2024

КЫК и его друзья

17 мая 2024 г.

Содержание

Часть I

КЯК парсер

Рассмотрим пример грамматики ниже. Возьмём для разбора строку *baaba*.

$$S \to AB \mid BC$$

$$A \to BA \mid a$$

$$B \to CC \mid b$$

$$C \to AB \mid a$$

Суть КЯК в рассмотрении подстрок от самых маленьких к целой строке. По итогу нам нужно узнать, является ли строка каким-то из нетерминалов (т.е. словом языка, порождённого грамматикой). Если совсем кратко - принадлежит ли строка языку.

Приступим к разбору. Ещё раз напомню, что мы его ведём снизу вверх - от терминалов к нетерминалам и так пока не охватим всю строку.

$$B \to b$$

$$A \to a$$

$$C \to a$$

Идём дальше. Это были подстроки длины 1. $baaba$ также имеет подстроки длины 2, которые формируются из подстрок длины 1 ($\times$ - декартово произведение):

$$ba = b \times a = B \times A = BA = A$$
$$ba = b \times a = B \times C = BC = S$$

Пока у нас есть $A \to ba, S \to ba$. Которые получаются, как мы выяснили, вот так:

$$A \to BA \to bA \to ba$$
$$S \to BC \to bC \to ba$$

Тупо последовательно заменяем нетерминалы, пока не доберёмся до терминалов. К сожалению, дальше мы упоремся и запутаемся, если будем так продолжать. Это была всего лишь первая 2-подстрока, а я уже устала.
Видимо, кто-то из авторов тоже устал, и придумал лестницу КЯК или как она там. Выглядит она следующим образом (цифры - длины подстрок, НТ - нетерминалы):

5	НТ				
4	НТ	НТ			
3	НТ	НТ	НТ		
2	НТ	НТ	НТ	НТ	
1	НТ	НТ	НТ	НТ	НТ
	b	a	a	b	a

Эта таблица показывает, из каких нетерминалов какую строку мы можем построить. Пока что для нас она выглядит так:

5	НТ				
4	НТ	НТ			
3	НТ	НТ	НТ		
2	НТ	НТ	НТ	НТ	
1	B	A,C	A,C	B	A,C
	b	a	a	b	a

И, учитывая что мы уже успели проанализировать первое ba:

5	НТ				
4	НТ	НТ			
3	НТ	НТ	НТ		
2	S,A	НТ	НТ	НТ	
1	B	A,C	A,C	B	A,C
	b	a	a	b	a

Теперь давайте заполнять эту таблицу. Для следующих подстрок мы получим:

$$aa = a \times a = A, C \times A, C = AA, CA, AC, CC = B$$
$$ab = a \times b = A, C \times B = AB, CB = S, C$$
$$bb = b \times b = B \times B = \varnothing$$

5	HT				
4	HT	HT			
3	HT	HT	HT		
2	S,A	B	S,C	S,A	
1	B	A,C	A,C	B	A,C
	b	a	a	b	a

Строки длины 3.

$$baa = b \times aa = B \times B = BB = \varnothing$$

$$baa = ba \times a = S, A \times A, C = SA, AA, SC, AC = \varnothing$$

$$aab = a \times ab = A, C \times S, C = AS, CS, AC, CC = B$$

$$aab = aa \times b = B \times B = \varnothing$$

$$aba = a \times ba = A, C \times S, A = AS, CS, AA, CA = \varnothing$$

$$aba = ab \times a = S, C \times A, C = SA, CA, SC, CC = B$$

5	HT				
4	HT	HT			
3	$\varnothing$	B	B		
2	S,A	B	S,C	S,A	
1	B	A,C	A,C	B	A,C
	b	a	a	b	a

Строки длины 4.

$$baab = b \times aab = B \times B = BB = \varnothing$$

$$baab = ba \times ab = S, A \times S, C = SS, AS, SC, AC = \varnothing$$

$$baab = baa \times b = \varnothing \times B = \varnothing$$

$$aaba = a \times aba = A, C \times B = AB, CB = S, C$$

$$aaba = aa \times ba = B \times S, A = BS, BA = A$$

$$aaba = aab \times a = B \times A, C = BA, BC = A, S$$

5	HT				
4	$\varnothing$	S,A,C			
3	$\varnothing$	B	B		
2	S,A	B	S,C	S,A	
1	B	A,C	A,C	B	A,C
	b	a	a	b	a

Строки длины 5.

$$baaba = b \times aaba = B \times S, A, C = BS, BA, BC = A, S$$

$$baaba = ba \times aba = S, A \times B = SA, AB = S, C$$

$$baaba = baa \times ba = \varnothing \times S, A = \varnothing$$

$$baaba = baab \times a = \varnothing \times A, C = \varnothing$$

5	S,A,C				
4	∅	S,A,C			
3	∅	B	B		
2	S,A	B	S,C	S,A	
1	B	A,C	A,C	B	A,C
	b	a	a	b	a

Таким образом, строка может быть получена путем последовательного раскрытия (развертки) нетерминалов S, A, C.

