الآليات المحدودة

FINITE AUTOMATA

FINITE AUTOMATON آلية محدودة

اختبار

1. ماهي لغة التعبير النظامي Regular Expression التالي:

(c|f|r|m)(at)

cat
fat
rat
mat

آليات الحالات المحدودة

بعد أن تعرفت على التعبيرات النظامية regular expressions،

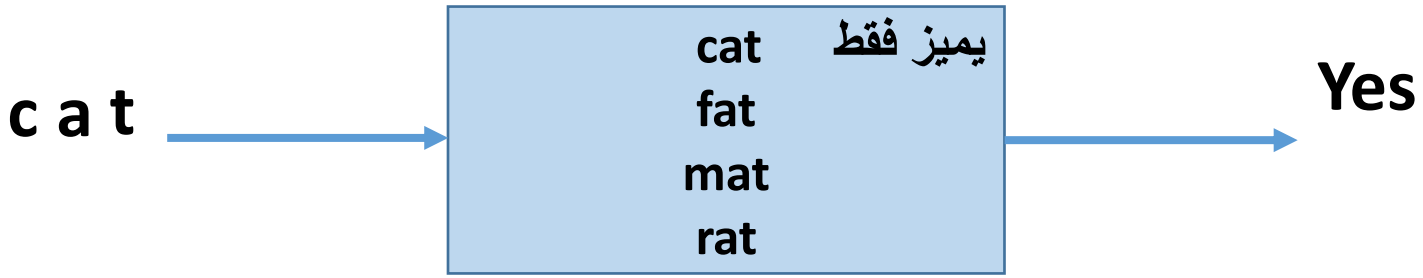
- كيف يمكن أن نقرر بأن نص ما string يتبع نمط تعبير نظامي معين؟
- يستخدم برامج تعرف بالآليات المحدودة يمكنها التمييز بين أنماط النصوص.
- تسمى بآلات/آليات الحلات المحدودة finite state machines
- وتسمى Recognizers أو Acceptors
- أي تميز النص أو تقبل/توافق على النص

آليات الحالات المحدودة

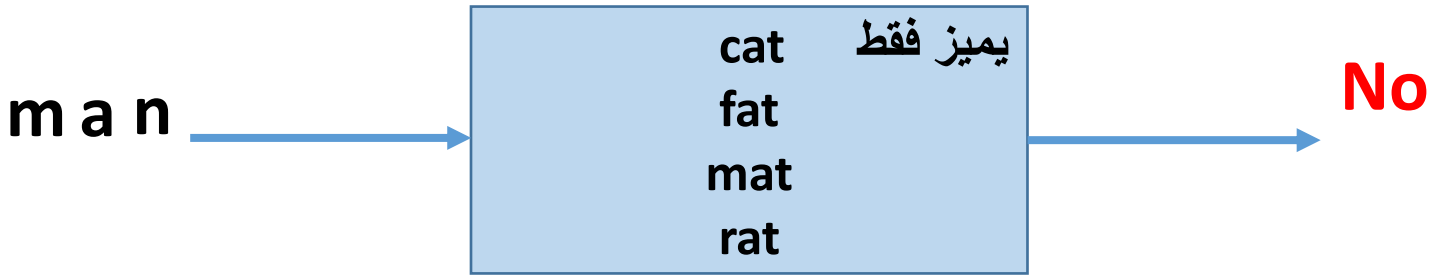
- المميز Recognizer أو Acceptor الخاص بلغة ما هو برنامج مدخلاته هي نص text (ولنرمز له بالمتغير str) والمميز يجيب:
  - ✓ "نعم" إذا كانت str مفردة lexem في اللغة
  - ○ "لا" غير ذلك
- يقوم المميزrecognizer بتفسير التعبير النظامي, ويمكننا متابعة سلوك المميز من خلال بناء مخطط انتقالي transitional diagram يسمى finite automaton

آلية عمل المميز (الآلية المحدودة)

آلية عمل المميز (الآلية المحدودة)



