《外语研究》2012 年，第 3 期，p1-6。CSSCI 检索。

树邻接语法[1]

冯志伟

（杭州师范大学， 浙江 杭州 310036）

摘要 树邻接语法是一种基于树的形式语法，本文比较了这种语法与短语结构语法的异同，介绍了树邻接语法的基本组成要素及其操作模式，重点介绍了接插和替换两种操作，并介绍了推导关系树、树集合、串语言等概念，最后介绍树邻接语法的新发展--词汇化树邻接语法。

关键词 树邻接语法，词汇化树邻接语法，接插，替换，推导关系树

1. 前言

1975 年，A. K. Joshi, L. S. Levy, M. Takahashi 等人提出“**树邻接语法**”（Tree Adjoining Grammar，简称 TAG），TAG 可以识别和生成“树邻接语言”(Tree Adjoining Langrage，简 称 TAL)。目前，在计算语言学和自然语言处理中，树邻接语法是一种颇有影响的形式语法，它是一个形式化的树重写系统。本文简要介绍这种重要的形式语法。

树邻接语法是在短语结构语法的基础上发展起来的。它以句法结构树作为核心操作对象，在树的基础上来组织语言知识，它的产生式规则也对应着树结构，它以线性的一维形式来表达二维的树结构。

树邻接语法与短语结构语法的不同之处在于，短语结构语法是一种基于符号串的形式语法，它以**符号串**（string）作为操作对象，而树邻接语法是基于**树**（tree）的形式语法，树邻接语法的规则比短语结构语法的规则写得更加细致。例如，

汉语双宾语的树邻接语法的规则可以写为：

VP → VP(V NP) NP

这个规则中包含子树 VP(V NP)，它是一个树结构规则，因此，这个规则实际上包含了短语结构语法的如下两条规则：

VP → VP NP

VP → V NP

如果我们使用短语结构语法的两条规则来生成汉语，由于第一条规则 VP→ VP NP 是自嵌入的规则，在推导过程中，可以用规则右部的 VP 来不断重写规则左部的 VP，从而产生

VP NP NP NP NP NP … NP

这样的包含若干个 NP 的符号串，显然这样的符号串在汉语中是不会出现的，是汉语中的不合法句子。

但是，如果我们使用树邻接语法，只有 VP → VP(V NP) NP 这样包含了树结构的规则，就只能产生

VP(V NP) NP

这样的树结构，它只包含两个 NP 作为双宾语，是符合汉语语法的。这样，就限制了短语结

[1] 本文受国家社会科学基金资助，项目号 03BYY019。

构语法过强的生成能力，保证了规则的准确性。

由此可见，短语结构语法是一个基于符号串的生成系统，而树邻接语法是基于树的生成系统。当然，由树邻接语法生成的树邻接语言仍然是符号串语言，最终的生成的结果中并不包含树结构。树邻接语法是对于短语结构语法的重要改进，它比短语结构语法更能反映自然语言的真实面貌。

2. 树邻接语法的基本组成要素

一个树邻接语法是一个五元组(∑，NT，I，A，S)，其中，

(i) ∑是终极符号的有限集合；

(ii) NT 是非终极符号的有限集合[2]，∑∩NT=ϕ；

(ii) S 是初始符号，它是一个特殊的非终极符号，S∈NT；

(iv) I 是**初始树**[3](见图 1 中 I 树)的有限集合，它有两个特征：

① 所有的非叶子结点都用非终极符号标记；

② 所有的叶子结点，或者用终极符号标记，或者用带有下箭头(↓)的非终极符号标记。下箭头(↓)是初始树的标志，它的含义是“**替换**”（substitution），它表示该结点可以被其他的树结构替换。

如果一个初始树的根结点为 X，则这个初始树在 TAG 系统中叫做 X 类型的初始树。

(v) A 是**辅助树**[4](见图 1 中 A 树)的有限集合，它也有两个特征：

① 所有的非叶子结点都用非终极符号标记；

② 辅助树叶子上的结点用终极符号或非终极符号标记。A 树叶子上的非终极符号结点带有**接插**（adjunction）符号，用星号（*）标注将被接插的结点，这个结点叫做“**落脚结点**”（foot node），落脚结点标记的非终极符号（短语类符号）要跟它所在的树结构的根结点相同。

