論文 / 著書情報
Article / Book Information

題目(和文)	自然言語処理のための統語的・意味的知識に関する基礎研究
Title(English)	
著者(和文)	徳永健伸
Author(English)	Takenobu Tokunaga
出典(和文)	学位:博士(工学), 学位授与機関:東京工業大学, 報告番号:乙第2241号, 授与年月日:1991年7月31日, 学位の種別:論文博士, 審査員:
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Tokyo Institute of Technology, Report number:乙第2241号, Conferred date:1991/7/31, Degree Type:Thesis doctor, Examiner:
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

自然言語処理のための
統語的・意味的知識に関する基礎研究

1991 年

徳 永 健 伸

$\mathcal{K}$ へ

目 次

第 1 章

序 論

1.1 研究の背景と目的

計算機による自然言語処理の研究分野において, その後の研究にもっとも影響を与えた研究のひとつとして Winograd の SHRDLU[101]があげられよう. SHRDLU は積木の世界というきわめて限定された世界において, 自然言語 (英語) のさまざまな現象を計算機で扱えることを示した. しかしながら, SHRDLU は限定された世界においてあまりに巧みに問題を解いたために, 計算機による自然言語処理に関する問題がすべて解決されたかのような誤解を与えてしまった面もある. 実際には, 積木の世界から現実の世界へ対象領域を広げるには, さまざまな問題を解決しなければならない. 本論文の目的は, 計算機による自然言語処理の対象領域をより現実世界へ近づけるために解決すべき問題を, 処理に必要な知識という側面から検討し, その解決策を与えることである.

計算機による自然言語解析システムの古典的な構成を図 1-1に示す.[44] 計算機に入力された文章は, 形態素解析, 統語解析, 意味解析, 文脈解析などの処理を経て, その文章が表現している意味構造に変換される. 形態素解析は, 入力された文を形態素の並びに分解する処理である. 形態素とは, 言語学の定義にしたがえば文法分析の最小単位であるが ,[78] 計算機による処理という観点からすれば辞書中に登録されている単語とほぼ同義と考えられる. 英語のように単語が空白で区切られる言語では単語の屈折変化形から原型を求める処理が形態素処理の中心となるのに対し, 日本語のように単語の区切りが明白でない言語では単語を切り出す処理が重要となる. 統語解析は形態素の並びから文の統語構造を抽出する処理である. 統語構造は多くの場合, 句構造の形で得られる. また, 句構造を統語木と呼ばれる木構造で表すことも多い. 意味解析は統語解析の結果を用いて 1 文が表す意味を抽出する処理である. 意味解析の結果は, 論理形式, 意味ネットワーク, フレーム表現などの知識表現形式で表現される. 意味解析まで

の処理は主として 1 文の情報のみを用いておこなうが, 文脈解析は, それまでに入力された文から得られた情報を用いて現在の入力文の意味を解釈する処理である. たとえば, 指示対象の同定, 省略語の補完などの処理が含まれる. 基本的にこれらの 4 つの処理は, 前の処理の結果を主に使って次の処理をおこなうという関係にあるが, 全体の処理をこの順に逐次的におこなわなければならないというわけではない. 正しい形態素分割をおこなうためには統語的, 意味的, さらには文脈的情報が必要になることもある. しかし, 伝統的には各処理をモジュールとして設計し, 逐次的に処理をおこなうことが多い.

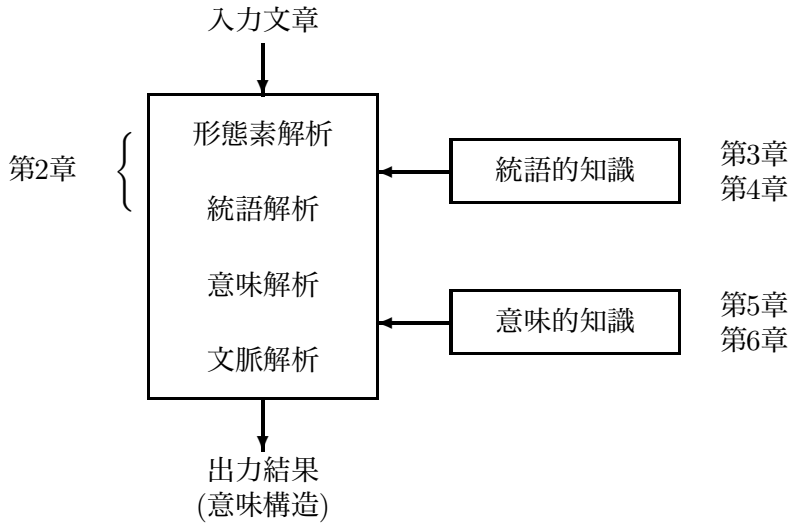


図 **1-1**　自然言語解析システムの構成

このような自然言語解析システムを用いて自然言語を処理するためには, さまざまな知識が必要となる. 図 1-1では知識を統語的知識と意味的知識の 2 つに分類している. 統語的知識は言語の統語的な制約に関する知識で, 辞書や文法などを含む. 統語的知識は解析システムの形態素解析, 統語解析の段階で主に使用する. 一方, 意味的知識は言語の意味的な制約に関する知識で, 辞書に登録された単語の意味, 統語構造が表す意味, さらには現実世界に関する常識などを含む. 意味的知識は主として意味解析, 文脈解析で使用する.

このうち, 統語的知識は比較的よく体系化されており, 入力された文の解析結果から正しい解析結果を選別するための判定基準としても使用されている. 特に計算言語学と呼ばれる新たな学問分野の発展は, 文法理論の精密化, および文法理論の計算機上での実現可能性という点に関して大きく貢献した. 計算言語学における計算機の主な役割として次の 2 つが考えられる. ひとつは言語理論

をプログラムとして実現し, 言語理論の検証をおこなうための道具としての役割, もうひとつは大量の言語データを高速に, かつ多様な方法で参照したり, 統計分析をおこなうための道具としての役割である. 本論文の第3章では前者のアプローチにしたがい, 言語理論の成果を計算機上に実現するひとつの手法を提案する. また, 第6章では後者のアプローチにしたがい, 機械可読の辞書から計算機によって自然言語処理に有用な情報を取り出すひとつの手法を提案する.

統語的知識がよく体系化されているという事実は, これまでの自然言語処理の研究の中心が形態素, 統語解析におかれ, これらの解析システムがツールとして整備できるくらいその理論的基礎が固まっていることとも関係が深い. 第2章で述べる自然言語解析システム LangLAB もこのようなツールのひとつである. これに対して意味的知識はその体系化が遅れており, どのような知識を, どのように, どこまで記述すればよいかが十分明らかでないし, 意味的な知識をどのように計算機処理に利用するかについても完全な合意があるわけではない.[86] 本論文では意味的知識として, 概念と概念の間の関係を記述した概念体系をとりあげる. 自然言語処理における概念体系の有用性はすでに多くの研究者によって指摘されている.[6,32,41,43] 本論文の第5章では概念体系における概念間の関係について, 第6章 では個々の概念をどのように設定するかについてそれぞれ論じる.

以上, 図 1-1に沿って古典的な自然言語解析の枠組について述べたが, この枠組が対象としているのは文字によって表現される文字言語である. これに対して音によって表現される音声言語については, 自然言語処理の分野より音声認識の分野で扱われることが多かった. 正しく音声を認識 (理解) するためには, 言語的な知識も必要不可欠であるが, 音声認識おける言語的知識の利用に関しては, これまで十分な研究がされてきたとはいえない. 第4章では, 日本語の語順に関する知識を利用し, 音声理解への応用が可能な言語処理の一手法を提案する.

1.2　論文の構成

自然言語解析の枠組と本論文の構成との関係を該当する章を図 1-1に付記することによって示す. 第2章では, 我々が開発した形態素解析, 統語解析のためのソフトウェアツールである LangLAB について述べる. LangLAB の利用者は文法と辞書を適切な形式で用意すれば, 入力文から統語構造を得るまでの処理を LangLAB によって自動的におこなうことができる. 本論文では LangLAB の特徴および処理の高速化手法を中心に述べる. 第3章と第4章では, 統語的知識に関する問題について論じる. 第3章では, LangLAB で用いる XGS と呼ばれる文法記述形式の限界について述べ, それを解決する新しい文法記述形式 LG^2を提案する. LG^2では XGS に支配制約という新しい概念を導入し, 構成素間の長距離依存関係の記述を容易にしている. 第4章では, 動詞の結合価情報を利用して日

本語の語順を推定するモデルを提案する．このモデルは音声認識や文生成に応用できる．第5章と第6章では, 意味的知識をどのように構築するかについて述べる．第5章では, 概念体系の中に視点という考え方を導入することを提案する．これにより概念間の関係について, より柔軟な議論が可能となる．第6章では, 個々の概念をどのように設定するかという問題を扱う．具体的な応用として機械翻訳を想定し, 機械可読な辞書から概念を自動的に抽出する一手法を提案する．最後に第7章では, 本論文の結論と残された研究課題について述べる．

第 2 章

自然言語解析システム LangLAB

本章では, 今野らの BUP-XG を統語解析の基礎においた自然言語解析システム LangLAB[8,36,99]について, 英語の統語処理を例にとり説明する. 2.1 節では, LangLAB システムの背景を説明する. 2.2 節では, LangLAB で採用している文法記述形式 XGS について説明する. さらに, XGS を Prolog プログラムに変換する際の最適化手法について述べる. 2.3 節では LangLAB で採用している辞書記述形式 DRF とその変換結果である TRIE 構造化辞書について説明する. DRF は DCG の拡張であり, 熟語を記述するための記法が用意されている. また, TRIE 構造化辞書を用いることによって, 辞書容量が節約でき, 辞書の参照も高速になる. 2.4 節では最適化手法の効果を検証するための実験結果を示す. 最適化により, BUP-XG[14]に比べ, インタプリタで約 10 倍, コンパイルコードで約 4 倍の解析の高速化が得られた.

2.1 背 景

Colmeraure の Metamorphosis Grammars[61]によって論理プログラミングの枠組みと自然言語処理との整合性の良さが明確に示されて以来, 文法記述を論理プログラムによっておこなう論理文法の枠組がいくつか提案されている.[54]

Metamorphosis Grammars (MG) は, Chomsky の階層[56]における 0 型文法の一形式であり, MG で記述した文法規則は Prolog プログラムに容易に変換できる. 変換後の Prolog プログラムに入力文を与えて実行すると, トップダウン縦型探索による統語解析をおこなうことができる. この方法の利点は Prolog インタプリタの機能でパーザを代用することが可能になり, 統語解析に必要なパーザの作成が不要になる点である. その後, Pereira らは MG を拡張文脈自由文法に制限した Definite Clause Grammars (DCG) を開発している .[91] MG と同様に DCG で記述した文法規則も Prolog プログラムに容易に変換することがで

き, Prolog インタプリタによってパーザを代用することができる. 両者に共通する重要な特徴として, 文法規則中に任意の Prolog プログラムを挿入することにより, 統語解析と意味解析とを融合した自然言語処理が可能になることが挙げられる. これは, より人間の言語理解に近いモデルを構築できる可能性があるという点で, 認知科学の観点からも重要である. また, この特徴は正しくない統語解析結果を意味的な制約を利用して早期に切り捨てることが可能になるという点で, 計算効率の観点からも重要である .[83]

MG, DCG で記述した文法規則は容易に Prolog プログラムに変換でき, Prolog インタプリタによりトップダウン縦型探索による統語解析が可能である. しかしながら, 一般にトップダウン縦型探索による統語解析は左再帰規則を含む文法を扱うのが困難であるという問題を持っている. もちろん, これは MG や DCG の制限ではなく, Prolog インタプリタの問題である. MG, DCG は文法記述形式であり, 解析アルゴリズムとは独立であるから, 左再帰規則を扱うのが困難であるという制限を MG や DCG に帰すのは適当ではない. Pereira の示した DCG の Prolog プログラムへの変換方法は, DCG で記述した文法を使ってトップダウン縦型探索の解析をおこなう解析システムのひとつの実装方法にすぎない.[91] DCG で記述した文法を使って統語解析をおこなう場合には種々の解析アルゴリズムを採用することできる. たとえば, Nilsson らは DCG で記述した文法から LR 表を作成し, ボトムアップに解析をおこなう AID というシステムを実現している.[87] AID はボトムアップ縦型探索を採用しているため, 左再帰規則は問題なく扱えるが ε 規則を効率よく扱えないという問題がある. 松本らの BUP システムでは DCG 形式で記述した文法規則をほぼそれと 1 対 1 に対応する Prolog プログラムに変換し, この Prolog プログラムといくつかの補助述語を使ってボトムアップ縦型探索の統語解析をおこなう.[80] また, 松本は DCG で記述した文法をボトムアップ横型探索で解析をおこなう Prolog プログラムに変換する手法も提案している. この解析システムは SAX と呼ばれている.[21] AID, BUP, SAX はいずれも DCG で記述した文法を使って解析をおこなう解析システムの実装方法である.

DCG で記述した文法を使って解析システムを実現する研究と同時に DCG の記述能力を強化する研究もおこなわれている. Pereira は DCG を拡張し, 英語の関係節などに現れる左外置現象を記述できる Extraposition Grammars (XG) と呼ばれる記述形式を提案している.[90] 同時に XG で記述した文法規則をトップダウン縦型探索をおこなう Prolog プログラムに変換する方法も与えている. また, 今野らは XG とほぼ同じ記述力を持つ文法記述形式 XGS を提案し, XGS で記述した文法規則を BUP と同様なボトムアップ縦型探索をおこなう Prolog プログラムに変換する手法を示している. このシステムは BUP の拡張であり, BUP-XG と呼ばれている.[14]

LangLAB は BUP-XG を統語解析の基礎におく自然言語解析システムである. LangLAB の構成を図 2-1に示す. 利用者は XGS 形式の文法と DRF 形式の辞書を用意する. 文法, 辞書はそれぞれ専用のトランスレータによって BUP-XG 節, TRIE 構造化辞書に変換された後, Prolog システムに入力される. 解析は Prolog に組込みの機能をそのまま用いておこなう.

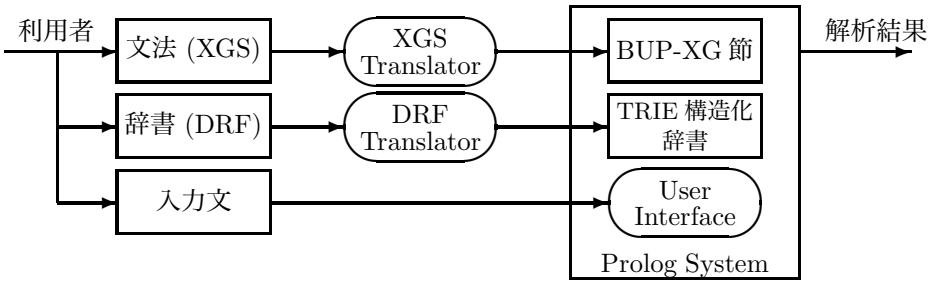


図 **2-1**　**LangLAB** の構成

2.2　文法記述形式 XGS と BUP-XG システム

本節では LangLAB 採用している文法記述形式 XGS と統語解析メカニズム BUP-XG について説明する. BUP-XG の説明の前に, その前身である BUP システムの動作原理について簡単に説明する.

