Системы программирования.

Часть 2: Элементы теории трансляции

Корухова Юлия Станиславовна
кафедра Алгоритмических языков (ауд. 747)
e-mail: yulia@cs.msu.su

web: ***al.cs.msu.su/node/67***

Литература

1. И.А. Волкова, И.Г. Головин, Л.Е. Карпов «Системы программирования» (2009)

2.И.А. Волкова, А.А. Вылиток, Т.В. Руденко «Формальные грамматики и языки. Элементы теории трансляции.» (3-е издание, 2009)

3. Материалы сайта cmcmsu.no-ip.info (раздел: 2 курс)

Элементы теории трансляции

Транслятор

позволяет преобразовать программу, написанную на ЯП, отличном от машинного языка, к виду, допускающему выполнение на ЭВМ.

Зачем нужно изучение теории трансляции?

Примеры областей, в которых возникает задача перевода одного языка в другой

Пример 1 («литературный»)

А. Конан-Дойль *Возвращение Шерлока Холмса*.
«Пляшущие человечки»

ПРИХОДИ НЕМЕДЛЕННО

Примеры областей,в которых возникает задача перевода одного языка в другой

Пример 2 («музыкальный»)

Перевод аккомпанемента (гитара) в другую тональность

B	***C***
Не сразу все устроилось,	
Dm G7	***Em A7***
Москва не сразу строилась,	
Cm Dm Gm	***Dm Em Am***
Москва слезам не верила,	
Cm F7 B	***Dm G7 C***
А верила любви...	

(Вальс из к/ф «Москва слезам не верит»)

Примеры областей,в которых возникает задача перевода одного языка в другой

Пример 3.
(Программирование)

Написать программу-клиент для интернет магазина.
Программа принимает запросы пользователей, переводит их в язык запросов к БД

Пользователь: Buy 2 cars Audi

Программа клиент: UPDATE Cars SET num = num -2 WHERE name=Audi

Программа сервер: обрабатывает запрос во внутреннем представлении

Пример 4.

Практически любая программа, принимающая на вход 'неоднородные' данные.

Основные понятия

Алфавит

Цепочка (слово)

Конкатенация (сцепление) цепочек

Обращение (реверс) цепочки

n-я степень цепочки

Длина цепочки

Язык в алфавите V

Способы описания языков

Перечисление всех слов

Механизм распознавания

- машина Тьюринга, нормальные алгоритмы Маркова
рекурсивно перечислимые языки
- линейно ограниченный автомат
контекстно-зависимые языки
- автомат с магазинной памятью
контекстно-свободные языки
- конечный автомат
регулярные языки

Механизм генерации

- порождающие грамматики

Порождающая грамматика

это четверка ⟨T, N, P, S⟩, где

• **T – алфавит** терминальных символов ***(*** терминалов ***)*****,**

• **N – алфавит** нетерминальных символов ***(***нетерминалов***), не пересекающийся с T,**

• **P – конечное подмножество множества** $(T \cup N)^{+} \times (T \cup N)^{*}$
элемент (α, β) множества P - это правило вывода

$$\alpha \rightarrow \beta,$$

α содержит хотя бы один нетерминал;

• **S –** начальный символ ***(***цель***)*** **грамматики, S ∈ N**

Примеры грамматик

$G_1 = \langle \{0,1\}, \{A, S\}, P, S \rangle$

P:

$S \to 0A1$

$0A \to 00A1$

$A \to \varepsilon$

G_2 $S \to 0S1 \mid 01$

Пусть $\beta \in (T \cup N)^*$ $\alpha \in (T \cup N)^+$

Цепочка β непосредственно выводима из α в грамматике $G=\{T,N,P,S\}$, если $\alpha = \xi_1 \gamma \xi_2$, $\beta = \xi_1 \delta \xi_2$, где $\xi_1, \xi_2, \delta \in (T \cup N)^*$, $\gamma \in (T \cup N)^+$ и $\gamma \to \delta \in P$.

Цепочка β выводима из цепочки α в грамматике $G = \langle T, N, P, S \rangle$, если существуют цепочки $\gamma_0, \gamma_1, \ldots, \gamma_n$ $(n \geq 0)$, такие, что

$\alpha = \gamma_0 \to \gamma_1 \to \ldots \to \gamma_n = \beta$.

Обозначение: $\alpha \Rightarrow \beta$

Основные определения (прод.)

Сентенциальная форма **в грамматике $G = \langle T, N, P, S \rangle$:**
$\alpha \in (T \cup N)^*$, для которой $S \Rightarrow \alpha$

Язык, порождаемый грамматикой **$G = \langle T, N, P, S \rangle$:**
множество $L(G) = \{\alpha \in T^* \mid S \Rightarrow \alpha\}$

Пример:
$S \rightarrow A \mid bB$
$A \rightarrow a$
$B \rightarrow b \mid \varepsilon$

Вопросы:
Сколько сентенциальных форм в грамматике?

Какой язык порождает грамматика?

Основные определения (прод.)

Эквивалентные грамматики

Почти эквивалентные грамматики

G_1:
$S \rightarrow 0A1$
$0A \rightarrow 00A1$
$A \rightarrow \varepsilon$

G_2 :
$S \rightarrow 0S1 \mid 01$

G_3 :
$S \rightarrow 0S1 \mid \varepsilon$

Классификация грамматик по Хомскому

Классификация проводится по виду правил вывода грамматики.

