Инструментальная среда для экспериментов с алгоритмами поверхностно-синтаксического анализа

Баталина А.М., Епифанов М.Е., Ивличева О.О.,
Кобзарева Т.Ю., Лахути Д.Г.

В процессе разработки алгоритмов поверхностно-синтаксического анализа текста важную роль играет возможность проверять различные пробные варианты алгоритмов и порядок их применения. Поскольку эта проверка должна происходить в автоматизированном режиме, она предполагает программную реализацию пробных вариантов разрабатываемой системы алгоритмов. Вносимые в такую систему изменения не должны требовать существенных модификаций в ее реализации. Поэтому необходимо иметь инструмент для наглядного рабочего представления алгоритмов, минимизирующий трудоемкость написания реализующих их программ. В работе предлагается новый подход к разработке инструментальной программной среды для экспериментов с алгоритмами поверхностно-синтаксического анализа. В качестве такой среды здесь выступает некоторое расширение объектной модели многофункциональных словарей, основанной на синтезе лингвистических единиц [Диалог-2003, стр. 223-231]. В такой модели словарей лингвистические единицы, информация о них и отношения между ними представлены в виде объектов, объединенных в динамическую многоссылочную структуру, отвечающую принципу локальности представления данных. Построенные объекты являются конструктором, позволяющим моделировать такие процессы, как словоизменение или синтез слов и различных типов словосочетаний. Применяемая в моделировании словарей объектная библиотека расширена объектами для представления правил поверхностно-синтаксического анализа. В свою очередь, сеть таких объектов, включенных в общую динамическую структуру данных, наглядно представляет совокупность применяемых алгоритмов похожим на блок-схемы образом.

При разработке автоматических систем поверхностно-синтаксического анализа возникает проблема обозримости всей совокупности лингвистических правил, составляющих теоретическую базу для реализации системы.
Для этой задачи характерны ситуации, требующие перебора различных вариантов создаваемой системы правил, т.е. опробования различных формулировок правил в различных сочетаниях (в частности, в разной последовательности) с целью отбора вариантов, обеспечивающих как наилучшее *качество* синтаксического анализа текста, так и наиболее эффективную *организацию* обработки больших массивов текстов.
При построении лингвистического описания уровня, или фрагмента явления некоторого уровня, или универсального правила, используемого на всех уровнях (например, правил согласования или проективности), описание удобно строить как конструктор, состоящий из поверхностно-комбинаторных манифестаций явлений (описания ситуаций, представляющие

собой маски и условия их наложения). Эти ситуации применяются иерархически, что упрощает их структуру и минимизирует условия применимости каждого нижележащего представления явления. В силу обилия представлений линейно-комбинаторных структур (1) изначально можно предусмотреть лишь самые вероятные структуры, (2) трудно чисто умозрительно определить оптимальную последовательность их примерения. Было бы удобно выделить повторяющиеся элементы структур и условий, которые далее можно использовать как «кубики лингвистического конструктора» при пополнении лингвистического базиса ситуаций и при оптимизации порядка их применения.

Кроме того, при расширении круга обрабатываемых текстов появляются новые слова, старые меняют модели управления, меняется структура их денотативной привязки, в связи с чем постоянно возникает потребность в изменении правил поверхностно-синтаксического анализа и после успешного создания эффективной системы правил.

Ясно, что умозрительно совершенствовать довольно обширную систему правил достаточно трудно. Поэтому необходимо иметь инструмент для автоматизации экспериментов с системой правил, обеспечивающий наглядное рабочее представление правил и взаимосвязей между ними, их изменения, моделирования процесса поверхностно-синтаксического анализа для пробных состояний организации этой системы.

В настоящей работе предлагается новый подход к разработке инструментальной программной среды для экспериментов с *алгоритмами* поверхностно-синтаксического анализа. Реализуемая в настоящий момент версия такой среды ориентирована на систему алгоритмов, разрабатываемую Т.Ю.Кобзаревой [1, 2].

В этой системе поверхностно-синтаксический анализ рассматривается как этап, необходимый для подготовки текста к семантическому анализу и решению связанных с ним задач: поиска информации, автоматического реферирования, машинного перевода и др.