2.2.1　BUP システム

BUP システムでは BUP トランスレータによって DCG 形式で記述された文法規則を BUP 節と呼ばれる DCG 形式の文法規則と補助述語 (後述する link 節, 停止条件節など) に変換する. さらに, この BUP 節は多くの Prolog システムに組み込まれている DCG トランスレータによって最終的に Prolog プログラムに変換される.

```
s → np, vp.        (2-1)
np → pron.         (2-2)
pron → [you].      (2-3)
vp → [walk].       (2-4)
```

(a) DCG 規則

```
np(G, S0, S2) :- link(np, G),          (2-1)'
    goal(vp, S0, S1),
    s(G, S1, S2).
pron(G, S0, S1) :- np(G, S0, S1).      (2-2)'
dict(pron,[you|S], S).                 (2-3)'
dict(vp,[walk|S], S).                  (2-4)'
```

(b) Prolog プログラム

図 **2-2**　**DCG** 規則の変換例

図 2-2に DCG で記述した文法規則と Prolog プログラムへの変換例を示す. ただし, ここでは link 節, 停止条件節は省略した. 図 2-2 (b) 中の各述語の引数のうち, S で始まる変数名を持つ変数の組は入力文の差分リストに対応する. この Prolog プログラムに入力文を与えるとボトムアップ縦型探索の解析がおこなわれる.

```
goal(G, S0, S1) :-                                          (2-5)
   ( wf_goal(G, S0, _) ; fail_goal(G, S0), !, fail ), !,
   wf_goal(G, S0, S1).
goal(G, S0, S2) :-                                          (2-6)
   dict(C, S0, S1),
   link(C, G),
   P =.. [C, G, S1, S2],
   call(P),
   assertz(wf_goal(P)).
goal(G, S0, S1) :-                                          (2-7)
   assertz(fail_goal(G, S0)), !, fail.
```

図 **2-3**　**goal** 節の定義

BUP システムでボトムアップに統語解析をおこなうためには述語 goal が必要である. goal 節の定義を図 2-3に示す. 図 2-2の文法規則を用いて “you walk.” という文の統語解析手順を説明する. “you walk.” の解析は次の goal 節の呼び出しにより開始する.

```
?- goal(s, [you, walk], [ ]).
```

これは, [you,walk] の後方から [ ] を差し引いた残りの部分, すなわち [you,walk] から s を根とする統語木ができるかどうかを調べることを意味している.

goal 節の呼び出しにより (2-5)を呼び出す. (2-5)では, まず成功結果と失敗結果とがそれぞれ述語 wf_goal と fail_goal として登録されているかどうかを調べ, 無駄な再計算を避ける. この例の場合には (2-5)の実行は失敗するので次の (2-6) を選ぶ. (2-6)のボディでは述語 dict(C, [you, walk], S1) を呼び出し, 入力文の単語リストの先頭から辞書引きをおこなう. これは図 2-2の (2-3)’ とマッチするので, 文法カテゴリ C に pron が, S1 には [walk] が束縛される.

(2-6)のボディの 2 行目では述語 link(pron, s) を呼び出すが, これは統語解析が進み統語木が下から上向きにに成長したとき, pron から s に到達可能かどうかを調べる述語である. link 節はあらかじめ BUP トランスレータが文法規則から計算する. 今の場合 link(pron, s) が成功するものとしよう. 次に述語=.. により述語 pron を構成し, これを呼び出す. すなわち,

P =.. [pron, s, [walk],[ ]], call(P).

ここで call(P) は実際には, call(pron(s, [walk], [ ])) となっている.

pron(s, [walk], [ ]) の実行により今度は (2-2)' を呼び出し, そのボディの np(s, [walk], [ ]) を実行する. これは np を根とする統語木が完成したことを意味している. np(s, [walk], [ ]) の呼び出しにより, 次は (2-1)' を呼び出す. (2-1)' のボディでは link(s, s) を調べるが, 到達可能性は反射律を満たすのでこれは必ず成功する. link 節での検査の後, 次の goal(vp, [walk], [ ]) を呼び出す. これは np までの処理が終り, 次に処理すべき単語の系列が 'walk' で始まり, そのゴールが vp であることを予測していることになる.

以下同様にしてトップダウンの予測を利用したボットムアップの統語処理が進み, 最終的には停止条件節により goal 節の実行が成功する. 停止条件節については文献 [80] を参照されたい. (2-6)のボディの最後では, 現在実行中のゴールが成功したので成功結果を述語 wf_goal として登録する. これにより, 同じ計算がバックトラックにより繰り返される時には (2-5)を用いて直ちに成功結果を取り出すことができる. (2-7)は失敗結果を登録するための節である.

2.2.2　BUP-XG システム

英語の関係代名詞節の埋め込み文は平叙文中の名詞句がひとつ欠落した構造を持っているが, これは先行詞が関係代名詞節の左方に移動してできたと考えることができる. このような名詞句の移動を言語学では左外置と呼んでいる. この場合, 移動した跡にはギャップを残すと考え, このギャップをシステムに発見させることを前提にして文法規則を記述すると, 文法規則の数や文法カテゴリの数が減り, 文法の見通しが良くなるとともに, 統語処理の速度も向上することが報告されている.[14]

トップダウン統語解析システムである ATNG[100,102] や Pereira の XG システム[90] には, このようなギャップ発見機構が組み込まれている. トップダウンに解析をおこなう場合には次に処理すべき部分がどの非終端記号にまとめられるかが予測できる. したがって, 特定の文法カテゴリ (たとえば np) が予測された時にだけギャップの存在を仮定することができるので効率良くギャップを発見することができる.

一方, 純粋にボトムアップな統語解析システムの場合には, このような予測が不可能なので, 入力文のすべての単語間でギャップを仮定して解析を進めなければならなくなり, 効率が悪い. BUP システムはボトムアップを基本とし, トップダウンの予測も利用している. 今野はこの点に着目して, BUP システムにギャップ発見機構を組み入れた BUP-XG システムを開発している .[14]

BUP-XG システムではギャップを発見するために, Pereira が XG システム[90]

で導入した X リスト (extraposition list) を用いる．このため X リストを扱うように goal 節も変更しなければならないが,[14] ここでは BUP-XG の実装の詳細については立ち入らない．LangLAB の利用者の観点からは左外置現象が容易に記述できるように DCG を拡張した文法記述形式 XGS を理解するだけで十分である．

XGS による文法記述例を以下に示す．この例で規則 (2-10)の “../”(スラッシュと呼ぶ) が XGS で導入された記法である．XGS では relC../np と記述することによって, カテゴリ relC の下にギャップを直接支配するカテゴリ np がひとつ存在することを表す．この記法は GPSG[70] の Slash category の考え方を参考にしたもので, XGS でもスラッシュの後のカテゴリをスラッシュカテゴリと呼んでいる．

```
s → np, vp.                          (2-8)
np → pron.                           (2-9)
np → det, noun, relC../np.           (2-10)
vp → vt, np.                         (2-11)
relC → relpro, s.                    (2-12)
```

規則 (2-10)は, det, noun, relC の 3 つのカテゴリから名詞句 np ができることを表しているが, relC の下のギャップは det と noun からなる名詞句が移動してできたものと考えることができる．したがって, 統語解析でギャップが発見された場合には, 図 2-4に示すように埋め込み文中のギャップと移動した先行詞 “man” とを対応づけることができる．文法記述者は XGS で文法を記述すれば, LangLAB に組み込まれた BUP-XG システムによって自動的に先行詞とギャップとの対応をつけることができる．

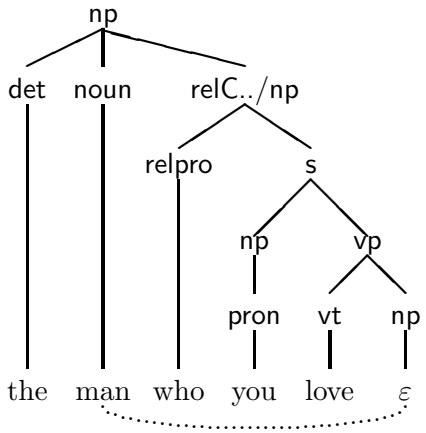


図 **2-4**　先行詞と痕跡の対応

変形文法には複合名詞句制約[29]と呼ばれる制約があるが, XGS ではこの制約を文法中に自然な形で記述できるようにカテゴリを “⟨” と “⟩” で囲む記法が用意されている. これを Pereira にならって, それぞれ open, close と呼ぶ.[90]

a → b, c, ⟨d⟩.

この規則はカテゴリ b, c, d から a ができることを示しているが, d を open, close で囲むことにより, 語句の左外置に関して, b, c の下から a の外に語句が移動してもよいが, d の下からは a の外に移動してはならないことを表している. 前述の文法が複合名詞句制約に違反した文を受理しないように規則 (2-10)に open, close を付け加えると次のようになる.

np → det, noun, ⟨relC../np⟩.

これによって relC 中の (スラッシュカテゴリ np に支配される) ギャップは det と noun とからなる名詞句以外とは対応付けができなくなる. そのほかに XGS は 2 重矢印 (⇒) の記法, X リストを明示的に記述する記法なども用意している.[14]

2.2.3　XGS トランスレータ

XGS で記述した文法は XGS トランスレータによって BUP-XG 節に変換される. この時, BUP と同様に link 節, 停止条件節も生成される. LangLAB では BUP-XG システムに比べ最適化された BUP-XG 節を生成するようにトランスレータに改良が加えられている. 生成コードの最適化に加え, 利用者の使いやすさを考慮して統語木の情報を自動付加できるように機能を拡張している. 変換速度は規則数約 200 の文法に対して約 3 秒である. 以下, これらの改良について説明する.

link 節の変更

文法規則の数が多くなるにつれて, トランスレータにより生成される link 節の数は著しく増大する. たとえば, 我々が使用している英語の文法[10]の規則数はおよそ 200 であるが, トランスレータにより生成される link 節の数は約 700 に達している. したがって, link 節の検索時間の短縮は解析速度の短縮につながる.

link 節の呼び出しが起こるのは BUP-XG 節のボディの先頭と goal 節の中である. いずれの場合も述語 link の 2 つの引数はアトムであることが保証されているのでカテゴリ a から b への到達可能性を表す link 節 “link(a, b).” を “a(b) :- !.” という形式に変更することができる. これによって述語名をキーとしたハッシュ検索が可能になるので到達可能性の検査のための時間が大幅に減少する. LangLAB の XGS トランスレータはこのような形式で link 情報を生成するように改良されている.

差分リストのインデックス化

2.2 節で述べたように, 入力文は差分リストとして表現している. また, 統語解析過程で生じる中間的な成功結果と失敗結果は, それぞれ述語 wf_goal, fail_goal としてデータベースに登録される. 述語 wf_goal の最後の 2 引数は解析に成功した部分文の差分リストなので, 一般に入力文の長さが長くなると wf_goal は長いリストを引数として持つことになる. また, 文法規則の数が増えると, 登録される wf_goal, fail_goal の数が増加するので大きな記憶容量を必要とする. この記憶容量は差分リストをインデックス化することで節約できる. また計算結果を再利用する場合にも検索時間が短縮されるため, 解析速度が改善できる.[13]

具体的には, たとえば “you walk.” という文が入力された段階で,

```
'**index'(0,[ ]).
'**index'(1,[walk]).
'**index'(2,[you,walk]).
```

をデータベースに登録する. 辞書引きはインデックスの対から述語**index を呼び出すことにより差分リストを復元してからおこなう.

wf_goal, fail_goal の分割

一般に解析する文が長くなると解析の途中で生成される wf_goal, fail_goal の数も増加し, その検索に時間を要する. wf_goal は引数として解析に成功した (fail_goal では失敗した) 入力文の部分単語列のインデックスとその解析結果を持つ. goal 節の中では, これから解析しようとする部分単語列のインデックスをキーとして, まず wf_goal, fail_goal を検索する. LangLAB では wf_goal, fail_goal という述語ではなく, 解析に成功した部分単語列のインデックスを述語名として計算結果を登録する. これによってハッシュを利用した検索をおこない, 検索時間を短縮している.

統語木情報の自動付加

入力された文を解析するとき, 解析結果として統語木を得たい場合が多い. BUP や BUP-XG システムでは利用者が文法を記述するときに各カテゴリの引数として統語木の情報を付加していた. しかしながら, 統語木の情報は機械的に付加することが可能であり, 文法記述時には本質的なものではないので, トランスレータが自動的に付加することが望ましい. LangLAB の文法トランスレータは特に指示しない限り変換の際に統語木の情報を自動的に付加する. この統語木情報を変換時に自動付加する機能は McCord の MLG (Modular Logic

Grammars) システム[81]と同様なものである．しかし, MLG では統語木に反映させるカテゴリを指定しなければならないが, LangLAB のトランスレータではすべての文法カテゴリを自動的に統語木に反映させる点が異なっている．

2.3　辞書記述形式 DRF と TRIE 構造化辞書

LangLAB では辞書と文法を分けて記述する．これにより辞書と文法の保守が容易になる．また, 辞書だけを 2 次記憶に置く場合にも都合がよい．LangLAB では辞書項目として単語だけではなく, 文法カテゴリを含む熟語も登録することができる．文法カテゴリを含む熟語は本来文法規則として記述すべきものだが, 熟語をすべて文法規則に含めると規則数が非常に多くなるという問題がある．LangLAB では, 熟語を辞書項目として扱い, この問題を解決している．

本節では LangLAB で採用している辞書記述形式 DRF と DRF で記述した辞書をトランスレータで変換した結果である TRIE 構造化辞書について説明する．TRIE 構造化辞書を用いることにより, 辞書に必要な記憶容量の節約, 辞書引きの高速化, 柔軟な熟語処理が可能となる.[22]

2.3.1　辞書記述形式 DRF

DRF は基本的には DCG と同じであるが, 複数の語からなる辞書項目 (熟語など) に関しては形態素変化の情報が必要となるので, DCG を拡張した記法で辞書を記述する必要がある．

1 単語からなる辞書項目

1 単語からなる辞書項目は通常の DCG でも記述できる．たとえば,

$$\mathsf{n(computer)} \rightarrow \mathsf{[computer]}.$$
$$\mathsf{v(get)} \rightarrow \mathsf{[get]}.$$

辞書項目の引数は利用者に解放されている．これと等価な DRF による記述は以下のようになる．

$$\mathsf{n(computer)\langle *n\rangle} \rightarrow \mathsf{[computer]}.$$
$$\mathsf{v(get)\langle *v\rangle} \rightarrow \mathsf{[get]}.$$

1 単語からなる辞書項目に関しては, DCG で記述しても DRF で記述しても DRF トランスレータはまったく同じ変換をおこなう．

複数の単語からなる辞書項目

熟語などのように複数の語からなる辞書項目については必ず DRF によって記述する必要がある. たとえば,

```
n(N)⟨n, ∗n⟩ → [computer, system].
v(V)⟨∗v, _⟩ → [get, on].
np(Np2)⟨_, _, _, _, _, _⟩ →
    [not, only], np(Np1), [but, also], np(Np2), { check(Np1, Np2) }.
```

DCG と異なるのは左辺のカテゴリ名の後に “⟨ ⟩” で囲まれた情報が付加されている点である. “⟨ ⟩” 内の各要素は, それぞれ右辺の語またはカテゴリに対応し, 右辺の各語がどのようなタイプの形態素変化をするかを指定している. “_” は対応する語が形態素変化をしないことを表し, “∗” は辞書項目全体の主辞を表す. “∗” のついた要素の形態素情報は, その辞書項目の形態素情報として採用される.

