# Разработка системы автоматического синтаксического анализа на основе мягко контекстно-зависимой унификационной грамматики

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

Разрабатываемая автором система автоматического синтаксического анализа предназначена для описания и анализа синтаксических структур при помощи мягко контекстно-зависимой унификационной грамматики. Анализ в данной версии системы осуществляется на уровне выделения составляющих и определения функциональной структуры предложения. Система включает в себя парсер, работающий с правилами структуры составляющих, и позволяющий, в частности, осуществлять анализ структур естественного языка, описываемых более мощными формализмами, чем КС-грамматики, как, например, разрывные составляющие и эллипсис. Парсер дополнен унификационным модулем, выполняющим анализ функциональной структуры предложения и определение морфологических параметров составляющих. Правила, описывающие функциональную структуру предложения и согласование морфологических параметров синтаксических единиц, записываются в нотации, в основу которой положена нотация HPSG. Для удобства и компактности записи в правилах допускается использование дизъюнкции. Разрабатывается также представление отрицания и гибкая система описания сложных значений списочного типа. Система реализована на языке программирования C++ без привязки к конкретным операционным системам.

В моделирование синтаксиса естественных языков для задач автоматического синтаксического анализа наиболее распространёнными являются формальные системы, относящиеся к одному из следующих классов:

- *Регулярные грамматики* – используются только для упрощенного анализа (shallow parsing), в общем случае не могут быть задействованы в качестве базового формализма системы полноценного синтаксического парсинга.
- *Контекстно-свободные грамматики* – используются достаточно широко, однако в чистом виде также непригодны для полноценного анализа, так как их мощности недостаточно для работы с синтаксическими явлениями, характеризующимися наличием нелокальных связей, как, например, разрывные составляющие и эллипсис[1].

[1] Тем не менее, большинство синтаксических структур естественного языка могут быть описаны при помощи КС-грамматики в слабом смысле, т.е. без приписывания им (корректных) структурных дескрипций, что в сочетании с относительно невысокой временной сложностью парсинга в языках данного типа ($n^3$, где $n$ – длина предложения)

- *Мягко контекстно-зависимые грамматики* – будучи несколько более мощными, чем КС-грамматики, грамматики данного класса позволяют анализировать большинство типов синтаксически релевантных нелокальных связей. Наиболее распространенными формальными системами данного класса являются древоприсоединительные грамматики [Joshi 1997].

В рамках работы по созданию системы автоматического синтаксического анализа автором был разработан парсер (parsing engine), работающий с формализмом, относящимся к классу мягко контекстно-зависимых грамматик, позволяющий, в частности, анализировать разрывные составляющие и эллипсис, а также реализован базовый модуль унификационного компонента, предназначенного для представления и анализа морфологических и функциональных характеристик синтаксических структур. Парсер работает со структурой синтаксических составляющих (c-структурой в терминологии лексико-функциональной грамматики), а унификационный компонент с функциональной структурой (f-структурой). Таким образом, в грамматике разрабатываемой системы используется представление синтаксических структур, аналогичное принятому в лексико-функциональной грамматике [Bresnan 1982], при этом парсерный компонент позволяет работать со структурами более сложными, чем те, которые могут быть описаны КС-грамматиками, т.е. с самого начала предполагается возможность более сложного устройства c-структуры, что позволяет компактно описывать структуру составляющих языков со свободным порядком слов. И парсер, и унификационный модуль реализуются на языке C++ без привязки к конкретным операционным системам.

## Парсер

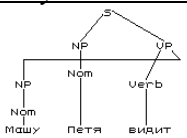
Парсер реализован как парсер Ерли, расширенный функциями для представления разрывных составляющих, эллипсиса, нелокальных связей и т.п. Есть также экспериментальная реализация данного модуля в виде рекурсивного нисходящего LR-парсера с предсказательной эвристикой для решения проблемы левой рекурсии и итерации пустой категории. Нотация записи правил приводится в [Перекрестенко 2003]. Помимо структур, описываемых КС-грамматиками, парсер позволяет анализировать, в частности, следующие синтаксические явления (графическое представление сгенерировано модулем визуализации, встроенным в парсер):

*Разрывные составляющие*. Для представления разрывных составляющих используется аппарат смещённых категорий [Chomsky 1995]. Так, предложения *Машу Петя видит* может быть описано следующим множеством правил:

*Правила:*

```
<S>::="<$NP><NP><VP>",
<VP>::="<Verb><_NP>",
<NP>::="<Nom>",
<Nom>::=" *Петя| *Машу",
<Verb>::=" *видит"
```

*Результат анализа:*




Более подробное описание представления разрывных составляющих в парсере, включая пример с представлением грамматически обусловленной непроективности, см. в [Перекрестенко 2003].

