TR-IT-0010

C言語によるCHART構文解析法(I)――プログラム解説書――

Chart Parser: C language version (I) Manual

ローケン・キム キュンホ セリグマン マーク 北川 美宏*

Kyung-ho LOKEN-KIM Mark SELIGMAN Yosihiro KITAGAWA*

1993 .8

概要

CHART解析法は、構文解析法の中でも柔軟性のある解析法である。下降型あるいは上昇型での解析が選択できる上、それぞれに対して、縦型あるいは横型の探索方式が選択できる。しかもそれぞれを混在させることも可能である。今回は、簡単な文脈自由文法上で動くCHART構文解析法をC言語により作成したので報告する。次葉から、第２章では実行モジュールの作成方法を、第３章では実行方法と実行時のオプションを、第４章ではプログラムの内容を、第５章ではテストに用いた文法ファイルを、それぞれ簡単に説明する。

ATR Interpreting Telecommunications Research Laboratories

ATR 音声翻訳通信研究所
*国際電気通信基礎技術研究所

目次

第１章

はじめに

CHART解析法は、構文解析法の中でも柔軟性のある解析法である。下降型あるいは上昇型での解析が選択できる上、それぞれに対して、縦型あるいは横型の探索方式が選択できる。しかもそれぞれを混在させることも可能である。今回は、簡単な文脈自由文法上で動くCHART構文解析法をc言語により作成したので報告する。

次葉から、第２章では実行モジュールの作成方法を、第３章では実行方法と実行時のオプションを、第４章ではプログラムの内容を、第５章ではテストに用いた文法ファイルを、それぞれ簡単に説明する。CHART解析法の詳細については文献 1、2を参考していただきたい。

第２章

実行モジュールの作成方法

実行モジュールの作成には、以下のファイルが必要である。

```
chart.c          chart_util.c          read_datafile.c
test_print.c     chart.h               chart_util.h          Makefile
```

これらのファイルを同じディレクトリに入れmakeを実行すると、実行ファイル"chart"ができる。

[実行例]

```
$ make
cc -g -c chart.c
cc -g -c chart_util.c
cc -g -c read_datafile.c
cc -g -c test_print.c
cc -g -o chart chart.o  chart_util.o  read_datafile.o  test_print.o
```

第３章

実行方法

解析方法(下降型or上昇型)、探索方式(縦型or横型)は、起動時オプションにより指定する。

[起動時オプション]

```
-t              :下降型解析
-u              :上昇型解析
-b              :横型探索
-d              :縦型探索
-g FILE-NAME    :文法ファイルの指定
-s FILE_NAME    :入力文ファイルの指定
```

[起動例]

文法ファイル"grammer5.dat"と入力文ファイル"sentence.dat"を読み込み、下降型縦型の解析を行う。

```
$ chart -t -d -g grammer5.dat -s sentence.dat
```

[実行結果例]

文法ファイル:"grammer5.dat"

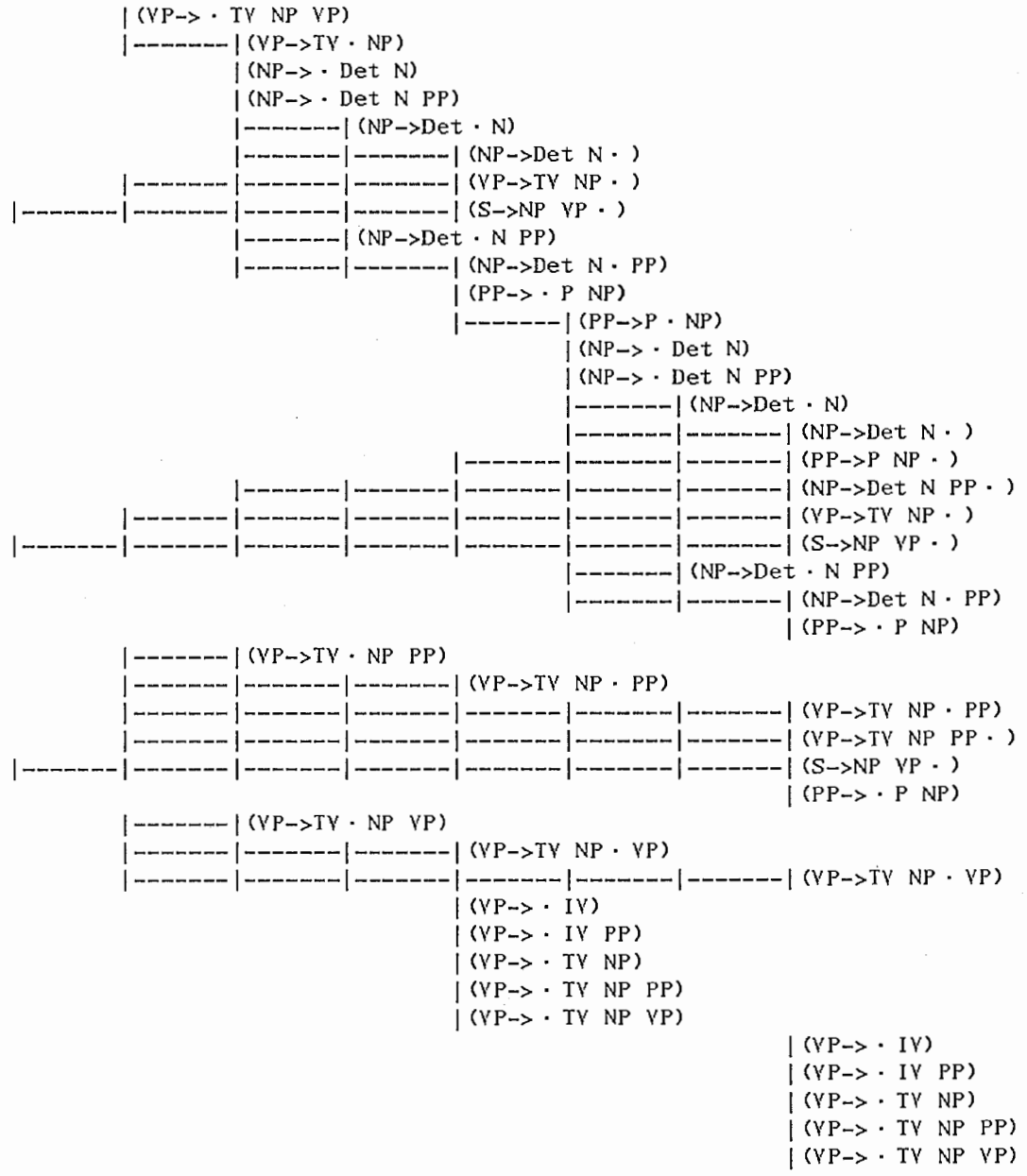
```
RULE S -> NP VP;
RULE VP -> IV;
RULE VP -> IV PP;
RULE VP -> TV NP;
RULE VP -> TV NP PP;
RULE VP -> TV NP VP;
RULE NP -> Det N;
RULE NP -> Det N PP;
RULE PP -> P NP;
WORD the = Det;
WORD her = Det;
WORD her = NP;
WORD they = NP;
WORD nurses = NP;
WORD nurses = N;
WORD book = N;
WORD travel = N;
WORD report = N;
WORD hear = IV;
WORD hear = TV;
WORD see = TV;
WORD on = P;
```

