treeprint.sty (ver 2.2) マニュアル

treeprint.sty 作成　　マニュアル作成
伝 康晴*　　高木 一広†

平成 10 年 12 月 22 日

目 次

*ATR 音声翻訳通信研究所 (den@itl.atr.co.jp)
†神戸大学大学院文化学研究科 (takagi@lit.kobe-u.ac.jp)

1 はじめに

1.1 treeprint.sty とは

treeprint.sty は，木構造を簡単に，そして美しく描くためのスタイルファイルです．ユーザには必要最低限の指示しか要求しませんが，出来上がりは常に完璧です．

treeprint.sty は，CMU の Kevin Knight という人が書いた treeprint.lisp を基に，作者が LaTeX に書き直したものです．オリジナルの treeprint.sty は，LISP の S 式を LaTeX の picutre 環境に変換するプログラムでした．が，picure 環境では，描ける線分の角度などに制約があるため，仕上がりは必ずしも満足行くものではありませんでした．

そこで，作者が eepic.sty を使って LaTeX に移植し，線分の描画上の制約をなくすとともに，その後さらにいくつかの機能拡張を加え現在に至っています．

1.2 treeprint.sty を使うために

treeprint.sty を使うためには，予め環境変数 $TEXINPUTS で指定されたディレクトリに epic.sty, eepic.sty, eclarith.sty が インストールされている必要があります．もちろん treeprint.sty も忘れずに $TEXINPUTS で指定されたディレクトリにコピーしておいて下さい．

treeprint.sty は，あらゆる LaTeX に対応していますので，安心して使うことができます．日本語 TeX はアスキー版・NTT 版の両方で出力を確認しています．また LaTeX 2$_\varepsilon$ でも出力を確認しています．

LaTeX 209 で使う場合は，

```
\documentstyle[a4j,epic,eepic,eclarith,treeprint]{jarticle}
```

LaTeX 2$_\varepsilon$ で使う場合は

```
\documentclass[a4paper]{jarticle}
\usepackage{epic,eepic,eclarith,treeprint}
```

のように文書の冒頭にオマジナイを書いて下さい．

1.3 treeprint.sty を使うとウレシイこと

大阪大学の郡司先生が書かれた資料[1]には，木構造を出力するためのスタイルファイルとして，chomsky.sty[2], treemacros.sty[3]や tree-dvips.sty[4]などが紹介されていますが，マニュアル作成者が使ってみたところ，いずれも一長一短があるように思えます．

[1] 郡司隆男 (1993)「LaTeX で言語学の論文を書くために」
[2] by Michael Barr さん (from TeXhax 1989, issue 54.)
[3] by 橋田浩一さん (1992)
[4] by Thomas Rockiki さん

例えば，chomsky.sty は，比較的簡単に木構造を扱えますが，その分出力はどうしても大雑把になります．逆に treemacros.sty は出力は美しいのですが，枝の傾きなど，かなり細かい指定が必要で，文化系のユーザには敷居が高いと言えます．また，tree-dvips.sty は，比較的簡単に美しい出力が得られるのですが， dvips というドライバに依存しますので，これを使うためには，dvips が使える環境にあることが大前提になり，PC ユーザは PC-UNIX をインストールしない限り利用することができません．

これに対し，treeprint.sty は，

1. 最小限の手続きで最大限に美しい木構造を出力
2. ドライバには依存せず，依存するのは，eepic.sty というスタイルファイルのみ

という特徴を持ちます．

手続きの少なさについては，次節以降で紹介します．もしかしたら指定することが少ないことに不満を覚えるユーザもおられるかもしれません．

また，ドライバに依存しないことも treeprint.sty の強みの一つです．epic.sty, eclarith.sty, eepic.sty というスタイルファイルには依存しますので，これらのスタイルファイルを持っていない場合にどうするのだ，という問題がないわけではありません．が，このパッケージに附属していますので IP-reachable でない場合でも安心です．

