Курс c++

Царьков Олег

Описание тикета OTSN-16

Контекстно-свободные грамматики. Возьмем некую совокупность символов, которую назовем алфавитом и поделим ее на две части – множество “терминальных“ символов и множество “нетерминальных символов“.

Пусть, для определенности, заглавные буквы анлийского алфавита будут нетерминальными, прописные буквы – терминальными.

Правило – это некое преобразование замены, которое можно произвести со строкой. Можно заменять только нетерминальные символы на что-то.

Например, $A \to AxB$ – это правило. По нему можно заменить строку $acbsAADEfG$ на $acbsAxBADEfG$.

Совокупность таких правил называется контекстно-свободной грамматикой.

Пусть S называется стартовым нетерминалом.

Если строку из терминальных символов можно получить с помощью правил грамматики G из строки S, то говорят, что строка принадлежит языку грамматики.

Пример:

Имеется грамматика из таких правил:

$S \to AB$
$A \to AA$
$A \to a$
$B \to b$

Слово $aaaab$ принадлежит языку грамматики, потому что его можно получить так $S \to AB \to AAB \to AAAB \to AAAAB \to aaaab$.

Слово Aab не принадлежит языку грамматики, потому что в нем есть нетерминальный символ A, а мы такие слова вообще не рассматриваем как конечный результат (“терминальные“ символы в переводе означает “завершающие“).

Слово abb не принадлежит грамматике, потому что его нельзя получить никакими заменами согласно правилам грамматики.

Еще один пример:

$S \to aSb$
$S \to <>$

Здесь символом $<>$ обозначена пустая строка, то есть последнее правило говорит о том, что S можно удалять. Языку такой грамматики принадлежат все слова в которых вначале идет сколько-то букв a, потом столько же букв b.

Earley Parser. Парсер – это механизм, способный понять по грамматике, является ли строка принадлежащей к языку грамматики и с помощью каких правил она могла быть получена.

То, с помощью каких правил могла бы быть получена строка, не является однозначным.

Например, в грамматике

$S \to A$
$S \to B$
$A \to c$
$B \to c$

строку "c" можно получить двумя способами.

Самый известный парсер – Earley Parser.

Его логика такова: он читает строку посимвольно и одновременно с этим думает, с помощью каких правил она может быть сгенерирована.

Пусть строка, про которую мы хотим понять, лежит ли она в языке грамматики, это $abcd$.

По ходу дела, парсер не пытается сразу прочитать строку целиком, он пытается прочитать ее хвостики:

0) $abcd$
1) bcd
2) cd
3) d

Элемент состояния парсера – это правило с отмеченной точкой(например, $X \to ATF \bullet SG$) и два числа, первое из которых – номер хвостика, который парсер пытается прочесть, второе — количество прочитанных символов, которому соответствует отмеченная точка.

Разберем на примере работу алгоритма. Пусть грамматика будет

$S \to aSb$
$S \to <>$

а строка, которую мы пытаемся прочесть – ab.

В начале имеется состояния, отвечающие правилам грамматики, то есть
$(S \to \bullet aSb, 0, 0)$
$(S \to \bullet, 0, 0)$,

Здесь оба числа в состояниях – нули, потому что мы пытаемся прочесть нулевой хвостик – то есть всю строку ab и не прочли еще ни одного символа

Мы совершаем шаги нескольких типов:

1) шаг сканирования – когда после отмеченной точки терминальный символ
2) шаг предсказания – когда после отмеченной точки нетерминальный символ
3) шаг завершения – когда хочется объединять два состояния в одно.

Итак, мы видим состояние $(S \to \bullet aSb, 0, 0)$, в нем после отмеченной точки стоит a, и первый символ в строке ab тоже a, поэтому мы совершаем шаг сканирования, получая состояние

$(S \to a \bullet Sb, 0, 1)$

Первое число осталось нулем, потому что запись $a \bullet Sb$ соответствует всей строке ab, а второе число стало равным 1, потому что мы прочли один символ a.

Теперь мы видим, что перед точкой нетерминал, значит надо совершить шаг предсказания. Мы имеем, что 1 символ мы уже прочли, значит нам нужно читать только хвостик b, и нужно предсказывать, куда перейдет при этом S, то есть добавляются такие состояния:

$(S \to \bullet aSb, 1, 1)$
$(S \to \bullet, 1, 1)$

Первое число – единица, потому что эти состояния пытаются прочесть хвостик b, второе число – единица, потому что мы уже прочли символ a, когда говорим об этих состояниях.

