自然言語処理 74－9
（1989. 9. 21）

横型トップダウン文解析システムの実現と評価

林達也* 宮俊司** 坂巻利哉** 吉田健一**
* 富士通研究所
** 富士通ソーシアルサイエンスラボラトリ

本稿では，横型トップダウン方式のＹＡＰＸＲ文解析システムの実現と評価について述べる．ＹＡＰＸＲは自然言語モデリングシステムＳＯＬＯＭＯＮの一つのコンポーネントとして位置付けられる．ＹＡＰＸＲが受け入れる文法機能は，従来のＹＡＰＸで許されていた拡張ＣＦＧを更に拡大し，文脈依存規則や文脈依存制約等を導入できる．これを Restricted Context Sensitive Grammar（ＲＣＳＧ）と呼んでいる．文脈依存規則はボトムアップ方式をとる他のシステムに於いても適用することができるものである．本方式を中規模英文法に対して適用した結果，記述性と時間的性能の両面で，他のシステムを上回る実用的なＰＲＯＬＯＧベースの文解析システムを構築できることを確認した．

Implementation and Evaluation of the Breadth_first Top_down Sentence Analysis System

Tatsuya Hayashi * Syunji Miya** Toshiya Sakamaki** Kenichi Yoshida**
* Fujitsu Laboratories LTD.
1015 Kamikodanaka Nakaharaku Kawasaki, 211, JAPAN
** Fujitsu Social Science Laboratory LTD.

The implementation and evaluation of the breadth-first top-down YAPXR sentence analysis system is described. YAPXR is positioned as one component of the natural language modeling system SOLOMON. The grammatical functions which YAPXR accepts further expand the extended CFG allowed by the current YAPX, and make it possible to introduce context sensitive rules, context sensitive constraints, etc. We call this Restricted Context Sensitive Grammar (RCSG). As a result of applying our method to a middle scale English grammar, we have confirmed that it is possible to construct a practical parsing system with PROLOG base better than other systems in terms of both descriptive capability and time efficiency.

1. はじめに

筆者等は，先に横型トップダウン方式の構文解析法ＹＡＰ及びその拡張系ＹＡＰＸについて述べた．[1)2)]

又，核位置の概念を中心とした効率的実現法についても考察した．[3)]

本稿では，ＹＡＰＸの文法機能を更に拡大し，文脈依存規則や文脈依存制約等を導入したＹＡＰＸＲ（Revised version of YAPX）システムを実現したので，その具体的な機能と性能について述べる．

以下では，2.でＹＡＰＸＲシステムで許される文法機能について述べた後，3.で実現方法を，4.で適用評価について述べる．

2. 文法機能

本システムで使用される文法をＲＣＳＧ（Restrictive Context Sensitive Grammar）と呼ぶ．ＲＣＳＧは，引数，条件式（本システムでは制約と呼ぶ），スラッシュカテゴリ等を含む文法形式を基本とし，新たに，トレース対象外指定，文脈依存規則，文脈依存制約を導入している．以下に，文法機能について説明する．

2.1 基本形式

文法の基本形式は，

左辺 --> 右辺

である．

左辺：非終端記号［(内部引数，・・) ］
［(/外部引数，・・/)］

右辺：ε又は右辺要素・・・・・・

右辺要素：構文記号［(内部引数，・・) ］
［(/外部引数，・・/)］
［スラッシュカテゴリ］
［ {基本制約} ・・・］

スラッカテゴリ：/(δ) 又は//B /(δ)