入力文ファイル:"sentence.dat"

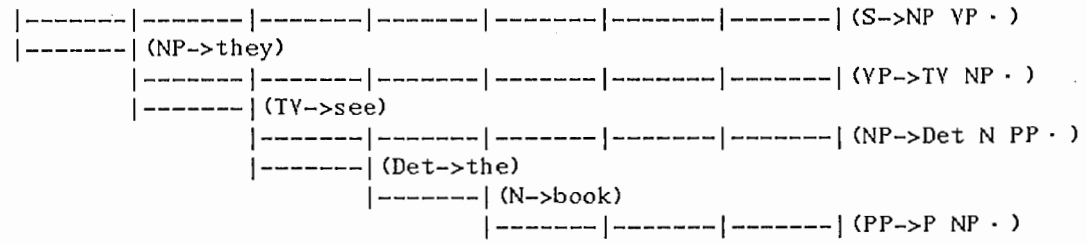
```
They see the book on the nurses.
```

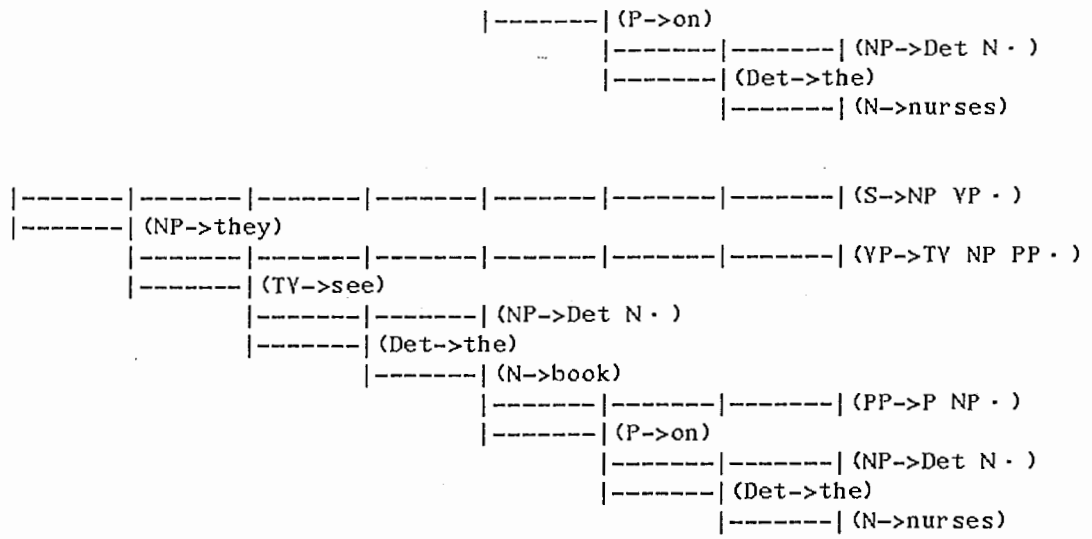
実行:

```
$ chart -t -d -g grammer5.dat -s sentence.dat

 |-------|(NP->they)                                  /* 生成エッジを逐次表示 */
         |-------|(TV->see)
                 |-------|(Det->the)
                         |-------|(N->book)
                                 |-------|(P->on)
                                         |-------|(Det->the)
                                                 |-------|(NP->nurses)
                                                 |-------|(N->nurses)

 |(S-> · NP VP)
 |-------|(S->NP · VP)
 |(NP-> · Det N)
 |(NP-> · Det N PP)
         |(VP-> · IV)
         |(VP-> · IV PP)
         |(VP-> · TV NP)
         |(VP-> · TV NP PP)
```




```
 they    see     the     book    on      the     nurses  /* 解析結果表示 */
0------ 1------ 2------ 3------ 4------ 5------ 6------ 7

>>>>>>> RESULTS <<<<<<<
```





