

Теория автоматов и формальных языков

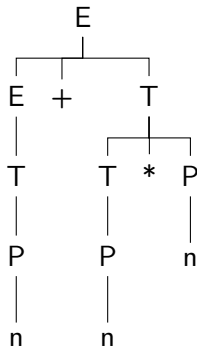
Синтаксически управляемая трансляция

Автор: Григорьев Семён

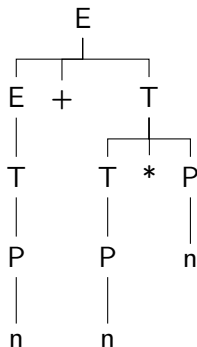
Санкт-Петербургский государственный университет

22 мая 2020

Дерево разбора — лишь цепочка в некотором языке



Дерево разбора — лишь цепочка в некотором языке



$[.E[.E[.T[.P[.n]]]]][.+][.T[.T[.P[.n]]][.*][.P[.n]]]$

Трансляция (перевод)

- **Трансляция** — преобразование некоторой входной строки в некоторую выходную
 - ▶ Σ — входной алфавит, Π — выходной алфавит.
Трансляцией с языка $L_i \subseteq \Sigma^*$ на язык $L_o \subseteq \Pi^*$ называется отображение $\tau : L_i \rightarrow L_o$
- Построение дерева разбора — простейший пример трансляции
- Другие примеры трансляции
 - ▶ Вычисление значения арифметического выражения
 - ▶ Преобразование арифметического выражения из инфиксной записи в постфиксную
 - ▶ Преобразование программы на языке Java в байт-код
 - ▶ Компиляция программ
 - ▶ ...
- Фактически синтаксический анализ нужен для трансляции

Схемы синтаксически управляемой трансляции

Схема синтаксически управляемой трансляции — пятерка
 (N, Σ, Π, P, S)

- N — конечное множество нетерминальных символов
- Σ — конечный входной алфавит
- Π — конечный выходной алфавит
- $S \in N$ — стартовый нетерминал
- P — конечное множество правил трансляции вида $A \rightarrow \alpha, \beta$, где $\alpha \in (N \cup \Sigma)^*, \beta \in (N \cup \Pi)^*$
 - ▶ Вхождения нетерминалов в цепочку β образуют перестановку нетерминалов из цепочки α
 - ▶ Если нетерминалы повторяются больше одного раза, то их различают по индексам: $E \rightarrow E^l + E^r, E^r + E^l$

Выводимая пара в СУ-схеме

- Если $A \rightarrow (\alpha, \beta) \in P$, то $(\gamma A^i \delta, \gamma' A^i \delta') \Rightarrow (\gamma \alpha \delta, \gamma' \beta \delta')$
- Рефлексивно-транзитивное замыкание отношения \Rightarrow называется отношением выводимости в СУ-схеме, обозначается \Rightarrow^*
- Трансляцией назовем множество пар $\{(\alpha, \beta) \mid (S, S) \xRightarrow{*} (\alpha, \beta), \alpha \in \Sigma^*, \beta \in \Pi^*\}$
- СУ-схема называется простой, если во всех правилах $A \rightarrow (\alpha, \beta)$, нетерминалы в α и β встречаются в одном и том же порядке

Определение

СУ-схема называется простой, если во всех правилах $A \rightarrow (\alpha, \beta)$, нетерминалы в α и β встречаются в одном и том же порядке

Однозначная СУ-схема

Определение

Однозначная СУ-схема — СУ-схема, в которой не существует двух правил $A \rightarrow \alpha, \beta$ и $A \rightarrow \alpha, \gamma$ таких, что $\beta \neq \gamma$

Пример СУ-схемы

$$\begin{array}{lcl} E & \rightarrow & E + T \quad , \quad ET + \\ & | & T \quad , \quad T \\ T & \rightarrow & T * F \quad , \quad TF * \\ & | & F \quad , \quad F \\ F & \rightarrow & n \quad , \quad n \\ & | & (E) \quad , \quad E \end{array}$$

