自然言語解析のための MSLR パーザ・ツールキット

白井　清昭[†]　植木　正裕[††]　橋本　泰一[†]
徳永　健伸[†]　田中　穂積[†]

本論文では，我々が現在公開している自然言語解析用ツール「MSLR パーザ・ツールキット」の特徴と機能について述べる．MSLR パーザは，一般化 LR 法の解析アルゴリズムを拡張し，日本語などの分かち書きされていない文の形態素解析と構文解析を同時に行うツールである．MSLR パーザを用いて解析を行う際には，まず LR 表作成器を用いて，文法と接続表から LR 表を作成する．このとき，LR 表作成器は，接続表に記述された品詞間の接続制約を組み込んだ LR 表を生成する．このため，接続制約に違反する解析結果を受理しない LR 表が作られるだけでなく，LR 表の大きさを大幅に縮小することができる．次に，MSLR パーザは，作成された LR 表と辞書を用いて辞書引きによる単語分割と構文解析を同時に行い，その結果として構文木を出力する．さらに，MSLR パーザは，文中の括弧の組によって係り受けに関する部分的な制約が与えられた文を入力とし，その制約を満たす構文木のみを出力する機能を持つ．また，文脈依存性を若干反映した言語モデルのひとつである確率一般化 LR モデル (PGLR モデル) を学習し，個々の構文木に対して PGLR モデルに基づく生成確率を計算し，解析結果の優先順位付けを行う機能も持つ．

キーワード：　形態素解析，構文解析，一般化 *LR* 法，パーザ

MSLR Parser Tool Kit — Tools for Natural Language Analysis

Kiyoaki Shirai[†] , Masahiro Ueki[††] , Taiichi Hashimoto[†] ,
Takenobu Tokunaga[†] and Hozumi Tanaka[†]

In this paper, we describe a tool kit for natural language analysis, the MSLR parser tool kit. The 'MSLR parser' is based on the generalized LR parsing algorithm, and integrates morphological and syntactic analysis of unsegmented sentences. The 'LR table generator' constructs an LR table from a context free grammar and a connection matrix describing adjacency constraints between part-of-speech pairs. By incorporating connection matrix-based constraints into the LR table, it is possible to both reject any locally implausible parsing results, and reduce the size of the LR table. Then, using the generated LR table and a lexicon, the MSLR parser outputs parse trees based on morphological and syntactic analysis of input sentences. In addition to this, the MSLR parser accepts sentence inputs including partial syntactic constraints denoted by pairs of brackets, and suppresses the generation of any parse trees not satisfying those constraints. Furthermore, it can be trained according to the probabilistic generalized LR (PGLR) model, which is a mildly context sensitive language model. It can also rank parse trees in order of the overall probability returned by the trained PGLR model.

KeyWords:　*morphological analysis, syntactic analysis, generalized LR method, parser*

1　はじめに

我々は，1998 年 10 月から自然言語解析用ツール「MSLR パーザ・ツールキット」を公開している [1]．MSLR パーザ (Morphological and Syntactic LR parser) は，一般化 LR 法の解析アルゴリズムを拡張し，単語区切りのない言語 (日本語など) を主に対象とし，形態素解析と構文解析を同時に行うパーザである[2]．本論文では，MSLR パーザ・ツールキットの特徴と機能について述べる．

MSLR パーザを用いて文を解析する場合には，以下の 3 つが必要になる．

文法　品詞を終端記号とする文脈自由文法．主に構文解析に用いる．

辞書　単語とそれに対応した品詞を列挙したデータで，形態素解析の基本単位を集めたものである．辞書の品詞体系は文法の品詞体系と一致していなければならない．

接続表　品詞間の接続制約を記述した表．品詞間の接続制約とは，ある 2 つの品詞が隣接できるか否かに関する制約である．

本ツールキットでは，文法・辞書・接続表を自由に入れ換えることができる．すなわち，ユーザが独自に開発した文法や辞書を用いて，MSLR パーザによって文の解析を行うことが可能である．また，MSLR パーザ・ツールキットには日本語解析用の文法，辞書，接続表が含まれている．したがって，文法等を持っていないユーザでも，ツールキットに付属のものを用いて日本語文の形態素・構文解析を行うことができる．

MSLR パーザは C 言語で実装され，動作する OS は unix のみである．具体的には，以下の OS で動作することが確認されている．

- SunOS 5.6
- Digital Unix 4.0
- IRIX 6.5
- FreeBSD 3.3
- Linux 2.2.11, Linux PPC(PC-Mind 1.0.4)

MSLR パーザを動作させるために必要なメモリ使用量・ディスク使用量は，使用する文法や辞書の規模に大きく依存する．例えば，ツールキットに付属の日本語解析用文法 (規則数 1,408) と辞書 (登録単語数 241,113) を用いる場合，50Mbyte のメモリと 10Mbyte のディスク容量を必要

† 東京工業大学 大学院情報理工学研究科 計算工学専攻, Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology
†† 国立国語研究所 日本語教育センター 日本語教育普及指導部 日本語教育教材開発室, Teaching Materials Development Section, Department of Educational Support Services, Center for Teaching Japanese as a Second Language, The National Language Research Institute

1 `http://tanaka-www.cs.titech.ac.jp/pub/mslr/`

2 MSLR パーザは，分かち書きされた文 (英語文など) を解析する機能も持っているが，もともとは単語区切りのない文を解析することを目的に作られた．

とする．

本ツールキットを用いた形態素・構文解析の流れを図 1 に示す．MSLR パーザの解析アルゴリズムは一般化 LR 法に基づいているため，まず最初に LR 表作成器を用いて，文法と接続表から LR 表を作成する．MSLR パーザは，作成された LR 表と辞書を参照しながら入力文の形態素・構文解析を行い，解析結果 (構文木) を出力する．

