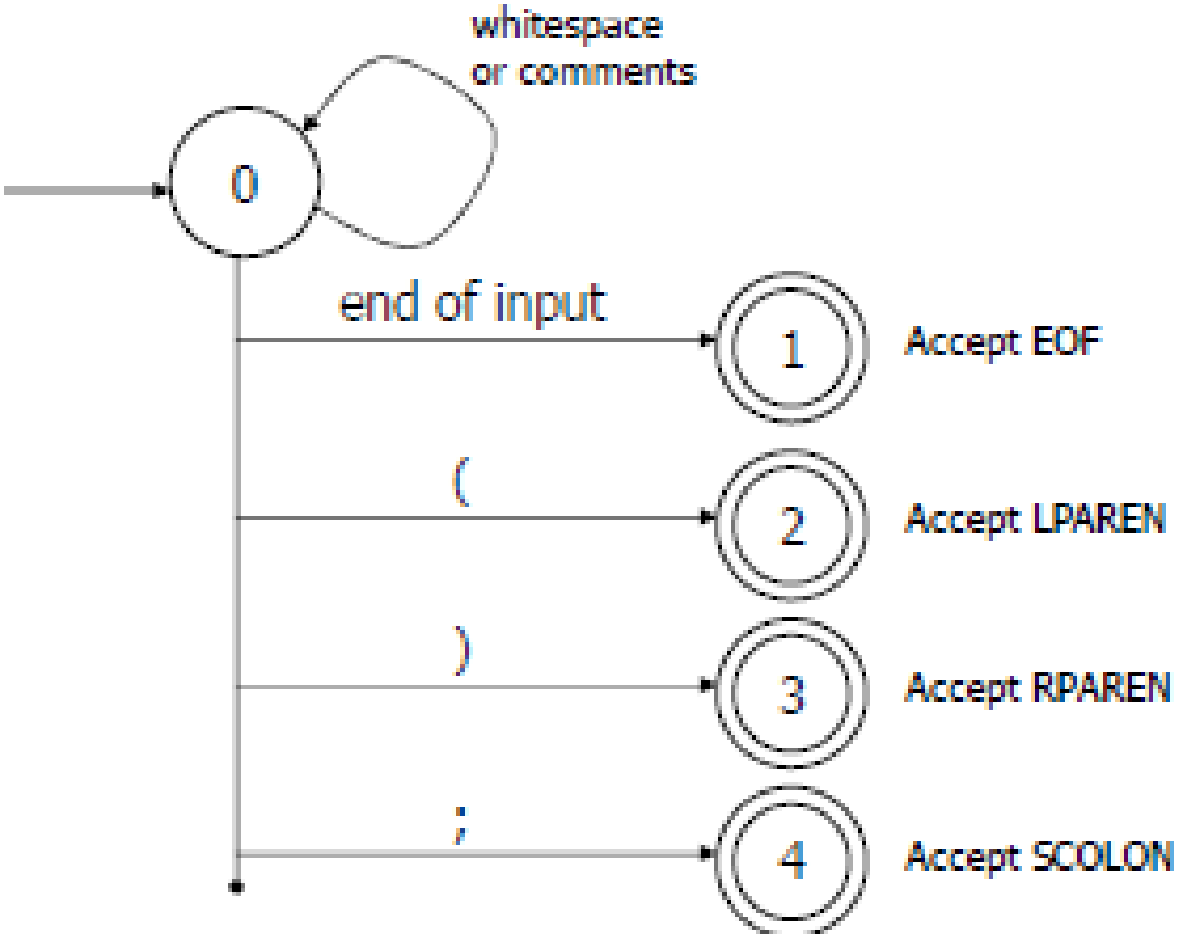
خطاهای لغوی

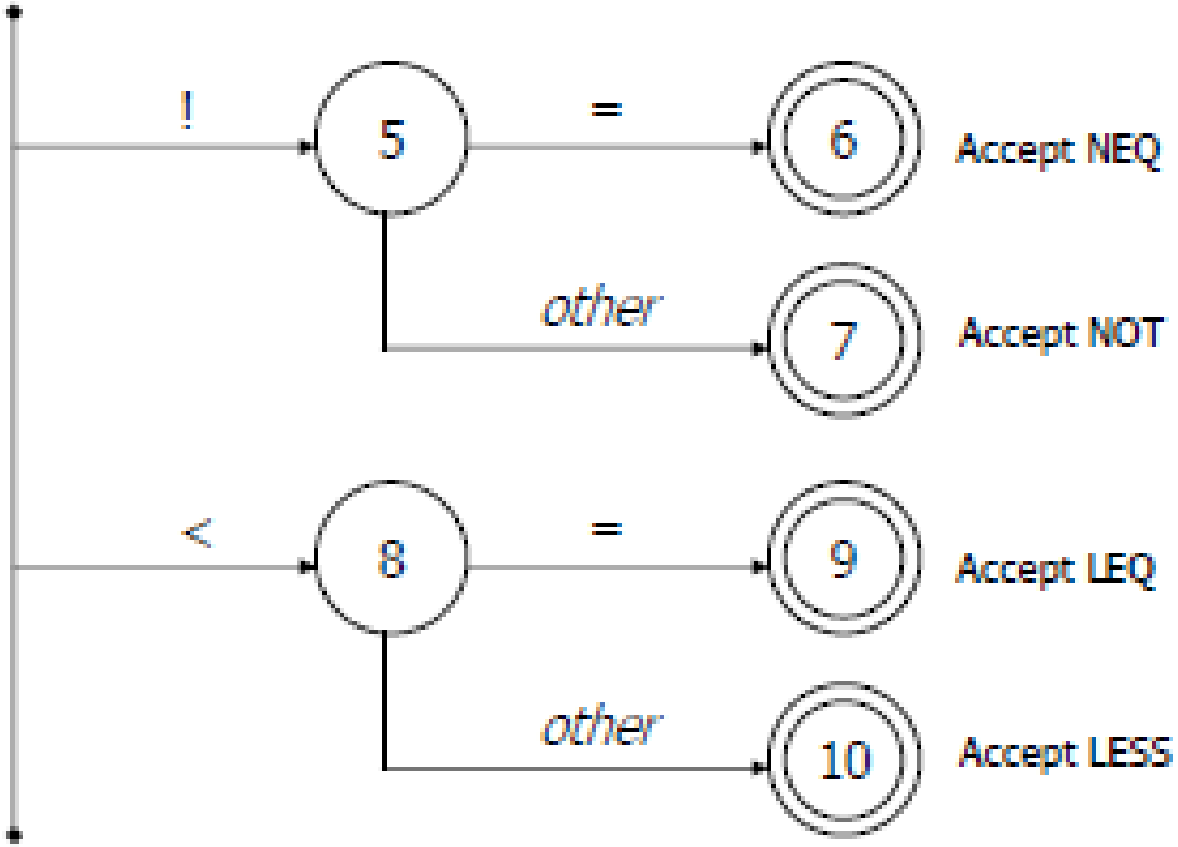
- تشخیص برخی خطاها در توانایی تحلیلگر لغوی نیست:
  - fi (a == f(x)) ...
- خطاهایی مثل مورد زیر قابل تشخیص هستند:
  - d = 2r
- این خطاها زمانی تشخیص داده می شوند که هیچ الگوی توکنی منطبق بر ورودی پیدا نشود.