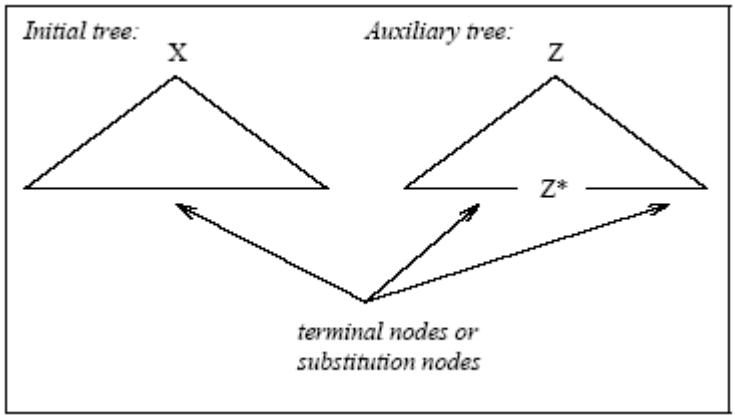


图 1 初始树与辅助树图例

在初始树集合 I 与辅助树集合中 A 的树都没有进行过任何操作，它们是本来就储存在集合中的，这样的树叫“**基础树**”（elementary tree），主要用于 TAG 中的一些操作。

在 I∪A 这个集合中所有的树都叫做基础树。如果一个基础树的根用非终极符号 X 标记，我们把它叫做 X 类型的基础树（X-type elementary tree）。在 TAG 中，基础树就是储存在 I 或 A 中没有进行过任何操作的树。

---

[2] 树邻接语法中，用小写字母表示终极符号，用大小字母表示非终极符号。

[3] 初始树是图 1 左边的 Initial tree，简称 I 树。

[4] 辅助树是图 1 右边的 Auxiliary tree，简称 A 树。

TAG 是树生成系统而不是一个串生成系统，但 TAG 最终生成的树可以用来分析和解释目标语言的串语言。

在一个 TAG 中，通过树的推导而生成目标语言中的树，下面介绍目标语言中的“树”及这些“树”的生成过程。

如果一棵树是由 I∪A 的集合中任意两棵树组合而成的，那么这棵树就叫做**“推导树”**（derived tree）。

得到推导树的过程叫做**“推导过程”**（derivation process），在这个过程中使用的操作有两个，一个是接插，另一个是替换。

3. 接插

“接插”（adjoining）是把辅助树 β 插到任意树 α[5]中而建立一棵新树的过程。

设 α 是一个包括非替换结点 n 的树，n 为非替换结点，在树 α 的非替换结点上有一个标记是 X，设 β 是一个辅助树，β 的根结点上的标记也是 X。由 β 与 α 在树 α 中的结点 n 处接插，可以获得一棵**结果树** γ（resulting tree）。这个过程在图 2 上图 A 中有图例，它的具体操作步骤如下：

（1）第一步是**“剪”**：设树 α 的子树为 t，t 由结点 n 支配，结点 n 上的标记为 X，在 α 中剪掉 t 后保存 n 的副本；

（2）第二步是**“接”**：从树 α 中剪下来的 t 的根结点与辅助树 β 的末端结点相同，将子树 t 的根结点接到辅助树 β 中标有*号的落脚结点上；

（3）第三步是**“插”**：由于辅助树 β 的根结点与树 α 中的结点 n 上的标记相同，都是 X，因此，可以将所得到的新的辅助树 β 插到树 α 中结点 n 的副本上，就可以得到结果树 γ 了。

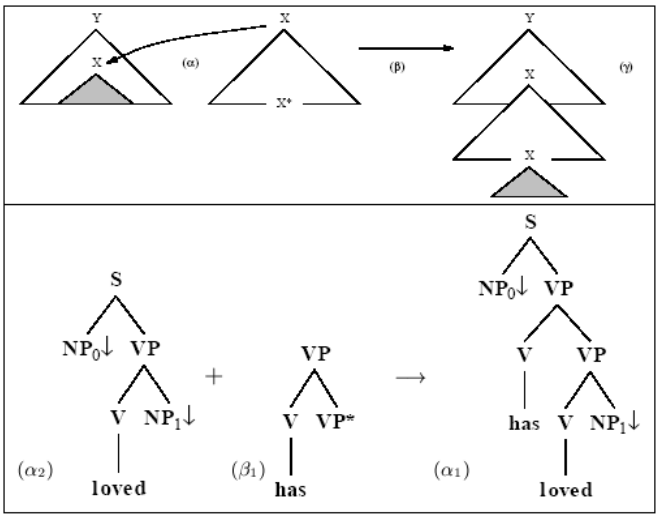
下面我们具体看一个树邻接语法中邻接操作的例子（见图 2 下图 B）。在 α_2 这棵树中，辅助树 β_1 邻接在 VP 这个结点上，α_1 是最终的结果树（见图 2 下图 B）。

