# 自然语言句法分析器自动构造系统

梁 欣　　臧德滋
（同济大学计算机系　上海 200092）

摘要：　在自然语言处理系统中，句法分析器是一个十分重要的部分，但用手工来构造句法分析器是一件非常耗时且易出错的工作。本文所提出的句法分析器自动构造系统其目的就是为自然语言句法分析器的构造提供一个高效而又方便的自动化手段。本文介绍了作者定义的句法分析规则描述语言以及利用程序变换方法由句法分析规则自动构造自然语言句法分析器的设计思想。

## An Automatic Parser Constructing System of Natural Languege

Liang Xin　　Zang Dezi
(Tongji University, Shanghai 200092)

Abstract: A natural language parser is a very important part in a natural language processing system while constructing parsers manually is a time-consuming and error-making work. The aim of the automatic parser constructing system proposed in this paper is to provide an efficient and automatic means to construct a natural language parser. The description language for parse rules defined by the authors is introduced. And the main idea of making use of programm transformation to automatically construct a parser is also introduced in this paper.

## 一、引　言

句法分析是自然语言处理中的一个重要组成部分，人们在研制自然语言处理系统时，通常都是用手工编制程序来构造句法分析器。在我们研制的基于动词配价的德汉翻译系统[2]中，采用结构语义属性文法[3]进行句法分析，并以完全句法结构树作为句法分析结果，同样是采用手工方式来构造句法分析器。

为了确保句法分析器的正确性，增强句法分析器的灵活性，提高分析效率，进一步提出了句法分析器的自动构造系统.该系统首先定义一组句法分析规则描述语言ParseSpec，然后，设计规则编辑器，建立相应的句法分析规则描述库，最后用规则编译器构造出句法分析器的语言代码，完成句法分析器的自动构造。

## 二、句法分析规则描述语言

句法分析规则描述语言ParseSpec是用于描述句法分析规则及句法结构树映射关系的甚高级语言。由于

ParseSpec的基础是结构语义属性文法，因此在用ParseSpec描述的句法分析规则中包含了对自然语言进行句法分析的各种动作(例如依照结构语义属性文法进行句法分析、根据句法分析结果构造句法分析树等)。所以，用ParseSpec描述的句法分析规则不是一种简单的规则，而是包含有各种分析与构造动作的规则。

由于ParseSpec仅用于描述句法分析规则和句法结构树映射关系，它的数据类型和语句结构都非常简单，只有一种类似于赋值语句形式的规则描述语句。在自然语言句法分析器自动构造系统中，采用数据库来保存用ParseSpec描述的句法分析规则，故用ParseSpec来描述句法分析规则时可以是"无序"的，即各条规则定义的先后顺序是无关紧要的。若将整个句法分析规则集看作是一个树型结构，则 ParseSpec唯一保留的规则名Sentence在逻辑上是句法分析规则树的"树根"。

下面我们通过例子来说明如何用ParseSpec来描述德语的句法分析规则。由文献[5] 可查得配价类型为"530"的动词所对应的句型为S+Vb+Akko+Dato，其中"S"为主语，"Akko"为第四格宾语，"Dato"为第三格宾语。"Vb"为谓语动词，它是一个三价动词，需要一个主语、一个第三格宾语和一个第四格宾语作它的配价伙伴。要想能分析这个句型，应该相应地提供句型、主语、第三格宾语、第四格宾语以及定语的分析规则。其中句型的分析规则可作如下的描述：

```
<S530>::=<Subject>#RealSubject#+[Verb]#Predicate#
        +<Akko>#DirObj#+<Dato>#IndirObj#                    (1)
```

这里，#RealSubject#表示非终结符<Subject>的分析结果应该映射到完全句法结构树的实际主语位置上。类似地，#DirObj#和#IndirObj#表示非终结符<Akko>和<Dato>要分别映射到直接宾语和间接宾语的位置上。

在规则描述(1)中，主语<Subject>、第三格宾格<Akko>和第四格宾语<Dato>这三个非终结符需要进一步给出其分析规则描述：

```
<Subject>::=[Article]?+[Noun]{1}"Human"$                    (2)
<Akko>::=<nounFMod>*+[Noun]{4}"Substance"$                  (3)
<Dato>::=[Article]?+<NounFMod>*+[Noun]{3}"Human"$           (4)
```

