КОНСТРУКЦИИ РЕГУЛЯРНОГО ЯЗЫКА

(kmp-урок рисования лингвистических графов)

Пусть регулярный язык задан словесным описанием «Множество цепочек из {0,1,a,b}*, заканчивающихся цепочкой ’01a’ и содержащих чётное количество символов ‘a’».

Требуется построить различные конструкции, задающие этот язык:

- регулярное выражение,
- регулярную,
- КС- грамматику
- детерминированный конечный автомат.

Построим для этого языка регулярное выражение.

Для начала определим минимально возможную цепочку этого языка. Самая короткая возможная цепочка ‘a01a’ – она заканчивается требуемой цепочкой ’01a’ и содержит 2 символа ‘a’. Это основа нашего регулярного выражения. Удлинить эту цепочку можно, поместив перед ней или перед цепочкой ’01a’ в ней любое количество любых символов алфавита, за исключением символа ‘a’ – для него нужно подсчитывать количество, поэтому его надо рассматривать особо. Заданную цепочку ’01a’ разрывать нельзя. Для генерации любого количества символов используется итерация, а выбор одного из нескольких символов записывается в виде суммы.

Итак, получим $(0+1+b)^*\underline{a}(0+1+b)^*\underline{01a}$.

Но по построенному таким образом выражению нельзя получить много символов ‘a’. Для увеличения количества ‘a’ можно добавить перед построенным выражением $(aa)^*$ для сохранения чётности количества символов ‘a’. Но при добавлении такой конструкции нельзя будет между символами ‘a’ поместить другие символы алфавита. Для того чтобы сделать это возможным, заменим добавляемую конструкцию на $((0+1+b)^*a(0+1+b)^*a(0+1+b)^*)^*$, сохранив чётность ‘a’ и добавив любое количество других символов. В общем случае, полученную конструкцию следовало бы поместить и после первого символа ‘a’ в ранее построенном выражении (вместо

выделенной цветом скобки), но в рассматриваемом языке это ничего не изменит, только усложнит выражение. Поэтому итоговое регулярное выражение будет иметь вид: $((0+1+b)^*a(0+1+b)^*a(0+1+b)^*)^*(0+1+b)^*\underline{a}(0+1+b)^*\underline{01a}$.
Можно несколько упростить построенное выражение, убрав одну из скобок $(0+1+b)^*$ – там, где она повторяется.

Окончательный ответ: **$((0+1+b)^*a(0+1+b)^*a)^*(0+1+b)^*\underline{a}(0+1+b)^*\underline{01a}$**.

Построим для этого языка регулярную праволинейную грамматику (в правой части правил вывода нетерминальные символы размещать только справа от терминальных цепочек).

Для повторения символа алфавита любое количество раз используется явная рекурсия на нетерминальном символе (вида A→0A – тогда ‘0’ может повториться хоть сколько раз), для подсчёта чётности применяется чередование двух нетерминалов (A→0B, B→0A). Начнём с минимальной цепочки ‘a01a’. Поскольку она начинается с символа ‘a’, количество которых необходимо подсчитывать, после его генерации нужно изменить нетерминал. Далее идёт неразрываемая обязательная цепочка ‘01a’, следовательно, её нужно получать всю целиком. Это позволит сделать грамматика с правилами S→aA, A→01a. Для того чтобы получить любое количество символов алфавита перед ‘a01a’, добавим рекурсивные правила для целевого символа S: S→0S|1S|bS. Для получения таких же символов после первого ‘a’ в построенной минимальной цепочке добавим такие же рекурсивные правила для нетерминала A: A→0A|1A|bA.

Правила грамматики приняли вид:
S→aA|0S|1S|bS, A→01a|0A|1A|bA.

Осталось дополнить правила генерацией чётного количества символов ‘a’. Поскольку нетерминал A означает, что получен один символ ‘a’ в итоговой цепочке, а S – что нет ещё ни одного символа ‘a’, то можно использовать S для получения любого чётного их количества, а A – любого нечётного, просто чередуя S и A: A→aS, S→aA, причём второе из этих правил уже есть.