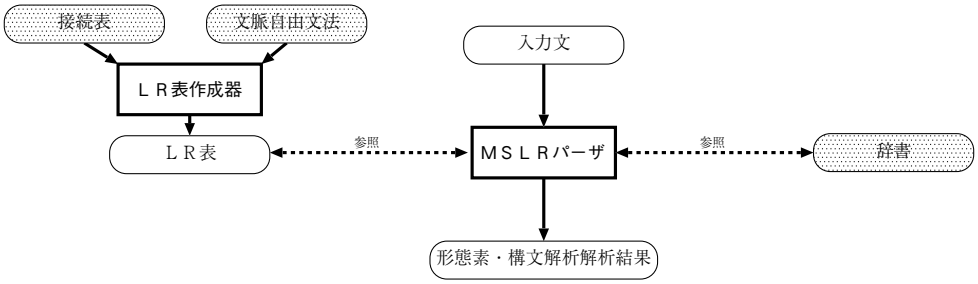


図 1　MSLR パーザを用いた形態素・構文解析の流れ

本ツールキットの主な特徴と機能は以下の通りである．

- MSLR パーザは，形態素解析と構文解析を同時に行う．まず最初に形態素解析を行い，その出力をもとに構文解析を行う逐次的な方法では，形態素解析の段階では文法などの構文的な制約を考慮しない場合が多く，その後の構文解析の段階で不適当と判断されるような無駄な解析結果も出力される．これに対し，MSLR パーザは形態的な情報 (辞書，接続表) と構文的な情報 (文法) を同時に用いて解析を行うため，このような無駄な解析結果を生成することはない．
- LR 表作成器は，接続表に記述された品詞間の接続制約を組み込んだ LR 表を作成する．すなわち，LR 表を作成する段階で品詞間の接続制約を考慮し，接続制約に違反する構文木を受理しない LR 表を作る．さらに，品詞間の接続制約を組み込んだ場合，接続制約を組み込まない場合と比べて LR 表の状態数・動作数を減らすことができ，メモリ使用量も小さくすることができるという利点がある．
- 品詞間の接続制約は，接続表という形式で記述する代わりに，文法に組み込むことも可能である．しかしながら，接続制約を文法に組み込んだ場合，規則数が組み合わせ的に増大する．このため，文法作成者の負担が大きくなり，また作成される LR 表の大きさも大きくなるために望ましくない．このような理由から，本ツールキットでは，接続表と文法を独立に記述する枠組を採用している．
- 平文を入力とした解析の他に，係り受けに関する部分的な制約を加えた文を入力とした解析を行うことができる．例えば,「太郎が渋谷で買った本を借りた」という文を解析す

る際に，以下のような括弧付けによる制約を付けた文が入力されたときには，括弧付けと矛盾した解析結果は出力しない．

[太郎が渋谷で買った] 本を借りた

すなわち,「太郎が」が「借りた」に係る以下のような解析結果は，A の括弧付けが入力の括弧付けと矛盾 (交差) しているために出力しない．

[[太郎が][A [[渋谷で][買った]][[本を][借りた]] A]

この機能は，例えば前編集により係り受けに関する部分的な制約をあらかじめ文に付加してから解析を行い，構文的曖昧性を抑制する場合などに利用できる．

- 確率一般化 LR モデル (Inui, Sornlertlamvanich, Tanaka, and Tokunaga 1998; Sornlertlamvanich, Inui, Tanaka, Tokunaga, and Toshiyuki 1999) (Probabilistic Generalized LR Model，以下 PGLR モデル) を取り扱うことができる．PGLR モデルとは，一般化 LR 法の枠組において構文木の生成確率を与える確率モデルである．PGLR モデルに基づく構文木の生成確率は，統計的な意味での正しさの尺度を構文木に与えることができるので，構文的な曖昧性の解消に利用することができる．

以下では，ここに挙げた本ツールキットの特徴と機能について詳しく説明する．2 節では品詞間の接続制約を組み込む LR 表作成器について述べ，3 節では MSLR パーザの概略について述べる．最後に 4 節で本論文のまとめと MSLR パーザ・ツールキットの今後の開発方針について述べる．

2　LR 表作成器

本節では，MSLR パーザ・ツールキットにおける LR 表作成器の機能と特徴について詳しく説明する．

2.1　3 種類の LR 表を作成する機能

一般化 LR 法で用いられる LR 表には，SLR (Simple LR), CLR (Canonical LR), LALR (Lookahead LR) の 3 種類がある．我々の LR 表作成器は，これら 3 種類の LR 表を作成する機能を持つ．

実際の自然言語文の解析では，最も状態数の少ない LALR が用いられる場合が多い．したがって，以後 LR 表といえば LALR を意味するものとする．これらの LR 表の違いの詳細については文献 (Aho, Sethi, and Ullman 1985) を参照していただきたい．

2.2　品詞間の接続制約を組み込む機能

本ツールキットにおける LR 表作成器の最も大きな特徴は，LR 表に品詞間の接続制約を反映させることができる点にある．品詞間の接続制約を LR 表に反映させるということは，接続制約に違反する構文木を生成する動作を LR 表からあらかじめ除去することに相当する．

このことを図 2 の文法 CFG_1 を例に説明する[3]．CFG_1 において，書き換え規則の右側にある数字は規則番号を表わす．また，終端記号は品詞である．CFG_1 から通常の LR 表作成アルゴリズムによって作成された LR 表を図 3 に示す．但し，図 3 の LR 表は action 部のみであり，goto 部は省略されている．今，この LR 表に図 4 の接続表に記述された接続制約を反映させることを考える．図 4 の接続表において，行列要素 (i, j) が 1 なら i 行目の品詞 x_i と j 列目の品詞 x_j がこの順序で連接可能であることを示し，(i, j) が 0 なら x_i と x_j が連接不可能であることを意味する．また，“$” は文末を表わす特殊な品詞である．

S → VP	(1)	VS1 → vs_1	(10)
VP → PP VP	(2)	VS → vs_5k	(11)
PP → VP PP	(3)	VS → vs_5m	(12)
VP → V AX	(4)	VS → vs_5w	(13)
V → VS VE	(5)	VE → ve_i	(14)
V → VS1	(6)	VE → ve_ki	(15)
PP → N P	(7)	VE → ve_ma	(16)
N → noun	(8)	AX → AX aux	(17)
P → postp	(9)	AX → aux	(18)

図 2　文法の例: CFG_1

CFG_1 では，VS を構成する品詞として vs_5k, vs_5m, vs_5w が，VE を構成する品詞として ve_i, ve_ki, ve_ma があるので，規則 (5) から，V を構成する品詞列は $3 \times 3 = 9$ 通りあることがわかる．これに対し，図 4 の接続表を考慮した場合，これら 9 通りの品詞列のうち “vs_5k ve_ki”，“vs_5m ve_ma”，“vs_5w ve_i” の 3 組だけが接続制約を満たす．したがって，これら以外の品詞列は受理すべきではない．

ここで，図 3 の LR 表の状態 4，先読み記号 ve_i の欄にある re 11 という reduce 動作に着目する．re 11 は，CFG_1 における規則 (11) に対応した部分木を作ることを意味する (図 5)．と

