MTA Számitástechnikai és Automatizálási Kutató Intézete, **Közlemények** 21/1978.

Анализатор для грамматики ван Вейнгаардена

Геревич Л.

1

1. Определение W-грамматики /грамматика ван Вейнгаардена/

Рассмотрим множества, образующие матаситемы W-грамматики:

M - конечное множество метапонятий;

τ_0 - конечное множество метатерминальных символов;

$\Pi \subset M \times (M \cup \tau_0)^*$ - конечное множество контекстно-свободных метаправил и

$\mathfrak{L}_x = \{w \in \tau_0^* : x \underset{\Pi}{\Rightarrow} M\}$ - множество представляющее собой цепочки метатерминальных символов, порождаемых метапонятием x.

Рассотрим множества языка:

$\tau \supset \tau_0$ - конечное множество, содержащее и метатерминальные символы;

Σ - конечное множество терминальных символов языка;

$\bar{\Phi} \subset \{\langle \bar{w} \rangle : \bar{w} \in (M \cup \tau)^+\}$ - конечное множество гиперпонятий;

$\bar{\Gamma} \subset \bar{\Phi} \times (\bar{\Phi} \cap \Sigma)^*$ - конечное множество гиперправил, служащих формами языка;

$\Phi \subset \{\langle w \rangle : w \in \tau^+\}$ - множество нетерминалов языка, $M \cap \tau = \emptyset$;

$\Gamma \subset \Phi \times (\Phi \cup \Sigma)^*$ - множество правил языка, полученное из $\bar{\Gamma}$ путем применения совместимой замены

$$\Psi : (M \cup \tau)^+ \to \tau^+$$

которая является однозначной функцией /полугруппой и гомоморфизмом/ со следующими свойствами:

$$\Psi(x) \in \mathfrak{L}_x \quad (x \in M)$$

$$\Psi(w) = w \quad (w \in \tau^+)$$

Из гиперправила $\langle \bar{w} \rangle \to \sigma_1 \langle \bar{w}_1 \rangle \sigma_2 \ldots \sigma_{n-1} \langle \bar{w}_{n-1} \rangle \sigma_n \in \Gamma$ $(\langle \bar{w}, \langle w_i \rangle \in \Phi, \delta_i \in \Sigma^* \rangle$

A^+ - множество всех цепочек над алфавитом A и $A^* = A^+ 0 \{\lambda\}$

получается контекстно-свободное правило:

$$\Psi(\bar{w}) \to \sigma_1 \Psi(\bar{w}_1)\sigma_2 \ldots \sigma_{n-1}\Psi(\bar{w}_{n-1})\ \sigma_n \in \Gamma$$

<S> является начальным символом или целью. Языком, порожденным с помощью W-грамматики, назовем следующее множество:

$$L = \{\sigma \in \Sigma^* : S \underset{\Gamma}{\Rightarrow} \sigma\} .$$

Пример 1. Рассмотрим язык $L_1 = \{x^k y^k x^k : k \geq 1\}$. W-грамматика W_1 , которая описывает язык, имеет следующий вид:

метаправила: $A \to nA$, $A \to \lambda$ (λ – пусто);

гиперправила: <S>→<aA><bA><aA>; <anA>→x<aA>; <bnA>→y<bA>;
<a>→λ; <b>→λ ,

то есть $W_1 = (M, \tau_0, \Pi, \Sigma, \bar{\Phi}, \bar{\Gamma}, S)$; $M = \{A\}$; $\tau_0 = \{n, \lambda\}$;

Π определено выше, $\mathfrak{L}_A = \{n^*\}$; $\tau = \tau_0 \cup \{<S>\}$; $\Sigma = \{x, y\}$;

<S>- цель языка, $\bar{\Gamma}$ задано выше, $\bar{\Phi}$ ={<S><aA><bA><anA><bnA><a><b>}.

Слова языка L порождаются с помощью грамматики W_1 с применением следующих шагов.