排他的文法カテゴリ

英語の熟語の中には動詞が前置詞/副詞をともなって, 意味的, 統語的にひとつの動詞のようにふるまうものがある. このような熟語は, その音声的/統語的性質によって群動詞と句動詞に分類される. 両者の違いを表 2-1に示す.[92] ただし, 単語の前の “’” は強勢を表す.

表 **2-1**　英語の群動詞, 句動詞の例

群動詞 *call on* = “visit”	句動詞 *call up* = “summon”
They ’call on the man	They call ’up the man
They call ’early on the man	*They call early up the man
The man on whom they call	*The man up whom they call
*They call the man on	They call the man ’up
*They call him on	They call him ’up

この例からわかるように句動詞では動詞と副詞/前置詞の間に目的語が挿入されることがある. この現象を容易に記述するために, DRF では排他的文法カテゴリという考え方を採り入れている. たとえば, “call up” の例を DRF で記述すると次のようになる.

phrasal-verb(summon)⟨_,_,_,_⟩ → [call], ^np, [up], ^np.

ここで, “^” を付加した文法カテゴリが排他的文法カテゴリである. 排他的文法カテゴリでは, “^” の付いたカテゴリのうちいずれかひとつが有効となる. この例で, “call up” の目的語となる np は “call” の直後, または “up” の直後の位置に来ることを表現している.

ここで注意しなければならないのは, このような記述をすると語彙カテゴリ phrasal-verb は X バー理論[96]でいうバーレベルが 1 つ上がり, もはや動詞相当のカテゴリではなく, 動詞句相当のカテゴリとなる点である. たとえば, 統語解析と並行して意味解析をおこなう場合に, 動詞と目的語の意味から動詞句の意味を計算する処理をどこに記述するかが問題となる. 方法としては次の 2 つが考えられる.

(1) 意味処理のプログラムを辞書中に記述する.

(2) なんらかの方法で動詞とその目的語である名詞句の意味情報を句動詞の情報として返し, 意味処理は文法中に記述する.

辞書項目中にはできるだけその語固有の情報だけを記述する方が辞書の保守が容易になるので, 保守という観点からは (2) の方法が望ましいが, そのためには自然な方法で情報の受け渡しを記述できる必要がある. この問題については第3章でもう一度触れる.

2.3.2　TRIE 構造化辞書

DRF 形式で記述した辞書は DRF トランスレータによって TRIE 構造化辞書に変換される. TRIE 構造化辞書とは, 節点として辞書項目の右辺の要素を持つ木構造をした辞書で, TRIE の名前は reTRIEval に由来する.[55] たとえば, 次の辞書は図 2-5の構造に変換される.

$\mathsf{cat}_1(\mathsf{C1})\langle \_ \rangle \rightarrow [\mathsf{word}_1]$.
$\mathsf{cat}_2(\mathsf{C2})\langle \_, \_ \rangle \rightarrow [\mathsf{word}_1, \mathsf{word}_2]$.
$\mathsf{cat}_3(\mathsf{C3})\langle \_, \_, \_ \rangle \rightarrow [\mathsf{word}_1], \mathsf{cat}_4(\mathsf{C4}), [\mathsf{word}_3], \{ \mathsf{check}(\mathsf{C4}) \}$.
$\mathsf{cat}_5(\mathsf{C5})\langle \_, \_, \_ \rangle \rightarrow [\mathsf{word}_1], \hat{}\mathsf{cat}_4(\mathsf{C4}), [\mathsf{wrod}_4], \hat{}\mathsf{cat}_4(\mathsf{C4})$.
$\mathsf{cat}_6(\mathsf{C6})\langle \_, \_, \_ \rangle \rightarrow [\mathsf{word}_1, \mathsf{word}_2, \mathsf{word}_6]$.

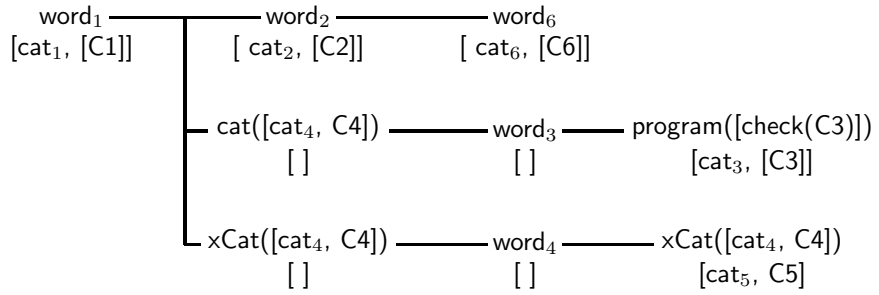


図 **2-5**　**TRIE** 構造化辞書の例

図 2-5からわかるように, TRIE 構造化辞書の節点には以下の 4 種類がある.

単語	word_1, word_2, word_3, word_4, word_6
文法カテゴリ	$\mathsf{cat}([\mathsf{cat}_4, \mathsf{C4}])$
排他的文法カテゴリ	$\mathsf{xCat}([\mathsf{cat}_4, \mathsf{C4}])$
補強項	$\mathsf{program}([\mathsf{check}(\mathsf{C3})])$

これらの各節点には属性が付加されている. 節点の属性は, 木の根からその節点までの経路中の単語列が表す辞書項目に対する文法カテゴリとその情報 (引数) を含んでいる. 対応する辞書項目がない場合は属性は空リストとなる. 辞書をこのような構造にすると, 同一の単語を先頭に持つ複数の単語列はひとつの節にまとめられるので, 記憶容量の節約になるとともに, 辞書参照時にも節レベルでのバックトラックが無くなり, 辞書の参照が高速になる.

2.3.3　辞書参照アルゴリズム

辞書の参照アルゴリズムは基本的に最長一致法を採用している. その過程で単語については形態素処理も並行しておこなう. アルゴリズムの概要は以下のとおりである.

1. 単語の節点では,

   (a) 解析中の単語との照合をおこなう.

   (b) 解析中の単語が節点の単語を形態素変化させたものかどうか調べる.

   以上のいずれかが成功したら節点を葉の方向に 1 つ進めて再帰的に本アルゴリズムを適用する.

2. 文法カテゴリの節点では, その文法カテゴリをゴールとする goal 節を呼ぶ. goal 節が成功したら節点を 1 つ進めて再帰的に本アルゴリズムを適用する.

3. 排他的文法カテゴリの節点では,

   (a) 根から現在までに排他的文法カテゴリが現れ, 入力単語列を消費していたら何もしない.

   (b) はじめて排他的文法カテゴリが現れる場合は, その存在を表すフラグを立てると同時にそのカテゴリをゴールとする goal 節を呼び出す. goal 節の成否を表すフラグを設定する.

   節点を葉の方向に 1 つ進めて再帰的に本アルゴリズムを適用する.

4. 補強項の節点では, その補強項を実行する. 補強項が成功したら節点を 1 つ進めて再帰的に本アルゴリズムを適用する.

5. 以上のいずれでもなければ, 1 つ前の節点の属性を結果として返す. もし, その節点が属性を持たなければ失敗する.

実際の辞書参照プログラムには goal 節と同様に成功した辞書参照を記録し再利用する機構が組み込まれているが, ここでは省略した.

2.3.4　熟語処理の例

BUP システム[80]では, 文法カテゴリや Prolog プログラムを含む熟語を, その先頭の単語と残りの文法カテゴリまたはプログラムの部分とに分割し, 前者を辞書項目, 後者を文法規則として扱っていた. しかし, 本来これらは分割して扱うべきではない. LangLAB では, このような熟語はすべて辞書項目として扱うことができる. ここでは熟語の辞書項目の記述例をいくつか説明する.

不規則変化動詞を含む熟語

次の例は動詞 “get” およびそれを含む熟語の記述例である.

```
v([sem:get]) → [get].
v(ref(get,[vf:ed])) → [got].
v(ref(get,[vf:en])) → [gotten].
v([sem:get-up]) → [get, up].
v([sem:get-on]) → [get, on].
```

ここで, vf, sem はそれぞれ動詞の変化形, 意味を表す素性である. この例で辞書項目 “got” と “gotten” は ref という構造体を引数として持っているが, これは構造体 ref の第 1 引数 (この例では “get”) へのポインタを意味している. 不規則変化動詞の原形と変化形のように, 一部の情報だけが異なる辞書項目を記述する

ときにはこのように記述する. 構造体 ref の第 2 引数のリストはこの辞書項目 (“got” または “gotten”) とポインタの先の辞書項目 (“get”) との差分情報で, 動詞の変化形がそれぞれ過去形, 過去分詞形であることを示している. このような機構を設けることによって多くの熟語をつくる不規則動詞について, その変化形に対する熟語をすべて登録する必要がなくなる. 規則変化動詞は形態素解析プログラムが処理するので辞書項目として登録する必要がない.

文法カテゴリを含む熟語

熟語の中には “not only ～ but also ～” のように熟語の一部に文法カテゴリを含むものもある. このような熟語の記述例を以下に示す.

```
adj → [not, only], adj, [but, also], adj.
adv → [not, only], adv, [but, also], adv.
np → [not, only], np, [but, also], np.
```

辞書参照アルゴリズムで説明したように, 辞書項目中の文法カテゴリは辞書参照プログラムの中から goal 節を呼び出すことによって処理する.

2.4 実験

この節では 2.2 節で述べた BUP-XG 節の最適化の効果を検証するためにおこなった実験の結果について考察する. 実験環境は以下の通りである.

使用計算機:	Sun3/260 ワークステーション
Prolog:	Quintus-Prolog Release 1.6
文法:	XGS 形式で 163 規則
例文:	章末の例文参照

実験は各例文について, すべての統語解析結果が得られるまでの時間 (単位はミリ秒) を以下の 5 つの場合に分けて測定した.

(1) 最適化する前の BUP-XG 節

(2) (1)+link 節の変更

(3) (2)+差分リストのインデックス化

(4) (3)+wf_goal,fail_goal の分割

(5) (4) に形態素解析を含めたもの

このうち (1) から (4) までは, 形態素処理を含まない統語処理だけに要した時間である.

表 2-2　インタプリタ・コードによる解析速度

例文	統語木	(1) 最適化前	(4) 最適化後	(5) 形態素解析	(1)/(4)	(5)/(4)
1	14	80415	8552	13565	9.40	1.59
2	4	18868	2700	4216	6.99	1.56
3	3	46700	4983	5999	9.37	1.20
4	1	30900	3600	5433	8.58	1.51
5	3	39634	4050	5767	9.79	1.42
6	4	95933	9550	17184	10.05	1.80
7	9	323167	26183	48146	12.34	1.84
8	2	87550	9349	12017	9.36	1.29
9	4	180300	15816	19367	11.40	1.22
10	1	116284	12083	25283	9.62	2.09
				平均	9.69	1.55

表 2-3　コンパイル・コードによる解析速度

例文	統語木	(1) 最適化前	(4) 最適化後	(5) 形態素解析	(1)/(4)	(5)/(4)
1	14	20485	4134	6500	4.96	1.57
2	4	2467	1299	1550	1.90	1.19
3	3	4783	2284	2250	2.09	0.99
4	1	2884	1566	2217	1.84	1.42
5	3	4383	1917	2283	2.29	1.19
6	4	18768	4500	6949	4.17	1.54
7	9	127400	14000	26600	9.10	1.90
8	2	13450	4450	5050	3.02	1.13
9	4	59468	8216	8682	7.24	1.06
10	1	23650	5801	10900	4.08	1.88
				平均	4.07	1.39

表 2-2にインタプリタによる結果, 表 2-3にコンパイルした場合の結果を示す. 表には, (1) 最適化する前, (4) 最適化した後, (5) 最適化して形態素処理も含む場合, および, (1) と (4) の比, (4) と (5) の比をあげた. 2 つの表より, 最適化前の BUP-XG 節に比べ LangLAB の最適化 BUP-XG 節では, インタプリタで約 10 倍, コンパイルすると約 4 倍の高速化が得られていることがわかる. また, 形態素解析をおこなうと処理時間が約 50%増加する. 表には現れていないが最も効果が大きいのはインタプリタにおける link 節の変更であった. 生成される link

節の数 (この文法では約 700) を考えれば納得のいく結果である. コンパイルするとインタプリタほどは最適化の効果が顕著でなくなるが, システムのデバッグ作業がインタプリタでおこなわれることが多いことを考えると, この最適化は意義がある. この最適化で LangLAB の統語処理速度は実用上十分なレベルに達したといえよう.

他のシステムとの比較としては, 松本の SAX システム[21] と LangLAB との比較が奥西らによって報告されている.[89] これによるとインタプリタでは LangLAB の方が SAX に比べ 6 倍から 10 倍速いが, コンパイルすると SAX の方が逆に 6 倍から 16 倍速くなることが報告されている. 同一の文法でも変換すると SAX の方が多くの節を生成するが, コンパイルによりハッシュが効く要素が多いため, このような結果になるものと考えられる.

2.5 要約

自然言語解析システム LangLAB について, その統語解析の道具立てを中心に説明し, 解析速度の高速化手法およびそれを検証するための実験結果を示した. 統語解析速度に関しては, 我々は満足できるレベルに達したと考えている. LangLAB の主な特徴は次のとおりである.

- 左外置現象を自然に記述できるように DCG を拡張した文法記述形式 XGS
- 熟語が容易に記述できる辞書記述形式 DRF
- トップダウン予測を含むボトムアップ深さ優先解析
- 利用者によるカスタマイズが容易なモジュール構成[8]

付録 : 実験で使用した例文

1. She was given more difficult books by her uncle.
2. Be careful not to break the vase when you put it down.
3. The structural relations are holding among constituents.
4. It is not tied to a particular domain of applications.
5. Diagram analyzes all of the basic kinds of phrases and sentences.
6. This paper presents an explanatory overview of a large and complex grammar that is used in a sentence.
7. The annotations provide important information for other parts of the system that interprets the expression in the context of a dialogue.
8. For every expression it analyzes, diagram provides an annotated description of the structural relations holding among its constituents.
9. Procedures can also assign scores to an analysis, rating some applications of a rule as probable.
10. These properties provide important pieces of information for the other part of the system that interpret the phrase when it is used in a discourse.

第 3 章

統語的知識とその記述

本章では前章で説明した LangLAB の文法記述形式 XGS の問題点を指摘し, それを解決するための新しい文法記述形式 LG^2 (Logic Grammar for handling Gap) を提案する. 3.1 節では過去の研究を概観する. 3.2 節では, XGS のスラッシュ記法とその記述力の限界について考察し, LG^2で導入した支配制約について説明する. 3.3 節では, XGS と対比させながら, いくつかの英語の移動現象が支配制約を用いて自然に記述できることを示す. 3.4 節では, Pereira が提案した DCG から Prolog プログラムへの変換手法を拡張することによって, 支配制約が容易に実現できることを示す.