3 CFG_1 における各記号のおおまかな意味は以下の通りである．S=文, VP=動詞句, PP=後置詞句, V=動詞, VS1=一段動詞語幹, VS=動詞語幹, VE=動詞語尾, N=名詞, P=助詞, AX=助動詞列 (以上，非終端記号)．vs_1=一段動詞語幹, vs_5k=カ行五段動詞語幹, vs_5m=マ行五段動詞語幹, vs_5w=ワ行五段動詞語幹, ve_i=動詞語尾イ, ve_ki=動詞語尾キ, ve_ma=動詞語尾マ, noun=名詞, postp=助詞, aux=助動詞 (以上，終端記号 (品詞))．

	vs_1	vs_5k	vs_5m	vs_5w	ve_i	ve_ki	ve_ma	aux	noun	postp	$
0	sh1	sh4	sh3	sh2					sh11		
1								re10			
2					re13	*re13	*re13				
3					*re12	*re12	re12				
4					*re11	re11	*re11				
5								sh13			
6	sh1	sh4	sh3	sh2					sh11		
7										sh16	
8								re6			
9					sh20	sh19	sh18				
10	sh1	sh4	sh3	sh2					sh11		re1
11										re8	
12											acc
13	re18	re18	re18	re18				re18	re18		re18
14	re4	re4	re4	re4				sh24	re4		re4
15	re2/sh1	re2/sh4	re2/sh3	re2/sh2					re2/sh11		re2
16	re9	re9	re9	re9					re9		
17	re7	re7	re7	re7					re7		
18								re16			
19								re15			
20								re14			
21								re5			
22	sh1	sh4	sh3	sh2					sh11		
23	re3/sh1	re3/sh4	re3/sh3	re3/sh2					re3/sh11		
24	re17	re17	re17	re17				re17	re17		re17

図 3　CFG_1 から生成される LR 表 (action 部のみ)

	vs_1	vs_5k	vs_5m	vs_5w	ve_i	ve_ki	ve_ma	noun	postp	aux	$
vs_1	0	0	0	0	0	0	0	1	0	1	1
vs_5k	0	0	0	0	0	1	0	0	0	0	0
vs_5m	0	0	0	0	0	0	1	0	0	0	0
vs_5w	0	0	0	0	1	0	0	0	0	0	0
ve_i	0	0	0	0	0	0	0	0	0	1	0
ve_ki	0	0	0	0	0	0	0	0	0	1	0
ve_ma	0	0	0	0	0	0	0	0	0	1	0
noun	1	1	1	1	0	0	0	1	1	1	1
postp	1	1	1	1	0	0	0	1	1	0	1
aux	1	1	1	1	0	0	0	1	1	1	1

図 4　接続表の例

ころが，先読み記号が ve_i であることから，“vs_5k ve_i” という品詞列に対してこの動作を実行することになるが，この品詞列は図 4 の接続制約に違反する．同様に，図 3 において，“*” のついた動作もまた接続制約に違反する動作である．したがって，このような動作を事前に LR 表から削除しておけば，接続制約に違反する解析結果の生成を防ぐことができる．

接続制約に違反する動作を LR 表から除去する方法としては，まず図 3 のように接続制約を考慮しない LR 表を作成してから，接続制約に違反する動作を LR 表から削除する方法が考え

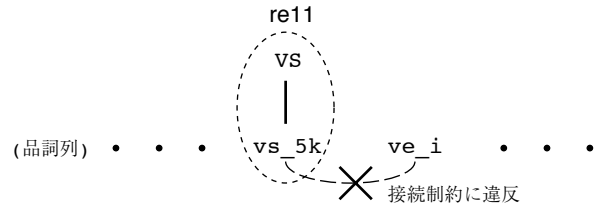


図 5　接続制約に違反する reduce 動作

$$\mathrm{V} \rightarrow \mathrm{VS\ VE} \quad (5) \qquad \Rightarrow \qquad \begin{array}{ll} \mathrm{V} \rightarrow \mathrm{vs\_5k\ ve\_ki} & (5\text{-}1) \\ \mathrm{V} \rightarrow \mathrm{vs\_5m\ ve\_ma} & (5\text{-}2) \\ \mathrm{V} \rightarrow \mathrm{vs\_5w\ ve\_i} & (5\text{-}3) \end{array}$$

図 6　接続制約を反映した文法規則

られる．しかしながら，文法の規模が大きくなると，接続制約を考慮しない LR 表の大きさが非常に大きくなるために望ましくない．これに対して，本ツールキットでは，LR 表を作成する段階で接続制約を考慮し，接続制約に違反する動作を除いた LR 表を直接生成する方法を採用している．接続制約を組み込みながら LR 表を作成するアルゴリズムの詳細については文献 (Li 1996) を参照していただきたい．

接続制約を LR 表に組み込む主な利点としては以下の 3 つが挙げられる．

(1)　接続制約を事前に組み込んだ LR 表を用いて解析を行った場合，解析時には品詞間の連接可能性をチェックする必要がないので，解析時の効率を上げることができる．

(2)　接続制約に違反する構文木を生成する動作を LR 表から除去することにより，LR 表の状態数・動作数を大幅に減らし，メモリ使用量を小さくすることができる．

(3)　品詞間の接続制約は，接続表として記述してから LR 表に組み込む代わりに，書き換え規則の細分化によって組み込むこともできる．例えば，CFG_1 の例では，規則 (5) の代わりに，図 6 に挙げる 3 つの規則を導入すれば，接続制約を満たす品詞列のみ受理することができる．しかしながら，このように接続制約を組み込んだ文法を作成することは，規則数が組み合わせ的に増大するために望ましくない．品詞間の接続制約は，接続表として文法とは独立に記述し，LR 表を作成する段階で接続制約を組み込む方が，最終的に得られる LR 表の状態数・動作数も少なく，メモリ使用量を小さくすることができる．また，文法記述者の負担も減らすことができる．

2.3　評価実験

LR 表に品詞間の接続制約を組み込む効果を調べる簡単な実験を行った．本ツールキットに付属されている日本語解析用の文法と接続表を用いて，品詞間の接続制約を組み込む場合と組み込まない場合の LR 表を比較した．使用した文法の規則数は 1,408，非終端記号数は 218，終端記号数は 537 である．実験に使用した計算機は Sun Ultra Enterprise 250 Server(主記憶 2GB, CPU 周波数 300MHz) である．結果を表 1 に示す．

表 1　品詞間の接続制約を LR 表に組み込むことの効果

	CPU 時間	状態数	動作数
接続制約なし	42.1(sec.)	1,720	379,173
接続制約あり	45.4(sec.)	1,670	197,337

表 1 において,「CPU 時間」は LR 表作成に要した CPU 時間を,「状態数」は作成された LR 表の状態の数を,「動作数」は作成された LR 表の動作 (shift 動作と reduce 動作) の数を示している．この表から，品詞間の接続制約を組み込むことによって，状態数はほとんど変わらないが，動作数は約半分に減ることがわかる．したがって，LR 表のために必要なメモリ使用量を大幅に縮小することができる．一方,「CPU 時間」は，接続制約を考慮する場合としない場合とでそれほど大きな差は見られなかった．一般に，接続制約を組み込む場合は，品詞間の連接可能性を調べながら LR 表を作成するために，それに要する時間は長くなることが予想される．しかしながら，接続制約に違反する無駄なアイテムが生成されなくなることから，LR 表作成に要する時間が短縮される効果も生じる．そのため，LR 表作成時間が劇的に増大するわけではないことが実験的に確かめられた．

