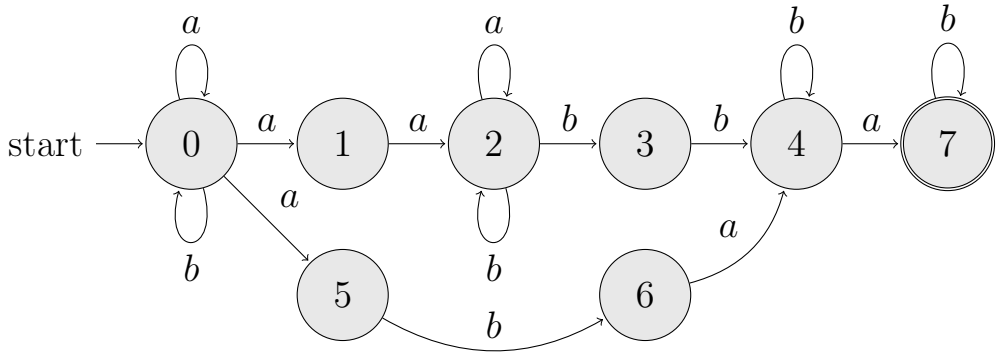
Formal languages HW4

Кушнерюк Сергей, Б05-925

Домашнее задание 4

Задача 1



Verticle	a	b
0	015	0
015	0125	06
0125	0125	0236
06	0145	0
0236	01245	0234
0145	01257	046
01245	01257	02346
0234	01257	0234
01257	0125	02367
046	01457	04
02346	012457	0234
02367	01245	02347
01457	01257	0467
04	0157	04
012457	01257	023467
02347	01257	02347
0467	01457	047
0157	0125	067
023467	012457	02347
047	0157	047
067	0145	07
07	015	07

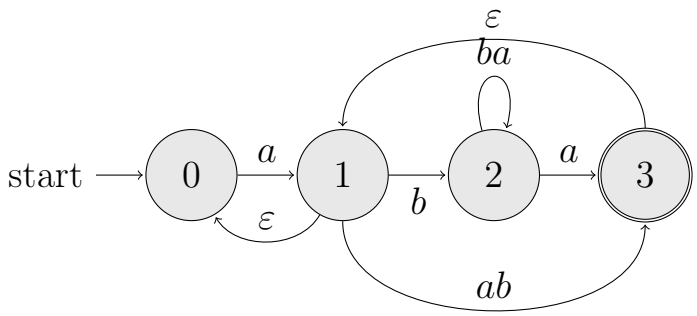
Как видим по таблице, это уже ПДКА. Минимизируем

vert.	type	a	b	type	a	b	type	a	b	type	a	b	type	a	b	type
0	0	0	0	1	1	1	1	1	1	1	2	1	1	2	1	1
015	0	0	0	1	1	1	1	1	2	2	3	4	2	3	4	2
0125	0	0	0	1	1	1	1	1	3	3	3	5	3	3	5	3
06	0	0	0	1	2	1	2	4	1	4	6	1	4	6	1	4
0236	0	0	0	1	2	2	3	4	4	5	6	7	5	6	7	5
0145	0	1	0	2	3	2	4	5	6	6	8	9	6	8	9	6
01245	0	1	0	2	3	2	4	5	6	6	8	9	6	8	9	6
0234	0	1	0	2	3	2	4	5	4	7	8	7	7	8	7	7
01257	1	0	1	3	1	3	5	1	7	8	3	10	8	3	10	8
046	0	1	0	2	4	2	6	8	4	9	11	7	9	11	12	9
02346	0	1	0	2	4	2	6	8	4	9	11	7	9	11	7	10
02367	1	0	1	3	2	4	7	4	8	10	6	12	10	6	13	11
01457	1	1	1	4	3	4	8	5	9	11	8	13	11	8	14	12
04	0	1	0	2	3	2	4	5	4	7	14	7	12	15	12	13
012457	1	1	1	4	3	4	8	5	9	11	8	13	11	8	14	12
02347	1	1	1	4	3	4	8	5	8	12	8	12	13	8	13	14
0467	1	1	1	4	4	4	9	8	8	13	11	12	14	11	16	15
0157	1	0	1	3	1	3	5	1	10	14	3	15	15	3	17	16
023467	1	1	1	4	4	4	9	8	8	13	11	12	14	11	13	17
047	1	1	1	4	3	4	8	5	8	12	14	12	16	15	16	18
067	1	0	1	3	2	3	10	4	5	15	6	16	17	6	18	19
07	1	0	1	3	1	3	5	1	5	16	2	16	18	2	18	20