规则描述(2)表明主语可以由一个第一格名词组成，这个名词的语义属性是"指人"的(Human)，作为主语的名词前可以有一个冠词，也可以没有冠词，该名词是构成主语的核心成分。规则描述(3)是第四格宾语的分析规则描述，它表明第四格宾语Akko应该由一个第四格名词和若干个名词的前置定语所组成，这个名词的语义属性是"指物"的(Substance)，名词的前置定语<NounFMod>还需由其它的分析规则来描述定义。第三格宾语的规则描述(4)与第四格宾语的规则描述(3)很相似，只是规则描述(4)表明第三格宾语Dato 的核心成分是一个语义属性为"指人"的第三格名词。

```
<NounFMod>::=[Adj]*|[PossPron]+[Adj]*                       (5)
```

分析规则描述(5)说明名词的前置定语可以由若干个形容词[Adj]构成，或者由一个所有格代词[PossPron]加上若干个形容词组成。这些定语的分析结果映射关系从属于它们所修饰的句子成分(Akko或Dato)，因此在这里就没有必要标出它们的映射关系。

## 三、句法分析器自动构造系统

句法分析器自动构造系统的设计目标是用程序设计自动化的手段来辅助构造句法分析器，减轻自然语言处理系统研究人员的工作量，使他们把精力集中在对自然语言的研究分析上，而不是把大量的精力花费在构造句法分析器的程序设计实现细节上。同时，自动构造句法分析器可以保证构造出的句法分析器在程序设计

语言上没有语法错误，提高程序的质量。从图1可以看出，句法分析器自动构造系统的核心部分是规则编辑器、句法分析规则描述库和规则编译器。

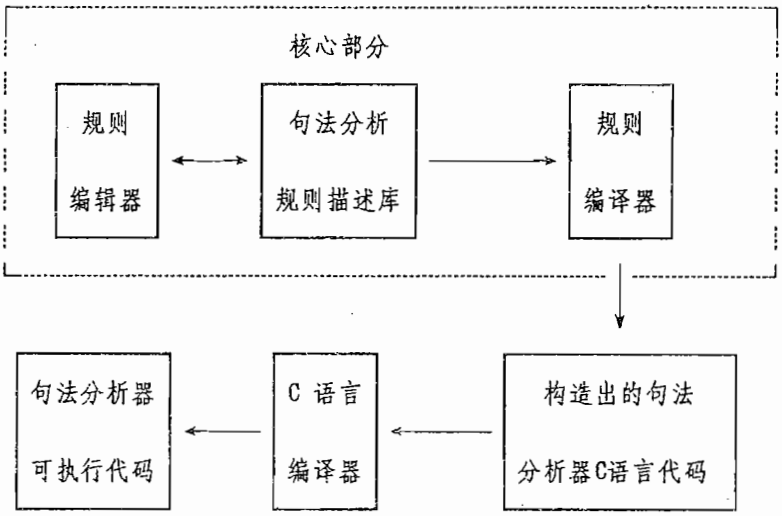


图1 句法分析器自动构造系统基本结构图

1. 规则编辑器

规则编辑器用于输入、修改或删除用分析规则描述语言ParseSpec描述的句法分析规则。这是一个语法制导的全屏幕编辑器，它依照ParseSpec的语法自动给出相应的语法框架，引导用户进行规则编辑。同时，在编辑过程中自动地进行相应的语法检查，以保证用户输入的句法分析规则描述在 ParseSpec语法上的正确性。

规则编辑器直接对句法分析规则描述库中的规则描述进行操作。 规则编辑器的语法制导机制着重于处理表达式(含标识符、重复符、格类型、配价类型等)，这些处理工作主要是根据ParseSpec的语法，在用户使用规则编辑器时自动根据当时的编辑状态和所编辑部分的内容，给出相应的弹出式输入框，使得用户只可能在输入框内进行编辑，同时对用户的操作进行必要的 ParseSpec语法检查，当发现有语法错误时立即给出必要的提示信息，直到用户改正了错误才能继续进行下一步的编辑。

例如对于"规则名"的语法制导处理，根据"规则名"及其相关项的定义：

```
<规则名>::=<标识符>
<标识符>::=<字母>{<字母>|<数字>}
<字母>::=A..Z|a..z
<数字>::=0..9
```

编辑器在对"规则名"进行编辑处理时，首先自动弹出一个规则名输入框，用户只可能限定在这个输入框内进行编辑。用户在这个输入框内进行编辑时，规则编辑器根据上面对"规则名"的定义进行语法检查，随时给出必要的提示信息，让用户改正输入的错误，以保证这个输入框内的"规则名"符合ParseSpec的语法。由于规则编辑器自动限定"规则名"的有效长度，并自动给出"规则名"的左、右边界字符，因此在这方面用户不可能造成语法错误。此外，规则编辑器还会根据当前的编辑操作状态(插入、删除还是改写)，自动对输入框进行必要的伸格或缩格处理。当用户对所编辑的"规则名"感到满意时，规则编辑器会自动将输入框的内容插入到屏幕的适当位置，然后输入框自动从屏幕上消失。