Пусть nn – терминальное порождение метапонятия A , тогда применяя это порождение к первому гиперправилу, получим правило языка следующего вида:

<S> → <ann><bnn><ann>

Исходя из цели языка /из S/ и применяя полученное правило, получим строку вывода:

<ann><bnn><ann>

Для нетерминала <ann> можно получить правило из гиперправила <anA> → x<aA> заменой A на его терминальное порождение n , то есть можно получить правило <ann> → x<an> строгого языка. Применяя это правило к нетерминалу <ann> строки вывода в его обоих вхождениях, получим строку вывода:

x<an><bnn>x<an>

Исходя из того же гиперправила и заменяя A на λ , получим другое правило строгого языка:

<an> → x<a>

Применяя его к строке вывода, получаем:

xx<a><bnn>xx<a> .

В случае нетерминала <bnn> исходим из гиперправила <bnA>→y<bA>.

Аналогичным образом, заменяя сначала A на n , а затем A на λ , в результате последовательного применения этих двух правил получим строку вывода:

xx<a>yy<b>xx<a> .

Применяя последние два гиперправила - правила <a> → λ и <b> → λ –, наконец, получим:

xxyyxx .

2. Прежде чем перейти к описанию анализатора для W-грамматики, остановимся на рассмотрении таких грамматик. Попытаемся применять гиперправила таким же образом, как применялись контекстно-свободные правила, проследим, как изменяются значения метапонятий.

Определим совместимую замену по-другому и в дальнейшем всегда будем ссылаться на это определение. Будем говорить, что цепочка β совместимо выведена из цепочки £ , если цепочка β была порождена из £ таким образом, что все вхожде-

ния одного и того же метапонятия в £ были заменены на одно и то же порождение этого метапонятия, не обязательно терминальное, т.е. возможно содержащее метапонятия, которые отличаются от метапонятия цепочки β. /Рассмотрим, например, α = aAEA, метаправило: A → eE. С помощью совместимого вывода получим цепочку $\beta = aeE_1EeE_1$. Метапонятие E_1 порождается как E, только его порождение не обязательно должно совпадать с порождением метапонятия E при совместимом выводе, то есть оно не подчиняется законам совместимости с метапонятием E./

Назовем гиперправило подходящим для некоторой цепочки δ, если из некоторого гиперпонятия α цепочки δ и из гиперпонятия, стоящего в левой части гиперправила, совместимо выводятся одинаковые гиперпонятия. Эти одинаковые гиперпонятия будем называть общими. /Пусть α = <aA> и δ = <aA><bA><aA>, метоправила для A : A → nA, а гиперправило <anA> → x<aA> является подходящим, поскольку <anA> совместимо выводимо из <aA> заменой A на nA./

Для порождения терминальной цепочки языка, который описывается W-грамматикой, будем исходить из цели грамматики. Применим к нему гиперправило, в результате получим цепочку, состоящую из гиперпонятий. Рассмотрим, как применяются гиперправила к цепочкам, состоящим из гиперпонятий. Сначала выберем некоторое гиперпонятие, а затем найдем гиперправило, подходящее к нему. После того, как найдено подходящее гиперправило, т.е. такое, что из выбранного гиперпонятия и из гиперпонятия в левой части гиперправила, подходящего для него, порождаются общие гиперпонятия, выберем одно из общих гиперпонятий, и из цепочки совместимо-порождаем такую цепочку, в которой из выбранного гиперпонятия исходной цепочки получилось общее гиперпонятие. Для этого выбранного гиперпонятия цепочки применим гиперправило, совместимо порожденное из исходного гиперправила таким образом, что левая часть есть то же самое общее гиперпонятие.

/Например, гиперправило имеет вид: <aAE> → <bA><dE> , а метаправила: A → eE и B → aeE . Рассмотрим цепочку <BE><gBC><bE>. Из гиперпонятия <BE> и из гиперпонятия <aAE> совместимо выводится гиперпонятие $<aeE_1E>$. Совместимо порожденные цепочка и гиперправило имеет вид:

$$<aeE_1E><gaeE_1c><aeE_1bE> \text{ и } <aeE_1E> \rightarrow <beE_1><dE>$$

Применим это новое гиперправило к новой цепочке, получим цепочку:

$$<beE_1><dE><gaeE_1C><bE> \text{ . /}$$

Повторим эту процедуру. Будем применять к полученной цепочке гиперправила описанным выше образом до тех пор, пока не получим терминальную строку языка, описываемого с помощью W-грамматики.