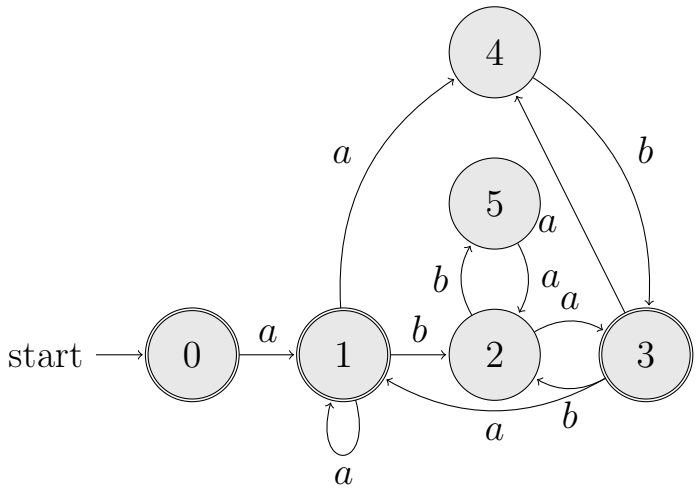
vert.	type	a	b	type	a	b	type
0	1	2	1	1	2	1	1
015	2	3	4	2	3	4	2
0125	3	3	5	3	3	5	3
06	4	6	1	4	6	1	4
0236	5	6	7	5	7	8	5
0145	6	8	9	6	9	10	6
01245	6	8	10	7	9	11	7
0234	7	8	7	8	9	8	8
01257	8	3	11	9	3	12	9
046	9	12	13	10	13	14	10
02346	10	12	7	11	15	8	11
02367	11	6	14	12	7	16	12
01457	12	8	15	13	9	17	13
04	13	16	13	14	18	14	14
012457	12	8	17	15	9	19	15
02347	14	8	14	16	9	16	16
0467	15	12	18	17	13	20	17
0157	16	3	19	18	3	21	18
023467	17	12	14	19	15	16	19
047	18	16	18	20	18	20	20
067	19	6	20	21	6	22	21
07	20	2	20	22	2	22	22

Как видим, исходный ПДКА уже был минимальным

Задача 2



Делаем однобуквенным



Делаем ПДКА

Получается достаточно большой граф, поэтому вот табличка. С помощью acc помечены завершающие вершины

Verticle	a	b
0 acc	1	drain
1 acc	14	2
14 acc	14	23
2	3	5
23 acc	134	25
3 acc	14	2
5	2	drain
134 acc	14	23
25	23	5
drain	drain	drain

Минимизируем

vert.	type	a	b	type	a	b	type	a	b	type	a	b	type	a	b	type	a	b	type
0	1	1	0	1	1	3	1	2	3	1	2	5	1	2	4	1	2	6	1
1	1	1	0	1	2	1	2	2	1	2	3	2	2	2	3	2	3	4	2
14	1	1	1	2	2	1	2	2	2	3	3	2	2	2	2	3	3	2	3
2	0	1	0	1	1	3	1	2	1	2	2	4	3	2	1	4	2	5	4
23	1	1	0	1	2	1	2	2	1	2	3	2	2	2	3	2	3	4	2
3	1	1	0	1	2	1	2	2	1	2	3	2	2	2	3	2	3	4	2
5	0	0	0	3	1	3	1	1	3	4	2	5	1	3	4	5	4	6	5
134	1	1	1	2	2	1	2	2	2	3	3	2	2	2	2	3	3	2	3
25	0	1	0	1	1	3	1	2	1	2	2	4	3	2	1	4	2	5	4
drain	0	0	0	3	3	3	3	3	3	5	5	5	4	4	4	6	6	6	6

МинПДКА выглядит так

Verticle	a	b
0 acc	1+23+3	drain
1+23+3 acc	14+134	2+25
14+134 acc	14+134	1+23+3
2+25	1+23+3	5
5	2+25	drain
drain	drain	drain

где плюс означает, что вершины в одном классе эквивалентности
Теперь, инвертируем и получаем требуемое

Verticle	a	b
0	1+23+3	drain
1+23+3	14+134	2+25
14+134	14+134	1+23+3
2+25 acc	1+23+3	5
5 acc	2+25	drain
drain acc	drain	drain

Последние две задачи решены с помощью кода на github

Задача 3.

