Specification of Token Lexical Phase

Lecture 4

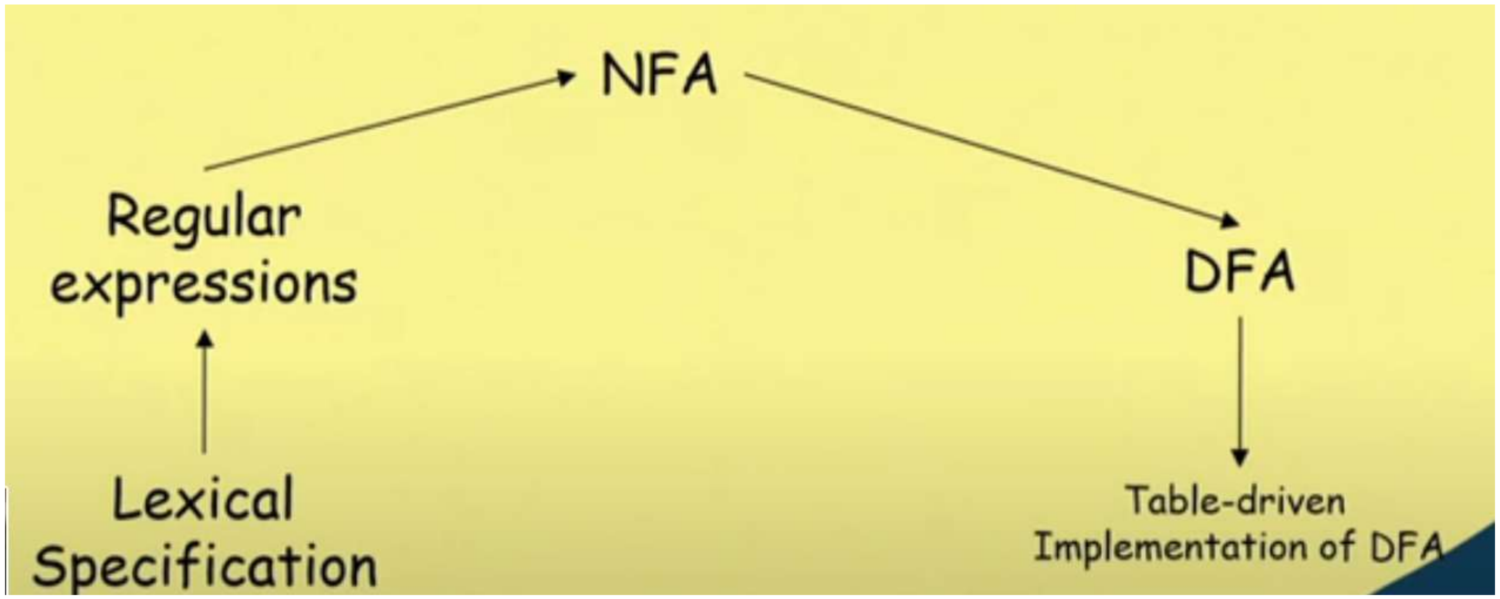
15CSE311 Compiler Design

Department of Computer Science

Scanner from the Specification

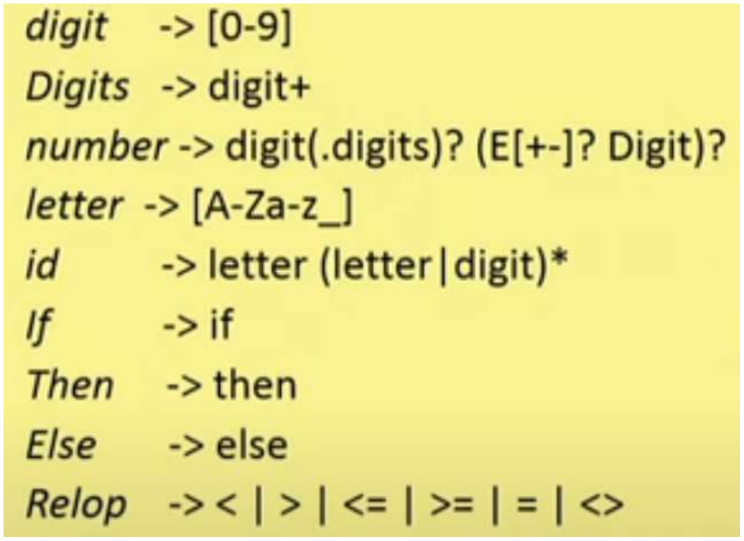
- The collection of tokens of a programming language can be specified by a set of regular expressions.
- A **scanner or lexical analyzer** for the language uses a DFA (recognizer of regular languages) in its core.
- **Different final states** of the DFA identifies different tokens.
- A scanner is a big DFA, essentially the "aggregate" of the automata for the individual tokens.

Lexical Analyser Tool



Recognition of Tokens

- Formalize the pattern for tokens