3.1 背 景

英語における関係節や Wh 疑問文に現れる構成素の移動現象を論理文法上で扱う枠組が Pereira の Extraposition Grammars (XG)[90]や Dahl の Discontinuous Grammars (DG)[66]をはじめとして, いくつか提案されている. これらはいずれも構成素が移動した後には仮想的なカテゴリがギャップとして残るという考え方に基づき, ギャップの存在を文法中に明示的に記述する. このような文法記述では, 一般に記述すべき文法規則の数が少なくてすむという利点がある. 本節では構成素の移動を扱うことを考慮した論理文法の代表的な研究を紹介し, それぞれの特徴について述べる.

3.1.1 Extraposition Grammars (XG)

XG は英語などの関係節に現れる左外置現象を扱うために Pereira が提案した論理文法の枠組である.[90] XG は Pereira が以前に提案した DCG[91]の拡張となっている. XG では DCG に加え, 次の形式の規則を許している.

$s_1 \ldots s_2 \cdots s_{k-1} \ldots s_k \rightarrow r.$

ここで, s_iは任意の終端, 非終端記号の並びを表す. ただし, s_1だけは非終端記号でなければならない. "..." は XG で導入された記法で, 任意の終端, 非終端記号の並びを表す (省略の "..." と混同しないこと). この記法により終端, 非終端記号の並びの不連続性を表現する. また, 右辺の r は DCG と同様に任意の終端記号, 非終端記号, 補強項の並びである. この規則は x_iを任意の終端記号, 非終端記号の並びとして, 記号列 $s_1x_1s_2x_2 \cdots s_{k-1}x_{k-1}s_k$を $rx_1x_2 \cdots x_k$に書き換えてよいことを表している.

文法 3-1は XG で英語の関係節に関わる文法の一部を記述したものである.

文法 **3-1**

s → np, vp. (3-1)
np → det, noun, relative. (3-2)
np → trace. (3-3)
vp → verb, np. (3-4)
vp → verb. (3-5)
relative → [ ]. (3-6)
relative → rel-marker, s. (3-7)
rel-marker... trace → rel-pronoun. (3-8)

規則 (3-8)によって, たとえば, "the mouse which the cat chased squeaks" は "the mouse rel-marker the cat chased trace squeaks" に書き換えられて解析される.

DCG を用いると, 同様な文法を文法 3-2のように記述することができる.

文法 **3-2**

s → np, vp. (3-1)
np → det, noun, relative. (3-2)
np → trace. (3-3)
vp → verb, np. (3-4)
vp → verb. (3-5)
relative → [ ]. (3-6)
relative → rel-pronoun, s. (3-9)
trace → [ ]. (3-10)

しかし, 文法 3-2には関係節の外でも規則 (3-3)によって名詞句が trace に展開されてしまうという問題がある. これを防ぐためには名詞句を trace に展開しても

よいかどうかを表す情報を引数として各カテゴリに付加する必要がある. XG は文法記述者をこのような情報の管理から解放しているという点で重要な意味を持つ.

3.1.2 Discontinuous Grammars (DG)

DG は Dahl によって提案された論理文法で, XG を一般化したものである.[64,66] DG の規則は一般に次のような形式をしている.

$$\alpha_0, \mathsf{skip(X}_1\mathsf{)}, \alpha_1, \cdots, \mathsf{skip(X}_n\mathsf{)}, \alpha_n \rightarrow \beta_0, \mathsf{skip(X'}_1\mathsf{)}, \beta_1, \cdots, \mathsf{skip(X'}_m\mathsf{)}, \beta_m$$

ここで, α_i, β_iは任意の終端または非終端記号列, $\mathsf{skip(X}_i\mathsf{)}$, $\mathsf{skip(X'}_i\mathsf{)}$ はスキップ記号と呼ばれ, 任意の終端, 非終端記号列に対応する. DG の $\mathsf{skip(X}_i\mathsf{)}$ は XG の "..." に対応する. XG の規則 (3-8)を DG で記述すると規則 (3-11)になる.

$$\mathsf{rel\text{-}marker, skip(X), trace} \rightarrow \mathsf{rel\text{-}pronoun, skip(X).} \qquad (3\text{-}11)$$

XG では "..." で表されるスキップ列が常に右辺の最右位置に置かれるのに対し, DG ではスキップ列を任意の位置に再配置できる. したがって, DG の記述力は XG の記述力を完全に含んでいる. Dahl は XG では記述できない例として次のような例文をあげている.[64]

the man with whose uncle john left
the man [john left with [the] uncle [of the man]]

この例文は下の構造から, "of the man" を "the" の左に外置し, "whose" で置き換えた後, "with whose" を "john left" の左に外置することによって導出できる. DG ではこの書き換え規則を規則 (3-12)のように記述できる.

$$\mathsf{np(X), skip(Y), prep, det, skip(Z), prep(of), np(X)} \rightarrow \\ \mathsf{np(X), prep, [whose], skip(Z), skip(Y).} \qquad (3\text{-}12)$$

左辺の skip(Y), skip(Z) の順序が右辺では入れ替わっている点に注意すると, この規則は XG の枠を越えていることがわかる.

3.1.3 Modular Structure Grammars (MSG)

MSG は XG に Modifier Structure と呼ばれる一種の論理式を構成する規則を付加した論理文法で, Dahl と McCord によって提案された.[65] 解析した文の論理式を構成する規則を文法に付加する考え方は基本的に McCord の以前の研究[82]を踏襲している. MSG で強調されているのは, "and", "but", "or" などの

等位接続詞によって構成される等位構造を扱うための機構を備えている点である. 文法記述者は等位構造に関する規則を記述する必要がなく, 等位構造はパーザによって自動的に処理される. これは Woods が ATN で用いた SYSCONJ[102] の考え方を論理文法上で実現したものと考えることができる. ただし, MSG では SYSCONJ では扱っていない埋め込み文の等位構造も扱えるし, 等位構造のスコープの計算もおこなうなど SYSCONJ を越える部分もあるので, 単に SYSCONJ の論理文法版というわけではない.

MSG で記述した文法は XG に変換され, さらにホーン節に変換される. XG や DG と異なり, MSG で記述された文法には等位構造を扱う特別なインタプリタが必要となる. したがって, MSG の枠組は純粋な文法記述の枠組ではなく, パーザも含めた論理文法の枠組であるといえる.

3.1.4 Restricted Logic Grammars (RLG)

RLG は XG にスイッチ規則と右外置のための記法を導入した論理文法で, Stabler によって提案された.[98] また, RLG は構成素の移動に関する制約として Chomsky の GB 理論[60,96]を基礎としている.

RLG のスイッチ規則は局所的な入力文字列の入れ換えを記述するための道具である. たとえば, 英語の疑問文形成はスイッチ規則を用いて次のように記述できる.

$$\mathsf{s} \rightarrow \mathsf{switch(aux\text{-}verb,\ np),\ vp.} \qquad \text{(3-13)}$$

規則 (3-13)は入力文字列の先頭の単語がカテゴリ aux-verb に属するなら, その単語を np の解析が終了した直後に挿入することを表している. 規則 (3-13)に対応する規則を XG で記述すると規則 (3-14), (3-15)になる.

$$\mathsf{s} \rightarrow \mathsf{fronted\text{-}verb,\ s.} \qquad \text{(3-14)}$$
$$\mathsf{fronted\text{-}verb...aux\text{-}verb(X)} \rightarrow \mathsf{aux\text{-}verb(X).} \qquad \text{(3-15)}$$

しかしながら, 規則 (3-14), (3-15)では, 補助動詞が主語の直後から文の先頭に外置されるという制約を満たしていないために次のように非文も受理してしまう.

Has he ε been saying that he has been succeeding?
*Has he has been saying that he ε been succeeding?

規則 (3-13)では最初の名詞句 (主語) の解析が終了した時点で補助動詞を挿入するので XG のような不都合はおこらない.

関係節に関する規則を RLG で記述すると規則 (3-16)のようになる.

relative <<< trace → rel-pronoun.　(3-16)

この規則は XG で記述した文法 3-1の規則 (3-7), (3-8)に対応する．XG と異なり, RLG では rel-marker などの言語学的に根拠のないカテゴリを導入する必要がない．同様に記号 “>>>” を用いて右外置も記述できる．

RLG のもう 1 つの大きな特徴は, RLG で記述した文法規則を Prolog プログラムに変換する際に, GB 理論の制約のうち c-統御と下接の原則を検査するためのプログラムが自動的に埋め込まれる点である．これによって非文法的な構成素の移動を検査し, 非文の受理を防いでいる．XG にも Ross の複合名詞句制約[29]を記述するための記法が用意されているが, 実際の記述は文法記述者の判断にまかされている．RLG の移動に関する制約は, 複合名詞句制約を含むより一般的なもので, その制約を自動的に付加するという考え方は文法記述者の負担を減らすという意味で重要である．

3.1.5　Static Discontinuous Grammars (SDG)

DG は任意の書き換え規則を許しているため, 文法の生成能力が強力である反面, 文法記述の適切なガイドラインを与えないと文法が記述しにくいという欠点がある．また, スキップ記号は任意の記号列を読み飛ばせるので効率の良い実現が難しい.[79] このような欠点を補うために, Dahl は DG の記述に制限を課した SDG という論理文法を提案している.[62,64] SDG では DG で許されている右辺のスキップ記号の再配置が禁止されている．すなわち, 左辺に現れるスキップ記号はその順番で右辺に現われなければならない．ただし, 規則中に明示的に記述された文法カテゴリの再配置は自由である．たとえば, 規則 (3-12)はスキップ skip(Y) と skip(Z) を右辺で再配置しているので SDG の枠組を越えている．

スキップ記号の位置は規則の右辺と左辺で変化しないので, SDG の規則はスキップ記号を省略し, DCG 規則の集合で表現できる．たとえば, 規則 (3-17)と規則 (3-18)は等価である．

xp(Cat, empty), skip(X), xp(Cat) →
　xp(Cat), skip(X), xp(Trace).　(3-17)

$$\begin{cases} \mathsf{xp(Cat, empty)} \rightarrow \mathsf{xp(Cat)}. \\ \mathsf{xp(Cat)} \rightarrow \mathsf{xp(Trace)}. \end{cases} \quad (3\text{-}18)$$

規則 (3-18)は見かけは DCG 規則の集合だが, もともと 1 つの規則なので, 規則を使用する際に変数の束縛は共有しなければならない．規則 (3-18)のもうひとつの利点は導出した結果がグラフではなく木になることである．XG や DG で

は規則の左辺に複数の記号が現れるので, 一般に導出結果はグラフとなり, 取り扱いが難しいという問題がある.[64]

Dahl は SDG を用いて Chomsky の GB 理論[26,96]を実現することを目指している. ただし, Stabler の RLG とは異なり, SDG では GB 理論の種々の制約を文法規則とは別に宣言的に記述する方法をとっている. また, 実際に, GB 理論の下接の条件を SDG の枠組で Prolog 上で実現している.[64] SDG の枠組で GB 理論の各原理がどの程度実現できるかは, まだ明らかではない.

3.1.6 XGS

XGS は今野が提案した文法記述形式で,[14] 第2章で説明した LangLAB システムで採用している. XGS では, DCG にスラッシュ記法 "../" を導入し, 移動現象を記述する. 関係節によって修飾された名詞句は規則 (3-19), (3-9) のように記述できる.

np → det, noun(X), relative../np(X). (3-19)
relative → rel-pronoun, s. (3-9)

RLG と同様に, XGS も XG で必要となる rel-marker のような言語学的に根拠のないカテゴリを導入する必要がない. XGS は XG と同様に複合名詞句制約を満たすためにギャップの現れる範囲を指定できる記法も備えている.

3.1.7 YAPX, RCSG

林の解析システム YAPX では XGS を拡張した文法記述形式を用いている.[52] 主な拡張点は XGS のスラッシュ記号の右に複数のカテゴリを指定できる点とドメインと呼ばれる概念の導入である. ドメインは, 一般に

D//B/(α)

と記述し, カテゴリ D が支配する B の下では常にαがギャップとして出現することを宣言する. ドメインを利用すると等位構造中のギャップを扱うことができる.

RCSG は YAPX の文法記述形式に文脈依存性を導入したものである. 林は RCSG で記述した文法を使って高速に統語解析をおこなうアルゴリズムも提案している .[51] RCSG では, 文法規則の右辺の各位置に指標を割り当て, 特定の規則を特定の指標位置のみで解析に使うことにより文脈依存性を実現している. このため高速な解析が可能となるが, どの位置でどの規則を適用するかを文法記述者が意識しなければならない. これは文法記述という観点からは好ましくない.

3.2　スラッシュ記法と支配制約

本節では, まず XGS のスラッシュ記法を簡単に説明し, 我々が提案する支配制約について説明する.

3.2.1　スラッシュ記法

スラッシュ記法は, 英語などに現れる左外置現象を簡潔に記述するために今野らが XGS で導入した記法である.[14] この記法は GPSG[70]のスラッシュ素性の考え方が基本になっている. たとえば, 英語の関係節を XGS によって記述した例を文法 3-3に示す. ただし, relative, rel-pronoun は, それぞれ関係節, 関係代名詞を表す.

文法 **3-3**

s → np, vp.	(3-1)
np → det, noun.	(3-20)
np → det, noun(X), relative../np(X).	(3-19)
vp → verb, np.	(3-4)
relative → rel-pronoun, s.	(3-9)

ここで, 記号 "../" をスラッシュ, スラッシュの右側のカテゴリをスラッシュカテゴリと呼ぶ. 一般に, カテゴリ M../C は,

> 「カテゴリ M を根とする統語木ができたときに, その根の下に, ギャップを直接構成素として支配するカテゴリ C が 1 つ存在する」

ことを表している. 規則 (3-19)の relative../np(X) は, 名詞句をギャップとして持つ関係節を表している. 文法 3-3により, "the girl who loves me" や "the girl whom I love" などの関係節を含む名詞句を統一的に扱うことができる. また, 先行詞 noun とスラッシュカテゴリ np の引数として同一の論理変数を与えることによって, ギャップとその先行詞の対応付けが容易に実現できる. この記法の利点は文法記述者がギャップが現れる位置を意識して文法規則を書く必要がない点である. スラッシュ記法を用いることによって文法規則の数が約 3 割減少した例が今野らによって報告されている.[14] この他にも XGS はギャップの現れる有効範囲を制御するための記法なども用意している. また, このスラッシュ記法を用いて英語の受動化, 疑問文などの規則も記述できる.

3.2.2　スラッシュ記法の限界

XG や XGS による関係節の記述では, 関係代名詞は関係節の存在を示す単なる指標としてしか扱われていない.[14,90] たとえば, 規則 (3-19)では先行詞 noun

と関係節が支配するギャップは論理変数 X によって対応付けられているが, 規則 (3-9)の関係代名詞 rel-pronoun と, 先行詞あるいはギャップは対応付けられていないので次のように関係代名詞の格が不適切な非文も受理してしまう.

*She is the girl whose I've been looking for.