唯一本当に問題があるとすると，依存しているスタイルファイルのうち，eepic.sty をサポートしていないドライバが若干あるということでしょうか．

ただ，DOS では dviprt, UNIX では dvi2ps という，広く使われているドライバが eepic.sty をサポートしていますので，おそらく，多くの環境で利用できることには間違いがありません．

もし不幸にしてお使いのドライバが eepic.sty をサポートしておらず，それ以外のドライバを入手する術がない場合でも御安心下さい．eepic.sty の代わりに ecleepic.sty を使って出力することができます．もっとも，ecleepic.sty は TEX の \special 命令を使っていませんので，枝の出力が乱れることはありますが，ドライバに関係なしに，とにかく出力はできます．

2　試しに使ってみましょう

2.1　えっ，コマンド三つだけでいいの？

この節では基本的なコマンドについて説明します．入力する必要があるのは，部分木が何本いるかということと，各ノードの範疇表示くらいです．

木構造の骨格部分は，`\(sub)tree{`⟨*Node_name*⟩`}`，`\end(sub)tree` の組合せで書いていきます．`\tree` コマンドはルート木を，`\subtree` コマンドは部分木を書くのに用いられます．`\subtree` はメモリが許す限り入れ子にできます．

終端ノードを描くには，`\leaf{`⟨*Terminal_name*⟩`}` コマンドを用います．`\leaf` コマンドは終端ノード名を引数にとり，`\(sub)tree` と `\end(sub)tree` の間におけば，自動的に終端ノードとして扱われます．

では，実際に上の三つのコマンド**だけ**を使って木構造を書いてみましょう．

```
\tree{S}
 \subtree{NP}\leaf{John}\endsubtree
 \subtree{VP}
  \subtree{V}\leaf{believes}\endsubtree
  \subtree{NP}
    \subtree{COMP}\leaf{that}\endsubtree
    \subtree{S}
      \subtree{NP}\leaf{Bill}\endsubtree
      \subtree{VP}
        \subtree{V}\leaf{loves}\endsubtree
        \subtree{NP}\leaf{Mary}\endsubtree
      \endsubtree
    \endsubtree
   \endsubtree
 \endsubtree
\endtree
```

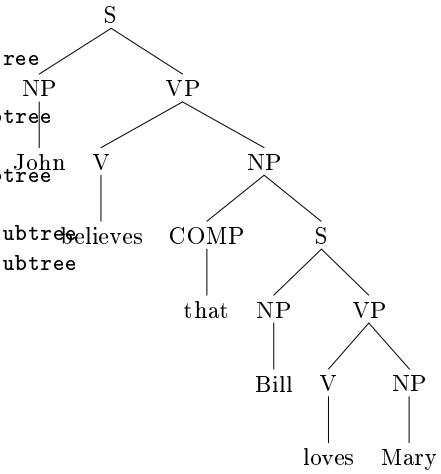

図 1: 基本的な木構造図

`\omittree` コマンドを使えば，句を終端ノードとして表示させることができます．

おそらく，普通の言語学の論文であれば[5]，これだけのコマンドがあれば必要にして十分だろうと思います．

[5] もっとも，最近の minimalist の論文で出てくる，移動の軌跡を含むようなものは，現在のバージョンではサポートされていません．が，作者は将来のバージョンではサポートする意向がないわけでもないようですので，作者宛に励ましのメールを送ると，案外すぐにインプリメントされるかもしれません．

```
\tree{S}
 \subtree{NP}\leaf{John}\endsubtree
 \subtree{VP}
  \subtree{V}\leaf{believes}\endsubtree
  \subtree{NP}
    \subtree{COMP}\leaf{that}\endsubtree
    \subtree{S}\omittree{Bill loves Mary}
    \endsubtree
  \endsubtree
 \endsubtree
\endtree
```

