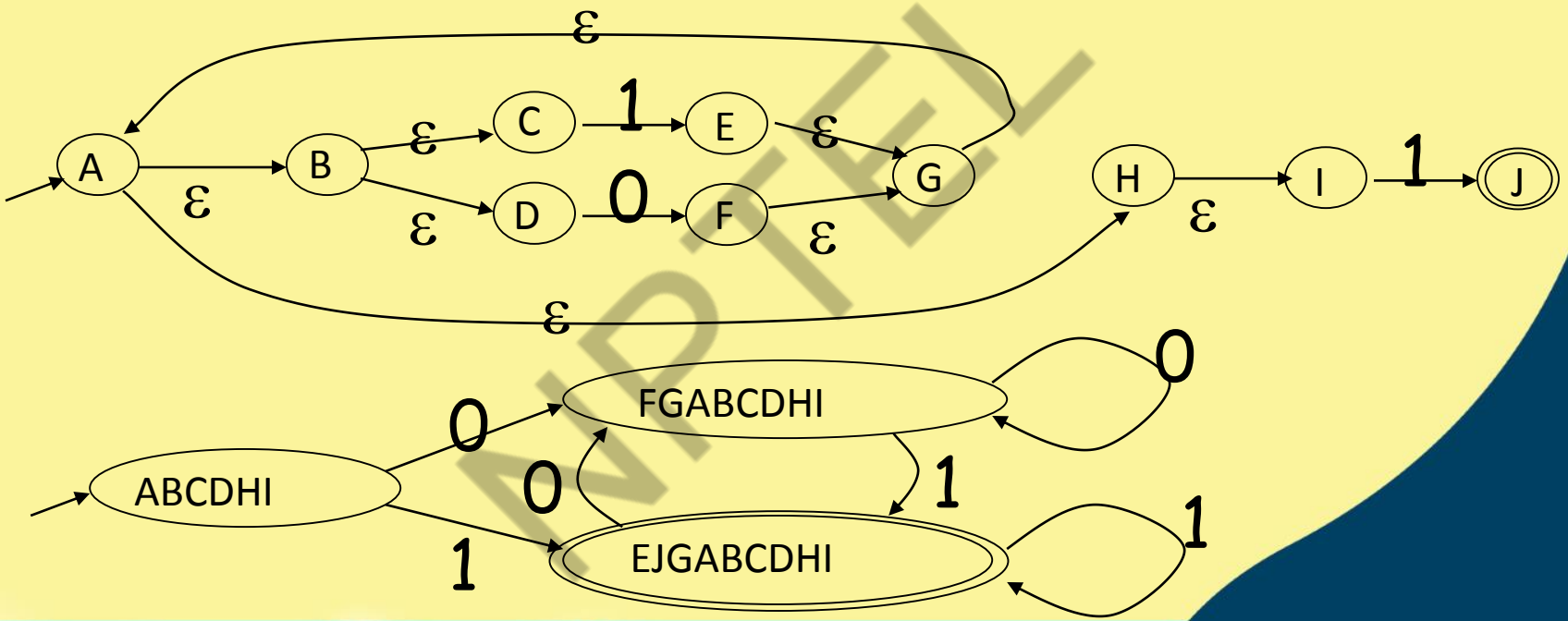
NFA -> DFA Example



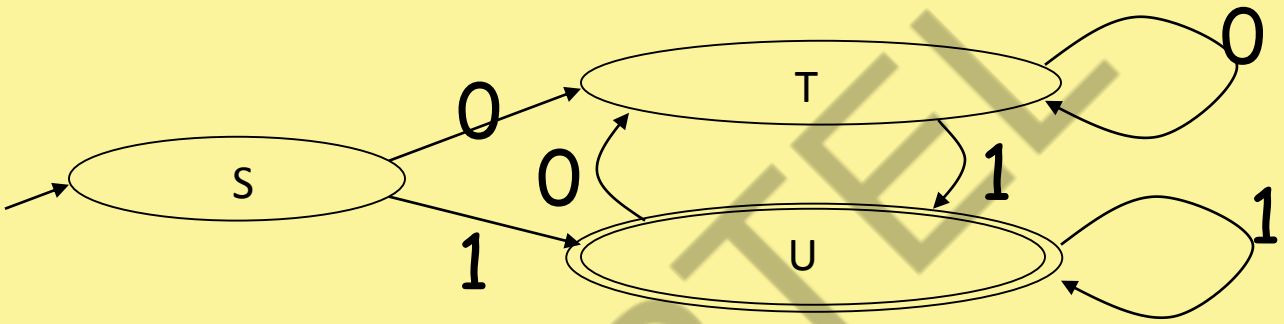
NFA to DFA. Remark

- An NFA may be in many states at any time
- How many different states ?
- If there are N states, the NFA must be in some subset of those N states
- How many non-empty subsets are there?
  - $2^N - 1$ = finitely many, but exponentially many

Implementation