规则编辑器所具有的另一个特点是"规则完整性检查"功能，即当用户完成对句法分析规则描述的编辑，退出规则编辑器时，该编辑器会自动调用"规则完整性检查"算法，对句法分析规则描述库内当前的全部分析规则描述进行检查，并将检查结果记录下来，然后才真正结束编辑器的运行。完整性检查的目的是要保证规则编辑器所构造出的句法分析器是"完整"的，即不会出现句法分析器调用一条未经定义过的句法分析规则的情况。

2. 句法分析规则描述库

句法分析规则描述库负责管理用于构造句法分析器的全部句法分析规则描述，这是一个由数据库管理系统来实现的规则描述库，句法分析器自动构造系统中的各个软件工具通过它而联系集成在一起。利用数据库管理系统来管理句法分析规则描述，便于句法分析器自动构造系统的集成和扩充，提高系统的灵活性。

分析规则描述库的数据结构与FoxBase数据库的数据结构相同，与数据管理有关的文件有两类：数据文件和索引文件。数据文件用于保存分析规则描述；索引文件用于保存为加快对规则描述的检索速度而建立的索引，分析规则描述库数据文件结构和索引文件结构与FoxBase的数据文件结构和索引文件结构相兼容。

3. 规则编译器

当用户利用规则编辑器输入或修改了用ParseSpec描述的句法分析规则后，就可利用规则编译器对句法分析规则进行编译。规则编译器采用的是程序变换技术，根据ParseSpec的语法以及句法分析规则描述，设置相应的语义动作，将一条句法分析规则变换成能与之等价的一组C语言代码。当规则编译器完成了对句法分析规则描述库中全部句法分析规则描述的编译后，就构造出了一个完整的、C语言代码形式的句法分析器。

规则编译器的输出是C语言代码形式的句法分析器，这样做有利于构造出的句法分析器在不同型号、不同操作系统机器之间的移植。当用户希望得到可执行的句法分析器时，就可调用一个C语言编译器，将C语言代码形式的句法分析器进一步编译成可执行的句法分析器目标代码。

规则编译器中采用的主要技术是程序变换技术与增量式编译技术。

(1) 程序变换技术

句法分析规则描述库中保存的是用ParseSpec描述的句法分析规则，这些规则描述从某种意义上来说就是用ParseSpec语言编写的程序。"程序变换"指的是为了某种目的而用一个程序去代替另一个程序，具体地说，就是利用一组程序变换规则，将用甚高级语言ParseSpec编写的"程序"逐步地向更加具体、高效的C语言代码变换。由于每一步变换都是依据变换规则而进行的，故只要变换规则选择得当，就能够保证变换前后的程序在功能上是等价的，由此保证整个变换过程的正确性。程序变换规则用于逐步降低变换对象(程序)的抽象程度，这是一种纵向变换，即用较低抽象程度的程序结构（C语言代码）来代替较高抽象程度的结构(ParseSpec描述)。

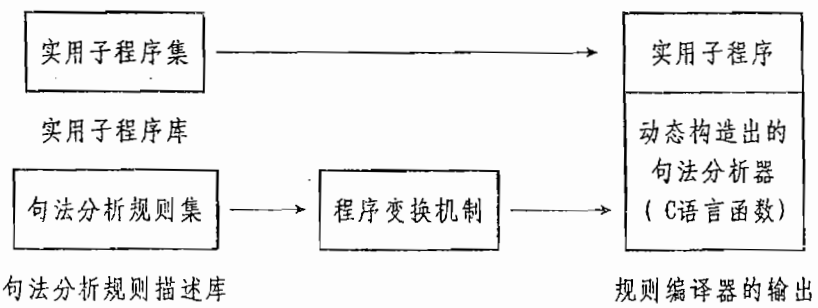


图2 规则编译(程序变换)原理

从图2可以看出，规则编译器的工作原理就是利用程序变换机制，根据规则描述语言ParseSpec和C语言的

语法，用一组功能与之等价的C语言代码(C语言函数)来代换句法分析规则描述库中的一条句法分析规则描述，如此逐条地对句法分析规则描述库中的规则描述进行替换，再加上一些实用子程序(Utilities)的C语言代码，就构造出一个在功能上与句法分析规则描述库中规则集相等价的自然语言句法分析器，下面我们结合一个具体的句法分析规则描述例子来说明编译过程。