В силу того, что множество автоматных и регулярных языков совпадают, то докажем, что регулярное выражение $1+a^+(b^+a^+)^*+b^+(a^+b^+)^*$ задает язык $\{w \in \{a,b\}^* \mid |w|_{ab} = |w|_{ba}\}$ По построению регулярного выражения видим, что число $|w|_{ab} = |w|_{ba}$:

1. Для пустого слова понятно
2. В втором блоке после первого плюса видим, что получаются слова состоящие их k блоков из ненулевого количества букв a, а между ними k-1 блок из ненулевого количества букв b для некоторого k натурального. При этом, на местах перехода от одного блока к другому, увеличивается либо $|w|_{ab}$, либо $|w|_{ba}$. При этом эти моменты также чередуются (так как сначала был переход от a к b, затем от b к a и тд) и этих переходов суммарно 2k - 2 (справа и слева от каждого блока с b-шками)
3. Аналогично предыдущему утверждению в точности, только меняем a на b

Таким образом $L_{reg} \subset L_{descr}$

С другой стороны, пусть слово непусто. Тогда пусть БОО оно начинается на букву a. Теперь разобьем слово по переходам от a к b и от b к a. Заметим, что они обязаны чередоваться, так как иначе есть например, два подряд идущих перехода от a к b, но тогда где-то между ними обязан быть переход от b к a

$$\dots ab\, bbb \underbrace{\dots\dots\dots\dots\dots\dots\dots\dots}_{\text{somewhere here should be ba transition}} aaa\, ab \dots$$

Тогда все слово этими переходами разбивается на блоки из a и из b. Так как первая буква a, то первый блок состоит из букв a. В силу того, что что $|w|_{ab} = |w|_{ba}$, то всего переходов от одной букве к другой четное число, следовательно первый и последний блок - блоки из букв a. Т.е. подходит под второе выражение (после первого плюса) регулярного выражения. Аналогично и для b

Задача 4.

Воспользуемся задачей с семинара, точнее ею немного измененной (Задача 1 (e)) $\{a^m b^n | m \leqslant n, m \in \mathbb{P}\}$

Почему язык из этой задачи не является автоматным? Воспользуемся напрямую леммой о разрастании, точнее ее отрицанием. $\forall p$ рассмотрим слово в $a^q b^{q+1}, q \in \mathbb{P}, q > p$. Тогда все условия соблюдены, слово в языке, тогда для любого x, y, z подходящих под условие $|xy| < p < q, |y| < 1 \implies y = a^r, r \geqslant 1$. Далее взяв k хотя бы 2 получим слово $xy^k z = a^u b^v \quad u > v$. Т.е. язык не является автоматным

Теперь рассмотрим пересечения языка $\{a^m b^n a^m | m \leqslant n, m \in \mathbb{P}\}$ с автоматным языком $\{a^m b^n | n, m \in \mathbb{N}\}$. Заметим, что в пересечении в точности язык, для которого мы доказали, что он не автоматный. Следовательно, язык в нашей задаче не может быть автоматным

Задача 5

Примем, что у нас бесконечная память (ибо иначе с какого-то момента код определенной длины просто не получится сохранить в памяти). Тогда воспользуемся отрицанием леммы о возрастании, $\forall p$ рассмотрим такой код

```
int main(){{...{{{int a;}}}...}}
```

где количество открывающих скобок в точности p. Пусть y будет состоять только из {. Тогда какой бы набор x, y, z подходящих под условие леммы мы не взяли, получим, что $y = \{^l$, а значит взяв $k \neq 1$ получим, что количество открывающих и закрывающих фигурных скобок для scope-ов будет отличаться $\implies$ код не скомпилируется

С другой стороны, рассмотрим какие-то еще такие варианты y:

1. Подслово в int. Тогда взяв k = 0 получим либо невалидный, либо отсутствующий тип функции
2. Подслово в main. Взяв k = 0 получим undefined reference to main
3. Пробел между int и main. Взяв k = 0 получим опять невалидное объявление