3　MSLR パーザ

本節では，MSLR パーザの機能と特徴について概説する．

3.1　形態素解析と構文解析を同時に行う機能

1 節で述べたように，MSLR パーザは形態素解析と構文解析を同時に行う (Tanaka, Tokunaga, and Aizawa 1995)．また，形態素・構文解析結果として構文木を出力する．例えば，図 2 の文法 (CFG_1)，図 4 の接続表，図 7 の辞書を用いたときの「あいこにたのまれた」という文の解析結果 (構文木) を図 8 に示す．実際には，MSLR パーザは以下のような括弧付けで表現された構文木を出力する．

単語	品詞	単語	品詞
あ	vs_5k, vs_5w	たの	vs_5m
あいこ	noun	に	postp, vs_1
い	ve_i	の	vs_5m
き	ve_ki	ま	ve_ma
た	aux	れ	aux

図 7　辞書の例

```
[<S>,[<VP>,[<PP>,[<N>,[noun, あいこ]],[<P>,[postp, に]]],[<VP>,[<V>,[<VS>,
[vs_5m, たの]],[<VE>,[ve_ma, ま]]],[<AX>,[<AX>,[aux, れ]],[aux, た]]]]]
```

解析結果が複数ある場合には，その中から N 個の構文木をランダムに選んで出力する．ただし，3.3 項で述べる PGLR モデルを用いる場合には，構文木の生成確率の大きい上位 N 個の構文木を取り出すことができる．また，N の値は起動時のオプション指定により変更できる．

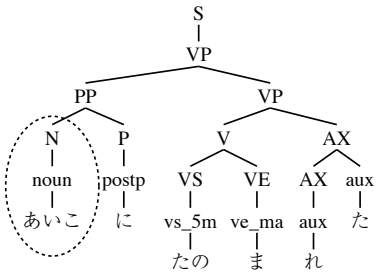


図 8「あいこにたのまれた」の解析結果

MSLR パーザのアルゴリズムは，一般化 LR 法の構文解析アルゴリズムを拡張したものである．一般化 LR 法が通常は品詞列を入力とするのに対して，MSLR パーザは文字列を入力とし，辞書引きによる単語分割と構文解析を同時に行う．以下，一般化 LR 法と MSLR パーザの解析アルゴリズムとの違いを簡単に説明する．MSLR パーザの解析アルゴリズムの詳細については文献 (Tanaka et al. 1995) を参照していただきたい．

(1)　入力文が与えられたとき，品詞と品詞の間に位置番号をつける代わりに，図 9 のように入力文の文字間に位置番号をつける．

(2)　解析が位置 i まで進んだとき，位置 i から始まる全ての単語を辞書引きし，その結果をスタックに登録する．例えば，図 9 の例文を図 7 の辞書を用いて解析した場合，位置 0 では “(あ,vs_5k)”,“(あ,vs_5w)”,“(あいこ,noun)” という 3 つの品詞付けの結果が解析スタックに登録される．これらの品詞付けの結果は，通常の一般化 LR 法におけ

あ い こ に た の ま れ た
(位置番号)　0　1　2　3　4　5　6　7　8　9

図 9　MSLR パーザにおける位置番号のつけ方

る多品詞語と全く同様に取り扱われる．

(3)　shift 動作を実行して先読み記号をスタックにプッシュする際には，その品詞を構成する文字列の一番最後の位置まで解析スタックを延ばす．例えば，位置 0 で vs_5k という先読み記号 (品詞) をプッシュする際には，vs_5k が位置 0〜1 に位置する単語「あ」の品詞であるので，スタックの先頭を位置 1 まで延ばす．そして，位置 1 から始まる単語の辞書引き結果をもとに以後の解析をすすめる．同様に，位置 0 で noun という品詞をプッシュする際には，noun が位置 0〜3 に位置する単語「あいこ」の品詞であるので，スタックの先頭を位置 3 まで延ばす．以後の解析は，位置 3 から始まる単語の辞書引き結果をもとに進められる．

例文「あいこにたのまれた」を解析する際，形態素解析結果の候補としては以下の 2 つがある．

a.　(あいこ,noun)(に,postp)(たの,vs_5m)(ま,ve_ma)(れ,aux)(た,aux)

b.　(あいこ,noun)(に,vs_1)(た,aux)(の,vs_5m)(ま,ve_ma)(れ,aux)(た,aux)

文法 CFG_1 は b. の品詞列を受理しないが，形態素解析と構文解析を逐次的に行う方法では，形態素解析結果の候補として a., b. ともに出力し，それぞれの品詞列に対して構文解析が試みられる．これに対し，MSLR パーザは形態素解析と構文解析を同時に行い，文法に記述された構文的な制約で排除される形態素解析の結果を早期に取り除くことができるため，解析効率がよい．例えば，位置 3 まで解析が進んだとき,「あいこ」という文字列が図 8 の点線で囲まれた部分木を構成することがわかっている．このとき，位置 3 から始まる単語を辞書引きする際に，品詞列 b. は受理されないという文法的な制約から，“(に,vs_1)” という品詞付けが適切でないことがわかる．具体的には，位置 3 におけるスタックトップの状態 7 において，“vs_1” を先読み記号とする動作が図 3 の LR 表に存在しないことから，“(に,vs_1)” という辞書引き結果を含む解析はこの時点で中断される．したがって，誤りである形態素解析結果の候補 b. を早期に取り除くことができる．このことは，MSLR パーザの大きな特徴の 1 つである．

3.2　括弧付けによる制約のついた入力文を解析する機能

MSLR パーザは括弧付けによる制約を加えた文を解析することができる．具体的には，MSLR パーザは以下のような文字列を入力として，括弧付けに矛盾しない解析結果のみを出力する機能を持つ．

[*, 太郎が渋谷で買った] 本を借りた

この例では括弧による制約はひとつしかないが，括弧による制約は複数あってもよい．また，複数の制約が入れ子になっても構わない．以下に例を挙げる．

[*, 太郎が [*, 渋谷で買った]][*, 本を借りた]

上記の入力例において，“*” は括弧で示された範囲を支配する非終端記号に特に制約がないことを表わしている．これに対し，“*” の位置に非終端記号を指定することにより，括弧に矛盾する解析結果だけでなく，括弧で囲まれた文字列を支配する非終端記号を限定することもできる．例えば，以下のような入力に対して，MSLR パーザは「あいこに」を支配する非終端記号が “<PP>” となる解析結果のみを出力する．

[<PP>, あいこに] たのまれた

括弧付けによる制約を取り扱う機能は，前編集によりあらかじめ部分的な制約を付加する際に利用することができる．構文解析を完全に自動で行うのではなく，インタラクティブに人間の知識を利用しながら半自動的に構文解析を行うことは，解析精度を向上させる有効な手段のひとつである．解析を行う前に，係り受けに関する部分的な制約をうまく人手で与えれば，構文的曖昧性を激的に減らすことができ，結果として構文解析の精度を飛躍的に向上させることが期待できる．