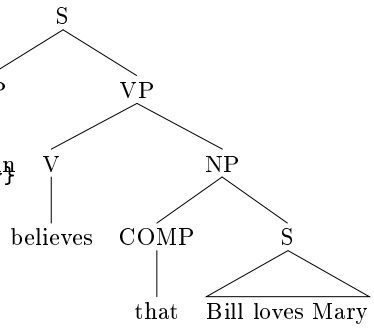

図 2: \omittree を使った基本的な木構造図

2.2 他にもいろいろできます

さて，前節で treeprint.sty を使った基本的な木構造の書き方を説明しました．もちろん，treeprint.sty の機能はこれだけにはとどまりません．色々と応用ができます．基本的に，

1. ノードに**任意の** LaTeX のテキストを書くことができる．
2. 枝の種類を変えることができる．
3. 枝にラベルを付けることができる．

という特徴をもちます．これを順番に説明していくことにします．

2.2.1 こんなノードも書けます

treeprint.sty を使えば，任意の LaTeX のテキストをノードにすることができます．試しに，HPSG の文献などでよくお目にかかる，素性構造を木構造で表したものを書いてみましょう[6]．

2.2.2 線の種類を変えてみました

1.3 節にも書きましたが，treeprint.sty は eepic.sty や eclarith.sty の機能を利用して木構造を書いています．つまり，これらのスタイルファイルで定義されている，豊富な描画機能がそのまま利用できるわけです．例えば，線の種類を変えてみたり[7]，先端に矢印をつけてみたり[8]，ということが，コマンドを利用して簡単にできます．

書式は，以下のように基本の書式に一つオプション引数を加えます．

- \subtree[**枝の種類**]{⟨*Node_name*⟩}
- \leaf[**枝の種類**]{⟨*Terminal_name*⟩}

[6] 素性構造の部分は lingmacros.sty の\outerfs というコマンドを使って書いています．
[7] これは eepic.sty の機能
[8] これは eclarith.sty の機能

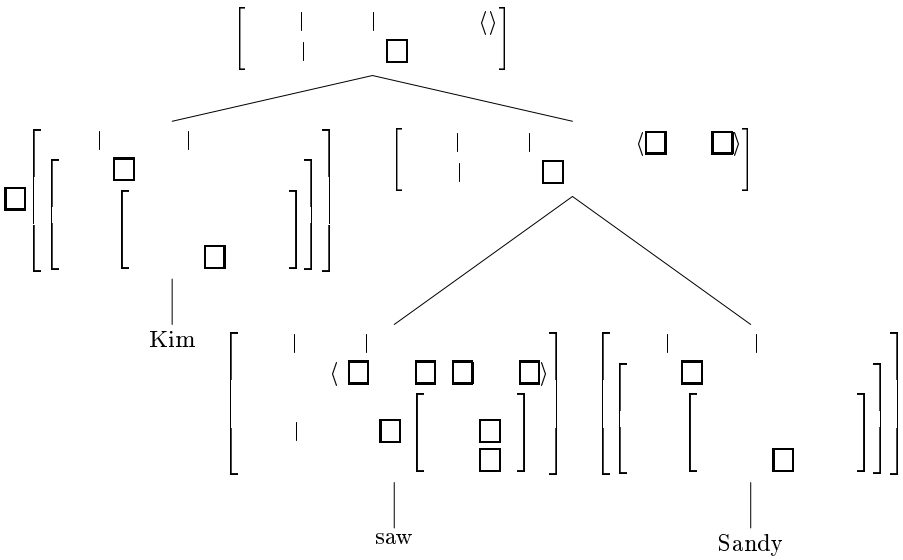

図 3: Pollard & Sag “Information-Based Syntax and Semantics vol.1”(p.102) より

この**枝の種類**というところに eepic.sty のコマンドを指定します．例えば，\dashline{点間隔} コマンドを用いると指定した点間隔の**破線**で，\dottedline{点間隔} コマンドを使うと指定した点間隔の**点線**で線が描かれます．なお，eepic.sty や eclarith.sty の使い方は，磯崎秀樹著『LaTeX 自由自在』pp.186–202 に分かりやすく書かれています．

