به نام خدا

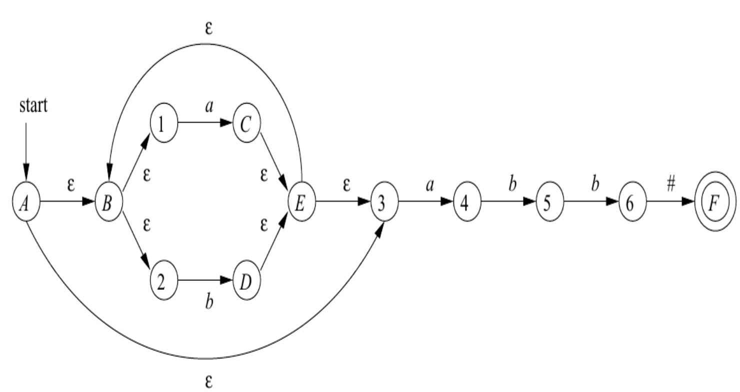
روش قبلی تبدیل RE به DFA: RE → NFA → DFA

روش تبدیل مستقیم RE به DFA (3.9.1)

- در ساخت NFA از یک RE، برای هر نماد a در RE یک حالت (state) با قانون a ایجاد می‌گردد و بقیه قانون‌ها ε-trans هستند.

- در ساخت DFA از NFA، برای هر نماد a در RE، ε-closure(move(s,a)) محاسبه می‌شود و اگر همه قوانین s، ε باشند move(s,a) هم تهی می‌شود. پس اگر ε-closure(move(s,a)) تهی نباشد باید قانونی برای نماد a در s باشد و در این صورت ε-closure(move) در DFA ظاهر می‌شود.

- پس به طور کلی در ساخت DFA از NFA فقط حالاتی که خروجی با برچسب غیر ε داشته باشند باعث تولید حالات DFA می‌شوند. این حالات متناظر با نمادهای عبارت RE هستند.



- مثال

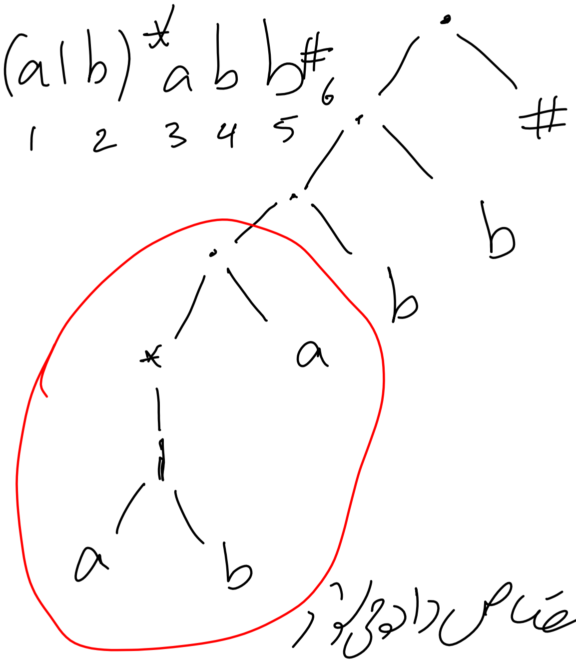
$(a|b)^* abb\#$

- در هر دو حالت accept ، NFA خروجی ندارد در نتیجه اگر F پذیرش NFA باشد (ــــــ epsilon closure ε تهی می گردد لذا یک # به انتهای RE اضافه می کنیم (augmented RE)

- مرحله اول الگوریتم

$(a|b)^* a\, b\, b\, \#$

1 2 3 4 5 6



- درخت augmented RE را رسم می کنیم
- هر کاراکتر در RE به برگ تبدیل می شود
- عملگرها یعنی cat (.) ، or (|) و * به گره داخلی تبدیل می شوند.
- به هر حرف RE از سمت چپ به ترتیب یک عدد اختصاص داده می شود

تعریف توابع لازم جهت اجرای الگوریتم مورد نظر

nullable (n) : اگر REی که از زیر درخت n بدست می آید بتواند ε را تولید کند، برابر T است و در غیر این صورت F است.