第4章

プログラム

4.1 ファイル構成

各ファイルには、おおまかに以下の内容の関数、データ定義等が記述してある。

"chart.c"	:main()関数の他、CHARTアルゴリズムを実行する関数群
"chart_util.c"	:メモリ管理、ストリング管理の関数群
"read_datafile.c"	:文法ファイル、入力文ファイルの読み込み関数
"test_print.c"	:デバッグ表示用の関数群
"chart.h"	:CHART用のデータ構造他
"chart_util.h"	:標準データタイプのtypedef他

4.2 データ構造

[エッジ]

```
struct _cpEDGE {
        cpEDGETYPE      Type;           /* エッジ種別 */
        SHORT           StartVertex;    /* エッジ出発点 */
        SHORT           FinishVertex;   /* エッジ到達点 */
        cpCATEGORY      Label;          /* ラベル */
        cpPCATEGORY_L   pToFind;        /* 未解析標識リスト */
        cpPCATEGORY_L   pFound;         /* 解析済標識リスト */
        union _Contents {
                cpPRULE         pRule;  /* 該当ルール */
                cpPWORD         pWord;  /* 該当単語(:前終端標識エッジ) */
        }               Contents;       /* エッジの内容 */
        cpPDAUGHTER_L   pDaughter;      /* 娘ノードに相当するエッジリスト */
        cpPEDGE         pNext;
};
```

```
enum _cpEDGETYPE {
        TERMINAL_SYMBOL_EDGE,           /* 前終端標識エッジ */
        INACTIVE_EDGE,                  /* 非活性エッジ */
        ACTIVE_EDGE                     /* 活性エッジ */
};

enum _cpCATEGORY {
        CATEGORY_S,                     /* 文 */
        CATEGORY_NP,                    /* 名詞句 */
        CATEGORY_VP,                    /* 動詞句 */
        CATEGORY_PP,                    /* 前置詞句 */
        CATEGORY_N,                     /* 名詞 */
        CATEGORY_IV,                    /* 自動詞 */
        CATEGORY_TV,                    /* 他動詞 */
        CATEGORY_P,                     /* 前置詞 */
        CATEGORY_D,                     /* 冠詞 */
        CATEGORY_NIL,                   /* 該当品詞無し */
        CATEGORY_NUM
};

struct _cpCATEGORY_L {
        cpCATEGORY      Category;       /* 標識 */
        cpPCATEGORY_L   pNext;
};

struct _cpDAUGHTER_L {
        cpPEDGE         pEdge;          /* 娘ノードに相当するエッジ */
        cpPDAUGHTER_L   pNext;
};

struct _cpEDGE_L {
        cpPEDGE         pEdge;          /* cpEDGE構造体へのポインタ */
        cpPEDGE_L       pNext;
};
```

以上の構造体により、エッジを管理する。CHARTは、cpEDGE構造体リスト(メンバpNextにより連結)により管理される全エッジの集合として扱う。このリストの先頭(pFirstEdge)と末尾(pLastEdge)はグローバル変数により管理する。また、処理の便宜上、cpEDGE_L構造体を用いてINACTIVE_EDGEおよびTERMINAL_SYMBOL_EDGEのみを管理する非活性エッジリストも使用する。このリストの先頭(pFirstInactiveEdge)と末尾(pLastInactiveEdge)もグローバル変数により管理する。

[辞書]

```
struct _cpRULE {
        cpCATEGORY      LhsCategory;    /* 標識(:文法ルール左辺) */
        cpCATEGORY      RhsCategory[RHS_MAX_NUM];
                                        /* 標識群(:文法ルール右辺) */
        cpPRULE         pNext;
};

struct _cpWORD {
        cpCATEGORY      Category;       /* 標識(:単語) */
        PCHAR           pString;        /* 単語文字列 */
        cpPWORD         pNext;
};
```

以上の構造体により、文法(句構造規則)および単語を管理する。なお便宜上、句構造規則において、RHS_MAX_NUM(=3)により右辺の標識数の上限を設定しているが、増減させることに問題はない。
句構造規則リストの先頭(pFirstRule)、単語リストの先頭(pFirstWord)はそれぞれグローバル変数により管理する。

[文]

```
struct _cpSENTENCE {
        PCHAR           pString;        /* 単語文字列 */
        cpPSENTENCE     pNext;          /* =NULL で文末 */
};
```

以上の構造体により、解析対象の文を管理する。入力文読み込み関数が、入力文ファイルから1文を読み込み、単語毎をリスト構造に展開する。このリストの先頭(pSentenceTop)はグローバル変数により管理する。

[アジェンダ]

```
struct _cpAGENDA {
        cpPEDGE         pEdge;          /* エッジ */
        cpPPARENTAGENDA pParentAgenda;  /* 親エッジ */
};

struct _cpPARENTAGENDA {
        cpPEDGE         pEdge;          /* エッジ */
        cpPPARENTAGENDA pParentAgenda;  /* 親エッジ */
        SHORT           DaughterCnt;    /* 埋め込み解析中の娘エッジの数 */
};

cpAGENDA        AgendaTable[AGENDA_MAX_NUM];
                                        /* AGENDA管理テーブル */
```

