自 然 言 語 処 理 82－2
（1991 3. 15）

チャートパージング法による べた書き文の直接解析

井上 準一郎　　二口 邦夫　　寺下 陽一
金 沢 工 業 大 学

日本語などのベタ書き形式テキストに適したチャート型パーサを開発した。単語抽出を含む形態素解析プロセスはパーサに組込まれており，構文解析や意味解析プロセスと協同的に進められる。したがって，形態素解析を分離した２－パス方式に比べ，無駄な解析結果の生成を防ぐことができる。形態素情報，構文規則，および（表層的な）意味規則は同一の素性構造により記述されるようになっている。現在までの実験の結果，このような１－パス方式パーサの実用化は充分可能であると考えられる。

A SINGLE-PATH, CHART-BASED PARSER SUITABLE FOR CONTINUOUS-TEXT LANGUAGES

Junichirou INOUE, Kunio FUTAKUCHI and Yoichi TERASHITA

Kanazawa Institute of Technology

We developed a chart-based parser suitable for continuous-text languages (Japanese, in particular). The parser has an integrated word-extracting module that is executed in parallel with syntactic and semantic analyses, thus eliminating costly pre-processing. The grammatical information, including morphological, syntactic and semantic rules, is described in terms of feature structures. Experiments with the parser has indicated that such a one-path approach to continuous-text analyses can be useful for practical NLP applications.

1．はじめに

我々は、以前にＡＴＮ[1),2)]法に基づいた自然言語処理用のパーサを開発した。そのパーサでは、一般化された意味記述が充分できなかった。そこで今回チャートパージング法[3),4)]を応用したパーサを開発した。

自然言語の解析を行なう場合、トップダウン法とボトムアップ法と大きくわけて２種類の方法がある。解析に際し、トップダウン法では左再帰の問題が、ボトムアップ法ではε－プロダクションの問題が起きてくる。これらの２つを回避するものとして、チャートパージング法というものが、最近よく使用されている。この方法は、vertex と edge という２つの要素からなるグラフをチャートに次々と記録していくことにより解析を進めていくもので、トップダウンでもボトムアップでもどちらでも解析ができるようになっている。従来のパーサではその後の解析に利用できる情報を捨てていたのに比べ、この方法では情報をチャートに記録しておくことにより、同じ構文木を２度も作るような無駄が省ける。特に埋め込み文を含む文では、バックトラックの回数が増え、同じ構文木を何度も作成しようとするが、この方法ではその必要がなく解析時間の短縮がはかられる。構文木をチャートに記録しておくことにより、複数個の構文解析木が可能な場合はそれら全ての構文解析木の出力も可能となる。

問題点は、解析文が長くなる場合チャートが非常に大きくなり、メモリの問題や探索時間が長くなったりすることである。これを回避するため、vertex の start ポイント、finish ポイントそれぞれについて配列を用意し、それぞれについて索引付けを行なった。これにより、active edge と inactive edge に分けて記憶させ、必要なものだけを探索することとし、探索も直接アクセスにより探索することができ、合わせて探索時間の短縮をはかった。

このチャートパージング法は元来、欧米系の言語のように単語が最初から分かち書きされているものに対して開発された手法であるため、べた書きされて入力される習慣のある言語や、語尾が活用する言語に対しては、そのままでは適用が不可である。そこで単語抽出機能を新たに付加した。ただこの際問題となるのは、従来のパーサによく見られるように、単語抽出だけを独立させて行なうと、単語抽出の可能性が非常に多くなり、効率が悪くなる点である。そこで、文法を素性構造という形で表現し、ここに単語間の接続情報、構文情報、意味情報を入れておき、解析を進めていくときにこれらの情報を同時並行的に処理することにより、無駄な単語抽出や構文木の作成を防止しようとした。

素性構造により日本語の意味情報を記述するときには、英語と違ったものになる。英語では表層格のみで意味が記述できるのに対して、日本語の場合、場合によっては深層格にまで立ち入らなければ、意味がはっきりしないことがあるからである。現在のところは、実験段階であるため、辞書に登録している単語数や文法規則の数も少しであるため、探索時間はそれほど問題とならないが、将来、実用的に稼働させる場合のことを考えて、単語そのものや非終端記号でハッシュ化し、直接アクセスできるようにした。

