УДК 004.021

ЛОГИКО-ЛИНГВИСТИЧЕСКИЙ МЕТОД ПОСТРОЕНИЯ ДЕТЕРМИНИРОВАННОГО КОНЕЧНОГО АВТОМАТА ЛЕКСИЧЕСКОГО АНАЛИЗАТОРА ГРАММАТИКИ, ЗАДАННОЙ РЕГУЛЯРНЫМ МЕТАСИМВОЛИЧЕСКИМ ВЫРАЖЕНИЕМ

Д.О. ГЛУХОВ, Р.П. БОГУШ, М.В. МАТЮШ, Т.М. ГЛУХОВА

Полоцкий государственный университет
Блохина, 29, Новополоцк, 211440, Беларусь

Поступила в редакцию 8 апреля 2010

Предлагается логико-лингвистический метод работы с грамматическими структурами, особенность которого заключается в формализации знаний о преобразовании лингвистических конструкций, в отличие от описания алгоритма преобразования с использованием теоретико-множественных операций. Такой подход применим в том случае, если результат сложного преобразования интуитивно понятен и является легко интерпретируемым. Последним свойством, в частности, и обладает комплексный алгоритм построения лексического анализатора формальной грамматики, заданной регулярным метасимволическим выражением.

Ключевые слова: лексические анализаторы, регулярные выражения, конечный детерминированный автомат.

Введение

Традиционно построение лексического анализатора для языка, заданного регулярным выражением, выполняется путем применения алгоритмов построения правил грамматики по регулярному выражению, записанному на языке метасимволов, приведения грамматики к автоматному виду и приведения, если такая необходимость существует, недетерминированного автомата к детерминированному виду [1, 2].

В данной работе приводится обобщение опыта авторов по конструированию лексических анализаторов в виде логико-лингвистического метода построения детерминированного конечного автомата по регулярному выражению, записанному на языке метасимволов.

В первую очередь, потребность в новом методе заключается в ресурсоемкости применения теоретико-множественных математических методов. Так, например, решение задачи перевода конечного автомата с n исходными состояниями к детерминированному виду требует построения 2^{n-1} новых состояний и выполнения ресурсоемкого алгоритма отсева недостижимых состояний. Хотя результат работы алгоритма приведения конечного автомата к детерминированному виду интуитивно понятен и заключается в объединении состояний, устраняющем недетерминированность.

Теоритико-множественные преобразования ресурсоемки, а результаты их интуитивно понятны, и, следовательно, могут быть описаны в рамках иного представления знаний об этих преобразованиях. Современные техники извлечения, формализации и использования знаний вербального уровня дают инструмент для работы с лингвистическими структурами в рамках не теоретико-множественных операций, а в рамках систем, основанных на знаниях [3–6].

Определение логико-лингвистического метода построения детерминированного конечного автомата

Определение 1. Фиксированное состояние — это такое положение в метасимволическом представлении регулярного выражения, которое следует за каждым обязательным или возможным единичным включением символа. К множеству фиксированных состояний мы также будем относить состояние, предшествующее символам, определяемым регулярным выражением: так называемое, начальное состояние, конечное состояние и состояние ошибки.

Определение 2. Альтернативный фрагмент — это фрагмент метасимволического представления регулярного выражения, который может рассматриваться в данном фиксированном состоянии как имеющий быть непосредственно следующим, с учетом необязательности присутствия предыдущих, либо при явном присутствии фрагментов, объединенных логической связкой ИЛИ.

Определение 3. Концовка альтернативного фрагмента — это такой фрагмент метасимволического представления регулярного выражения с некоторой позиции альтернативного фрагмента до, либо явного метасимвола закрытия группы альтернатив, соединенных логическими связками ИЛИ, либо, если альтернативный фрагмент получен путем опускания необязательных предшествующих символов, положение, непосредственно предшествующее следующему фиксированному состоянию.

Определение 4. Одинаковая концовка альтернативных фрагментов — это такой фрагмент метасимволического представления регулярного выражения, который одинаков для концовок рассматриваемых альтернативных фрагментов.

Определение 5. Одинаковое начало альтернативных фрагментов — это такой фрагмент метасимволического представления регулярного выражения, который одинаков для начал рассматриваемых альтернативных фрагментов.

Определение 6. Допустимый альтернативный фрагмент — это альтернативный фрагмент, для которого выполняется дополнительное условие разрешенности (допустимости).

Утверждение 1. Соответствующие фиксированные состояния одинаковых начал альтернативных фрагментов и одинаковых концовок альтернативных фрагментов есть одни и те же состояния для всех рассматриваемых альтернативных фрагментов.

