УДК 519.7

Н.А. Валенда, В.Н. Ляпота

Харьковский национальный университет радиоэлектроники, Харьков

ОБ ОДНОМ МЕТОДЕ РЕАЛИЗАЦИИ ЛИНГВИСТИЧЕСКОГО ПРОЦЕССОРА

Статья посвящена построению лингвистического процессора на основе методов, которые используются в формальных языках для построения трансляторов. Рассматриваются особенности совместного применения формальных методов и средств для анализа естественных языков.

Ключевые слова: *словарь, лексема ,конечный автомат, семантическая функция, лингвистический процессор.*

Введение

Обработка постоянно увеличивающихся объемов информации является одной из актуальных задач в области информационных технологий. Значительная часть информации представляется в виде текстов на естественном языке. Задача анализа естественного языка (ЕЯ) является одним из направлений исследований в области искусственного интеллекта (ИИ). Работы по созданию систем анализа ЕЯ начались в 50-х годах 20-го века. Одной из предпосылок, способствовавших возникновению этого направления, была успешная разработка компиляторов для языков программирования, что подталкивало к выводам о возможности применения аналогичных подходов для ЕЯ. Статья посвящена построению лингвистического процессора, использующего математический аппарат, разработанный для формальных языков.

Этапы разбора языковых конструкций

Компьютерную систему, реализующую формальную лингвистическую модель и способную обрабатывать ЕЯ, будем называть лингвистическим процессором (ЛП). Основная функция ЛП состоит в сопоставлении конструкциям ЕЯ формального представления, которое отражает значение исходной конструкции [1]. ЛП поэтапно анализирует языковые конструкции и преобразует их в суперпозиции семантических функций (рис. 1).

ЛП является многопроходным транслятором. Первый проход соответствует лексическому анализу (ЛА) [2]. На этом этапе входной текст разбивается на лексемы – последовательности символов, объединенные в единицы текста. Рассматриваются следующие классы лексем: слова, цифры, знаки препинания, признак конца предложения.

Лексема имеет вид:

$$l = X_{p_1,\ldots,p_n}, \qquad (1)$$

где X задает класс лексемы, а p_1 – ссылку на элемент текста x; $p_2,\ldots,p_{n-1}$ – морфологические признаки $M(P_{morf})$; p_n – ссылка на статью семантического словаря $V_i(x)$.

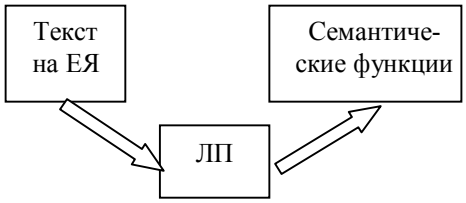

Рис. 1. Схема разбора ЯК

В результате работы лексического анализа формируется последовательность лексем:

$$\{L_1 L_2 \ldots L_m\},$$

где L_i определяется как

$$L_i = \begin{cases} l_j,\ x \to X_{p_1,\ldots,p_n} \\ l_1 \vee \ldots \vee l_k,\ x \to \{X^1_{p_1,\ldots,p_n},\ldots,X^k_{p_1,\ldots,p_n}\} \end{cases}.$$

Последовательность лексем подается на вход синтаксическому анализу (СА), задачей которого, является построение дерева разбора языковой конструкции (ЯК). Традиционным методом анализа является разбор на основе грамматик и морфологической информации, полученной на предыдущем этапе. Многолетняя история анализа ЕЯ показывает, что невозможно создать систему производящую корректный разбор предложения только на основании морфологической информации и формального аппарата КС-грамматик. Особенно сложно добиться хороших результатов разбора ЯК с помощью аппарата грамматик для таких языков как русский, где нет четко заданной последовательности слов в предложении [3]. Можно сказать «водитель ведет автобус» или «автобус ведет водитель», и оба высказывания являются корректными и имеют одинаковое значение.

В данной работе предлагается подход, позволяющий использовать семантику на уровне синтаксического анализа. Тесная связь синтаксиса и семантики не дает возможности получить эффективные системы и алгоритмы синтаксического анализа

в отрыве от смысла высказываний и текста в целом. Установление синтаксической связи требует проверки на семантическом уровне.

СА производит анализ предложений, для которых строит одно или несколько деревьев разбора. Эти деревья разбора подаются на вход семантического анализатора. Такой подход приводит к значительным затратам времени и ресурсов сначала на построение деревьев, а затем их анализ и выбор единственного, на основе которого строится семантическое представление.