2．チャートパーサ

2．1　チャートパーサの説明

”鬼が来た”という文の解析過程を図１に示す。これらはＷＦＳＴ[4)]と呼ばれる表現法で記述してある。ここにある矢印（edge）は以下のような属性によって構成される。

＜start＞　：始めの vertex の番号
＜finish＞　：終わりの vertex の番号
＜label＞　：生成規則の矢印の左側部分
＜found＞　：既に解析済みの要素のリスト
＜tofind＞　：これから解析を行なおうとしている要素のリスト
＜connect＞　：次に来るべき単語に関する接続

情報

図１において①や②のようなものを vertex といい、曲線の弧を edge という。先に述べた属性を用い、vertex 間を結ぶ edge は次のように表現できる。

｛〈start〉〈finish〉〈label〉→
〈found〉.〈tofind〉〈connect〉｝

図１での edge を上の表記法で記述したものの一部分を下に示す。

｛0　1　AV → た.（renyou V)｝
｛1　2　V → 来る.｝
｛0　2　VP → AV V.｝
｛2　3　P → が.｝
｛0　4　VTS → VP RPS.｝
｛0　4　S → VTS.｝

解析はこの表記法で記述された edge を全てチャートと呼ばれる表に登録をしておき、これを用いて解析を進めていく。＜tofind＞が空のものを inactive edge といい、解析が完了した edge を表わしている。また、空でないものを active edge といい、解析が完了していない egde を表わしている。

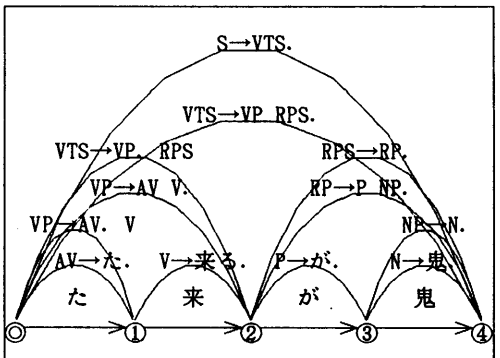

図１　”鬼が来た”の解析の様子

２．２　チャートパーサで使用される規則

○　基本規則

Ａ及びＢを非終端記号、ｗ１、ｗ２、ｗ３を終端記号並びに非終端記号とすると、チャートに［ｉ　ｊ　Ａ→ｗ１．Ｂ　ｗ２］と［ｊ　ｋ　Ｂ→ｗ３．］という edge が存在するならば［ｉ　ｋ　Ａ→ｗ１　Ｂ．ｗ２］という edge をチャートに付け加える。

○　下向き規則

① active edge の初期化にあたって、入力文の全てを受理する非終端記号をＡとすると、文法規則の中に存在するＡ→Ｗという形の全ての規則に対して、［0　0　Ａ→．Ｗ］という edge をチャートに加える。
② もしチャートに［ｉ　ｊ　Ｃ→Ｗ１．Ｂ　Ｗ２］という edge をつけ加えるとき、Ｂ→Ｗなる文法規則があるならそれら全ての規則に対して［ｊ　ｊ　Ｂ→．Ｗ］という edge をチャートにつけ加える。

この他にも上向き規則があるが、今回は全てトップダウンの手法を用いて解析を行なったためこの説明は省略する。

２．３　チャートを使った解析の方法

解析を進めるにあたって、最低限１個の active edge と inactive edge が必要である。まず下向き規則の①を用いて、｛0　0　S →．VTS｝なる edge を作る。次に下向き規則の②を使い、｛0　0　VTS →．VP RPS｝と｛0　0　VP →．AV V｝という edge を次々に作る。ここで”．”の右側部分にはじめて終端記号の AV が現れたため、ここで辞書を参照し、辞書情報により、｛0　1　AV → た.（renyou V)｝なる inactive edge を作る。この edge と｛0　0　VP →．AV V｝なる