3.3 PGLR モデルを取り扱う機能

PGLR モデル (Inui et al. 1998) は，一般化 LR 法の枠組に基づいて構文木の生成確率を与える確率モデルである．PGLR モデルにおける構文木の生成確率は，構文木を作り出す際に実行される LR 表上の動作 (shift 動作もしくは reduce 動作) の実行確率の積として推定される．この生成確率は，生成される複数の構文木の中から最も正しい構文木を選択する構文的曖昧性解消に利用できる．ここで注意すべき点は，PGLR モデルによって与えられる構文木の生成確率は品詞を葉とする構文木の生成確率だということである．すなわち，単語の導出確率や単語の共起関係などの語彙的な統計情報は考慮されていない[4]．

LR 表の動作の実行確率には若干の文脈依存性が反映されていると考えられる．したがって，PGLR モデルは，文脈自由な言語モデルである確率文脈自由文法よりも推定パラメタ数は多くなるが，文脈依存性が考慮されたより精密なモデルを学習することが可能であり，構文的曖昧性解消の精度も向上することが実験的にも確かめられている (Sornlertlamvanich et al. 1999)．

本ツールキットでは，PGLR モデルを学習する機能，及び PGLR モデルによる構文木の生成確率を計算する機能を備えている．以下，それぞれの機能の概要について説明する．

4 PGLR モデルと，PGLR モデルとは独立に学習された語彙的な統計情報を組み合わせて構文解析を行う試みも行われている (白井, 乾, 徳永, 田中 1998).

3.3.1 PGLR モデルの学習について

PGLR モデルの学習は，LR 表上の各動作の実行確率を推定することにより行われる．動作の実行確率の推定に必要なものは，構文木が付与された構文木付きコーパスである．まず，例文に付与された構文木に対して，構文木を生成する際に実行する LR 表上の動作の使用回数 $C(s_i, l_j, a_k)$ を数え上げる．ここで，s_i は LR 表における状態を，l_j は先読み記号を，a_k は動作を表わし，$C(s_i, l_j, a_k)$ は，状態が s_i で先読み記号が l_j のときに動作 a_k が実行された回数を表わす．

LR 表上の各動作の実行確率は式 (1)(2) によって推定する．

$$P(l_j, a_k|s_i) = \frac{C(s_i, l_j, a_k)}{\sum_{j,k} C(s_i, l_j, a_k)} \quad if \quad s_i \in S_s \tag{1}$$

$$P(a_k|s_i, l_j) = \frac{C(s_i, l_j, a_k)}{\sum_{k} C(s_i, l_j, a_k)} \quad if \quad s_i \in S_r \tag{2}$$

式 (1)(2) において，S_s は shift 動作直後に到達する状態の集合，S_r はそれ以外の状態の集合を表わす．LR 表における全ての状態は S_s または S_r のどちらか一方に必ず属する．図 3 の LR 表の例では，$S_s = \{0, 1, 2, 3, 4, 11, 13, 16, 18, 19, 20, 24\}$, $S_r = \{5, 6, 7, 8, 9, 10, 12, 14, 15, 17, 21, 22, 23\}$ である．初期状態 0 は S_s に属することに注意していただきたい．

式 (1) は，$s_i \in S_s$ のときには，状態 s_i で実行されうる全ての動作で実行確率を正規化することを意味する．言い換えれば，LR 表における同じ行に属する動作の実行確率の和は 1 となる．例えば，図 3 の LR 表の状態 0 にある 5 つの shift 動作は，これらの実行確率の和が 1 になるように正規化される．これに対して式 (2) は，$s_i \in S_r$ のときには，状態 s_i，先読み記号 l_i のときに実行されうる全ての動作で実行確率を正規化することを意味する．すなわち，LR 表における同じマス目に属する動作の実行確率の和は 1 となる．例えば，図 3 の LR 表の状態 15，先読み記号 vs_1 の欄にある 2 つの動作 (re 2 と sh 1) の実行確率は，これらの和が 1 になるように正規化される．また，S_r に属する状態の場合，shift/reduce コンフリクトがない限り，その状態に属する動作の実行確率は必ず 1 となる．

本ツールキットにおける PGLR モデル学習の手続きは以下の通りである．まず，MSLR パーザは，構文解析を行う際に，LR 表の各動作の使用回数を出力する機能を持っている．さらに，3.2 項で述べた括弧付けによる制約を取り扱う機能を利用し，訓練用コーパスに付与された構文木を入力として解析を行うことにより，訓練用コーパス中の構文木を生成する際に使われた各動作の使用回数 $C(s_i, l_j, a_k)$ を求めることができる．また本ツールキットには，このようにして得られた $C(s_i, l_j, a_k)$ から式 (1)(2) に従って各動作の実行確率を推定し，その実行確率が付与された LR 表を作成するツールが含まれている．このツールは，パラメタ推定の平滑化のために，LR 表に登録されている全ての動作の実行回数にある一定の頻度を加える機能を備えている．

表 2　解析実験の結果

	Set A	Set B
平均単語数	8.12	19.6
平均解析木数	13.1	15,500
平均解析時間 (ms)	6.53	27.7

3.3.2　PGLR モデルを用いた解析について

MSLR パーザは，解析結果となる構文木とその PGLR モデルに基づく生成確率を同時に出力することができる．また，生成確率の高い順に構文木を並べて出力することができる．すなわち，PGLR モデルに基づく生成確率を用いた解析結果の優先順位付けを行うことができる．

MSLR パーザは，まず文法が受理する全ての解析結果を求め，それらをまとめた圧縮統語森を生成する．次に，この圧縮統語森を展開して個々の構文木を出力する際に，PGLR モデルに基づく構文木の生成確率を考慮し，生成確率の上位の構文木から優先して出力する．解析の途中で生成確率の低い部分木を除去するなどの枝刈りを行っていないため，生成確率の上位 N 位の構文木が必ず得られることが保証される代わりに，長文など構文的曖昧性が非常に多い文を解析する際にメモリ不足によって解析に失敗する可能性も高い．したがって，我々は解析途中で生成確率の低い部分木を除去して探索空間を絞り込む機構も必要であると考えている．Sornlertlamvanich は PGLR モデルを利用した効率の良い枝刈りのアルゴリズムを提案しているが (Sornlertlamvanich 1998)，現在公開している MSLR パーザには実装されていない．

3.4　解析例

本項では，MSLR パーザを用いた簡単な日本語文解析実験について報告する．実験用コーパスとして，ATR が作成した日本語対話コーパス (Morimoto, Uratani, Takezawa, Furuse, Sobashima, Iida, Nakamura, and Sagisaka 1994) を使用した．実験に用いた文法は，対話文解析用の文脈自由文法で，非終端記号数 172，終端記号数 441，規則数は 860 である (田中, 竹澤, 衛藤 1997)．今回の実験では，日本語対話コーパス約 20,000 文のうち，上記の文法による構文木が付与された例文 10,020 文を使用した．辞書及び接続表は，これら 10,020 文から自動的に作成した．

