Compilers Techniques
Lecture4
المحاضرة الرابعة

Reduce DFA and code the lexical analyser using LEX

اختصار DFA وبناء المحلل المعجمي برمجياً باستخدام LEX

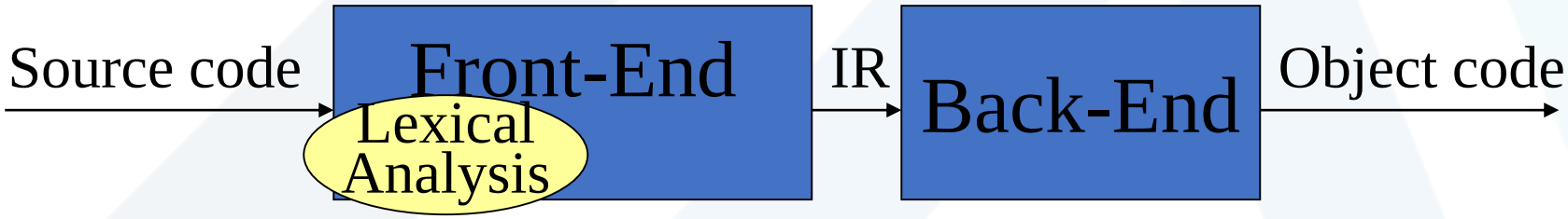
Reference: Aho2 Sections 3.6-3.7; Aho1 pp. 113-125; Grune 2.1.6.1-2.1.6.6 (different style); Hunter 3.3 (very condensed); Cooper1 2.4-2.4.3

السنة الرابعة – المستوى السابع- الهندسة المعلوماتية

Lectures Topics

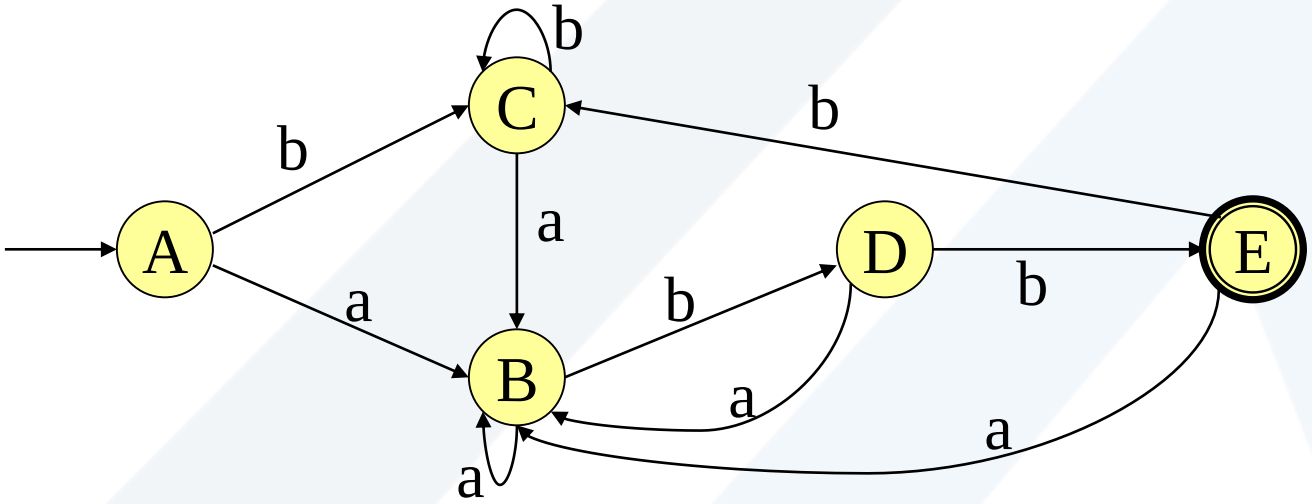
كيفية اختصار الـDFA الناتجة.

اعتبارات عملية باستخدام lex.



اختصار DFA

- **السؤال:** هل نستطيع اختصار عدد الحالات؟
- **الجواب:** نعم، إذا استطعنا إيجاد مجموعات من الحالات حيث، يسبب كل رمز من رموز الدخل انتقالات من كل حالة من حالات هذه المجموعة إلى نفس المجموعة.



التحليل المعجمي Lexical Analysis

النسخة المبسطة من (Hopcroft's algorithm)

تقسيم حالات الـ DFA إلى **مجموعتين**: **الأولى**: تلك التي تحوي الحالات النهائية؛ **والأخرى**: تلك التي لا تحوي الحالات النهائية non-final states.

while there are group changes
  for each group
    for each input symbol
      if for any two states of the group and a given input symbol, their transitions do not lead to the same group, these states must belong to different groups.

