Создание парсера для некоторых классов контекстно-зависимых языков для задач автоматического синтаксического анализа

Александр Перекрестенко

Грамматику, описывающую синтаксис какого-либо языка, будь-то формального или естественного, можно рассматривать как абстрактный механизм двойного назначения – для порождения последовательностей символов, являющихся предложениями того или иного языка, и для определения, принадлежит ли подаваемая на вход последовательность символов данному языку. Если при этом предложениям языка приписываются структурные дескрипции, то мы говорим о сильной порождающей или, соответственно, распознающей способности грамматики. Парсинг – это анализ последовательности символов на предмет принадлежности к данному языку и приписывание последовательности, принадлежащей к языку, её структурной дескрипции, т. е. описание её структуры. Соответственно парсер – это устройство, осуществляющее такой анализ.

В моделировании синтаксиса естественных языков для задач автоматического синтаксического анализа широко используются контекстно-свободные грамматики. Однако их выразительных средств оказывается недостаточно для моделирования синтаксиса естественного языка даже на уровне описания структуры составляющих, не говоря уже об описании синтаксических зависимостей. Один из самых серьёзных недостатков контекстно-свободных грамматик состоит в том, что они не позволяют описывать разрывные составляющие. Так, при помощи КС-правил невозможно описать структуру предложения *Что он видит?*, так как оно включает в себя составляющую VP, которая состоит из группы существительного (NP) *что* и глагола (Verb) *видит*, между которыми расположены другие составляющие, не входящие в состав данной VP. С другой стороны, у синтаксиса естественных языков и у КС-грамматик есть особенности, которые делают данный тип грамматик исключительно удобной основой для построения систем моделирования синтаксиса естественных языков для целей автоматического синтаксического анализа. Во-первых, КС-грамматики всё же позволяют описывать *достаточно большое* количество синтаксических структур естественного языка *в сильном смысле*, т. е. с приписыванием им корректных структурных дескрипций, и описывают *подавляющее большинство* синтаксических структур *в слабом смысле*, т. е. без приписывания им (корректных) структурных дескрипций. Так, для предложений, аналогичных приведённому выше предложению с разрывной глагольной группой, можно написать КС-правила, исходя из которых можно будет порождать (или воспринимать) этот класс предложений, не отображая однако структуру таких предложений (по крайней мере, правильно). Во-вторых, определение принадлежности последовательности символов к языку, задаваемому КС-грамматикой, имеет полиномиальную вычислительную сложность[1]. Эти практические достоинства контекстно-свободных грамматик имели следствием то, что данные грамматики в той или иной форме легли в основу ряда синтаксических теорий, ориентированных на автоматическую обработку языка. Естественно, КС-грамматики в таких теориях дополнялись различными расширениями, позволяющими, в частности, описывать разрывные составляющие, а также другие типы дистантных зависимостей, на описание которых у „чис-

[1] Распознавание принадлежности вместе с приписыванием структурной дескрипции (парсинг) имеет в худшем случае экспоненциальную сложность, причём такая ситуация имеет место также и в случае применения таких эффективных моделей парсеров, как парсер Эрли, если количество распознанных (под)деревьев растёт экспоненциально относительно длины входа.

тых“ контекстно-свободных грамматик не хватает мощности[2]. Из этих теорий наибольшее распространение получили HPSG (Head Driven Phrase Structure Grammar) [Pollard 1994, Sag 1997][3], GPSG (Generalized Phrase Structure Grammar) [Gazdar 1985], а также Лексико-функциональная грамматика [Bresnan 1982]. КС-грамматики позволяют описывать в слабом смысле *подавляющее большинство* синтаксических структур естественных языков, но не все возможные структуры. По всей видимости, в естественных языках существуют только два типа структур, не описываемые контекстно-свободными грамматиками в слабом смысле[4]. Это редупликационные структуры, где редуплицируется какая-то сложная составляющая, т.е. структуры вида $\{ vv \mid v \in \mathbf{\Sigma}+\}$, а также структуры, где грамматически обусловленная непроективность приводит к кросс-серийной синхронизации количества повторения тех или иных фрагментов, т. е. структуры вида $\{ a^m b^n c^m d^n \mid a,b,c,d \in \mathbf{\Sigma}+\}$. Примеры явлений данного типа приводятся в [Culy 1985] и [Shieber 1985] соответственно.