評価用テキストとして，単語数 4〜14，15 以上の文をランダムに 1000 文ずつ取り出し，それぞれ Set A, Set B とした．これらの評価用例文について，分かち書きされていない文字列を入力とし，MSLR パーザを用いて形態素・構文解析を行った．また，評価用テキスト以外の例文約 9000 文から PGLR モデルを学習し，その PGLR モデルに基づく構文木の生成確率によって解析結果の順位付けを行った．使用した計算機は，2.3 項の実験と同じ Sun Ultra Enterprise 250 Server である．実験結果を表 2, 3 に示す．また，解析結果の具体例を付録 A に示す．

表 3　解析実験の結果 (文正解率)

n	【形態素解析の文正解率】 Set A	Set B	【構文解析の文正解率】 Set A	Set B
1	88.3%	63.7%	80.1%	36.3%
2	94.4%	75.1%	90.6%	50.4%
3	96.8%	80.6%	95.0%	58.8%
4	97.6%	83.6%	96.4%	65.0%
5	98.8%	87.2%	97.6%	69.6%

表 2 において,「平均解析木数」は 1 文あたりに生成される構文木の平均であり,「平均解析時間」は 1 文の解析に要した時間 (単位はミリ秒) の平均を表わしている．Set A のような短い文の場合は 7 ミリ秒程度，Set B のような長めの文の場合でも 27 ミリ秒程度で解析を行うことができる．また，表 3 の【形態素解析の文正解率】は，PGLR モデルに基づく構文木の生成確率の上位 n 位の解析結果の中に，単語分割と品詞付けの結果がコーパスに付加されたものと一致する構文木が含まれる文の割合を表わしている．同様に【構文解析の文正解率】は，上位 n 位の解析結果の中にコーパスに付加されたものと一致する構文木が含まれる文の割合を示している．この表から，例えば生成確率の 1 位の構文木について，Set A では約 80%，Set B では約 36%の文に対して正しい形態素・構文解析結果が得られたことがわかる．今回の実験で使用したコーパスがドメインの限られたコーパスであり，また辞書と接続表を評価用テキストと訓練用テキストの両方を用いて作成したこともあり，比較的良い結果が得られている．

4　おわりに

本論文では，我々が現在公開している自然言語解析用ツール「MSLR パーザ・ツールキット」の機能と特徴について述べた．最後に，本ツールキットの今後の開発方針について述べる．

まず，複数の接続制約を同時に組み込む LR 表作成器，さらにそれを用いて解析を行うパーザの実装を進めている．現在のツールでは，LR 表に組み込める接続制約の数は 1 種類のみである．しかしながら，例えば音声認識と同時に構文解析を行う場合，品詞間の接続制約だけでなく，音素間の接続制約も同時に利用した方が効率の良い解析ができると考えられる (今井 1999)．この場合，音素と品詞の 2 つの接続制約を LR 表に組み込む必要がある．また，これに合わせて，MSLR パーザの解析アルゴリズムも変更する必要がある．現在，複数の制約を取り扱う LR 表作成器および MSLR パーザのプロトタイプは完成しているが，効率の面でまだ問題があり，改良を進めている．

次に，よりロバストな解析ができるようにパーザを拡張することが挙げられる．特に，辞書にない単語 (未知語) が入力文中に現われたときには，原則的には解析に失敗する．現在の

MSLR パーザは，カタカナが続いた文字列を未知語として登録するなど，非常に簡単な未知語処理機能が付加されているが，まだ改良の余地も多い．また，解析に失敗した場合でも，部分的な解析結果を表示する機能なども追加していきたいと考えている．

最後に，本ツールキットに付属の日本語解析用の文法，辞書，接続表を改良することが今後の課題として挙げられる．これらを用いて新聞記事の解析を行った場合，解析に成功して何らかの結果を返すことのできる文の割合は約 85%である．解析に失敗する原因としては，前述の未知語処理の不完全さや文法規則の不備によるものが多い．より多様な文を解析できるようにするためには，特に文法を改良していかなければならない．また，本ツールキットに付属の文法を用いて解析を行った場合，PGLR モデルを学習するための構文木付きコーパスが存在しないために，PGLR モデルに基づく生成確率によって解析結果に優先順位を付けることはできない[5]．現在，構文木付きコーパスを必要としない PGLR モデルの学習方法について研究をすすめている．

謝辞

MSLR パーザ・ツールキットは多くの方の協力を得て開発されました．李輝氏，日本アイ・ビー・エム株式会社の綾部寿樹氏には初期の LR 表作成器を実装していただきました．九州工業大学の乾健太郎助教授には，PGLR モデルの理論及び実装について議論していただきました．Sussex 大学の John Carroll 氏，National Electronics and Computer Technology Center の Sornlertlamvanich Virach 氏には，MSLR パーザの実装に関する貴重な助言をいただきました．以上の皆様を始め，本ツールキットの開発に御協力いただきました全ての人々に感謝いたします．

MSLR パーザの辞書引きモジュールは，奈良先端科学技術大学院大学・松本研究室で開発された高速文字列検索システム SUFARY をベースに作成しています．SUFARY の転用を許可下さいました松本研究室の皆様に深く感謝いたします．

本ツールキットに付属の日本語解析用の辞書は，日本電子化辞書研究所が作成した EDR 日本語単語辞書 (日本電子化辞書研究所 1995) をもとに構築されています．本辞書の公開を許可下さいました日本電子化辞書研究所の皆様に深く感謝いたします．

参考文献

Aho, A. V., Sethi, R., and Ullman, J. D. (1985). *Compilers — principles, techniques, and tools*. Addison Wesley.

今井宏樹 (1999). 音声認識のための PGLR パーザに関する研究. Ph.D. thesis, Department of Computer Science, Tokyo Institute of Technology. `ftp://ftp.cs.titech.ac.jp/pub/TR/99/TR99-0016.ps.gz`.

5 公開されているツールでは，付属の文法を用いて解析を行った場合でも，単語数最小法，文節数最小法のヒューリスティクスに基づく解析結果の優先順位付けを行うことができる．

Inui, K., Sornlertlamvanich, V., Tanaka, H., and Tokunaga, T. (1998). "Probabilistic GLR Parsing: A new Formalization and Its Impact on Parsing Performance." 自然言語処理, **5** (3), 33–52.

Li, H. (1996). *Integrating Connection Constraints into a GLR Parser and its Applications in a Continuous Speech Recognition System*. Ph.D. thesis, Department of Computer Science, Tokyo Institute of Technology. `ftp://ftp.cs.titech.ac.jp/pub/TR/96/TR96-0003.ps.gz`.