لكل مجموعة ولكل رمز دخل، في حال أن هناك حالتين من المجموعة لهما انتقالين عن طريق رمز ما وكان هذين الانتقالين لا يؤديان إلى ذات المجموعة عندها يجب فصلهما لمجموعتين مختلفتين.

هنالك مقاربة بديلة: إنشاء بيان يضم حرف بين كل زوج من أزواج الحالات التي لايمكن أن توجد معاً في مجموعة بسبب التناقض أو التباين المذكور أعلاه. بعدها نستخدم مسألة خوارزمية تلوين البيان لإيجاد العدد الأصغري من الألوان التي نحتاجها، بحيث لا يكون للعقدتين المرتبطين بحرف نفس اللون.

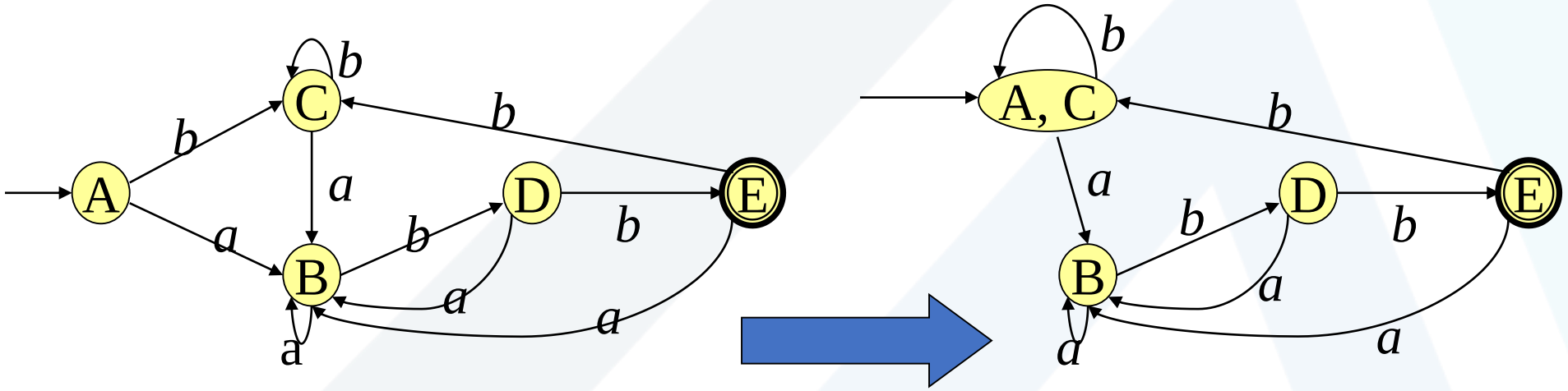
التحليل المعجمي Lexical Analysis

اختصار DFA – مثال

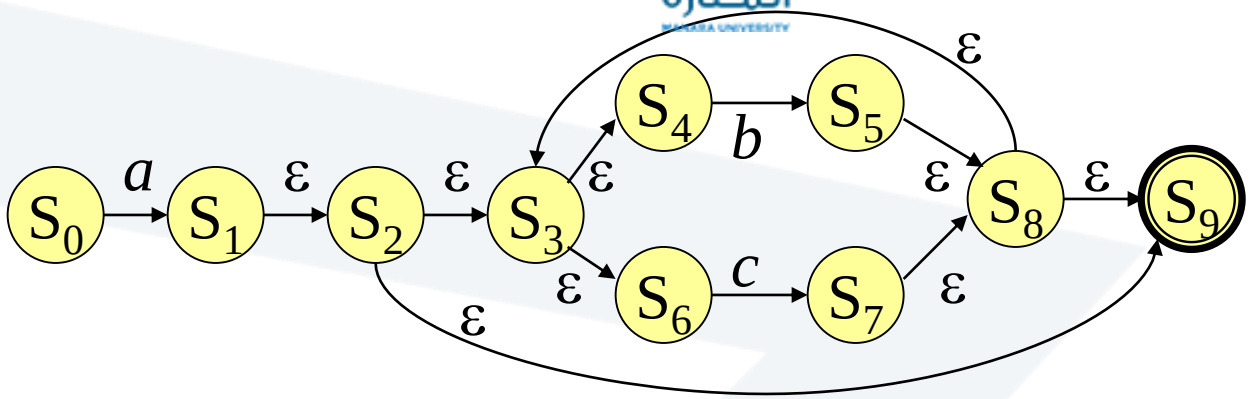
(a | b) abb*

نأخذ بعين الاعتبار أية مجموعة لا تحوي عنصر وحيد في كل تكرار وندرس معايير التجزيء لجميع أزواج الحالات في المجموعة.