Пример СУ-схемы

$$\begin{array}{lcl} E & \rightarrow & E + T \quad , \quad ET + \\ & | & T \quad , \quad T \\ T & \rightarrow & T * F \quad , \quad TF * \\ & | & F \quad , \quad F \\ F & \rightarrow & n \quad , \quad n \\ & | & (E) \quad , \quad E \end{array}$$

$$\begin{aligned} (\underline{E}, \underline{E}) &\Rightarrow (\underline{T}, \underline{T}) \Rightarrow (\underline{T} * F, \underline{T} F *) \Rightarrow (\underline{F} * F, \underline{F} F *) \Rightarrow (n * \underline{F}, n \underline{F} *) \Rightarrow \\ (n * (\underline{E}), n \underline{E} *) &\Rightarrow (n * (\underline{E} + T), n \underline{E} T + *) \Rightarrow (n * (\underline{T} + T), n \underline{T} T + *) \Rightarrow \\ (n * (\underline{F} + T), n \underline{F} T + *) &\Rightarrow (n * (n + \underline{T}), n n \underline{T} + *) \Rightarrow \\ (n * (n + \underline{F}), n n \underline{F} + *) &\Rightarrow (n * (n + n), n n n + *) \end{aligned}$$

Обобщенные схемы синтаксически управляемой трансляции

Обобщенная схема синтаксически управляемой трансляции — шестерка $(N, \Sigma, \Pi, \Gamma, P, S)$

- Γ — конечное множество символов перевода вида $A_i, A \in N; i \in \mathbb{Z}$
- P — конечное множество правил трансляции вида $A \rightarrow \alpha, A_1 = \beta_1, \dots, A_n = \beta_n$, где $\alpha \in (N \cup \Sigma)^*$
 - ▶ $A_i \in \Gamma, 1 \leq i \leq n$
 - ▶ Каждый символ x , входящий в β_i , либо $x \in \Pi$, либо $x = B_k \in \Gamma$, где $B \in \alpha$
 - ▶ Если α имеет более одного вхождения символа B , то каждый символ B_k во всех β соотнесен (верхним индексом) с конкретным вхождением B

Входной грамматикой назовем четверку (N, Σ, P', S) , где $P' = \{A \rightarrow \alpha \mid A \rightarrow \alpha, A_1 = \beta_1, \dots, A_n = \beta_n \in P\}$

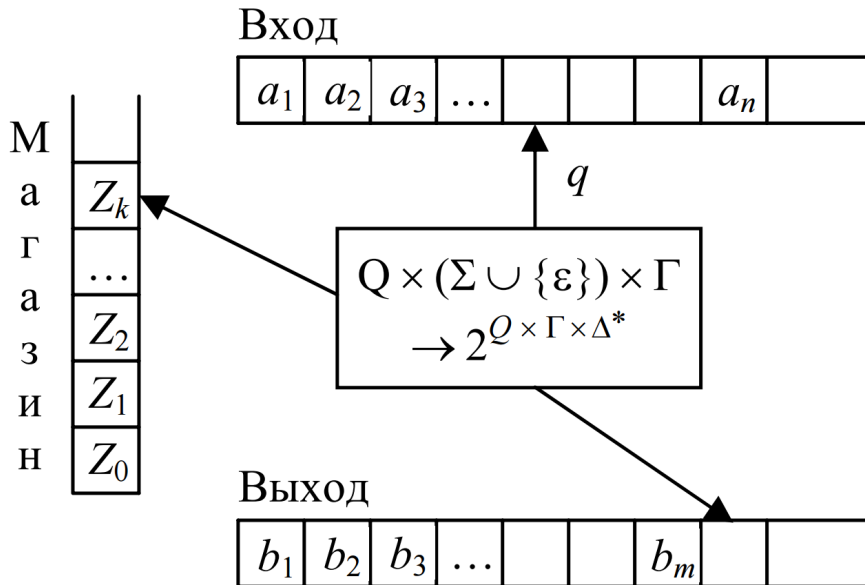
Выход обобщенной СУ-схемы

- Для каждой внутренней вершины дерева, соответствующей нетерминалу A , с каждым A_i связывается одна цепочка
 - ▶ Такую цепочку назовем значением (трансляцией) символа A_i
- Каждое значение определяется подстановкой значений символов трансляции данного элемента $A_i = \beta_i$, определенных в прямых потомках вершины
- **Трансляция**, определяемая данной схемой — множество $\{(\alpha, \beta)\}$
 - ▶ α имеет дерево разбора в данной входной грамматике
 - ▶ β — значение выделенного символа S_k

