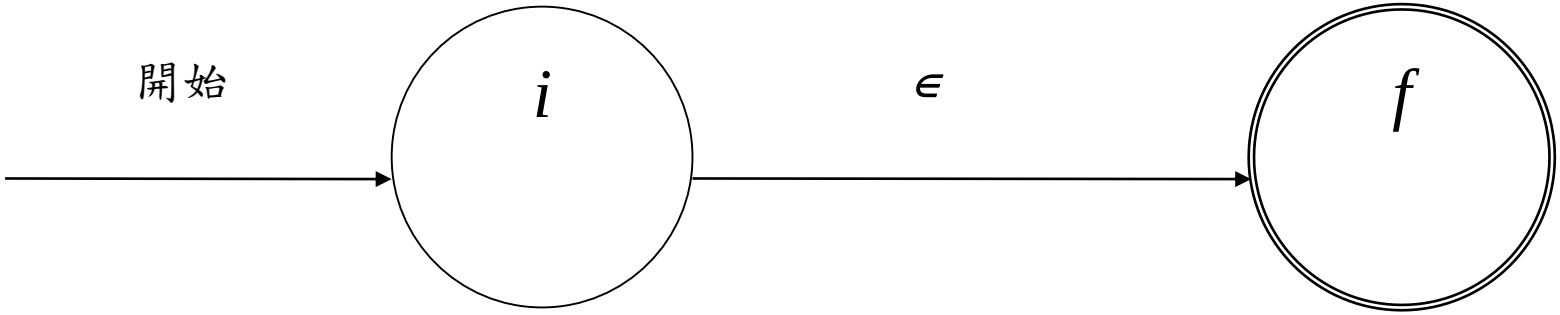

Fig.1

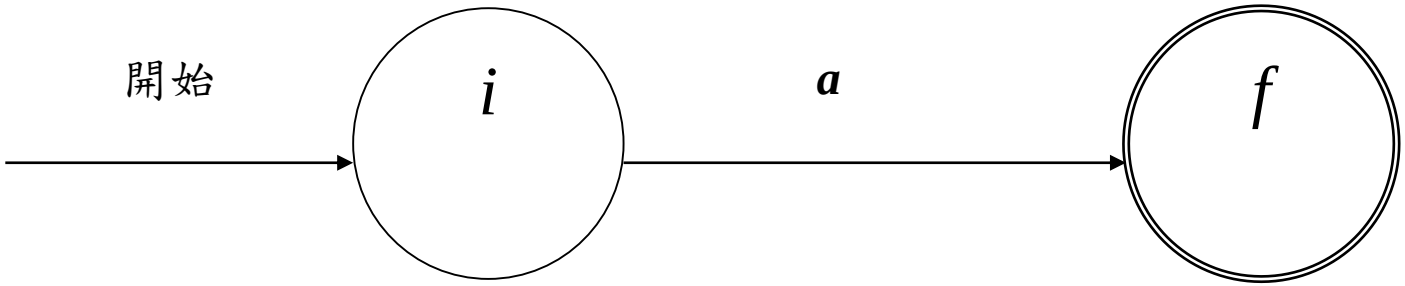

Fig.2

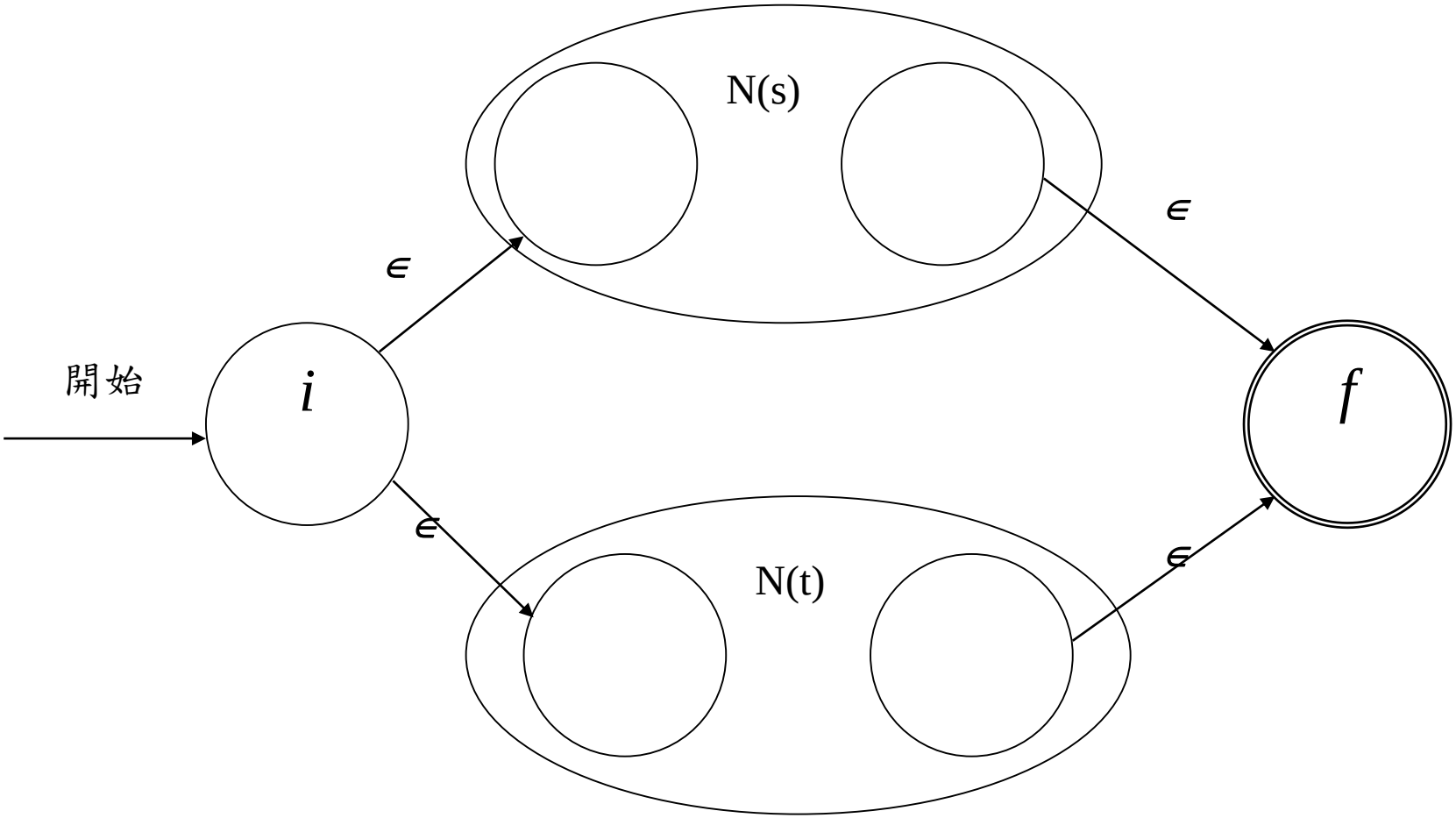

Fig.3a

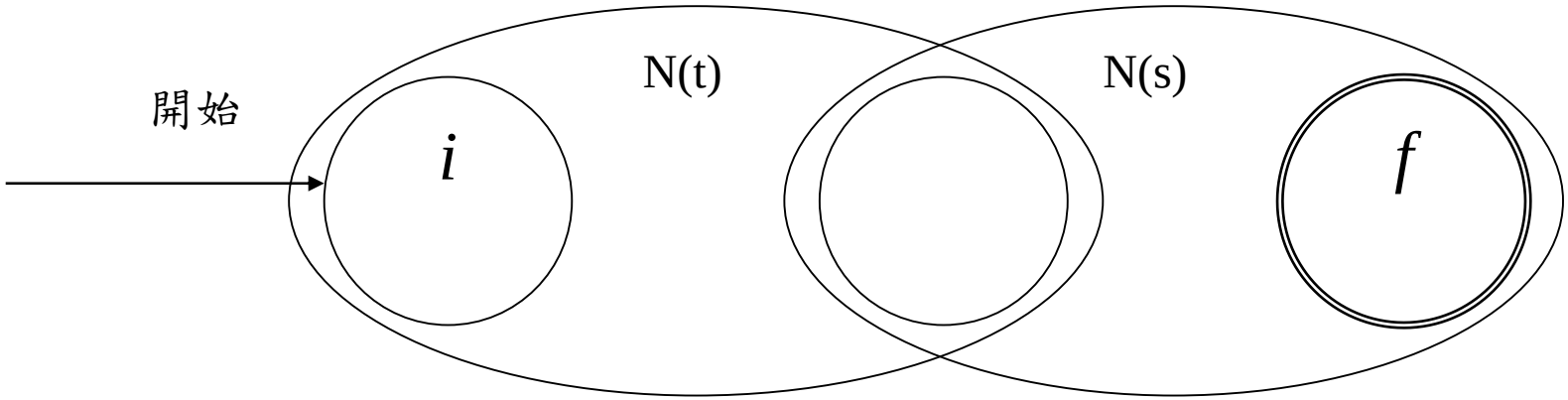