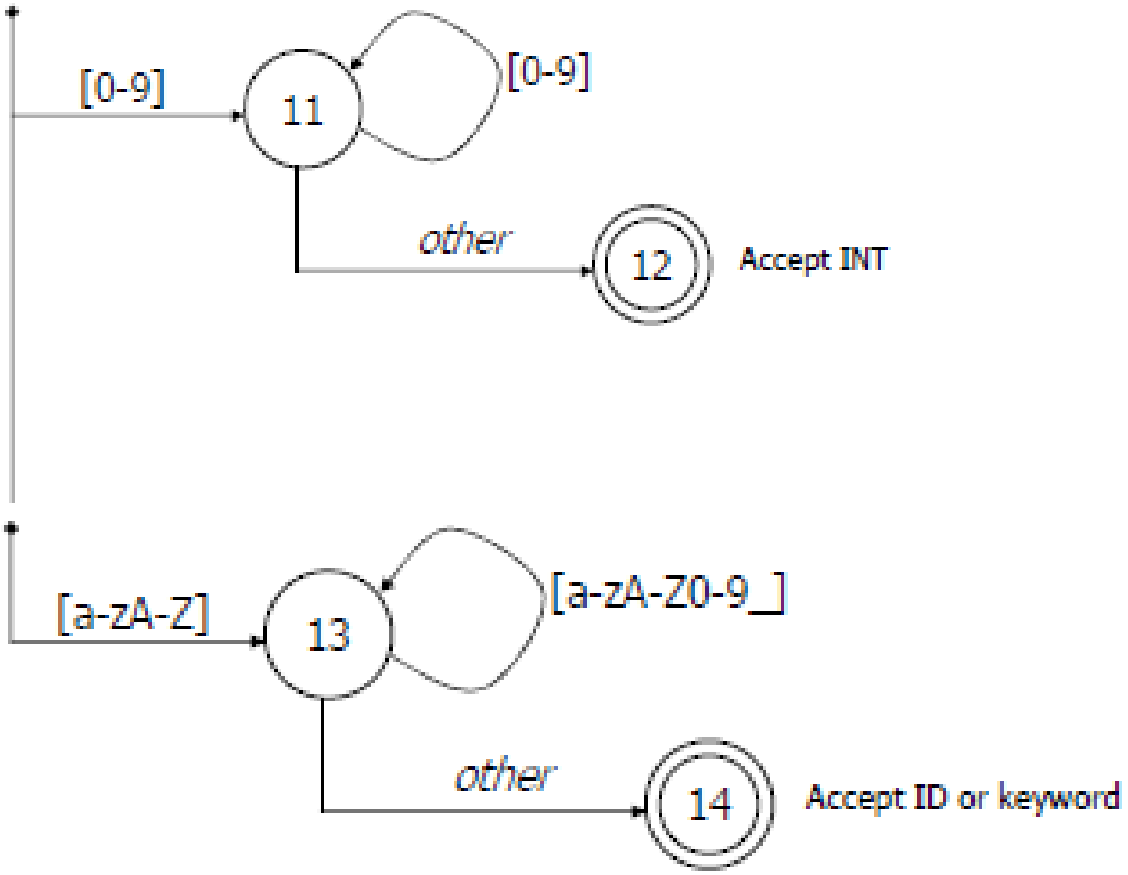
پوشش خطا

- Panic mode : کاراکترهای متوالی تا زمانی که به یک توکن خوش شکل برسد در نظر گرفته نمی شوند.
- یک کاراکتر از باقیمانده وروی حذف می شود.
- کاراکتری که جا افتاده به ورودی باقیمانده درج می شود.
- یک کاراکتر با کاراکتر دیگری جایگزین می شود.
- دو کاراکتر همسایه با هم جابجا می شوند.

مثال


whitespace
or comments
0
end of input
1
Accept EOF
(
2
Accept LPAREN
)
3
Accept RPAREN
;
4
Accept SCOLON


!
5
=
6
Accept NEQ
other
7
Accept NOT
<
8
=
9
Accept LEQ
other
10
Accept LESS


[0-9]
11
[0-9]
other
12
Accept INT
[a-zA-Z]
13
[a-zA-Z0-9_]
other
14
Accept ID or keyword

مثال:

digit -> [0-9]

Digits -> digit+

number -> digit(.digits)? (E[+-]? Digit)?