Если в регулярном отношении встречаются последовательности необязательных цепочек вида $_{A}x*p*r*x*t*u*$, где A — начальное состояние последовательности, то с целью построения детерминированного конечного автомата разбора эти последовательности приводятся к виду:

$$_{A}x?_{A1}x*p?_{A2}p*r?_{A3}r*x?_{A4}x*t?_{A5}t*u?_{A6}u*,$$

с вводом дополнительных состояний для позиций последовательности.

Правило 1. Множество допустимых альтернативных фрагментов для состояний последовательности ограничивается. Для альтернативных фрагментов с одинаковыми началами, допустимым является фрагмент, начало которого следует ранее в метасимволическом представлении регулярного выражения.

По сути, правило 1 обеспечивает детерминированность (однозначность) интерпретации записи при логическом построении конечного автомата разбора.

Например, $_{A}\, x*p*r*t*x*n*p*a\, _{B}$, преобразуем к виду с дополнительными состояниями:

$$_{A}\, x?\, _{A1}\, x*p?\, _{A2}\, p*r?\, _{A3}\, r*t?\, _{A4}\, t*x?\, _{A5}\, x*n?\, _{A6}\, n*p?\, _{A7}\, p*a\, _{B.}$$

Для каждого состояния приведем множество альтернативных фрагментов, выделяя альтернативные фрагменты с одинаковыми началами (табл. 1).

Поскольку, в принципе, нас интересуют исключительно первые символы альтернативных фрагментов, то процесс определения альтернативных фрагментов от некоторой позиции регулярного выражения будет заключаться в последовательном опускании необязательных элементов, следующих за данной позицией.

Ограничим множество альтернативных фрагментов до множества допустимых фрагментов, используя правило ограничения 1 (табл. 2).

Правила регулярной грамматики для последовательности тогда будут (табл. 3):

Таблица 1. **Альтернативные фрагменты для каждого состояния последовательности**

A	A1	A2	A3	A4	A5	A6	A7
x? A1 x...* *p? A2 p*...* *r? A3 r*...* *t? A4 t*...* ***x? A5 x*...*** *n? A6 n*...* ***p? A7 p*...*** *a B*	*A1 x*...* *p? A2 p*...* *r? A3 r*...* *t? A4 t*...* ***x? A5 x*...*** *n? A6 n*...* ***p? A7 p*...*** *a B*	*A2 p*...* *r? A3 r*...* *t? A4 t*...* *x? A5 x*...* *n? A6 n*...* ***p? A7 p*...*** *a B*	*A3 r*...* *t? A4 t*...* *x? A5 x*...* *n? A6 n*...* *p? A7 p*...* *a B*	*A4 t*...* *x? A5 x*...* *n? A6 n*...* *p? A7 p*...* *a B*	*A5 x*...* *n? A6 n*...* *p? A7 p*...* *a B*	*A6 n*...* *p? A7 p*...* *a B*	*A7 p*...* *a B*

Таблица 2. **Допустимые фрагменты для каждого состояния последовательности**

A	A1	A2	A3	A4	A5	A6	A7
x? A1 x...* *p? A2 p*...* *r? A3 r*...* *t? A4 t*...* *n? A6 n*...* *a B*	*A1 x*...* *p? A2 p*...* *r? A3 r*...* *t? A4 t*...* *n? A6 n*...* *a B*	*A2 p*...* *r? A3 r*...* *t? A4 t*...* *x? A5 x*...* *n? A6 n*...* *a B*	*A3 r*...* *t? A4 t*...* *x? A5 x*...* *n? A6 n*...* *p? A7 p*...* *a B*	*A4 t*...* *x? A5 x*...* *n? A6 n*...* *p? A7 p*...* *a B*	*A5 x*...* *n? A6 n*...* *p? A7 p*...* *a B*	*A6 n*...* *p? A7 p*...* *a B*	*A7 p*...* *a B*

Таблица 3. **Правила регулярной грамматики для каждого состояния последовательности**

A	A1	A2	A3	A4	A5	A6	A7
A1 → Ax *A2 → Ap* *A3 → Ar* *A4 → At* *A6 → An* *B → Aa*	*A1 → A1x* *A2 → A1p* *A3 → A1r* *A4 → A1t* *A6 → A1n* *B → A1a*	*A2 → A2p* *A3 → A2r* *A4 → A2t* *A5 → A2x* *A6 → A2n* *B → A2a*	*A3 → A3r* *A4 → A3t* *A5 → A3x* *A6 → A3n* *A7 → A3p* *B → A3a*	*A4 → A4t* *A5 → A4x* *A6 → A4n* *A7 → A4p* *B → A4a*	*A5 → A5x* *A6 → A5n* *A7 → A5p* *B → A5a*	*A6 → A6n* *A7 → A6p* *B → A6a*	*A7 → A7p* *B → A7a*