При таком порождении может случиться, что получится такое гиперпонятие, для которого нет подходящего гиперправила. Такие случаи назовем тупиками.

Сущность такого порождения состоит в том, что мы пытаемся ссылаться в цепочках вывода на всевозможные выводы W-грамматики /вхождения метапонятия в цепочку вывода/. Поскольку общих гиперпонятий может быть больше, чем одно /их может быть бесконечно много/, а в таких выводах мы пользуемся только одним из них, поэтому мы можем проследить только за некоторым подмножеством множества всевозможных выводов /подмножество совпадает с множеством, если общее гиперпонятие всегда только одно/.

Пример 2. В качестве примера рассмотрим W-грамматику W_1 , которая задана в примере 1. Рассмотрим, как порождаются терминальные цепочки с помощью этой грамматики:

а/ Для цели языка применим гиперправило и получим следующую цепочку:

<aA><bA><aA> /1/

б/ Для того, чтобы для гиперпонятия <aA> можно было применить второе гиперправило, из цепочки /1/ нужно порождать цепочку:

<anA><bnA><anA>

После применения гиперправила получим цепочку:

x<aA><bnA><anA>

в/ Применим второе гиперправило еще раз. Подобным образом получим:

xx<aA><bnnA><annA>

г/ Четвертое гиперправило будет применимо, если из цепочки будет совместимо порождена цепочка:

xx<a><bnn><ann> .

Применяя четвертое гиперправило, получим:

xx<bnn><ann> .

д/ Для гиперпонятия <bnn> можем порождать следующее гиперправило из третьего гиперправила заменой A на n :

<bnn> → y<bn> .

С помощью этого гиперправила из этой цепочки получим следующую цепочку:

xxy<bn><ann> .

е/ Из третьего гиперправила порождаем гиперправило <bn>→y<b>. Применяя его к цепочке, получим:

xxyy<b><ann> .

ж/ Применяя пятое гиперправило, получим цепочку:

xxyy<ann> .

Аналогичным образом из гиперпонятия **<ann>** получим **xx** .

Перейдем к формальному описанию нового определения W-грамматики. Множества метасистемы W-грамматики:

$M_i = \{m_j^i\}$ – бесконечное множество, состоящее из одного метапонятия с разными индексами.

$M = \{M_i\}$ – конечное множество множеств метапонятий.

τ_0 – конечное множество метатерминальных символов.

$\Pi \subset M \times (M \cup \tau_0)^*$ – множество контекстно-свободных правил и

$\mathfrak{L}_x = \{w \in (M \cup \tau_0)^* : x \underset{\Pi}{\Rightarrow} w\}$ – множество всех цепочек, выводимых из метапонятия **x** .

Множества языка и функция Ψ определяются как в предыдущем определении.

Говорим, что из цепочки α выводима цепочка β в смысле W-грамматики, если существуют α_1, α_2, ρ, δ, функции Ψ и Ψ^1 и гиперправило $\bar{w} \to \delta$, что $\alpha = \alpha_1 \rho \alpha_2$, $\Psi^1(\rho) \equiv \Psi(\bar{w})$ и $\beta = \Psi^1(\alpha_1)\ \Psi(\delta)\ \Psi^1(\alpha_2)$.

3. Основная идея анализатора W-грамматики заключается в том, что гиперправила рассматриваются как контекстно-свободные /КС/ правила, а гиперпонятия как "добавления" к ним. Для КС-правил – для метаправил и для гиперправил – используется метод рекурсивного спуска. В этом методе правила реализуются в виде булевских процедур, считывание входной строки происходит слева направо. Будем рассматривать только такие грамматики, в которых нет левой рекурсии.

Гиперправила реализуются в виде булевских процедур, а гиперпонятия рассматриваются как их параметры, являющиеся грамматическими объектами.

Поскольку для гиперпонятия существует несколько подходящих гиперправил, и заранее неизвестно, какие из гиперправил будут подходящими, так как зависит от конкретных значений метапонятий, поэтому в отличие от метода рекурсивного спуска для КС-грамматик здесь приходится вызывать все гиперправила /некоторое подмножество множества гиперправил/, чтобы найти подходящее. Для этой цели введем понятие вызова множества процедур, а затем перейдем к описанию грамматических объектов.