УПРАЖНЕНИЕ. Убедиться, что все указанные нетерминалы преобразуются в искомую строку.

Часть II

LL1 парсер

Всё начинается с таблицы парсинга. Для составления таблицы необходимо вначале посчитать множества символов FIRST и FOLLOW для каждого нетерминала NT. Рассмотрим следующую грамматику:

$$A \to CB$$

$$B \to +CB \mid \epsilon$$

$$C \to ED$$

$$D \to *ED \mid \epsilon$$

$$E \to id \mid (A)$$

FIRST(NT) - множество символов на первой позиции строки NT (aka begin()):

$$FIRST(A) = FIRST(C)$$

$$FIRST(B) = \ '+' \cup \epsilon$$

$$FIRST(C) = FIRST(E)$$

$$FIRST(D) = \ '*' \cup \epsilon$$

$$FIRST(E) = \ 'id' \cup \ '('$$

FOLLOW(NT) - множество символов на первой после последней позиции строки NT (aka end()) (т.е.
1) если видим $NT^{''} \to ...NTNT^{'}$, то добавляем $FIRST(NT^{'})$ (если $NT^{'}$ непусто) и $FOLLOW(NT^{''})$
2) если $NT^{'}$ может быть ϵ, то добавляем $FOLLOW(NT^{'})$
3) ϵ не может быть в FOLLOW, если появляется, то его надо убрать)

$$FOLLOW(A) = \ ')' \cup \ '\$'$$

$$FOLLOW(B) = FOLLOW(A) \cup \ '\$'$$
$$FOLLOW(C) = FIRST(B) \cup FOLLOW(A) \cup FOLLOW(B) \ \setminus \epsilon \cup \ '\$'$$
$$FOLLOW(D) = FOLLOW(C) \ \setminus \epsilon \cup \ '\$'$$
$$FOLLOW(E) = FIRST(D) \cup FOLLOW(C) \cup FOLLOW(D) \ \setminus \epsilon \cup \ '\$'$$

Получм:

NT	FIRST	FOLLOW
A	id, (	), \$
B	+, ϵ	), \$
C	id, (	+,), \$
D	*, ϵ	+,), \$
E	id, (	*, +,), \$

Теперь, таблица парсинга. Она содержит производящие правила на пересечении терминалов и нетерминалов. Заполняется она так для каждого нетерминала NT:
1. Если $\epsilon \in FIRST(NT)$, то пишем правило, которым получился ϵ, во всех строках из FOLLOW(NT).
2. Для всех прочих символов из FIRST(NT) в соответствующих строках пишется правило, которым этот символ был получен.

NT/T	id	(	+	)	*	\$
A	$A \to CB$	$A \to CB$				
B			$B \to +CB$	$B \to \epsilon$		$B \to \epsilon$
C	$C \to ED$	$C \to ED$				
D			$D \to \epsilon$	$D \to \epsilon$	$D \to *ED$	$D \to \epsilon$
E	$E \to id$	$E \to (A)$				

Удивительно, но алгоритм предельно прост. Мы всего лишь инициализируем стек стартовым нетерминалом, а в конец рассматриваемой строки добавляем конечный символ \$.

Далее происходит следующее. Мы достаём символ из строки и нечто со стека. Если нечто - терминал и совпало с символом, то убираем и вершину стека, и двигаем текущий символ. Если нет - ошибка.
Если же нечто - нетерминал, то мы ищем в таблице запись на пересечении нечто и текущего символа строки. Запись добавляется в стек, так что первый её элемент наверху. Если запись не найдена, то ошибка. Так повторяется пока текущий символ не станет концом строки или не будет получена ошибка. Если ошибок не было, то строка принадлежит языку.

УПРАЖНЕНИЕ: написать алгоритм самостоятельно. Проверить его для грамматики
$$S' \to S$$

$$S \to xYzS \mid a$$
$$Y \to xYz \mid y$$

и строк $xxyzza$ и $xxyzzz$.