B は非終端記号，δはギャップ指定子，終端・非終端記号からなる記号列

基本制約：引数に関する論理式

本システムでは，規則の右辺第１項にも，スラッシュカテゴリを指定することができる．更に，右辺第１項がトレースになることも差支えない．

2.2 トレース対象外指定

これは文献８）で既に採り入れられているが，ＹＡＰＸＲに於いても導入したものである．即ち，右辺の要素がトレースにならない場合は，規則の

左辺 --> 右辺 の矢印 '-->' を '==>' と指定することにより，右辺の要素がトレースの対象外であることを示す．

2.3 文脈依存規則

本システムの文脈依存規則は，一般の文脈依存文法に比べて極めて制限が強いが，これの導入によって，不要な解析を行わずにすみ，又，非終端構文要素や，文法規則の不必要な増大を防ぐことができる．以下に，文脈依存規則の形式について示す．但し，ここでは引数及び制約は省略する．

{ [～] [( ] α，・・・[ )] }
[{ [～] [( ] α'，・・・[ )] }]
B --> β
B ∈ V_N　β ∈ V*
≡ p { A_p --> τ α A α' δ，
A ≐> Bη，τ，α，α'，δ，η ∈ V* }

又は

{ [～] < n，・・> }　　B --> β
B ∈ V_N　β ∈ V*
≡ p { A_p --> μ <n> Aδ，
A ≐> B η，μ，η，δ ∈ V* }

[～] は否定を表す．

n は核位置ラベルである．なお，α，α' は，一つの規則の右辺に含まれていなければならない．

文脈依存規則の例を以下に示す．

a --> b c d ・・・・・・・・・ (1)
d --> [e] ・・・・・・・・・ (2)
{ ～c }
d --> [f] g ・・・・・・・・・ (3)

上記の例の場合，(1)のd に対して，(2)の規則は適用するが，(3)の規則は適用しない．このようにして，非終端要素や規則数の増大を防止することができる．

2.4 基本制約

構文的条件を引数と制約を用いて指定することにより，解析パスの絞り込みを行う．以下に基本制約の例を示す．

```
sdec01 ==> subj  aux  advx
           vp(FLEX)/( ～obj) {FLEX=en }
           ppx              ・・・・・・・・  (4)
sdec01 ==> subj  advx  vp {FLEX≠en}
                          ・・・・・・・・  (5)
vp(FLEX)--> [vt] (FLEX)  obj ・・・・・  (6)
```

この例の場合，(4) (受動態の規則) の動詞句(vp)は活用形 (FLEX) が過去分詞 (en) でなければならないことを示している．又，(5)の場合は，逆にFLEXが，en であってはならないことを示している．vtのFLEX属性は，(6)によって，vp のFLEX情報として引き継がれる．

2.5 文脈依存制約

基本制約が，右辺のシンボルの引数に対する無条件の制約であることに対して，文脈依存制約は，ある核位置の下での制約である．以下に文脈依存制約の形式を示す．

{ 文脈制約 , ・・・}

文脈制約： < 核位置ラベル, ・・ >: 基本制約

これは，解析パスが当該核位置にある場合，対応する制約条件をみたす時に限り，規則が適用されることを示している．

以下に，文脈依存制約の例を示す．

```
sdec01 ==> subj  aux  advx
           <n1> vp/(～obj)  ppx   ・・・・(7)
sdec01 ==> subj  advx  <n2> vp    ・・・・(8)
vp     -->   [vt] (FLEX)
          {<n1>:FLEX = en,  <n2>:FLEX≠en  }
             obj                  ・・・・(9)
```

ここで，n_1，n_2は核位置ラベルである．

この例の場合，核位置ラベルn_1の下での [vt] のFLEX はenでなければならず，核位置ラベルn_2の下での [vt] のFLEXは，en であってはいけないことを示している．(4)～(6)と(7)～(9)は，機能的には同等の制約条件を表しているが，(7)～(9)のように記述することにより，[vt] を読み込んだ時点で，直ちに，FLEX のチェックを行う．従って，当該核位置の下で解析パスの早期絞り込みが可能となる．つまり，核位置は，一種の文脈の役割を果たしていると言える．文脈依存制約は，A (x) ≐> [T] (x) α のように上位の構文要素Aの引数が最左導出される終端記号の引数を継承したものである場合有効である．