active edge を組み合わせて基本規則を用い、｛0 1 VP → AV(た). V (renyou V)｝なる edgeを作る。edge の接続情報より次に来るべき単語が動詞の連用形でなければならないことがわかる。次に”．”の右側部分が V であるので、辞書を参照する。ここで文では次に”来”が接続しており語尾活用処理ルーチンを用いて、”来”が”来る”の連用形であることを見いだし、”来る”で辞書を参照する。辞書には終止形のみを登録しておく。もし”来”が動詞の連用形でない場合、解析はここで失敗する。今回は成功し、｛1 2 V → 来る. ｝なる edge ができる。この edge と先のedgeより｛0 2 VP → AV(た) V(来る). ｝なる edge ができる。ここで基本規則を用いて、｛0 0 VTS →. VP RPS｝と、今作成された edge を組み合わせることにより｛0 2 VTS → VP(AV(た) V (来る)). RPS｝という edge 作る。ここで下向き規則の②を用い、｛2 2 RPS →. RP｝と｛2 2 RP →. P NP｝なる edge を作る。以下同様に P が終端記号であることより辞書を参照し｛2 3 P →が. ｝なる edge を作り、解析を進めていく。最終的に｛0 4 S → VTS(VP(AV(た)V(来る)) RPS(RP(P(が) NP(N(鬼)))))). ｝なる inactive edge が作成され、全ての解析が終了する。このように構文解析と単語抽出を同時並行的に進めていくことにより、誤った単語の並びに対するチェックも行なえる。

3． 素性構造について

3．1 ＤＡＧについて

素性構造を表現するために、ＤＡＧ（Directed Acyclic Graph）がある。図２にＤＡＧの例を示す。これは、

$$\left\{\begin{array}{l}\langle \text{cat}\rangle = \text{N}\\ \langle \text{pred}\rangle = \text{太郎}\\ \langle \text{type}\rangle = \text{person}\\ \langle \text{arg cat}\rangle = \text{X}\end{array}\right\}$$

のような素性構造を表現したものである。”＝”の左側部分が素性ラベル、右側が素性値になっている。

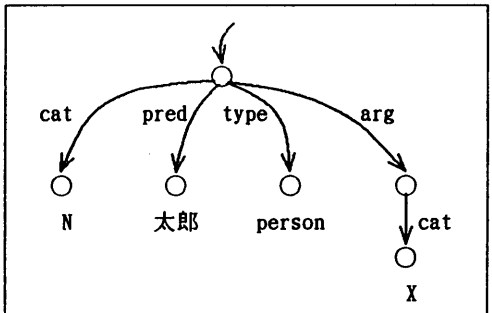

図２ ＤＡＧの例

3．2 Lisp による素性構造表現

前節にあげたＤＡＧは ((cat N)(pred 太郎)(type person)(arg ((cat X)))) のように Lisp ではリスト構造を用いて表現する。

4． 素性構造による意味表現

4．1 意味表現の辞書による記述方法

意味関係の辞書記述については次のように行なう。

○ 名詞

”りんご”

(cat) = N

(sem pred) = りんご

(sem type) = food

○ 動詞

”食べる”

(cat) = v

(cases を sem pred) = (x)･･･を格の標識

(cases を sem type) = food･･を格の意味素性

(tense) = 現在

○ 助詞

”を”

(cat) = p

(z を sem) = (x)

4．2　意味表現の文法による記述方法

意味関係についての文法記述は次のように行なう。

S -〉 V RPS

(S cases) = (V cases)・・・・・格の持ち上げ

(V cases) = (RPS cases)・・・格の照合

RPS -〉 RP

(RPS cases) = (RP y)

RP -〉 P N

(P x) = (N sem)・・・・意味情報の同一化

(RP y) = (P z)・・・・・格の決定と同一化

4．3　文法と辞書による意味の照合

ここでは文法と辞書による意味関係についての照合について実際にどのように行なわれているかを説明する。

例として、”りんごを食べる”なる文を解析するプロセスについて述べる。

始めに”食べる”が解析され、V は格助詞”を”を取り、その type の値は food であることがわかる。ただし、pred の値はまだ未定である。次に”を”を解析した時、V の取る格”を”と一致しているため、さらに解析を進める。この段階では P の情報は RP に引き継がれるということはわかるが、まだ P の値は未定である。さらに、”りんご”を解析したところで (P x)=(N sem) により N のもつ pred=りんご、type=food の情報が P に引き継がれる。また (RP y)=(P z) により、この情報は RP に引き継がれ、さらに、(RPS cases)=(RP y) により RPS にも引き継がれる。(V cases)=(RPS cases) により、V の(cases を sem type) の値 food と、下から持ち上げられた food との照合に成功し、V の(cases を sem pred) に”りんご”という値が入る。最後に、(S cases)=(V cases) により V の意味情報が全て S に引き継がれ、解析は成功し終了する。