Поверхностно-синтаксический анализ понимается как многоуровневая задача грамматического анализа, использующего порядок слов – с приданными им грамматической системой управления (падежами, предлогами, инфинитивом, подчинительным союзом) и некоторыми семантическими классами (предмет, одушевленный предмет, параметр и т.д.) – и функции сочинительных союзов и знаков препинания. Ему предшествуют морфологический анализ (словарный, несловарный для слов продуктивных моделей, отсутствующих в словаре) и постморфологический анализ (система морфо-синтаксических алгоритмов анализа необычных слов, морфологически автономных словосочетаний, разрешения морфологической неоднозначности). Синтаксический анализ в целом состоит из следующих частей: предсинтаксический анализ (формирование некоторых синтагматических связей, необходимых для сегментации предложения), сегментация (установление границ значимых частей предложения), внутрисегментный анализ, межсегментный анализ, построение подграфа отношений кореференции.

Анализ, проводимый в соответствии с каждым алгоритмом (изображаемым как блок-схема), можно представить как дерево поиска ситуаций исследуемого уровня, расположенных удобным образом. На каждом уровне анализа оно представляет собой иерархию процедур, определяющую порядок работы с линейной структурой предложения, то есть поиска признаков в этой структуре «грамматической маски» - набора признаков ситуации – и проверки условий ее применимости.

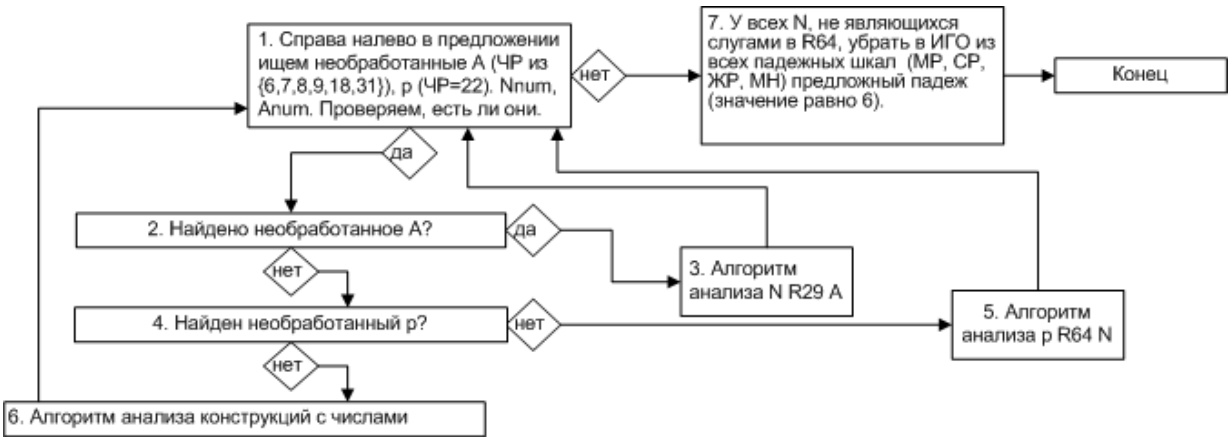

Рис. 1. Распределитель.

В качестве иллюстрации к описанию данной системы алгоритмов можно привести фрагмент алгоритма построения именной группы (рис. 2). Алгоритм, часть которого мы рассмотрим, входит в итеративную процедуру блока трех взаимодействующих алгоритмов анализа именных и предложных групп (N R29 A (именная группа), р R64 N (предложная группа) и конструкций с числительными).

Для того, чтобы пояснить структуру операций, приведем схему-распределитель, определяющую обращение в блоке предсегментации к алгоритму построения именной группы. Взаимодействие алгоритмов этого блока не случайно. Рекурсивность структуры именных и предложных групп (и конструкций с числами, переплетающихся с ними в линейной структуре), является первым показательным случаем, когда мы работаем с многоуровневыми вложениями именных и предложных групп друг в друга с заранее не предсказуемой структурой. Не вдаваясь в лингвистические подробности, приведем схему взаимодействия составляющих блока построения этих групп (рис. 1).