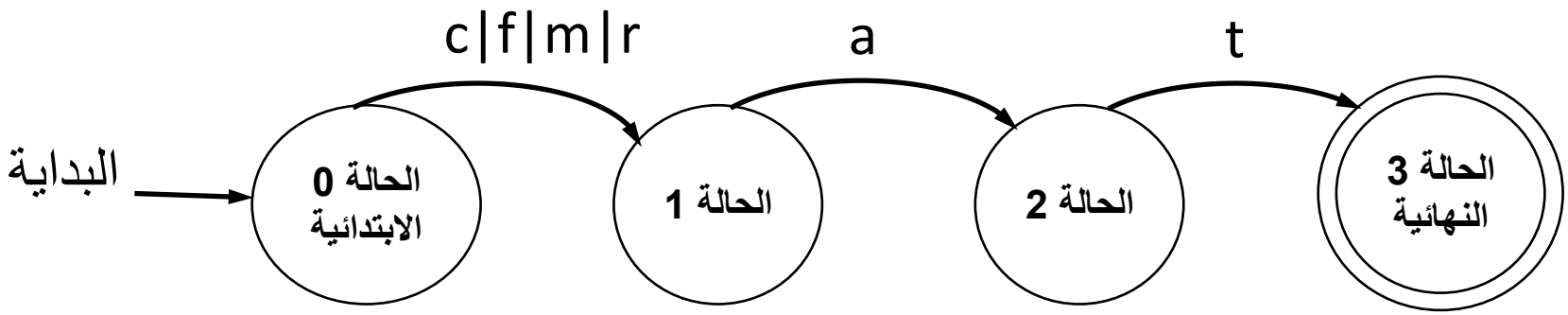
آلية عمل المميز (الآلية المحدودة)



فكرة آلية محددة

مخطط لآلية تتعرف على نمط لغة التعبير النظامي:

(c|f|m|r)(at)

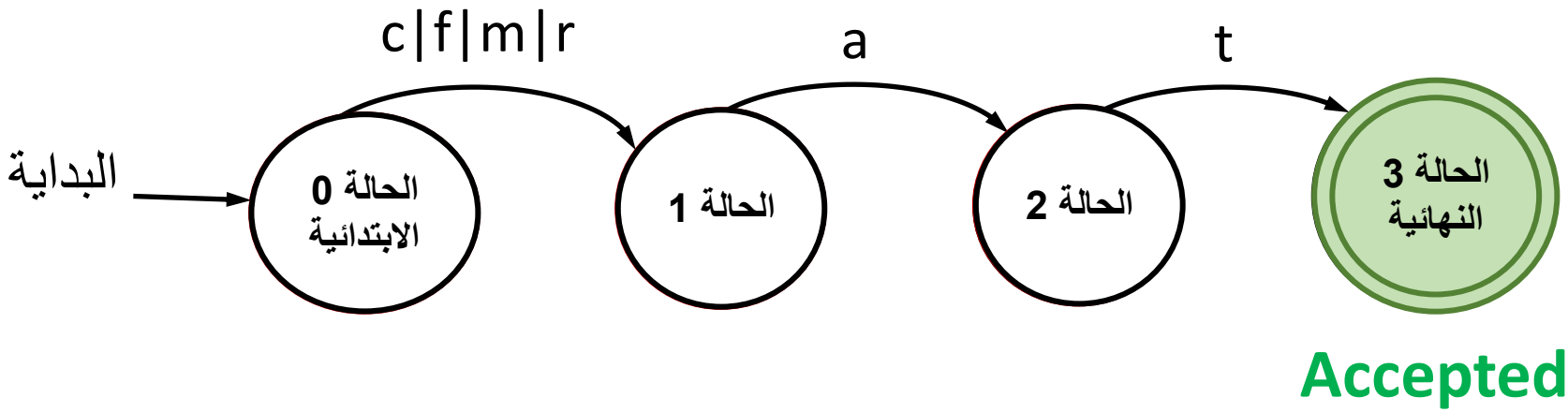


فكرة آلية محددة

مخطط لآلية تتعرف على نمط لغة التعبير النظامي:

(c|f|m|r)(at)