Fig.3b

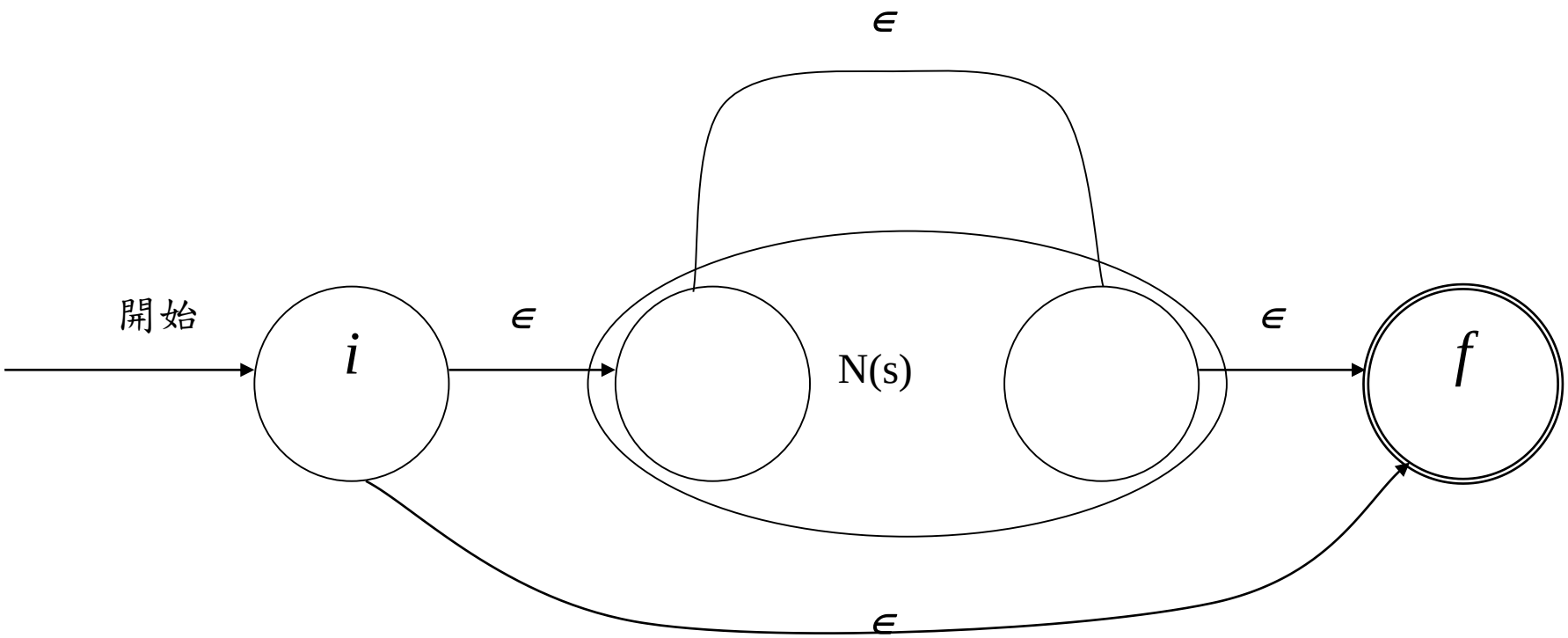

Fig.3c

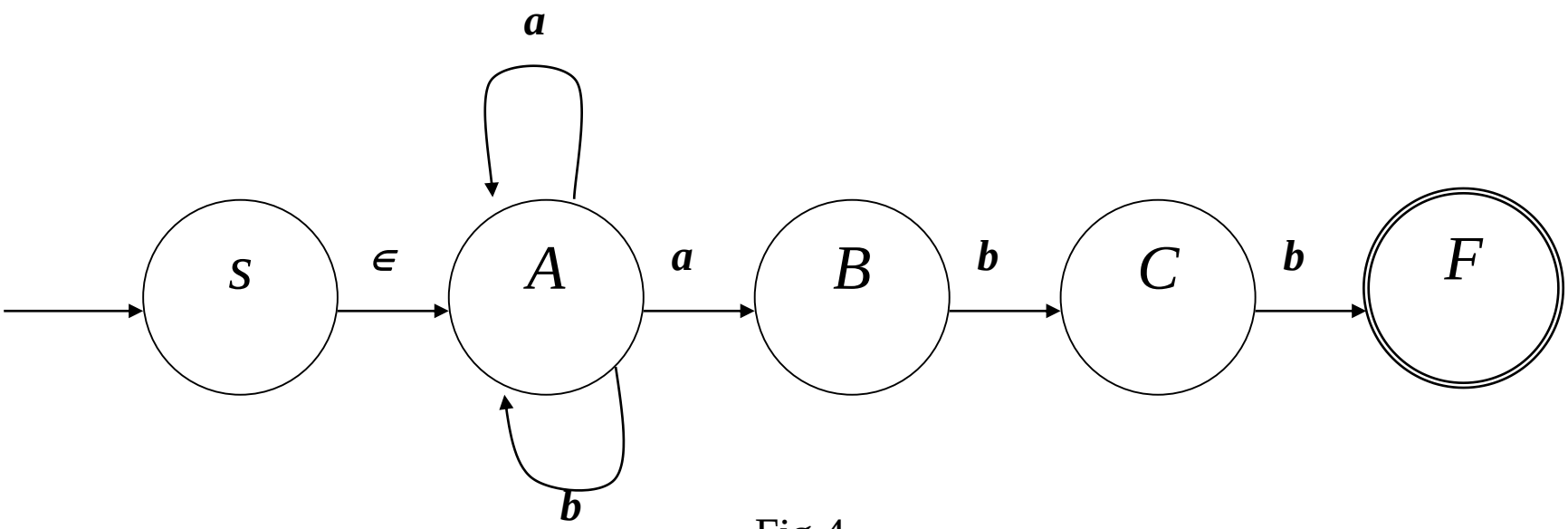

Fig.4

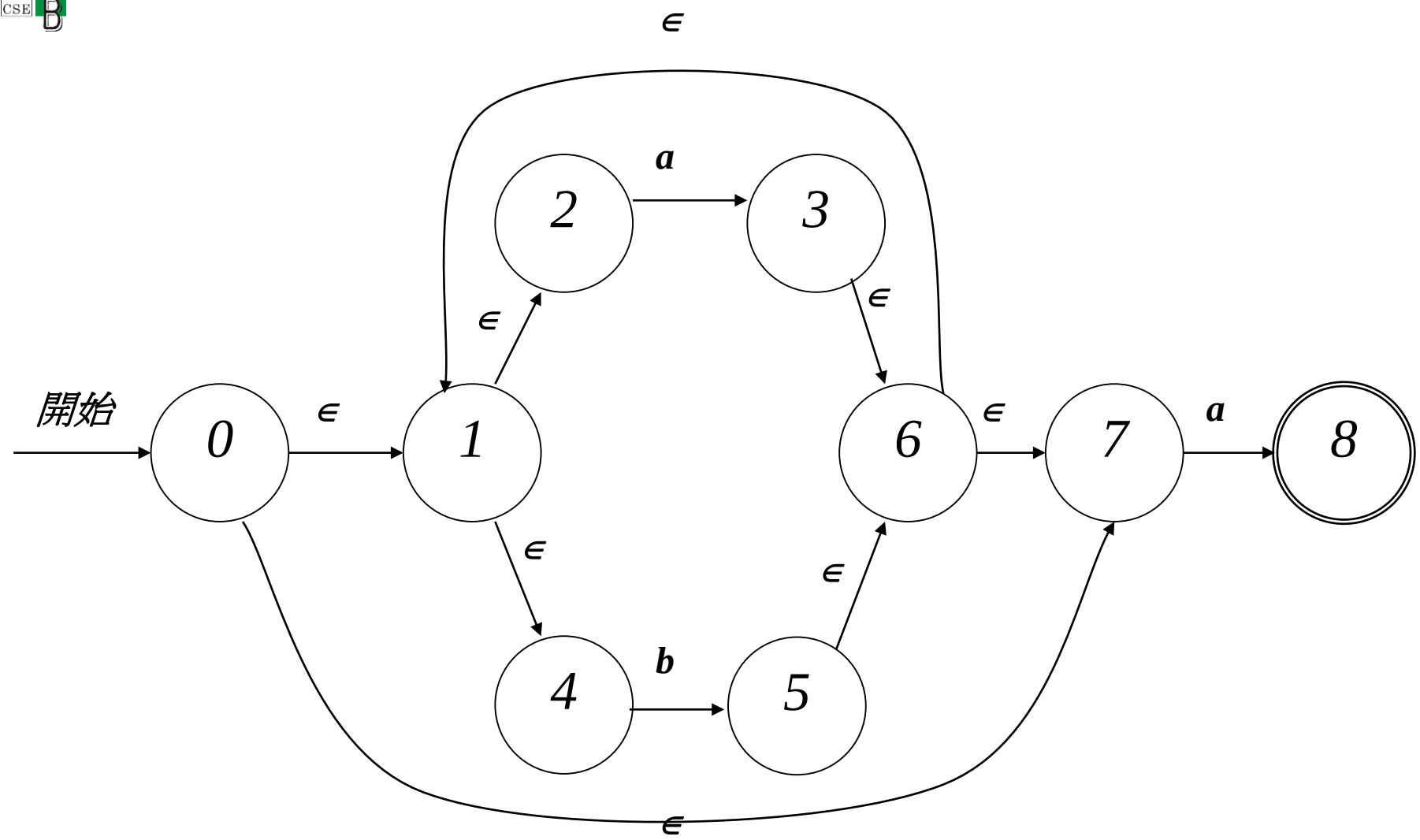

Fig.5