Под вызовом множества P булевских процедур $P = \{P_1,\ldots,P_n\}$ будем понимать последовательные вызовы элементов множества. Результат вызова будет истиной, если результат вызова некоторой процедуры P_i был истиной, и тогда последовательность вызова заканчивается. Результат вызова множества P будет ложным, если результат всех вызовов $P_1,\ldots,P_n$ был ложным. Обозначим этот вызов через P .

Введем понятия грамматической переменной и константы и операции над ними. Пусть G - контекстно-свободная грамматика ,
$G = (N, T, P, S)$,
N - конечное множество нетерминальных символов;
T - конечное множество терминальных символов;
P - конечный набор правила вида $A \to \gamma$, где $A \in N$ и $\gamma \in V^*$ $(V=N \cup T)$;
S - выделенный нетерминальный символ. Будем говорить, что β выводится из α $(\alpha,\beta \in V^*)$ одним шагом, если $A \to \gamma$ есть правило в P , и существуют такие α_1, $\alpha_2 \in V^*$, что $\alpha = \alpha_1 A \alpha_2$ и $\beta = \alpha_1 \gamma \alpha_2$. Тогда можно записать $\alpha \underset{G}{\to} \beta$. Выводимость является рефлексивным транзитивным замыканием выводимости с одним шагом. Обозначим выводимость через $\Rightarrow$.

Будем говорить, что A есть грамматическая переменная, если $A \in N$, а z - грамматическая константа, если $z \in T$. Грамматические переменные могут иметь значения, которые представляют собой строку, состоящую из грамматических переменных и констант. Строка $\gamma \in V^*$ является значением A в смысле G , если $A \Rightarrow \gamma$, при этом будем говорить, что γ имеет тип A . Таким образом, грамматикой определяется область значений грамматической переменной A . Задание грамматики G служит определением грамматических переменных и констант, и они являются начальными значениями и типами для себя. Под значением некоторой строки α будем понимать те строки β , которые выводим из нее, и строка α является типом β . Будем говорить, что два типа α и β подобны, если существует строка γ , типом которой является как α , так и β , и назовем ее общей.

Грамматическим переменным могут присваиваться значения только в том случае, если тип нового значения есть старое значение.

Между ними определена операция конкатентации, по значениям операндов которой строится гиперпонятие. Операцию конкатенации будем обозначать точкой /например, A.B / . Грамматическим выражением является такое выражение, в которое входят грамматические переменные и константы, соединенные операцией конкатенации.

Грамматические выражения могут стоять в качестве фактических параметров в обобщенном вызове P и в качестве формальных параметров в заголовках булевских процедур.

Параметры, которые являются грамматическими выражениями, передаются следующим образом. Сначала проверяется подобность фактических и формальных параметров. Если они не подобны, тогда производится возврат с результатом "ложь". В противном случае метапонятиям формального параметра присваиваются значения, на которые они были заменены при порождении выбранного

общего гиперпонятия. В том случае, когда процедура заканчивается с результатом "ложь", надо выбрать другое общее гиперпонятие /только в том случае, когда их число конечно/ и присвоить другие значения метапонятиям формального параметра соответственно замене. Если нет больше гиперпонятий, тогда нужно восстановить исходные значения метапонятий формального параметра и произвести возврат с результатом "ложь". Если результат - "истина", тогда метапонятиям фактического параметра присваиваются значения, на которые они были заменены при порождении выбранного гиперпонятия при входе в данную процедуру, и управление передается вызывающей процедуре.

В определении передачи параметров, являющихся грамматическими выражениями, важное место занимает проверка подобности, или другими словами, нахождение общего гиперпонятия.

Вообще говоря, подобность двух типов - неразрешимая проблема, поскольку она является определением пустоты пересечения двух КС-языков /описанных КС-грамматикой/.

Будем рассматривать только такие W-грамматики, в которых пересечение языков, определенных гиперпонятиями, определено и является КС-языком.