c a t

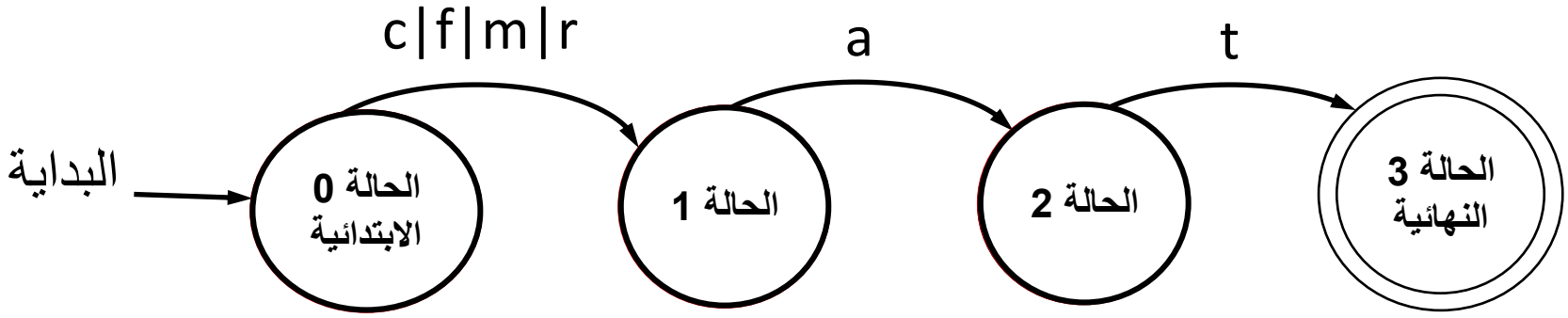


فكرة آلية محددة

مخطط لآلية تتعرف على نمط لغة التعبير النظامي:

(c|f|m|r)(at)

m a n

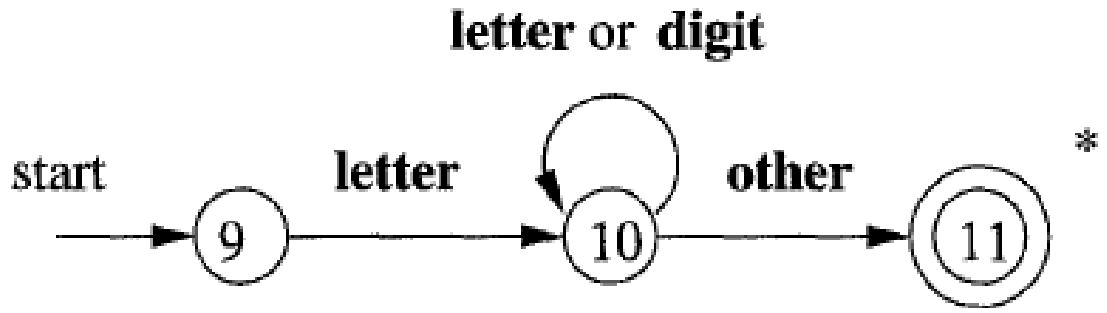


Not Accepted

TRANSITIONAL DIAGRAM

المخطط الانتقالي

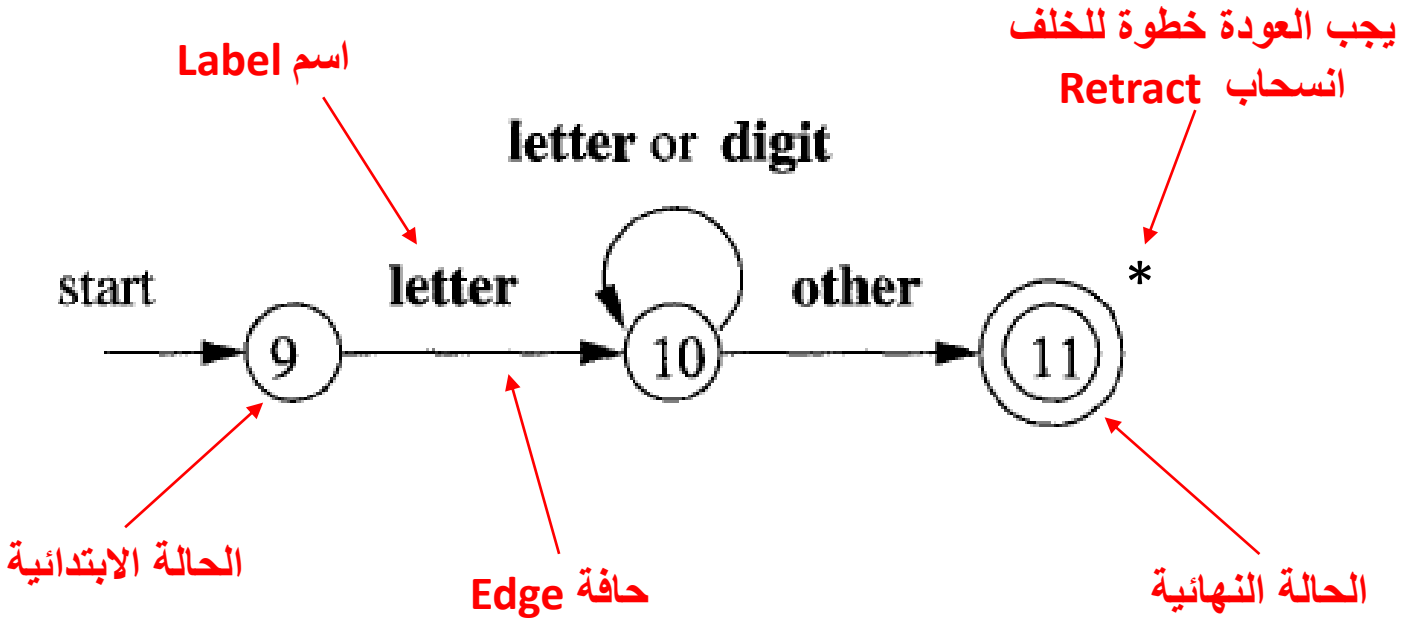
- كخطوة أولية في بناء محلل النص Lexical Analyzer, نقوم بتحويل نمط النص إلى مخطط انسيابي يسمى بالمخطط الانتقالي transitional diagrams
- وهو يحتوي على عقد/دوائر تسمى بالحالات states كل منها تمثل حالة يمكن أن تحدث أثناء عملية تمشيط/مسح المدخلات بحثاً عن المفردات التي تطابق أنماطاً معينة.