Morimoto, T., Uratani, N., Takezawa, T., Furuse, O., Sobashima, O., Iida, H., Nakamura, A., and Sagisaka, Y. (1994). "A Speech and Language Database for Speech Translation Research." In *Proceedings of the International Conferrence on Spoeken Language Processing*, pp. 1791–1794.

日本電子化辞書研究所 (1995). "EDR 電子化辞書仕様説明書第 2 版." テクニカル・レポート TR–045.

白井清昭, 乾健太郎, 徳永健伸, 田中穂積 (1998). "統計的構文解析における構文的統計情報と語彙的統計情報の統合について." 自然言語処理, **5** (3), 85–106.

Sornlertlamvanich, V. (1998). *Probabilistic Language Modeling for Generalized LR Parsing*. Ph.D. thesis, Department of Computer Science, Tokyo Institute of Technology. `ftp://ftp.cs.titech.ac.jp/pub/TR/98/TR98-0005.ps.gz`.

Sornlertlamvanich, V., Inui, K., Tanaka, H., Tokunaga, T., and Toshiyuki, T. (1999). "Empirical Support for New Probabilistic Generalized LR Parsing." 自然言語処理, **6** (3), 3–22.

田中穂積, 竹澤寿幸, 衛藤純司 (1997). "MSLR 法を考慮した音声認識用日本語文法–LR 表工学 (3)–." 情報処理学会音声言語情報処理研究会, 97 巻, pp. 145–150.

Tanaka, H., Tokunaga, T., and Aizawa, M. (1995). "Integration of Morphological and Syntactic Analysis based on LR Parsing Algorithm." 自然言語処理, **2** (2), 59–73.

略歴

白井 清昭: 1993 年東京工業大学工学部情報工学科卒業．1995 年同大学院理工学研究科修士課程修了．1998 年同大学院情報理工学研究科博士課程修了．同年同大学院情報理工学研究科計算工学専攻助手，現在に至る．博士 (工学)．統計的自然言語解析に関する研究に従事．情報処理学会会員．

植木 正裕: 1995 年東京工業大学工学部情報工学科卒業．1997 年同大学院情報理工学研究科修士課程修了．2000 年同大学院情報理工学研究科博士課程満期退学．同年 4 月同大学院情報理工学研究科計算工学専攻技術補佐員．同年 7 月国立国語研究所日本語教育センター研究員，現在に至る．自然言語解析に

関する研究に従事．情報処理学会会員．

橋本 泰一： 1997 年東京工業大学工学部情報工学科卒業．1999 年同大学院情報理工学研究科計算工学専攻修士課程修了．同年同大学院情報理工学研究科計算工学専攻博士課程進学，在学中．統計的自然言語解析に関する研究に従事．

徳永 健伸： 1983 年東京工業大学工学部情報工学科卒業．1985 年同大学院理工学研究科修士課程修了．同年 (株) 三菱総合研究所入社．1986 年東京工業大学大学院博士課程入学．現在, 同大学大学院情報理工学研究科計算工学専攻助教授．博士 (工学)．自然言語処理, 計算言語学に関する研究に従事．情報処理学会, 認知科学会, 人工知能学会, 計量国語学会, Association for Computational Linguistics, 各会員．

田中 穂積： 1964 年東京工業大学工学部情報工学科卒業．1966 年同大学院理工学研究科修士課程修了．同年電気試験所 (現電子技術総合研究所) 入所．1980 年東京工業大学助教授．1983 年東京工業大学教授．現在, 同大学大学院情報理工学研究科計算工学専攻教授．博士 (工学)．人工知能，自然言語処理に関する研究に従事．情報処理学会, 電子情報通信学会，認知科学会, 人工知能学会, 計量国語学会, Association for Computational Linguistics, 各会員．

(2000 年 1 月 6 日 受付)
(2000 年 3 月 14 日 採録)

付録

A　MSLR パーザによる解析例

3.4 項の実験で得られた解析結果の例を挙げる．まず，以下の例文 (1),(2),(3) を解析し，PGLR モデルによる生成確率の最も大きい解析結果のみを表示させたときの MSLR パーザの出力を示す．

(1)　七日までのご予約ですので八日と九日の分でございますか
(2)　十日と十一日のご予約を十一日と十二日に変更なさりたいわけですね
(3)　御社の場合には割引価格が適用されますので朝食も含めて割と良いお部屋を百九十三ドルでご提供できます

• MSLR パーザの出力

```
% mslr -g atr.gra -l atr.prtb.set2 -d atr-all.dic.ary -i -p -P -N 1 < sentence
reading the grammar file 'atr.gra' Done
reading LR table file 'atr.prtb.set2' Done
### 1 ###
$TAC23034-0030-3
七日までのご予約ですので八日と九日の分でございますか

accept
```

[<sent>,[<cl>,[<adv-cl>,[<verb>,[<verb/ga>,[<np>,[<n-sahen>,[<mod-n>,[<pp>,[<np>,[<n-date&time>,[<n-day>,[meisi-hi, 七日]]]],[<p-kaku-optn>,[p-kaku-made, まで]]],[<p-rentai>,[p-rentai-no, の]]],[<n-sahen>,[<n-sahen/ga-o>,[<prefix>,[prefix-go, ご]],[sahen-meisi/ga-o, 予約]]]]],[<aux>,[<auxstem>,[auxstem-desu, で]],[<infl>,[infl-spe-su, す]]]]],[<p-conj-advcl>,[p-conj-syusi, ので]]],[<cl>,[<vaux>,[<vaux>,[<verb>,[<verb/ga>,[<np>,[<n-hutu>,[<mod-n>,[<np>,[<n-date&time>,[<mod-n>,[<np>,[<n-date&time>,[<n-day>,[meisi-hi, 八日]]]],[<p-para>,[p-para-to, と]]],[<n-date&time>,[<n-day>,[meisi-hi, 九日]]]]],[<p-rentai>,[p-rentai-no, の]]],[<n-hutu>,[hutu-meisi-post, 分]]]],[<aux>,[aux-de, で]]]],[<aux>,[<auxstem>,[auxstem-copula-masu, ございま]],[<infl>,[infl-spe-su, す]]]],[<aux>,[aux-sfp-ka, か]]]]]] 5.716416e-23