- A DFA can be implemented by a 2D table T
  - One dimension is “states”
  - Other dimension is “input symbols”
  - For every transition $S_i \rightarrow^a S_k$ define T[i,a] = k
- DFA “execution”
  - If in state S_i and input a, read T[i,a] = k and skip to state S_k
  - Very efficient

Table Implementation of a DFA

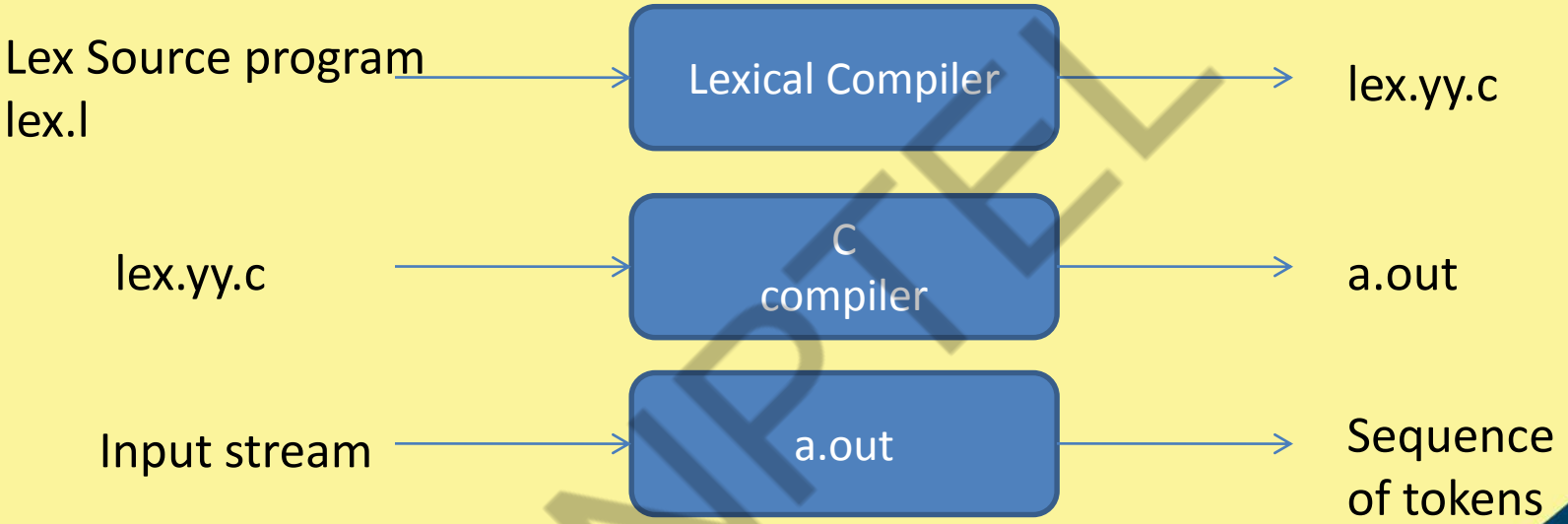


	0	1
S	T	U
T	T	U
U	T	U

Implementation (Cont.)