以上のことを図3にＤＡＧを用いて表わす。

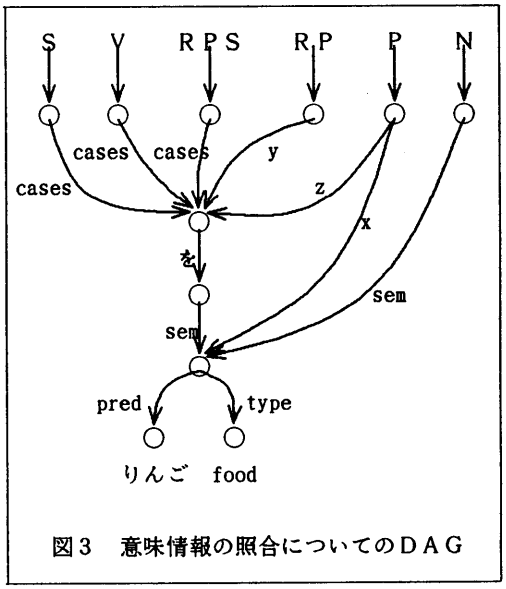

図3　意味情報の照合についてのＤＡＧ

5．　べた書き文処理について

べた書き文での単語抽出を容易にするために、解析文は逆向きに解析していくことにする。これは接続情報が容易に記述できるからである。ここで一例として、”鬼が島から来ました”という文に対して解析を行なう場合について説明する。まずチャートの初期化を行ない、vertex が以下のように打たれる。

た　し　ま　来　ら　か　島　が　鬼

◎→①→②→③→④→⑤→⑥→⑦→⑧→⑨

まず、辞書を参照することにより、｛0　1　AV　→　た.　((renyou AV)(renyou YOUGEN))｝なる edge ができる。これより”た”に接続するものは、助動詞もしくは用言の連用形でなければならない

ことがわかる。次に”まし”を辞書より検索し、これが助動詞の連用形であることより、”まし”の切り出しは成功する。”まし”の辞書情報より、{1 3 AV → まし. (renyou V)} なる edge ができる。次に接続する”来”が動詞”来る”の連用形であることより、”まし”との接続に成功し、{3 4 V → 来る. } なる edge ができる。
以上の過程を図4に示す。

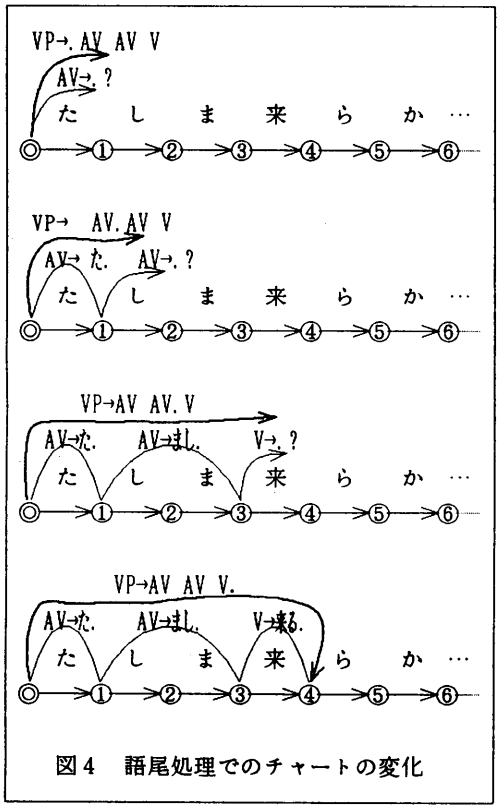

図4 語尾処理でのチャートの変化

次に、さらに解析が進み、vertex 6から9までの”鬼が島”につての解析においては、”鬼”+”が”+”島”と、”鬼が島”の2つの可能性が存在する。この状態を図5に示す。この両方を切り出し、チャートに記録しておくことで、最後に両方についての解析木を出力することが可能となる。