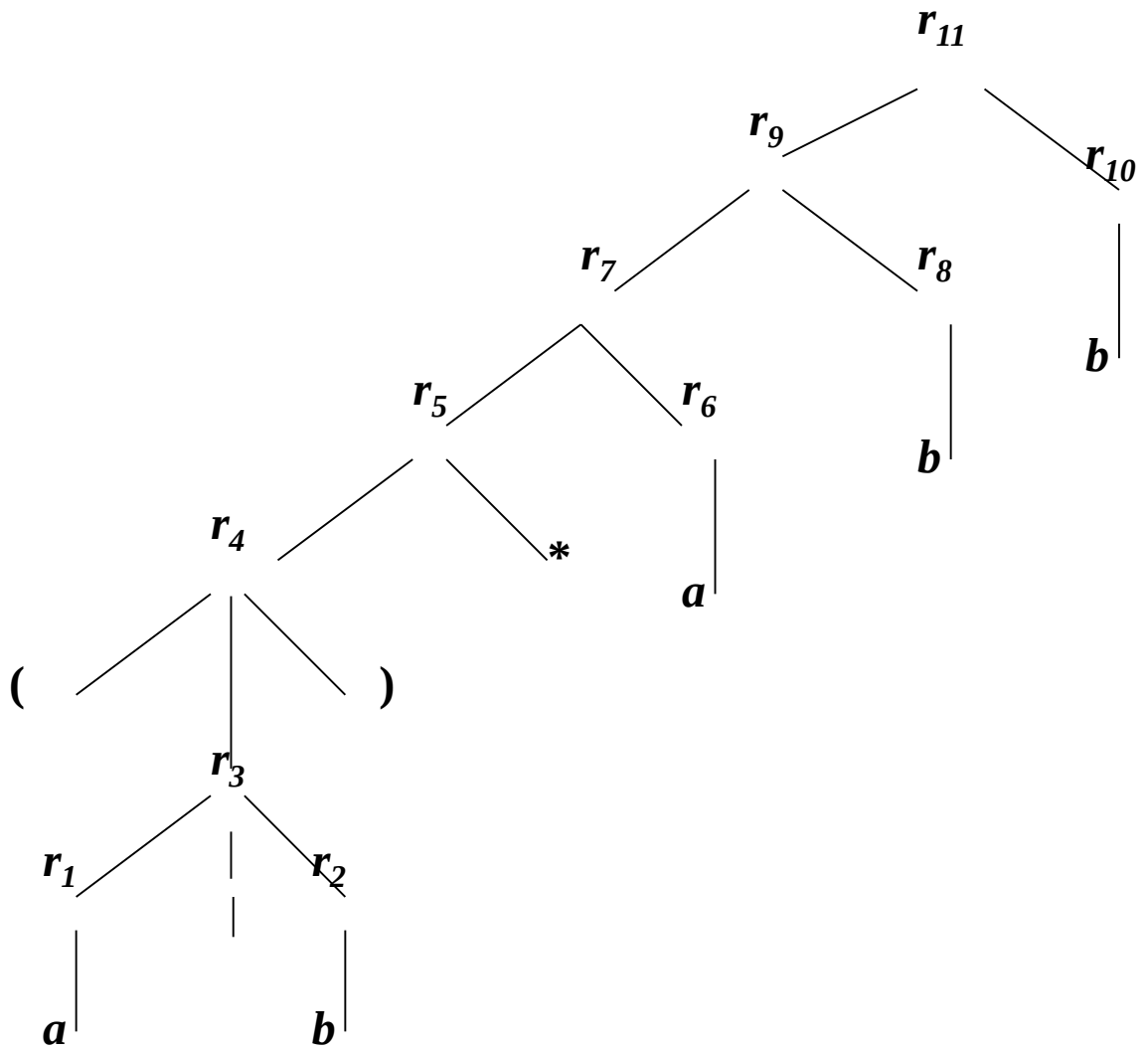

Fig.6 (*a*|*b*)**abb*的分解方式

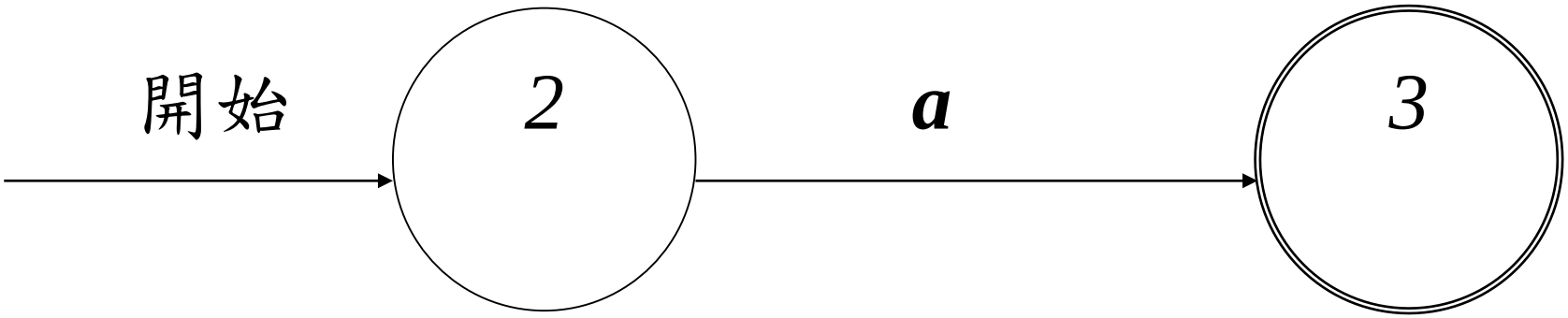

Fig.7

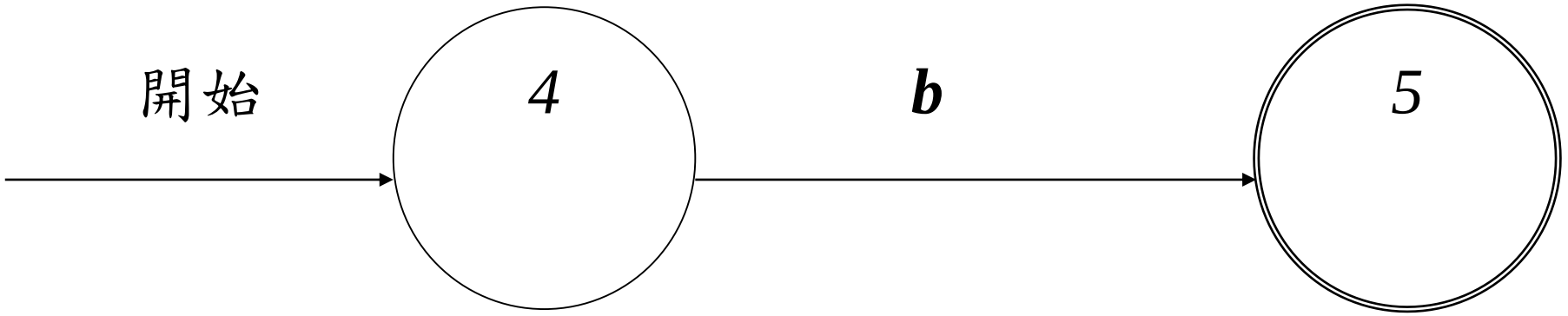

Fig.8