以上の構造体により、アジェンダ(解析対象エッジ)を管理する。cpAGENDA構造体は、下降型解析においてアジェンダテーブル(縦型ではスタック処理、横型ではキュー処理)として利用する。cpPARENTAGENDA構造体は、下降型解析において埋め込み解析の発生時に、当該アジェンダ(親エッジ)の一時退避に用いる。

4.3 処理概要

[用語]

・原始CHART　　:語をラベルとするエッジのみで構成されるCHART。
・基本CHART　　:原始CHARTに前終端標識をラベルとするエッジを加えてできるCHART。

[上昇型横型:関数BottomUpBreadthFirst()]
(1)基本CHARTを生成する。
(2)CHARTの左端から右に向け、この時点で存在する非活性エッジの組合せを右辺とする句構造規則から新たに非活性エッジを生成する。ただし、生成された非活性エッジとの組合せは行わない。
(3)新たに生成された非活性エッジも対象にして(2)を繰り返す。
(4)新たに生成される非活性エッジがなくなれば処理終了。

[上昇型縦型:関数BottomUpDepthFirst()]
(1)原始CHARTの左端から使っていないエッジを1つ取りだす。
(2)非活性エッジの組合せを右辺とする句構造規則から新たに非活性エッジを生成する。
(3)新たに生成された非活性エッジも対象にして(2)を繰り返す。
(4)新たに生成される非活性エッジがなくなれば(1)に戻り、最も左にある使っていない原始CHARTのエッジをひとつ対象に加え、(2)以降を繰り返す。
(5)使っていない原始CHARTのエッジがなくなり、新たに生成される非活性エッジがなくなれば処理終了。

[下降型横型:関数TopDownBreadthFirst()]
(1)基本CHARTを生成する。
(2)CHART左端にSを左辺に持つ句構造規則に対応するヌル活性エッジを加え、アジェンダテーブル(キュー)に入れる。
(3)アジェンダテーブルから最古アジェンダを1つ取りだす(FirstInFirstOut)。
(4)アジェンダ(活性エッジ)のToFind左端標識と同じラベルを持ち、アジェンダのFinishVertexと自身のStartVertexが同じ非活性エッジを検索し、あればその標識をFoundに移したエッジを新たに生成する。また、それが活性エッジであればアジェンダテーブル(キュー)に入れる。
(5)検索対象の非活性エッジがなくなるまで(4)を繰り返す。
(6)アジェンダ(活性エッジ)のToFind左端標識と同じ標識を左辺に持つ句構造規則を検索し、あればその句構造規則に対応するヌル活性エッジをアジェンダのFinishVertexの位置に生成する。生成したヌル活性エッジをアジェンダテーブルに入れる。現アジェンダは、新たに生成したヌル活性エッジの処理(埋め込み解析)が終了するまで退避しておく。
(7)検索対象の句構造規則がなくなるまで(6)を繰り返す。
(8)アジェンダがなくなるまで(3)以降を繰り返す。アジェンダがなくなれば処理終了。

[下降型縦型:関数TopDownDepthFirst()]
(1)基本CHARTを生成する。
(2)CHART左端にSを左辺に持つ句構造規則に対応するヌル活性エッジを加え、アジェンダテーブル(スタック)に入れる。
(3)アジェンダテーブルから最新アジェンダを1つ取りだす(LastInFirstOut)。
(4)アジェンダ(活性エッジ)のToFind左端標識と同じラベルを持ち、アジェンダのFinishVertexと自身のStartVertexが同じ非活性エッジを検索し、あればその標識をFoundに移したエッジを新たに生成する。また、それが活性エッジであればアジェンダテーブル(スタック)に入れる。
(5)検索対象の非活性エッジがなくなるまで(4)を繰り返す。
(6)アジェンダ(活性エッジ)のToFind左端標識と同じ標識を左辺に持つ句構造規則を検索し、あればその句構造規則に対応するヌル活性エッジをアジェンダのFinishVertexの位置に生成する。生成したヌル活性エッジをアジェンダテーブルに入れる。現アジェンダは、新たに生成したヌル活性エッジの処理(埋め込み解析)が終了するまで退避しておく。
(7)検索対象の句構造規則がなくなるまで(6)を繰り返す。
(8)アジェンダがなくなるまで(3)以降を繰り返す。アジェンダがなくなれば処理終了。

4.4 関数

「付録A 関数概要」を参照。

第5章

句構造規則および単語辞書、解析対象文

句構造規則および単語辞書は、次の規則に従って記述する。記述ファイルは、関数ReadGrammarFile()により起動時に読み込まれ、リスト構造に展開される。読み込みファイル名は、

オプション-g FILE_NAMEで指定する。

[句構造規則]

```
句構造規則      :RULE 標識 -> 標識列;
標識            :S | NP | VP | PP | N | IV | TV | P | Det
標識列          :標識 [標識列]

単語            :WORD 文字列 = 標識;
文字列          :(任意の単語を表す文字列)
```

[例]

```
RULE S -> NP VP;
RULE VP -> TV NP PP;
RULE NP -> Det N;
RULE PP -> P NP;
WORD they = NP;
WORD see = TV;
WORD on = P;
WORD the = Det;
WORD book = N;
```

