الگوریتم تبدیل NFA به DFA

الگوریتم تبدیل NFA به DFA

1- گرفتن NFA از کاربر

2- اعمال الگوریتم تبدیل

3- چاپ DFA خروجی

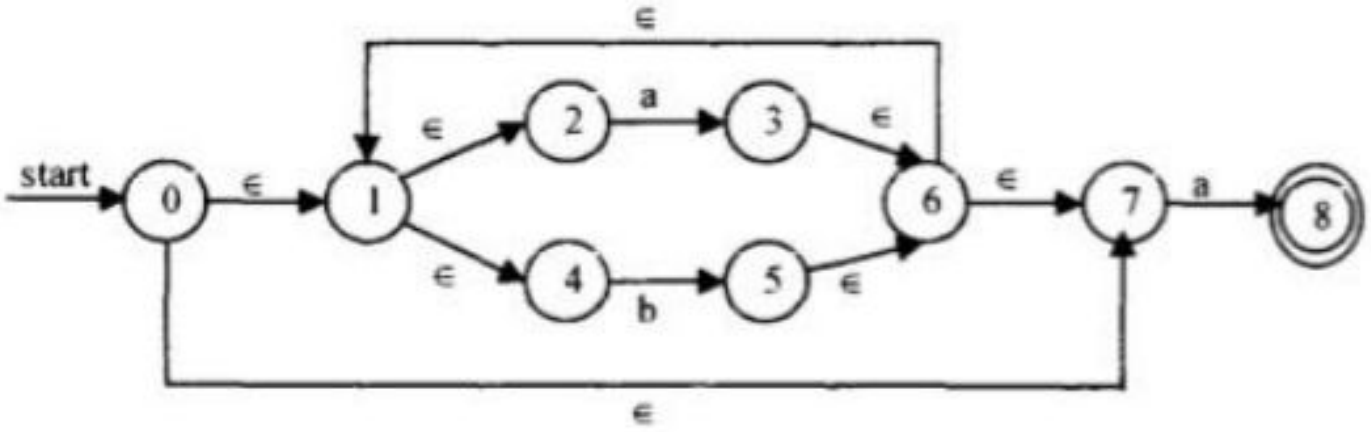
الگوریتم تبدیل NFA به DFA

1- گرفتن NFA از کاربر

2- اعمال الگوریتم تبدیل

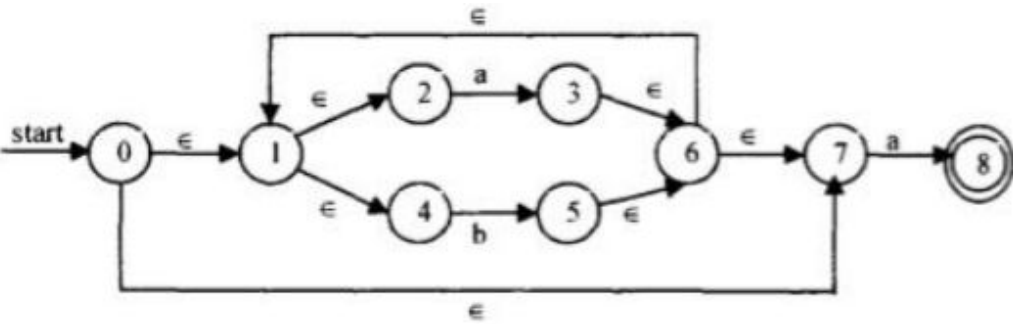
3- چاپ DFA خروجی

1- گرفتن NFA از کاربر



شکل ۲-۳۱NFA مربوط به (a|b)*a

1- گرفتن NFA از کاربر



شکل ۲-۳۱ NFA مربوط به (a|b)*a

	epsilon	a	b
0	1-7	-	-
1	2-4	-	-
2	-	3	-
3	6	-	-
4	-	-	5
5	6	-	-
6	7-1	-	-
7	-	a	-
8	-	-	-

1- گرفتن NFA از کاربر

	epsilon	a	b
0	**1-7**	**-**	**-**
1	**2-4**	**-**	**-**
2	**-**	**3**	**-**
3	**6**	**-**	**-**
4	**-**	**-**	**5**
5	**6**	**-**	**-**
6	**7-1**	**-**	**-**
7	**-**	**a**	**-**
8	**-**	**-**	**-**

q0= epsilon->1,epsilon->7

q1= epsilon->2,epsilon->4

q2= a->3

q3= epsilon->6

.