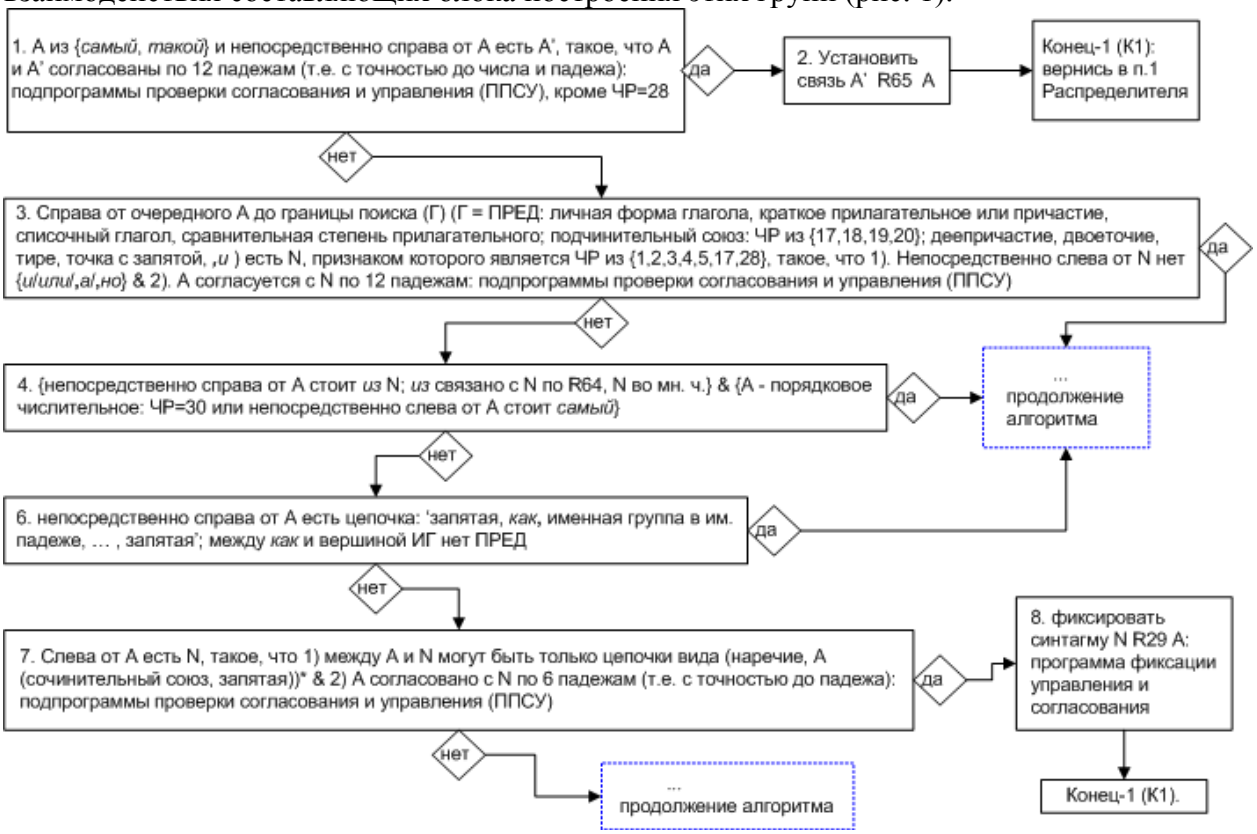

Рис. 2. Алгоритм анализа N R29 A.

Оговорим некоторые обозначения: N – существительное, A – прилагательное/ причастие/ местоименное прилагательное/ несклоняемое прилагательное/ порядковое числительное (тип **A**), p – предлог, Anum – порядковое числительное, Nnum – количественное числительное, ПРЕД - личная форма глагола, краткое прилагательное или причастие, списочный глагол, ЧР – часть речи. Синтаксические связи: **R29** – существительного как хозяина со словом типа **A, R64** – предлога с управляемым им существительным. Элементы в круглых скобках факультативны. Правила записаны в виде условий и действий. Дальнейшие проверки или действия определяются с помощью переходов «да», «нет» блок-схемы алгоритма, выходящих из рассматриваемой вершины.
Рассмотрим пример работы алгоритма, соответствующий данному его фрагменту. В правиле 1 распределителя генерируется первая необработанная лингвистическая единица одного из перечисленных в правиле типов. И если это оказывается A, оно поступает на вход алгоритма N R29 A. В алгоритме N R29 A правило 1 проверяет, есть ли A, такой, что выполняются наложенные условия; если есть – проводится (правило 2) связь определенного типа, если нет – происходит переход к правилу 3. Здесь мы игнорируем альтернативу, при которой A′ связано не с A, а с N – хозяином A (*такой красивый дом*). Правило 3 ищет N, удовлетворяющее накладываемым условиям и, если такого N не находится, происходит переход к правилу 4. Далее происходит переход к правилу вершины 8, в котором происходит фиксация связи, называемой R29. После этого действия работа алгоритма для данного A заканчивается, и начинается поиск следующего «подходящего» A.
Покажем теперь представление алгоритмов поверхностно-синтаксического анализа в предлагаемой нами инструментальной объектной среде.