Тип 0.

Тип 1. Неукорачивающие и контекстно-зависимые грамматики.

Тип 2.
Контекстно-свободные грамматики.

Тип 3.
Регулярные грамматики.

Язык, порождаемый грамматикой типа *k (k=0,1,2,3)*, является языком типа *k*.

Иерархия Хомского

Для грамматик справедливы следующие соотношения:

1) любая регулярная грамматика является КС-грамматикой;
2) любая неукорачивающая КС-грамматика является КЗ-грамматикой;
3) любая неукорачивающая грамматика является грамматикой типа 0.

$$Неукорач.. регулярные \subset Неукорачивающие\ КС \subset КЗ \subset Тип\ 0$$

Для языков справедливы следующие соотношения:

$$Тип\ 3\ (Регулярные) \subset Тип\ 2\ (КС) \subset Тип\ 1\ (КЗ) \subset Тип\ 0$$

Примеры грамматик и языков

G_1:

$S \to a \mid Sa$

G_2:

$S \to aSb \mid \varepsilon$

G_3:

$S \to aSBC \mid abC$

$CB \to BC$

$bB \to bb$

$bC \to bc$

$cC \to cc$

Задача распознавания

Даны грамматика G и цепочка x. $x \in L(G)$?

Для грамматик типа 1, 2 и 3задача распознавания разрешима, т.е. существует алгоритм, отвечающий на вопрос: $x \in L(G)$?

Разбор цепочек по КС-грамматикам

Грамматика	**цепочка**
S → T | T+S	**a+b+a**
T → a | b	

Левый *(*левосторонний*)* **вывод**

S → T+S → a+S → a+T+S → a+b+S → a+b+T → a+b+a

Правый *(*правосторонний*)* **вывод**

S→T+S → T+T+S → T+T+T → T+T+a → a+T+a → a+b+a

→ T+ b+a → a+b+a

Дерево вывода (дерево разбора)

Упорядоченное ориентированное дерево называется деревом вывода (или деревом разбора) в КС-грамматике G = (T, N, P, S), если выполнены следующие условия:

(1) каждая вершина дерева помечена символом из $N \cup T \cup \{\varepsilon\}$, при этом корень дерева помечен символом S;
листья -символами из $T \cup \{\varepsilon\}$;

(2) если вершина дерева помечена символом A , а ее непосредственные потомки - символами $a_1, a_2, \ldots, a_n$, где каждое $a_i \in T \cup N$,
то $A \to a_1 a_2 \ldots a_n$ - правило вывода в этой грамматике;

(3) если вершина дерева помечена символом A , а ее единственный непосредственный потомок помечен символом ε,
то $A \to \varepsilon$ правило вывода в этой грамматике.

Неоднозначные и однозначные грамматики

Неоднозначная грамматика

$E \to E * E \mid E + E \mid T$
$T \to a$

Однозначная грамматика

$E \to E + F \mid F$
$F \to F * T \mid T$
$T \to a$

Проблема определения, является ли заданная КС-грамматика однозначной, является алгоритмически неразрешимой.

Проблема определения, порождает ли данная КС-грамматика однозначный язык (т.е. существует ли эквивалентная ей однозначная грамматика), является алгоритмически неразрешимой

Преобразования грамматик

Бесполезные символы — бесплодные, недостижимые

КС-грамматика называется приведенной, если в ней нет недостижимых и бесплодных символов.

Пример:
исходная грамматика

S → ABc | Ff
F → Ff | a
A → a
B → aB | dD
D→ ddD | aBK
K → c

приведенная грамматика

S → Ff
F → Ff | a

Преобразования грамматик

Устранение правил с пустой правой частью из КС-грамматики

1. Построить множество $X=\{A \in N \mid A \Rightarrow \varepsilon\}$.

2. Удалить правила с пустой правой частью.

3. Если $S \in X$, то S' – новый начальный символ, $S' \to S \mid \varepsilon \in P$.

4. $\forall A \in X$ правило вида $B \to \alpha_1 A_1 \alpha_2 A_2 \ldots \alpha_n A_n \alpha_{n+1}$, где $\alpha_i \in ((N - X) \cup T)^*$ заменить $2n$ правилами, соответствующими всем возможным комбинациям вхождений A между α_i:

$B \to \alpha_1 \alpha_2 \ldots \alpha_n \alpha_{n+1}$

$B \to \alpha_1 \alpha_2 \ldots \alpha_n A_n \alpha_{n+1}$

...

$B \to \alpha_1 \alpha_2 A_2 \ldots \alpha_n A_n \alpha_{n+1}$

$B \to \alpha_1 A_1 \alpha_2 A_2 \ldots \alpha_n A_n \alpha_{n+1}$

***Замечание*: если получено правило $B \to \varepsilon$, не включать его в новую грамматику.**

5. Удалить бесполезные символы и правила, их содержащие.

Преобразования грамматик

Устранение правил с пустой правой частью из КС-грамматики

Пример.

	S→BC | Ab		**S→C | b | Ab**
исходная	**B→ε**	**эквивалентная**	**C→c**
грамматика	**C→c**	**грамматика**	**A→Aa | a**
	A→Aa | ε		