Домашняя работа по ТРЯП №2

Автор - Айвазов Денис из 671 группы

14 сентября 2017

1 НКА по языку L_3. Построить ДКА по НКА

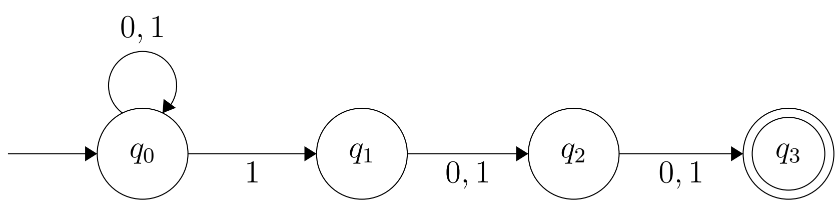
Следуя алгоритму построим ДКА из НКА(алгоритм как на семинаре, но немного адаптированный под две таблички)

δ_1	0	1
q_0	q_0	q_0, q_1
q_1	q_2	q_2
q_2	q_3	q_3
q_3	-	-

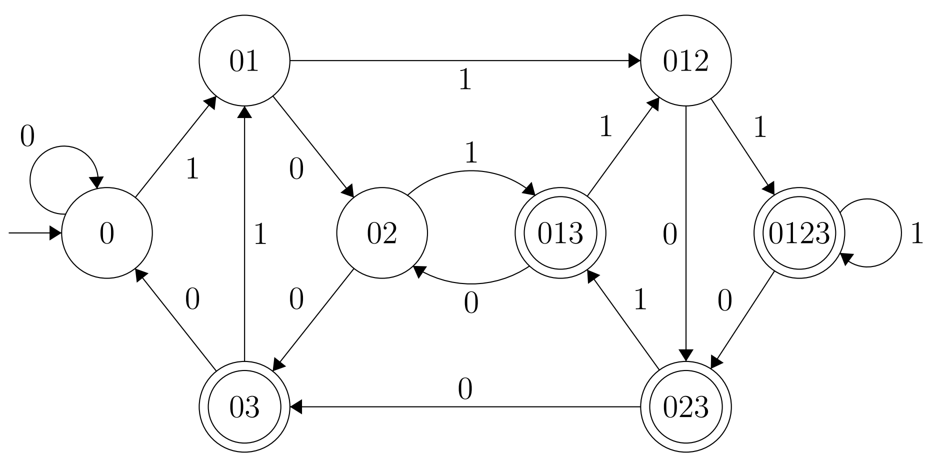
$\Rightarrow$

δ_2	0	1
0	0	01
01	02	012
02	03	013
012	023	0123
03	0	01
013	02	012
023	03	013
0123	023	0123

из этого НКА:



Принимающие состояния - все состояния, содержащие 3 в названии Получим этот прекрасный симметричный детерменированный автомат на 8ми вершинах. Доказательство не требуется потому что мы просто тренировались применять алгоритм, разобранный на семинаре.

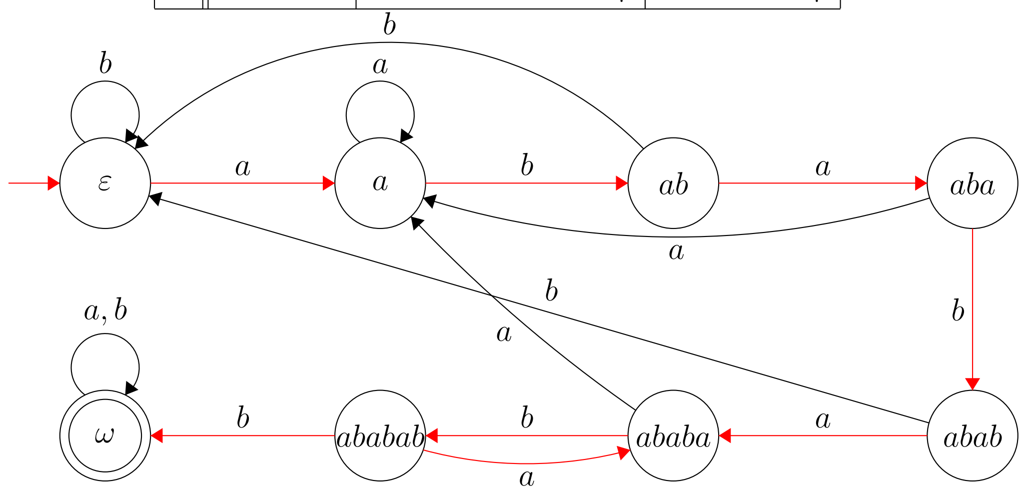


2 Доказать экспоненциальный разрыв

3 КМП-автомат и его демонстрация

Построим автомат по алгоритму приведённому на листке с заданием.

