ИССЛЕДОВАНИЕ МЕТОДА АВТОМАТИЧЕСКОЙ ГЕНЕРАЦИИ ПРАВИЛ ФРАГМЕНТАРНОГО АНАЛИЗА

Востриков А.В.*(sanchs@inbox.ru)*, Ермолаев А.М., Морозов С.Н., Пационов А.В. *(patsionov@gmail.com)*, Московский государственный институт электроники и математики
Клышинский Э.С. *(klyshinsky@mail.ru)*, Максимов В.Ю. *(vadimmax2000@mail.ru)*, Институт прикладной математики им. М.В. Келдыша РАН

В статье изложены результаты машинного эксперимента, целью которого было определение возможности автоматической генерации правил фрагментарного анализа на основе вычисления функций FIRST, LAST, FIRST2 и LAST2 для имеющихся грамматик синтаксического анализа текстов на естественном языке.

1. Введение

На данный момент системы автоматической обработки текстов (АОТ) являются быстро развивающейся отраслью науки, которой присуща повышенная сложность. Классический подход решения задач АОТ обладает экспоненциальным ростом вычислительных затрат. Кроме того, создание систем правил для АОТ сопоставимо со сложностью создания программного обеспечения для таких систем. Количество правил в реальных задачах таково, что в них также легко запутаться как в многочисленном и объемном программном коде. В связи с этим в последнее время начали активно развиваться методы, позволяющие автоматически генерировать правила для АОТ. Это произошло благодаря тому, что в сети Интернет накопилось достаточное количество параллельных текстов, в которых одна и та же информация примерно одними и теми же словами изложена на разных языках.

Одной из наиболее сложных задач при АОТ является синтаксический анализ. Задача полностью автоматической генерации правил для него до сих пор не решена. Это связано с тем, что порожденные правила могут приводить к увеличению многозначности при синтаксическом и других видах анализа, сами правила нуждаются в доработке и корректном встраивании в общую систему правил, а сам метод не решает проблемы экспоненциального роста сложности задачи. При этом следует подчеркнуть, что сама по себе автоматизация процесса генерации правил приводит к существенному росту производительности лингвистов и является чрезвычайно важной задачей.

Для ускорения работы синтаксического анализа обычно используется предшествующий ему этап фрагментарного анализа, который пытается выделить информацию о структуре предложения на основе выделения его фрагментов [1]. Одной из задач фрагментарного анализа является определение границ фрагментов предложения. Если границы фрагментов определять в соответствии с границами правил синтаксического анализа, то можно получить существенный прирост в скорости работы последнего. Однако на практике выделение и написание правил фрагментарного анализа

является трудоемкой и сложной задачей, а организация связей между правилами синтаксического и фрагментарного анализа еще больше увеличивает эту сложность. В связи с этим встает проблема автоматической генерации правил фрагментарного анализа на основе имеющихся правил синтаксического анализа.

2. Функции FIRST, LAST, FIRST2 и LAST2

Для начала изложим некоторые теоретические сведения, необходимые для дальнейшего изложения [2].

Грамматикой называется четверка G={VN, VT, P, E}. Здесь VN – множество нетерминалов, VT – множество терминалов, P – набор правил, описывающих цепочки, принадлежащие описываемому языку, E – начальный символ грамматики. В нашем случае терминалом будет являться некоторый шаблон, который будет применяться для поиска слова, отвечающего заданным требованиям, на заданном месте в предложении. Для унификации знаки препинания также будем считать словами.

Продукцией или правилом подстановки называется упорядоченная пара (U,x) записываемая как U::=x, где U – некоторый символ (левая часть правила), а x – цепочка символов (правая часть правила). U либо является единственным символом и является нетерминалом, либо произвольной цепочкой терминалов и нетерминалов, содержащей хотя бы один нетерминал. Множество всех левых частей правил составляет множество нетерминальных символов грамматики. Символы, входящие в x могут принадлежать как VT, так и VN. В нашем случае продукция задает порядок следования слов в фразе.