*abc+x*p*x*(a|k)*p** — *1. s = expand(s),*
*abc+x?x*p?p*x?x*(a|k)?(a|k)*p?p** — *2. s = expand2(s),*
*abcc*x?x*p?p*x?x*(a|k)?(a|k)*p?p** — *3. s = addStates(s),*
A a B b C c D c x? E x* p? F p* x? J x* (a | k)? H (a | k)* p? L p* M* — *4. a = makeFA(s).*

*a(b|c)*d* — *2. s = expand2(expand(s)),*
*a(c|d)?(c|d)*d* — *3. s = addStates(s),*
A a B (c | d)? C (c | d) d D.*

Преимущества логико-лингвистического метода с точки зрения наглядности дидактического материала заключаются в следующем:

1) работать с метасимволическим представлением регулярной грамматики удобнее в силу компактности записи, определяющей грамматику;

2) операции, выполняемые над метасимволической записью, являются более наглядными и понятными, в отличие от громоздких представлений исходных данных и результатов преобразования грамматик, заданных набором грамматических правил;

3) при построении конечного автомата разбора логико-лингвистическим методом формируется четкое представление о взаимосвязи позиций и символов исходного регулярного выражения с состояниями и переходами конечного автомата;

4) при построении конечного автомата разбора логико-лингвистическим методом формируется понимание смысла итераторов и их эквивалентных представлений на основе других итераторов.

Алгоритм преобразования метасимволического представления регулярной грамматики в детерминированный конечный автомат

Логико-лингвистический подход к описанию преобразования метасимволического представления регулярной грамматики в детерминированный конечный автомат (ДКА) позволяет задать следующий алгоритм преобразования.

1) Преобразовать метасимволическое представление регулярного выражения к единому формату для определения фиксированных состояний путем заменены конструкции *x*+ с оператором итерирования + на конструкции с оператором * итерирования *xx** и элементы последовательности необязательных цепочек *x** с итератором * на конструкцию *x?x**.

2) Определить фиксированные состояния.

3) Для каждого фиксированного состояния определить альтернативные фрагменты.

4) Для альтернативных фрагментов с одинаковыми началами и концовками определить общие состояния.

5) Построить ДКА, состояния которого берутся из определенных нами фиксированных состояний. Переходы между состояниями осуществляются по обязательным или возможным единичным символам, предшествующим в метасимволическом представлении фиксированным состояниям, в том числе с учетом итераторов. А петли в конкретных состояниях задаются всеми возможными началами альтернативных фрагментов, не включающими переходы между состояниями.

Рассмотрим несколько примеров:

Пример 1. Регулярное выражение для чисел с плавающей точкой:

[0-9][.]?[0-9]+([eE][+-]?[0-9]+)?.*

Заменим конструкции с оператором итерирования *x*+ на конструкции с оператором итерирования *x** и изначальную конструкцию *x** с итератором * на конструкцию *x?x**, получим:

[0-9]?[0-9][.]?[0-9][0-9]*([eE][+-]?[0-9][0-9]*)?.*

Определим фиксированные состояния:

	[0-9]?	*[0-9]*	*[.]?*	*[0-9]*	*[0-9]**	*([eE]*	*[+-]?*	*[0-9] [0-9]*)?*
|	|	|	|		|	|	|	|
A	*B*	*C*	*D*		*E*	*F*	*G*	*H.*

Построим ДКА разбора:

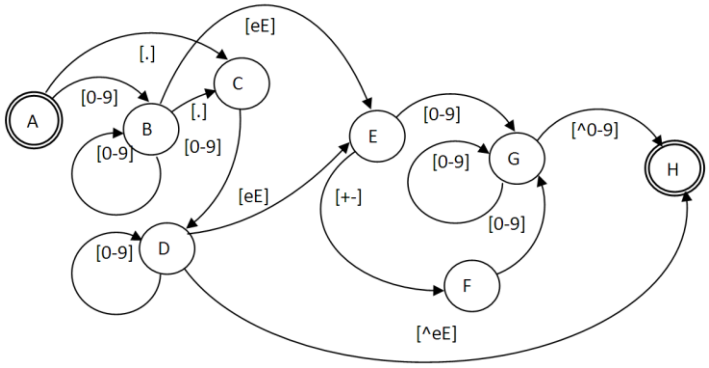

Рис. 1. Схема ДКА разбора текста на соответствие регулярному выражению, описывающему запись чисел с плавающей точкой в нотации языка C++, построенная с применением логико-лингвистического метода

Для получения полностью определенного ДКА требуется ввести еще одно состояние — состояние ошибки, и определить переходы из каждого состояния для всех неопределенных еще в качестве переходов для данного состояния символов переход в состояние ошибки.