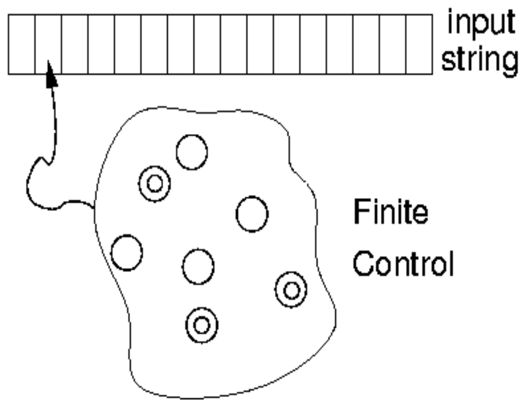
```
digit    -> [0-9]
Digits   -> digit+
number -> digit(.digits)? (E[+-]? Digit)?
letter  -> [A-Za-z_]
id       -> letter (letter|digit)*
If       -> if
Then     -> then
Else     -> else
Relop    -> < | > | <= | >= | = | <>
```

- We also need to handle whitespace

Recognizing Tokens: Finite Automata

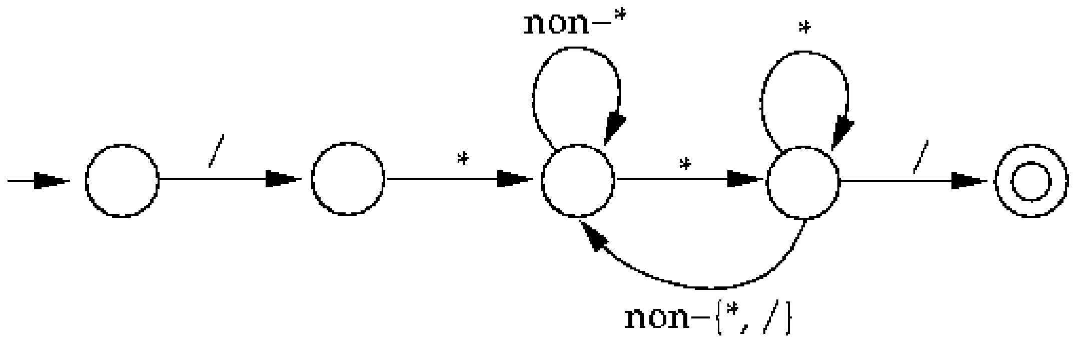
A *finite automaton* is a 5-tuple (Q, Σ, T, q_0, F), where:

- Σ is a finite alphabet;
- Q is a finite set of states;
- T: $Q \times \Sigma \rightarrow Q$ is the transition function;
- $q_0 \in Q$ is the initial state; and
- $F \subseteq Q$ is a set of final states.