Грамматика построена, выпишем её полностью. **G({0,1,a,b},{S,A},P,S)**, где правила P:

S→0S|1S|bS|aA,
A→0A|1A|bA|aS|01a.

Построим КС-грамматику для этого же языка.

Её правила удобно строить по регулярному выражению. Для этого каждую большую скобку обозначаем своим нетерминалом. Если на скобке стоит звёздочка (итерация), значит, на этом нетерминале будет явная рекурсия и пустое правило. Регулярное выражение имело вид $((0+1+b)^*a(0+1+b)^*a)^*(0+1+b)^*a(0+1+b)^*01a$. Обозначим $A=((0+1+b)^*a(0+1+b)^*a)^*$, $B=(0+1+b)^*$. Тогда первое правило будет иметь вид: S→ABaB01a. В правиле для A будут присутствовать B: A→BaBa, надо добавить рекурсию и пустое правило: A→BaBaA|λ. Нетерминал B рекурсивно порождает любые символы кроме 'a': B→0B|1B|bB|λ.

Итак, грамматика построена, выпишем её полностью.
G({0,1,a,b},{S,A,B},P,S), где P:
S→ABaB01a,
A→BaBaA|λ,
B→0B|1B|bB|λ.
Эта грамматика более наглядна, чем регулярная.

Построим распознаватель – детерминированный конечный автомат (ДКА).

При построении ДКА полезно следовать трём правилам:

1) Если распознаваемый язык допускает пустую цепочку, то начальное состояние автомата является также и конечным.
2) Минимальная непустая цепочка языка должна переводить автомат из начального состояния в конечное.
3) Если все цепочки языка должны иметь длину заданной кратности, то в функции переходов количество состояний в цикле равно этой кратности (и не может быть меньшим!!!). Тогда для цепочек произвольной длины на вершинах графа функции переходов допустимы петли, для чётной длины циклы включают по два состояния, и т.д.

Для наглядности построение функции переходов ДКА будем выполнять сначала в виде графа. Построение начинается так же – с минимальной цепочки заданного языка – 'a01a', которая является основой будущего автомата. Каждый символ цепочки читается за один переход из

состояния в состояние. При этом после прочтения минимальной цепочки ДКА должен оказаться в конечном состоянии, которое обозначаем двойным кружком.

Охарактеризуем все построенные состояния с точки зрения основных ограничений заданного языка – количества прочитанных символов ‘a’ и наличия требуемой цепочки. Состояние q_0 будет характеризоваться «чётное количество символов ‘a’, заданной цепочки ещё нет». Состояние q_1 – «нечётное количество символов ‘a’, заданной цепочки ещё нет». Состояние q_2 – «нечётное количество символов ‘a’, прочитан первый символ ‘0’ заданной цепочки». Состояние q_3 – «нечётное количество символов ‘a’, прочитано первые два символа ‘01’ заданной цепочки». Состояние q_4 – «чётное количество символов ‘a’, прочитана вся заданная цепочка».

Теперь для каждого из построенных состояний рассматриваем возможные переходы по всем остальным символам алфавита, учитывая, что любой посторонний символ между символами цепочки ‘01a’ вызовет возвращение в предыдущие состояния.

Поскольку автомат детерминированный, из любого состояния по каждому из символов алфавита возможно делать только по одному переходу. Начинаем с q_0. В нём можно прочитать хоть сколько любых символов, кроме ‘a’; поэтому делаем цикл – рисуем петлю для ‘0,1,b’. В состоянии q_1 – точно так же, но переход по ‘0’ из него уже задан. Поэтому в цикле можно использовать только ‘1,b’. Для того чтобы получить любое количество нулей, зацикливаем ся по нулю в q_2. Получаем:

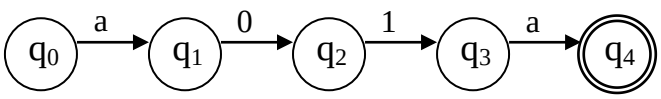

РИС 1

В состоянии q_1 осталось рассмотреть символ ‘a’. Поскольку эти символы нужно подсчитывать, после прочтения ‘a’ (это будет уже второй прочитанный символ ‘a’) необходимо вернуться в состояние их чётного количества – q_0. Из состояния q_2 осталось задать переходы по ‘b’ и по ‘a’. Прочтение ‘b’ нарушает цепочку ‘01a’, но не меняет количества ‘a’, значит, нужно вернуться в q_1, где было столько же ‘a’ и ещё не прочитана цепочка. Прочтение ‘a’ нарушает цепочку и меняет количество ‘a’ – значит, нужно вернуться в q_0.

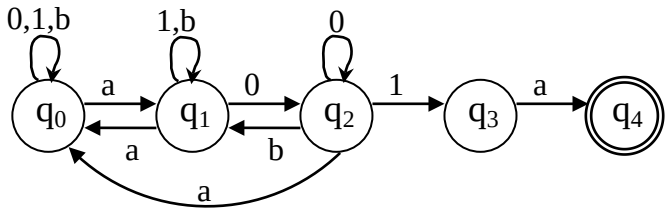

РИС 2

Рассмотрим состояние q_3. В нём ДКА окажется после прочтения первых двух символов ‘01’ нашей цепочки ‘01a’, значит, любой из символов кроме ‘a’ её нарушит. Но если это будет символ ‘0’ – т.е. прочитано ‘a010’, то тем самым уже есть первый символ нашей цепочки, а это состояние q_2. Значит, по ‘0’ надо вернуться в q_2. Символы ‘1’ и ‘b’ равноправны, они требуют заново читать цепочку и не меняют количества ‘a’, поэтому по ним одинаковый переход – в q_1.

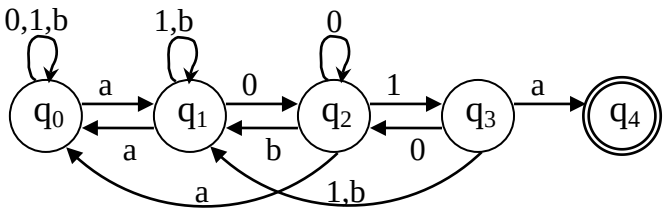

РИС 3

Рассуждая совершенно аналогичным образом, построим все переходы из конечного состояния q_4. Если нужно прочесть ‘a’, то получится цепочка ‘a01aa’ – надо перейти в состояние другой чётности, чем q_4, и при этом снова нужна вся цепочка. Это q_1.

Если нужно прочесть ‘0’, то получится цепочка ‘a01a0’ – есть начало цепочки и, казалось бы, это состояние q_2, но количество ‘a’ не изменилось, а в q_2 их было нечётное количество. Значит, придётся вернуться в состояние той же чётности – q_0. По остальным символам – так же. В итоге получим:

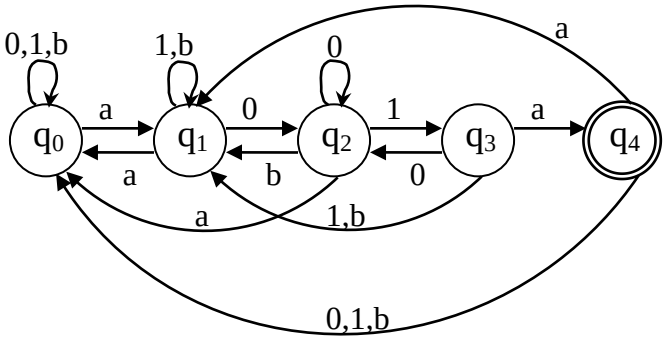

РИС 4

Выпишем полностью построенный ДКА: **M({q0,q1,q2,q3,q4},{0,1,a,b},δ,q0,{q4})**