解析対象文は、次の規則に従って記述する。記述ファイルは、関数ReadSentenceFile()により起動時に読み込まれ、リスト構造に展開される。読み込みファイル名は、オプション-s FILE_NAMEで指定する。なお、読み込んだ文字列は、大文字小文字の区別なく、以降すべて小文字として扱う。

[文]

```
文              :単語列.
単語列          :文字列 [単語列]
文字列          :(任意の単語を表す文字列)
```

[例]

```
They see the book on the nurses.
```

参考文献

[1]「自然言語処理の基礎技術」野村浩郷著(社団法人　電子情報通信学会)ISBN4-88552-070-3
[2]「Natural Language Processing in PROLOG(An Introduction to Computational Linguistics)」Gerald Gazdar,Chris Mellish著(Addison-Wesley Publishing Company)ISBN0-201-18053-7

付録

関数概要

```
/*************************************************************************/
/*      FILE-ID         : "chart.c"                                      */
/*      CONTENTS        : チャートパーザーのプロトタイププログラム       */
/*************************************************************************/
```

■SHORT main(SHORT argc, PCHAR argv[])
[機能]
チャートパーザーのmain関数。
[オプション]
-g 文法ファイル(デフォルトは"grammar5.dat")
-s 入力文ファイル(デフォルトは"sentence.dat")
-u (上昇型解析、デフォルト)
-t (下降型解析)
-b (横型解析、デフォルト)
-d (縦型解析)

■SHORT InitializeChart(VOID)
[機能]
上昇型横型および下降型横型、縦型解析の場合、基本CHARTを作成する。さらに、トップダウン解析の場合、Sをラベルに持つACTIVEエッジを生成するとともに、アジェンダテーブルへPUTする。
[戻り値]
SUCCESS(=0)　　: 成功

■SHORT StartChartParser(VOID)
[機能]
ParseModeに従い、それぞれのCHARTアルゴリズムによる構文解析を行う。
[戻り値]
SUCCESS(=0)　　: 成功

■SHORT BottomUpBreadthFirst(VOID)
[機能]
上昇型横型の解析を行う。処理内容は以下のとおり。解析の実処理は、関数BottomUp()を呼び出す。
(1)基本CHARTを作成する。
(2)解析開始時に存在するINACTIVEエッジにより右辺が満足されるルールから新たにINACTIVEエッジを生成する。
(3)INACTIVEエッジが生成されなくなるまで(2)を繰り返す。
[戻り値]
SUCCESS(=0)　　: 成功

■SHORT BottomUpDepthFirst(VOID)
[機能]
上昇型縦型の解析を行う。処理内容は以下のとおり。解析の実処理は、関数BottomUp()を呼び出す。
(1)原始CHARTからひとつINACTIVE(TERMINAL_SYMBOL)エッジを生成する。
(2)解析開始時に存在するINACTIVEエッジにより右辺が満足されるルールから新たにINACTIVEエッジを生成する。
(3)INACTIVEエッジが生成されなくなるまで(2)を繰り返す。
(4)(1)に戻り、次のINACTIVE(TERMINAL_SYMBOL)エッジを生成し以降の処理を繰り返す。新たに生成するINACTIVE(TERMINAL_SYMBOL)エッジがなくなれば処理終了。
[戻り値]
SUCCESS(=0)　　: 成功

■SHORT BottomUp(cpPEDGE pEdge, cpPCATEGORY_L pCategoryL, cpPDAUGHTER_L pDaughterL,
cpPEDGE_L pSearchEndEdgeL)
[機能]
上昇型解析を行う。処理内容は以下のとおり。
(1)自身pEdgeを標識リストpCategoryLに加える。
(2)標識リストpCategoryLと同じ内容を右辺に持つルールから左辺をラベルとするINACTIVEエッジを生成する。ただし、既に同じ内容のINACTIVEエッジが存在する場合は生成しない。
(3)INACTIVEエッジが生成されなくなるまで(2)を繰り返す。
(4)pEdgeのすぐ右側にINACTIVEエッジがあれば、これをpEdgeにセットしてBottomUp()を再帰呼び出しする。

(5)右側のINACTIVEエッジがなくなるまで(4)を繰り返す。
[戻り値]
SUCCESS(=0)　　：成功

■SHORT TopDownDepthFirst(VOID)
[機能]
下降型縦型解析を行う。処理内容は以下のとおり。
(1)アジェンダテーブルから最新アジェンダを1つGETする(LastInFirstOut)。
(2)アジェンダ(ACTIVEエッジ)のToFind左端標識と同じラベルを持ち、アジェンダのFinishVertexと自身のStartVertexが同じINACTIVEエッジを検索し、あればその標識をFoundに移したエッジを新たに生成する。また、それがACTIVEエッジであればアジェンダテーブル(スタック)へPUTする。
(3)検索対象のINACTIVEエッジがなくなるまで(2)を繰り返す。
(4)アジェンダ(ACTIVEエッジ)のToFind左端標識と同じ標識を左辺に持つ句構造規則を検索し、あればその句構造規則に対応するACTIVEエッジをアジェンダのFinishVertexの位置に生成する。生成したACTIVEエッジをアジェンダテーブルにPUTする。現アジェンダは、新たに生成したACTIVEエッジの処理(埋め込み解析)が終了するまで退避しておく。
(5)検索対象の句構造規則がなくなるまで(4)を繰り返す。
(6)アジェンダがなくなるまで(1)以降を繰り返す。アジェンダがなくなれば処理終了。
[戻り値]
SUCCESS(=0)　　：成功