letter -> [A-Za-z_]

id -> letter (letter|digit)*

If -> if

Then -> then

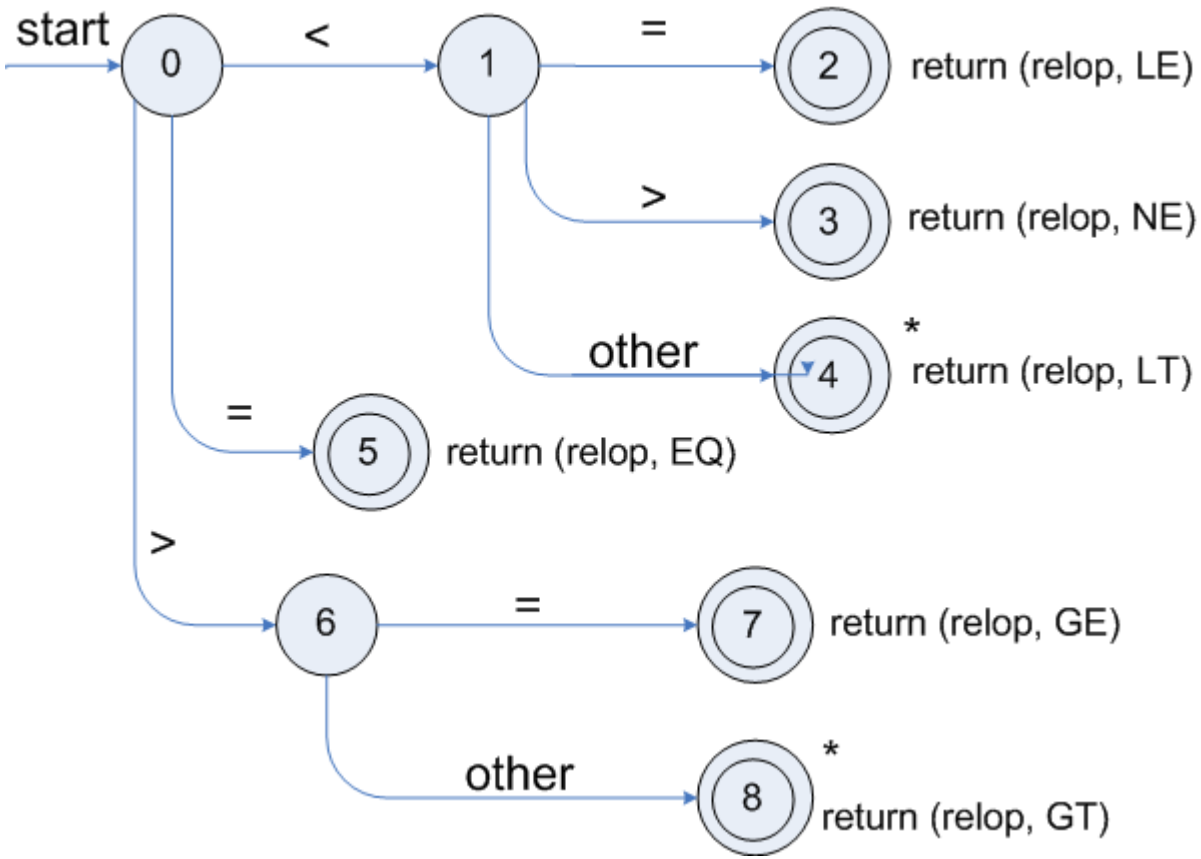
Else -> else

Relop -> < | > | <= | >= | = | <>

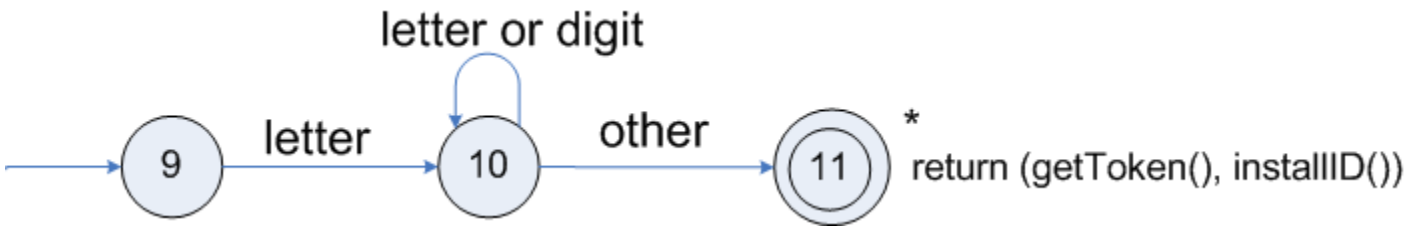
- فضای خالی:

