補充資料 I (Regular Grammar)

資科系
林偉川

Grammar Definition

G=(V,T,S,P) , where

V : a finite set of variables(Non-terminal)
T : a finite set of terminal symbols(terminal)
$S \in V$ is a special symbol called ***start*** variable
P is a finite set of productions(rules)

Example

G=({S},{a,b}, S, P), where P are given by

$S \to aSb \mid \lambda$ (recursive grammar rule)

| is logical OR for production.

Derivation processes:

$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$

Grammar produces a language result:

$L(G)=\{ a^n b^n : n >= 0 \}$

Example

G=({S,A},{a,b}, S, P), where P are given by

$S \to Ab$

$A \to aAb \mid \lambda$

Grammar produces a language result:

$L(G)=\{ a^n b^{n+1} : n >= 0 \}$

Example

G=({S},{a,b}, S, P), where P are given by

$S \rightarrow SS \mid \lambda \mid aSb \mid bSa$

Grammar produces a language result:

$L(G)=\{ w: n_a(w)=n_b(w) \}$

5

Example

G=({S,A},{a,b}, S, P), where P are given by

$S \rightarrow aAb$

$A \rightarrow aAb \mid \lambda$

Grammar produces a language result:

$L(G)=\{ a^n b^n: n > 0 \}$

6

DFA Definition(Regular Grammar)

$M=(Q, \Sigma, \delta, q_0, F)$ where

Q：a finite set of internal states

Σ ： a finite set of input alphabets

δ ： $Q * \Sigma \rightarrow Q$ is a transition function

$q_0 \in Q$ is the initial state

$F \subseteq Q$ is a set of finite states

7

DFA Example

$M=(\{q_0,q_1,q_2\}, \{0,1\}, \delta, q_0, \{q_1\})$

Where δ is given by

$\boldsymbol{\delta(q_0,0)=q_0}$ ， $\boldsymbol{\delta(q_0,1)=q_1}$,

$\boldsymbol{\delta(q_1,0)=q_0}$ ， $\boldsymbol{\delta(q_1,1)=q_2}$,

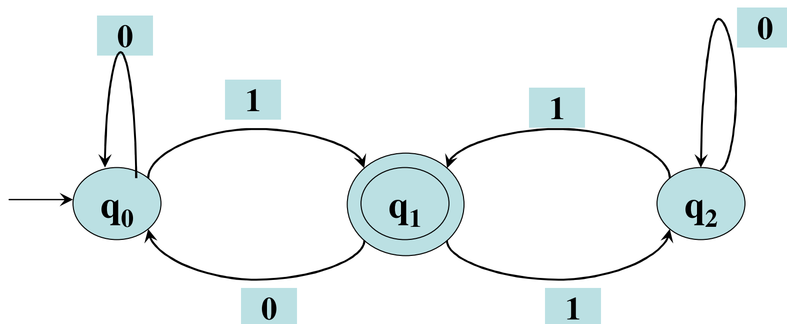
$\boldsymbol{\delta(q_2,0)=q_2}$ ， $\boldsymbol{\delta(q_2,1)=q_1}$,

Input symbol / Current State	0	1
q_0	q_0	q_1
q_1	q_0	q_2
q_2	q_2	q_1

DFA produces a language result:

L(DFA)={ accept the string **001(奇數個1結尾) }**

DFA Example(continued)



9

DFA Program Example

```
….
 switch (state) {
Case 0: ….
   if (ch == ‘1’) state=1; break;
Case 1: …
   if (ch == ‘0’) state=0; break;
}
…..
```

10

```
#include <stdio.h>
void main() {
  int state=0;  char c;
  clrscr();
  while ((c=getchar())
!= '\n') {
    switch (state) {
    case 0:
      switch (c) {
      case '0':
        state=0; break;
      case '1':
        state=1; break;
      }
      break;
case 1:
      switch (c) {
      case '0':
         state=0; break;
      case '1':
         state=2; break;
      }
      break;
    case 2:
      switch (c) {
      case '0':
        state=2; break;
      case '1':
        state=1; break;
      }
      break;
    }
printf("%c stat=%d\n",
c,state);
  }
  if (state == 1)
    printf(" --> this
string is accept by this
DFA\n");
  else
     printf(" --> this
string is not accept by
this DFA\n");
}
```

DFA and Regular Grammar Example

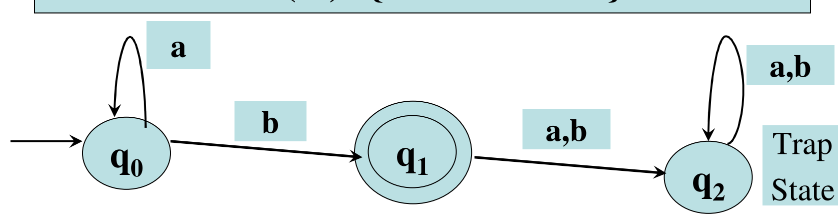
G=({S,A},{a,b}, S, P), where P are given by

$S \rightarrow Ab$

$A \rightarrow aA \mid \lambda$

Grammar produces a language result:

$L(G)=\{ a^n b: n >= 0 \}$



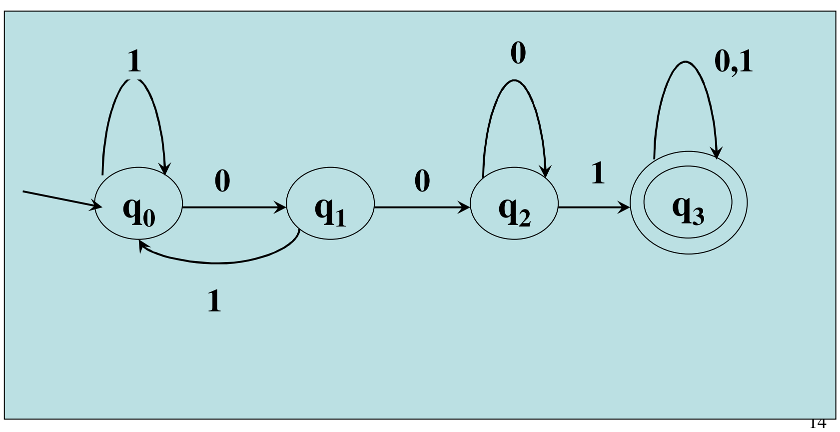
DFA and CFG Example(continue)

Current State \ Input symbol	a	b
q_0	q_0	q_1
q_1	q_2	q_2
q_2	q_2	q_2

13

DFA and CFG Example

Find a DFA accept all strings on {0,1} containing the substring 001 ?



14

```
accept=0;
while ((c=na[i++]) != '\0') {
      switch (state) {
      case 0: if (c == '0') state=1;      break;
      case 1: if (c == '1') state=0;    if (c == '0') state=2;   break;
      case 2: if (c == '1') state=3;   break;
      case 3:  accept=1;   break;
      }
 }
 if (accept == 0) printf("%s is not accept for substring 001\n",na);
 else printf("%s is accept for substring 001\n",na);
```

15

DFA and CFG Example

Find a DFA, 判斷一運算式的對錯？

(以 = 為 final state) [至少需? 個states]

Ex. 12 + 23   - 45 *   67=

67 = 視為error!!!

16