Приведенный пример иллюстрирует легкость и наглядность логико-лингвистического преобразования по сравнению с традиционными громоздкими алгоритмами на основе теоретико-множественных преобразований грамматик.

Так для решения данной задачи в традиционной постановке пришлось бы выполнить следующие ресурсоемкие алгоритмы [1, 2, 7–10]:

1) алгоритм построения набора правил регулярной грамматики по метасимволическому представлению;

2) алгоритм приведения грамматики к автоматному виду;

3) алгоритм построения конечного автомата по множеству правил автоматной грамматики;

4) алгоритм приведения конечного автомата к детерминированному виду в пространстве состояний, представляющих собой всевозможные сочетания состояний исходного автомата;

5) алгоритм отсева недостижимых состояний.

Последний алгоритм искусственно выделен из алгоритма приведения конечного автомата к детерминированному виду с целью иллюстрации ресурсоемких операций.

Пример 2. Регулярное выражение для константных строк языка C++:

["]([^"\n\\]|[\\].)["],*

или, что тоже

["]([^"\n\\]|[\\][^\n])["].*

Определим фиксированные состояния:

["] ([^"\n\\] | [\\] [^\n]) ["]*

| | | | |

A B C D.

В состоянии B мы имеем явным образом определенные альтернативные фрагменты [^"\n\\] и [\\][^\n], завершающиеся закрывающей скобкой группировки фрагментов.

Таким образом, схема ДКА будет следующей:

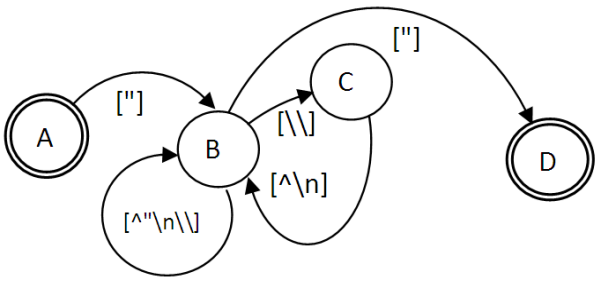

Рис. 2. Схема ДКА разбора текста на соответствие регулярному выражению, описывающему константные строки в нотации языка C++, построенная с применением логико-лингвистического метода

Пример 3. Регулярное выражение для блочных комментариев языка C++:

[/][*]([^*]|[*]+[^*/])[*]+[/].*

Заменим конструкции с оператором итерирования + на конструкции с оператором итерирования *, получим:

[/][*]([^*]|[*][*][^*/])*[*][*]*[/].*

Определим фиксированные состояния:

[/] [*] ([^*] | [*] [*] [^*/])* [*] [*]* [/]*

| | | | | | |

A B C D E F.

Определим альтернативные фрагменты для фиксированного состояния *C*, указав имена состояний буквами нижнего регистра:

$$. [\hat{}*], \quad (1)$$
$$[*]_D [*]^* [\hat{}*/], \quad (2)$$

и, в силу необязательности группы явно определенных альтернатив (...)* третьей альтернативой будет концовка регулярного выражения:

$$[*]_E [*]^* [/]_F . \quad (3)$$

Покажем для альтернативных фрагментов (2) и (3) наличие одинаковых начал:

$$[*]_D [*]^* ...,$$
$$[*]_E [*]^* ....$$

Следовательно, в соответствии с утверждением 1 состояния *D* и *E* есть одно состояние. Обозначим данное состояние именем *H*. Тогда имеем:

$$[/] \quad [*] \quad ([\hat{}*] \mid [*] \quad [*]^* [\hat{}*/])^* \quad [*] \quad [*]^* \quad [/]$$
$$A \quad B \quad C \qquad H \qquad H \qquad F.$$

Рассмотрим альтернативные фрагменты для состояния *H*:

1) $[*]^*_H$;
2) $[\hat{}*/])^*_C$;
3) $[/]_F$.