さて，簡単な応用例として図 4 を作ってみましたが，こんなのを使うことがあるのでしょうか[9]．

treeprint.sty では，subtree, leaf コマンドに対して，予め枝の種類を組み込んだコマンドをいくつか定義しています[10]．表 1 にそれらを一覧にしておきます．

2.2.3 ノードにラベルを付けることもできます

treeprint.sty では，ノードにラベルをつけることもできます．
書式は，さきほどのよりオプション引数が更に二つ増えます．

- \tree{⟨*Node_name*⟩}[**ラベルの出力位置**][**ラベル**]
- \subtree[枝の種類]{⟨*Node_name*⟩}[**ラベルの出力位置**][**ラベル**]
- \leaf[枝の種類]{⟨*Terminal_name*⟩}[**ラベルの出力位置**][**ラベル**]

です．ラベルには任意の LaTeX のテキストが使えます．また，ラベルの出力位置は r で枝の右側に，c で枝の上に，l で枝の左側になります．

[9] \UpArrow 命令は，作者が eclarith.sty の \ArrowHead コマンドを利用して定義したコマンドです．
[10] \dotgap, \dashlength の初期値はそれぞれ 3 です．

```
\tree{S}
  \subtree{NP}
     \subtree[\UpArrow\dottedline{3}]{N}
        \leaf[\UpArrow\dottedline{3}]{ム
シ}
     \endsubtree
     \subtree[\DownArrow\dottedline{3}]{P}
        \leaf[\DownArrow\dottedline{3}]{
が}
     \endsubtree
  \endsubtree
  \subtree{VP}
     \subtree{NP}
        \subtree{N}\leaf{線}\endsubtree
        \subtree{P}\leaf{を}\endsubtree
     \endsubtree
     \subtree[\UpArrow\dashline{3}]{VP}
        \subtree[\dashline{\dashlength}]{V}
            \leaf[\dashline{\dashlength}]{
食べ}
        \endsubtree
        \subtree{AUX}\leaf{た}\endsubtree
     \endsubtree
  \endsubtree
\endtree
```

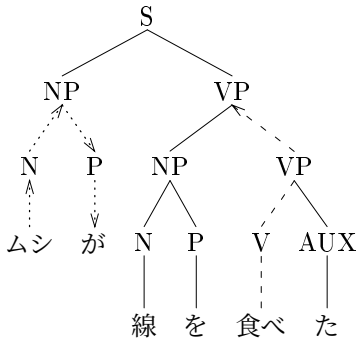

図 4: いくつかの種類の線を使ってみました

図 5 の出力[11]を参考にして下さい.

最後にこの節のまとめとして，図 6 に，自然言語処理の教科書などに出てくる，左隅構文解析の例をあげておきます．参考にして下さい.

[11] 実は，左の枝を揃えるために多少細工をしています．このままでは図と同じ出力は得られませんので注意して下さい．あくまでもラベルの付け方の参考にして下さい.

コマンド	枝の種類
\dottedsubtree{\dotgap}	\dotgap 間隔の点線の枝をもつ \subtree
\dashsubtree{\dashlength}	\dashlength 間隔の破線の枝をもつ \subtree
\downsubtree	枝が下向きの矢になっている \subtree
\upsubtree	枝が上向きの矢になっている \subtree
\updownsubtree	枝の上下に矢印がついている \subtree
\dottedleaf{\dotgap}	\dotgap 間隔の点線の枝をもつ \leaf
\dashleaf{\dashlength}	dashlength 間隔の破線の枝をもつ \leaf
\downleaf	枝が下向きの矢になっている \leaf
\upleaf	枝が上向きの矢になっている \leaf
\updownleaf	枝の上下に矢印がついている \leaf

表 1: 拡張された \subtree, \leaf コマンド

```
\tree{A$^{\prime\prime}$}
  \subtree{}[l][\sc head]
    \subtree{}[l][\sc head]
      \subtree{}[l][\sc head]
         \leaf{very}[l][\sc head]
         \leaf{many}[r][\sc comp]
      \endsubtree
      \leaf{times}[r][\sc comp]
    \endsubtree
    \leaf{as}[r][\sc comp]
  \endsubtree
  \subtree{A$^\prime$}[r][\sc comp]
    \omittree{productive}
  \endsubtree
\endtree
```