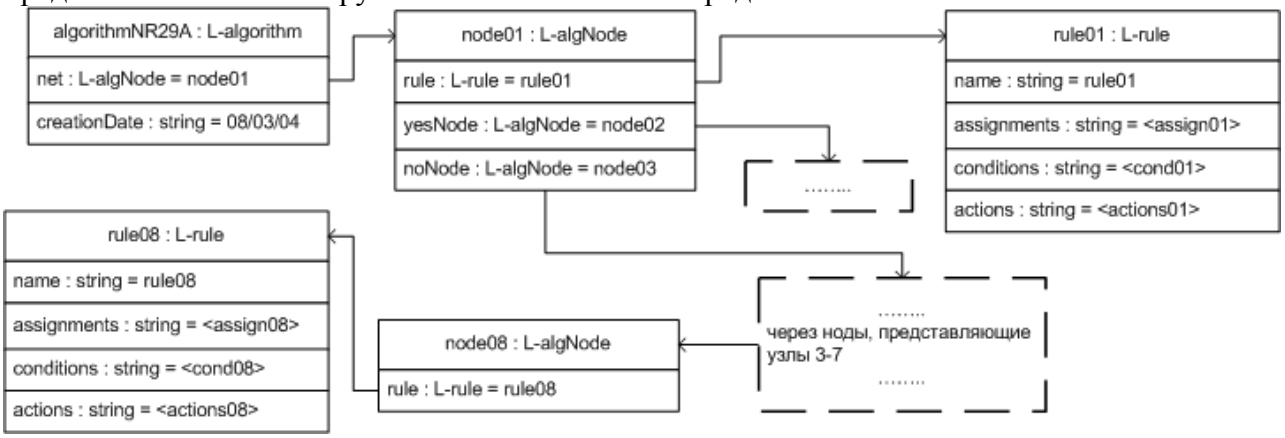

Рис. 3. Представление фрагмента алгоритма рисунка 2 в объектной модели.

Эта модель является развитием объектной модели многофункциональных словарей, основанной на синтезе лингвистических единиц [3]. Последняя реализует идею многоуровневого представления лексических единиц и их свойств как иерархии объектов, объединенных в динамическую многоссылочную структуру. Особенностью такой модели является объектное представление не только самих лингвистических единиц, но и их свойств (то есть, например, свойство «быть в таком-то падеже» обладает своим поведением, на соответствующих уровнях иерархии модели влияющим на синтез одних лингвистических единиц из других).
Для моделирования поверхностно-синтаксического анализа (в рамках подхода Т.Ю.Кобзаревой) добавляются еще две категории объектов: 1) алгоритмы и их узлы 2)

правила, на которые ссылаются узлы алгоритмов. В остальном используются объекты модели многофункциональных словарей.

В описываемой здесь модели под алгоритмом понимается сеть правил. Каждое правило содержит условия, применяемые к анализируемому предложению или некоторой его части и, возможно, операторы (инструкции), инициирующие преобразования в представлении предложения, если условия правила выполняются. После чего, в зависимости от выполнения условий правила, осуществляется переход к одному из двух правил-сыновей, если таковые имеются. Т.о. наглядная форма представления алгоритмов поверхностно-синтаксического анализа, их применение и способы их изменения в данной модели похожи на выбранные автором алгоритмов.

Каждый алгоритм представлен в модели одним объектом класса **L-algorithm**. Этот объект хранит ссылку на сеть узлов алгоритма и набор служебных характеристик, аналогичных свойствам документов (дата открытия, автор, дата последнего изменения и т.д.). В дальнейшем предполагается при самом алгоритме и при всех его узлах сохранять их историю (так называемый «журнал изменений»). Каждая запись в журнале состоит из трех полей: дата изменения, его автор и соответствующий комментарий. Наличие истории особенно важно при коллективной работе над совокупностью алгоритмов.