■SHORT TopDownBreadthFirst(VOID)
[機能]
(1)アジェンダテーブルから最古アジェンダを1つGETする(FirstInFirstOut)。
(2)アジェンダ(ACTIVEエッジ)のToFind左端標識と同じラベルを持ち、アジェンダのFinishVertexと自身のStartVertexが同じINACTIVEエッジを検索し、あればその標識をFoundに移したエッジを新たに生成する。また、それがACTIVEエッジであればアジェンダテーブル(キュー)にPUTする。
(3)検索対象のINACTIVEエッジがなくなるまで(2)を繰り返す。
(4)アジェンダ(ACTIVEエッジ)のToFind左端標識と同じ標識を左辺に持つ句構造規則を検索し、あればその句構造規則に対応するACTIVEエッジをアジェンダのFinishVertexの位置に生成する。生成したACTIVEエッジをアジェンダテーブルにPUTする。現アジェンダは、新たに生成したACTIVEエッジの処理(埋め込み解析)が終了するまで退避しておく。
(5)探索対象の句構造規則がなくなるまで(4)を繰り返す。
(6)アジェンダがなくなるまで(1)以降を繰り返す。アジェンダがなくなれば処理終了。
[戻り値]
SUCCESS(=0)　　：成功

■SHORT SearchRightEdgeFromEdge(cpPEDGE pEdge, cpPEDGE *ppRightEdge,
　　cpPEDGE_L pSearchEndEdgeL, SHORT SearchMode)
[機能]
pEdgeのすぐ右側にあるINACTIVEエッジを検索する。SearchModeにより2種類の検索を行う。
SEARCH_INITIAL　：エッジリストの先頭からpSearchEndEdgeLまでを検索する。
SEARCH_CONTINUE：エッジリストの*ppRightEdgeの次のエッジからpSearchEndEdgeLまでを検索する。
[戻り値]
SUCCESS(=0)　　：検索成功
FAIL(=!SUCCESS)：該当するINACTIVEエッジなし

■SHORT SearchRuleFromLabel(cpCATEGORY Label, cpPRULE *ppRule, SHORT SearchMode)
[機能]
Labelと同じ内容を左辺に持つルールを検索する。SearchModeにより2種類の検索を行う。
SEARCH_INITIAL　：ルールリストの先頭から検索を開始する。
SEARCH_CONTINUE：ルールリストの*ppRuleの次のルールから検索を開始する。
[戻り値]
SUCCESS(=0)　　：検索成功
FAIL(=!SUCCESS)：該当するルールなし

■SHORT SearchRuleFromCategoryL(cpPCATEGORY_L pCategoryL, cpPRULE *ppRule,
SHORT SearchMode)
[機能]
pCategoryLと同じ内容を右辺に持つルールを検索する。SearchModeにより2種類の検索を行う。
SEARCH_INITIAL : ルールリストの先頭から検索を開始する。
SEARCH_CONTINUE : ルールリストの*ppRuleの次のルールから検索を開始する。
[戻り値]
SUCCESS(=0) : 検索成功
FAIL(=!SUCCESS) : 該当するルールなし

■SHORT DeleteZCategoryL(cpPCATEGORY_L pCategoryL)
[機能]
標識リストpCategoryL末尾の構造体を削除する。
[戻り値]
SUCCESS(=0) : 成功

■SHORT DeleteZDaughterL(cpPDAUGHTER_L pDaughterL)
[機能]
標識リストpDaughterL末尾の構造体を削除する。
[戻り値]
SUCCESS(=0) : 成功

■SHORT ParseSuccessCheck(cpPEDGE pEdge)
[機能]
pEdgeが解析結果としての条件を満足しているかをチェックする。
[戻り値]
TRUE(=1) : 真
FALSE(=0) : 偽

■SHORT SearchInactiveEdgeFromToFind(cpPEDGE pActiveEdge, cpPEDGE *ppInactiveEdge,
SHORT SearchMode)
[機能]
pActiveEdgeのpToFind先頭の標識と同じラベルを持つINACTIVEエッジを検索する。SearchModeにより2種類の検索を行う。
SEARCH_INITIAL : エッジリストの先頭から検索を開始する。
SEARCH_CONTINUE : エッジリストの*ppInactiveEdgeの次のエッジから検索を開始する。
[戻り値]
SUCCESS(=0) : 検索成功
FAIL(=!SUCCESS) : 該当するINACTIVEエッジなし

■SHORT SearchInactiveEdgeFromRule(cpPRULE pRule, SHORT StartVertex, cpPEDGE *ppInactiveEdge)
[機能]
pRuleとStartVertexに一致するINACTIVEエッジを検索する(->INACTIVEエッジあればpRuleの解析を行わない)。
[戻り値]
SUCCESS(=0) : 検索成功
FAIL(=!SUCCESS) : 該当するINACTIVEエッジなし

■SHORT CreateEdge(cpEDGETYPE Type, SHORT StartVertex, SHORT FinishVertex,
cpCATEGORY Label, cpPCATEGORY_L pToFind, cpPCATEGORY_L pFound,
PVOID pContents, cpPDAUGHTER_L pDaughter, cpPEDGE *ppEdge)
[機能]
引き数の内容を持つエッジを生成する。
[戻り値]
SUCCESS(=0) : 成功

■SHORT CopyEdge(cpPEDGE pEdge, cpPEDGE *ppNewEdge)
[機能]
pEdgeと同じ内容を持つエッジを生成する。
[戻り値]
SUCCESS(=0)　　: 成功