接插的具体操作过程是：

（1）**“剪”**：设 α_2 的子树是 t，t 的根结点是 n，n 上的标记是 VP，剪掉子树 t 后保存 n 的副本，结点 n 上的标记还是 VP；

（2）**“接”**：子树 t 的根结点与 β_1 上标有*号的落脚结点相同，都是 VP，将子树 t 的根结点接到 β_1 的有*号的落脚结点上，这时，辅助树 β_1 的根结点上的标记与树 α_2 的副本中结点 n 的标记相同，都是 VP；

（3）**“插”**：将所得到的这个新的辅助树 β 插到 α_2 的副本中标记为 VP 的结点 n 上，就可以得到结果树 α_1。



[5] α 可以是初始树、辅助树，也可以是推导树。

图 2　接插操作

在 TAG 中有三种接插，标记经过接插之后，就可以确定某个辅助树可以接插在初始树中某个指定的结点上，这种标记可以使接插操作更方便。

假设 G 是 TAG，G＝(Σ，NT，I，A，S)，可以为 I 与 A 中的每个基础树上的结点规定下面的一种接插操作：

（1）**选择接插**（Selective Adjunction，简称 SA（T））：辅助树可以接插到指定结点上，但这种辅助树上的接插不是强制的，因此这种接插叫做选择接插。

（2）**空接插**（Null Adjunction，简称 NA）：在指定结点上不允许有任何邻接成分。

（3）**强制接插**（Obligatory Adjunction，简称 OA（T））：令 T 是辅助树集合，当 T⊆A 时，T 中的树一定要接插在指定结点上。在这种情况下，辅助树的接插成分是强制的，因此这种接插叫做强制接插。

4. 替换

下面再来看推导过程中的另外一个操作—“**替换**”（substitution）。

替换是用一个推导树 β 替换初始树 α 而建立一棵新树的过程。

替换只发生在一棵树叶子的非终极结点上(见图 3 的上图 A)。在 TAG 中将要发生替换的结点都用下箭头(↓)做标记。

我们来看 TAG 中一个具体的替换操作的例子，在图 3 的下图 B 中，树 α_3 的根结点 NP 与初始树 α_2 中 VP 子树下的 NP1 结点相同，NP1 上标有下箭头(↓)，因此可以用树 α_3 中 NP 替换初始树 α_2 中 NP1，这样就得到了结果树 α_4。