.

.

1- گرفتن NFA از کاربر

q0= epsilon->1,epsilon->7

q1= epsilon->2,epsilon->4

q2= a->3

q3= epsilon->6

.

.

.

q0= 1,7

q1= 2,4

q2= a3

q3= 6

.

.

.

Code :

1- گرفتن NFA از کاربر

```python
states = {}
for i in range(100):
    user_input = input(f'q{i}: ')
    state = user_input.split(",")

    if 'end' in state:
        last_state = len(states)
        break

    states[f'q{i}'] = state
```

```python
states = {'q0': ['1', '3'], 'q1': ['a2'], 'q2': ['5']}
```

الگوریتم تبدیل NFA به DFA

1- گرفتن NFA از کاربر

2- اعمال الگوریتم تبدیل

3- چاپ DFA خروجی

2- اعمال الگوریتم تبدیل

۱- ϵ_closure(s) را محاسبه می‌کنیم (s حالت شروع NFA است). به مجموعه ϵ_closure(s) یک نام اختصاص می‌دهیم (مانند A) و به DTrans اضافه می‌کنیم. این مجموعه حالت شروع DFA است.

2- اعمال الگوریتم تبدیل

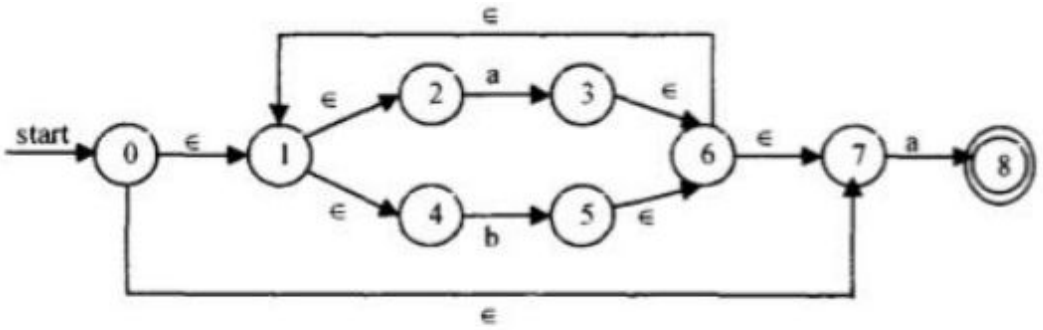
مرحله اول :

الف) گرفتن اپسیلون کلوژور حالت شروع

ب) اضافه کردن آن به جدول DTrans

2- اعمال الگوریتم تبدیل

الف) گرفتن اپسیلون کلوژور حالت شروع



شکل ۲-NFA۳۱ مربوط به (a|b)*a

q0= 1,7

q1= 2,4

q2= a3

q4= b5

-
-
-

2- اعمال الگوریتم تبدیل

الف) گرفتن اپسیلون کلوژور حالت شروع

```python
def ep(s):
    state = {int(x) for x in states[f'q{s}'] if 'a' not in x and 'b' not in x}
    if not state:
        return {s}
    for i in list(state):
        state.update(ep(i))
    state.add(int(s))
    return state
```

```python
def multy_ep(ts):
    result = set()
    for i in ts:
        result.update(ep(i))
    return result
```

```python
ep_first = ep(0)
```