一般に, 論理文法では情報のやりとりを論理変数を介しておこなうが, 論理変数の有効範囲は同一節内に限られているため, この例において先行詞, 関係代名詞, ギャップを正しく対応付けるためには次のいずれかの方法をとらなければならない.

(1) 同一節内に先行詞, 関係代名詞, ギャップが現れるように規則を展開する.

(2) 引数を設けて情報を伝播する.

(1) の方法をとるならば, たとえば, 次のように規則を展開しなければならない.

np → det, noun(X), rel-pronoun(X), s../np(X). (3-21)

ここで, 変数 X は格に関する情報だけを持っていると仮定すると, 単一化により, 先行詞, 関係代名詞, ギャップの格の一致は保証される. しかし, このような規則の展開をおこなうと, relative を規則の右辺に持つ規則数と relative を規則の左辺に持つ規則数の積に等しい数の文法規則が必要となる. これは文法規則数を減少させるというスラッシュ記法の利点を損なうものである. さらに, Pied-piping のように, 関係代名詞を含む構造が移動する場合には, このように規則を展開することが不可能な場合がある. これらの具体的な例については 3.3 節で説明する.

一方, (2) の方法をとるならば, Pereira が DCG で左外置を扱う場合にとった方法[90]と同様に専用の引数を設けて情報を伝播しなければならない. たとえば, 文法 3-3は文法 3-4のように修正できる.

文法 **3-4**

s → np, vp.	(3-1)
np → det, noun.	(3-20)
np → det, noun(X), relative(X)../np(X).	(3-19)'
vp → verb, np.	(3-4)
relative(X) → rel-pronoun(X), s.	(3-9)'

しかしながら, このような方法をとると, どの文法規則を経由して情報が伝播するかを意識して文法を記述しなければならないので文法記述者の負担が増える. ある程度一般性のある現象を扱うためには, メタな記法を用意した方が文法の記

述が容易になる. たとえば, XG では左外置という一般的な現象を表現するのに "..." 記法を導入している.

別の考え方として, 単一化文法[97]の枠組でおこなわれているように, 各文法カテゴリを素性の束と考え, 素性の伝播と単一化によって文を解析するという方法もある. しかし, すべてを単一化操作によって処理するのは効率の面で問題がある. 以上のように, 移動した構成素とギャップの間で正しく情報を受け渡すことを考えるとスラッシュ記法には限界があることがわかる. 重要なことは規則を越えて, いかに情報を簡潔に伝播するかという点である. より柔軟な情報の受け渡しを実現するために支配制約という概念を導入する.

3.2.3　支配制約

支配制約はスラッシュ記法を一般化した概念である. スラッシュ記法が「スラッシュカテゴリが親カテゴリの下でギャップを支配すること」を要請するのに対し, 支配制約は「あるカテゴリが親カテゴリの下に存在する」ことを要請する. 例として文法 3-3に対応する次の英語の文法規則を考えよう.

文法 **3-5**

s → np, vp.	(3-1)
np → det, noun.	(3-20)
np → det, noun(X), relative@rel-pronoun(X).	(3-22)
vp → verb, np.	(3-4)
relative → np(X), s//np(X).	(3-23)
np → rel-pronoun.	(3-24)

ここで, "//" は XGS のスラッシュと同じである. XGS と区別するために, ここではこの記号を使う. "@" は支配制約を記述するために導入した記法で, "@" の右側を被支配カテゴリという. 一般に, カテゴリ "M@C" は,

> 「カテゴリ M を根とする統語木ができたときに, その根の下にカテゴリ C が存在する」

ことを表している. スラッシュ記法と異なり, カテゴリ C は必ずしもギャップを支配する必要はない. 規則 (3-22)の relative@rel-pronoun(X) は relative が rel-pronoun を支配していることを表している. この場合もスラッシュ記法と同様に, 論理変数によって情報を受け渡すことができる. 支配制約は被支配カテゴリとそれを支配するカテゴリの間で情報の受け渡しをおこなう手段として使うことができる. 具体的な記述例については次節で述べる.

3.3 記述例

本節では, 英語のいくつかの移動現象を支配制約を用いて記述し, その有効性について述べる.

3.3.1 関係節

スラッシュ記法と支配制約を用いて, 移動した構成素とギャップの対応関係も含めて関係節が正しく記述できることを示す. 文法 3-5に所有格の関係代名詞を扱うための規則 (3-25)を追加したのが文法 3-6である.

文法 **3-6**

s → np, vp.	(3-1)
np → det, noun.	(3-20)
np → det, noun(X), relative@rel-pronoun(X).	(3-22)
vp → verb, np.	(3-4)
relative → np(X), s//np(X).	(3-23)
np → rel-pronoun.	(3-24)
det → rel-pronoun.	(3-25)

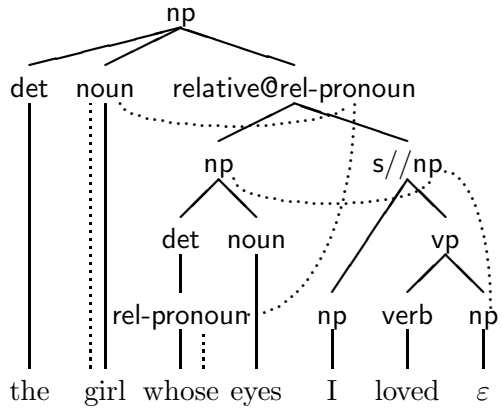


図 **3-1**　支配制約を用いた関係節の解析例

文法 3-6を用いて所有格の関係代名詞を含む名詞句 “the girl whose eyes I loved” を解析して得られる統語木を図 3-1に示す. ここで, 実線は構成素の支配関係を, 点線は論理変数によって構成素が対応付けられていることを表している. 記号 “ ε ” は構成素が移動した後のギャップを表す. 図 3-1から先行詞 “girl” が noun, relative@rel-pronoun, rel-pronoun を経て “whose” と対応付けられていることがわかる. したがって, 意味構造を構成することを考えると, “whose eyes”

から “*girl's eyes*” という意味が構成できる. また, “whose eyes” から構成される np も s//np を経てギャップと正しく対応付けられている. これによって埋め込み文から “*I loved the girl's eyes*” という意味が構成できる.

一方, 同じ例文を XGS の枠組を用いて解析する場合を考えよう. 文法 3-3に所有格の関係代名詞を扱うために, 規則 (3-25), (3-2), (3-26)を追加する.

文法 **3-7**

s → np, vp.	(3-1)
np → det, noun.	(3-20)
np → det, noun(X), relative../np(X).	(3-19)
np(X) → det, noun(X), relative.	(3-2)
vp → verb, np.	(3-4)
relative → rel-pronoun, s.	(3-9)
relative → np(X), s../np(X).	(3-26)
det → rel-pronoun.	(3-25)

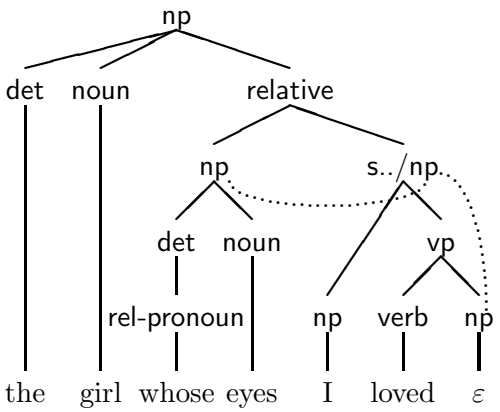


図 **3-2**　**XGS** による関係節の解析例

規則 (3-19), (3-9)は主格, 目的格の関係代名詞を扱う規則, 規則 (3-2), (3-26)は所有格の関係代名詞を扱う規則であるが, 両者を比べると, スラッシュカテゴリの位置が異なっている. 主格, 目的格の関係代名詞を扱う規則では, スラッシュカテゴリが relative に付いているが, 所有格の関係代名詞を扱う規則では, relative ではなく s に付いている. XGS では, 移動するカテゴリは, そのカテゴリを支配するカテゴリと同一規則の右辺に現れなければならない. 所有格の関係代名詞では関係代名詞を含む構造が移動する. この例では移動するのは “girl's eyes” であって “girl” ではないから, 規則 (3-19)は使うことができない. 文法 3-7を用いて “the girl whose eyes I loved” を解析した結果を図 3-2に示す. 図 3-2では,

左に移動した “whose eyes” と埋め込み文の目的語のギャップは対応付けられているが, 先行詞 “girl” と関係代名詞 “whose” との対応が付かないので “whose eyes” が “*girl's eyes*” であることが解析できない. これを解決するには, 3.2 節でも述べたように規則を展開するか, 特別な引数を設けて情報を伝播しなければならない. しかし, これが規則数を増やす原因となることはすでに 3.2 節で指摘した.

所有格の関係代名詞を含む関係節の記述を XG でおこなうと文法 3-8となる.

文法 **3-8**

s → np, vp.	(3-1)
np → det, noun, relative.	(3-2)
np → det, noun, pp.	(3-27)
np → trace.	(3-3)
vp → verb, np.	(3-4)
vp → verb.	(3-5)
relative → [ ].	(3-6)
relative → rel-marker, s.	(3-7)
rel-marker. . . trace → rel-pronoun.	(3-8)
rel-marker,[the]. . . [of],trace → [whose].	(3-28)
pp → p, np	(3-29)

文法 3-8からわかるように, XG の場合も先行詞, 関係代名詞, ギャップを対応付けるためには, XGS 同様, 規則を展開するか, 引数を使って情報を伝播する必要がある. XG で所有格の関係代名詞を解析する場合は, 規則 (3-28)を使い, “whose X” を “the X of trace” に書き換え, さらに規則 (3-27)によって解析をおこなう. 林らの YAPX でも所有格の関係代名詞を同様の書き換えによって扱っている.[52] しかし, このような書き換えをおこなう言語学的根拠はない. たとえば, GPSG では, 関係節を WH 素性を用いて次のように分析している.[70]

$$[\mathsf{N'}\ \mathsf{N}_j\ [\mathsf{S'}\ \mathsf{NP}_i\{+\mathsf{R}_j\}\ [\mathsf{S}\ \ldots \varepsilon_i\ \ldots]\ ]\ ]$$

ここで, +R は [WH NP[WHMOR R]] の省略形で, 関係代名詞を表す. NP_iは関係代名詞を含む名詞句で S 中のギャップε_iと同一の指標 i が付けられている. また, 先行詞 N_jと NP_i中の関係代名詞も同一の指標 jが付けられている. GPSG の分析によれば, S の主格, 目的格の位置がギャップとなる場合は, 指標 i と jが同一の特殊な場合となる. 一方, 所有格の関係代名詞の場合は i と jが異なる.

支配制約を使って記述した文法 3-6と GPSG の関係節の解析を比べると規則 (3-22)中の relative が S' に対応し, rel-pronoun が$+\mathsf{R}_j$に対応していることが

わかる．GPSG では, 素性を種々の原則によって制御しながらカテゴリ間で伝播させるが, 支配制約はその伝播のショートカットを与えていると考えることができる．このように, 文法 3-7, 文法 3-8と比べると文法 3-6の方が関係節の記述に一貫性があるといえる．実際, 次に述べる Pied-piping のような関係節化では, XGS や XG の方法では破綻をきたす．

3.3.2 Pied-piping

関係節を形成する時に, 関係節中から節頭に移動する構成素が単に先行詞と同一の名詞句あるいは Wh を付与された名詞句だけでなく, その名詞句を支配する上位の名詞句が移動する場合がある．このような現象を Pied-piping という．この現象を説明するために Ross は Pied-piping convention という変形操作に対する制約を提案した.[29] たとえば, 次の例を考えよう.[29]

(a) $[_{\mathrm{s}}$ the government prescribes $[_{\mathrm{np}_1}$ the height of
$[_{\mathrm{np}_2}$ the lettering on $[_{\mathrm{np}_3}$ the covers of $[_{\mathrm{np}_4}$ the reports]]]]]

(b) the reports which the government prescribes the height of the lettering on the covers of
(表紙の文字の位置が政府によって規定された報告書)

(c) the reports the covers of which the government prescribes the height of the lettering on

(d) the reports the lettering on the covers of which the government prescribes the height of

(e) the reports the height of the lettering on the covers of which the government prescribes

(a) の文から np_1, np_2, np_3, np_4をそれぞれ節頭に移動すると (e), (d), (c), (b) の名詞句ができる．XGS の場合, 文法 3-7の規則 (3-2), (3-26)を使って任意の np_iを移動することができるが, すでに述べたようにこれらの規則では先行詞とギャップの対応付けができない．この場合, 規則を展開して対応付けをしようとすると, 一般には不定個の前置詞句を引き連れた名詞句が移動するので無限個の規則を用意しなければならなくなる．

XG の場合, 文法 3-8では np_4しか移動できない．つまり, (b) しか解析できない．たとえば, np_3を移動するには規則 (3-30)を文法 3-8に追加する必要がある．

rel-marker. . . trace → det, noun, p, rel-pronoun. (3-30)

このように, XG では各 np_iを移動するのに 1 つの規則を追加する必要があるので, 一般に不定個の前置詞を持つ名詞句の移動を記述するためには無限個の規則が必要となる. 支配制約を使うと, この例を文法 3-6と規則 (3-27), (3-29)を用いて所有格の関係代名詞とまったく同様に解析できる. (c) の解析例を図 3-3に示す. 先行詞 “reports” と関係代名詞 “which” の対応, 移動した名詞句 “the covers of which” とギャップの対応がそれぞれとれていることがわかる.

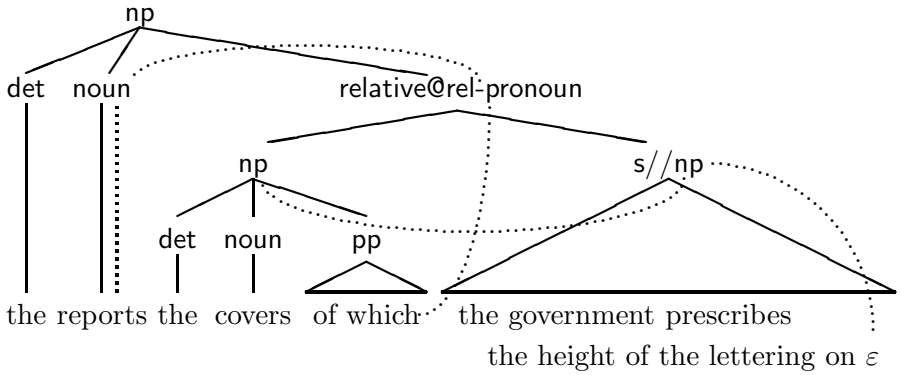


図 **3-3**　先導の規約を必要とする関係節の解析例

3.3.3　等位構造

“and” や “or” などの等位接続詞によって任意の構成素が接続される構造を等位構造という. もちろん, 等位接続される構成素の組合せには制約があるが,[93] ここでは言語学的な詳細には立ち入らない.

等位構造は自然言語解析において統語木のあいまい性やスコープのあいまい性など多くの問題を引き起こす. これまでの等位構造の扱いに関する研究は, 等位構造の記述を文法記述者に透明にするか, 文法記述者の手に委ねるかによって大きく 2 種類に分類できる. 前者のアプローチとしては, 3.1 節で紹介した Dahl と McCord の MSG をはじめ, Sedogbo のメタ文法,[95] Fong と Berwick の RPM,[68] 武舎の Predictive Analyzer,[85] Hirschman の Meta-Restriction Grammars (MRG)[71] などがある.