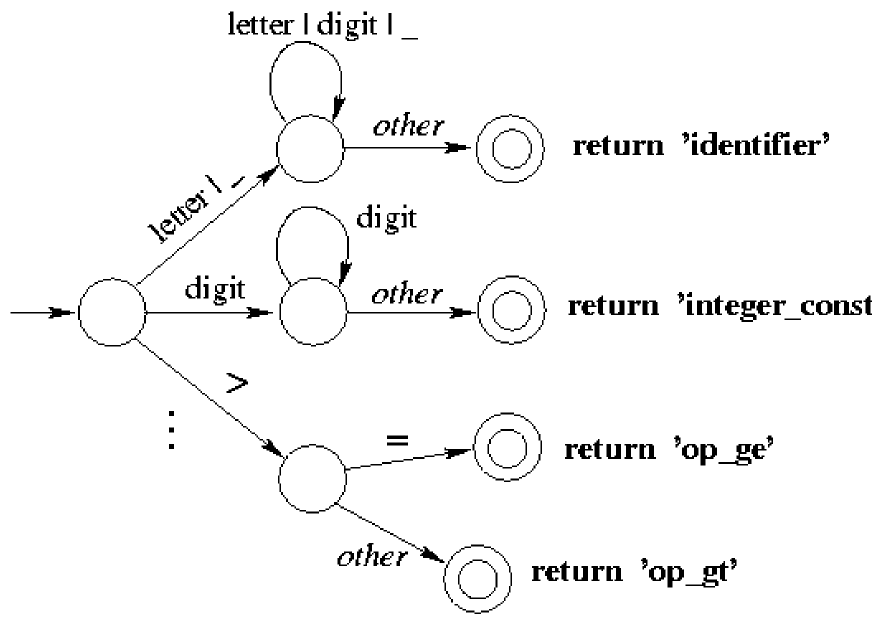


Finite Automata: An Example

A (deterministic) finite automaton (DFA) to match C-style comments:



Structure of a Scanner Automaton



How much should we match?

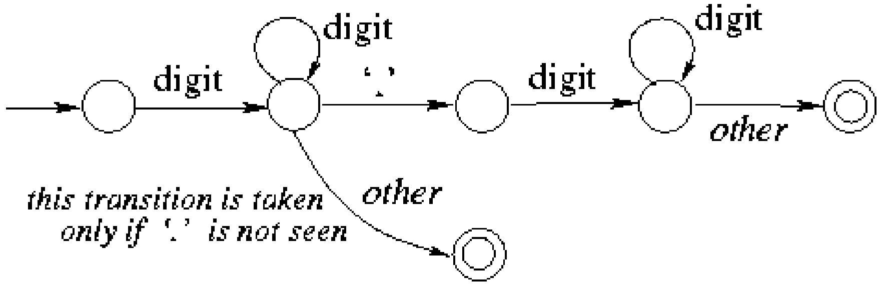
In general, find the longest match possible.