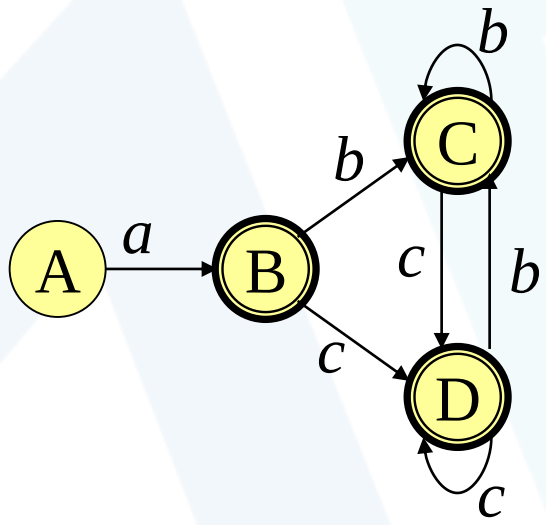
Iteration	Current groups	Split on a	Split on b
0	{E}, {A,B,C,D}	None	{A,B,C}, {D}
1	{E}, {D}, {A,B,C}	None	{A,C}, {B}
2	{E}, {D}, {B}, {A, C}	None	None



من NFA إلى DFA المختصر:
المثال a (b | c)*



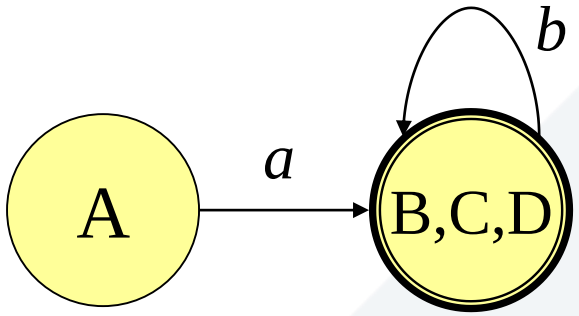
DFA states	NFA states	ε-closure(move(s,*))		
		a	*b*	*c*
A	S0	S1,S2,S3, S4,S6,S9	None	None
B	S1,S2,S3, S4,S6,S9	None	S5,S8,S9, S3,S4,S6	S7,S8,S9, S3,S4,S6
C	S5,S8,S9, S3,S4,S6	None	S5,S8,S9, S3,S4,S6	S7,S8,S9, S3,S4,S6
D	S7,S8,S9, S3,S4,S6	None	S5,S8,S9, S3,S4,S6	S7,S8,S9, S3,S4,S6

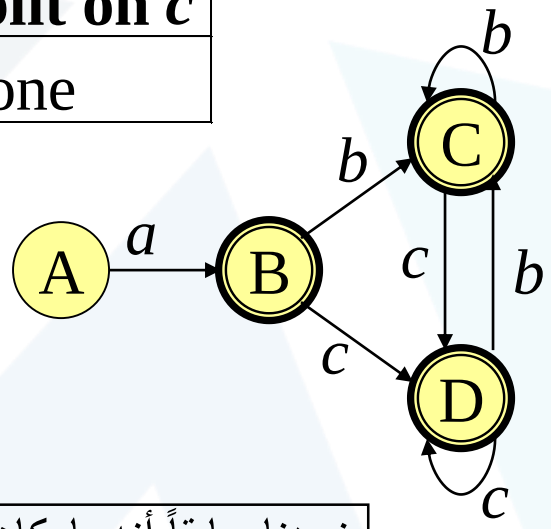


من NFA إلى DFA المختصر:
المثال a (b | c)*

تطبيق خوارزمية الاختصار لتوليد DFA الأصغري.

	Current groups	Split on *a*	Split on *b*	Split on *c*
0	{B, C, D} {A}	None	None	None





نوهنا سابقاً أنه بإمكان المستخدم أن يبني آلية أكثر بساطة من بنية Thompson.
يمكن أن يؤدي تطبيق الخوارزميات إلى توليد نفس الـDFA!.

عند انتقالنا من RE إلى DFA؛ قمنا بما يأتي

- إذا كان جدول الانتقال معلوماً والحالة (أو الحالات النهائية) معلومة، يمكننا مباشرةً بناء مميز Recognizer ليكشف قبول الدخل أو رفضه:

```
char=input_char();
state=0;     // starting state
while (char != EOF) {
     state ← table(state,char);
     if (state == '-') return failure;
     word=word+char;
     char=input_char();
}
if (state == FINAL) return acceptance; else return failure;
```

إلاّ أن المميزات المقادة بالجداول يمكن أن تضيع الكثير من الجهد المبذول.