Sedogbo のメタ文法では, 等位構造に関する規則を含まない文法規則の右辺に等位構造を処理する CONJ という特別な述語を付加する. CONJ は解析を制御し, 解析時に動的に等位構造の処理をおこなう. 武舎の Predictive Analyzer は特別な AND スタックというデータ領域を用いて等位構造を解析する. これらの手法では, 解析中にあらかじめ決められた “and”, “or”, “but” などのキーワードに出合うと, 解析器は処理を一時中断し, 等位構造のための特別な処理をおこなう.

Fong と Berwick は入力単語列を部分的に非終端記号にまとめあげた記号列の集合 (これを RPM と呼ぶ) の間の集合演算に基づいて等位構造を扱う方法を提案している.[68]

Hirschman の Meta-Restriction Grammars (MRG)[71]は等位構造に関する記述を含まない文法を等位構造を扱える文法にコンパイルし, 解析をおこなう手法を与えている. この方法は文法規則をあらかじめコンパイルするので, MSG などの割り込み型の方法に比べ効率がよいとされている.

Kac と Rindflesch の Reconnaissance-Attack パーザ[74]は述語の等位構造だけを対象とし, 2 パス方式を採用した現実的な処理法を与えている.

これらの枠組では, 等位構造に関する規則を文法記述者が記述する必要がなく, 等位構造は解析器によって解析中に動的に, あるいはコンパイラによって事前に処理される. このアプローチには等位構造に関する記述が不要であるという利点はあるが, 等位構造の処理に変更を加えることが困難であるという欠点がある.

これに対して, DCG, XG などの論理文法は, 等位構造に対する特別な処理機構を備えておらず, その扱いも文法記述者の手に委ねられている. 我々も文法記述者から等位構造の処理を隠蔽するのではなく, 記述のための道具を与える方がよいと考えている.

もっとも単純な等位構造は,

Tom loves Mary and John loves Jane.

のように完全な構成素が等位接続詞によって接続される場合である. これは DCG の枠組でも容易に記述できる. しかしながら, 等位接続される構成素の一方がギャップを含む場合は簡単ではない. たとえば,

Mary saw and John heard the train.

(a) $[_{s}[_{s}[_{np}\text{Mary}][_{vp}[_{v}\text{saw}][_{np}\text{the train}]]]$ and $[_{s}[_{np}\text{John}][_{vp}[_{v}\text{heard}][_{np}\text{the train}]]]]$

(b) $[_{s}[_{s}[_{s}[_{np}\text{Mary}][_{vp}[_{v}\text{saw}][_{np}\ \varepsilon\ ]]]$ and $[_{s}[_{np}\text{John}][_{vp}[_{v}\text{heard}][_{np}\ \varepsilon\ ]]]][_{np}\text{the train}]]$

伝統的な変形文法では, このような文は, (a) の深層構造から Right Node Raising という変形を適用して (b) の表層構造が派生するという分析をおこなうが, 以下の議論では, 木村の空照応分析[75]に近い方法を採用する.

このような文を解析するために Dahl は次のような DG の文法を与えている.[64]

文法 **3-9**

s(and(S1,S2)) → sent(S1),and,sent(S2). (3-31)

sent(S) → name(K),verb(K,P,S),object(P). (3-32)

object(P) → det(X,P1,P),noun(X,P1). (3-33)

object(P), and, skip(G),object(P) → [and],skip(G),object(P). (3-34)

上述の例文は規則 (3-34)によって, “and John heard the train” を “the train and John heard the train” に書き換えて解析する. 規則 (3-34)の右辺の object と左辺の 2 つの object に共通な論理変数 P を引数として持たせることによって, 等位接続された 2 つの目的語 (“the train”) が同一であることを表現している. ところが DG の代表的な実装である SYNAL システムでは, 実はこの同一性は保証されない. Dahl らの実装によると,[54,63] 規則 (3-34)は次のようなホーン節に変換される.

```
object(P,X0,X) :-
    C(X0,and,X1),skip(G,X1,X2),object(P,X2,Z),
    and(X,Y1), skip(G,Y1,Y2), object(P,Y2,Z).
```

ボディの最初の skip では, バックトラックにより文字列を必要なだけ読み飛ばし, 最初の object の解析を成功させる. skip の引数 G には読み飛ばされた文字列が束縛される. 前述の例文では, G に “John heard” が束縛される. ボディの 2 番目の skip では引数 G を入力文字列の先頭に連接し, 2 番目の object では引数 P の情報 (ここでは, 簡単な意味情報) をもとに文字列 (この例では “the train”) を生成する. したがって, 2 番目の object の生成が終了した時点では差分リスト X0-X は “and John heard the train” を保持している. この時点で等位接続される前半の文 “Mary saw the train” が解析できたことになる. 次に “and” を解析し, 後半の文を解析するが, この文の目的語である “the train” と先行する文の目的語 “the train” が同一であることは, 規則 (3-34)の両辺の object に同一の論理変数 P が引数として与えられていても保証されない. なぜなら, 規則 (3-34)によって 1 つの目的語をコピーする際に文字列を経由してコピーをおこなっているからである. SDG の実装ではこのような問題は解決されているが,[62] 3.1 節で述べたように SDG は DG に比べて記述力が制限されている.

XGS で, 同じ例文を解析するためには, たとえば, 次の文法が必要となる.

文法 3-10

```
s → sd.                                          (3-35)
s → sd../np([case:obj]), [and], s.               (3-36)
sd → np, vp.                                     (3-37)
np → det, noun.                                  (3-20)
vp → verb, np.                                   (3-4)
```

スラッシュカテゴリ np の引数として [case:obj] を渡している点に注意して欲しい. これにより, sd の下でギャップとなっている np は目的格の素性を持たなければならないことを要請している. しかしながら, この規則では等位接続される

最初の sd の目的語と次の s の目的語が同一であることを記述していない. XGS では, この規則中で 2 番目の s が支配する構成素に関する情報を参照する手段がないため, この記述は不可能である. したがって, “love” の目的語が “the girl” であるということが解析できない.

一方, 支配制約を用いると規則 (3-36)の代わりに規則 (3-38)のように記述することができる.

文法 **3-11**

s → sd.	(3-35)
s → sd//np([case:obj | X]), [and], s@np([case:obj | X]).	(3-38)
sd → np, vp.	(3-37)
np → det, noun.	(3-20)
vp → verb, np.	(3-4)

規則 (3-38)は右辺の sd の下でギャップを支配する np の情報と, s が支配する np の情報がともに目的格の素性を持ち, かつ単一化可能であることを要請している. したがって, 意味的にも sd と s の目的語が同一であることが正しく解析できる. 等位接続詞が 2 つ以上ある場合も規則 (3-38)を再帰的に適用することによって同様に解析できる. 文法 3-11による解析例を図 3-4に示す.

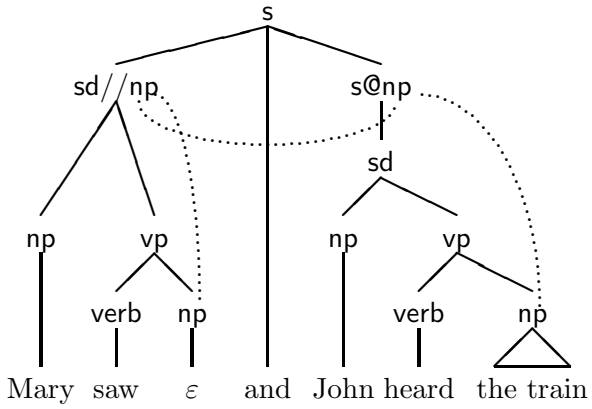


図 **3-4**　ギャップを含む等位構造の解析例

さらに, このような等位構造を持つ文を埋め込み文として持つ関係節も同様に解析できる. 例として名詞句 “the train which Mary saw and John heard” の解析例を図 3-5に示す. 図 3-5において, ε_2がスラッシュカテゴリ np に対応していると同時に, 被支配カテゴリの np に対応している点に注意して欲しい. ε_2を直接支配しているのは np なので, ε_2と被支配カテゴリの np の対応をとることができる. これにより, ε_1とε_2の対応がとれ, さらにε_2と先行詞 “train” の対応がとれる.

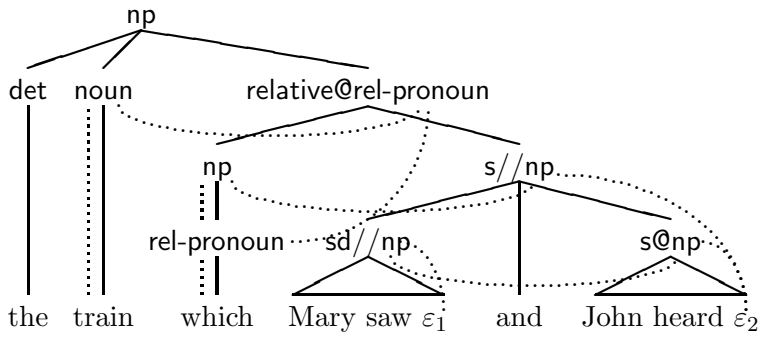


図 **3-5**　等位構造を含む関係節の解析例

3.3.4　排他的文法カテゴリ

第2章で述べたように排他的文法カテゴリを用いて句動詞の記述をおこなうと, 辞書項目のバーレベルが 1 つ上がり, 動詞の目的語の情報を文法からどのように参照するかが問題となる. 句動詞の目的語の情報を素性の値として解析をおこなうことも考えられるが, そのためには句動詞だけに特別な素性を用意しなければならない. 一方, 支配制約を用いて記述すると特別な素性を用意することなくこの問題を解決することができる. 記述例を以下に示す.

vp([V, object:N]) → verb(V), np(N). (3-39)
vp([V, object:N]) → phrasal-verb(V)@np(N). (3-40)
verb(summon) → [summon]. (3-41)
phrasal-verb(summon) → [call], ^np(N), [up], ^np(N). (3-42)

規則 (3-39), (3-40)は動詞句を構成する規則, 規則 (3-41), (3-42)は辞書項目である. 各カテゴリの引数はそのカテゴリの意味を表している. 規則 (3-40)では, 支配制約を使うことにより, 句動詞 “call up” の目的語の np の情報を参照している. これにより通常の動詞句を表す規則 (3-39)とまったく同様に文法規則中で意味を計算できる.

3.4　実装

本節では 3.2 節で述べたスラッシュ記法と支配制約の 1 つの実装法について述べる. ここでは, 多くの Prolog インタプリタに組み込まれている DCG トランスレータの存在を仮定し, スラッシュ記法と支配制約を含む文法規則を, トランスレータにより DCG に変換するという方法をとる. DCG に変換された文法規則はスラッシュカテゴリや被支配カテゴリを管理する補助述語とともに Pereira

の手法[91]により Prolog プログラムに変換され, Prolog インタプリタで直接実行される.

3.4.1　トランスレータ

スラッシュ記法と支配制約を含む文法規則を DCG に変換するトランスレータは 2 引数述語 trans として実現されている. 述語 trans の完全な定義は文献 [38] を参照されたい.

(a) 変換前

```
(a1)  :- op(680,xfy,':').
(a2)  s(_,s(T)) →
        sd(_,T).
(a3)  s(_,s(T1,and,T2)) →
        sd(_,T1)//np([case:obj|N],T),
        [and],
        s(_,T2)@np([case:obj|N],T).
(a4)  sd(_,sd(T1,T2)) →
        np(N,T1),
        {unify(N,[case:sbj|_])},
        vp(_,T2).
(a6)  np(N,np(T)) →
        n(N,T).
(a7)  vp(_,vp(T1,T2)) →
        v(_,T1),
        np(N,T2),
        {unify(N,[case:obj|_])}.
(a8)  n(_,n(john)) → [john].
(a9)  n(_,n(mary)) → [mary].
(a10) n(_,n(train)) → [the,train].
(a11) v(_,v(saw)) → [saw].
(a12) v(_,v(heard)) → [heard].
```

(b) 変換後

```
(b1)  :- op(680,xfy,':').
(b2)  s(_,s(T),A,X0,X1) →
        sd(_,T,A,X0,X1).
(b3)  s(_,s(T1,and,T2),A,X0,X2) →
        sd(_,T1,A,[np([case:obj|N],T)|X0],X1),
        {found_gap([np([case:obj|N],T)|X0],X1)},
        [and],
        s(_,T2,[a(F,np([case:obj|N],T))|A],X1,X2),
        {found_dom([a(F,np([case:obj|N],T))|A])}.
(b4)  sd(_,sd(T1,T2),A,X0,X2) →
        np(N,T1,A,X0,X1),
        {unify(N,[case:sbj|_])},
        vp(_,T2,A,X1,X2).
(b5)  np(N,T,A,[np(N1,T1)|X0],X0) → [ ],
        {unify_args([N,T],[N1,T1])},
        {found(np(N,T),A)}.
(b6)  np(N,np(T),A,X0,X1) →
        n(N,T,A,X0,X1),
        {found(np(N,np(T)),A)}.
(b7)  vp(_,vp(T1,T2),A,X0,X2) →
        v(_,T1,A,X0,X1),
        np(N,T2,A,X1,X2),
        {unify(N,[case:obj|_])}.
(b8)  n(_,n(john),_,X,X) → [john].
(b9)  n(_,n(mary),_,X,X) → [mary].
(b10) n(_,n(train),_,X,X) → [the,train].
(b11) v(_,v(saw),_,X,X) → [saw].
(b12) v(_,v(heard),_,X,X) → [heard].
```

図 **3-6**　文法の変換例

文法の変換例を図 3-6に示す. 図 3-6 (a) は英語の等位構造を解析するための文法で, 3.3 節で例として使用した文法 3-11を修正したものである. この文法により “Mary saw and John heard the train” が解析できる. 各文法カテゴリは 2

つの引数を持ち, 第 1 引数は格に関する情報を, 第 2 引数は統語木の情報を保持する. 解析結果として統語木が第 2 引数にリストの形で得られる. 述語 unify は CIL[84]で採用されている部分項の単一化をおこなう述語である. 部分項の単一化は基本的にマージ操作なので, 同じ変数を用いて値を返すことができる. 以下これを Prolog の単一化と区別するために拡張単一化とよぶ. ここでは最後尾の要素として変数を持つ Prolog のリストを用いて部分項を実現している.

変換により文法中の Prolog プログラム, および規則中の “{” と “}” で囲まれた補強項はそのまま出力に複写する (図 3-6 (b1) および (b4), (b7) の補強項参照). 各文法カテゴリには, 2 つの X リスト[14,90]と 1 つの A リストを引数として付加する. X リスト, A リストは, それぞれスラッシュカテゴリの情報と被支配カテゴリの情報を保持する引数である. 以下では, 文法規則の変換をスラッシュ記法の変換, 支配制約の変換を中心に説明する.