- NFA -> DFA conversion is at the heart of tools such as lex, flex or jflex
- But, DFAs can be huge
- In practice, lex-like tools trade off speed for space in the choice of NFA and DFA representations

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 */
}
```

Conclusion

- Words of a language can be specified using regular expressions
- NFA and DFA can act as acceptors
- Regular expressions can be converted to NFA
- NFA can be converted to DFA
- Automated tool lex can be used to generate lexical analyser for a language

FREE ONLINE EDUCATION
swayam
शिक्षित भारत, उन्नत भारत

NPTEL ONLINE CERTIFICATION COURSES

Thank you!

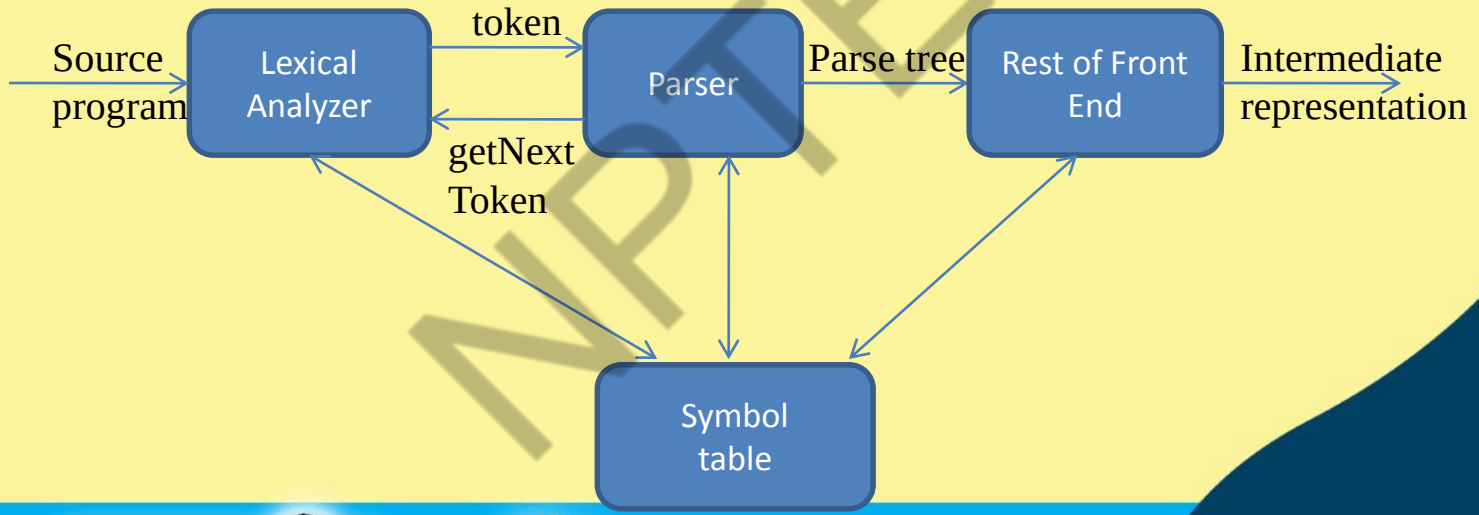
Compiler Design

Syntax Analysis

Santanu Chattopadhyay
Electronics and Electrical Communication Engineering

- Role of Parsers
- Context Free Grammars
- Top-Down Parsing
- Bottom-Up Parsing
- Parser Generators
- Conclusion

The role of parser



Grammar

- A 4-tuple G = <V_N, V_T, P, S> of a language L(G)
  - V_N is a set of nonterminal symbols used to write the grammar
  - V_T is the set of terminals (set of words in the language L(G))
  - P is a set of production rules
  - S is a special symbol in V_N, called the start symbol of the grammar
- Strings in language L(G) are those derived from S by applying the production rules from P
- Examples:

E -> E + T | T
T -> T * F | F
F -> (E) | **id**

E -> TE’
E’ -> +TE’ | Ɛ
T -> FT’
T’ -> *FT’ | Ɛ
F -> (E) | **id**