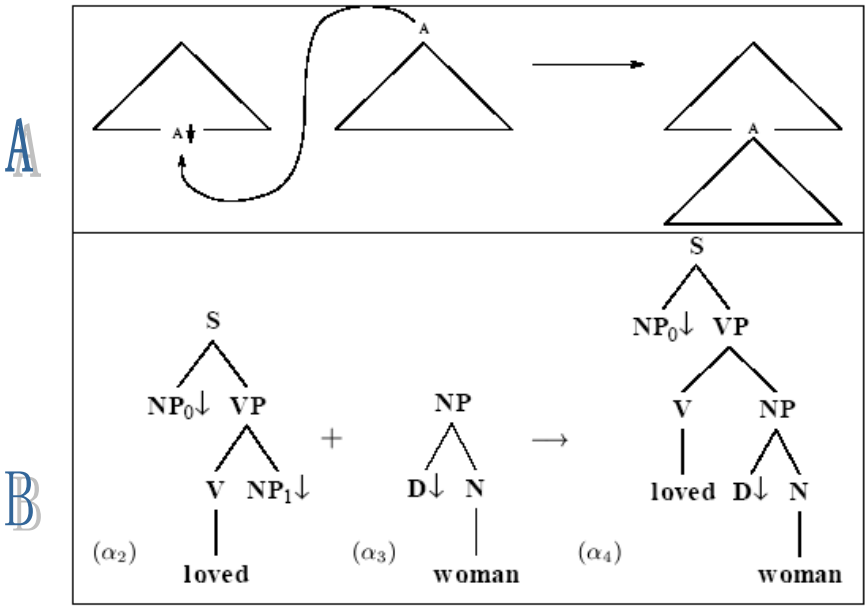


图 3　替换操作

当替换在一个结点 n 上发生时，这个结点由将要替换的树代替。当一个结点被标记上要被替换时，只有推导树可以替换它。在标记了替换的结点上不允许出现任何接插操作。例如，在图 2 与图 3 的下图 B 中的 α_2 这棵树上，在 NP_0 与 NP_1 这两个结点上都不允许有任何接插操作。

在 TAG 的推导树中，没有给出足够的信息来确定这棵树是如何构成的，但是，这些信息在 TAG 的推导关系树中都给出了。

5. 推导关系树

“推导关系树“（derivation tree）是一个确定推导树构成过程的树。接插和替换这两种操作在 TAG 的推导过程中都被包括进去了。例如推导树 α_5 可以产生下面的句子（图 4）：

（1）Yesterday a man saw Mary.

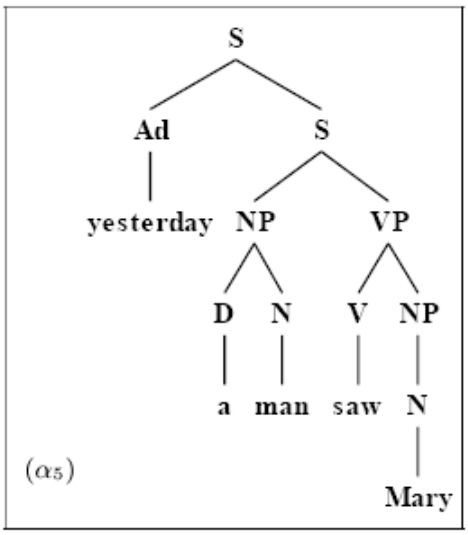


图 4　句（1）的推导树

图 4 是句（1）的推导树，它表明了句（1）的内部组成结构，但却没有告诉我们句（1）的获得过程。在下面的图 5 中，句（1）被分解为几棵局部的基础树，图 6 是句（1）的推导关系树，它表明了图 4 中的推导树是如何获得的。