مخطط انتقالي للتعرف على نمط المعرفات

TRANSITIONAL DIAGRAM

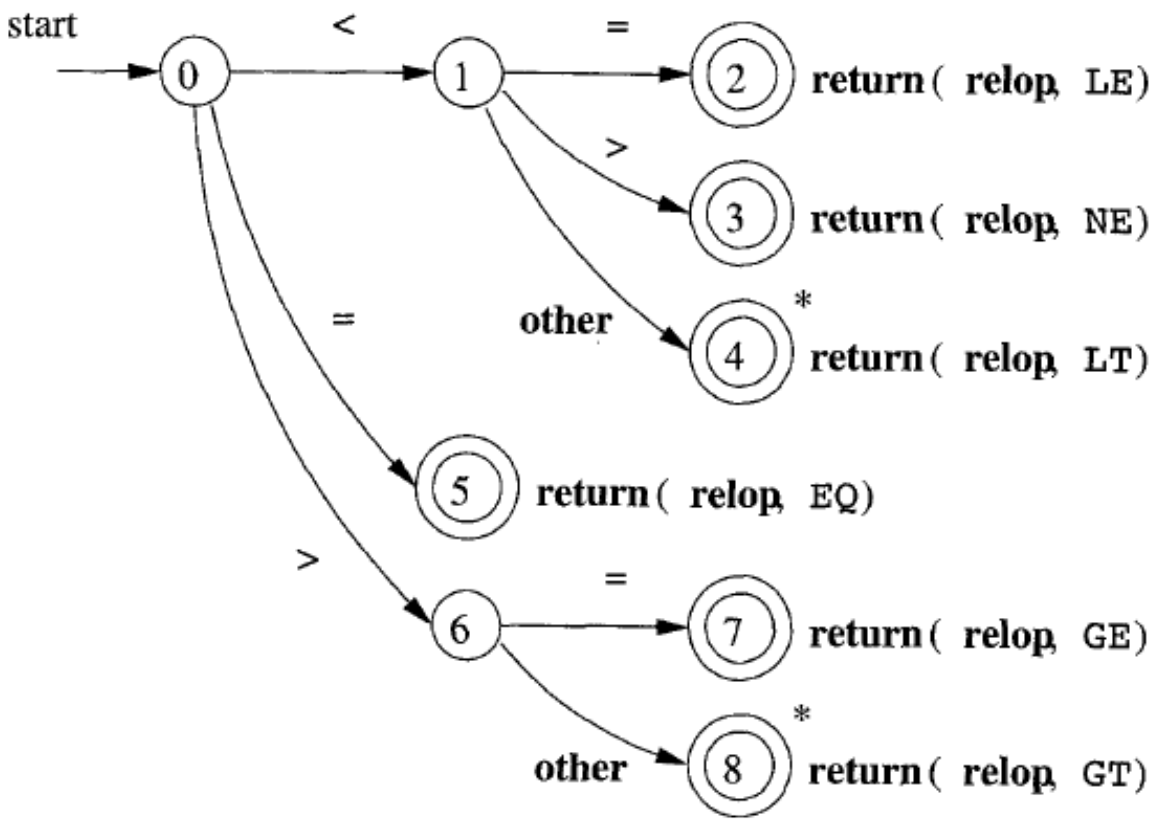
المخطط الانتقالي



- إذا نحن في حالة ما s وجاء مدخل ما i, عندها نبحث عن حافة خارجة من الحالة s عليها اسم label بالمدخل i ثم نتبع اتجاه سهم تلك الحافة لتوصلنا للحالة التالية t