2.6 外部引数の適用

(10)のように他の構文要素の引数を制約に含む場合，外部引数の概念を用いて，他の構文要素の引数を参照する．

```
sdec01 ==> subj advx auxd(CATX1) advx
      <n3> vp(FLEX)
          { FLEX ≠ en ¦CATX1 = have  }
                              ・・・・・・・(10)
```

を，

```
sdec01 ==> subj advx auxd(CATX1) advx
          <n3> vp(/CATX1/)    ・・・・・・・(11)
vp(/CATX1/) ==> [vt] (FLEX)
                {<n3>:
                   FLEX≠en ¦CATX1=have  }
                obj              ・・・・・・(12)
```

とする．ここで，(11)，(12)の(/CATX1/) はauxdのCATX1属性が vp の規則中で使用されることを示している．

一般には，

```
A --> α B  β C  γ (α, β, γ ∈ V* )
                                 ・・・・・(13)
C ≐> σ D  η (σ, η ∈ V* )      ・・・・(14)
```

なるC,D を定義する規則中で，構文要素B の引数を参照する場合には，外部引数を用いる．その場合，外部引数は，(13)で宣言され，C,D を定義する規則中で参照されると言う．尚，内部，外部引数とも意味処理においても活用される．

3. 実現方法

ＹＡＰＸＲシステムは，最終的には図１に示すように，我々が目指している自然言語モデリングシステム，ＳＯＬＯＭＯＮ (SOphisticated natural Language Oriented MOdeliNg system)のコンポーネントの１つである．そして，全体の中では文解析サブシステムと

して位置づけられる.

ＹＡＰＸＲ実現の基本的メカニズムについては既に述べた通り[2)3)]であり，ここでは，それと異なる部分について述べる.

構文解析時には，４種類のスタックを用いて解析を行い，それぞれがホーン節の引数と対応する．以下に構文要素に対応する述語で用いられる引数について示す.

n　・・・・　解析スタック（入力）の先頭要素
Ai　・・・・　解析スタック（入力）の残り
Ao,Ao1・・・　解析スタック（出力，差分リスト）
Oi　・・・・　省略スタック（入力）
Oo,Oo1・・・　省略スタック（出力，差分リスト）
Li　・・・・　引数スタック（入力）
Lo,Lo1・・・　引数スタック（出力，差分リスト）
Ti　・・・・　解析木スタック（入力）
To,To1・・・　解析木スタック（出力，差分リスト）

但し，以下では煩雑を避けるため，必要な引数のみを差分リスト，カット記号を省略して示す.

3.1　ε規則処理の最適化

述語呼び出しの overhead を減らすためにε規則適用の際は，当該述語を呼び出す代わりに，対応する解析パスを直接，スタックに出力する.

例えば，

```
sdec01 --> subj advx 10bep 11advx 12pred
```

においてadvxがε規則を持つ場合 bepのホーン節を，

```
bep(10, Ai, [ [11 ¦ Ai] ¦ Ao] ,
     Ti,  [Ti ¦ To] ):-
   advx(11, Ai, Ao, [ advx(ε) ¦ Ti] ,To).
```

のように，advxの述語を呼び出す代わりに，

```
bep(10, Ai, [ [11 ¦ Ai] ,  [12 ¦ Ai] ¦ Ao] ,
 Ti,  [Ti,  [ advx(ε) ¦ Ti] ¦ To] ).
```

とする.

同様に，ε規則を持つ構文要素が規則中に連続して出現した場合には，対応する複数個の解析パスを直接出力する.

3.2　トレース処理の最適化

トレース処理に於いてもε規則の処理と同様にスラッシュの適用に際して，対応するパスを直接出力する.