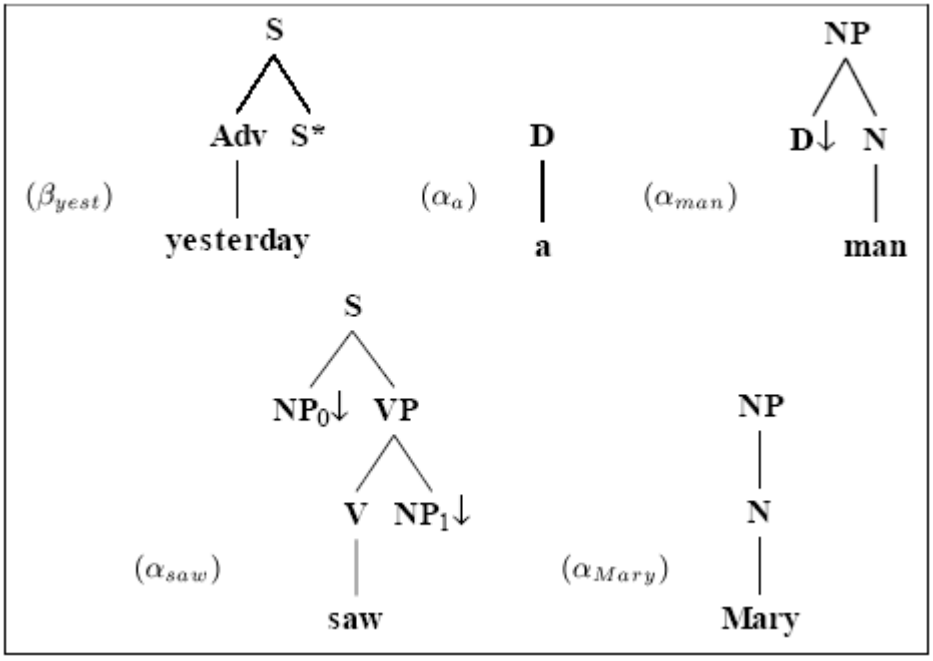


图 5　句（1）的局部基础树

在 TAG 的推导关系树中，除根结点外，树的地址与各个结点联系在一起。结合图 5 中句

（1）的局部基础树，图 6 中的推导关系树应做如下解释：α_{a}在α_{man}这棵树中，在地址 1 被替换，地址 1 与图 5 中结点 D 相关联，α_{man}在树α_{saw}中，在地址 1 被替换，地址 1 还与图 5 中的结点 NP_0相关联，α_{Mary}在树α_{saw}中，在地址 2.2 被替换，地址 2.2 与图 5 中的结点 NP_1相关联，β_{yest}这棵树在地址 0 邻接，地址 0 与图 5 中的结点 S 相关联[6]。

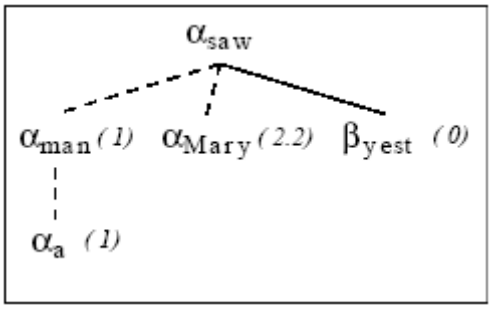


图 6　句（1）的推导关系树

在 TAG 的推导关系树中，如果两棵树是接插关系，用实线连接；如果是替换关系，用虚线连接。比如在图 6 中，β_{yest}与α_{saw}之间由实线相连，它们是接插关系，在图 5 中树β_{yest}上有 S^*，α_{saw}的根结点是 S，因此，树β_{yest}中的 S^*要由α_{saw}接插，这时，由于初始树α_{saw}已经完全地接到了辅助树β_{yest}中，接插操作所得到的新的辅助树就可以直接作为接插操作的结果了。再看图 6 中α_{a}与α_{man}之间是虚线，在图 5 中，α_{man}中的叶子结点 D 上标有下箭头，α_{a}的根结点是 D，因此α_{a}可以替换 D↓。其它树之间的关系可以依此类推。

由于接插在一个结点只能发生一次，因此在推导关系树中，一个父亲结点的所有子结点的地址都不相同，也就是说在同一层级上的兄弟结点地址不能相同。比如在句（1）的推导关系树中，α_{saw}的所有子结点地址都不相同，α_{man}与α_{a}是父子关系，所以地址可以相同。

6. 树集合的定义和属性

树邻接语法是一个树生成系统，它生成的树组合在一起可以形成“**树集合**”（tree sets）。下面介绍一下 TAG 生成的树集合的定义和属性。

TAG 中的“树集合”是指从某个以 S 为根的初始树推导出的“**绝对初始树**”（completed initial trees）的集合。在这里，绝对初始树是指叶子上没有替换结点的初始树。它有下面一些属性：

（1）**可识别树集合**（recognizable tree sets）[7]严格包括在树邻接语法的树集合中；[8]

（2）在给定的 TAG 的树集合中，树集合中所有树的**路径集合**（set of path）都是**上下文无关语言**（Context-Free Language），记为 P（T（G））；

（3）对于 TAG 中每个语法 G 来说，G 的树集合 T（G）都可以多次被识别。

树邻接语法可以识别和最终生成的语言是“**树邻接语言**”（Tree Ajoining Language，简称 TAL），树邻接语言中不再包含任何形式的树，它是一种“**串语言**”（string languages）。树邻接语言生成的**树结构**（tree structure）虽然不是**串**（string），但最终还是为了识别和生成这样的串语言。