Итак, пришло время понять, что это за шаг завершения. Он делает из двух состояний одно. Рассмотрим два состояния $(S \to a \bullet Sb, 0, 1)$ и $(S \to \bullet, 1, 1)$. Первое говорит "ага я читаю строку ab с самого начала и прочло уже 1 символ a, но мне нужно прочитать S и я не знаю как это делается", второе говорит "а я читаю строку начиная с символа b, потому что символ a уже кто-то до меня прочел, и я знаю, что символ S можно отобразить в пустое место". Они могут объединиться и создать такое состояние:

$(S \to aS \bullet b, 0, 1)$

то есть состояние "я прочло только что символ S, и читаю всю строку ab, при этом 1 символ a уже прочло".

Далее, перед отмеченной точкой терминальный символ b и можно сделать состояние

$(S \to aSb\bullet, 0, 2)$

Оно говорит " я читало всю строку ab, прочла два символа из нее, и начинала с S, при этом отмеченная точка у меня в конце, поэтому я применила правило замены до конца", вот это состояние и говорит, что ab принадлежит языку грамматики.

Теперь все тоже самое нормальным языком:

• Для каждого правила в грамматике добавляем состояния с отмеченной точкой в начальной позиции и с двумя нулями.

• Состояния, у которых второе число равно k, называем k-ым слоем состояний

• Обрабатываем k-ый слой:

1) Увидели в k-ом слое состояние $(..., m, k)$, у которого после отмеченной точки терминал, который равен k-ому символу итоговой строки, сделали сканирование, то есть добавили в $k+1$-ый слой состояние $(..., m, k+1)$, в котором отмеченная точка перепрыгнула через терминальную букву.

2) Увидели в k-ом слое состояние $(..., m, k)$, у которого после отмеченной точки нетерминал X. Сделали предсказание, пихнув все правила вида $X \to ...$,с отмеченной точкой в самом начале, в состояния вида $(X \to \bullet ..., k, k)$.

3) Если в слой k на втором шаге добавилось что-то новое, то надо для k-ого слоя попробовать поделать шаг завершения: для каждого состояния в k-ом слое вида $(X \to ..\bullet, m, k)$, у которого отмеченная точка в конце, берем все состояния в слое m вида $(... \bullet X..., n, m)$, которые как раз остановились на моменте m и им нужно прочитать X, и генерируем новое состояние $(...X \bullet ..., n, k)$, добавляем его в k-ый слой

4) Повторяем 1), 2), 3) пока на шаге 2 ничего нового в слой k не добавится, и переходим к слою $k + 1$, снова повторяя все тоже самое.

Заметим, что если мы с каким-то слоем работаем, то мы добавляем состояния либо в него же на шагах 2) и 3), либо в следующий на шаге 1), поэтому предыдущие слои никак не меняются. То есть, как только мы отработали до конца с каким-то слоем, нам не придется в него больше ничего добавлять.

Поэтому мы и делаем для каждого слоя шаги 1), 2) и 3) пока в него не перестанет добавляться что-то новое – тогда его можно оставить в покое и больше ничего не добавлять.

5) Пусть K – длина строки, про которую мы хотели определить, лежит ли она в языке грамматики.

Если да, то должно иметься состояние $(S \to ...\bullet, 0, K)$, то есть достаточно доделать до конца все слои от 1 до K и проверить, если ли в K-ом слое такое состояние.

Если его там нет, то строка не принадлежит языку грамматики.

Чтобы уметь выяснять, какими правилами была получена строка, надо просто вместе с каждым состоянием хранить историю состояний, из которых оно получалось, и при генерации нового состояния дописывать в эту историю шаг, который был сделан.

Задание 666. Реализовать этот алгоритм. Ввод с экрана реализовывать не надо. Должен быть класс “грамматика“, в который можно запихнуть правила грамматики, и класс “парсер“, в который можно запихнуть грамматику и у которого есть функция, принимающая строку и определяющая, лежит ли строка в грамматике. Не лежит – вернуть пустой массив. Лежит – вернуть массив операций, которыми она должна быть получена из правил грамматики.