TRANSITIONAL DIAGRAM

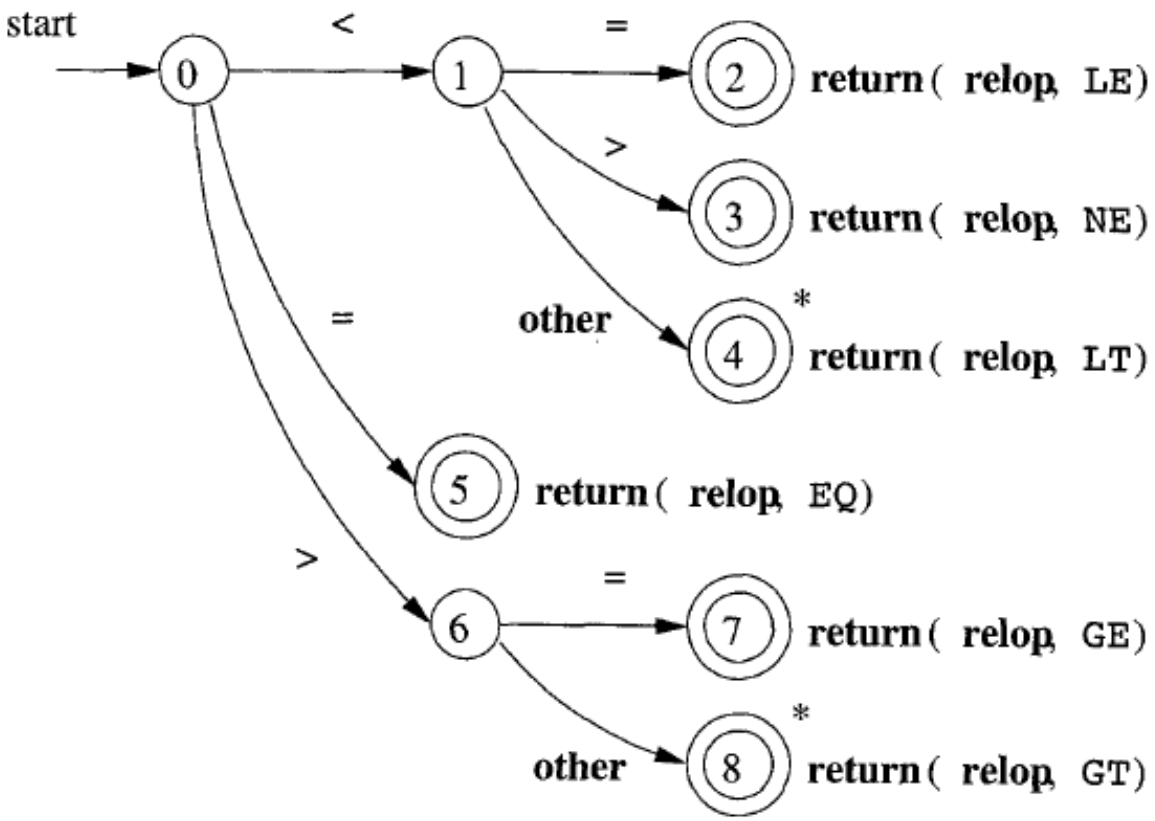
مثال للمخطط الانتقالي



- نبدأ من حالة البداية 0
- إذا كان المدخل الأول هو الرمز < فننتقل للحالة التالية (1), ولنقرر هل المدخل هو من البطاقة token "relop" فعلينا النظر للأمام forward لنقرر ذلك.
- إذا كان المدخل التالي هو الرمز = فنكون وصلنا حالة قبول/نهاية (2) ونتحصل على المفردة LE المعرفة من البطاقة token "relop"