- اگر n برگی از درخت (با برچسب غیر ε) باشد nullable(n) = F
- اگر n برگی با برچسب ε باشد nullable(n) = T
- اگر n یک گره ستاره دار باشد nullable(n) = T

مثال) a^* , ab^* , $(a|b)^*$

T F T

$firstpos(n)$ مجموعه تمام مکان حروفی که میتواند منطبق با اولین حرف رشته‌های تولیدی RE (متناظر با n) باشد.

مثال) ab^* $\{1\}$

$(a|b)^*c$ $\{1,2,3\}$

$(a|b)^+c$ $\{1,2\}$

$lastpos(n)$ مجموعه تمام مکان حروفی که میتواند منطبق با آخرین حرف رشته‌های تولیدی از RE متناظر با n باشد

مثال) ab^* $\{1,2\}$

$(a|b)^*c$ $\{3\}$

روش محاسبه توابع نامبرده برای همه نودهای درخت عبارت منظم r:

۱- $r = r_1 | r_2$

$$fp(r) = fp(r_1) \cup fp(r_2)$$
$$lp(r) = lp(r_1) \cup lp(r_2)$$
$$null(r) = null(r_1) \text{ or } null(r_2)$$

۲- $r = r_1 \cdot r_2$

if $(null(r_1) = T)$
  $fp(r) = fp(r_1) \cup fp(r_2)$
else $fp(r) = fp(r_1)$

if $(null(r_2) = T)$
  $lp(r) = lp(r_1) \cup lp(r_2)$
else $lp(r) = lp(r_2)$

$$null(r) = null(r_1) \text{ and } null(r_2)$$

۳- $r = r_1^*$

$$fp(r) = fp(r_1)$$
$$lp(r) = lp(r_1)$$
$$null(r) = T$$

$(a|b)^* a\, b\, b\, \#$
$\quad 1 \quad 2 \quad 3\ 4\ 5\ 6$

followpos (p):

مجموعه مکان حروفی که پس از مکان p در رشته تولید شده RE قرار گیرند.

(برای مکان p، followpos(p) برابر تمام و کلیه مکان در RE است) دیگر

رشته $x = a_1 a_2 \dots a_n \in L(r\#)$ باشد که برای یکی از a_k ها، a_k منطبق با مکان p و a_{k+1} منطبق با q باشد.

مثال) ab^* $\quad$ $followpos(1) = \{2\}$

$(a|b)^* c$ $\quad$ $followpos(1) = \{1, 2, 3\}$

قوانین تولید followpos

۱- اگر n یک cat node $(r_1 \cdot r_2)$ باشد و p مکانی در مجموعه $lpos(r_1)$ باشد تمام مکان های مجموعه $fpos(r_2)$ در followpos(p) قرار دارند.

۲- اگر n یک $*$node ($n = r^*$) باشد و p مکانی در $lpos(r)$ باشد تمام مکان های مجموعه $fp(r)$ در followpos(p) قرار دارند.