2- اعمال الگوریتم تبدیل

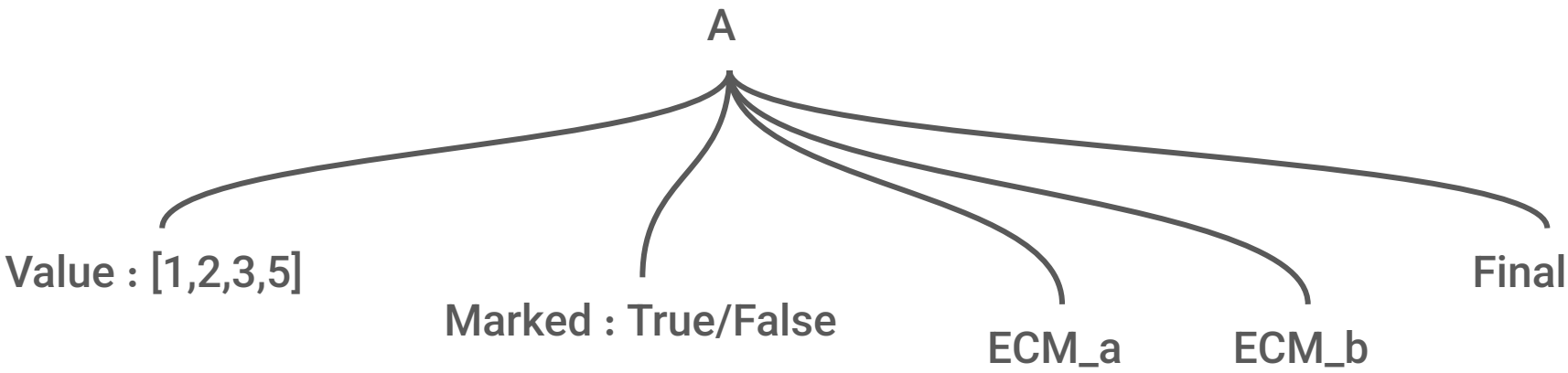
جدول DTrans:

جدول ۲-۱۰ اضافه شدن A به dtrans

	a	b
A		

2- اعمال الگوریتم تبدیل

جدول DTrans:



2- اعمال الگوریتم تبدیل

جدول DTrans:

```
Dtrans = dict()
def add_DTrans(name, value, marked, ecm_a, ecm_b, final):
    DTrans[name] = {"marked": marked, "value": value, "ecm_a": ecm_a, "ecm_b": ecm_b, "final": final}
 # {'A': {'marked':False, 'value':[1,2,4,7], 'ecm_a': 'B', 'ecm_b': 'C', 'final': False}}
```

2- اعمال الگوریتم تبدیل

ب) اضافه کردن آن به جدول DTrans

```
1  ep_first = ep(0)
2  add_DTrans('A', ep_first, False, set(), set(), last_state in ep_first)
3
```

2- اعمال الگوریتم تبدیل

۲- یک حالت علامت نخورده درون Dtrans را یافته (این حالت را در مراحل بعدی T می‌نامیم) و مراحل ذیل را روی آن اعمال می‌کنیم(اولین بار که این مرحله اجرا می‌شود تنها حالت علامت نخورده حالت بدست آمده از مرحله ۱ الگوریتم است). اگر چنین حالتی وجود ندارد الگوریتم پایان می‌یابد.

۳- حالت T علامت می‌زنیم.

2- اعمال الگوریتم تبدیل

٤- برای هر نماد الفبای زبان (این نماد را در مراحل بعدی a می‌نامیم) مراحل ذیل را تکرار می‌کنیم.

٤-١- مجموعه ϵ_closure((move(T,a)) را محاسبه می‌کنیم(این مجموعه را در مراحل بعدی الگوریتم U می‌نامیم). اگر این مجموعه با مجموعه‌هایی که قبلا محاسبه شده و در DTrans موجود است یکسان نباشد، نام جدیدی به آن اختصاص داده و این مجموعه را به DTrans اضافه می‌کنیم (دقت کنید که این حالت علامت نخورده است).

٤-٢- حالت بعدی T به ازای ورودی a است. در نتیجه در DTrans بخش ذیل را اضافه می‌کنیم.