7. 串语言的定义和属性

---

[6] 在 TAG 中，用哪种顺序解释推导关系树对最终的推导树没有影响。

[7] 可识别树集合也叫做规则树集合（regular tree sets）。

[8] 形式化公式为：recognizable tree sets $\subset$ T(G)。

设 T_G＝｛t|t 是从某个以 S 为根的初始树的推导结果｝

假设 L（G）是 TAG 的串语言，则 L（G）是树集合中所有树的生成结果的集合。定义如下：

L_G＝｛w|w 是 T_G中某个树 t 的生成结果｝

TAG 中的串语言有以下一些属性：

（1）树邻接语言完全包括上下文无关语言；如果我们用 CFL 表示**上下文无关语言**（Context Free Language），用 TAL 表示**树邻接语言**（Tree Ajoining Language），则有：CFL ⊂ TAL；

（2）树邻接语言是**半线性**（semi-linear）的；

（3）树邻接语言是**语言的完整抽象集合**（full family of languages，简称 full AFLs）；

（4）与树邻接语法 TAG 相对应的自动机是**嵌入式下推自动机**（embedded push-down automaton，简称 EPDA）,它精确地概括了树邻接语言集合的特征；

（5）树邻接语言都有一个**起动词条**（pumping lemma）；

（6）树邻接语言可以被多次分析。

总起来说，一个 TAG 语法包括一组有限的初始树和辅助树，用一个 TAG 语法生成自然语言中的句子，就是从 S 类型的初始树开始，不断地进行替换和接插操作，直到所有带替换标记的结点都已经被替换了，所有带接插标记的结点都已经被接插了，最后，把所得到的树的叶子结点按顺序列出，就可以得到该 TAG 语法所生成的句子的集合。

TAG 也可以用来进行自然语言句子的分析。在分析时，从包含树中词语的树结构开始，通过替换和接插操作，形成一个以 S 为根结点的树结构。这个树结构就是 TAG 的分析结果。

以上我们介绍和讨论了树邻接语法的基本定义和操作，在这些定义和操作的共同作用下，树邻接语法最终可以生成目标语言中的树集合，这些树又可以反过来分析目标语言中的串语言。虽然树邻接语法是在语言学领域里建立起来的，但它已经变成了数学和计算机科学领域都感兴趣的理论，研究 TAG 后产生了很重要的数学结果，这些结果将会反过来推动语言学的研究。因此，可以说 TAG 是一种语法、一种形式语言理论，一种自动机理论。总而言之，TAG 是自然语言处理的一种重要的形式模型，它体现了形式语言学、数学及计算机各领域之间的相互作用和完美结合，是计算语言学中的一朵绽放的奇葩。

8. 词汇化树邻接语法

从 TAG 发展至今，它的面貌已经发生了很大的变化，它对于自然语言的描述也越来越精细。最近又提出了“**词汇化树邻接语法**”（Lexicalized TAG，简称 LTAG），把词汇信息引入 TAG 的规则中。

LTAG 对于 TAG 的扩充主要在于把每一个初始树和辅助树都与某一个或某一些具体的单词关联起来，LTAG 树结构中带有词的结点叫做这个树的“**锚点**”（anchor）。下面是 LTAG 树的例子（图 7）：

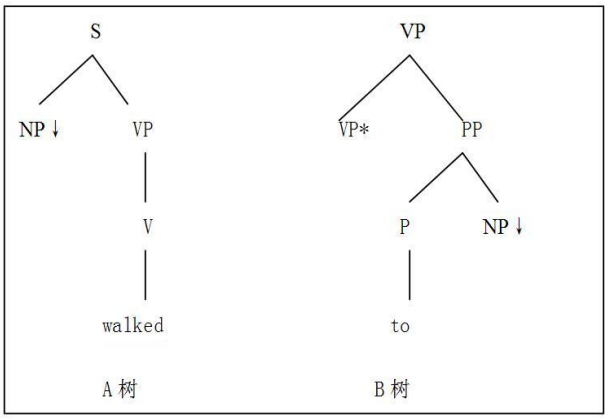


图 7 带锚点的树

A 树是一个初始树，它锚接在 walked 这个动词上，B 树是一个 VP 类型的辅助树，它锚接在 to 这个介词上。在 A 树中，由于锚点上的 walked 是一个不及物动词，因此，就限制了这个词不能带宾语，也就不可能由 A 树参与而生成出“John walked Beijing”这样的不合法的句子，但是，如果用 B 树与 A 树进行接插操作，可以得到“John walked to Beijing”这样合乎语法的句子。其树结构如下（图 8）：

