РАЗРАБОТКА ПАРСЕРА И МОДУЛЯ УНИФИКАЦИИ ДЛЯ СИНТАКСИЧЕСКОГО ПРОЦЕССОРА

DEVELOPING A PARSING ENGINE AND A UNIFICATION MODULE FOR A SYNTACTIC PROCESSOR

А. А. Перекрестенко
A.Perekrestenko@gmx.net
Институт проблем информатики РАН

В рамках работы по созданию системы автоматического синтаксического анализа автором был реализован парсер, позволяющий анализировать структуры естественного языка, описываемые более мощными формализмами, чем КС-грамматики, и базовый компонент унификационного модуля.

В рамках работы по созданию системы автоматического синтаксического анализа автором был разработан и реализован парсер (parsing engine), работающий с грамматикой, относящейся к классу мягко контекстно-зависимых формализмов, а также базовый модуль унификационного компонента, предназначенного для представления и анализа морфологических и функциональных характеристик синтаксических единиц. Парсер работает со структурой составляющих (c-структурой в терминологии лексико-функциональной грамматики 4)), а унификационный компонент с функциональной структурой (f-структурой). Парсер позволяет работать со структурами более сложными, чем те, которые могут быть описаны КС-грамматиками, т. е. с самого начала предполагается возможность более сложного устройства структуры синтаксических составляющих, что позволяет компактно и лингвистически неконтринтуитивно описывать структуру предложений в языках со свободным порядком слов.

Парсер

Парсер реализован как парсер Ерли (Early parser), расширенный функциями для представления разрывных составляющих, эллипсиса, нелокальных связей и т. п. Есть также экспериментальная реализация данного модуля в виде рекурсивного нисходящего LR-парсера с предсказательной эвристикой для решения проблемы левой рекурсии и итерации связанной пустой категории. В основу нотации записи правил положена расширенная форма Бэкуса-Наура (EBNF), дополненная различными элементами для представления нелокальных связей. Помимо структур, описываемых КС-грамматиками, парсер позволяет анализировать, в частности, синтаксические явления, приводимые ниже.

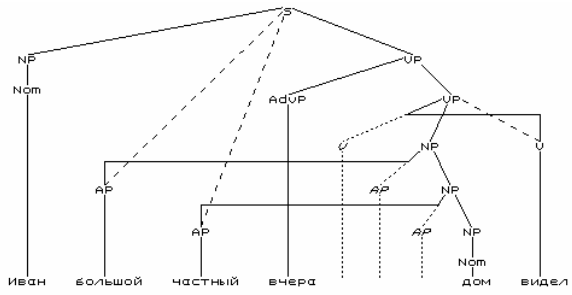
Разрывные составляющие

Для представления разрывных составляющих используется аппарат смещённых категорий, аналогичный применяемому в универсальной грамматике 5).

Пример анализа разрывных составляющих может быть проиллюстрирован на примере предложения *Иван большой частный вчера дом видел*, структура которого может быть описана следующим множеством правил:

```
<S>::="<NP><$AP _traces=":NR">*<VP>",
<VP>::="<AdvP><VP>|(<V>|<_V>)<NP><$V>?",
<NP>::="(<AP>|<_AP>)<NP>|<Nom>",
<AP>::=" *(частный|большой)",
<AdvP>::=" *вчера",
<Nom>::=" *(дом|Иван)",
<V>::=" *видел".
```

Результат парсинга со следами смещённых категорий выглядит следующим образом[1]:



В правилах указывается место порождения категории и то место, куда она была смещена, при этом возможны несколько опций линейного поиска. Этих средств, естественно, недостаточно для описания, в частности, запретов на некоторые типы перемещений. Модуль, описывающий этот аспект синтаксиса, будет реализован позднее.

Данный парсер не является законченным модулем автоматического синтаксического анализа, а представляет собой лишь базовый компонент системы, работающий с чисто абстрактным материалом, в данном случае с символами (т. е. с буквами). В дальнейшем, когда будет полностью реализован унификационный модуль, парсер будет

[1] Все графические представления порождены модулем визуализации, встроенным в парсер.