Алгоритм N_R29_A, фрагмент которого приведен на рис.2, на рис.3 изображается экземпляром объекта с именем algorithmNR29A. Изображение объектов на рис.3 – это упрощение изображения этих объектов в документах MSVisio®, реализующих пользовательский интерфейс нашей объектной модели. Верхний прямоугольник содержит заголовок объекта в формате <имя объекта> : <класс>. В каждом из нижних прямоугольников располагается какое-либо данное объекта в формате <имя> : <тип> = <значение> для простых типов. Если же таким данным является другой объект, то это показывается как <имя> : <класс> = <уникальное имя экземпляра объекта>. При этом от поля «объектного» данного проводится стрелка, указывающая на объект-значение. Мы используем здесь термин «данное объекта» вместо «свойство», «атрибут», «слот», потому что здесь речь идет именно о данных, предоставленных пользователю для редактирования или только просмотра. Вообще говоря, эти данные могут сохраняться некоторым способом, отличным от их визуализации, более того, некоторые из них могут не храниться в объекте, а вычисляться методами доступа. Мы не показываем все данные объектов и обозначаем идентификатором в угловых скобках длинные строковые выражения в объектах, представляющих правила.

Каждый узел алгоритма представляется парой объектов. Собственно узлу соответствует объект класса **L-algNode** со следующими данными: rule - ссылка на представление правила узла - объект класса **L-rule**, yesNode - ссылка на узел, к которому совершается переход при выполнении условий из rule, noNode ссылка на узел, к которому совершается переход при их невыполнении. Т.о. собственно правила могут быть описаны и сохранены отдельно от алгоритмов, и одним правилом могут при необходимости «пользоваться» несколько узлов различных алгоритмов.

Представляющий правило объект класса **L-rule** хранит следующие данные: name - имя правила, assignments - означивания участвующих в правиле переменных, conditions - условия применимости правила, actions - действия, совершаемые при применении правила.

Покажем представление в объектной модели первого и второго правил алгоритма, фрагмент которого приведен на рис.2. Эти правила в нашей модели удобно представить всего одним узлом (хранящим единственное правило) – node01 и rule01 на рис.3. Правило rule01 описывается следующими данными:

```
assign01 = «((:assign A1 (:neighbour A :right 0)))»
cond01 = «((and (:member (:IGOpartOfSpeech A1) '(6 7 8 18 31))
```

```
                (or (eq (:text A) "самый") (eq (:text A) "такой"))
                (:assign a1Ra (:coord12 A A1)) ) )»
actions01 = «((:make L-group :text (:concat (:text A) (:text A1))
                             :children (list A A1)
                             :aData (list (list :numGenderCase a1Ra))
                             :iData ((:IGOconnection R65) ...) ))»
```

Для записи данных используется синтаксис так называемых S-выражений языка программирования Лисп. Это не случайно: реализация объектов модели и связей между ними выполнена на языке Common Lisp с использованием встроенной в него библиотеки CLOS (Common Lisp Object System) в инструментальной среде разработки Corman Common Lisp ® версии 2.5.

S-выражение представляет собой линейный список, понимаемый как вызов функции. Первый элемент в нем – это имя функции, остальные – ее аргументы, которые также могут быть S-выражениями. В показанных здесь S-выражениях участвуют:

- атомы встроенных в Лисп простых типов данных (`0`, `"самый"`);
- символы – в роли 1) имен встроенных (в язык) или определенных в модели функций (здесь – предикаты: `and`, `or` – булевы связки, `member` – проверяет, содержится ли первый аргумент в списке-значении второго аргумента, `eq` – проверяет, равны ли значения аргументов; функция `list`, строящая список из значений аргументов), 2) переменных (`A`, `A-right`, `a1Ra`), 3) данных (имя класса `L-group`);
- ключевые слова (начинаются с двоеточия).