Мы стремились выбрать для проверки подобности двух метапонятий простой, нетрудоемкий алгоритм, который тем не менее является достаточно мощным для того, чтобы он был применим для W-грамматики, описываемой языка АЛГОЛ-68.

Подобность двух гиперпонятий проверяется следующим образом. Сначала проверяется совпадение. Если хотя бы одна из оставшихся, несовпадающих частей начинается метапонятием, делается попытка породить из этого метапонятия часть другого. Если оба гиперпонятия начинаются метапонятием и ни одно из них не порождалось из другого, тогда делается попытка порождения такого общего гиперпонятия, которое выводимо из второ-

го, в частности, делается подстановка вместо метапонятия его альтернативы. Если из некоторого метапонятия первого гиперпонятия выводима часть второго, но продолжить процесс дальше нельзя, тогда следует уменьшить эту часть второго гиперпонятия и продолжить проверку . Это исправление производится до тех пор, пока уменьшенная часть не станет пустой.

Рассмотрим этот метод проверки на примерах. Пусть гиперпонятия α и β имеют вид $\alpha = \langle aA \rangle$ и $\beta = \langle anA \rangle$, а метаправило – $A \to nA$. Первые символы совпадают, и из метапонятия **A** выводимо **nA** , то есть они подобны. Пусть $\alpha = \langle nnn \rangle$ и $\beta = \langle An \rangle$, а метаправило $A \to \alpha$ и $A \to nA$. Из **A** выводимо **nnn** , но тогда в β остается **n** . Уменьшим цепочку, выводимую из **A** , то есть возьмем **nn** в качестве порождения **A** . Последние символы совпадают. Подобность установлена. Пусть $\alpha = \langle Ab \rangle$ и $\beta = \langle aB \rangle$, а метаправила: $A \to aC$, $B \to Cb$. Из **A** не выводимо **aB** , а выводимо **aC** . Вместо $\alpha = \langle Ab \rangle$ рассмотрим в качестве α такое гиперпонятие, которое было получено из α заменой метапонятия **A** на его альтернативу /**aC**/ , то есть $\alpha = \langle aCb \rangle$. Первые символы совпадают /символ **a**/, и из **B** выводимо **Cb** .

Пример 3. Рассмотрим грамматику W_1 . Порождение цепочки **xxyyxx** было рассмотрено в примере 2., рассмотрим эту же цепочку с точки зрения распознавания с помощью выше описанных идей.

Имеются 7 процедур, первые пять процедур соответствуют гиперправилам, а последние два метаправилам. Процесс распознавания начинается вызовом первой процедуры, соответствующей первым гиперправилам.

В первой процедуре формируется фактический параметр **aA** и обобщенно вызываются остальные процедуры, соответствующие гиперправилам. Вторая процедура является подходящей, если метапонятие **A** было заменено в фактическом параметре на **nA**.

С помощью процедур, соответствующих метаграмматике, проверяется, допустима ли замена A на nA. Во второй процедуре считывается символ x , формируется фактический параметр aA и обобщенно вызываются процедуры. Подобным образом проверяется, является ли подходящей вторая процедура. Заменим A на nA, затем считывается следующий символ x , формируется aA и вызываются процедуры. Опять устанавливается, что вторая процедура является подходящей, заменим A на nA . При попытке считывания следующего символа x в этой процедуре, оказывается, что следующий символ не x . Тогда восстанавливается старое значение A , и продолжаются вызовы процедур. Третья процедура не является подходящей, поскольку фактический параметр aA начинается метатерминалом, а первый символ формального параметра bnA есть b , то есть они не совпадают. Четвертая процедура подходит, если A заменить на "пусто". Применим эту замену и вернемся к вызывающей процедуре /второй/ с результатом "истина" и со значением λ метапонятия A . Из второй процедуры вернемся к вызывающей процедуре, из которой был сделан возврат с результатом "истина" два раза и со значением n метапонятия A . Наконец, вернемся к первой процедуре с результатом "истина" и со значением nn метапонятия A .