DTrans[T,a]=U

2- اعمال الگوریتم تبدیل

مرحله دوم: مارک کردن اعضای جدول DTrans

الف) بررسی وجود عضو مارک نشده در DTrans

ب) درصورت وجود عضو مارک نشده مارک کردن آن

پ . ن : تا وقتی که همه اعضای DTrans مارک نشدن باید ادامه بدیم (حلقه while)

2- اعمال الگوریتم تبدیل

مرحله دوم: مارک کردن اعضای جدول DTrans

الف) بررسی وجود عضو مارک نشده در DTrans

```
def check_unmarkeds():
    return next((i for i, info in DTrans.items() if not info['marked']), None)

```

2- اعمال الگوریتم تبدیل

مرحله دوم: مارک کردن اعضای جدول DTrans

ب) درصورت وجود عضو مارک نشده مارک کردن آن

■ مارک کردن : محاسبه epsilon_closure(move(state,a/b)

2- اعمال الگوریتم تبدیل

مرحله دوم: مارک کردن اعضای جدول DTrans

ب) درصورت وجود عضو مارک نشده مارک کردن آن

Move function :

q0= 1,7

q1= 2,4

q2= a3

q4= b5

```python
def move(t, s):
    state = [x for x in states[f'q{t}'] if s in x]
    return state[0].replace(s, '') if state else ''

def multy_move(ts, s):
    result = set()
    for i in ts:
        result.update(move(i, s))
    return result
```

2- اعمال الگوریتم تبدیل

مرحله دوم: مارک کردن اعضای جدول DTrans

ب) درصورت وجود عضو مارک نشده مارک کردن آن

■ مارک کردن : محاسبه epsilon_closure(move(state,a/b)

الف) محاسبه move حالت / حالت ها

ب) محاسبه epsilon_closure حاصل move

ج) بررسی وجود نتیجه در DTrans - اضافه کردن حاصل به DTrans درصورت نبود

د) قرار دادن نام حاصل

Epsilon_closure - Move function :

2- اعمال الگوریتم تبدیل

مرحله دوم: مارک کردن اعضای جدول DTrans

ب) درصورت وجود عضو مارک نشده مارک کردن آن

■ مارک کردن : محاسبه epsilon_closure(move(state,a/b)

```
1  def epc_move(ts, s):
2      epc = multy_move(ts, s)
3      return multy_ep(int(x) for x in epc)
```

2- اعمال الگوریتم تبدیل

مرحله دوم: مارک کردن اعضای جدول DTrans

ب) درصورت وجود عضو مارک نشده مارک کردن آن

■ مارک کردن : ج) بررسی وجود نتیجه در DTrans - اضافه کردن حاصل به DTrans درصورت نبود

```
1  def check_Dtrans(value):
2      return next((key for key, info in DTrans.items() if info['value'] == value), None)
3
```

2- اعمال الگوریتم تبدیل

مرحله دوم: مارک کردن اعضای جدول DTrans

```
# Second Step: Marking DTrans
while unmarked := check_unmarkeds():
    value = DTrans[unmarked]['value']

    # Compute ecm_a
    ecm_a = epc_move(value, 'a')
    if not (name := check_Dtrans(ecm_a)):
        name = DT_names.pop()
        add_DTrans(name, ecm_a, False, set(), set(), last_state in ecm_a)
    DTrans[unmarked]['ecm_a'] = name

    # Compute ecm_b
    ecm_b = epc_move(value, 'b')
    if not (name := check_Dtrans(ecm_b)):
        name = DT_names.pop()
        add_DTrans(name, ecm_b, False, set(), set(), last_state in ecm_b)
    DTrans[unmarked]['ecm_b'] = name

    DTrans[unmarked]['marked'] = True
```

الگوریتم تبدیل NFA به DFA

1- گرفتن NFA از کاربر

2- اعمال الگوریتم تبدیل

3- چاپ DFA خروجی

3- چاپ DFA خروجی

```python
# Output DFA
for i in DTrans:
    ecm_a = DTrans[i]['ecm_a']
    ecm_b = DTrans[i]['ecm_b']
    print(f"{i}---(a)--->{ecm_a}")
    print(f"{i}---(b)--->{ecm_b}")

```