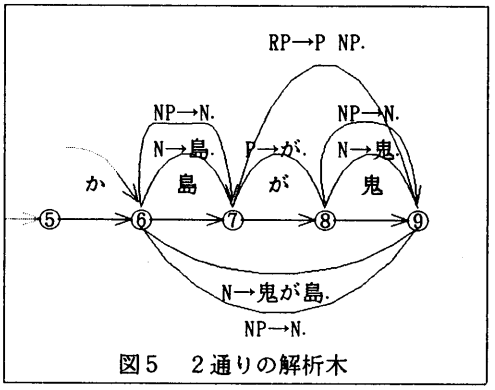

図5 2通りの解析木

6. 高速化のための処置

6. 1 チャートに関する処置

チャートパーサでは途中の解析結果を保存しながら解析を行なうため、チャート自身が大きくなり edge がすでに存在するかの重複チェックを行なうにも時間がかかる。そこで、inactive edge は start (0～n) に、active edge は finish (0～n) の配列を用意しこれらに格納する。ここで、n は解析文の長さ (1文字を1とした場合) である。解析を進めていく上では、inactive edge か active edge のどちらかが必要であり、同時に両方共必要になることがないため、分割して格納しておくほうが効率がよくなる。またこれまでにできた全ての edge は vertex (n, n) の2次元配列を用意し、先のとは別個に格納する。これにより検索は配列の添字を指定するだけでよく、直接アクセスが可能である。

6. 2 文法規則に関する処置

生成規則の左側部分の非終端記号単位で、ハッシュ化を行なう。一例をあげる。
”VP”について2つの生成規則が存在した場合、次のような形で、ハッシュ化する。これにより探索時には、VP で直接アクセスが可能となる。

```
(( VP→V          ) ( VP→V NP       ))
 ( <VP CAT> = VP )  ( <VP CAT> = VP )
 ( <V CAT> = V   )  ( <V CAT> = V   )
                    ( <NP CAT> = NP )
```

6．3　辞書に関する処置

辞書については、単語単位でハッシュ化を行ない、検索時に直接アクセスが行なえるようにする。

”太郎”なる単語は次のようにハッシュ化する。

（ 太郎（cat）= N（type）= person ）

7．　解析実験

以下の３つの例文について解析実験をおこなった。

1．太郎が食堂でりんごを食べた
2．太郎が家を食べた
3．食堂で家へ太郎が食べた

例文についての解析結果をそれぞれ、図６、図７、図８に示す。

例文１については、正しい文であり、図６（b）のような構文木が得られる。例文２は”食べる”にたいする格”を”の意味素性と”家”の意味素性の不一致により解析は失敗する。例文３は、”食べる”についての格”へ”についての情報と文中の情報の不一致により解析は失敗した例である。