Сформулируем фактический параметр bnn и будем вызывать процедуры. Установим, что третья процедура является подходящей при замене A на n , то есть A присваивается значение n . Считывается символ y , формируется по грамматическому выражению bA фактический параметр bn и вызываются процедуры. Снова третья процедура является подходящей, метапонятию A присваивается значение λ . Считывается символ y , и фактический параметр будет b . Вызываются процедуры. Пятая процедура является подходящей. Вызывается эта процедура и производится возврат из нее с результатом "ложь".

Аналогичным образом считывается символ xx для нетерминала ann W-грамматики. Получим, что строка xxyyxx порождается

грамматикой W_1.

4. Описание метода. Наш метод требует два стэка. Первый стэк небходим для рекурсивных вызовов с параметрами, то есть для описания грамматических выражений /для ссылок на элементы грамматического выражения/. Он предназначен также для хранения информации, связанной с рекурсивным вызовом, такой, как указатель стэка и другие указатели. Стэк используется также для хранения набора формальных параметров вызванной процедуры, то есть для гиперпонятия левой части гиперправила.

Второй стэк содержит таблицу ссылок на метапонятия гиперправила, одно и то же метапонятия гиперправила ссылаются на некоторый элемент таблицы, соответствующей ему. Стэк используется также для хранения значений метапонятий, присваиваемых им в процессе анализа. Значения метапонятий организованы в виде деревьев их порождений, сами метапонятия являются корнями деревьев.

Перед вызовом булевских процедур, соответствующих гиперправилам, необходимо занести элементы грамматического выражения в первый стэк. Если элемент является метапонятием, тогда в первый стэк помещается ссылка на соответствующий ему элемент таблицы.

Вызов булевских процедур, соответствующих гиперправилам, означает, что необходимо вызвать булевскую процедуру, соответствующую некоторому гиперправилу, и если результат вызова окажется ложью, тогда вызвать следующую процедуру. Эти вызовы продолжаются до тех пор, пока результат вызова не будет истиной или не возникает такая ситуация, что все гиперправила проверены и все результаты были "ложь", вследствие чего необходимо вернуться с результатом ложь к вызывающей процедуре.

После вызова, то есть организации рекурсии /в частности сохранения указателей/, необходимо сформировать во втором стэке

таблицу ссылок /например, занести адрес процедуры, соответствующей метапонятию/ и поместить элементы формальных параметров вызванной процедуры в первый стек.

Последним шагом входа в процедуру является нахождение общего гиперпонятия δ, которое может быть выведено из грамматического выражения α и из формальных парметров β /в дальнейшем просто δ, α и β /. Для этого надо вызвать булевскую процедуру с двумя параметрами α и β, реализующую следующий алгоритм /ANAL/.

Исходные параметры алгоритма: α и β. Алгоритм состоит из следующих шагов: Шаг 1. Пропуск одинаковых частей. При $\alpha = \gamma\alpha^1$, $\beta = \gamma\beta^1$, γ, α^1, $\beta^1 \in V^*$ в качестве α и β следует взять α^1 и β^1. Если гиперпонятие δ содержит и метапонятие, то в элемент таблицы ссылок, соответствующий метапонятию из α, нужно записать ссылку на элемент таблицы ссылок, соответствующий метапонятию из β, и поскольку они должны иметь одинаковые значения, запомнить место ссылок в первом стэке. После этого следует продолжить работу алгоритма следующим образом.

Шаг 2. Вывод первой части β /или α/ из первого символа-метапонятия α /или β/ при $\alpha = A\alpha^1$ и $\beta = \gamma\beta^1$ /или $\alpha = \gamma\alpha^1$ и $\beta = A\beta^1$ /, где γ, α^1, $\beta^1 \in V^*$, $A \in N$, $A \underset{G}{\Rightarrow} \gamma$. Построить дерево γ, то есть перенести его элементы во второй стэк. Вместо элемента γ записать ссылку на эту информацию. Запомнить эту замену в первом стэке при $\alpha = A\alpha^1$. Взять α^1 и β^1 в качестве α и β и продолжить работу алгоритма, переходя на шаг 1.

Шаг 3. Если $\alpha = A\alpha^1$ и $\beta = B\beta^1$, где $A, B \in N, \alpha^1, \beta^1 \in V^*$, тогда поставить вместо первого символа-метапонятия A его альтернативу. Альтернативы подставляются только вместо первого символа-метапонятия гиперпонятия α, являющегося грамматическим выражением. Записать дерево этой замены во второй

стэк, ссылку на него занести в элемент таблицы, соответствующий метапонятию **A**, и запомнить это соответствие в первом стэке.