В качестве базового компонента разрабатываемой автором системы автоматического выделения синтаксических составляющих был создан парсер для предложений любых контекстно свободных языков, а также предложений следующих структурных типов:

- $\{ v_1^{k_{f(1)}} v_2^{k_{f(2)}} \ldots v_n^{k_{f(n)}} \mid v \in \mathbf{\Sigma}+ \}$ – предложения с синхронизацией количества повторения произвольных не пересекающихся фрагментов;
- $\{ x_k (\ldots(x_1 (x_0 v^n y_0)^n y_1)^n \ldots)^n y_k \mid v \in \mathbf{\Sigma}+; x_1,\ldots,x_k,y_1,\ldots,y_k \in \mathbf{\Sigma}^* \}$ – предложения с синхронизацией количества повторения произвольных вложенных друг в друга фрагментов;
- $\{ x_1 v x_2 v \ldots x_{k-1} v x_k \mid v \in \mathbf{\Sigma}+; x_1,\ldots,x_k \in \mathbf{\Sigma}^* \}$ – предложения с редупликациями, в том числе с дистантными.

Приписывание структурных дескрипций в парсере осуществляется как для неразрывных составляющих, как в контекстно свободных грамматиках, так и для разрывных. В настоящий момент автором ведётся разработка расширения, которое позволит анализировать предложения с эллипсисом, описывать различные синтаксические зависимости, а также осуществлять морфологическую унификацию.

Грамматика распознаваемого парсером языка – $G = <\mathbf{\Sigma}, \mathbf{V}, S, \mathbf{P}>$, где $\mathbf{\Sigma}$ и $\mathbf{V}$ – множества соответственно терминальных и нетерминальных символов, $S \in \mathbf{V}$ – выделенный, начальный, нетерминальный символ, $\mathbf{P}$ – множество правил. Правила записываются в нотации, в основе которой лежит традиционная нотация записи КС-правил, в которую внесены изменения и расширения, представленные ниже.

Альтернативные разложения одного символа записываются при помощи знака дизъюнкции и, где необходимо, скобок в правой части правила. Например, вместо традиционной записи вида $V \rightarrow \alpha\beta_1\gamma$, $V \rightarrow \alpha\beta_2\gamma$, $V \rightarrow \alpha\beta_3\gamma$ используется запись $V \rightarrow \alpha(\beta_1|\beta_2|\beta_3)\gamma$.

Опровержения запрещают вывод некоторых терминальных последовательностей. Так, в правиле $V \rightarrow \alpha(\beta_1|\beta_2\text{-}\chi\text{-}\psi|\beta_3)\gamma$ блокируется вывод из β_2 таких последовательностей тер-

---

[2] Явления подобного типа описываются также в рамках различных т. н. слабо контекстно-зависимых грамматик (mildly context-sensitive grammars), наиболее популярными из которых на сегодняшний день являются древовприсоединяющие грамматики, оперирующие не продукционными правилами, а правилами конструирования деревьев [Joshi 1997].

[3] В HPSG правила построения составляющих играют второстепенную роль, однако в слабом смысле HPSG эквивалентна контекстно-свободной грамматике.

[4] Здесь имеются в виду последовательности конкретных лексических сущностей. Если оперировать такими понятиями, как перемещение категории и оставляемый ей след, то возникнут также и другие типы контекстно-зависимых структур.

минальных символов, которые могут быть выведены из χ либо из ψ. Знак „-“ имеет в записи приоритет над „|“.

Итерация, т. е. повторение какого-либо фрагмента правой части правила обозначается при помощи *итератора* – пары чисел, указывающей нижнюю и верхнюю границу количества повторения последовательностей терминальных символов, выводимых из данного фрагмента правила. Так, в правиле $V \to \alpha(\beta_1|\beta_2\text{-}(\chi|\psi)|\beta_3)^{\{0,\infty\}}\gamma$ последовательности терминальных символов, выводимые из фрагмента $\beta_1|\beta_2\text{-}(\chi|\psi)|\beta_3$, на который „навешан“ итератор {0,∞}, могут быть повторены произвольное количество раз, либо данный фрагмент может быть проигнорирован (так как нижняя граница количества повторений ноль). Используются также следующие сокращённые обозначения итераторов: „*****“ – {0,∞}, „+“ – {1,∞}, „**?**“ – {0,1}.