На последних остановимся подробнее. В Лиспе ключевые слова не заносятся в таблицу символов (т.е. не «загромождают» память) и обычно используются либо в роли данных – как флаги, моды и т.п., либо при передаче аргументов функции в формате (… <ключ формального параметра> <значение> …). Подобным образом ключевые слова используются и в описании правил:`::right` указывает направление выбора соседа в предложении, `:text`, `:children` и `:aData` – это ключи полей данных в конструкторе класса `L-group`. Кроме того, мы дополнительно используем ключи в начале списков для именования форм, которые при препроцессинге правил преобразуются в обычные S-выражения Лиспа с соответствующими форме обращением к функции и набором аргументов. Конечно, такое смешение символов и ключевых слов в «одинаковых ролях» не украшает синтаксис правил, однако без него выражения в правилах станут более громоздкими.

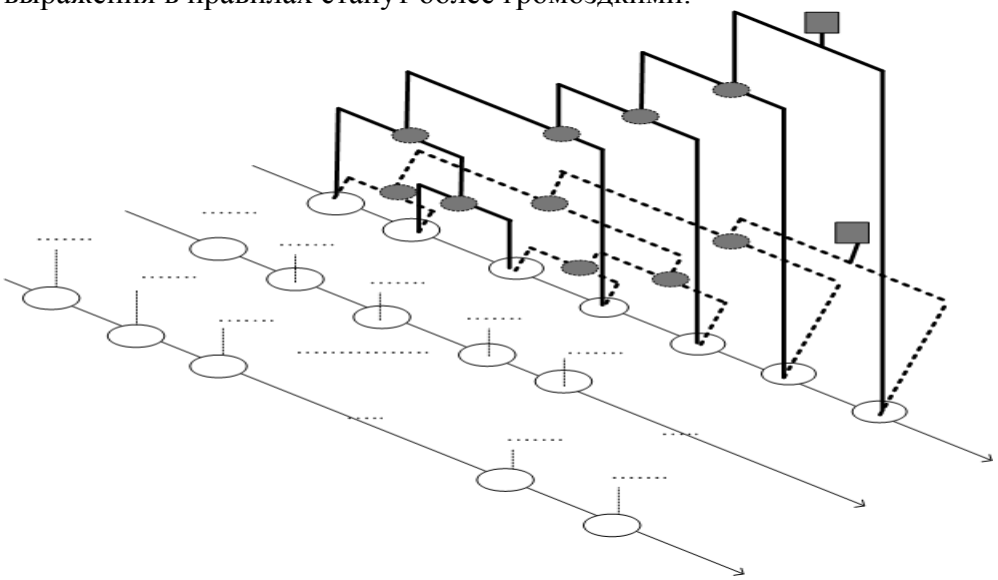

Рис. 4. Пространство альтернатив вычислительной модели.