k	ω	a	b
0	ε	ω_1	$l(ab) = \varepsilon$
1	a	l(aa)=a	ω_2
2	ab	ω_3	$l(abb) = \varepsilon$
3	aba	$l(abaa) = a$	ω_4
4	abab	ω_5	$l(ababb) = \varepsilon$
5	ababa	$l(ababaa) = a$	ω_6
6	ababab	$l(abababa) = ababa$	ω_7
7	abababb	ω_7	ω_7



Демонстрация. Так же на самом ДКА для удобства отмечено красными стрелками, как будет обрабатываться слово ababababab
$\varepsilon(a) \to a(b) \to ab(a) \to aba(b) \to abab(a) \to ababa(b) \to ababab(a) \to abababa(b) \to abababab(b) \to ababababb$ — final

4 in и out для ДКА

Напишем эти алгоритмы на псевдокоде с элементами языка Си.
В начале для определённости договоримся, что $\Sigma = \{a, b\}$, ДКА состоит только из непринимающего состояния $q_0 = \varepsilon$ без переходов из него. Множество состояний в начале $Q = \{\varepsilon\}$. А множество принимающих состояний F в начале пусто. так же договоримся, что запись вида $\omega_i = \omega[1, i]$ - как и в КМП это префикс слова ω (первые i букв). И договоримся о нумерации букв в слове с 1. Так же договоримся что $\omega_0 = \varepsilon$ и всегда присутствует в словаре и на него можно ссылаться и переходить
Алгоритм in по добавлению (несуществующего в словаре) слова: Дока-

IN:
получили слово ω
for *(i=1; $i \leq |\omega|$; i++)* **do**
    if $\exists \delta(\omega_{i-1}, \omega[i])$ **then**
        если переход есть, "переходим"в это состояние. Т.е. буква обработана.
    else
        if $\nexists \omega_i \ \mathit{in} \ Q$ **then**
            если вершины нет, добавим вершину ω_i в Q
        end
        $\delta(\omega_{i-1}, \omega[i]) = \omega_i$ - добавили переход.
    end
end
Добавим в F вершину ω т.е. сделаем её принимающей. Теперь дойдя по всем переходам до состояния ω, наш ДКА будет принимать это слово.

зательство принимания слова следует из математической индукции, которое содержится в самом цикле For(база - пустое слово на начальном состоянии, шаг по буквам и приём описан в алгоритме и в итоге мы получаем, что проходим слово до конца и оно принимается). И слово принимается потому что соответствующее ему состояние принимающее. А доказательство того, что никаких случайных других слов мы не добавили следует сразу из того, что единственная новое принимающее состояние это состояние ω.

Алгоритм процедуры out гораздо проще, но сначала опишем булеву функцию test:
Теперь напишем алгоритм по удалению слова из словаря: out

TEST:
получили слово ω которое надо проверить.
По умолчанию test=false.
for *(i=1; $i \leq |\omega|$; i++)* **do**
  if $\delta(\omega_{i-1}, \omega[i]) = \omega_i$ **then**
    break; - перехода нет, значит слово точно не принято.
  end
  Ну а если он есть, то цикл продолжается дальше.
end
if *($\omega_i \in F$)* **then**
  test=true - мы дошли до состояния соответствующего слову и оно принимающее
end
return test; Доказательство алгоритма так же по индукции цикла For.

OUT:
получили слово ω которое надо удалить.
просто удаляем его вершину из множества принимающих состояний: delete(ω,F) - просто какая-то функция удаления из множества. Уже от языка программирования зависит.(Нам даже переходы удалять не надо, на принимаемость влияет только принадлежность состояния множеству F)

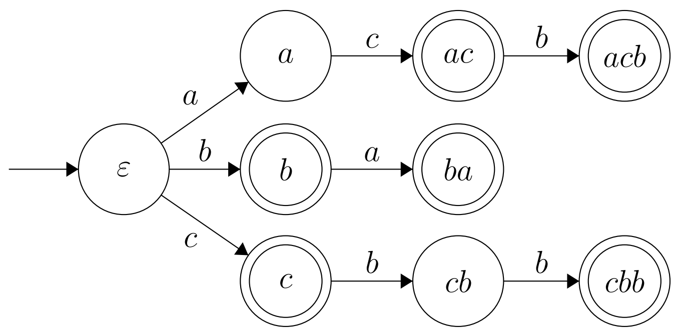
5 ДКА по множеству S

Еще раз повторюсь, мы договорились о всех условностях сказанных выше. И мы расчитываем на разумность словаря в случайный момент времени (он принимает только нужные ему слова, нет случайных состояний болтающихся в воздухе без связей и все такое, о чём говорит здравый смысл). Далее напишем алгоритм принимающий набор слов из множества S и строющий по нему ДКА-словарь. Таким образом мы будем извлекать слова из S и добавлять их в ДКА пока S не станет пустым множеством.