При неудовлетворительном результате, когда замена не помогает, необходимо удалить из таблицы ссылку на дерево и перейти к шагу 4.

Шаг 4. Если α исчерпано /то есть все его символы пересмотрены/, а β нет, и $\beta = A\beta^1$, $A \in N$ /или наоборот/, или никакой шаг не применим, и последним был применен шаг 2, тогда сначала надо взять в качестве α гиперпонятие α_1 / $\alpha = \alpha_1\alpha_2$, где α_1, $\alpha_2 \in V^*$ / и попытаться вывести из β, и только потом выводить α_2. Этот процесс заканчивается, когда α_2 переходит в α. Перейти к шагу 5.

/В АЛГОЛе-68 такой ситуацией является, например, вызов гиперправила 1.3.3.a (**NOTION option**), на самом деле метатерминал **option** выводим из метапонятия **NOTION**, поэтому при первой попытке из **NOTION** выводится все гиперпонятие α. Правильный вывод получается лишь тогда, когда берется $\alpha = \alpha_1\ \alpha_2$, где α_2 = **option**, α_1 выводимо из **NOTION** и α_2 совпадает с **option**.

Примером иного рода является вызов гиперправила 4.8.a, где набор формальных параметров: **QUALITY NEST new PROPSETY1 QUALITY TAX PROPSETY2 definig INDICATOR with TAX**, где **QUALITY TAX** есть **PROP**, выводимое из **PROPSETY**/.

Шаг 5. Конечный шаг алгоритма. Если α и β исчерпаны одновременно, то производится останов с результатом "истина". В противном случае, когда с помощью шагов 3,4 уже проверены все возможные продолжения, необходимо аннулировать занесенные ссылки на новые деревья и произвести останов с результатом "ложь".

В шаге 2 описанного выше метода для проверки выводимости можно использовать любой метод синтаксического анализа, только выбранный метод должен быть применен для анализа строк, содержащих и метопонятия.

Скелет анализатора для W-грамматики является обычным анализатором для контекстно-свободных грамматик.

Попытаемся написать с использованием описанных выше идей анализатор на языке, подобном АЛГОЛ, для W-грамматики, заданной в примере 1. Исходным методом является метод рекурсивного спуска.

Будем использовать для обозначения элементов списков /деревья организуются в виде списков/ следующие типы: t1 - элемент, который является метатерминалом, t2 - ссылка на элемент типа t1, t3 - ссылка на метапонятие в таблице ссылок, t4 - ссылка на некоторый элемент таблицы ччылок, t5 - ссылка на некоторый элемент таблицы ссылок, в который была занесена ссылка на значение, t0 - конец списка.

Пусть Z [1 : u] будет входной строкой.

```
BEGIN INT ARRAY L [1:M] ; INT K; K:=1; INT H; H:=1 ;
¢  определение второго стека и его указателей ¢ ;
¢H - указатель входной строки ¢ ;
¢ - метаправила ¢ ;
¢T - такая функция, которая применяется к элементу списков и
     результатом которой является метапонятие, если оно не
     имеет только начальное значение или является метатерми-
     налом ¢ ;
¢ NEXT - такая функция, которая продвигает указатель списка ¢;

BOOL PROC A ; IF NOT T=n THEN GO TO R2; NEXT
IF NOT T=REF(A) THEN GO TO R2; NEXT; A:=TRUE; R2:A:=TRUE
END
```

```
¢  гиперправило для цели языка  ¢ ;
INT  ARRAY  S [ 1:15 ] ;
¢  определение места для гиперпонятий - фактических параметров, т.е. определение первого стека  ¢ ;
INT I ; I:=1 ;  ¢  определение указателя начала гиперпонятия  ¢ ;
INT J ; J:=K ;  ¢  указатель на начало таблицы ссылок ¢ ;
¢  формирование таблиц ссылок  ¢ ;
L [K] :=t3 ; L [K+1]:= REF(A); L [K+2] := ∅ ; K: = K+3 ;
S [I] :=t1 ; S [I+1]:='a' ;  ¢ элемент метатерминала ¢ ;
S [I+2] :=t4 ; S [v+3] :=J ;  ¢  элемент метапонятия ¢ ;
S [I+4] :=t0 ;  ¢  конец фактического параметра  ¢ ;
IF NOT HRULE (J,S,I)  THEN GO TO ERROR;  ¢  вызов гиперправила  ¢;
¢  формирование второго гиперпонятия правой части гиперправила в первом стэке  ¢  ;
S [I] : = t3; S [I+1] : ='b' ; S [I+1] : = t4; S [I+3] : = J ;
S [I+4] : = t∅ ;
IF NOT HRULE (J,S,I)  THEN GO TO ERROR;
```