В данной работе предлагается сузить область применения СА с предложения до словосочетания. Структура словосочетаний значительно проще, что позволяет строить более простые грамматики, на основе которых можно производить детерминированный анализ без возвратов. Это ведет к сокращению времени анализа. Кроме того, в словосочетаниях значительно реже, чем в предложениях встречаются ситуации неоднозначности.

В результате работы синтаксического анализа формируется последовательность лексем:

$$\{L'_1 L'_2 ... L'_m\},$$

где $L'_i = \begin{cases} L_i, \text{если } x - \text{слово}; \\ f_i(x_1^l, ..., x_n^l), \text{если } x - \text{словосочетание}. \end{cases}$

На этапе семантического анализа происходит построение семантического представления предложения, которое задает отношения между объектами, действиями, их свойствами, а для слов, которыми они выражены, задает соответствующие значения. На этапе семантического анализа происходит заполнение позиций предиката лексемами данного отрезка текста. В процессе получения единственных значений единиц текста получаем результирующее представление в виде суперпозиции семантических функций.

Выделение лексем из текста документа

Первым этапом анализа является преобразование текста документа из набора символов в множество элементов различных типов. Такими типами являются слова, цифры, знаки препинания, символы-разделители, рисунки, формулы. Для анализа текстовой информации важны первые три типа, разделители можно удалить, а рисунки и формулы перенести в результирующий документ без изменений. Для решения этой задачи можно воспользоваться конечными автоматами, разработанными для анализа языков программирования [4]. Конечный автомат – это пятерка

$$M = (V_T, Q, \delta, q_0, F), \qquad (2)$$

где V_T – терминальный алфавит, конечное множество допустимых символов; Q – конечное множество состояний; δ – множество функций перехода, имеющих вид $\delta(Q, V_T) = Q$; $q_0 \in Q$ - начальное состояние; $F \subseteq Q$ - множество заключительных состояний.

Пара $(q, aw) \in Q \times V_T^*$ называется конфигурацией автомата M.

Конфигурация (q_0, w) называется начальной, а (q, ε) – заключительной, если $q \in F$.

Такт автомата M представляется бинарным отношением $\vdash$, определенным на конфигурациях. Если $\delta(q, a)$ содержит $q`$, то $(q, aw) \vdash (q`, w)$ для всех $w \in V_T^*$. Говорят, что автомат M допускает цепочку w, если $(q_0, w) \vdash^* (q, \varepsilon)$ для некоторого $q \in F$.

Автомат M называется детерминированным, если множество $\delta(q, a)$ содержит не более одного состояния для любых $q \in Q$ и $a \in V_T$.

Чтобы построить конечный автомат, распознающий конструкции языка, необходимо описать допустимые элементы языка. Для этих целей используется аппарат регулярных выражений. Допустимыми конструкциями языка являются слова, числа, знаки препинания, пробелы. Зададим соответствующие регулярные выражения в виде программы для генератора лексических анализаторов – flex (листинг 1).

```
digit   [0-9]
intconst        [+\-]?{digit}+
realconst       [+\-]?{digit}+\.{digit}+(e[+\-]?
{digit}+)?
letter  [а-я]
capital [А-Я]
word  ({letter}|{ capital}){letter}*|{letter}+-{letter}+
abbreviation {capital}*
punctuation {,|:|;|-| (|)|{|}| [|]}
stops {.|!|?}
%%
{intconst}  Процедура формирования лексемы
– целое число;
{realconst}  Процедура формирования лексе-
мы – дробное число;
{word}  Процедура поиска слова в словаре и
формирования лексемы слово;
{abbreviation}  Процедура формирования лек-
семы аббревиатура;
{punctuation}  Процедура формирование лек-
семы – знак препинания;
{stops}  Процедура формирование лексемы –
признак конца предложения;
%%
```

Листинг 1. Программа flex

На основании заданных регулярных выражений flex автоматически сгенерирует конечные автоматы, распознающие соответствующие конструкции языка. Дополнив КА процедурами формирования лексем, получим конечные распознаватели (КР) ЯК.

Задачей КР является распознавание языковой конструкции и формирование лексемы, которая со-

относит распознанную конструкцию к одному из классов.

Наиболее сложным является распознавание слов, при котором КР должен обращаться к словарям.

Алгоритм работы КР для слов.