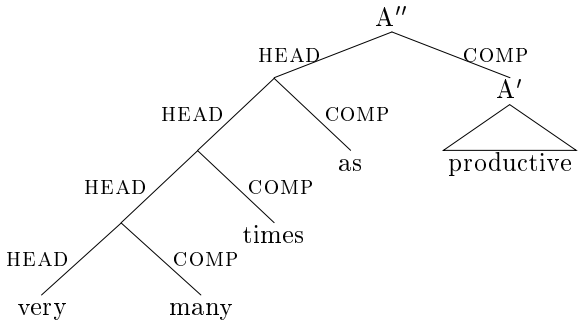

図 5: Pollard & Sag “Head-Driven Phrase Structure Grammar” (p.369) より

3 細かく調整したいんだけど

treeprint.sty では，出力形式を変えるオプションがいくつか用意されています．また好みに応じて，ノード間の幅を調整することもできます．

3.1 表示の仕方を変えるには

treeprint.sty では，表示の仕方を変えるコマンドとして，三項以上からなる要素の出力位置を調整する \balancing{on,off}，そして，終端ノードの並び方を指定する \(non)flatmode という二種類のコマンドを用意しています．

どちらのコマンドも，任意の位置で指定すれば，それ以降有効になります．\tree . . . \endtree コマンドの間で指定すると，その次の木構造から有効になります．

```
\tree{S}
  \upsubtree{NP}[l][\bf 2]
    \downleaf{John}[l][\bf 1]
  \endsubtree
  \downsubtree{VP}[r][\bf 3]
    \subtree[\UpArrow\dottedline{3}]{VP}[r][\bf 13]
      \upsubtree{V}[l][\bf 5]
        \downleaf{loves}[l][\bf 4]
      \endsubtree
      \downsubtree{NP}[r][\bf 6]
        \subtree[\UpArrow\dottedline{3}]{NP}[r][\bf 12]
          \upsubtree{Det}[l][\bf 8]
            \downleaf{a}[l][\bf 7]
          \endsubtree
          \downsubtree{N}[r][\bf 9]
            \subtree[\UpArrow\dottedline{3}]{N}[r][\bf 11]
              \downleaf{girl}[r][\bf 10]
            \endsubtree
          \endsubtree
        \endsubtree
      \endsubtree
    \endsubtree
  \endsubtree
\endtree
```

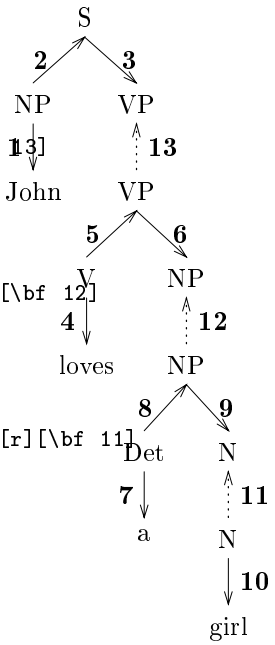

図 6: 左隅構文解析を意識した例

3.1.1 \balancingon, \balancingoff

\balancing{on,off} という命令で，三項以上からなる要素の配列を調節できます．図 7 と 図 8 の V″ に直接支配されている D″ の出力位置を比べてみて下さい．

図 7 のような場合，treeprint.sty は V と P′ 以下の終端ノードの長さから V″ ノードの木の角度を決めます．が，同時に，特に指定がなければ，treeprint.sty は，各要素をその**左**の要素に対して一定の間隔で出力しようとします．したがって，図 7 では，D″ の位置がどうしても左に寄ってしまうことになります．で，これを終端ノードの位置からもう一度計算し直してセンタリングするというのが，\balancingon のやっていることです．

デフォールトでは，\balancingoff になっていますが，これは\balancingon にすると，どうしても計算量が多くなり遅くなるからです (^^;)