Пример обобщенной СУ-схемы: дифференцирование

$$\begin{array}{llll} E & \rightarrow & E + T & , \quad E_1 = E_1 + T_1 \\ & & & , \quad E_2 = E_2 + T_2 \\ & | & T & , \quad E_1 = T_1, E_2 = T_2 \\ T & \rightarrow & T * F & , \quad T_1 = T_1 * F_1 \\ & & & , \quad T_2 = T_1 * F_2 + T_2 * F_1 \\ & | & F & , \quad T_1 = F_1, T_2 = F_2 \\ F & \rightarrow & (E) & , \quad F_1 = (E_1) \\ & & & , \quad F_2 = (E_2) \\ & | & \sin(E) & , \quad F_1 = \sin(E_1) \\ & & & , \quad F_2 = \cos(E_1) * E_2 \\ & | & \cos(E) & , \quad F_1 = \cos(E_1) \\ & & & , \quad F_2 = -\sin(E_1) * E_2 \\ & | & x & , \quad F_1 = x, F_2 = 1 \\ & | & n & , \quad F_1 = n, F_2 = 0 \end{array}$$

Магазинный преобразователь



Что такое магазинный преобразователь: неформально

Магазинный автомат, который при каждом переходе пишет что-то в выходную строку

Формальное определение

Магазинный преобразователь это набор $(Q, \Sigma, \Gamma, \Delta, \delta, q_0, Z_0, F)$

- Q — конечное множество состояний
- Σ — конечное множество символов, входной алфавит
- Γ — конечное множество символов, стековый алфавит
- Δ — конечное множество символов, выходной алфавит
- $\delta \subseteq Q \times (Z \cup \varepsilon) \times \Gamma \rightarrow 2^{Q \times \Gamma^* \times \Delta^*}$ — отношение переходов
- $q_0 \in Q$ — стартовое состояние
- $Z_0 \in \Gamma$ — начальный элемент стека
- $F \subseteq Q$ — множество принимающих (конечных) состояний

$\delta(p, a, Z) = \{(q_i, \gamma_i, \alpha_i) \mid 1 \leq i \leq n\}$ означает

- Если магазинный преобразователь находится в состоянии $p \in Q$, на вершине стека находится $Z \in \Gamma$, а со входа читается символ $a \in \Sigma \cup \varepsilon$, то для некоторого i :
 - ▶ Изменяем состояние на $q_i \in Q$
 - ▶ Снимаем со стека символ Z , записываем на стек строку $\gamma_i \in \Gamma^*$
 - ▶ В выходную строку дописываем $\alpha_i \in \Delta^*$
- $\Sigma \cup \varepsilon$ сигнализирует о том, что вход можно и не читать
- Если $\gamma_i = \varepsilon$, символ со стека стирается
- Если $\alpha_i = \varepsilon$, в выходную строку ничего не пишем

- Мгновенное описание МП: $(p, \omega, \beta, \alpha) \in Q \times \Sigma^* \times \Gamma^* \times \Delta^*$
 - ▶ p — текущее состояние автомата
 - ▶ ω — непрочитанный фрагмент входного потока
 - ▶ β — содержимое стека (верхушка записана первой)
 - ▶ α — содержимое выходной ленты
- Отношение \vdash на мгновенных описаниях (шаг)
 - ▶ Для каждого $(q, \gamma, \alpha) \in \delta(p, a, Z)$, верно $(p, ax, Z\eta, \zeta) \vdash (q, x, \gamma\eta, \alpha\zeta)$ для произвольных $x \in \Sigma^*, \eta \in \Gamma^*, \zeta \in \Delta^*$
- Шаг не определен, если стек пуст

Семантика магазинного преобразователя: вычисление

- Вычисление — последовательность шагов
 - ▶ \vdash^* — транзитивно рефлексивное замыкание отношения \vdash
- Начальное мгновенное описание $(q_0, \omega, Z_0, \varepsilon)$
- Два варианта окончания работы
 - ▶ По достижении конечного состояния
 - ★ $\tau(M) = \{(\omega, \alpha) \mid \omega \in \Sigma^*, \alpha \in \Delta^*, (q_0, \omega, Z_0, \varepsilon) \vdash^* (f, \varepsilon, \gamma, \alpha), f \in F, \gamma \in \Gamma^*\}$
 - ▶ По опустошении стека
 - ★ $\tau_\varepsilon(M) = \{(\omega, \alpha) \mid \omega \in \Sigma^*, \alpha \in \Delta^*, (q_0, \omega, Z_0, \varepsilon) \vdash^* (q, \varepsilon, \varepsilon, \alpha), q \in Q\}$
 - ▶ Эти варианты эквивалентны: по преобразователю, завершающемуся по первой схеме, можно посмотреть преобразователь, завершающийся по второй схеме, и наоборот