Прежде чем объяснить «обработку» данного правила в рамках алгоритма, неформально опишем используемую для этого вычислительную модель.
Правила применяются к некоторой конкретной цепочке объектов, представляющих лингвистические единицы со связями между ними – *альтернативе* анализируемого предложения. Принципом организации пространства альтернатив в описываемой модели (рис. 4) является изначальная общность «происхождения» и полное последующее разделение возможных альтернатив. Из предложения с точностью до неоднозначностей (например, омонимии) на стадии морфологического анализа выделяется исходное множество альтернатив. В результате применения правил поверхностно-синтаксического анализа над каждой исходной альтернативой формируется множество возможных вариантов организации синтаксических связей внутри нее. Синтаксические связи представляются в виде объединения сначала отдельных лексем, а затем и более крупных единиц, в единицы, обладающие некоторыми расширенными общими свойствами – группы. Свойства каждой группы формируются на основе свойств входящих в нее элементов. (В рассматриваемом примере правила rule01 выражение `actions01` как раз и определяет построение объекта для группы из двух слов: «самый» либо «такой» и значения переменной `A`. Объект сохранит свойства такой группы и конкатенацию этих слов в качестве текста.)
Таким образом каждый такой вариант связывания лексем определяет новую, производную от исходной, альтернативу. В любой момент вычисления можно говорить о *состоянии альтернативы* как о полученном к этому моменту в результате анализа предложения (в данной альтернативе) знании о синтаксических связях между его членами. Состояние альтернативы – это граф, в котором листьями являются лексемы исходной альтернативы, а внутренними вершинами – лингвистические единицы, сформированные в результате применения правил поверхностно-синтаксического анализа. В процессе применения правил альтернативы могут как добавляться, так и отвергаться. Множества внутренних вершин графов альтернатив над общей исходной не пересекаются (при необходимости выполняется копирование), а графы производных от разных исходных альтернатив - не пересекаются вовсе.
Важно заметить, что часто при осуществлении поверхностно-синтаксического анализа необходимо не только связывать элементы предложения в некоторые общности, но и «заглядывать» внутрь уже построенных групп. В данной модели предусмотрена возможность достижения из любой вершины всех вершин, связанных с ее происхождением. У объекта, представляющего группу, хранятся ссылки на те объекты, которые являются его родителями и детьми. Соответствующие формы в записи правил позволяют осуществлять доступ к свойствам предка от потомков и наоборот, а также, при желании, заменять в анализируемой цепочке объектов детей их родителем и обратно.
Значениями переменных в описании правила являются как сами объекты анализируемой цепочки, так и все, что может быть вычислено из свойств этих объектов. Сокращая описание доступа к объектам, форма `:assign` служит для присваивания своему первому аргументу (переменной) значения второго аргумента. В нашем случае переменной `A1` присваивается объект, являющийся в анализируемой цепочке непосредственным соседом справа объекта – значения переменной `A`. Сама же переменная `A` приходит в алгоритм, представленный объектом algorithmNR29A, из объемлющего алгоритма-распределителя (рис.1) уже «означенной» некоторым еще не обработанным словом типа **A**. Причем в реализации правила 1 алгоритма-распределителя применяется форма `:enumerate`, в отличие от формы `:assign` перечисляющая при каждом обращении к правилу все объекты, удовлетворяющие заданному условию.
В Лиспе все, что не «ложь» (атом NIL – «пусто»), то «истина». Поэтому формы, соответствующие операторам, мы тоже можем рассматривать как предикаты и использовать

их внутри условий (как в нашем примере – присваивание переменной a1Ra результата проверки согласования по 12 падежам, а именно – множества троек «род, число, падеж», по которым представленные объектами лингвистические единицы действительно согласуются). Отсюда следует, что разделения условий правила на два поля – assignments и conditions – можно было и не делать. Однако вынесение связывания переменных в отдельное поле делает применение правила более эффективным. Успешность связывания всех переменных в группе assignments (если ни одна из форм не вернула в качестве значения NIL) является необходимым условием применимости правила. С целью безопасного использования переменных применение правил выполняется в динамическом контексте (scope).
Количество системно-интерпретируемых форм (начинающихся с ключевого слова) на каждой стадии развития системы ограничено. Однако ко всем свойствам объектов (в частности, объектов новых классов, которыми со временем может пополняться система) можно обратиться универсальным образом через соответствующие системно определенные функции. Вообще, в этом смысле в выражениях правил можно использовать все функциональное богатство Лиспа.
Эксперименты по моделированию алгоритмов поверхностно-синтаксического анализа показали удобство и наглядность представленной в данной работе инструментальной объектной среды. В настоящее время проводится дальнейшая ее реализация в таких направлениях, как пошаговое выполнение правил (аналог debug в инструментальных средах разработки в программировании), создание удобного интерфейса пользователя.

Список литературы

1. Кобзарева Т.Ю., Лахути Д.Г., Ножов И.М. Сегментация русского предложения. // Труды конференции. Седьмая национальная конференция по искусственному интеллекту с международным участием. КИИ' 2000 Москва. Издательство Физико-математической литературы. 2000.с.879-880.
2. Кобзарева Т.Ю., Лахути Д.Г., Ножов И.М. Модель сегментации русского предложения. // Диалог'2001. Аксаково 2001. т.2 с.185-194.
3. Ивличева О.О., Епифанов М.Е., Лахути Д.Г. Объектная модель многофункциональных словарей, основанная на синтезе лингвистических единиц // Компьютерная лингвистика и интеллектуальные технологии: Тр. Междунар. конференции Диалог'2003 (Протвино, 11-16 июня 2003 г.), стр. 223-231
4. Буч Г. Объектно-ориентированный анализ и проектирование. М.: Издательство Бином, 2000.
5. Graham, Paul. ANSI Common Lisp. Prentice-Hall, New Jersey (USA), 1996
6. Corman Lisp© Version 2.5 User Guide, 2003