3.1.2 \flatmode, \nonflatmode

ノードから終端までの枝の長さを一定にする (\nonflatmode) か，終端の出力位置を揃える (\flatmode) かというオプションです．図 9 と 図 10 とを比べてみて下さい．標準では \nonflatmode になっています．

TEX では文字は深さを持つために，\flatmode にした場合，終端ノードの文字列の組合せによっては，文字列の並びが凸凹になることがあります．その

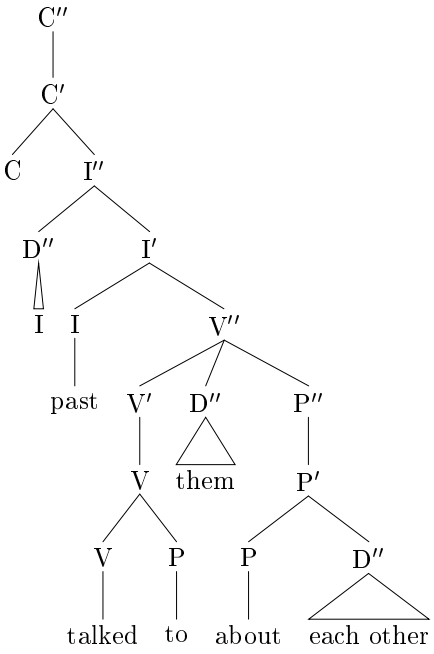

図 7: \balancingoff

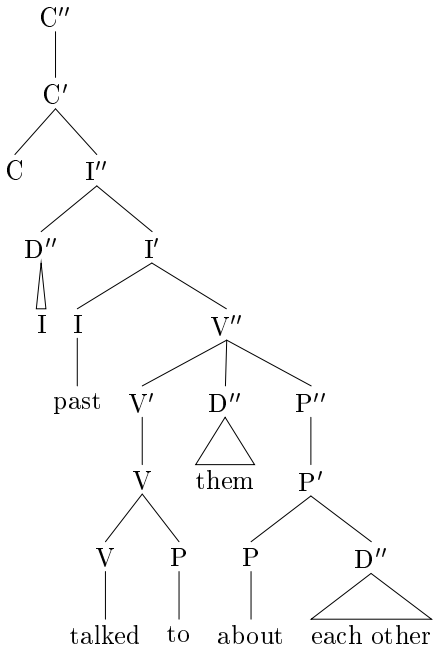

図 8: \balancingon

ようなときは，\mathstrut コマンドを使うと凸凹を防ぐことができます[12]．

3.2 長さを変えるには

多くの場合，特に指定しなくても treeprint.sty まかせにしておけば，美しい出力が得られるようになっていますが，複雑な木を書かせたり，凝ったことをしようとすると，場合によっては，手動で調整したくなることがあるかもしれません．

treeprint.sty で指定されている標準の長さは，表 2 のようになっています．長さを変更するには，\setlength{\branchlength}{6ex} というように宣言します．

[12] \strut コマンドを使ってもいいのですが，ちょっと上下が深すぎるように思います．どのように指定するかは，このマニュアルの図 10 のソースを見て下さい．

長 さ	意 味	初期値
\branchlength	枝の高さ	4ex
\internodesep	ノード間の幅	1em
\nodesep	枝とノードとの距離	.5ex
\arclabelsep	ラベルと枝の距離	1ex