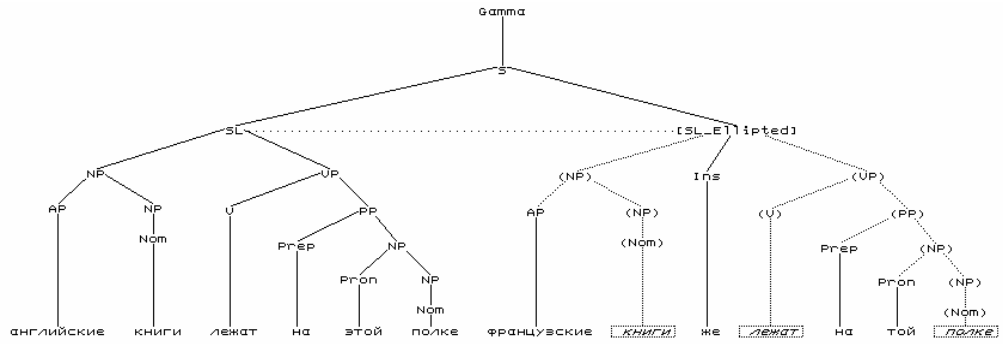
переведён на терминальные символы, соответствующие лексемам естественного языка.

Эллипсис

Эллипсис также относится к классу явлений, не описываемых в рамках КС-грамматик. В парсере проблема представления эллипсиса решается через импорт структуры неэллептированной составляющей в составляющую с эллиптированными фрагментами в сочетании с восстановлением этих фрагментов. Для этого сначала устанавливается нелокальная связь между составляющей, содержащей эллептированные фрагменты, с её неё неэллептированным прототипом, после чего определяется то, как структура будет импортироваться, для чего устанавливаются нелокальные связи между существующими фрагментами усечённой составляющей и некоторыми частями составляющей, из которого импортируется структура. В качестве примера приведём анализ предложения *английские книги лежат на этой полке, французские же на той*. Структура данного предложения описывается следующим множеством правил:

```
<Gamma>::="<S _comb>",
<S>::="<SL>,<SL_Ellipted _importstruct="SL">",
<SL>::="<NP><Dummy _hide><VP>",
<VP>::="<V><PP>",
<SL_Ellipted>::="<AP><Ins _elref="Dummy"><Prep><Pron>",
<PP>::="<Prep><NP>",
<NP>::="<Nom>|(<AP>|<Pron>)<NP>",
<AP>::=" *(английские|французские)",
<Nom>::=" *(книги|полке)",
<Pron>::=" *(этой|той)",
<Prep>::=" *на",
<V>::=" *лежат",
<Ins>::=" *же",
<Dummy>::=.
```

Результат парсинга выглядит следующим образом[2]:



[2] Необходимость фиктивного узла Gamma, доминирующего над собственно предложением S, обусловлена в данном случае конструкционными особенностями парсера Ерли. Никакой лингвистической нагрузки этот символ не несёт.

В данной версии парсера, однако, пока не реализован импорт структуры предложений, в составе которых имеются смещённые категории.

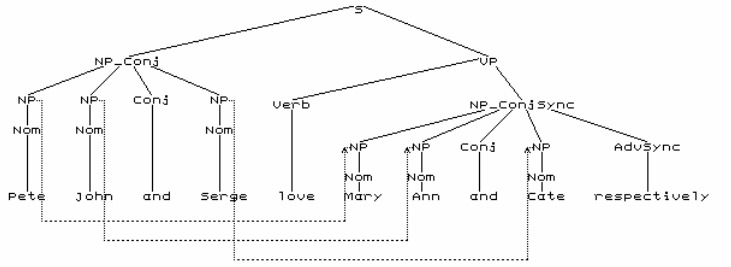
На уровне описания категория, в которой имел место эллипсис, отличается от регулярной грамматической конструкции плоской структурой, в которой учитывается только линейное расположение присутствующих в составе данной категории неэллиптированных элементов, в сочетании с наличием инструкций об импорте структуры (**_importstruct** и **_elref**). В приводимом примере это последовательность **<AP><Ins><Prep><Pron>**, описывающая последовательность составляющих эллиптированного предложения *…французские же на той*. Регулярные грамматические конструкции не могут содержать инструкций об импорте структуры.