Они задают переходы в состояния *H, C, F* соответственно. Построим ДКА:

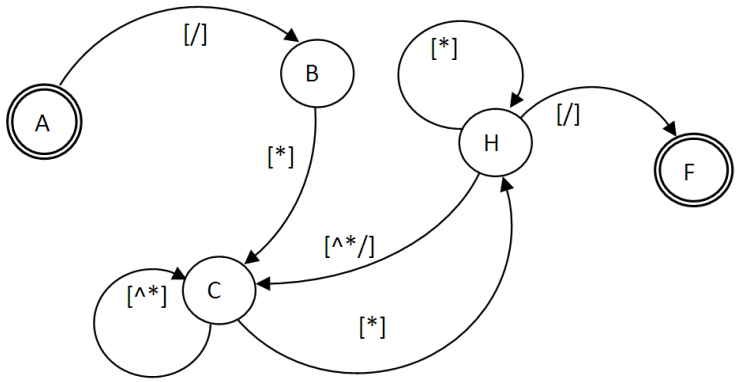

Рис. 3. Схема детерминированного конечного автомата разбора текста на соответствие регулярному выражению, описывающему блочные комментарии в нотации языка C++, построенная с применением логико-лингвистического метода

Заключение

Логико-лингвистические преобразования обладают легкостью и наглядностью по сравнению с традиционными громоздкими алгоритмами на основе теоретико-множественных преобразований грамматик.

Применение данного подхода позволяет не только решить задачу быстрого построения детерминированного конечного автомата по метасимволическому представлению регулярного выражения, но и раскрыть интуитивно понятные закономерности преобразований при обучении студентов теории формальных грамматик и правилам построения лексических анализаторов, что формирует понимание цели выполняемых преобразований на каждой стадии преобразований.

Логико-лингвистические преобразования представляют собой перспективное направление в области преобразований лингвистических структур.

LOGICAL-LINGUISTIC METHOD OF CONSTRUCTING DETERMENITIC FINITE AUTOMAT OF LEXER GRAMMAR, GIVEN WITH REGULAR METASYMBOLIC EXPRESSION

D.O. GLUKHOV, R.P. BOGUSH, M.V. MATSIUSH, T.M. GLUKHOVA

Abstract

In the article proposed logical-linguistic methods of work with grammatical structures, feature of which is formalization of knowledge about the transformation of linguistic structures in contrast to the description of the transformation algorithm using set-theoretic operations. This approach is applicable in the case if the result of a complex transformation is intuitively understandable and an easily interpretable. The last property, in particular, has and a complex algorithm for constructing the lexical analyzer of formal grammar, given with regular metasymbolic expression.

Литература

1. *Гордеев А.В., Молчанов А.Ю.* Системное программное обеспечение. СПб., 2001.
2. *Серебряков В.А., Галочкин М.П.* Основы конструирования компиляторов. М., 2001.
3. *Брой М.* Информатика. Структуры систем и системное программирование: В 4 частях: Пер. с нем. М., 1996. Ч. 3.
4. *Глухов Д.О.* Экспертная система на нечетких продукционных правилах для обследования сложного объекта / Сб. докл. Междунар. конф. по мягким вычислениям и измерениям. Т. 2. СПб., 1998. С. 174–176.
5. *Hlukhau D.* Dynamic expert system by fuzzy inference rules to automations an examination of complex objects / Budownictwo i Inzynieria Srodowiska / Zielonogorsk: Politechnika Zielonogorska, 1998. ISBN 83-85911-60-X. P. 105–109
6. *Глухов А.О., Глухов Д.О., Трофимов В.В.* Мониторинг по нечетким продукционным правилам в сложных динамических системах с изменяющейся структурой / Вестник факультета менеджмента СПбГУ. СПб: СПбГУ, 1998
7. *Глухов А.О., Глухов Д.О., Глухова Т.М.* Методические указания к лабораторным работам по курсу "Системное программное обеспечение. Лексические анализаторы" для студентов специальности 40.02.01. Новополоцк: ПГУ, 2002.
8. Stephen C. Johnson Yacc: Yet Another Compiler-Compiler. Bell Laboratories Murray Hill, New Jersey 07974.
9. Compilers: Principles, Techniques, and Tools (http://www1.fatbrain.com/asp/bookinfo/bookinfo.asp?theisbn=0201100886&vm=c) Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman Addison-Wesley Pub Co ISBN: 0201100886
10. Lex & Yacc (http://www.ora.com/catalog/lex/index.html) John R. Levine, Tony Mason, Doug Brown Paperback - 366 pages 2nd/updated edition (October 1992) O'Reilly & Associates ISBN: 1565920007