3.4.2 スラッシュ記法の変換

スラッシュ記法の変換は, Pereira,[90] 今野[14]の手法とほぼ同じである. 各カテゴリに付加する 2 つの X リストはスタックで, それぞれ, そのカテゴリを解析する前 (入力 X リスト) と解析した後 (出力 X リスト) のスタックの状態を保持している. スラッシュカテゴリを含むカテゴリを変換する場合は, 入力 X リストにスラッシュカテゴリを引数とともにプッシュする (図 3-6 (b3) の sd を参照). スラッシュカテゴリを X リストからポップするのはスラッシュカテゴリと同じカテゴリを解析する時であるから, 文法規則中にスラッシュカテゴリとして出現するカテゴリについては次の規則を追加する.

$$\mathsf{Cat(Args1,A,}\underline{\mathsf{[Cat(Args2)|X0]}}\mathsf{,X0)} \rightarrow \mathsf{[\,]},\ \{\mathsf{unify\_args(Args1,Args2)}\}. \tag{3-43}$$

この規則は ε 規則であり, 入力文字列中にギャップが存在する時に使用する. 図 3-6の例では (b5) がこの規則に相当する. この例では, np がスラッシュカテゴリであると同時に被支配カテゴリでもあるため, 次項で述べる述語 found が追加されている. 規則 (3-43)において, Cat はスラッシュカテゴリ名, Args2 はスラッシュカテゴリが持つ情報, A は支配制約のための A リストである. A リストについては次項で述べる. 下線を引いた部分が X リストである. 入力 X リストの先頭 (スタックトップ) にはスラッシュカテゴリ Cat が存在し, 出力 X リストは入力 X リストの先頭を取り除いた残り部分になっている. すなわち, この規則を使ってカテゴリ Cat の解析をおこなうと, 入力文字列を消費するかわりに, 入力 X リストの先頭にカテゴリ Cat があれば, これをポップし, 残りの X リストを出力 X リストとして返すことになる. ただし, この規則を使う時点で検出

したギャップの持つ情報 (Args1) はスラッシュカテゴリに引数として指定された情報 (Args2) と矛盾してはならない．この検査をおこなうために述語 unify_args を用いて Args1 と Args2 を拡張単一化する．unify_args はこのための述語である．また, スラッシュカテゴリを持つカテゴリを解析した直後には, 出力 X リスト中のスラッシュカテゴリがポップされたことを検査する述語 found_gap を補強項として挿入する．章末に述語 found_gap, unify_args の定義を示す．

3.4.3　支配制約の変換

支配制約も A リストを用いてスラッシュ記法と同様な手法によって実現できる．スラッシュカテゴリとギャップの対応付けは非交差なので X リストはスタックによって実現しているが, 支配制約では被支配カテゴリと実際に見つかったカテゴリの対応付けが非交差である必要はない．このため, A リストはスタックではなく集合となる．文法規則の変換の概略を以下に示す．

- 支配制約を持つカテゴリの A リストには被支配カテゴリを以下の構造で追加する．

  a(Found, Cat(Args))

  これを A 構造と呼ぶ．第 1 引数の Found は被支配カテゴリを検出するとアトムに束縛される変数である．支配制約を持つカテゴリの解析が終了した時点で, この変数の束縛状態を調べることにより支配制約を満足しているかどうかを検査する．この検査をおこなうのが以下で説明する述語 found_dom である．Cat は被支配カテゴリ名, Args はその引数である．例として図 3-6 (b3) の右辺の s の変換結果を参照されたい．

- 被支配カテゴリとして文法中に現れるカテゴリを左辺に持つ文法規則の末尾には, 述語 found を補強項として挿入する．述語 found は, 第 1 引数に解析によって完成したカテゴリ, 第 2 引数に A リストをとり, 第 1 引数のカテゴリと引数を含めて拡張単一化できるカテゴリを含む A 構造を A リストから探し, その A 構造の第 1 引数の変数をアトムに束縛する．この変数の束縛によって, 被支配カテゴリが検出されたことを A リスト中に記録する．章末に述語 found の定義を示す．

- 支配制約を持つカテゴリの直後には補強項として述語 found_dom を挿入する．found_dom は, 直前の支配制約を持つカテゴリの A リストに挿入した A 構造の第 1 引数がアトムに束縛されているかどうか調べる．ここで, A 構造の第 1 引数が変数であれば, 支配制約を満たしていないことになるので found_dom は失敗する．もし支配制約を満たしていれば, その被支配

カテゴリと残りの A リストに対して述語 found を再帰的に適用する．これによって, 複数の被支配カテゴリを同一のカテゴリと対応付けることが可能となる．述語 found_dom の定義を章末に示す．

3.4.4 解析実行例

図 3-6の変換後の文法 (b) と補助述語を用いて, 例文 “Mary saw and John heard the train” の解析を例にとって解析器の動作を説明する．解析は次のゴールを呼び出すことによって始まる．

```
?- s(_, Tree, [ ], [ ], [ ], [mary, saw, and, john, heard, the, train], [ ]).
```

このゴールは (b3) の左辺と照合し, 右辺のゴールを呼び出す．(b4) により等位接続された前半の文 “mary saw ε ” を解析する．この時, 主語の np の解析が (b6) により成功すると found が呼び出されるが, この段階では A リストは空なので found は単に成功する．その後, 補強項により np の第 1 引数には主格の素性が追加される．次に, (b7) により動詞句の解析をおこなうが, 目的語はギャップとなっているので, 入力文字列を消費するかわりに (b3) で X リストにプッシュされたスラッシュカテゴリの np を (b5) によりポップし vp の解析を成功させる．この時に (b5) の unify_args によってギャップの格情報 (N) とスラッシュカテゴリの格情報 (N1) を拡張単一化することにより, このギャップはスラッシュカテゴリで指定された素性 case:obj を持つことになる．(b7) の補強項 unify によって目的語 (ギャップ) の格情報が case:obj を持つことを検査するが, これは今述べた理由から成功する．

以上で, (b3) の右辺の sd の解析が終了したので補強項の found_gap を実行するが, 目的語の解析でスラッシュカテゴリをポップしたので, これは成功する．次に後半の s の解析をおこなう．この例では 2 文が等位接続されているので, 今度は (b2) を使う．引続き (b4) が呼び出され, (b6) を用いて主語の np の解析をする．(b6) のボディの最後で, found を呼び出し, A リストの中の A 構造の np と, ここで完成した np(“john”) の格情報を拡張単一化する．これで, この主語の np は目的格の素性を持つことになる．しかし, (b4) の補強項では, 主語の np は主格の素性を持つことを要請しているので, 補強項の失敗によりバックトラックを起こす．結局, 入力 A リストをそのまま出力 A リストに返すことにより found は成功する．

次に (b4) の動詞句の解析に移る．動詞の解析を終ると (b6) により目的語の np の解析をおこなう．ここで, 先ほどと同様に found が A リスト中の a 構造の np と目的語の np を拡張単一化し, A 構造の第 1 引数をアトムに束縛する．次に補強項を実行するが, 今度は格に関する素性が一致し, vp の解析, そして s の解

析が成功する. 次に (b3) の最後の補強項 found_dom を実行する. (b6) で A リスト中の A 構造の第 1 引数はアトムに束縛されているので, これは成功する. 以上で解析が成功し, 以下のような統語木の情報が得られる.

```
s(sd(np(n(mary)),vp(v(saw),np(n(the,train)))), and,
  s(sd(np(n(john)),vp(v(heard),np(n(the,train))))))
```

第 1 文の目的語にも正しく “the train” が補われている点に注意して欲しい.

3.5　要約

本章では, 論理文法上でギャップを扱うために支配制約という新しい概念を導入し, いくつかの英語の移動現象が自然に記述できることを示した. また, Pereira が提案した DCG から Prolog プログラムへの変換手法を拡張することによって支配制約が容易に Prolog 上で実現できることも述べた. Pereira の DCG はすでに文脈自由文法の枠組を越えている. 文法の生成能力という観点からは, 本章で提案した支配制約は DCG を越えるものではない. しかしながら, Dahl も指摘しているように, DCG が文脈自由文法を越えているのは, 補強項という手続き的な部分においてであり, これを宣言的な文法記述に反映させることは, 文法記述の容易さという観点からは重要である.[62] 支配制約の考え方はこの線に沿うものであり, 移動した構成素とそのギャップを正しく対応付ける規則を宣言的に記述できる能力を与えている.

付録 : 補助述語の定義

```
:- op(680,xfy,':').

found(_,[]) :- !.
found(T1,[a(Found,T2)|_]) :-
    var(Found),
    Found=found,
    T1=..[P|Args1],
    T2=..[P|Args2],
    unify_args(Args1,Args2).
found(T,[_|R]) :-
    found(T,R).

found_dom([a(Found,Cat)|A]) :-
    nonvar(Found),
    found(Cat, A).

found_gap([_|_],[]) :- !.
found_gap([_|As],[_|Bs]) :-
    found_gap(As,Bs).

unify_args([],[]) :- !.
unify_args([A|As],[B|Bs]) :-
    unify(A,B),
    unify_args(As,Bs).

unify(A,B) :-
    A = B, !.
unify([A|As],Bs) :-
    unify_slot(A,Bs,R),
    unify(As,R).

unify_slot(A,Bs,R) :-
    var(Bs), !,
    Bs = [A|R].
unify_slot(C:Va,[C:Vb|Bs],Bs) :- !,
   unify(Va,Vb).
unify_slot(A,[B|Bs],[B|R]) :-
   unify_slot(A,Bs,R).
```

第 4 章

日本語の語順に関する知識とその利用

本章では, 佐伯の提案したかかりの広さを定量化し, かかりの広さに基づく日本語語順の推定モデルを提案する.[35] このモデルは日本語の文要素 (後置詞句または, 動詞に限定) が次々に入力されるとき, 各入力段階で次入力に対する推定を優先度という形で与える. このモデルの妥当性を検証するために情報処理振興事業協会技術センターが作成した計算機用日本語基本動詞辞書 (以下 IPAL 動詞辞書) を用いた実験をおこなった. 以下, 4.2 節では, 佐伯の語順に関する知見について説明し, その定量化と我々のモデルについて述べる. 4.3 節では, 実験に用いた IPAL の内容と実験について述べる. 4.4 節では, 我々のモデルの自然言語処理への応用について述べ, 最後に 4.5 節では, モデルの拡張について述べる.

4.1 背景

日本語は語順が比較的自由な言語であるといわれているが, 語順にまったく制限がないわけではない.[1] 特に述部の語順に関しては言語学者による多くの研究があり, かなり厳格な語順規則があることが明らかにされている.[49] 一方, 述部にかかる要素間の語順については述部の語順ほど厳格な規則は見い出されていない.

児玉は依存文法の立場から言語に普遍的な語順を規定する原則をまとめおり, その一例として日本語をとりあげている.[18] 児玉は語順を決定する要因として,

- 依存関係 (主要語と修飾語)
- 品詞 (名詞, 動詞, 前/後置詞)
- 文法機能 (主語, 目的語, 補語, 述語)

- 形態 (屈折, 接辞などの語形態と語連鎖の重み)
- 意味 (語用論, 認知構造)

の 5 つを挙げ, 中でも依存関係を重要視している. 児玉の研究は言語普遍性を重視しているために, 日本語固有の問題については深く立ちいっていない.

久野は文を越えた談話という観点から, 情報の新旧, 話題, 焦点などが語順に及ぼす影響について論じている.[12] 本論文では単文を考察対象とし, 談話情報については考慮しない.

佐伯は小説 67 ページ中の文を手作業で分析し, 補語, すなわち, 名詞句と格助詞の組の語順について以下の 9 つの傾向を抽出している.[15]

(1) 位格 (ニ, デ, カラ, ヲ) は他の格に先行する

(2) トキの位格はトコロの位格に先行する

(3) ガは位格を除く他の格に先行する

(4) 与格のニは対格のヲに先行する

(5) カラは着格のニ, へに先行する

(6) 長い補語は短い補語に先行する

(7) 文脈指示を含む補語は先行する

(8) 補語同士がかかり受けを構成する場合, かかりの補語が先行する

(9) 慣用句では, 特定の補語は動詞の直前に位置する

佐伯はこれらの傾向を成分条件に基づく語順傾向と構文条件に基づく語順傾向の大きく 2 つに分類している.

成分条件に基づく語順傾向とは, 補語そのものの意味, 機能に備わった支配条件に基づいて生じる語順傾向のことで, これは児玉のいう文法機能に関する要因にほぼ相当する. (1) から (9) のうち, (1) から (5) がこれにあたる. これらの傾向をまとめると, おおよそ次のようになる.

$$\text{位格 (トキ} > \text{トコロ)} > \text{主格} > \text{与格} > \text{対格}$$

ただし, X > Yは X が Y に先行することを意味する.

一方, 構文条件に基づく語順傾向とは, 補語の構文的な意味, 機能に備わった支配条件に基づいて生じる語順傾向のことで, 児玉のいう依存関係, および形態に関する要因に相当する. (6) から (9) がこれにあたる.

佐伯は (1) から (5) の成分条件に基づく語順傾向については, かかりの深さと広さという観点から, (6) から (9) の構文条件に基づく語順傾向については, かかり先のあいまい性という観点から語順傾向の必然性を説明している. このような語順を規定する要因を計算機上に実現できる形で定量化すると, 音声認識における認識結果の候補の絞り込みのための情報や文生成における基本語順としての情報として使用することができる. このような応用については 4.4 節で述べる. 次節では語順傾向に影響を与える佐伯のかかりの広さと深さについて説明し, かかりの広さに基づく語順の推定モデルを提案する.

4.2　語順の推定モデル

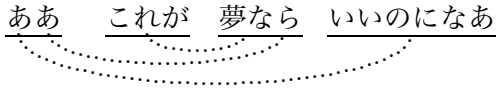
4.2.1　かかりの深さと広さ

日本語の単文は単純化すると次の規則で表現できる.[1] ここでいう後置詞句は 4.1 節で述べた佐伯の補語に対応する.

文 → 後置詞句*, 述語, 時制.
後置詞句 → 名詞句, 助詞.
述語 → 動詞 | 形容詞 | 形容動詞語幹, 「だ」| 名詞句, 「だ」.

ここで “*” は直前の要素の 0 回以上の繰り返しを, “|” は選言を表す. 本論文では助詞としてガ, ヲ, ニ, カラ, ヘ, ト, ヨリ, デの 8 つの格助詞, 述語としては動詞のみを考える. また, 時制, アスペクトは扱わない. すなわち, 我々が扱うのは格助詞によって有標化された名詞句の並びの後に動詞が来るような単文である.

我々の関心は, このように単純化した日本語の単文について, かかり要素 (後置詞句) がどのような順序に並ぶかという問題である. 佐伯はこの問題をかかりの深さと広さという概念を用いて説明している.[15] かかりの深さとは, 直観的にはかかり要素と述部の距離である. たとえば, 「ああ, これが夢ならいいのになあ」という文は次のようなかかり受け関係を持っている.[15]



より文末の受けにかかる方がかかりの深さが深いという. かかりの深い要素のほうが先行しやすいというのが佐伯の観察である. この例では, 「ああ」と「これが」を比べると「いいのになあ」にかかる「ああ」の方がかかりが深く, 先行しやすい. これは非交差の原則とも関係する. また, 4.1 節でも触れたように日本語の述部の語順にはかなり厳格な規則があるので, 述部の要素を語順によって

階層化し, かかり要素が述部のどの階層にかかるかによって, かかりの深さの絶対的な指標を求めることができる. 佐伯はこの指標を実験によって求め, おおむね次のような結果を得ている.

