Лекция №3 Грамматики.

Формальное определение грамматики. Типы грамматик и их свойства.

Для нас наибольший интерес представляет одна из систем генерации языков - грамматики. Понятие грамматики сначала было формализовано лингвистами при изучении естественных языков. Предполагалось, что это может помочь при их автоматической трансляции. Однако, лучшие результаты в этом направлении достигнуто при описании не естественно языков, а языков программирования. примером может служить способ описания синтаксиса языков программирования с помощью БНФ - формы Бекуса-Наура.

Определение. Грамматика - это четверка G = (N, T, P, S), где

N - алфавит нетерминальных символов;

T - алфавит терминальных символов, NT =;

P - конечное множество правил вида, где (NT) * N (NT) * (NT) *;

SN - начальный символ (или аксиома) грамматики.

Мы будем использовать большие латинские буквы для обозначения нетерминальных символов, малые латинские буквы с начала алфавита для обозначения терминальных символов, малые латинские буквы с конца алфавита для обозначения цепочек с T * и, наконец, малые греческие буквы для обозначения цепочек с (NT) *.

Будем использовать также сокращенную запись A 1 | 2 | ... | n для обозначения группы правил A 1, A 2, ..., A n.

Определим на множеств (NT) * бинарное отношение выводимости таким образом: если P, то для всех, (NT) *. Если 1, 2, то говорят, что цепочка 2 непосредственно выведена из 1.

Мы будем использовать также рефлексивно-транзитивное и транзитивное замыкания отношения, а также его степень k 0 (обозначаются соответственно * + и k). Если 1 * 2 (1 + 2, 1 k2), то говорят, что цепочка 2 выведена (нетривиально выведена, выведена за k шагов) с 1.

Если k (k 0), то существует последовательность шагов где = 0 и = k. Последовательность цепочек 0, 1, 2, ..., k в этом случае называют выводом из.

Сентенциальный форме грамматики G называется цепочка, выводится из ее начального символа.

Языке, порождаемой грамматикой G (обозначается L (G)), называется множество всех ее терминальных сентенциальных форм, то есть

Грамматики G1 и G2 называются эквивалентными, если они порождают одну и ту же речь, то есть L (G1) = L (G2).

Пример 2.5. Грамматика G = ({S, B, C}, {a, b, c}, P, S), где

P = {S aSBC, S aBC, CB BC, aB ab, bB bb, bC bc, cC cc} порождает язык L (G) = {anbncn | n> 0}.

Действительно, применяем n-1 раз правило 1 и получаем an-1S (BC) n-1, затем один раз правило 2 и получаем an (BC) n, затем n (n-1) / 2 раз правило 3 и получаем anBnCn.

Затем используем правило 4 и получаем anbBn-1Cn. затем применяем n-1 раз правило 5 и получаем anbnCn. Затем применяем правило 6 и n - 1 раз правило 7 и получаем anbncn. Можно показать, что язык L (G) состоит из цепочек только такого вида.

Пример 2.6. Рассмотрим грамматику G = ({S}, {0, 1}, {S 0S1, S 01}, S). легко видеть, что цепочка 000111 L (G), так как существует заключение

Нетрудно показать, что грамматика порождает язык L (G) = {0n1n | n> 0}.

Пример 2.7. Рассмотрим грамматику G = ({S, A}, {0, 1}, {S 0S, S 0A, A 1A, A 1}, S). Нетрудно показать, что грамматика порождает язык L (G) = {0n1m | n, m> 0}.

Типы грамматик и их свойства

Рассмотрим классификацию грамматик (предложенную Н. Хомского), основанную на виду их правил.

Определение. Пусть дана грамматика G = (N, T, P, S). тогда

если правила грамматики не удовлетворяют никаким ограничениям, то ее называют грамматикой типа 0, или грамматикой без ограничений.

если

каждое правило грамматики, кроме S e, имеет вид, где | | | |, И

в том случае, когда S e P, символ S не встречается в правых частях правил,

то грамматику называют грамматикой типа 1 или несокращения.

если каждое правило грамматики имеет вид A, где AN, (NT) *, то ее называют грамматикой типа 2, или контекстно-свободной (КС-грамматикой).

если каждое правило грамматики имеет вид или A xB или A x, где A, BN, x T

* то ее называют грамматикой типа 3, или праволинийною.

Легко видеть, что грамматика в примере 2.5 - несокращения, в примере 2.6 - контекстно-свободная, в примере 2.7 - праволинийна.

Язык, породжуванаграматикою типа i, называют языком типа i. Язык типа 0 называют также языком без ограничений, речь типа 1 - контекстно-зависимым (КО), речь типа 2 - контекстно-свободным (КС), речь типа 3 - праволинейним.