E.g., on input 123.45, match this as

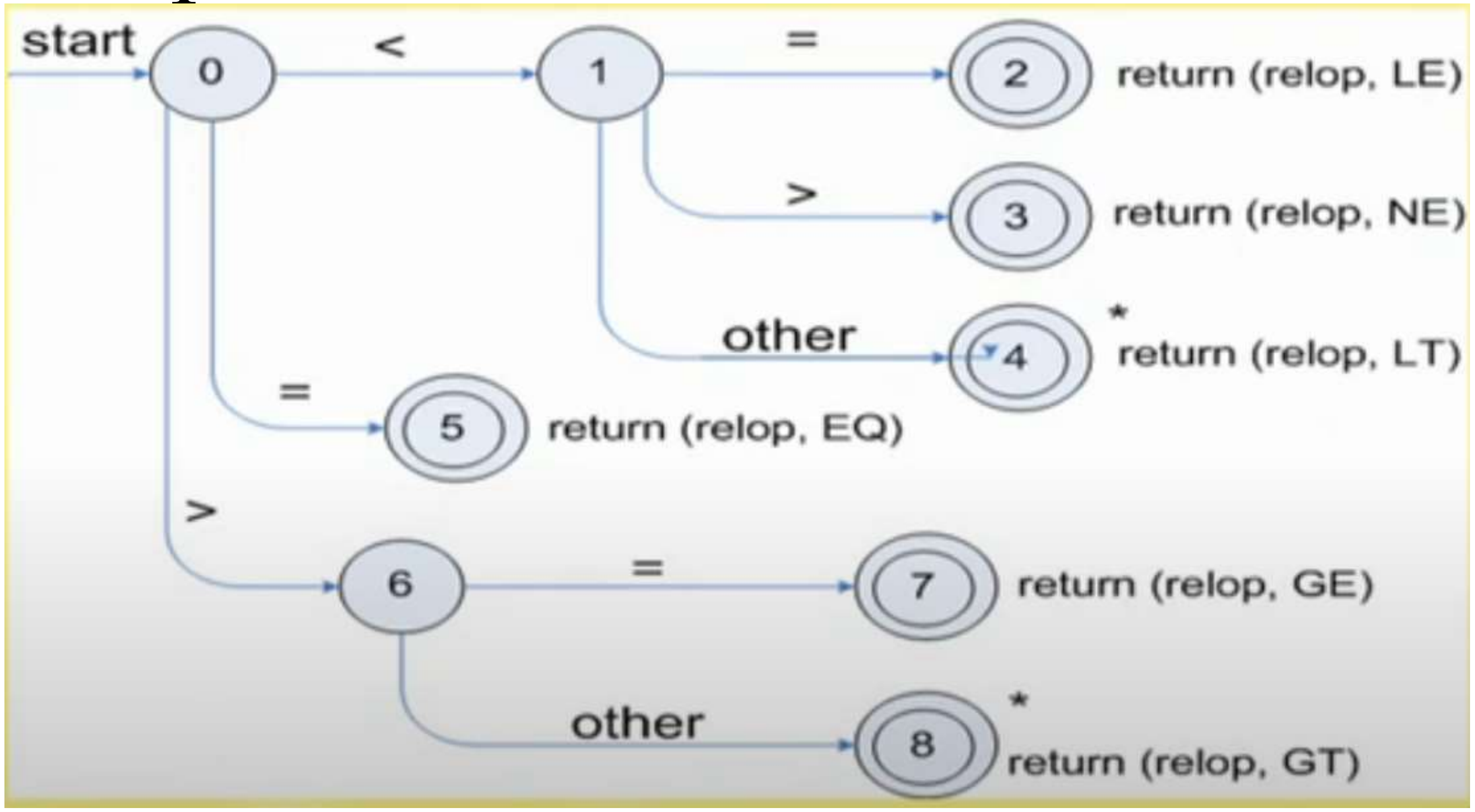
num_const(123.45)

rather than

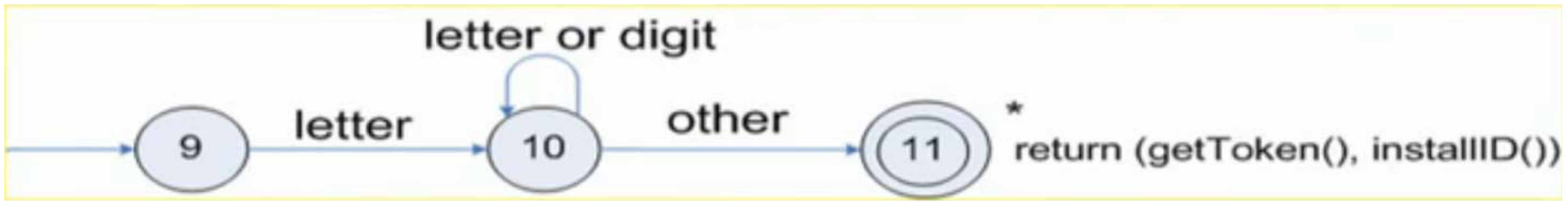
num_const(123), ".", num_const(45).