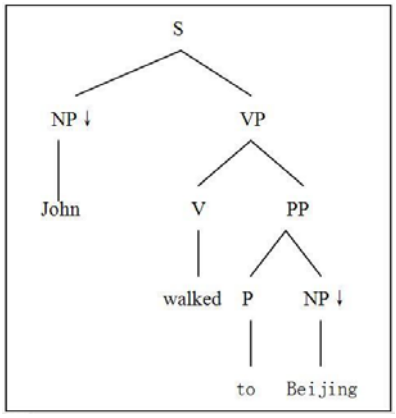


图 8 B 树与 A 树接插而生成的树

显而易见，由于在树结构中引入单词的信息，词汇化树邻接语法 LTAG 又进一步限制的短语结构语法过强的生成能力，提高了自然语言处理的精确度和效率，树邻接语法和词汇化树邻接语法在国外的自然语言处理中已经得到广泛的应用，显示出蓬勃的生命力。这种语法是比短语结构语法更好的自然语言处理的形式模型，在当代计算语言学的研究中，我们应当关注树邻接语法和词汇化树邻接语法，并且在中文信息处理中进一步发展这种富有生命力的形式语法。

参考文献

1. Joshi, A. K., Levy, L. S., and Takahashi, M., 1975, Tree Ajunct Grammar, *Journal of Computer and System Sciences*, 10(1), 55-75.
2. Joshi. A. K. and Y. Schabes, 1997, Tree-adjoining grammars, In G. Rosenberg and A. Salomaa (eds.), *Handbook of Formal Languages*, Berlin: Springer, 69-123.
3. Joshi. A. K. and K. Vijay-Shanker, Compositianl Semantics with lexicalized tree-adjoining

grammar (LTAG): how much underspecification is necessary? In H. C. Bunt and E. G. C. Thijsse (eds.), Proceedings of the 3rd International Workshop on Computational Semantics (IWCS-3) (Tilburg), 131-145.
4. 冯志伟，机器翻译研究，中国对外翻译出版公司，2004 年。
5. 冯志伟，自然语言处理的形式模型，中国科学技术出版社，2009 年。
6. Mitkov, R., Oxford Handbook of Computational Linguistics（牛津计算语言学手册），外语教学与研究出版社& Oxford University Press 联合出版，2009 年。

作者简介

冯志伟，男，教育部语言文字应用研究所研究员，博士生导师。研究兴趣：计算语言学，数理语言学，机器翻译，理论语言学，应用语言学。代表作：《数理语言学》《现代术语学引论》和《Chinesische Schriftzeichen－Vergangenheit und Gegenwart》(《汉字的历史和现状》，德文版，德国 TWV 出版社 1994 年版)。
电子邮件： zwfengde2010@hotmail.com
zwfengde2010@gmail.com
个人主页：http://www.lingviko.net/feng/feng.htm
http://ling.cuc.edu.cn/Staff/fengzhiwei/index.htm

FENG Zhiwei，male, is a research fellow of Institute of Applied Linguistics. His research interests include computational linguistics, machine translation, theoretical linguistics, applied linguistics. His major publications are: “Mathematical Linguistics”, “Introduction of Contemporary Terminology” and “Die chinesischen Schriftzeichen in Vergangenheit und Gengenwart” (Trier Wissenschftlicher Verlag, Trier, 1994).
Email: zwfengde2010@gmail.com
zwfengde2010@hotmail.com
Web Site：http://www.lingviko.net/feng/feng.htm
http://ling.cuc.edu.cn/Staff/fengzhiwei/index.htm

Tree Adjoining Grammar

Feng Zhiwei
（Hangzhou Normal University, Zhejiang Hangzhou 310036）

Abstract Tree Adjoining Grammar (TAG) is a kind of tree-based formal grammar, this paper compares the TAG and Phrase Structure Grammar (PSG), introduces TAG’ basic components and its operation model, especially two important operations: adjoining and substitution. The paper also introduces the conceptions as derivation tree, tree sets and string languages in TAG. The new development of TAG -- Lexicalized Tree-Adjoining Grammar (LTAG) is introduced finally.

Keywords Tree Adjoining Grammar (TAG), Lexicalized Tree-Adjoining Grammar (LTAG), adjoining, substitution, derivation tree