مثال) $(a|b|c)^* \cdot (c|d)$

$followpos(3) \supset \{4, 5\}$

$followpos(3) \supset \{1, 2, 3\}$

قانون 1 $\Rightarrow followpos(3) = \{1, 2, 3, 4, 5\}$

الگوریتم تولید مستقیم DFA از روی عبارت منظم r

۱- درخت پارس r# را بساز

۲- توابع null ، fp ، lp و follow pos را در پیمایش عمقی درخت محاسبه کنید

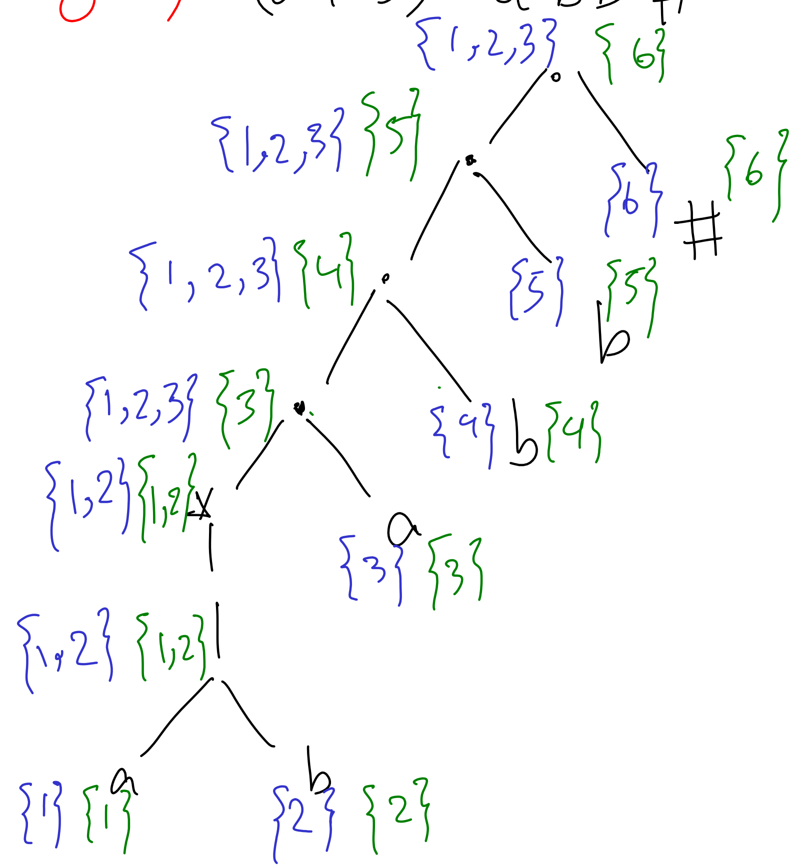
۳- Dstates و Dtran را بدست بیاور

۴- حالتی که شامل مکان # باشد final میباشد

firstpos(n_0) ← ریشه درخت — علامت گذاری نشده

```
Dstates = { firstpos(n0) }
while ( در Dstates هست S غیر علامت گذاری شده state ) {
    mark S;
    for ( در الفبای RE هر سمبل a ) {
        U = ∪ followpos(p), for p∈S that corresponds to a
        if ( U ∉ Dstates )
            add U as an unmarked state to Dstates
        Dtrans [S, a] = U
    }
}
```

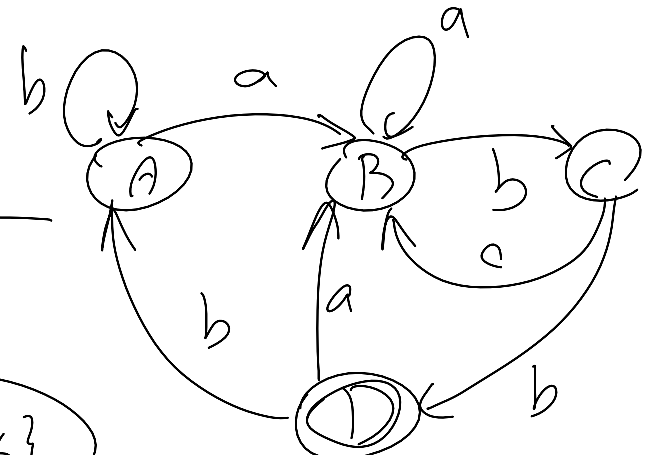
حل) $(a|b)^* a b b \#$



n	followpos(n)
1	{1,2,3}
2	{1,2,3}
3	{4}
4	{5}
5	{6}
6	{ }

Dtrans

Dstates	a	b
A ✓ {1,2,3}	{1,2,3,4}	{1,2,3}
B ✓ {1,2,3,4}	{1,2,3,4}	{1,2,3,5}
C ✓ {1,2,3,5}	{1,2,3,4}	{1,2,3,6}
D ✓ {1,2,3,6}	{1,2,3,4}	{1,2,3}



تمرین) $(a|b)^* a b b \rightarrow$ NFA $\rightarrow$ DFA