■SHORT DestroyEdge(cpPEDGE pEdge)
[機能]
pEdgeが使用しているメモリエリアを解放する。
[戻り値]
SUCCESS(=0)　　: 成功

■SHORT AddZEdgeL(cpPEDGE pEdge)
[機能]
エッジリスト末尾にpEdgeを追加する。
[戻り値]
SUCCESS(=0)　　: 成功

■SHORT AddZInactiveEdgeL(cpPEDGE pEdge)
[機能]
INACTIVEエッジリスト末尾にcpPEDGE_L構造体を追加し、pEdgeを格納する。
[戻り値]
SUCCESS(=0)　　: 成功

■SHORT CheckEdgeExist(cpPEDGE pEdge)
[機能]
エッジリストにpEdgeと同じ内容のエッジが存在するかをチェックする。
[戻り値]
TRUE(=1)　　: 真(既存)
FALSE(=0)　　: 偽

■SHORT AddZCategoryL(cpPCATEGORY_L *ppCategoryL, cpPCATEGORY Category)
[機能]
AllocしたcpCATEGORY_L構造体にCategoryを入れ、標識リスト(先頭:*ppCategoryL)の末尾に追加する。*ppCategoryLがNULLの場合、cpCATEGORY_L構造体のアドレスを*ppCategoryLに返す。
[戻り値]
SUCCESS(=0)　　: 成功

■SHORT DeleteACategoryL(cpPCATEGORY_L *ppCategoryL)
[機能]
標識リスト(先頭:*ppCategoryL)先頭のcpCATEGORY_L構造体を切り離し、それの使用メモリを解放する。
*ppCategoryLには、次のcpCATEGORY_L構造体のアドレスを返す。
[戻り値]
SUCCESS(=0)　　: 成功

■SHORT ToFindCategoryWasFound(cpPEDGE pEdge)
[機能]
pEdgeのpToFind先頭の標識をpFound末尾へ移動する。
[戻り値]
SUCCESS(=0)　　: 成功

■SHORT AddZDaughterL(cpPDAUGHTER_L *ppDaughter, cpPEDGE pEdge)
[機能]
AllocしたcpDAUGHTER_L構造体にpEdgeを入れ、娘エッジリスト(先頭:*ppDaughterL)の末尾に追加する。
*ppDaughterLがNULLの場合、cpDAUGHTER_L構造体のアドレスを*ppDaughterLに返す。
[戻り値]
SUCCESS(=0)　　: 成功

■SHORT CreateBasicChart(VOID)
[機能]
入力文(pSentenceTop)と単語辞書(pFirstWord)から基本チャート(BasicChart)を作成する。
[戻り値]
SUCCESS(=0)　　: 成功

■SHORT PutZAgenda(cpPEDGE pEdge, cpPPARENTAGENDA pParentAgenda)
[機能]
アジェンダテーブルの末尾にpEdgeとpParentAgendaを格納する。
[戻り値]
SUCCESS(=0)　　: 成功
[注意]
アジェンダテーブルがオーバーフローした場合、プロセスを終了する。

■SHORT PutAAgenda(cpPEDGE pEdge, cpPPARENTAGENDA pParentAgenda)
[機能]
アジェンダテーブルの先頭にpEdgeとpParentAgendaを格納する。
[戻り値]
SUCCESS(=0)　　: 成功
[注意]
アジェンダテーブルがオーバーフローした場合、プロセスを終了する。

■SHORT GetAAgenda(cpPAGENDA pAgenda)
[機能]
アジェンダテーブルの先頭からアジェンダを1つ取り出す。
[戻り値]
SUCCESS(=0)　　: 成功
FAIL(=!SUCCESS) : 失敗(アジェンダテーブルが空の場合)

■SHORT GetZAgenda(cpPAGENDA pAgenda)
[機能]
アジェンダテーブルの末尾からアジェンダを1つ取り出す。
[戻り値]
SUCCESS(=0)　　: 成功
FAIL(=!SUCCESS) : 失敗(アジェンダテーブルが空の場合)

■SHORT SaveParentAgenda(cpPAGENDA pAgenda, SHORT DaughterCnt, cpPPARENTAGENDA *ppParentAgenda)
[機能]
埋め込み解析発生時にアジェンダを親アジェンダとして退避させる。
[戻り値]
SUCCESS(=0)　　: 成功

■SHORT DeleteParentAgenda(cpPPARENTAGENDA pParentAgenda)
[機能]
埋め込み解析用に退避した親アジェンダの使用メモリを解放する。
[戻り値]
SUCCESS(=0)　　: 成功

```
/*****************************************************************************/
/*      FILE-ID          : "chart_util.c"                                    */
/*      CONTENTS         : チャートパーザーのユーティリティプログラム        */
/*****************************************************************************/
```

■SHORT AllocMemory(ULONG AreaSize, PVOID *ppAllocArea)
[機能]
AreaSizeバイトの領域を確保し、そのアドレスを*ppAllocAreaに返す。
[戻り値]
SUCCESS(=0)　　: 成功
FAIL(=!SUCCESS) : 失敗

■SHORT FreeMemory(PVOID pFreeArea)
[機能]
pFreeAreaの指す領域を解放する。
[戻り値]
SUCCESS(=0)　　: 成功
FAIL(=!SUCCESS) : 失敗(pFreeAreaがNULLの場合)