TRANSITIONAL DIAGRAM

مثال للمخطط الانتقالي



- أما إذا كان المدخل التالي هو الرمز > فنكون وصلنا حالة قبول/نهاية (3) ونتحصل على المفردة NE المعرفة من البطاقة token "relop"
- أما إذا كان الرمز التالي أي شئ آخر other بالتالي تحصلنا على المفردة LT من البطاقة "relop" ويتوجب رجوع مؤشر سلسلة النص للخلف retract
- إذا كان المدخل الأول هو الرمز = فإننا تحصلنا على المفردة EQ
- وهكذا مع > و = و other

بناء خوارزمية مخطط انتقالي

```
TOKEN getRelop( )                    // TOKEN has two components: name and value
{
        TOKEN  retToken = new(RELOP);  // First component set here
        while (true) {
          switch(state) {
                case 0:  c = nextChar( );
                         if (c == '<') state = 1;
                         else if (c == '=') state = 5;
                                 else if (c == '>') state = 6;
                                         else fail();
                         break;
                case 1:  ...


}}}
```

بناء خوارزمية مخطط انتقالي

```
TOKEN getRelop( )                    // TOKEN has two components: name and value
{
        TOKEN  retToken = new(RELOP);  // First component set here
        while (true) {
            switch(state) {
                    case 0:  …
                    case 1: c = nextChar( );
                                if (c == '=') state = 2;
                                else if (c == '>') state = 3;
                                        else state = 4;
                                break;
                     ...




}}}
```

بناء خوارزمية مخطط انتقالي

```
TOKEN getRelop( )                    // TOKEN has two components: name and value
{
        TOKEN  retToken = new(RELOP);  // First component set here
        while (true) {
            switch(state) {
                    case 0:  …
                    case 1:   …
                    case 2:  …
                    case 3:  …
                    case 4:  retToken.attribute = "LT";  // second component is set her
                             retract( );   // an accepting state with a star
                             return(retToken);
                    case 5:  …
                    case 6:  …
                    case 7:  …
                    case 8:  …
}}}
```

بناء خوارزمية مخطط انتقالي

- خوارزمية الدالة/الوظيفة getRelop( ) وضيفتها محاكاة مخطط relop الانتقالي وتعود returns بكائن من نوع TOKEN
- الكائن TOKEN يتكون من عنصرين وهما اسم البطاقة, وقيمة صفة البطاقة.
- بدايةً تكون الوظيفة getRelop( ) كائن جديد retToken وتجهزه بالعنصر الأول (اسم البطاقة) RELOP
- الحالة الابتدائية 0 ممثلة في التعليمة :case 0
- الدالة nextChar( ) تجلب الرمز التالي من سلسلة المدخلات وتسلمها للمتغير c
- بعدها نختبر c حول الرموز الثلاثة المتوقعة (<, >, =) لننتقل بين الحالات

بناء خوارزمية مخطط انتقالي

- إذا كان الرمز المدخل التالي ليس من ضمن رموز الاختبارات, عندها تستدعى الدالة fail( ) لتوقف قراءة الرموز والعودة إلى بداية سلسة الرموز المدخلة lexemeBegin لكي يتولى مخطط انتقالي آخر التعامل مع السلسة
- توضح الخوارزمية التصرف في الحالة 8 التي تحمل رمز النجمة *
- على الخوارزمية الانسحاب خطوة للخلف لمؤشر قراءة الرموز وذلك باستدعاء الدالة retract( )
- بما أن الحالة 8 تمثل تمييز المفردة =>, فعندها يتم وضع العنصر الثاني لكائن البطاقة (قيمة صفة token) لتكون في هذه الحالة GT