例1. <S530>::=<Subject>#RealSubject#+[Verb]#Predicate#
+<Akko>#DirObj#+<Dato>#IndirObj#

这是一个用于句型分析的句法分析规则描述，这里有三个非终结符(<Subject>、<Akko>和<Dato>) 需要由其它分析规则来进一步加以描述，但这并不影响规则编译器对例1中句法分析规则的编译和构造动作。

规则编译器按从左到右的顺序依次进行处理。首先处理的是规则项<Subject>#RealSubject#，此规则项由一个非终结符(规则名)和一个完全句法结构树映射关系组成，按照变换规则，这个规则项被变换成如下的C语言代码：Subject('RealSubject')，然后处理第二个规则项[Verb]#Predicate#，它由一个终结符和一个映射关系组成，按照非终结符(词性)的变换规则，它被变换成如下的C语言代码：

```
Word=GetNextWord();
if (Word,Category==Verb)
   Attach(Word,'Predicate');
else
   return(-1);
```

这里，Word是一个结构型的全程变量，函数Attach是一个实用子程序，用于将给定的字符串(变量Word的值)按给定的映射关系('Predicate')映射到完全句法结构树上(构造句法分析树)。函数GetNextWord() 也是一个实用子程序，用于从待分析句子中取出一个单词供分析之用。

例1是一个用于分析德语句型的分析规则描述。考虑到德语中常有第三格宾语和第四格宾语互换的情况，就要对宾语作些特殊处理，当句法分析规则描述中有两个宾语映射关系(#DirObj#和 #IndirObj#)时，变换处理就和只有一个宾语映射关系时稍有不同。在这里，变换构造出的相应C语言代码就是：

```
if (Akko('DirObj'))
   Dato('IndirObj');
else{
   Dato('IndirObj'),
   Akko('DirObj'),
    }
```

至此，例1的变换已经完成。综合起来，经过程序变换，就得到与例1所描述的句法分析规则所对应的一组C 语言代码：

```
S530()
{
  Subject('RealSubject');
  Word=GetNextWord();
```

```
if (Word.Category==Verb)
   Attach(Word,'Predicate');
else
 return(-1);
if (Akko('DirObj'))
   Dato('IndirObj');
else {
     Dato('IndirObj');
     Akko('DirObj');
     }
return 0;
}
```

(2)增量式编译

在本文中,"增量式编译"指的是增量式地构造句法分析器。前面已经谈到,规则编译器的任务就是利用程序变换机制构造句法分析器。当一个句法分析器已根据给定的分析规则被构造出来后,若对分析规则集中的某些规则进行修改操作(包括增加和删除某些分析规则),这个句法分析器必然也要作出相应的修改,即要重新构造一个句法分析器,实际上,在重新构造句法分析器时,没有受到分析规则修改影响的那些C 语言代码部分仍然可以加以利用,而只需构造那些受修改影响的代码部分。将修改过的代码与未修改过的代码重新组合起来,就得到新的句法分析器,这就是"增量式编译"的思想。

规则编辑器在对句法分析规则描述库中的分析规则描述进行编辑操作时经过适当的处理,可使得增量式编译变得十分简便。

## 四、结束语

到目前为止,由于对自然语言处理系统的研究工作具有很大的探索性,常常需要一边试验,一边改进,经常需要修改,甚至重新构造相应的用于句法分析的程序系统。为了克服手工修改或重构时费时、低效及难以保证正确性等问题,自动构造句法分析器是一个有效的方法。

## 参 考 文 献


[1] Peter Pepper, Ed., Programm Transformation and Programming Environments, Spring Verlag, 1984

[2] 柴佩琪,臧德滋,许玉祥,赵晓东,梁欣,基于动词配价的德汉机器翻译1992 年全国机器翻译学术会议论文集<<机器翻译研究进展>>

[3] 梁欣,臧德滋,柴佩琪,德汉机器翻译系统中基于结构语义属性文法的句法分析机制,1992年全国机器翻译学术会议论文集<<机器翻译研究进展>>

[4] Jan Heering et al., Incremental Generation of Parsers, IEEE Trans. on Software Engineering, Vol 16, No.12, 1990

[5] Gerhard Wahrig, dtv_Wortbuch der deutsche Sprache, Munchen, 1978