По заданному регулярному выражению (Вариант 10)

- Построить недетерминированный КА;
- По полученному НДА построить ДКА;
- Минимизировать полученный ДКА;
- Для минимального ДКА написать программу-распознаватель предложений языка, порождаемого регулярным выражением.

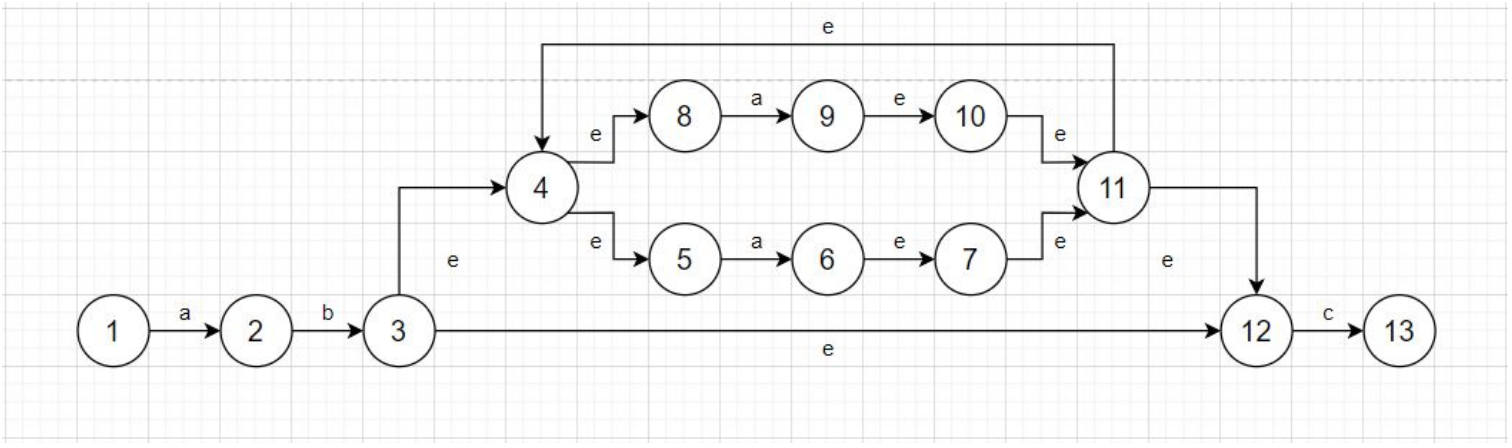
Продемонстрировать работу распознавателя на различных примерах (не менее трех правильных) предложений.

Использование символов + и ? в регулярных выражениях.
Символ + используется для определения регулярного выражения, повторяющегося один или более раз. В этом смысле $p^+ = pp^*$.
Символ ? используется для указания того, что регулярное выражение встречается ноль или один раз, тогда p? = ε|p.

Внимание!
Операции итерации, конкатенации и объединения имеют приоритеты, причем приоритет итерации высший, а объединения – низший. Обычно скобки будут опускаться везде, где их отсутствие не влияет на определение регулярного множества. Регулярное выражение ((a)(b*))|(c) может быть записано следующим образом: ab*|c.