Согласование итерации (количества повторений) вложенных либо непересекающихся последовательностей терминальных символов, выводимых из различных фрагментов одного и того же правила или из фрагментов разных правил, указывает на то, что последовательности терминальных символов, выводимые из данных фрагментов, должны повториться одинаковое количество раз. Оно реализуется через использование *связанных итераторов*, из которых только в одном эксплицитно указываются границы количества повторения, в нём также проставляется *метка* – начинающаяся с двоеточия последовательность символов. В остальных же итераторах, связанных с данным, проставляется только метка. Так, в правиле $V \to \alpha^{\{0,\infty:L1\}}(\beta_1|\beta_2{}^{\{:L1\}}\text{-}(\chi|\psi)|\beta_3{}^{\{0,\infty:L2\}})^{\{:L1\}}\gamma^{\{:L2\}}$ согласованными по количеству повторений являются фрагменты, на которые „навешан“ итератор с одной и той же меткой. Связанными могут быть также итераторы в пределах разных правил.

Согласование по значению в данной версии парсера вводится только для вхождений одноимённых нетерминальных символов. Оно указывает, что из согласованных таким образом вхождений должны быть выведены либо одинаковые последовательности терминальных символов (*слабое согласование*), либо одинаковые последовательности терминальных символов с одинаковыми структурными дескрипциями (*сильное согласование*). Это согласование реализуется через использование *связанных вхождений нетерминальных символов*, из которых только в одном эксплицитно присутствует сам нетерминал, при нём также проставляется *метка* – начинающаяся с двоеточия последовательность символов (как и в случае связанных итераторов), в остальных же вхождениях, связанных с данным, вместо нетерминала проставляется только метка. Если в метке после двоеточия следует символ подчёркивания (например „:_Label“), то объединённые данной меткой вхождения нетерминальных символов согласуются в слабом смысле, т. е. только по разложению до терминальной последовательности. Если метка устроена иначе (например „:Label“), то согласование осуществляется в сильном смысле, т. е. требуется не только равенство разложений до терминальных последовательностей, но и равенство приписываемых данным разложениям структурных дескрипций. Так, например, если в правиле $V \to A\ A\ A\ A\ A$ необходимо согласовать по терминальному разложению первое вхождение нетерминального символа *A* с четвёртым в сильном смысле, а второе с пятым в слабом смысле, это правило должно быть записано как $V \to \langle A{:}L\rangle\ \langle A{:}\_L\rangle\ A\ \langle{:}L\rangle\ \langle{:}\_L\rangle$. (Во избежание разночтения связанные вхождения будут заключаться в угловые скобки.) Связанными по значению могут быть вхождения нетерминалов также и в пределах разных правил.

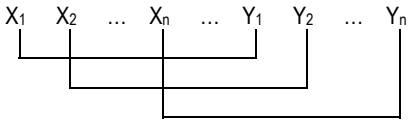
Смещение категорий – один из способов описания разрывных составляющих. В данном парсере была использована идея представления разрывных составляющих через указание условного „канонического“ места категории и места, куда категория была смещена. (Данные подход используется в Универсальной грамматике [Chomsky 1995].) Название сме-

щённой категории начинается с символа „$“, а название следа смещённой категории – с символа подчёркивания Так, предложения типа рус. *Дом я вижу* описывается следующим множеством правил (упрощённо):

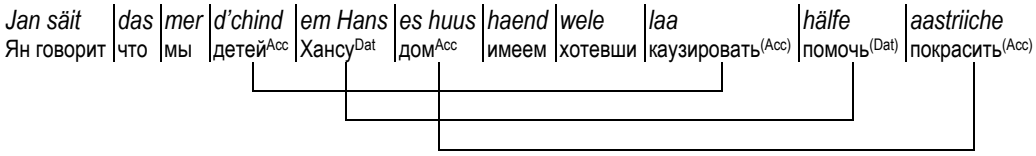
$S \rightarrow <\$NP> NP VP$
$VP \rightarrow Verb <\_NP>$

Эмпирические корреляты согласования по итерации и по значению

Согласование по итерации позволяет описывать в слабом смысле непроективные зависимости, встречающиеся в некоторых естественных языках, т. е. зависимости вида



Один из достаточно показательных примеров такой зависимости в естественных языках описывается в [Shieber 1985] на примере одного из швейцарских диалектов немецкого языка. Так, в предложении *Jan säit das mer d'chind em Hans es huus haend wele laa hälfe aastriiche*, ‘Ян говорит, что мы хотели заставить детей помочь Хансу покрасить дом’, глаголы связаны со своими несубъектными актантами следующим образом:



Из подобных придаточных предложений возможны только такие, в которых группа аккузативных глаголов в инфинитиве (кроме последнего) предшествует группе дативных, при этом дополнения глаголов располагаются в той же последовательности, что и сами глаголы. Несколько огрублено можно сказать, что глагольная группа (точнее, её фрагмент) в этом придаточном описывается языком $a^m b^n v c^m d^n$, который не может быть задан ни одной контекстно-свободной грамматикой, но описывается в слабом смысле при помощи согласования итерации:

$VP_{cross\text{-}serial} \rightarrow V_{Acc}{}^{m} V_{Dat}{}^{n} V_{Acc} V_{Aux} V_{Part} NP_{Acc}{}^{m} NP_{Dat}{}^{n} NP_{Acc}$

Механизм согласования по итерации позволяет также приписывать корректные структурные дескрипции в конструкциях типа рус. *Петя, Ваня и Серёжа любят, соответственно, Машу, Аню и Катю* или англ. *Pete, John and Serge love Mary, Ann and Cat, respectively*. Подобную конструкцию можно, естественно, описать и при помощи обычных КС-правил, однако только в слабом смысле: приписать предложению корректную структурную дескрипцию в этом случае не удастся. В данной же грамматике такое предложение описывается при помощи следующего множества правил (естественно, описание несколько упрощённое):

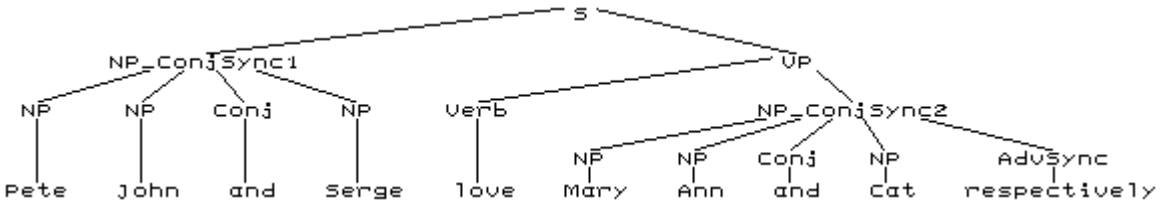
$S \rightarrow NP_{ConjSync1} VP$
$VP \rightarrow Verb\ NP_{ConjSync2}$
$NP_{ConjSync1} \rightarrow NP^{\{1,\infty:Label\}} Conj\ NP$
$NP_{ConjSync2} \rightarrow NP^{\{:Label\}} Conj\ NP\ AdvSync$

NP → «Pete»|«John»|«Serge»|«Mary»|«Ann»|«Cat»
Verb → «love»
AdvSync → «respectively»
Conj → «and»

В парсере множество правил, описывающее указанное предложение (с точностью до знаков пунктуации и пробелов между словами), будет выглядеть следующим образом[5]:

```
<S>::="<NP_ConjSync1><VP>",
<VP>::="<Verb><NP_ConjSync2>",
<NP_ConjSync1>::="<NP>{1,:Label}<Conj><NP>",
<NP_ConjSync2>::="<NP>{:Label}<Conj><NP><AdvSync>",
<NP>::=" *Pete| *John| *Serge| *Mary| *Ann| *Cat",
<Verb>::=" *love",
<AdvSync>::=" *respectively",
<Conj>::=" *and".
```

Если мы на вход парсера подадим данное множество правил и предложение `Pete John and Serge love Mary Ann and Cat respectively` (знаки препинания для простоты опускаются), то мы получим следующий результат:



Однако если попытаться проанализировать исходя из этих правил предложение `Pete John and Serge love Mary and Cat respectively`, то результат будет отрицательный (`The input string does not belong to the language specified`), так как NP, входящие в состав данных двух $NP_{ConjSync}$ должны повторяться одинаковое количество раз (синхронизация количество повторения фрагментов обеспечивается за счёт связанности итераторов одной меткой).

Стандартный случай согласования по терминальному разложению в естественных языках – это редупликация. Достаточно сложный случай этого явления приводится в [Culy 1985], где описываются ситуации, когда редупликации подвергаются фрагменты, которые сами описываются КС-правилами.