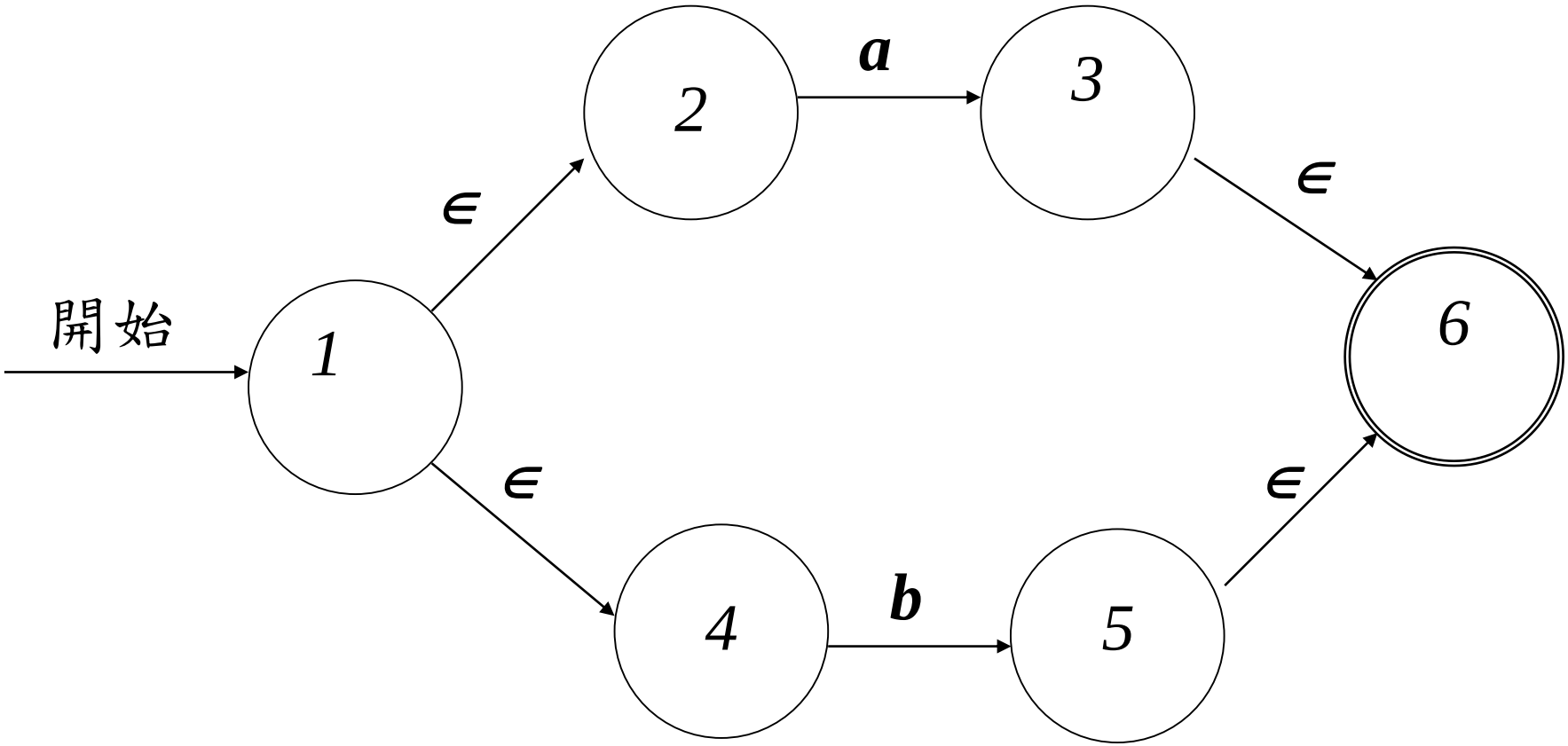

Fig.9

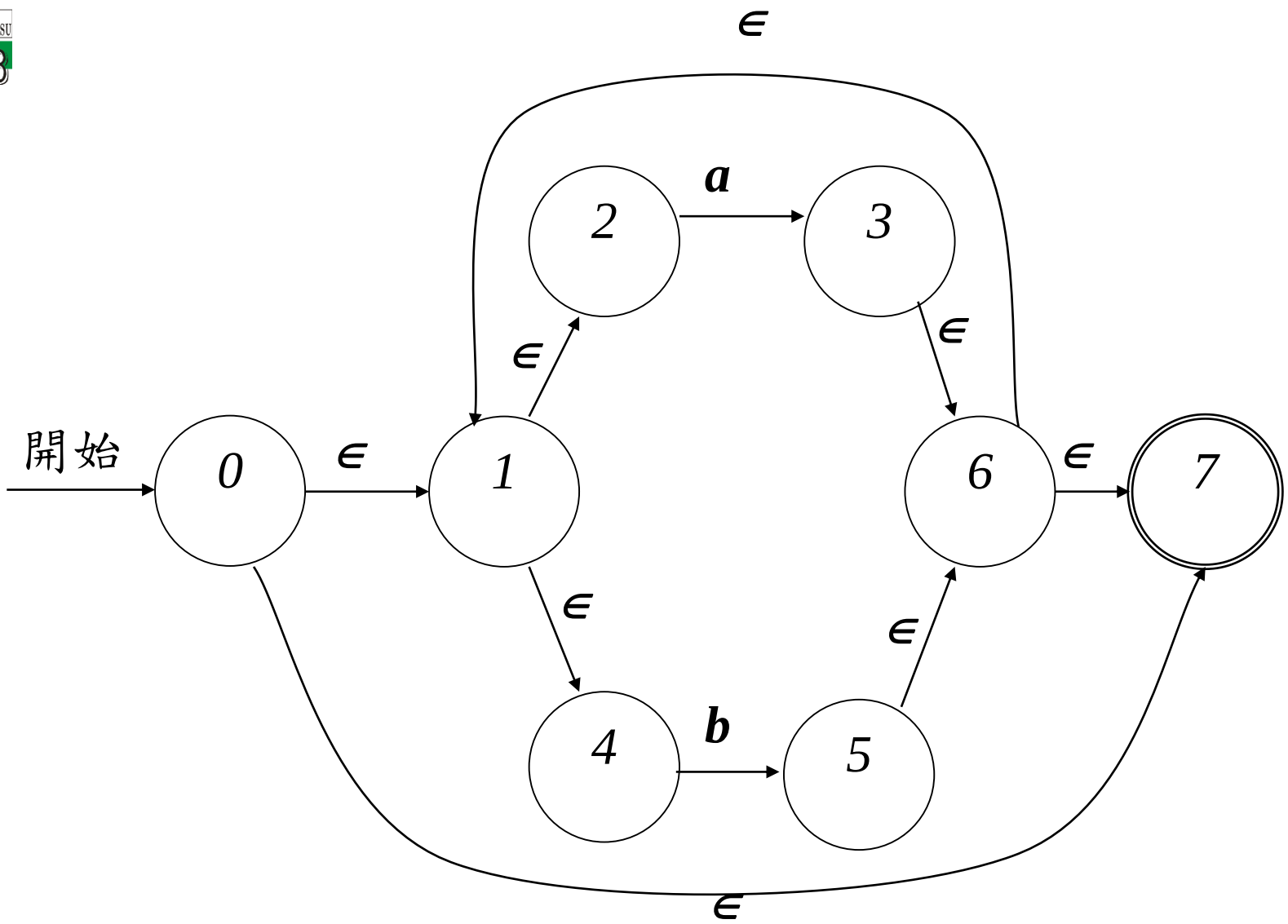

Fig.10

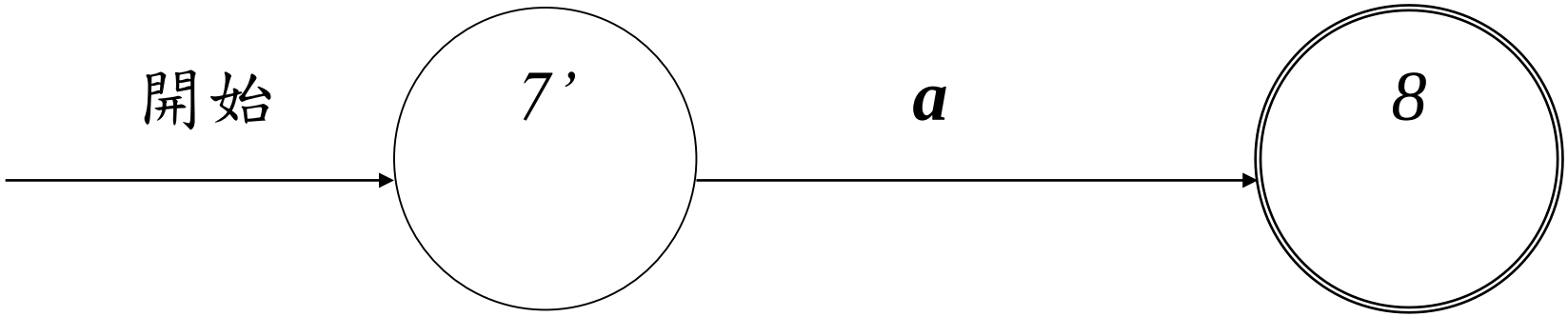

Fig.11

模擬DFA

- **輸入**：一個檔案終了符號eof結束的輸入字串 x；另外D是一個已知的DFA，它的開始狀態是S_0，接受狀態是狀態集合F。

- **輸出**：如果D接受x，得來的結果就是yes，要不然就是no。

- **方法**：對輸入字串x執行下圖的程序。在下圖中的move(s,c)函數，就會從狀態s對於輸入符號c找出下一個狀態；另外，nextchar函數就傳回在輸入字串x中的下一個符號。

模擬DFA

NFA→DFA

```
s := s0;
c := nextchar;
while c ≠ eof; do
        s := move(s,c);
        c := nextchar
end;
if s is in F then
        return "yes"
else return "no";
```