ws -> (blank | tab | newline)+

Transition diagrams

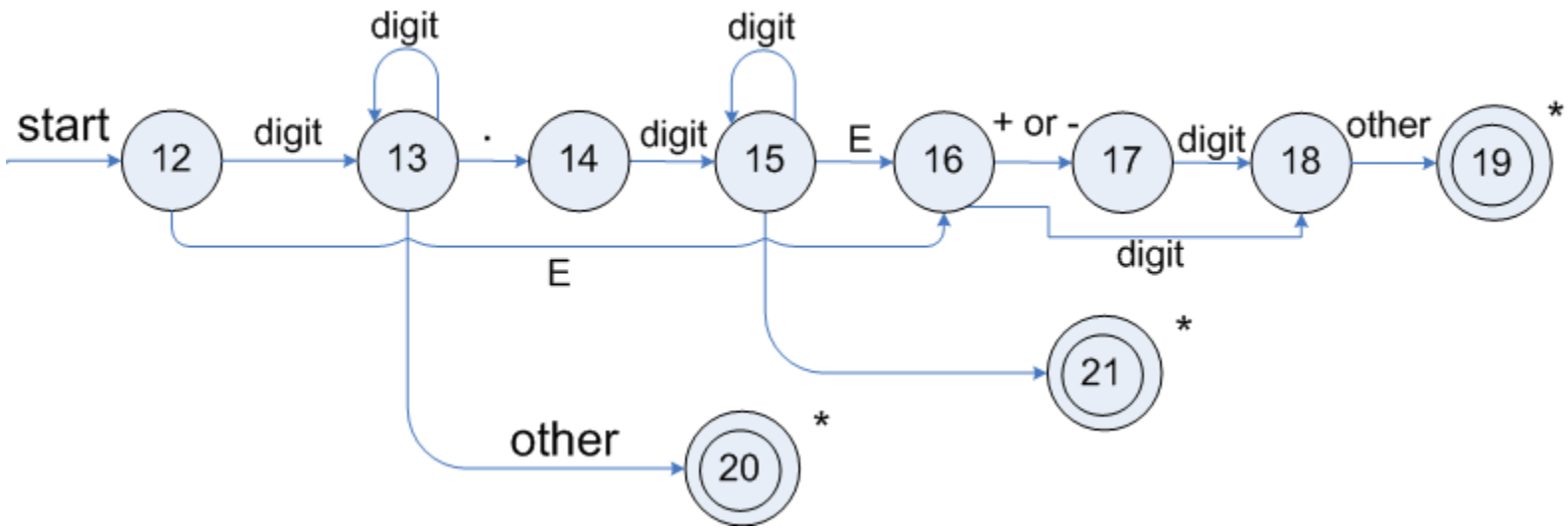


Transition diagrams (cont.)



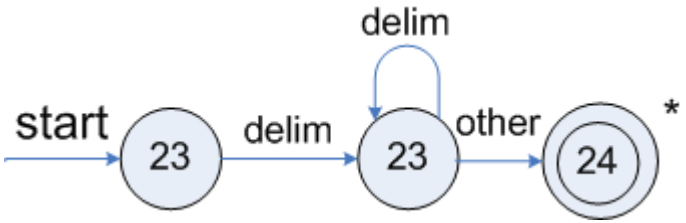
Transition diagrams (cont.)