Описание разрывных составляющих

Как было указано выше, разрывные составляющие описываются в парсере через аппарат смещённых категорий. Так, упоминаемое выше предложение на швейцарском диалекте немецкого языка, содержащее кросс-серийные зависимости, может быть описано следующими правилами (описание несущественных в данном контексте категории опустим):

---

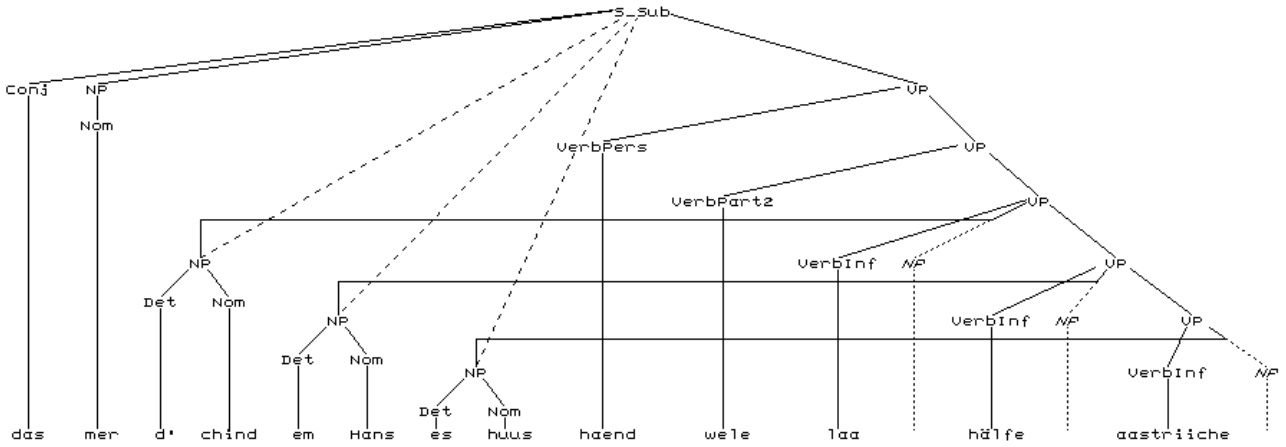
[5] Данный парсер не является законченным компьютерно-лингвистическим приложением, поэтому запись лексем в правилах носит чисто условный характер – вместо лексем здесь можно было просто указать символы, однако такая запись сделала бы примеры менее наглядными. В основе синтаксиса записи правил в парсере лежит несколько видоизменённая расширенная нотация Бэкуса-Наура [ISO 14977:1996]. От „теоретической“ нотации она отличается тем, что нетерминальные символы в парсере всегда - и в левой и в правой части - заключаются в угловые скобки, а не только тогда, когда они имеют при себе метки или представлены как ссылки через метки. Помимо этого правая часть правила заключается в кавычки или апострофы, правила разделяются запятой или точкой с запятой, вместо стрелки используется сочетание символов "::=".

S_{Sub} → Conj NP <$NP>* VP
VP → ($Verb_{Pers}$|$Verb_{Part2}$) VP | $Verb_{Inf}$ <_NP> VP?

В качестве иллюстрации приведём то, как указанное выше придаточное предложение может быть описано в парсере:

```
<S_Sub>::="<Conj><NP><$NP _traces=".alpha:LF">*<VP>$",
<VP>::="(<VerbPers>|<VerbPart2>)<VP>|<VerbInf><_NP.alpha><VP>?",
<NP>::="<Det>?<Nom>",
<VerbInf>::=" *laa| *hälfe| *aastriiche",
<VerbPart2>::=" *wele",
<VerbPers>::=" *haend",
<Nom>::=" *mer| *chind| *huus| *Hans",
<Conj>::=" *das",
<Det>::=" *d\'| *es| *em".
```

Результат применения данного множества правил к придаточному предложению `das mer d'chind em Hans es huus haend wele laa hälfe aastriiche` будет выглядеть следующим образом:



В парсере реализованы различные возможности управления связыванием смещённых категорий и их следов, которые используются, если есть несколько возможностей связывания, из которых реально могут быть реализованы не все. Так, можно указать, какие смещённые категории с какими именно следами должны связываться. Для этого следу присваивается идентификационный параметр, а при смещённых категориях указываются идентификационные параметры следов, с которыми данные категории могут быть связаны. Кроме того, можно указывать линейные ограничения на возможные связывания смещённых категорий и их следов. Это необходимо делать, в частности, для корректного описания упоминаемых выше синтаксических структур с непроективными зависимостями. Данные линейные ограничения представлены двумя параметрами (если какой-то параметр не указан, то выдаются все альтернативные варианты связывания):

- *Направление перемещения категории* – искать ли след слева или справа от данной смещённой категории.
- *Линейный порядок* – осуществлять ли связывание с линейно наиболее удалённым или наиболее близко расположенным следом, если есть альтернативы.