■SHORT AddString(PCHAR pString, PCHAR *ppAllocArea)
[機能]
pStringの先頭からヌル文字('0')まで分の領域を確保し、文字列をその領域にコピーする。領域のアドレスを*ppAllocAreaに返す。
[予定]
将来的には文字列の管理を行い、等しい文字列には同じアドレスを返す。
[戻り値]
SUCCESS(=0)　　: 成功
[注意]
領域の確保に失敗した場合、プロセスを終了する。

■SHORT StringCompare(PCHAR pString1, PCHAR pString2)
[機能]
文字列pString1とpString2を比較する。一致した場合、TRUEを返す。
[戻り値]
TRUE(=1)　　: 一致(Match)
FALSE(=0)　　: 不一致(UnMatch)

■SHORT StringCompareNotAa(PCHAR pString1, PCHAR pString2)
[機能]
アルファベット大/小文字の区別をせずに、文字列pString1とpString2を比較する。一致した場合、TRUEを返す。
[戻り値]
TRUE(=1)　　: 一致(Match)
FALSE(=0)　　: 不一致(UnMatch)

```
/*************************************************************************/
/*      FILE-ID          : "read_datafile.c"                             */
/*      CONTENTS         : 辞書&文法データの読み込み                     */
/*      SPECIAL THANKS   : Mr.Hayashi                                    */
/*************************************************************************/
```
■SHORT ReadSentenceFile(PCHAR pFileName, cpPSENTENCE *ppSentence)
[機能]
入力文ファイルを読み込み、cpSENTENCE構造体のリストに展開する。先頭の構造体へのポインタを*ppSentenceに返す。
サブ関数GetWord()がピリオド('.')までの文字列を順次切り出すので、入力文はピリオドまでの文となる。なお、文中の英大文字は小文字に変換する。
[戻り値]
TRUE(=1)　　: 一致(Match)
FALSE(=0)　　: 不一致(UnMatch)
[注意]
単語辞書にない単語が文中に現れた場合、プロセスを終了する。
構造体領域の確保に失敗した場合、プロセスを終了する。

■SHORT ReadGrammarFile(PCHAR pFileName, cpPRULE *ppFirstRule,
cpPWORD *ppFirstWord)
[機能]
文法ファイルを読み込み、cpRULEおよびcpWORD構造体のリストに展開する。先頭の構造体へのポインタをそれぞれ*ppFirstRule,*ppFirstWordに返す。
[書式]
句構造規則　　:RULE 標識 -> 標識列;
単語　　:WORD 文字列 = 標識;
標識　　:S | NP | VP | PP | N | IV | TV | P | Det
標識列　　:標識 [標識列]
文字列　　:(任意の単語を表す文字列)
[戻り値]
SUCCESS(=0)　　: 成功

■PCHAR GetWord(PCHAR pBuf, PCHAR pBufEnd, CHAR String[])
[機能]
領域pBufの先頭から区切り文字(改行'n',タブ't',ブランク' ',カンマ',')、および文法終了文字(セミコロン';')、文終了文字(ピリオド'.')までを切り出し、Stringに返す。
先頭が区切り文字で始まる場合は、区切り文字以外が現れるまで読み飛ばす。
先頭が文法終了文字で始まる場合は、文法終了文字のみを切り出し、Stringに返す。
先頭が文終了文字で始まる場合は、バッファの終了と同じくNULLを関数戻り値とする。
先頭が'#'で始まる場合は、コメント行と解釈し、改行文字'n'までを読み飛ばす。
[戻り値]
次回のpBuf　　: 読み込んだバッファ内容の次バイト目へのポインタ
[注意]
文法終了文字および文終了文字が先頭以外に現れた場合は、バッファポインタを1戻し(=次回の先頭になるようにして)、そこまでの文字列をStringに返す。

■static SHORT　GetCategoryFromString(PCHAR pString, cpPCATEGORY pCategory)
[機能]
標識文字列pStringから内部処理用の標識番号を*pCategoryに返す。
[戻り値]
SUCCESS(=0)　　: 成功
FAIL(=!SUCCESS) : 失敗(該当する標識文字列がなかった場合)

```
/*************************************************************************/
/*      FILE-ID          : "test_print.c"                                */
/*      CONTENTS         : チャートパーザー構造体プリント(テスト用)      */
/*************************************************************************/
```
■SHORT PrintCHART(cpPSENTENCE pSentence, cpPEDGE pEdge)
[機能]
入力文、解析結果、TERMINAL_SYMBOL_EDGE、INACTIVE_EDGE、ACTIVE_EDGEを表示する。

■static VOID　DrawEDGE(cpPEDGE pEdge)
[機能]
エッジを表示(図)する。

■static VOID　DrawDaughter(cpPDAUGHTER pDaughter)
[機能]
娘エッジを再帰的に表示(図)する。

■SHORT PrintEDGE_L(cpPEDGE pEdge)
[機能]
エッジリストをpEdgeから順に表示する。

■SHORT PrintEDGE(cpPEDGE pEdge)
[機能]
エッジを表示する。

■SHORT PrintRULE(cpPRULE pRule)
[機能]
文法ルールリストをpRuleから順に表示する。

■SHORT PrintWORD(cpPWORD pWord)
[機能]
単語辞書リストをpWordから順に表示する。

■SHORT PrintSENTENCE(cpPSENTENCE pSentence)
[機能]
入力文(cpSENTENCE構造体リスト)をpSentenceから順に表示する。