Вариант 10: ab(ac|ab)*c

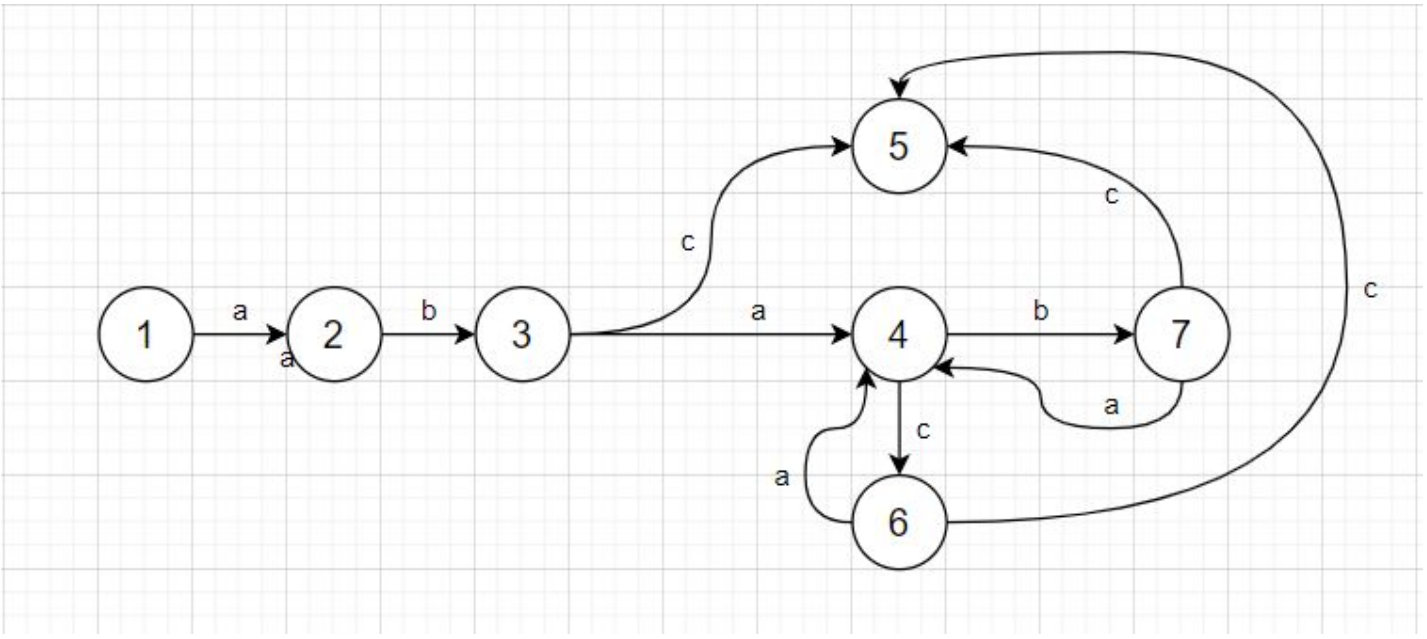
НКА



ДКА

Таблица состояний ДКА из состояний НКА:

Состояние	a	b	c
1	2	-	-
2	-	3,4,5,8,12	-
3,4,5,8,12	6,9	-	13
6,9	-	4,5,7,8,11,12	4,5,8,10,11,12
13	-	-	-
4,5,7,8,11,12	6,9	-	13
4,5,8,10,11,12	6,9	-	13



Можно еще минимизировать.

Таблица переходов

№	a	b	c
1	2	-	-
2	-	3	-

3	4	-	5
4	-	6	7
5	-	-	-
6	4	-	5
7	4	-	5

Минимизация ДКА

Построим таблицу состояний

P0 = [A0 = {1,2,3,4,6,7}, B0 = {5}]

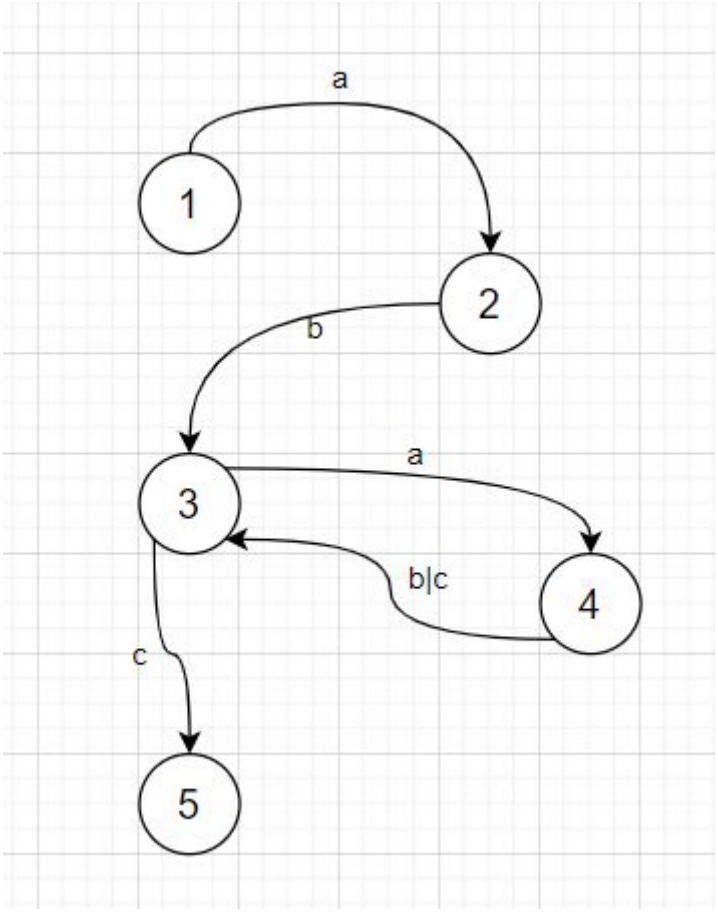
При дальнейшем разбиение получим следующие группы эквивалентности

P1 = [A1 = {1}, B1 = {2}, C1 = {3,6,7}, D1 = {4}, E1 = {5}]

P2 = [A2 = {1}, B2 = {2}, C2 = {3,6,7}, D2 = {4}, E2 = {5}]

				P0			P1			P2		
S	a	b	c	a	b	c	a	b	c	a	b	c
1	2	-	-	A0	-	-	B1	-	-	B2	-	-
2	-	3	-	-	A0	-	-	C1	-	-	C2	-
3	4	-	5	A0	-	B0	D1	-	E1	D2	-	E2
4	-	6	7	-	A0	A0	-	C1	C1	-	C2	C2
5	-	-	-	-	-	-	-	-	-	-	-	-
6	4	-	5	A0	-	B0	D1	-	E1	D2	-	E2
7	4	-	5	A0	-	B0	D1	-	E1	D2	-	E2

Получаем МНКА



Программа распознаватель

Исходный код:
https://github.com/MansurovB-source/The-development-of-compilers

Пример:
Строка abc - корректная

```
The expression is - ab(ac|ab)*c
Let's check if the string matches the regular expression
Enter your string (max length is a 1024 ASCII characters):
abc
Your string matches a regular expression
```

Строка abcc - некорректная

```
The expression is - ab(ac|ab)*c
Let's check if the string matches the regular expression
Enter your string (max length is a 1024 ASCII characters):
abcc
Your string contains characters, not from our alphabet or the string does not match a regular expressions
```

Строка abacacababacc – корректная