- اعداد بدون علامت

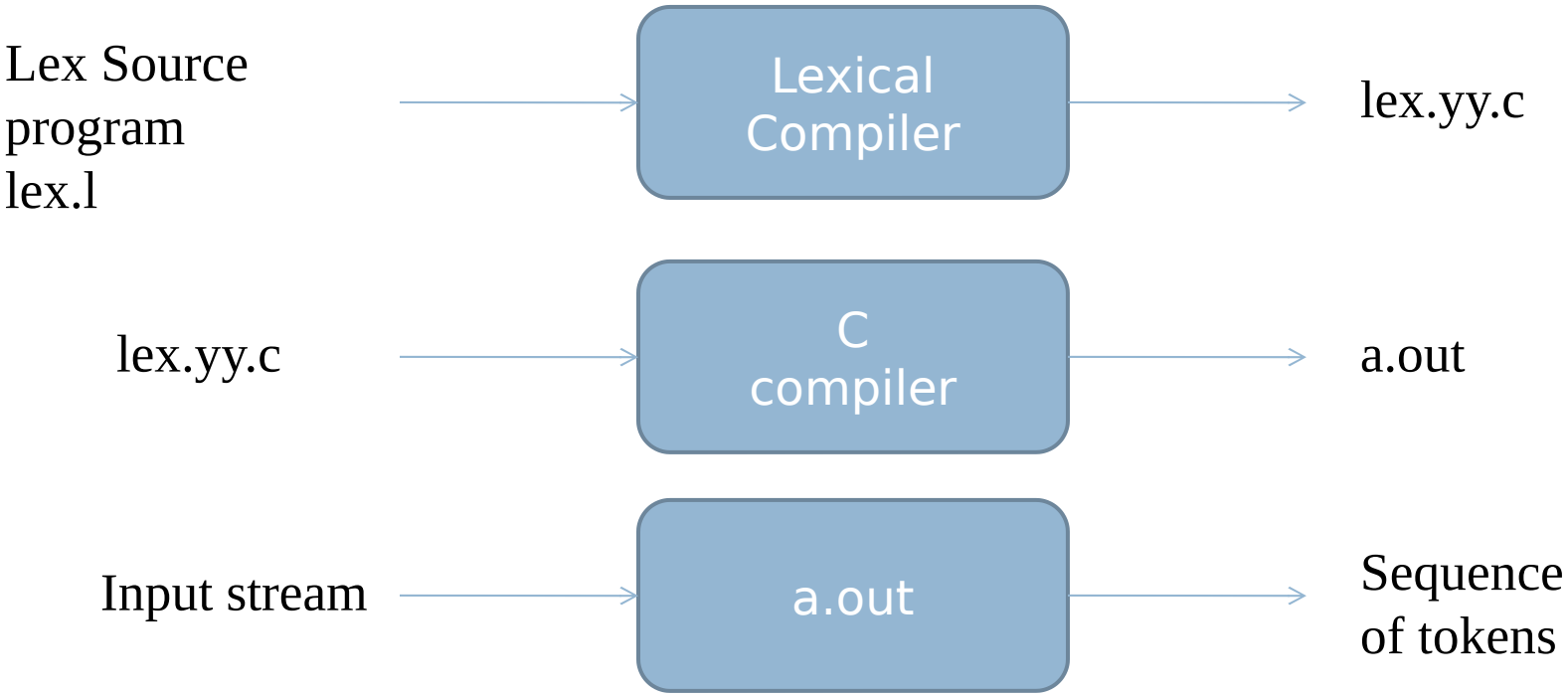


Transition diagrams (cont.)

- فضای سفید



Lexical Analyzer Generator - Lex



Structure of Lex programs

declarations
%%
translation rules
%%
auxiliary functions

Pattern {Action}

Example

```
%{
    /* definitions of manifest constants
    LT, LE, EQ, NE, GT, GE,
    IF, THEN, ELSE, ID, NUMBER, RELOP */
%}

/* regular definitions
delim     [ \t\n]
ws  {delim}+
letter     [A-Za-z]
digit      [0-9]
id   {letter}({letter}|{digit})*
number  {digit}+(\.{digit}+)?(E[+-]?{digit}+)?

%%
{ws}      {/* no action and no return */}
if    {return(IF);}
then      {return(THEN);}
else      {return(ELSE);}
{id}      {yylval = (int) installID(); return(ID); }
{number}      {yylval = (int) installNum(); return(NUMBER);}
...
```

```
Int installID() {/* funtion to
    install the lexeme, whose first
    character is pointed to by
    yytext, and whose length is
    yyleng, into the symbol table
    and return a pointer thereto
    */
}

Int installNum() { /* similar to
    installID, but puts numerical
    constants into a separate
    table */
}
```