Аналогичным образом формируются гиперправила третьего гиперпонятия правой части.

```
¢  гиперправила  ¢ ;
BOOL PROC H1 (OJ,OS,OI); REF TO INT ARRAY OS; INT OJ ;
REF TO INT OI ; INT ARRAY S [1:14] ;  ¢  первый стэк  ¢ ;
INT J; J:=K; INT R;  ¢  для начала таблицы изменений  ¢ ;
INT I ;I:=1 ;
L [K] : = t3 ; L [K+1] : =REF(A); L [K+2] : =∅ ; K : = K+3 ;
```

```
S [I] : = t1 ; S [I+1] : = 'a'; S [I+2] : = t1; S [I+3] : = 'n' ;
S [I+4] : = t4 ; S [I+5] : = J; S [I+6] : = tØ ; R: = I+7 ;

IF NOT ANAL (OJ,OS,OI,J,S,I,R) THEN GO TO FALSE

IF NOT Z[H] = 'x' THEN GO TO FALSE ;  H: = H+1

S [I] : = t1; S [I+1] : ='a' ; S [I+2] : = t4; S [I+3] : = J ;
S |I+4| : tØ ;

IF NOT HRULE (J,S,I) THEN GO TO FALSE;  H1: = TRUE ;

FALSE: WHILE S[R] = t5 DO BEGIN L[S[R+1]] : = Ø ; R:=R+2 END

HI: = FALSE END
```

Аналогично формируются и остальные гиперправила.

ЛИТЕРАТУРА

1. Van Wijngaarden, A., Maillonx, B.J., Peck, J.E.L., Koster, C. H. A., Sintzoff, M., Lindsey, C. H., Meertens, L. G. L. T. and Fisker. R. G. Revised Report on the Algorithmic Language ALGOL-68. Supplement to ALGOL BULLETIN № 36, March 1974, Edmonton, Alberta.

2. Маслов, А.Н.: Лекции по алгоритмическому языку АЛГОЛ-68.

3. Baker, J.L. Grammars with structured vocabulary: A model for the ALGOL-68 definition. Infomation and Control, Volume 20, /1972/, 351-395.

4. Greibach, S.A. Some restrictions on W-grammars. ACM proceedings of the 6th symposium on the theory of computing, Seattle, May, 1974.

5. Deussen, P.A. Decidability criterion for van Wijngaarden grammars . Acta Informatica 5., /1975/, 353-374.

6. Цейтин, Г.С. и другие. АЛГОЛ-68. Методы реализации, изд-во ЛГУ, 1976 г.

7. Братчиков, И.Л. Синтаксис языков программирования. Изд-во Наука, 1975.

8. Маслов, А.Н. Индексные грамматики и грамматика Вейнгаардена. Проблемы передачи информации. Том., XI., Вып. 3, 1975, 81-89.

S u m m a r y

Syntax-parser of W-grammars

L. Gerevich

A new definition is given for W-grammar. The method of recursive descent is made applicable for W-grammars through the introduction of a new, grammar-type variable.

Ö s s z e f o g l a l ó

Szintaktikai analizátor kétszintü nyelvtanokra

Gerevich László

A dolgozat uj meghatározását adja a Wijngaarden-féle kétszintü nyelvtannak. Egy uj tipusu, nyelvtan-tipusu változó bevezetésével teszi alkalmassá a rekurziv leereszkedés módszerét kétszintü nyelvtanokra.