Детерминированные магазинные преобразователи

Магазинный преобразователь является **детерминированным**, если

- $\forall q \in Q, a \in \Sigma \cup \{\epsilon\}, Z \in \Gamma : |\delta(q, a, Z)| \leq 1$
- Если $\delta(q, \epsilon, Z) \neq \emptyset$, то $\forall a \in \Sigma : \delta(q, a, Z) = \emptyset$
- Детерминированный магазинный преобразователь является частным случаем недетерминированного

Пример: преобразование префиксных арифметических выражений в постфиксные

$$M = \{\{q\}, \{a, +, *\}, \{E, +, *\}, \{a, +, *\}, \delta, q, E, \{q\}\}$$

$$\delta(q, a, E) = \{(q, \varepsilon, a)\}$$

$$\delta(q, +, E) = \{(q, EE+, \varepsilon)\}$$

$$\delta(q, *, E) = \{(q, EE*, \varepsilon)\}$$

$$\delta(q, \varepsilon, +) = \{(q, \varepsilon, +)\}$$

$$\delta(q, \varepsilon, *) = \{(q, \varepsilon, *)\}$$

$$\begin{aligned} &(q, + * aaa, E, \varepsilon) \vdash (q, *aaa, EE+, \varepsilon) \vdash (q, aaa, EE * E+, \varepsilon) \vdash \\ &(q, aa, E * E+, a) \vdash (q, a, *E+, aa) \vdash (q, a, E+, aa*) \vdash (q, \varepsilon, +, aa * a) \vdash \\ &(q, \varepsilon, \varepsilon, aa * a+) \end{aligned}$$

Взаимоотношение между простыми СУ-схемами и магазинными преобразователями

Теорема

По простой СУ-схеме $(N, \Sigma, \Delta, R, S)$ можно построить магазинный преобразователь, задающий эквивалентную трансляцию

Теорема

По магазинному преобразователю $P = (Q, \Sigma, \Gamma, \Delta, \delta, q_0, Z_0, \emptyset)$ можно построить простую СУ-схему, задающую эквивалентную трансляцию

Теорема

Класс трансляций, задаваемых простыми СУ-трансляциями совпадает с классом трансляций, задаваемых магазинными автоматами

Однозначные СУ-схемы и левосторонний вывод

Теорема

Выходная цепочка однозначной СУ-схемы может быть сгенерирована при левостороннем выводе входной цепочки

Транслирующие грамматики

- КС-грамматика, терминальный алфавит которой разбит на два множества: входных и выходных символов
- **Транслирующая грамматика** — пятерка $(N, \Sigma_i, \Sigma_o, P, S)$
 - ▶ N — алфавит нетерминалов
 - ▶ Σ_i — алфавит входных терминалов
 - ▶ Σ_o — алфавит выходных терминалов
 - ▶ $S \in N$ — стартовый нетерминал
 - ▶ $P = \{A \rightarrow \alpha\}, \alpha \in (\Sigma_i \cup \Sigma_o \cup N)^*$ — множество правил вывода

Пример транслирующей грамматики

$$\begin{array}{lcl} E & \rightarrow & E + T \{+\} \\ & | & T \\ T & \rightarrow & T * F \{*\} \\ & | & F \\ F & \rightarrow & n \{n\} \\ & | & (E) \end{array}$$

Пример транслирующей грамматики

$$\begin{array}{lcl} E & \rightarrow & E + T \{+\} \\ & | & T \\ T & \rightarrow & T * F \{*\} \\ & | & F \\ F & \rightarrow & n \{n\} \\ & | & (E) \end{array}$$

$$\begin{aligned} E &\Rightarrow E + T \{+\} \Rightarrow T + T \{+\} \Rightarrow F + T \{+\} \Rightarrow n \{n\} + T \{+\} \Rightarrow \\ n \{n\} + T * F \{*\} \{+\} &\Rightarrow n \{n\} + F * F \{*\} \{+\} \Rightarrow \\ n \{n\} + n \{n\} * F \{*\} \{+\} &\Rightarrow n \{n\} + n \{n\} * n \{n\} \{*\} \{+\} \end{aligned}$$