例えば，

```
vp --> vt 21obj1 22obj2
```

において,obj1 がスラッシュ要素の場合,vt のホーン節を，

```
vt(n, Ai, [ [21,n ¦ Ai] ¦ Ao] ,
                  [Ocar ¦ Oi] ,Oo, Ti, To):-
   ((Ocar == (g,obj1),
     obj1(21,  [n ¦ Ai] , Ao, Oi, Oo,
          [obj1(trace) ¦ Ti] , To));
   ( Ao= [] , Oo= [] , To= [] )).
```

のように，スラッシュに対応するシンボルの述語を呼び出す代わりに，

```
vt(n, Ai, [ [21,n ¦ Ai] ¦ Ao] ,
                  [Ocar ¦ Oi] ,Oo, Ti, To):-
   ((Ocar == (g,obj1),
     Ao =  [22,n ¦ Ai] ,Oo = Oi,
     To =  [obj1(trace) ¦ Ti] );
    ( Ao= [] , Oo= [] , To= [] )).
```

とする.

3.3　文脈依存規則

当該文脈（核位置）に於いて，該当する規則による導出を指示に基づき，あらかじめ中止あるいは実施すればよい.

3.4　引数と制約の処理

引数（内部，外部）及び制約の処理のために，解析スタックを兼用することは可能であるが，煩雑を避けるため，本システムでは引数スタックを用いている.

内部引数は，その述語が呼ばれた時点では，引数スタックの先頭に置かれているように処理される．次にその内部引数が規則の右側で参照されていれば，引数スタックに残し，参照されていなければ，引数スタックから削除する．なお引数の参照の有無を判断する場合，規則左辺の要素は，右辺の最後に位置するものとみなす.

外部引数も，規則の右側で宣言されている場合に，引数スタックの先頭に格納され，外部引数を参照する要素の内部引数の次に引数スタックに存在するように

する．処理系は制約の直前の要素に対応するホーン節の中で，引数スタック中のこれらの引数を用いて制約の評価を行うようなコードを生成する．

以下に引数と制約の処理の例を示す．

規則

```
vp(/CATX1/) ==> [vt] (FLEX)
                {<n3>:(FLEX≠en ¦ CATX1=have) }
                    obj
```

の vt に対応する述語は，

```
vt(・・・, [FLEX,CATX1 ¦ Li] ,
                [ [ CATX1 ¦ Li] ¦ Lo] ):-
              FLEX \== en, CATX1 == have .
```

となる．

FLEXはその後不要なので，引数スタックから消去する．CATX1 についても不要であるが，便宜上ここでは消去しない．(1)による還元で sdec01が呼ばれる時点で消去される．

3.5 辞書の処理

ＹＡＰＸＲ文解析システムは，文法コンパイラ，辞書コンパイラ，入力インタフェースから構成される．辞書コンパイラは単語辞書を辞書ホーン節に変換する．

各単語には構文処理に必要な最小限の属性を与えている．属性には，単語見出し，品詞（終端記号），品詞区分（CATX），活用形（FLEX）があり，１品詞，１エントリであり，多品詞語の場合は，マルチエントリとなる．

辞書コンパイラにより，生成されるホーン節の形式を以下に示す．

```
word00(Ai,Ao,0i,0o,Li,Lo,Ti,To,word):- !,
   posl__0(Ai,Ao,Aol,0i,0o,0ol,Li,Lo,Lol,
       Ti,To,Tol, word, [catxl,flexl ] ),
                  .
                  .
   posn__0(Ai,Aon-1,Aon,0i,0on-1,0on,
                Li,Lon-1,Lon,Ti,Ton-1,Ton,
                     word, [catxn,flexn ] ),
   Aon = [] ,0on = [] ,Lon = [] , Ton = [] .
```

```
posl__0( [ [ N ¦ Acar] ¦ Acdr] , Ao , Ao2
         [0car ¦ 0cdr] , 0o , 0o2 ,
         [Lcar ¦ Lcdr] , Lo , Lo2 ,
         [Tcar ¦ Tcdr] , To , To2,
         Word, Attr):- !,
      posl__01(N,Acar,Ao,Aol,0car,0o,0ol,
                  [Attr ¦ Li] ,Lo,Lol,
                  [posl(Word) ¦ Ti] ,To,Tol),
      posl__0(Acdr,Aol,Ao2,0cdr,0ol,0o2,Lcdr,
              Lol,Lo2,Tcdr,Tol,To2,Word,Attr).
posl__0( [] ,Ao,Ao, __,0o,0o, __,Lo, Lo,
                          __,To,To, __, __ ):- !
```