Если α любая строка символов грамматики, то FIRST(α) – это множество терминалов, с которых могут начинаются строки, выводимые из α. Если $\alpha \overset{*}{\Rightarrow} e$, то $e \in$ FIRST(α). Здесь e – пустая цепочка.

FIRST(X) вычисляется для всех символов X грамматики с применением следующих правил.

1. Если X – терминал, то FIRST(X) = {X}
2. Если X=e, то добавляем e к FIRST(X)
3. Если X – нетерминал и $X \rightarrow Y_1Y_2\ldots Y_k$ - продукция, то помещаем a в FIRST(X) если для некоторого i $a \in$FIRST(Y_i) и $e \in$FIRST(Y_i) $\forall$i таких что $Y_1Y_2\ldots Y_k \overset{*}{\Rightarrow} e$. В последнем случае добавляем e к FIRST(X).

Правила применяются до тех пор, пока не окажется больше терминалов и e, для которых их можно применить.

Если b любая строка символов грамматики, то LAST(b) – это множество терминалов, которыми могут заканчиваться строки, выводимые из b. Если $b \overset{*}{\Rightarrow} e$, то e$\in$ LAST(b).

Правила построения LAST(Y) для символов грамматики Y аналогичны правилам построения FIRST(X) для символов грамматики X.

Заметим, что для грамматик, не содержащих и не порождающих e-цепочки, правила вычисления FIRST, LAST и вводимых ниже FIRST2 и LAST2 будут значительно проще. Рассмотрением именно таких грамматик мы и займемся в данной работе.

Поясним вышесказанное на примере.

Пусть имеется набор продукций:

(1)

SENT::=<A1>

A1::=<A3><A4>|<A4>

A3::=[1]|[1]<A3>

A4::=[2]|[3]|[4][5][6]

Здесь терминалы ограничиваются квадратными, а нетерминалы – треугольными скобками.

Тогда FIRST и LAST для продукций будут следующими:

(2)

FIRST(SENT) = {[1],[2],[3],[4]}

FIRST(A1) = {[1],[2],[3],[4]}

FIRST(A3) = {[1]}

FIRST(A4) = {[2],[3],[4]}

LAST(SENT) = {[2],[3],[6]}

LAST(A1) = {[2],[3],[6]}

LAST(A3) = {[1]}

LAST(A4) = {[2],[3],[6]}

На основе полученной информации можно рассчитать значения функций FIRST2 и LAST2, показывающих с каких двух терминалов могут начинаться цепочки, генерируемые при помощи правила или какими двумя терминалами такие цепочки могут заканчиваться. При этом будем считать, что грамматика не содержит *e*-символов. В связи с этим алгоритм будет иметь некоторые отличия от приведенного в [2].

Если α любая последовательность символов грамматики, то FIRST2(α) – это множество пар терминалов, с которых могут начинаются строки, выводимые из α.

Если β любая последовательность символов грамматики, то LAST2(β) – это множество пар терминалов, которыми могут заканчиваться строки, выводимые из β.

FIRST2 в нашем случае рассчитывается по следующему алгоритму. Пусть рассматривается продукция вида $B \rightarrow A_1 A_2 \alpha$, причем α может являться произвольной (в том числе и пустой) цепочкой символов. В этом случае возможны следующие варианты.

1.Если $A_1 \in VT$, то $A_1 b \in FIRST2(B)$, где $b \in FIRST(A_2)$.

2.Если $A_1 \in VN$, то

a. $FIRST2(A_1) \in FIRST2(B)$;

b. Если $A_1^{+} \Rightarrow b$, где $b \in VT$, то $bc \in FIRST2(B)$, где $c \in FIRST(A_2)$; то есть если из A_1 выводятся цепочки состоящие из единственного терминала, то к FIRST2(B) прибавляются все комбинации конкатенаций, состоящие таких цепочек и терминалов, входящих в $FIRST(A_2)$.