Пример транслирующей грамматики

$$\begin{array}{lcl} E & \rightarrow & E + T \{+\} \\ & | & T \\ T & \rightarrow & T * F \{*\} \\ & | & F \\ F & \rightarrow & n \{n\} \\ & | & (E) \end{array}$$

$$\begin{aligned} E &\Rightarrow E + T \{+\} \Rightarrow T + T \{+\} \Rightarrow F + T \{+\} \Rightarrow n \{n\} + T \{+\} \Rightarrow \\ n \{n\} + T * F \{*\} \{+\} &\Rightarrow n \{n\} + F * F \{*\} \{+\} \Rightarrow \\ n \{n\} + n \{n\} * F \{*\} \{+\} &\Rightarrow n \{n\} + n \{n\} * n \{n\} \{*\} \{+\} \end{aligned}$$

- Если вычеркнуть все выходные символы, получим $n + n * n$
- Если вычеркнуть все входные символы, получим $nnn + *$ — постфиксная запись выражения

Постфиксная транслирующая грамматика

- Если выходные символы встречаются только в конце правил, транслирующая грамматика называется постфиксной
- Это требование формально не выдвигается: транслирующие грамматики могут быть не постфиксными
- На практике постфиксные транслирующие грамматики удобнее

Атрибутная транслирующая грамматика

- Входной алфавит — алфавит лексем
 - ▶ Лексема характеризуется **типом** и **значением**
- Транслирующая грамматика описывает перевод только **типа** лексемы
 - ▶ Это существенно снижает выразительность формализма
- Для борьбы с этим недостатком предложены атрибутные грамматики
 - ▶ Модификация транслирующих грамматик, снабженная атрибутами

Атрибут — дополнительные данные, ассоциированные с грамматическими символами

- Если X — символ, а a — его атрибут, то значение a в узле дерева, помеченном X , записывается как $X.a$
- Узлы дерева могут реализовываться как записи или объекты, а атрибуты — как поля
- Атрибуты могут быть любого типа
- Если в каждом узле дерева атрибуты уже вычислены, оно называется **аннотированным**
- Процесс вычисления этих атрибутов называется **аннотированием** дерева разбора

Вычисление атрибутов не всегда возможно

$$\begin{array}{ll} A \rightarrow B & A.s = B.i \\ & B.i = A.s + 1 \end{array}$$

Синтезируемый атрибут, S-атрибутная грамматика

- Атрибут, значение которого зависит от значений атрибутов детей данного узла или от других атрибутов этого узла, называется **синтезируемым**
- Если в транслирующей грамматике используются только синтезируемые атрибуты, она называется **S-атрибутной грамматикой**
- Аннотирование дерева разбора S-атрибутной грамматики возможно путем выполнения семантических правил снизу вверх (от листьев к корню)

Пример S-атрибутной грамматики

$$S \rightarrow E \quad \{S.val = E.val\}$$

$$E^0 \rightarrow E^1 + T \quad \{E^0.val = E^1.val + T.val\}$$

$$E \rightarrow T \quad \{E.val = T.val\}$$

$$T^0 \rightarrow T^1 * F \quad \{T^0.val = T^1.val * F.val\}$$

$$T \rightarrow F \quad \{T.val = F.val\}$$

$$F \rightarrow n \quad \{F.val = n.val\}$$

$$F \rightarrow (E) \quad \{F.val = E.val\}$$

Наследуемый атрибут, L-атрибутная грамматика

Атрибут, значение которого зависит только от атрибутов братьев узла слева или атрибутов родителей, называется **наследуемым**

Грамматика называется **L-атрибутной**, если каждый наследуемый атрибут узла X_j в правиле $A \rightarrow X_1 \dots X_n$ зависит только от:

- Атрибутов узлов $X_1 \dots X_{j-1}$ (братья слева)
- Наследуемых атрибутов узла (предок)

Синтезируемые атрибуты тоже разрешены

Любая S-атрибутная грамматика является L-атрибутной

Пример L-атрибутной грамматики

$$D \rightarrow TL \quad \{L.inh = T.type; \\ D.val = L.val\}$$
$$T \rightarrow int \quad \{T.type = integer\}$$
$$T \rightarrow real \quad \{T.type = real\}$$
$$L^0 \rightarrow L^1, id \quad \{L^1.inh = L^0.inh \\ \{L^0.val = (id.text, L_0.inh) :: L^1.val\}$$
$$L \rightarrow id \quad \{[id.text, L.inh]\}$$