أساليب تعرف محلل المفردات على نمط بطاقة token

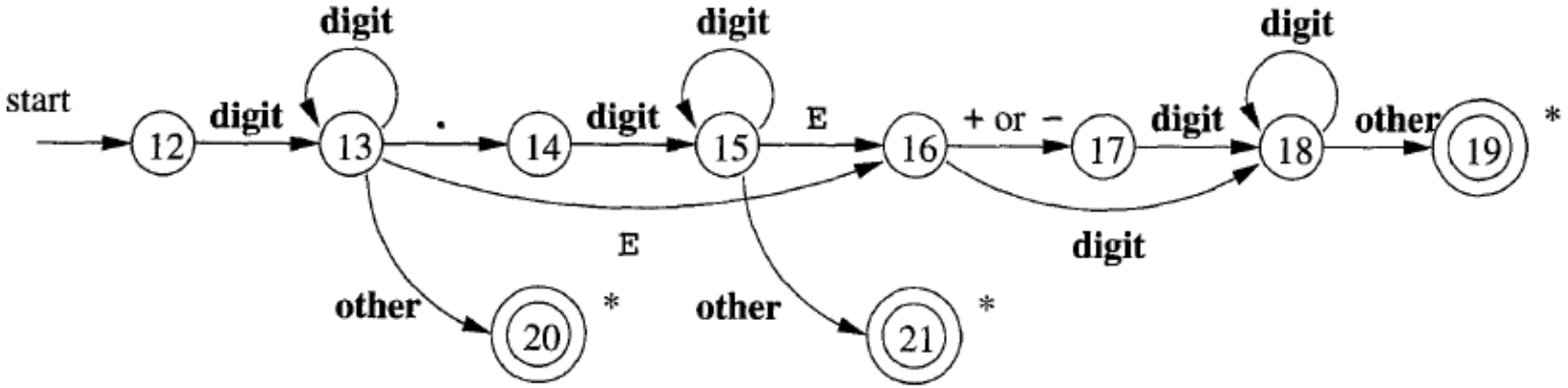
- عند فشل مخطط ما في التعرف على بطاقة token تستدعى الدالة fail( ),
  ويعاد مؤشر القراءة للبداية, يوجد عدَّة أساليب للتعامل مع هذا الموقف:

1. يتم التنسيق لتجربة البطاقات الفاشلة failed على المخططات الانتقالية الواحد تلو الآخر حتى نتحصل على المخطط الصحيح
2. تنفيذ المخططات على التوازي, وتغذية الرمز التالي لهم جميعاً.

- كيف التصرف عندما يتعرف مخطط على نمط المفردة قبل بقية المخططات؟
- يمكن أن تكون المفردة كلمة محجوزة أو قد تكون جزء من معرف ما
- thenext قد يظنها مخطط أنها الأمر then وفي الواقع هي اسم متغير!
- يمكن تفضيل المخططات التي تميز أطول مفردة من السلسة المدخلة

تعرف محلل المفردات على مخطط انتقالي لبطاقة token

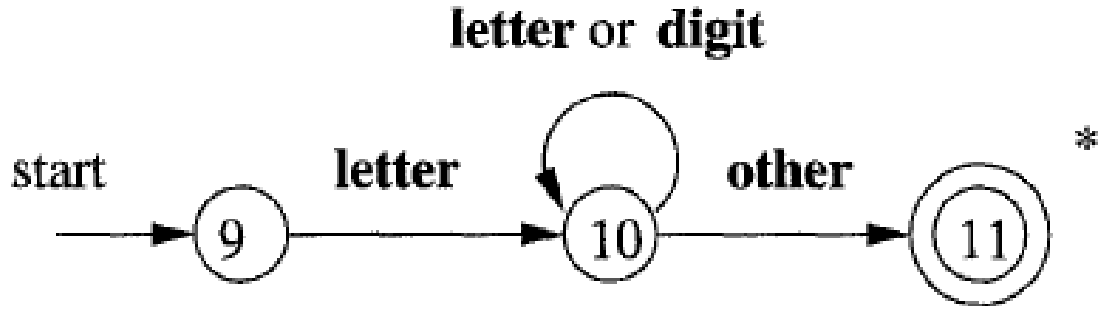
3. الأسلوب المفضل هو دمج كل مخططات الانتقال كمخطط واحد
   - يقرأ المخطط المدخلات إلى أن لا يوجد حالة تالية
   - ثم يأخذ أطول مفردة تتوافق مع أي نمط
   - طبعاً لهذا الأسلوب تعقيداته



مخطط انتقالي يضم مخططات التعرف على عدد موجب صحيح 123, وعدد عشري 1.23, وعدد حقيقي 1.2E3 ($1.2x10^3$)

مخططات التعرف على المعرفات والكلمات المحجوزة

- تمييز الكلمات المحجوزة keywords و المعرفات identifiers يحتاج لخطة
- الكلمات المحجوزة مثل if أو then تشبه اسماء متغيرات (المعرفات)



مخطط انتقالي للتعرف على نمط المعرفات

- هذا المخطط الانتقالي يتعرف على مفردات (lexemes) المعرفات وأيضاً يمكنه التعرف على الكلمات المحجوزة مثل if أو then أو else أو ...