LAST2 вычисляется аналогичным образом, но при этом рассуждения применяются не слева направо, а справа налево к продукциям вида $B \rightarrow \alpha A_2 A_1$.

Для пояснения вернемся к примеру. Ниже приведены результаты расчетов функций FIRST2 и LAST2 гля грамматики, приведенной в примере (1).

(3)

FIRST2(SENT) = {[1][1],[1][2],[1][3],[1][4],[4][5]}

FIRST2(A1) = {[1][1],[1][2],[1][3],[1][4],[5][6]}

FIRST2(A3) = {[1][1]}

FIRST2(A4) = {[4][5]}

LAST2(SENT) = {[1][1],[1][2],[1][3],[1][4],[5][6]}

LAST2(A1) = {[1][1],[1][2],[1][3],[1][4],[5][6]}

LAST2(A3) = {[1][1]}

LAST2(A4) = {[5][6]}

3. Метод

Итак, функции FIRST и LAST показывают, с каких терминалов могут начинаться или, соответственно, заканчиваться правила. При этом на основании результатов вычисления этих функций можно решить обратную задачу – определить какие правила могут начинаться или заканчиваться данным терминалом. При этом гипотетически возможен случай, когда с данного терминала будет начинаться небольшое количество правил. В этом случае можно будет говорить о том, что данный терминал является характеристическим. То есть, обнаружив данный терминал на входе, можно выдвинуть гипотезу или небольшое количество гипотез о структуре данного фрагмента. При этом количество комбинаций терминалов, получаемых с помощью функций FIRST2 и LAST2, и являющихся характеристическими, должно возрастать. На основе этой информации

можно будет попытаться построить правила фрагментарного анализа в автоматизированном или автоматическом режиме.

Исходя из вышеизложенных предположений, нами был проведен вычислительный эксперимент. Была разработана программа, которая вычисляла значения функций FIRST, LAST, FIRST2 и LAST2. На вход программы были поданы реально действующие грамматики русского и английского языков, разработанные для системы машинного перевода «Кросслятор 2.0». По результатам вычислений были выделены наборы правил, соответствующие терминалам или парам терминалов.

Работу программы можно разделить на 3 этапа.

На первом этапе работы ПО проводится преобразование исходной грамматики. Здесь вычленяются терминалы и записываются в отдельный список. Продукции преобразуются более в простой вид, удобный для проведения эксперимента.

На втором этапе проводится расчет FIRST и LAST для каждой продукции. Расчет основан на методе рекурсивного спуска по продукциям, описанном выше.

На третьем этапе проводится расчет FIRST2 и LAST2. Расчет также основан на методе рекурсивного спуска по продукциям, также описанном выше.

4. Результаты эксперимента

Для экспериментов использовалась грамматика английского языка, содержащая в себе более 550 правил и порядка 250 уникальных терминалов. В результате генерации множеств FIRST2 и LAST2 было обнаружено, что более 500 пар терминалов однозначно идентифицируют начало правила и порядка 50 пар терминалов однозначно идентифицируют окончание правила. Кроме того, было обнаружено более 20 терминалов, однозначно идентифицирующих окончание правила, а примерно такое же количество терминалов не может заканчивать ни одно правило. Аналогичные результаты были получены и для функции FIRST.

Исследование полученных правил показало, что среди них присутствуют такие, которые и в самом деле являются претендентами на правила фрагментарного анализа. Под каждым правилом, приведенным ниже, показан позитивный пример. Формат терминала здесь следующий: [часть речи; нормальная форма; параметры].

(4)

[pn_pers;'WHO';][mod;'MAY';form prf] начинает фрагмент WHO_FRASE

Jane went to ask who may show us the way.