word: 単語

posl,posn:品詞

catxl,flexl,catxn,flexn:属性値

posl__01: 文法ホーン節を呼び出す述語

3.6 入力インタフェース

入力インタフェースは，入力された英文を，パーサの述語を呼び出すようなホーン節に変換する．

以下に入力インタフェースの例を示す．

入力文 $a_1 \cdots\cdots\cdots a_n$

```
yapx:- yopen(A1,A2,01,02,L1,L2,T1,T2),
       a1(A2,A3,02,03,L2,L3,T2,T3,a1),
                  .
                  .
       an (An, An+1, On, On+1, Ln, Ln+1,
                                Tn, Tn+1),
     yclose(An+1, On+1, Ln+1, Tn+1).
```

yopen:文法に固有の初期化述語

yclose: 解析木を生成するシステム固有の述語

4. 適用評価

4.1 文法規模

ここでは，本システムを中規模英文法 [8] に対して適用した結果について述べる．

表１に文法の規則数，種類を示す．

文法規則の総数は，ＳＡＸ，ＬａｎｇＬＡＢの場合の約５６０，４３０に相当する．[8]

表1　文法の規則数，種類

項　　目	規則数
規則の総数	299
終端記号	28
非終端記号	75
ε規則	18
左回帰則	23
スラッシュ則	21
スラッシュシンボル	7
右辺第1項のスラッシュ則	2
トレース対象外規則	241
文脈依存規則	24
文脈依存制約規則	35
基本制約規則	17
外部引数宣言	4
外部引数参照	10

表1に示したようにＹＡＰＸＲでは，文法全体が，記述性よくコンパクトにまとめられている．そのため，文法の保守，改良が容易に行える．特にＹＡＰＸＲでは，右辺の第1項にスラッシュカテゴリが記述でき，またε規則や文脈依存規則が許される点が，顕著な効果をみせている．

4.2　解析時間及び所要スペース

上記英文法に基づき例文を入力して，解析時間及びメモリを集計した結果を表2に示し，評価環境について以下に示す．なお，現在のバージョンはεやトレース処理の最適化を行っていない．また，制約を持つホーン節が本来不要なバックトラックも行うようになっている．

㊀計算機環境・・・SUN3/60 UNIX(Sun OS)
㊁使用言語　・・・Quintus prolog (Release 2.4)
㊂文法規則　・・・299
㊃辞書・・・・・・180 エントリ
㊄例文・・・・・・41文（〈付録〉参照）
㊅評価項目
・解析時間・・コンパイル，インタプリット両モードの全解探索に要する時間
（但しガーベイジコレクション，スタックシフトの時間は除く）
・内部メモリ使用量・・グローバルスタックの使用量