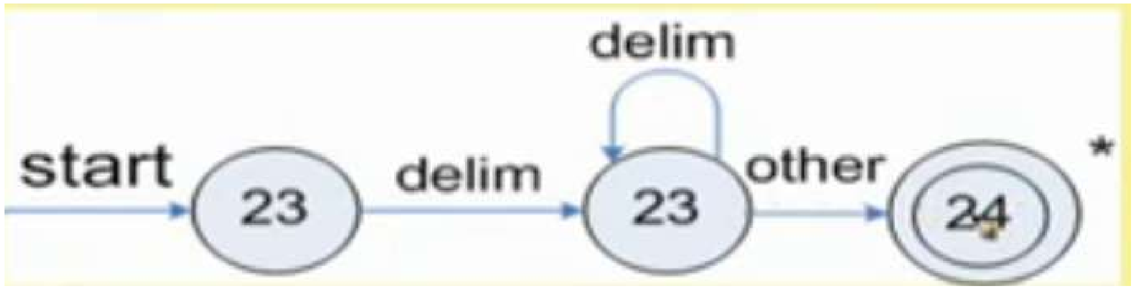
Transition Diagram for Relational Operator



Transition Diagram for Identifiers



Transition of whitespaces



Example

```
if                                          { return IF; }
[a-z][a-z0-9]*                              { return ID; }
[0-9]+                                      { return NUM; }
[0-9]”.”[0-9]*|[0-9]*”.”[0-9]+         { return REAL; }
(\-\-[a-z]*\n)|(“  “|\n|\t)             { ; }
.                                           { error(); }
```

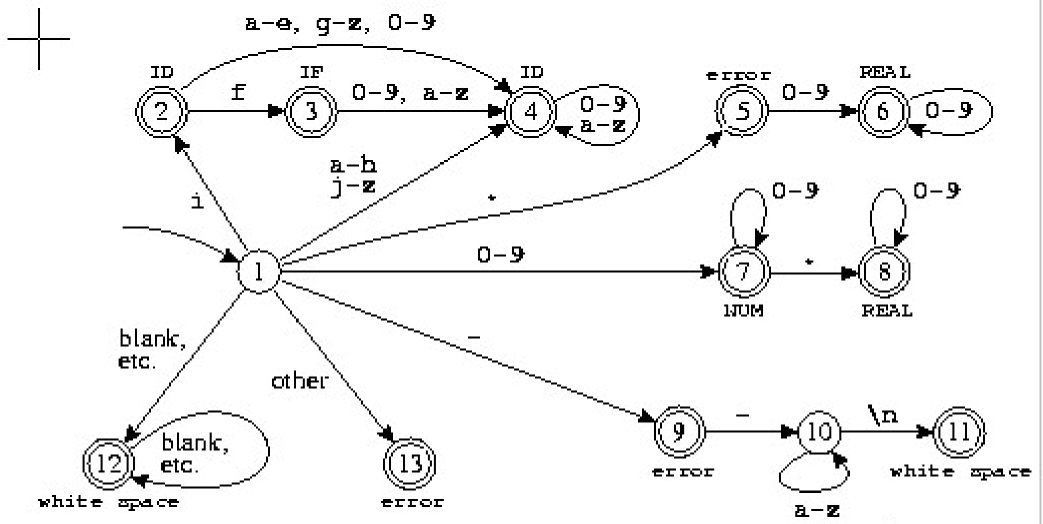


FIGURE 2.4. Combined finite automaton.
From *Modern Compiler Implementation in ML*,
Cambridge University Press, ©1998 Andrew W. Appel

Theory vs. Practice

- Two differences:
- DFAs recognize lexemes. A lexer must return a type of acceptance (token type) rather than simply an accept/reject indication.
- DFAs consume the complete string and accept or reject it.

A lexer must find the end of the lexeme in the input stream and then find the next one, etc.