Синхронизация итерации

Синхронизация итерации также являет проблемным моментом при описании синтаксиса естественных языков при помощи КС-грамматик. В парсере она может быть описана через аппарат нелокальных связей. В качестве иллюстрации приведём пример анализа предложения англ. *Pete, John and Serge love Mary, Ann and Cate respectively*. Структура данного предложения может быть описана следующим множеством правил:

```
<S>::="<NP_Conj _comb><VP>",
<VP>::="<Verb><NP_ConjSync>",
<NP_Conj>::="<NP.1 _comb>+<Conj><NP.2>",
<NP_ConjSync>::="<NP _comb, _setrelfrom=".1:K">+<Conj><NP _setrelfrom=".2:K"><AdvSync>",
<NP>::="<Nom>,?",
<Nom>::=" *(Pete|John|Serge|Mary|Ann|Cate)",
<Verb>::=" *love",
<AdvSync>::=" *respectively",
<Conj>::=" *and".
```

Применение данного множества правил к предложению даёт следующую структуру на выходе:



Техническая реализация парсера

Парсер, включающий в себя модуль визуализации[3] для наглядного представления результатов анализа, полностью реализован на языке программирования C++ платформенно независимым образом. Парсер выполнен без привязки к конкретному типу анализируемых линейных объектов (т. е. цепочек терминальных символов, подаваемых на вход) в виде шаблона классов, в терминах C++. Это позволяет настроить его для работы с любыми линейными объектами, заменяя подставляемый класс, описывающий интересующие нас объекты. В приведённых выше примерах был задействован символьный вариант парсера, для работы с реальными лексическими единицами будет создан соответствующий подставляемый класс, описывающий лексемы естественного языка.

В динамических операциях, требующих регулярного выделения и освобождения памяти, используется собственный менеджер памяти парсера, что позволяет уменьшить зависимость эффективности работы парсера от реализации функции выделения памяти операционной системы. Помимо этого, таким образом проще отслеживать в программе ошибки, связанные с невысвобождением ранее выделенной памяти или с попытками незаконного использования ранее аннулированных объектов.

В парсере реализована система анализа правил, позволяющая отслеживать различные некорректности в записи правил, в частности, синтаксические ошибки (как, например, пропущенные скобки, вхождение неизвестных нетерминальных символов в правой части правила и т. п.), а также производить анализ множества правил на предмет наличия содержательных ошибок – бесконечной рекурсии, в т. ч. косвенной, и применения неограниченного итератора (т. е. без верхней границы количества итераций) по несвязанным пустым категориям[4].

[3] Модуль визуализации генерирует представление результатов анализа в виде Bitmap-графики.

[4] В настоящий момент блокировка применения неограниченного итератора по *несвязанным* пустым категориям реализована только в экспериментальной рекурсивной версии парсера. В парсере Ерли блокируется применения неограниченного итератора по любым пустым категориям.

Помимо указанных элементов, в парсере также реализованы несколько вспомогательных функций, осуществляющих различные операции с нетерминальными узлами. Есть также функция внутренней регистрации терминальных символов. Данные вспомогательные элементы задействуются при использовании парсера для порождения первичных унификационных структур.

Модуль унификации

Модуль унификации, предназначенный для работы с функциональной структурой предложения и морфологическим согласованием синтаксических единиц, реализуется аналогично унификатору HPSG 7), 8). Нотация записи правил унификатора в основном повторяет нотацию PATR 9), принятую в HPSG, при этом допускается использование дизъюнкции как для простых, так и для сложных значений. Характеристики синтаксических единиц записываются в виде заключённой в квадратные скобки матрицы параметров, представляющей собой множество пар вида *Параметр = Значение*, где *Значение* может быть либо атомом, как, например, **Sg** (единственное число) или **Dat** (датив), либо само быть матрицей параметров, как, например, в случае **Agr=[Gen=M, Num=Sg, Case=Dat]**. Значение также может быть объявлено общим для нескольких параметров (structure sharing). Для этого используется ссылка на значение, которая записывается как *@X*, где *X* – последовательность букв и/или цифр. Так, тот факт, что лицо и число глагольной группы – это лицо и число личной формы глагола, находящейся в вершине данной группы, может быть задан следующим образом:

```
[SyntClass=VP, Agr=@1, Head=[SyntClass=VerbPer, Agr=@1]]
```

Циклические описания вида [A1=@1[B=@2], A2=@2[C=@1]] не допускаются.