文の長さに関わらず，1単語平均10msecで解析している．また，解釈数が増えても2～3倍程度の解析時間で解析を終了している．

表2　評価結果

番号	語数	解釈数	解析時間 (sec)		内部データ領域使用量 (bytes)	
			a	b	a	b
1	7	2	0.416	0.067	151,800	21,168
2	8	1	0.216	0.083	85,904	11,828
3	6	1	0.183	0.033	62,508	10,912
4	4	2	0.200	0.033	67,616	11,296
5	8	2	0.933	0.133	310,396	43,776
6	8	1	1.066	0.200	411,800	51,452
7	6	1	0.366	0.050	140,220	16,820
8	11	1	0.516	0.100	173,522	24,136
9	7	1	0.134	0.033	59,492	9,128
10	8	4	0.900	0.117	312,248	46,064
11	8	2	0.584	0.083	195,844	26,712
12	5	1	0.200	0.017	67,280	9,584
13	8	2	0.518	0.100	201,880	31,736
14	7	2	0.267	0.034	104,444	16,268
15	11	1	0.300	0.083	107,712	12,892
16	5	1	0.184	0.066	65,824	10,592
17	11	1	0.567	0.100	205,844	26,224
18	10	1	0.317	0.100	147,796	21,156
19	7	1	0.250	0.050	87,852	14,096
20	9	4	1.184	0.200	423,916	60,868
21	4	1	0.200	0.033	65,484	7,964
22	9	2	0.333	0.050	123,408	15,984
23	8	1	0.617	0.117	246,064	31,100
24	8	1	0.650	0.134	252,684	40,724
25	23	14	2.650	0.483	1,010,624	165,260
26	6	2	0.617	0.034	81,196	14,624
27	7	1	0.283	0.083	96,524	15,604
28	9	1	0.250	0.050	110,276	15,900
29	10	1	0.450	0.083	157,344	21,968
30	11	2	0.350	0.067	145,640	23,632
31	18	5	1.100	0.300	358,276	67,372
32	16	5	1.717	0.317	652,844	99,076
33	22	10	19.916	1.217	1,222,456	325,036
34	19	4	1.100	0.200	398,456	64,592
35	20	1	2.417	0.433	881,960	111,768
36	25	21	24.067	1.000	1,287,256	346,936
37	24	2	4.722	0.783	1,684,076	220,952
38	28	3	4.183	0.767	1,576,252	224,376
39	33	2	3.150	0.616	1,239,396	173,888
40	4	1	0.084	0.017	36,368	6,020
41	5	1	0.183	0.017	57,500	8,608

a・・ interpret
b・・ compile

解析時間に関しては他のシステムのデータがある.[4)5)6)] 使用した計算機の性能やカバーする言語の規模が異なるのでそれらを勘案して比較すると，表2の値はいずれをも上回る良好な結果を示している.

一方，スペース効率に関しては，ＹＡＰＸＲの場合左端要素に対応する核位置が複数個存在する為，プログラムサイズが一般に増大する.

5. あとがき

本稿ではＹＡＰＸ[1)2)3)]を基本とし，更に文脈依存規則等を認めたＲＣＳＧに基づく文解析システムを開発し，中規模英文法を対象に性能を測定して良好な結果を得た．現在のシステムは基本システムであり，最適化処理や文法開発支援のための良好なＭＭＩが整備されておらず，これらが今後の課題である．又，英文法も現在例文に関係する部分しかｔｕｎｉｎｇされていないで，文法全体の整備が今後の課題として残されている．更に，ＳＯＬＯＭＯＮシステム実現の為，意味文脈サブシステムと文生成サブシステム等についても現在開発を進めている.

参考文献

1)林達也：論理型言語による構文解析法ＹＡＰについて・情報処理論文誌 vol.29, No.9 (1988)

2)林達也：拡張ＣＦＧとその構文解析法ＹＡＰＸについて・情報処理論文誌 vol.29, No.5 (1988)

3)林達也：ＹＡＰＸの効果的実現法・情報処理論文誌(採録) (1988)

4)T. Okunishi, R. Sugimura, Y. Matsumoto, N. Tamura, T. Kamiwaki, H. Tanaka : Comparison of Logic Programing Based Natural Language Parsing Systems・North-Holand NLU and LP vol.2 (1988)

5)杉村領一：ロジックプログラミングをベースにした自然言語解析システムの比較・情報処理学会自然言語処理研究会 vol.86 (1987)

6)田中穂積，上脇正，奥村学，沼崎浩明：自然言語処理の為のソフトウエアシステムＬａｎｇＬＡＢ・Proc. LPC86 (1986)

7)Aho A. V., Ullman J. D. : The Theory of Parsin g, Translation, and Compiling, Preutice Hell, Englewood Cliffs (1972)

8)東工大田中研究室：中規模英文法 (仮称) (1987)