   Вообще говоря, все что коснется int main() удалением вызывает ошибку неправильного объявления int main или его отсутствие: затрагивание круглых скобок, например, также приведет к неправильному объяв-

лению функции, как и любая комбинация предыдущих, например $y =$ "$nt\ main($"

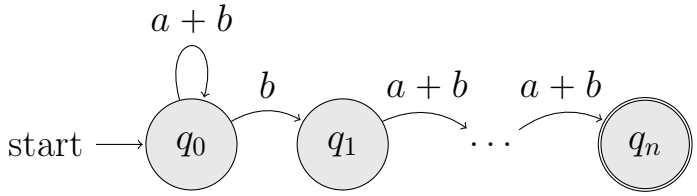
4. Все, что коснется хоть одной фигурной скобки сведется к ситуации в самом начале

Таким образом, (неприятным разбором случаев) доказали, что язык компилируемых программ на C++ не является автоматным

(Изначально хотелось еще попровать как-то раздуть какую-нибудь переменную i так, чтобы при вызове i-того элемента, например, вектора, появлялся выход за границы массива, т.е. проблема была бы не в компиляции, а в исполнении программы)

Задача 6

Рассмотрим язык $L_n = \{w \mid w[-n] = b\},\ \Sigma = \{a, b\}$. Докажем, что следующий НКА задает именно этот язык



С одной стороны, по построению видим, что автомат задает именно такие слова: произвольный префик, затем буква b, затем n-1 произвольных букв. С другой стороны, любое слово из языка L_n можно разбить по n-той с конца букве b на две части, первая из которой задается циклом, а последняя часть фиксированной длины n-1 получается проходом по n-1 переходам от q_1 к q_n

Теперь проэмулируем построение ДКА из данного НКА. Начнем мы с состояния q_0. Будем теперь группы состояний называть по номерам содержащихся там состояний. Заметим, что в силу существующих переходов, 0 будет в любом состоянии. Пусть у нас есть некоторый набор состояний $0i_1 \dots i_k$. Заметим, что по построению НКА из него мы попадем по букве a в набор $0(i_1 + 1) \dots (i_k + 1)$, а по букве b в набор $01(i_1 + 1) \dots (i_k + 1)$.

Рассматривая теперь каждое состояние как битовую маску длины n+1 (для этого нужно будет считать, что если число l лежит в наборе состояний, то на l-той позиции стоит 1, иначе 0). Тогда, переход по a - побитовый сдвиг влево, по b - побитовый сдвиг влево и +1.

Например, из маски 1 (ведущие нули пока не смотрим) мы попадаем в 1 или 11, из 1001 в 10001 и 10011 по a и b соответственно и т.д. По построению видим, что последний бит всегда 1. Тогда также по построению понятно, как добраться до произвольного набора состояний: убирая каждый раз 2-ой с конца бит мы получаем, по каким наборам состояния мы добрались до нашего. Например:

$$10011 \implies 1001 \implies 101 \implies 11 \implies 1$$

Состояний 2^n в этом ДКА (по количеству масок). Докажем теперь, что именно состояния, соответствующие таким наборам и будут состояниями минимального ПДКА.

Чтобы не путать теперь состояний ПДКА и НКА, будем их теперь называть ПДКА-состояниями и НКА-состояниями соответственно

От противного, пусть минПДКА-состояний меньше (больше их быть не может). Тогда так как они соответствуют классам эквивалентности ПДКА-состояний, то по принципу Дирихле найдутся два различных набора НКА-состояний, который будут соответствовать одному классу. Раз они различны, то есть НКА-состояние, которое входит в одно из них, но не входит в другое.

Пусть это НКА-состояние с номером d. Пойдем по этим наборам по набору букв a^{n-d}. По сути это будет побитовый сдвиг влево на n-d. Тогда различавший их бит встанет на самую первую позицию. Заметим, что тогда мы дойдем до ПДКА-состояний, одно из которых является завершающим, а другое нет (ПДКА-состояние будет завершающим тогда и только тогда, когда в него входило завершающее НКА-состояние. В нашем случае оно было только одно - q_n. т.е. соответствующее самому первому биту). Но тогда два этих ПДКА-состояний не эквиваленты, следовательно полученный ПДКА и был минимальным

Все эти размышления работали для произвольного L_n. Значит для любого n нашли требуемый язык, константа c при этом будет равно в точности $\frac{1}{2}$