Так, например, запись вхождения некоторой смещённой категории Cat

```
<$Cat _traces="t1:NR, t2:LF">
```

означает, что след данной категории должен иметь идентификационный параметр t1 или t2, т.е. он должен выглядеть как <_Cat _id="t1"> или <_Cat _id="t1"> (альтернативная запись – <_Cat.t1> или <_Cat.t1>), при этом если след имеет идентификационный параметр t1, то он должен находиться справа от смещённой категории (параметр R), и в случае, если будет обнаружено несколько удовлетворяющих данному условию следов, то связывание будет осуществлено с тем из них, который расположен ближе всего к данной смещённой категории (N). Если же след обладает идентификационным параметром t2, то он должен находиться слева от смещённой категории (L), и связывание будет производиться с наиболее удалённым следом (F), если их несколько.

Алгоритм парсинга

Парсер реализован автором на языке программирования C++ на основе алгоритма нисходящего рекурсивного однонаправленного парсинга (Nondeterministic Recursive Top-Down LR-Parser). В алгоритм парсинга встроены две специальные эвристики, реализующие *стратегию управляемого поиска*. Первая из них представляет видоизменённый принцип look-ahead, заключающийся в том, что в ходе поиска осуществляется проверка того, сколько символов *будет* поглощено при том или ином пути прохода правил и сколько их реально остаётся на входе, и блокирующий поиск по путям, по которым он заведомо кончится неудачей из-за несоответствия количества символов, остающихся на входе, количеству символов, которые должны быть поглощены при поиске по тому или иному пути. Применение этого принципа позволяет сократить вероятность пессимистических сценариев (Worst Case Behavior) поиска и решить проблему левой рекурсии. Вторая эвристика блокирует поиск по тем путям, по которым заведомо не произойдёт ассоциации всех следов со всеми смещёнными категориями, что, в частности позволяет указывать итераторы с бесконечной верхней границей при следах (которые являются пустыми категориями, – в случае отсутствия данной эвристики подобная ситуация привела бы в данной модели парсера к бесконечной рекурсии).

Список литературы

1) [Bresnan 1982] Bresnan, J. (ed.) The mental representation of grammatical relations. MIT Press, 1982.
2) [Chomsky 1995] Chomsky, N., Lasnik, H.: The Theory of Principles and Parameters. // The Minimalist Program, 1995.
3) [Culy 1985] Culy, C. The complexity of the vocabulary of Bambara. // Linguistics and Philosophy, 8, 1985, pp. 345-351.
4) [Gazdar 1985] Gazdar, G., Klein, E., Pullum, G., Sag, I.: Generalized Phrase Structure Grammar, Harvard University Press, 1985.
5) [ISO 14977:1996] Extended Backus-Naur Form. ISO/IEC 14977:1996, 1996.
6) [Joshi 1997] Joshi, A. K., Schabes, Y.: Thee Adjoining Grammars. // Handbook of Formal Languages, 1997, pp. 69-123.
7) [Pollard 1994] Pollard, C., Sag, I.: Head Driven Phrase Structure Grammar. University of Chicago Press, 1994.
8) [Sag 1997] Sag, I., Wasow, Th.: Synactic Theory: a formal introduction. SCLI. 1997.
9) [Shieber 1985] Shieber, S. M. Evidence against the context-freeness of natural language. // Linguistics and Philosophy, 8, 1985, pp. 333-343.

Developing a parser for some classes of context-sensitive languages for automatic syntactic analysis

Alexander Perekrestenko

As a part of the work to develop a unification-based parser for German and Swedish the author has developed a parser for context-free language and some classes of context-sensitive languages. It processes both local and long-distance dependencies and assigns appropriate structural descriptions to continuous as well as discontinuous constituents. Special cases of non-local dependencies, so called cross-serial dependencies like those met in Swiss German and Dutch, are also processed. The parser is implemented in the C++ programming language as a recursive top-down LR-parser with several look-ahead-style heuristics to increase the efficiency of the parsing, solve the problem of both direct and indirect left recursion and to make it possible to process unbounded iterations of some kinds of empty categories.