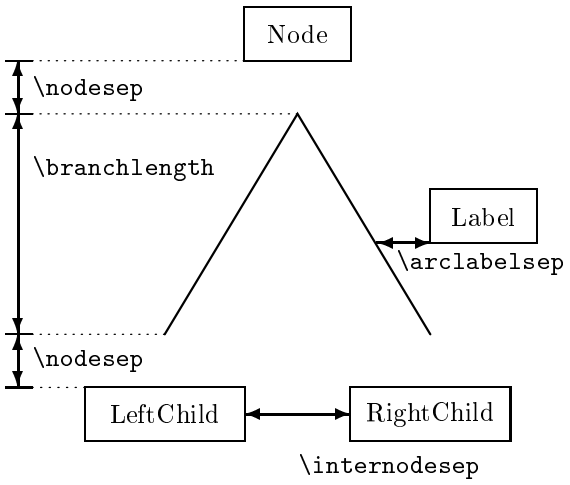

表 2: treeprint.sty の長さの初期値

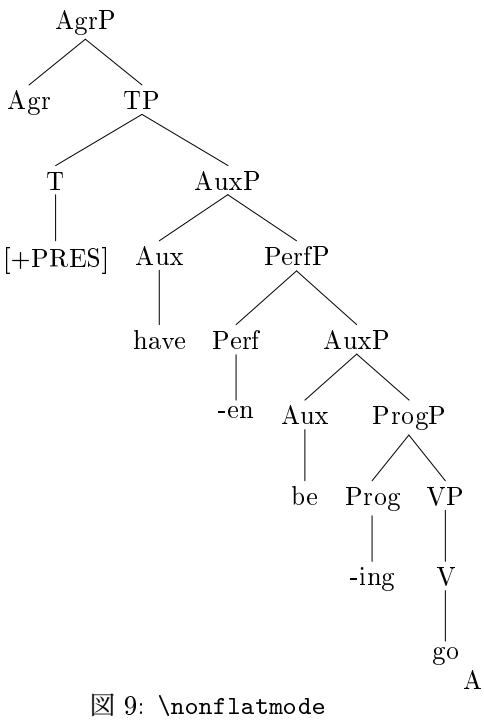

図 9: `\nonflatmode`

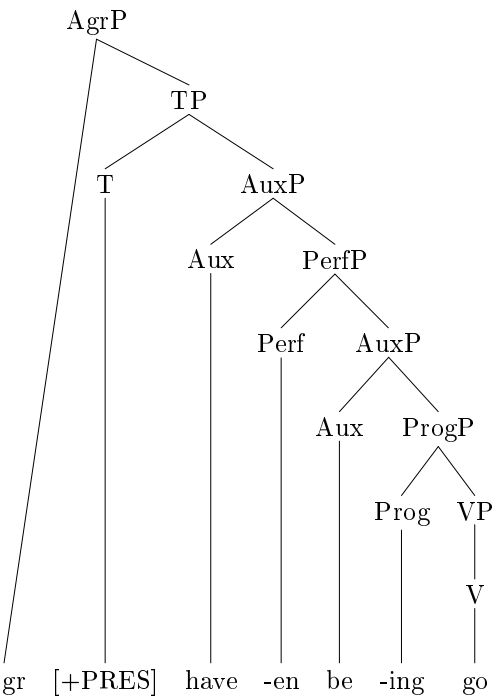

図 10: `\flatmode`

4 コマンド リファレンス

`treeprint.sty` のシンタクスは，BN 記法で記すと以下のようになります.

⟨Tree⟩	::=	`\tree` {⟨*Node*⟩}[⟨Pos⟩][⟨Label⟩] ⟨Children⟩ `\endtree`
⟨Chidlren⟩	::=	a sequence of ⟨Child⟩'s
⟨Child⟩	::=	⟨SubTree⟩ | ⟨Leaf⟩ | ⟨OmitTree⟩
⟨SubTree⟩	::=	`\subtree`[⟨Branch⟩]{⟨*Node*⟩}[⟨Pos⟩][⟨Label⟩]
		⟨Children⟩
		`\endsubtree`
⟨Leaf⟩	::=	`\leaf`[⟨Branch⟩]{⟨*Node*⟩}[⟨Pos⟩][⟨Label⟩]
⟨Omit Tree⟩	::=	`\omittree`⟨*Node*⟩
⟨Branch⟩	::=	an arbitrary epic/eepic drawline command
⟨Node⟩	::=	an arbitrary LaTeX text
⟨Pos⟩	::=	`r` | `c` | `l`
⟨Label⟩	::=	an arbitrary LaTeX text

表 3: `treeprint.sty` のシンタクス