total 1314
CPU time 0.2 sec

2
$TAS13004-0100-1
十日と十一日のご予約を十一日と十二日に変更なさりたいわけですね

accept

[<sent>,[<cl>,[<vaux>,[<verb>,[<verb/ga>,[<np>,[<vaux>,[<vaux>,[<verb>,[<verb/ga>,[<pp-o>,[<np>,[<n-sahen>,[<mod-n>,[<np>,[<n-date&time>,[<mod-n>,[<np>,[<n-date&time>,[<n-day>,[meisi-hi, 十日]]]],[<p-para>,[p-para-to, と]]],[<n-date&time>,[<n-day>,[meisi-hi, 十一日]]]]],[<p-rentai>,[p-rentai-no, の]]],[<n-sahen>,[<n-sahen/ga-o>,[<prefix>,[prefix-go, ご]],[sahen-meisi/ga-o, 予約]]]]],[p-kaku-o, を]],[<verb/ga-o>,[<mod-v>,[<pp>,[<np>,[<n-date&time>,[<mod-n>,[<np>,[<n-date&time>,[<n-day>,[meisi-hi, 十一日]]]],[<p-para>,[p-para-to, と]]],[<n-date&time>,[<n-day>,[meisi-hi, 十二日]]]]],[<p-kaku-optn>,[p-kaku-ni, に]]]],[<n-sahen/ga-o>,[sahen-meisi/ga-o, 変更]]]]],[<aux>,[<auxstem>,[auxstem-sahen-5-r, なさ]],[<infl>,[infl-5-ri, り]]]],[<aux>,[<auxstem>,[auxstem-wish, た]],[<infl>,[infl-adj-i, い]]]],[<np>,[<n-hutu>,[n-keisiki, わけ]]]],[<aux>,[<auxstem>,[auxstem-desu, で]],[<infl>,[infl-spe-su, す]]]],[<aux>,[aux-sfp-ne, ね]]]]] 6.846102e-32

total 2583
CPU time 0.3 sec

3
$TAS12006-0080-1
御社の場合には割引価格が適用されますので朝食も含めて朝と良いお部屋を百九十三ドルでご提供できます

accept

[<sent>,[<cl>,[<adv-cl>,[<vaux>,[<vaux>,[<vaux>,[<verb>,[<verb/o>,[<mod-v>,[<pp>,[<pp>,[<np>,[<n-hutu>,[<mod-n>,[<np>,[<n-hutu>,[hutu-meisi, 御社]]],[<p-rentai>,[p-rentai-no, の]]],[<n-hutu>,[hutu-meisi, 場合]]]],[<p-kaku-optn>,[p-kaku-ni, に]]],[<p-kakari>,[p-kakari-wa, は]]]],[<verb/o>,[<pp-ga>,[<np>,[<n-hutu>,[<n-sahen>,[<n-sahen/ga-o>,[sahen-meisi/ga-o, 割引]]],[<n-hutu>,[hutu-meisi, 価格]]]],[p-kaku-ga, が]],[<n-sahen/ga-o>,[sahen-meisi/ga-o, 適用]]]]],[<aux>,[aux-suru-sa, さ]]],[<aux>,[<auxstem-deac>,[auxstem-deac-reru, れ]]]],[<aux>,[<auxstem>,[auxstem-masu, ま]],[<infl>,[infl-spe-su, す]]]],[<p-conj-advcl>,[p-conj-syusi, ので]]],[<cl>,[<adv-cl>,[<verb>,[<verb/ga-ni-o>,[<mod-v>,[<pp>,[<np>,[<n-hutu>,[hutu-meisi, 朝食]]],[<p-kakari>,[p-kakari-mo, も]]]],[<verb/ga-ni-o>,[<vstem1/ga-ni-o>,[vstem-1/ga-ni-o, 含め]]]]],[<p-conj-advcl>,[p-conj-renyo-te, て]]],[<cl>,[<vaux>,[<vaux>,[<verb>,[<verb/ga>,[<pp-o>,[<np>,[<n-hutu>,[<mod-n>,[<verb>,[<verb/ga>,[<mod-v>,[<advp>,[<adv>,[hukusi, 割と]]]],[<verb/ga>,[adjstem/ga, 良],[<infl>,[infl-adj-i, い]]]]]],[<n-hutu>,[<prefix>,[prefix-o, お]],[<n-hutu>,[hutu-meisi, 部屋]]]]],[p-kaku-o, を]],[<verb/ga-o>,[<mod-v>,[<pp>,[<np>,[<n-quant>,[<n-num>,[<n-num-hyaku>,[<n-num-keta-hyaku>,[<num-suf-hyaku>,[num-hyaku, 百]]],[<n-num-zyuu>,[<n-num-keta-zyuu>,[<n-num-ichi>,[num-kyuu, 九]],[<num-suf-zyuu>,[num-zyuu, 十]]],[<n-num-ichi>,[num-san, 三]]]]],[<suffix-unit>,[suffix-doru, ドル]]]],[<p-kaku-optn>,[p-kaku-de, で]]]],[<n-sahen/ga-o>,[<prefix>,[prefix-go, ご]],[sahen-meisi/ga-o, 提供]]]]],[<aux>,[<auxstem>,[auxstem-sahen-1, でき]]]],[<aux>,[<auxstem>,[auxstem-masu, ま]],[<infl>,[infl-spe-su, す]]]]]]]] 6.264841e-45

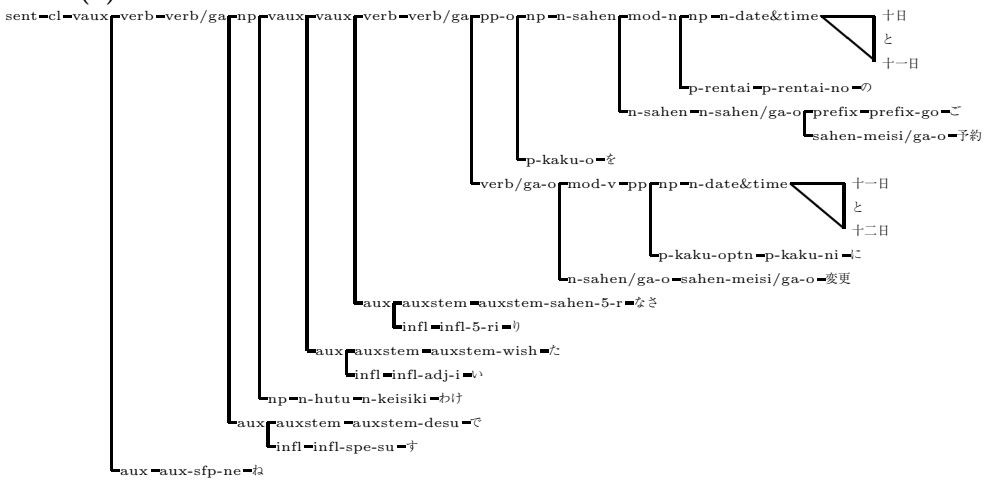
total 19284
CPU time 0.13 sec

解析結果は括弧付けで表現された構文木として出力される．構文木の右にある数値はその構文木の PGLR モデルによる生成確率である．「total」は得られた解析結果の総数を，「CPU time」は解析に要した時間を表わす．

以下，得られた解析結果を構文木の形で示す．但し，紙面の都合により，構造の一部を簡略している．

- 例文 (1) の解析結果

- 例文 (2) の解析結果



- 例文 (3) の解析結果