Произвольное значение записывается как **@**. Например, тот факт, словоформа *пальто* может иметь любые число и падеж, может быть записан в матрице параметров как **Agr=@**. Однако в данной версии унификатора не реализована механизм представления типов значений, что в данном примере означает, что параметр **Agr** может интерпретироваться как имеющий действительно любое (даже необязательно лингвистически осмысленное) непротиворечивое значение, что требует осторожности при использовании данной конструкции. В дальнейшем планируется ввести типизацию значений.

Помимо значений указанных типов в унификаторе поддерживаются также списочные значения. Так, тот факт, что некоторый глагол требует два дополнения – прямое (а аккузативе) и косвенное (в дативе) – может быть задан в матрице параметров глагола следующим образом:

```
Comps=<[SyntClass=NP, Agr=[Case=Acc]], [SyntClass=NP, Agr=[Case=Dat]]>
```

Списочное значение, также как и значения типа атом и матрица, может быть объявлено общим для нескольких параметров.

Допускается дизъюнктивная запись значений. Так, например, факт морфологической омонимии словоформы *большой* может быть отражён в записи посредством дизъюнкции (логическое „или“ записывается как вертикальная черта):

```
Agr=([Num=Sg, Gen=M, Case=Nom]|[Num=Sg, Gen=F, Case=(Dat|Instr|Prp|Gen)])
```

Использование дизъюнкции не накладывает никаких ограничений на использование ссылок (помимо общего запрета циклов). Одноимённые ссылки, содержащиеся в разных дизъюнктах, интерпретируются как разные ссылки. В процессе унификации структуры, содержащие одноимённые ссылки в разных дизъюнктах, подвергаются нормализации (включающей в себя расщепление структуры, если ссылка данного типа «ведёт» за пределы дизъюнкции). Первичные структуры подобного рода нормализуются на стадии инициализации.

В настоящее время ведётся работа по реализации модуля унификации списков, записанных с применением дизъюнкции и содержащих не(до)определённые фрагменты. Пример унификации спискам подобного типа, осуществляемый данным модулем:

Структура 1	`[A=a+(<b, c>|@1)+d, B=@1]`
Структура 2	`[A=<a, b>+(c|q)+d]`

Результат унификации	`([A=a+b+c+d, B=@]|`
	`[A=a+@2<b, c>+d, B=@2]|`
	`[A=a+@1<b, q>+d, B=@1])`

Унификационный модуль реализован по конструктивной модели, т. е. он работает с сохранением исходных структур. Начальные структуры, по которым осуществляется унификация, порождаются при помощи символьной версии парсера. Парсер, таким образом, используется также и как модуль инициализации унификатора. Так же как и парсер, унификатор реализуется без привязки к конкретным операционным системам и с использованием собственного менеджера памяти.

Список литературы

1) Перекрестенко, А. Об автоматическом синтаксическом анализе в некоторых классах контекстно-зависимых языков. // Московский лингвистический журнал, том 6, №2. Москва, 2003.

2) Перекрестенко, А. Создание парсера для некоторых классов контекстно-зависимых языков для задач автоматического

синтаксического анализа. // Сборник трудов конференции Диалог 2003.

3) Перекрестенко, А. Разработка системы автоматического синтаксического анализа на основе мягко контекстно-зависимой унификационной грамматики. // Компьютерная лингвистика и интеллектуальные технологии. Сборник трудов конференции Диалог 2004.

4) Bresnan, J. (ed.) The mental representation of grammatical relations. MIT Press, 1982.

5) Chomsky, N., Lasnik, H. The Theory of Principles and Parameters. // The Minimalist Program, 1995.

6) Joshi, A.K., Schabes, Y. Tree Adjoining Grammars. // Handbook of Formal Languages, 1997, pp. 69-123.

7) Pollard, C., Sag, I. Head Driven Phrase Structure Grammar. University of Chicago Press, 1994.

8) Sag, I., Wasow, Th. Synactic Theory: a formal introduction. SCLI. 1997.

9) Shieber, S. M. An introduction to unification-based approaches to grammar. University of Chicago Press, Chicago, 1986.