طريقتين للتعرف على المعرفات والكلمات المحجوزة

1. منذ البداية يتم إدراج install الكلمات المحجوزة في جدول الرموز symbol table

أو جدول مخصص للكلمات المحجوزة

- يحدد حقل قي الجدول ليوضح أن هذه الكلمات ليست معرفات
- عندما يعثر مخطط انتقالي على مفردة ما ,

✓ أولاً يجرى بحث في الجدول هل توجد كلمة محجوزة باسمها في جدول الرموز؟

✓ لو نعم: فهذه المفردة هي كلمة محجوزة

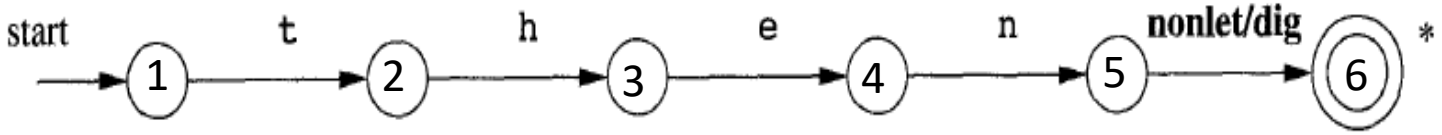
✓ لو لا: فهذه المفردة هي معرف وتمنح لبطاقة id

- الدالة *getToken( )* تختبر جدول الرموز عن وجود المفردة وترد باسم البطاقة المناسبة id token للمعرفات, keyword token للكلمات المحجوزة
- ثم الدالة *installID( )* تعمل على إدراج المعرف في جدول الرموز

طريقتين للتعرف على المعرفات والكلمات المحجوزة

2. **هذه الطريقة تعتمد على مخططات الانتقال بدلاً من جدول الرموز**

- **لكل كلمة مجوزة يوجد مخطط انتقالي يميز نمطها**



- **يوجد حالات تمثل موقف/حالة المخطط بعد كل حرف من حروف الكلمة**
- **يلي حالات حروف الكلمة المحجوزة اختبارٌ عن وجود أرقام أو رموز أخرى nonletter-or-digit لا تكون عادة جزء من كلمة محجوزة؟**
  - ✓ لا يوجد حرف أو رقم: إذن هذه مفردة كلمة محجوزة
  - ✓ نعم يوجد حرف أو رقم : هذه مفردة ليست كلمة محجوزة
- **في هذا الأسلوب يكون الأولوية لاختبار أنماط الكلمات المحجوزة ثم تأتي الأنماط الأخرى**

DESCRIPTING FINITE STATE AUTOMATA

وصف آليات الحالات المحدودة

- عرفنا أن آلية الحالات المحدودة finite state automaton هي عبارة عن جهاز معالجة معلومات ينقل المدخلات ويحولها لمخرجات
  - مدخلات: لغة من هجائية *input alphabet A*
  - مخرجات: نعم أو لا
    - نعم: المدخلات مقبولة accepted
    - لا: المدخلات غير مقبولة not accepted

- لذلك تسمى أيضاً بآلات القبول المحدودة finite acceptors

واجب,

- **اكتب برنامج جافا لتطبيق كامل خاص بآلية محدودة تتعرف على بعض الكلمات المحجوزة في لغة سي++**
- **.[int float main if goto cout cin]**
- **يجب على برنامجك أن يستخدم:**
  - ✓ **وظيفة** nextChar( ) تقوم بقراءة حرف من الملف المصدر
  - ✓ **وظيفة** ( ) تمسح التعليقات والمساحات الزائدة والاسطر الفارغة
  - ✓ **صنف class لإحتواء البطاقة** TOKEN
  - ✓ **صنف class لتمثيل جدول الرموز**
  - ✓ وأي وظائف أخرى تراها ضرورية
- **يسلم الواجب عبر الإيميل على هيئة مشروع مكتمل يعمل بلا أخطاء**
- **آخر موعد للتسليم يوم الخميس القادم 13 يوليو 2023. لا يقبل أي واجب بعد هذا التاريخ**
- **الواجبات المتماثلة لاترصد لها أي درجة**
- **أرسل مع الواجب ملف ميكروسفت وورد به بياناتك الشخصية وشرح لكيفية عمل برنامجك**

المحاضرة التالية:

الآليات المحدودة ... يتبع

FINITE AUTOMATA ...