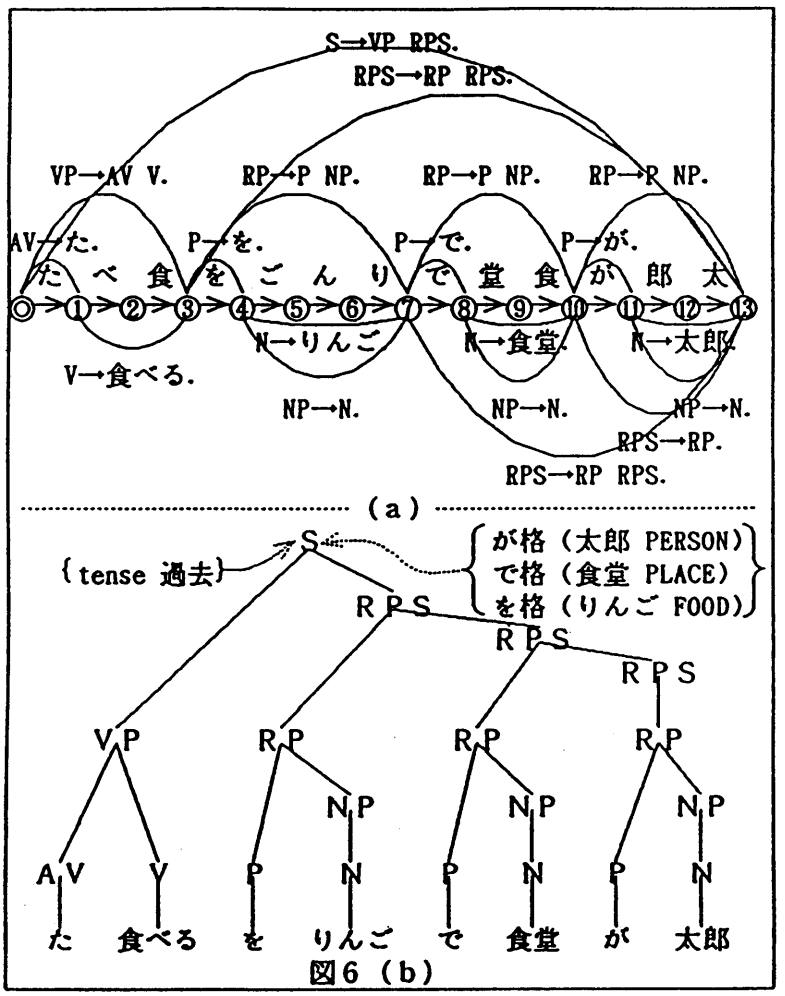

図６（b）

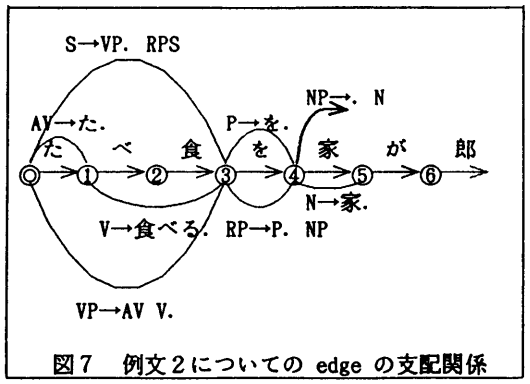

図7　例文2についての edge の支配関係

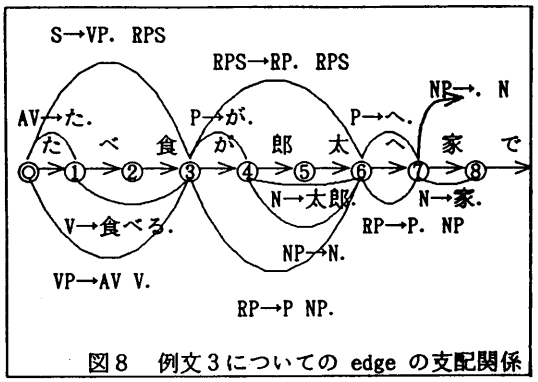

図8　例文3についての edge の支配関係

8.　まとめ

① 単語分かち書きされた文に対するチャートパーサを改良し、日本語などのように、文がべた書き入力されたり、単語が語尾変化する言語に対しても処理できるようにした。

② 単語抽出を単独で行なうことによる危険性（切り出しの単語群が爆発的に増大するおそれ）を回避するために、素性構造を用い、形態素、構文、意味のそれぞれの情報を文法に同時に記述することで、3つのプロセスを同時並行的に進める方法をとった。

③ 素性構造を文法、辞書記述において用いることにより、情報の記述の一般化が容易になった。また、全ての情報を1ヵ所にまとめて記述できるため非常に便利である。意味情報については今回は、格関係と時制関係についてのみ含めたが、他の意味情報も入れて拡張することが可能だと思われる。

④ チャートパージング法を用いることにより、左再帰やε－プロダクションの問題も回避でき、解析の途中結果を逐次チャートに記録し、後の解析で用いる可能性のある情報を捨てずに保存しておくことにより、バックトラックを何度もおこすような解析文の場合、同じ構文木を作成する無駄が省ける。これにより解析効率が非常によくなった。また、チャートの増大にともなう探索時間の増加についての問題も、各種の索引付けを行なうことにより、かなり解決された。

○　参考文献

1）　寺下，二口：分かち・構文・意味の平行処理をおこなう日本語パーサ，情報処理学会「自然言語処理」研究会資料，65-4，pp. 1-8(1988).

2）　二口，寺下：ＬＦＧに基づく並列型パージング法，情報処理学会「自然言語処理」研究会資料，72-6，pp. 1-8(1989).

3）　Algorithm schemata and data structures in syntactic processing. In Readings in Natural Language Processing (Barbara J. Grosz, Karen Sparck Jones and Bonnie Lynn Webber, eds.), pp. 35-70, Morgan Kaufmann：Los Altos, (1986).

4）　Gerald Gazdar and Chris Mellish, "Natural Language Processing in Lisp. An Introduction to Computational Linguistics", Addison-Wesley(1989).