[noun;'MILLION';][noun;";number pl]

[noun;'BILLION';][noun;";number pl]

[noun;'THOUSAND';][noun;";number pl]

[noun;'HUNDRED';][noun;''; number pl] заканчивают фрагмент NOUN_FRASE

two hundred cars

[part;'NO';][noun;'MATTER';] начинает фрагмент NO_MATTER

no matter

[pn_poss;'';][noun;'';] начинает фрагмент DEF_OBJECT

his cat

[prep;'OF';][article;'';]

[prep;'OF';][pn_poss;'';]

[prep;'OF';][part;'NO';] начинают фрагменты S_DET

one of the finest smells

[mod;'';form pr][adv;'';]

[mod;'';form pr][nadv;'NOT';] начинают фрагменты BEPT_ADJ0, M_A1

may easily go

can not say

5. Обсуждение

Полученные результаты не являются окончательными. Так, с одной стороны, если мы имеем правило вида $A \rightarrow B\alpha$ и $t \in FIRST(B)$, то $t \in FIRST(A)$ и, следовательно, один и тот же терминал будет встречаться несколько раз у разных правил. Аналогичные рассуждения применимы и к LAST. Но если изменить алгоритм расчета FIRST так, чтобы символ входил только в то правило, которое непосредственно порождает цепочку, начинающуюся с данного терминала, то количество правил, однозначно идентифицируемых единственным терминалом, должно увеличиться.

С другой стороны, преобразование терминалов проводилось таким образом, что терминалы, различающиеся частью хранимой в них информации, считались различными. То есть, например, терминал, проверяющий произвольное существительное, будет отличаться от терминала, требующего существительного в именительном падеже. Таким образом, с одним словом входного предложения могут успешно сравниться сразу несколько терминалов. Кроме того, входное предложение в подавляющем большинстве случаев будет содержать в себе морфологическую омонимию. В связи с этим вероятность применения нескольких терминалов к одному и тому же слову существенно возрастает.

Однако даже простое сужение количества возможных вариантов применения правил в ходе синтаксического анализа должно приводить к существенному ускорению этапа анализа предложения.

Еще одной проблемой является тот факт, что парное сочетание терминалов может встретиться не только в начале или в конце цепочки, генерируемой правилом, но и в середине. Это может произойти для пары правил вида $A \rightarrow \alpha\beta$, $B \rightarrow \gamma$ таких, что $ab \in FIRST2(\gamma)$ или $ab \in LAST2(\gamma)$ и $a \in LAST(\alpha)$ и $b \in FIRST(\beta)$. В этом случае парное сочетание не будет однозначно идентифицировать начало или окончание цепочки, выводимой из правила. В связи с этим эксперимент необходимо продолжить и найти и исключить из результатов все пары, попадающие под описанный случай.

Среди сгенерированных правил, например, встречались следующие (под правилом приведено два примера – позитивный и негативный).

(5)

[sign;',';][conj;'AS';] начинает фрагмент AS_CLAUSE

The result was obvious, as the pressure had fallen almost to zero.

The result was obvious, as obvious as "2 x 2 = 4".

[prep;'IN';][noun;'ORDER';] начинает фрагмент IN_ORDER_TO

in order to

Keep your files in order to save time.

[conj;'AND';][card;'';] завершает фрагмент числительного

two hundred and three

There were 5 tasks. He solved five and three of them were wrong

Однако количество кандидатов на правила позволяет предположить, что количество сгенерированных правил может быть все равно достаточно велико. В связи с этим полученные результаты позволяют надеяться на практическую разрешимость задачи автоматизированной генерации правил фрагментарного анализа после соответствующей доработки алгоритма.

1. Puscasu G. A Multilingual Method for Clause Splitting. / In Proceedings of the 7th Annual CLUK Research Colloquium. Birmingham. 2004.

2. Ахо А.В., Сети Р., Ульман Д.Д. Компиляторы: принципы, технологии и инструменты // М.: Вильямс, 2003.