1. Выделение КА из текста словоформы.
2. Поиск словоформы в словаре.
3. Если словоформа не найдена переход на пункт 10.
4. Извлекается ссылка на комплекс морфологической информации и позиция в семантическом словаре.
5. Формируется множество лексем.
6. Проверяется вхождение словоформы в устойчивое словосочетание.
7. Если словоформа входит в устойчивое словосочетание, то считывается следующая словоформа и проверяется на принадлежность к устойчивому словосочетанию.
8. Если словоформы образуют устойчивое словосочетание, то формируется лексема словосочетания на основе семантической функции.
9. Переход в пункт 1.
10. Применение алгоритма поиска слов с ошибками.
11. Если ошибка найдена, то изменяется словоформа. На основании новой словоформы формируются лексемы.
12. Если ошибка не найдена, то слово считается не распознанным.

Рассмотрим возможные виды лексем, заданных формулой (1), и их признаки.

Для слов класс лексемы задается частью речи. X принимает следующие значения:

С = существительное; П = прилагательное; Г = глагол; Пр = причастие; Д = деепричастие; Н = наречие; Чс = числительное; М = местоимение; Пре = предлог; Со = союз; Ч = частица.

Перечень морфологических признаков:

p_2 – одушевленность; p_3 – род; p_4 – число; p_5 – падеж; p_6 – степень сравнения; p_7 – краткость; p_8 – репрезентация; p_9 – вид; p_{10} – время; p_{11} – лицо; p_{12} – пассивность; p_{13} – разряд местоимений; p_{14} – тип числительных.

Частям речи соответствуют следующие множества морфологических признаков:

$$M(P_{morf})_С = \{p_2, p_3, p_4, p_5\};$$

$$M(P_{morf})_П = \{p_3, p_4, p_5, p_6, p_7\};$$

$$M(P_{morf})_Г = \{p_3, p_4, p_8, p_9, p_{10}, p_{11}, p_{12}\};$$

$$M(P_{morf})_{Пр} = \{p_3, p_4, p_5, p_7, p_9, p_{10}\};$$

$$M(P_{morf})_Д = \{p_9\};$$

$$M(P_{morf})_Н = \{p_6\};$$

$$M(P_{morf})_{Чс} = \{p_3, p_4, p_5, p_{14}\};$$

$$M(P_{morf})_М = \{p_3, p_4, p_5, p_{11}, p_{13}\};$$

Выводы

Результаты данной работы могут быть использованы для реализации блока лексического анализа в лингвистических процессорах. Подход, предложенный в данной работе, позволяет генерировать анализатор автоматическим образом на основе формального описания, что дает возможность проще производить настройку на конкретный естественный язык. Основным преимуществом данного подхода является детальная обработка многозначных слов, что позволяет существенно сократить смысловой недетерминизм для большинства слов и словосочетаний в результирующем формальном представлении.

Список литературы

1. Апресян, Ю.Д.Лингвистический процессор для сложных информационных систем [Текст] / Ю.Д. Апресян, И.М. Богуславский, Л.Л. Иодмин и др. – М.: Наука, 1992. – 256 с.

2. Ахо, А. Компиляторы: принципы, технологии инструменты [Текст] / А. Ахо, Р. Сети, Дж.Ульман – М.: Издательский дом «Вильямс», 2003. – 768 с.

3 Марчук, Ю.Н. Основы компьютерной лингвистики [Текст] / Ю.Н. Марчук – М.: МПУ «Народный учитель», 2000. – 226 с.

4. Льюис, Ф. Теоретические основы проектирования компиляторов [Текст] / Ф. Льюис, Д. Розенкранц, Р. Стирнз – М.: Наука, 1983. – 360 с.

Поступила в редколлегию 30.05.2013

Рецензент: д-р физ.-мат. наук, проф. А.В. Грицунов, Харьковский национальный экономический университет, Харьков.

ПРО ОДИН МЕТОД РЕАЛІЗАЦІЇ ЛІНГВІСТИЧНОГО ПРОЦЕСОРА

Н.А. Валенда, В.М. Ляпота

У статті розглядається метод побудови лінгвістичного процесора на основі кінцевих автоматів і семантичного словника.Розглядаються особливості сумісного застосування формальних методів і засобів для аналізу природних мов.

Ключові слова: *словник, лексема,конечный автомат, семантична функція, лінгвістичний процесор.*

METHOD OF IMPLEMENTATION OF THE LANGUAGE PROCESSOR

N.A. Valenda, V.N. Lyapota

The article considers the method of constructing the language processor based on finite automata and semantic dictionary. The features of joint application of formal methods and facilities are examined for the analysis of human languages.

Keywords: *dictionary, a lexeme is a,конечный automat, semantic function, linguistic processor.*