- يمكننا أن نحسن ذلك بواسطة ترميز الحالات والعمليات في الشيفرة السابقة.

كيف يمكننا التعامل مع العديد من الكلمات المفتاحية؟

- بعض المترجمات تميز الكلمات المفتاحية كمعرفات identifiers ومن ثم تختبرها في جدول.
- **الأسهل** تضمينها كـ REs في مرحلة تخصيص المحلل المعجمي.

الترميز المباشر انطلاقاً من جدول الانتقال

التحليل المعجمي Lexical Analysis

العلاقة $Register \rightarrow r\ digit\ digit^*$ (من المحاضرة السابقة)

```
      word=''; char=''; goto s0;
   s0: word=word+char;
       char=input_char();
       if (char == 'r') then goto s1 else goto error;
   s1: word=word+char;
       char=input_char();
       if ('0' ≤ char ≥ '9') then goto s2 else goto error;
   s2: word=word+char;
       char=input_char();
       if ('0' ≤ char ≥ '9') then goto s2
          else if (char== EOF) return acceptance
               else goto error
error: print "error"; return failure;
```

- عمليات أقل.
- تجنب عمليات الذاكرة (لهذا أهمية خاصة عند وجود جداول كبيرة).
- قد تؤدي إلى زيادة تعقيد الشيفرة (تصبح غير مقروءة وصعبة الفهم).

التحليل المعجمي Lexical Analysis

- **قد يعقد التصميم غير الجيد للغة عملية التحليل المعجمي.**
  - `if then then = else; else else = then` (PL/I)
  - `DO5I=1,25` بدلاً من `DO5I=1.25` (فورتران: سبب فشل رحلة NASA)
  - ينعكس تطوير أسس نظرية متينة إيجابياً على تصميم اللغات.
- **قالب قواعدي في C++:**
  - `aaaa<mytype>`
  - `aaaa<mytype<int>>` (<< معامل الإدخال من لوحة المفاتيح في لغة C++)
  - يعامل المحلل المعجمي << كرمزين < متتاليين. حيث يحل الإشكال من قبل المعرب parser (من خلال مطابقة ، > و <)

التحليل المعجمي Lexical Analysis

Lex/Flex: توليد المحللات الإعرابية

Flex هو أداة لتوليد الماسحات: البرامج التي تميز النماذج المعجمية في النص

- يتألف دخل Lex من 3 مقاطع:
  - التعابير النظامية regular expressions؛
  - أزواج التعابير النظامية وشيفرة C؛
  - شيفرة C المساعدة auxiliary C code.
- عندما تتم ترجمة دخل الـ lex، تولد برنامج C مصدري كخرج lex.yy.c الذي يحتوي على البرنامج yylex() والذي يعمل كماسح لرموز الدخل وفق التعابير المنتظمة التي تم تحديدها في ملف وصف الماسح Scanner Description File.
- وبعد ترجمة ملف C سيبدأ الجزء التنفيذي بعزل العلاّمات tokens من الدخل وفقاً للتعابير النظامية، وسينفذ من أجل كل علاّم، الشيفرة المرفقة به. ويضمن النسق array char yytext[] تمثيلات العلامات token.

Lexical Analysis التحليل المعجمي

Code Example (LEX)

```
%{
#define ERROR -1
int line_number=1;
%}
whitespace    [ \t]
letter        [a-zA-Z]
digit         [0-9]
integer       ({digt}+)
l_or_d        ({letter}|{digit})
identifier    ({letter}{l_or_d}*)
operator      [-+*/]
separator     [;,(){}]
%%
{integer}     {return 1;}
{identifier}  {return 2;}
{operator}|{separator}  {return (int)yytext[0];}
{whitespace}  {}
\n            {line_number++;}
.             {return ERROR;}
%%
int yywrap(void) {return 1;}
int main() {
    int token;
    yyin=fopen("myfile","r");
    while ((token=yylex())!=0)
       printf("%d %s \n", token, yytext);
    printf("lines %d \n",line_number);
}
```

Input file ("myfile")

```
123+435+34=aaaa
329*45/a-34*(45+23)**3
bye-bye
```

Output:

```
 1 123
43 +
 1 435
43 +
 1 34
-1 =
 2 aaaa
 1 329
42 *
 1 45
47 /
 2 a
45 -
 1 34
42 *
40 (
 1 45
43 +
 1 23
41 )
42 *
42 *
 1 3
 2 bye
45 -
 2 bye
lines 4
```