Теорема 2.1. Каждая контекстно-свободная речь может быть порожден неукорачивающей контекстно-свободной грамматикой.

Доказательство. Пусть L - контекстно-свободная речь. Тогда существует контекстно-свободная грамматика G = (N, T, P, S), порождает L.

Построим новую грамматику G '= (N', T, P ', S') следующим образом:

1. Если в P есть правило вида A 0B11 ... Bkk, где k 0, Bi + e для 1 ik, и ни с одной цепочки j (0 jk) не выводится e, то включить в P 'все правила (кроме A e) вида

где Xi - это либо Bi, или e.

2. Если S + e, то включить в P 'правила S' S, S 'e и положить N' = N {S '}. В противном случае положить N '= N и S' = S.

Порождает грамматика пустую цепочку можноо установить следующим простым алгоритмом:

Шаг 1. Строим множество N_0 = N | N -> e

Шаг 2. Строим множество N_i = N | N -> α • Ni - 1 *

Шаг 3. Если N_i = N_i-1, перейти к шагу 4, иначе шаг 2.

Шаг 4. Если S -> Ni значит S -> e /

Легко видеть, что G '- неукорачивающая грамматика. Можно показать по индукции, L (G ') = L (G). __

Пусть Ki - класс всех языков типа i. Доказано, что справедливо следующее (Строгое) включения:

K3 • K2 • K1 • K0.

Заметим, что если речь порождается некоторой грамматикой, это не значит, что он не может быть порожден грамматикой с более сильными ограничениями на правила. Приведенный ниже пример иллюстрирует этот факт.

Пример 2.8. Рассмотрим грамматику G = ({S, A, B}, {0, 1}, {S $\longrightarrow$ AB, A $\longrightarrow$ 0A, A ~~0, B~~ 1B, ~~B 1}~~, S). . Э~~та г~~рамматика является контекстно-свободной. Легко показать, что L (G) = $\{0^n 1^m | n, m > 0\}$.

Однако, в примере 2.7 приведен праволинейная порождающая грамматика ту же язык.

Покажем что существует алгоритм, позволяющий для произвольного КЗ-языка L в алфавите T, и произвольной цепочки w T * определить, принадлежит ли w языке L.

Теорема 2.2. Каждый контекстно-зависимый язык является рекурсивным языке.

Доказательство. Пусть L - контекстно-зависимый язык. тогда существует некоторая неукорачивающая грамматика G = (N, T, P, S), порождает L.

Пусть w T * и | w | = N. Если n = 0, то есть w = e, то принадлежность w L проверяется тривиальным образом. Так что будем предполагать, что n> 0.

Определим множество Tm как множество строк u (NT) + длины не более n

таких, что вывод S * u имеет не более m шагов. Ясно, что T0 = {S}.

Легко показать, что Tm можно получить из Tm-1 просматривая, какие строки с длиной, меньшей или равной n можно вывести из строк с Tm-1 применением одного правила, то есть

$$T_m = T_{m-1} \cup \{u \mid v \Rightarrow u \text{ для некоторого } v \in T_{m-1}, \text{ где } |u| \leqslant n\}.$$

если S $\Rightarrow^*$ u и | u | n, то u T $_m$ для некоторого m. Если с S не выводится u или | u |> n, то u не принадлежит Tm ни для какого m.

Очевидно, что T $_m$ T $_{m-1}$ для всех m 1. Поскольку Tm зависит только от m 1, если Tm = Tm-1, то Tm = Tm + 1 = Tm 2 = ... процедура будет вычислять T1, T2, T3 ,. . . пока для некоторого m не обнаружится Tm = Tm-1. если w не принадлежит Tm, то не принадлежит и L (G), поскольку для j> m выполнено Tj = Tm. если w T $_m$, то S $\Rightarrow^*$ w.

Покажем, что существует такое m, что Tm = Tm-1. Поскольку для каждого i $\geqslant$ 1 справедливо T $_i$ $\supseteq$ T $_{i-1}$, то если T $_i$ T $_{i-1}$, то число элементов в Ti крайней мере на 1 больше, чем в Ti-1. пусть | NT | = K. Тогда число строк в (NT) + длины меньшей или равной n равно k + k $_2$ + ... + k $_n$ $\leqslant$ nk $_n$. Только эти строки могут быть в любом Ti. Значит, Tm = Tm-1 для некоторого m nkn. Таким образом, процедура, вычисляемый Ti для всех i 1 до тех пор, пока не будут найдены два равных множеств, гарантированно заканчивается, значит, это алгоритм.