*Эллипсис*. Парсер позволяет описывать различные виды эллипсиса, в том числе эллипсис со вставкой. Так, предложение *Ваня пьёт тёмное пиво, Петя же светлое* может быть описано и проанализировано следующим образом:

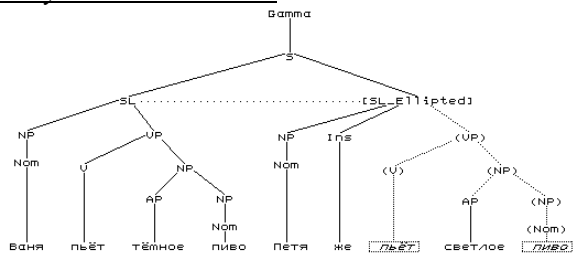
---

имело следствием то, что данные грамматики в той или иной форме легли в основу ряда синтаксических теорий, ориентированных на автоматическую обработку языка.

*Правила:*

```
<Gamma>::="<S _comb>",
<S>::="<SL>,<SL_Ellipted _importstruct="SL">",
<SL>::="<NP><Dummy _hide><VP>",
<VP>::="<V><NP>",
<SL_Ellipted>::="<NP><Ins _elref="Dummy"><AP>",
<NP>::="<Nom>|<AP><NP>",
<AP>::=" *(светлое|тёмное)",
<Nom>::=" *(Ваня|Петя|пиво)",
<V>::=" *пьёт",
<Ins>::=" *же",
<Dummy>::=.
```

*Результат анализа:*



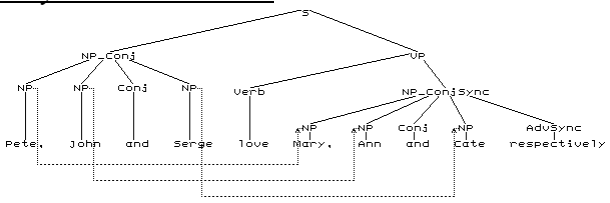

Эллипсис восстанавливается при помощи вычисления нелокальной связи в сочетании с импортом структуры параллельной неэллиптированной категории. В парсере реализован достаточно сложный аппарат представления эллипсиса, позволяющий вводить различные варианты разрешения эллипсиса через восстановление эллиптированных категорий. (Функция `_importstruct` в правилах импортирует в данную категорию структуру другой категории.)

*Синхронизация повторения*. В версии парсера, описанной в вышеупомянутой работе автора, реализована только синхронизация итераторов. В настоящей версии она дополнена синхронизацией с экспликацией нелокальной связи. Так, предложение англ. *Pete, John and Serge love Mary, Ann and Cate respectively* может описано и проанализировано следующим образом:

*Правила:*

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

*Результат анализа:*




## Модуль унификации

Модуль унификации, работающий с функциональной структурой и морфологическим согласованием синтаксических единиц, реализуется аналогично унификатору HPSG [Pollard 1994, Sag 1997]. Нотация записи правил унификатора основана на нотации, принятой в HPSG, при этом допускается использование дизъюнкции как для простых, так и для сложных значений. Характеристики синтаксической единицы записываются в виде матрицы параметров, представляющий собой множество пар вида *Параметр = Значение*, заключённое в квадратные скобки, 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, которая в данном унификаторе записывается как *@X*, где *X* – любая 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 (частный случай реализации Head-Feature Principle в HPSG):

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

Данная запись означает, что значение согласовательного параметра **Agr** глагольной группы (VP) общее со значением этого же параметра вершины группы – личной формы глагола, при этом эксплицитно не указывается, какое именно это значение. В более конкретном случае, например, если у нас есть группа глагола в 3-м лице единственном числе, характеристика этой группы будет выглядеть так:

```
[SyntClass=VP, Agr=@1, Head=[SyntClass=Verb, Agr=@1[Per=3, Num=Sg]]]
```