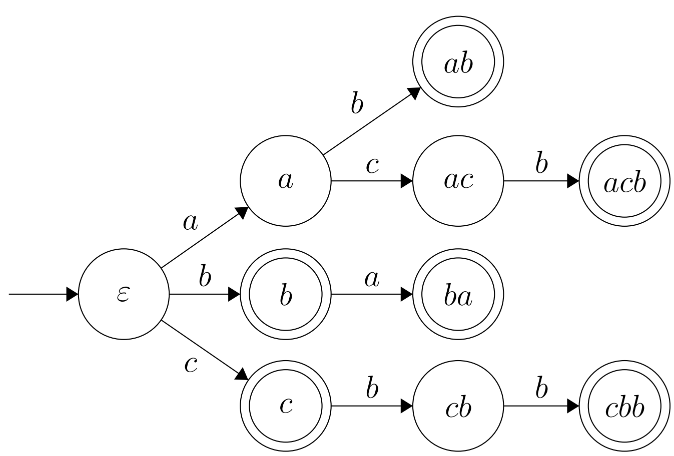
MAKE DFA:
Получили на вход множество слов S.
while *($\omega = getfromset(S)$!=NULL)* **do**
    delete(ω,S) получаем в ω слова из S в каком-то порядке и удаляем их из S.
    if *(test(ω)!=true)* **then**
        in(ω) - добавляем слово в ДКА с помощью описанной выше процедуры
    end
end
Конец. Теперь в ДКА лежат все слова из S

6 ДКА для заданного словаря

Построим ДКА по предыдущему алгоритму. Изначально у нас есть только непринимающее состояние ε. Далее получаем слова поочереди из S. Первое слово: ac. Видим, что состояния a нет и добавляем его(и переход к нему через a). Далее добавляем состояние ac (и переход к нему по c) и делаем это состояние принимающим. Аналогично со всеми остальными словами из S.



Добавим ab и удалим ac.
По алгоритму in из предыдущей задачи для добавления ab мы сначала смотрим, есть ли это состояние. Если его нет, то побуквенно обрабатываем: переход по a в состояние a есть. А вот перехода по b в состояние ab(как и самого состояния) нет. Создаём состояние ab, делаем его принимающим и создаём переход по b к ab. Теперь наш ДКА принимает ab ведь все переходы до него существуют и определены, а само состояние - принимающее. Так же заметим, что при этих изменениях мы не добавили никаких новых слов кроме ab, ведь кроме него новых принимающих состояний создано не было.
Теперь по алгоритму out удалим слово ac. Поскольку мы договорились о том, что нас не будут просить удалить несуществующее слово (ну или иначе просто проводим в начале проверку $test(\omega)$ и если слова нет, то всё уже хорошо. Если же слово есть, то выполняем дальше по алгоритму), то мы просто удаляем состояние ω из множества принимающих.



7 Протокол работы алгоритма КМП

Реализуем алгоритм КМП для слова abbbababbab#abba основываясь на книге Шеня(ссылка была в листочке с заданием). И напишем протокол работы

i	len	a	b	b	b	a	b	a	b	b	a	b	#	a	b	b	a
1	0	0	0														
2	0	0	0	0													
3	0	0	0	0	1												
4	1	0	0	0	1	0											
5	0	0	0	0	1	0	1										
6	1	0	0	0	1	0	1	2									
7	2	0	0	0	1	0	1	2	3								
8	3	0	0	0	1	0	1	2	3	0							
9	0	0	0	0	1	0	1	2	3	0	1						
10	1	0	0	0	1	0	1	2	3	0	1	2					
11	2	0	0	0	1	0	1	2	3	0	1	2	1				
12	1	0	0	0	1	0	1	2	3	0	1	2	1	2			
13	2	0	0	0	1	0	1	2	3	0	1	2	1	2	3		
14	3	0	0	0	1	0	1	2	3	0	1	2	1	2	3	4	
15	4	0	0	0	1	0	1	2	3	0	1	2	1	2	3	4	2

abba найдено в abbbababbab т.к. у нас в массиве l(основная таблица) нашлось число 4= длине abba.