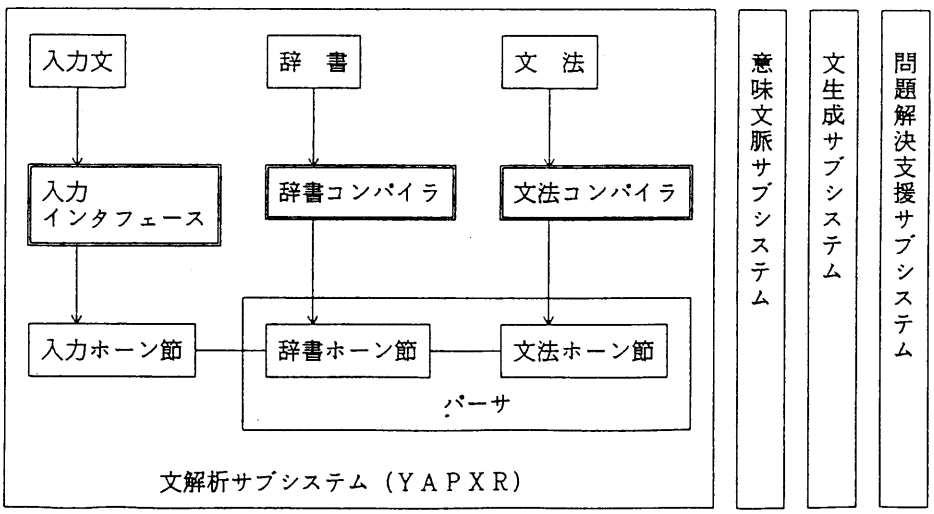

図1　自然言語モデリングシステム　ＳＯＬＯＭＯＮ

< 付録 > 以下に評価に用いた例文を示す.

1. There are three on the table now .
2. There are some men here from the city .
3. Those are the books she read .
4. Were they given her .
5. Could she have been given many of them .
6. I saw many more books than she did .
7. I saw that they were there .
8. The books that were on the table are difficult to read .
9. My uncle gave the girl several books .
10. She was given some books by her uncle .
11. The books were given her by her uncle .
12. He gave up her the books .
13. You could break this vase with that hammar .
14. That hammar could break this vase easily .
15. Did not those people want him to try to do it .
16. Are there any more left .
17. Is this any harder for him to do than that was .
18. This is a film that is developed in the research .
19. The system for interpreting dialogues is developed .
20. She was given more difficult books by her uncle .
21. Do not do that .
22. Be very careful not to do it too quickly .
23. Read these books , if you have time .
24. If you have time , read these books .
25. This paper presents an explanatory overview of a large and complex grammar that is used , in a computer system for interpreting English dialogues .
26. He explains the example with rules .
27. The structural relations are holding among constituents .
28. He explained the example and he illustrates the rule .
29. It is not tied to a particular domain of applications .
30. Diagram analyzed all of the basic kinds of phrases and sentences .
31. This paper presents an explanatory overview of a large and complex grammar that is used in a sentences .
32. Diagram analyzes all of the basic kinds of phrases and sentences and many quite complex ones .
33. The annotions provide important information for other patrs of the system that interpret the expression , in the context of a dialogue .
34. For every expression it analyzed , diagram provides an annotated description of the structural relations holding among its constituents .
35. Procedures can also aassign scores to an analysis , rating some applications of a rule as probable or as unlikely .
36. This paper presents an explanatory overview of a large and complex grammar , diagram that is used , in a computer for interpreting english dialogue .
37. Its procedures allow phrases to inherit attributes from their constituens and to acquire attributes from the larger phrases in which they themselves are constituents .
38. Consequently , when these attributes are used to set context sensitive constraints on the acceptance of an analysis , the contextual constrains can be imposed by conditions on constituents .
39. It is not tied to a particular domain of applications , and it can be extended to analyzed additional constructions , using the formalism in which it is currently written in the rule .
40. I open the window .
41. Diagram is an augmented grammar .