Здесь непринципиально, при каком из параметров **Agr** записано значение, а при каком стоит только метка. Т.е. матрица параметров представляет собой направленный ациклический граф, как и в HPSG. Циклические ссылки вида `[A1=@1[B=@2], A2=@2[C=@1]]` не допускаются.
Произвольное значение, если не требуется его согласования с чем-то ещё, записывается как "`@`". Например, тот факт, словоформа *пальто* может иметь любые число и падеж, может быть записан в матрице параметров как `Agr=@`. Однако данной версии унификатора не реализован механизм представления типов значений, что в данном примере означает, что параметр **Agr** может быть проинтерпретирован как имеющий действительно *любое* (даже необязательно лингвистически осмысленное) непротиворечивое значение, что требует осторожности при использовании данной конструкции. В следующей версии унификационного модуля 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 описан 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 (логическая связка "или" записывается как вертикальная черта):

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

Использование дизъюнкции не накладывает никаких ограничений на использование ссылок (помимо общего запрета циклов). Однако следует учитывать, что одноимённые ссылки, содержащиеся в разных дизъюнктах, интерпретируются как *разные* ссылки. Так, в структуре

```
[A=([X1=@1 x, X2=@1]|[Y1=@1, Y2=@1])]
```

общими являются значения параметров $X_1$, $X_2$ первого дизъюнкта и $Y_1$, $Y_2$ второго, значения же параметров X и Y, принадлежащих разным дизъюнктам одной и той же дизъюнкции, между собой общими не являются. Т.е. указанная выше запись эквивалентна структуре

```
[A=([X1=@1 x, X2=@1]|[Y1=@2, Y2=@2])],
```

где метки эксплицитно записаны как разные. Аналогично рассматриваются структуры, где имеются одноимённые метки в разных дизъюнктах одной и той же дизъюнкции и за пределами этой дизъюнкции. Так, запись

```
[A=([X=@1 x]|[Y=@1]), B=@1]
```

эквивалентна структуре

```
[A=[X=@1 x], B=@1]|[A=[Y=@2], B=@2]]
```

Работа унификатора со структурами, содержащими дизъюнкции и ссылки, может быть проиллюстрирована следующими абстрактными примерами:

Пример 1

*Значение 1:* `[A=[X=@1], B=[Y=@1]]`

*Значение 2:* `[A=([X=u]|[X=v]), B=([Y=u]|[Y=v])]`

*Результат унификации:* `([A=[X=@2], B=[Y=@2 u]]|[A=[X=@1], B=[Y=@1 v]])`

Пример 2

*Значение 1:* `[A=(x|@1 y), B=(@1|z)]`

*Значение 2:* `[B=@1, C=@1]`

*Результат унификации:* `([A=x, B=@4(@|z), C=@4]|[A=y, B=@3 z, C=@3]|[A=y, B=@1, C=@1])`

В настоящий момент разрабатывается представления логической операции отрицания, а также гибкое представление сложных списочных структур, которое позволит компактно описывать объединение списков, объявлять подсписки общими для нескольких списков и использовать дизъюнкцию в описании списков.

## Технические аспекты

Парсер выполнен без привязки к конкретному типу анализируемых линейных объектов. То, с какими объектами он работает, определяется классом, описывающим терминальные символы[2]. В приведённых выше примерах был задействован символьный вариант парсера, для работы с реальными лексическими единицами будет создан соответствующий класс. 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.

## Литература


1. [Перекрестенко 2003] Перекрестенко, А.: Создание парсера для некоторых классов контекстно-зависимых языков для задач автоматического синтаксического анализа. Сборник трудов конференции Диалог 2003.
2. [Bresnan 1982] Bresnan, J. (ed.) The mental representation of grammatical relations. MIT Press, 1982.
3. [Chomsky 1995] Chomsky, N., Lasnik, H.: The Theory of Principles and Parameters. // The Minimalist Program, 1995.
4. [Joshi 1997] Joshi, A. K., Schabes, Y.: Thee Adjoining Grammars. // Handbook of Formal Languages, 1997, pp. 69-123.
5. [Pollard 1994] Pollard, C., Sag, I.: Head Driven Phrase Structure Grammar. University of Chicago Press, 1994.
6. [Sag 1997] Sag, I., Wasow, Th.: Synactic Theory: a formal introduction. SCLI. 1997.


[2] Т.е. класс, реализующий сам парсер, выполнен в виде шаблона (в терминах C++). Это позволяет настроить его для работы с любыми линейными объектами, заменяя подставляемый класс, описывающий интересующие нас объекты.