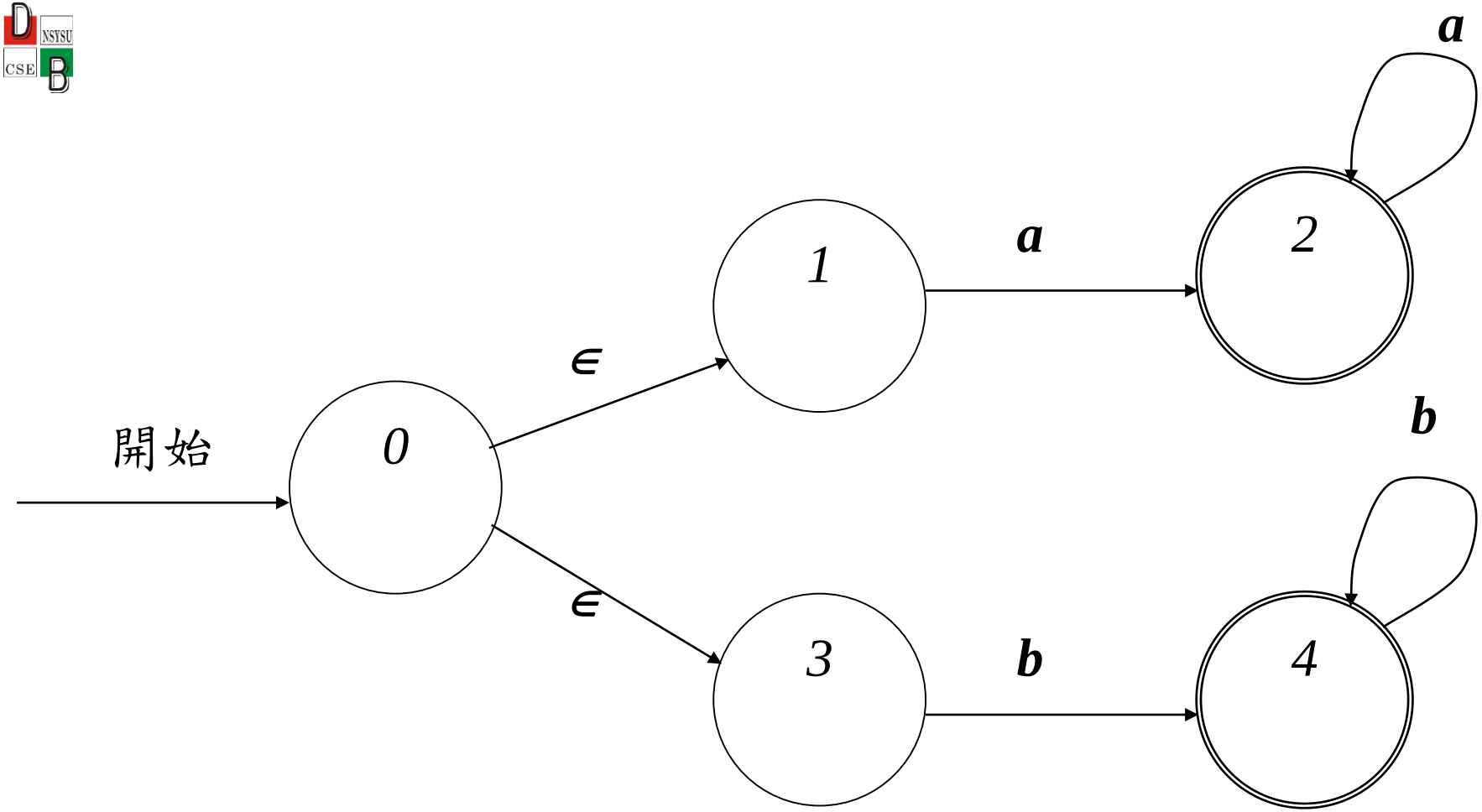

Fig.13 接受aa*|bb*的NFA

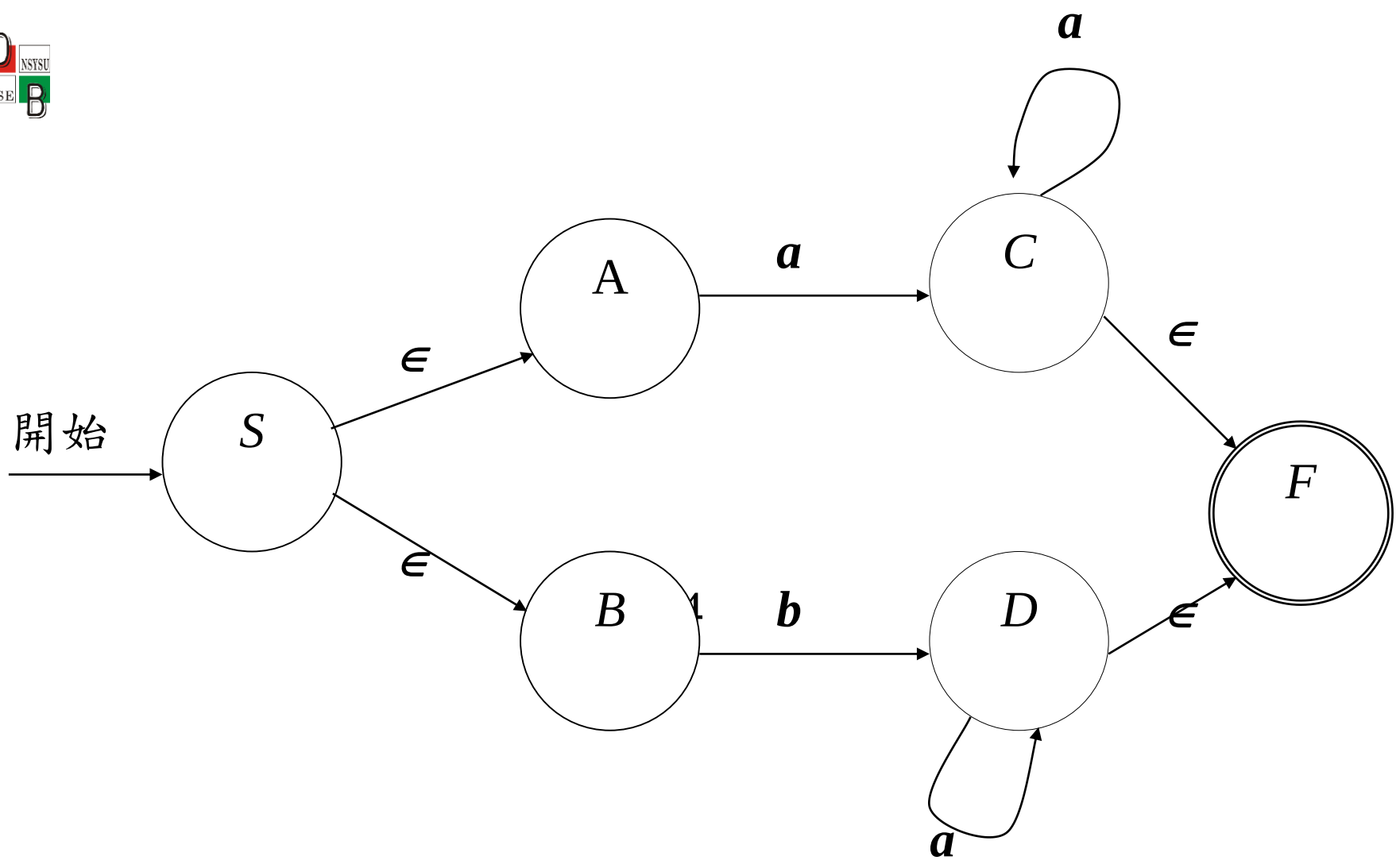

Fig.14

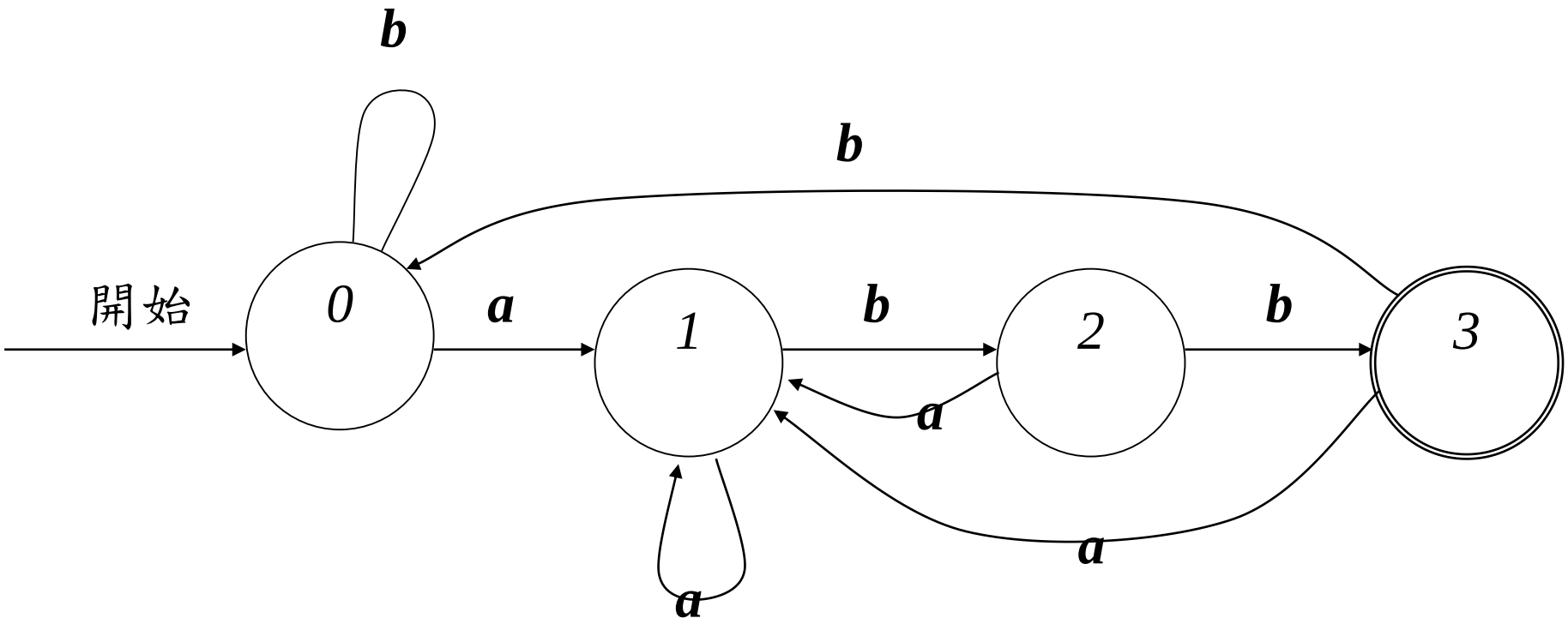

Fig.15 接受(a|b)*abb的DFA

運算	說明
∈-closure(s)	從NFA的某一個狀態s開始，可以透過標上∈的邊線所能夠到達的所有NFA狀態所構成的集合。
∈-closure(T)	從某一個NFA狀態集合T中的某一個狀態s開始，能夠透過標上∈的邊線所能夠到達的NFA狀態所構成的集合。
move(T , a)	從NFA狀態集合T中某一個元素（也就是狀態）s開始，能夠經過輸入符號*a*到達的所有NFA狀態所構成的集合。

Fig.16 對NFA狀態的運算

在開始的時候，$\in$-closure(s_0)是在Dstates中唯一的一個狀態，而且它也是沒有標上記號的；

While 在Dstates中有一個沒有標上記號的狀態T do begin
把T標上記號；
for對於某一個輸入符號*a* do begin
U := $\in$-closure(move(T , a));
if U 不是Dstates中的元素then
把U當作是一個沒有標上記號的狀態，加到Dstates中;
Dtran|T ,a| := U
end
end

Fig.17 部分集合構作法

把所有T中的狀態都推入堆疊；
為∈-closure(T)給定初值T；
while stack 中還有元素 **do begin**
    把t（也就是堆疊頂端的元素）從stack中取出；
    for狀態u，如果有一條標了∈的邊線從t到u do
        if u不在∈-closure(T)中do begin
            把u加到∈-closure(T)中；
            把u推入stack
    end
end

Fig.18 計算∈-closure