感動語 ＞ 接続語 ＞ 題目語 ＞ 評釈語 ＞ 時間的情態語
＞ 主格補語 ＞ 情態語 ＞ 着格補語 ＞ 対格補語

このように, かかりの深さという概念は述部の階層構造を前提としているので, 述部に動詞のみを仮定する我々の考察対象では使えない.

一方, かかりの広さとは, かかり要素がどの程度述部を限定するかという概念で, 述部を厳しく限定するほどかかりの広さが狭いと考える. たとえば, 「東京カラ」や「京都へ」のようなかかり要素は, 何らかの移動を表す動詞が文末に来ることを予想させるが, 「私ガ」のようなかかり要素は, 動詞の限定がよりゆるやかである. かかりの広さは述部に動詞しか含まない単文にも適用できるため我々の考察対象となりうる.

本節の最初でも触れたように, 英語などの言語と異なり, 日本語では文の主辞である動詞が文末に位置するという大きな特徴がある. 一般に動詞は文全体の意味の中心的な役割を担うので, 言語による情報伝達を考えるとこれは一見不合理なように思える. しかし, 人間は実際に何の不自由もなく日本語を通じて情報を伝達できるし, 相手の発話中に動詞が出現する前に返答することさえできる. これは, 人間が他人の発話を理解する際に, 文の後半を予測しながら理解していることを示唆している. 情報伝達の効率化という観点から考えると, 聞き手が発話の後半を自然に予測できるような発話が望ましいということになる. 佐伯はこのような日本語の特徴をふまえ, かかりの深さが深く, かかりの広さが広いかかり要素から順に提示することが望ましいとしている. いいかえれば, より制約の少ないかかり要素を先に提示することになる. 佐伯はこの主張の根拠として言い誤りの訂正に要するコストをあげている. 以下は佐伯の例である.[15]

(1) しかし, → ×涙を (いや) ○涙が → あふれてきた.

(2) ×涙を → しかし, (いや) ○涙が → しかし → あふれてきた.

(1) は佐伯のいう基本語順であり, (2) はその逆である. 各要素を順次発話するときに, 述部の発話の直前で述部とかかり要素のかかり受けの矛盾に気がついたとすると, (1) の場合, 述部を強く限定するかかり要素 (涙を) が述部に近い位置にあるので, 訂正は 1 要素ですむが, (2) では, 述部と述部を限定するかかり要素 (涙を) の間に, 述部を必ずしも限定しない他の要素 (しかし) があるために 2 要素の訂正が必要となる. このような語順は文の表すことがらの構成要素を周辺的なものから順次列挙していって述語の意味を補完し, 最後に述語で全体の意

味を締めくくるという日本語の求心性[31]とも一致する. 我々の目的は佐伯のかかりの広さを定量化し, それを用いて語順の推定モデルを構築することである. 次項では, 動詞の結合価情報に基づくモデルについて述べる.

4.2.2　結合価行列と優先度行列

日本語の動詞は, 動詞によって必須要素としてとる格助詞の種類, および格助詞と結び付く名詞句の意味的な性質が異なっている. このような側面から文を分析しようとするのが結合価文法である.[27] たとえば, 「歩く」という動詞はガ格の名詞句として動物性の名詞句をとり, ヲ格として場所を表す名詞句をとる, といった分析ができる. このように, 動詞がどのような意味的性質をもつ名詞句をどの格にとるかを規定するものを, その動詞の結合価パタンと呼ぶ. また, 名詞句の意味的性質を表現するために, あらかじめ用意したいくつかの意味素性を用いて表す. 上述の例では, 「歩く」という動詞は [ANI:ガ, LOC:ヲ] という結合価パタンを持つことになる. ただし, ANI, LOC はそれぞれ動物性, 場所性を表す意味素性である.

ある動詞がある格をとらないときは, その格が仮想的な意味素性 NON をとると考えると, 結合価パタンは N_p項行ベクタで表現できる. これを結合価ベクタと呼ぼう. ただし, N_pは格の数で, ここでは $N_p = 8$ である. 結合価ベクタにおける要素の位置は考察の対象としている 8 つの格助詞に対応し, ベクタの要素は意味素性となる. たとえば, 「歩く」は次のような結合価ベクタを持つ.

ガ	ヲ	ニ	カラ	ヘ	ト	ヨリ	デ
[ANI	LOC	NON	NON	NON	NON	NON	NON]

一般に 1 つの動詞は統語的, 意味的な違いによって複数の結合価ベクタをとることがある. 以下, 「動詞」は結合価パタンの違いを考慮したものであるとする. したがって, 各動詞は唯一の結合価ベクタを持つことになる.

ここで, N_v個の動詞について, 各動詞の結合価ベクタを縦方向に連接した N_v行 N_p列の行列 VMを考えよう. これを結合価行列と呼ぶ.

$$VM = \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1N_p} \\ s_{21} & s_{22} & \cdots & s_{2N_p} \\ \vdots & \vdots & \ddots & \vdots \\ s_{N_v1} & s_{N_v2} & \cdots & s_{N_vN_p} \end{bmatrix}$$

次に, 各意味素性について, その意味素性 N_v個を要素とする N_v項行ベクタを考え, これらの行ベクタを意味素性の数 N_sだけ縦方向に連接した次のような N_s

行 N_v列の行列 SMを考える. これを意味素性行列と呼ぶ.

$$SM = \underbrace{\begin{bmatrix} s_1 & s_1 & \cdots & s_1 \\ s_2 & s_2 & \cdots & s_2 \\ \vdots & \vdots & \ddots & \vdots \\ s_{N_s} & s_{N_s} & \cdots & s_{N_s} \end{bmatrix}}_{N_v}$$

意味素性行列を結合価行列に左からかけて得られる N_s行 N_p列の行列を優先度行列と呼ぶ. ただし, 意味素性同士の積は以下のように定義する.

$$s_i \times s_j = \begin{cases} 1 & \text{if } s_i = s_j \\ 0 & \text{otherwise} \end{cases}$$

すなわち, 優先度行列 PMは,

$$PM = SM \times VM = \begin{array}{c} \\ s_1 \\ s_2 \\ \vdots \\ s_{N_s} \end{array} \begin{array}{c} \begin{array}{cccc} p_1 & p_2 & \cdots & p_{N_p} \end{array} \\ \begin{bmatrix} n_{11} & n_{12} & \cdots & n_{1N_p} \\ n_{21} & n_{22} & \cdots & n_{2N_p} \\ \vdots & \vdots & \ddots & \vdots \\ n_{N_s1} & n_{N_s2} & \cdots & n_{N_sN_p} \end{bmatrix} \end{array}$$

優先度行列の各要素 n_{ij}は N_v個の動詞の中で意味素性 s_iと格 p_j の組合せをとることのできる動詞の数を示している. したがって, 優先度行列の要素の値が小さい場合には, その要素に対応する意味素性と格の組を結合価ベクタに含む動詞が少ないということなので, それだけ動詞を限定する力が強いことになる. 逆に優先度行列の要素の値が大きい場合には, その意味素性と格の組は動詞を限定しにくいということになる. 優先度行列が佐伯のいうかかりの広さに関する定量的な尺度を与えるというのが我々の主張である. したがって, 後置詞句の語順を考えると, 優先度行列の要素の値が大きい格と意味素性を持つ後置詞句ほど先行しやすいことが予想できる. ここで注意して欲しいのは我々の議論では, 単にどの格が先行しやすいかだけではなく, 格とそれに結び付く意味素性までも考慮している点である. これにより, より詳細な語順に関する議論が可能になる. これについては 4.3 節でふれる.

ここで, 確率理論の観点からかかりの広さについて考察する. 優先度行列の各要素の値を優先度行列のすべての要素の値の総和で割ると, 動詞の出現確率が一様分布にしたがうと仮定したときの意味素性と格の組合せの出現確率が得られる. すなわち, かかりの広さは動詞の出現確率が一様分布にしたがうと仮定したときの後置詞句 (意味素性と格の組) の出現確率と考えることができる. この立場に立つと, かかりの広さが広い, つまり, 出現確率の高い後置詞句が先行しや

すいという傾向は自然な帰結である．現実には文脈などの影響によって特定の後置詞句が前置されるなどの現象がある．文脈の影響を我々のモデルに取り込む方法に関しては 4.5 節で考察する．

4.2.3　結合価行列の縮退

優先度行列は, 辞書中の, あるいは, 聞き手の意識にのぼっているすべての動詞に関して, どのような後置詞句が入力されやすいかという優先度を与える．これに対して, ある後置詞句が入力された後にどのような後置詞句が入力されやすいかを考えよう．たとえば, 名詞句の意味素性が s_i, 格助詞が p_jである後置詞句が入力された時点での優先度行列はどのようになるかを調べよう．この場合, 格 p_jに意味素性 s_iをもたない動詞が文末に現れることはないので, 結合価行列において p_j列の要素に値 s_iを持たない行は削除することができる．次に p_jの意味素性が決定されたので, 結合価行列の p_j に相当する列も削除することができる．この操作を結合価行列の縮退と呼ぼう．

縮退した結合価行列に意味素性行列を左からかけると, 意味素性 s_iを持つ名詞句と格助詞 p_jからなる後置詞句が入力された時点での優先度行列が得られる．この優先度行列は後置詞句 (s_i, p_j) が入力されたという条件つきのかかりの広さを表している．以下, 同様に新しい後置詞句が入力されるたびに, 結合価行列の縮退と優先度行列の計算をおこない, 次に入力される後置詞句の優先度を計算することができる．

後置詞句が次々に入力されると結合価行列を縮退することができるが, この縮退によって文末に現れる可能性のある動詞の候補も徐々に絞られる．何も入力されない状態ではすべての動詞が候補となる．各動詞の結合価ベクタは固有の数の非 NON 要素を持っており, 後置詞句が入力されるたびに結合価行列の縮退によって非 NON 要素が削除される．初期状態の非 NON 要素の数から現在の非 NON 要素の数を引き, これを初期状態の非 NON 要素の数で割ったものをその結合価ベクタの充足度と呼ぶことにする．すると, 各動詞の持つ結合価ベクタの充足度によって次入力が動詞であると仮定した場合の動詞の優先度を求めることができる．

たとえば, ある段階で結合価ベクタの充足度が 1 の動詞はその直後に動詞が入力されるとしたらもっとも入力されやすい動詞であるといえる．もし, 次の入力が動詞ではなく後置詞句であれば, 結合価行列の縮退によって充足度 1 の結合価ベクタを持つ動詞に相当する行は削除される．なお, 我々のモデルは後置詞句, 動詞の各々については優先度を与えるが, 次入力が後置詞句か動詞かに関する優先度は与えない．

ここで, 結合価行列の縮退と優先度行列の計算に関する計算コストについて調

べよう. 2 つの意味素性の比較にかかるコストを C_{cmp}, 2 つの自然数の加算のコストを C_{add}とすると, 結合価行列の縮退は入力された後置詞句の意味素性と各動詞の該当する格の意味素性を動詞の数 N_vだけ比較すればよいので, $C_{cmp} \times N_v$ のコストで計算できる. また, 優先度行列の計算は, 行列式の計算であるから $N_s \times N_p \times (C_{cmp} + C_{add}) \times N_v$のコストで計算できる. ここで, N_s, N_pはそれぞれ意味素性の数, 格の数である. 意味素性同士の積は前節で定義したように意味素性の比較なので, そのコストは C_{cmp}に等しい. したがって, 新たに後置詞句が入力された時に, 結合価行列を縮退し, 優先度行列を計算するには, $C_{cmp} \times N_v + N_s \times (N_p - 1) \times (C_{cmp} + C_{add}) \times (N_v - N'_v)$ のコストがかかる. ここで, N'_vは結合価行列の縮退によって削除された動詞の数である. これらの定数の中で, 一般的にもっとも大きいのは動詞の数 N_vであるが, 結合価行列の縮退と優先度行列の計算には N_vに対して線形の計算コストしかかからない. また, 後置詞句の入力にともなって結合価行列は小さくなるので計算コストも減少する.

4.3 実験

本節では, IPAL 動詞辞書の結合価パタンの情報を用いて優先度行列を計算し, 佐伯の提案した語順傾向と比較することによって我々の推定モデルの有効性を検討する.

4.3.1 IPAL 動詞辞書

IPAL 動詞辞書[53]は計算機による日本語処理のために作成された動詞辞書で, 基本的な和語動詞 861 語に関する統語的, 意味的情報を含んでいる. ここでは, 実験に必要な情報についてのみ述べる.

IPAL 動詞辞書の見出し語はひらがな表記で, 同音意義語は同音意義語番号を見出し語にふることによって区別している. 861 語という数は同音意義語を区別したときの数である. 一般にひとつの動詞が複数の結合価パタンをとることができ, 結合価パタンによって意味が異なる場合がある. IPAL 動詞辞書ではサブエントリ番号によって結合価パタンおよび意味の違いを区別している. IPAL 動詞辞書は 3370 のサブエントリを持っている. まとめると, IPAL 動詞辞書の各見出し語は次のような階層を持つ.

見出し (ひらがな表記) > 同音意義語番号 > サブエントリ番号

IPAL 動詞辞書で扱う格は, ガ, ヲ, ニ, カラ, ヘ, ト, ヨリ, デ, Φである. ここで, Φは格助詞を必要としないはだか格を表す. 本実験では, はだか格以外の

8 つの格助詞について考慮する. それぞれの格には, その格がとりうる名詞句の意味素性が割り当てられている. 意味素性は表 4-1に示す 20 個からなる. このうち---は文要素を表す. ---は「〜ト思う」などのようにト格にしか現れない. 正確にはこれらの意味素性には 2 レベルの階層があり, CON は ANI から PRO の上位素性, ABS は ACT から QUA までの上位素性となっているが, 簡単のため実験ではすべての素性を同一に扱った.

表 4-1　IPAL の意味素性

略号	意味素性名	例
CON	具体物	
ANI	動物	鼠, 牛, 虎
HUM	人間	父, 友達, 先生
ORG	組織・機関	政府, 企業, 大学
PLA	植物	せり, なずな
PAR	生物の部分	手, 足, 鼻
NAT	自然物	海, 山, 川
PRO	生産物・道具	はさみ, 自動車
PHE	現象	雨, 風, 匂い
ABS	抽象物	
ACT	動作・作用	散歩, 勉強
MEN	精神	心, 喜び, 苦悩
LIN	言語作品	小説, レポート
CHA	性質	優しさ, 美しさ
REL	関係	原因, 結果, 根拠
LOC	空間・方角	西, 左
TIM	時間	月曜, 朝
QUA	数量	3 人, 10kg, 10m
DIV	制限緩やか	
---	文要素	「太郎が死んだ」

格, 意味素性は交替可能な場合があり, その場合は交替可能な要素を “/” で区切ってある. たとえば, 「あおぐ」の同音意義語番号 001, サブエントリ番号 002 の結合価パタンは,

[HUM/ORG:ガ, HUM/ORG:ニ/カラ*, ABS:ヲ]

であるが, 交替可能な要素をすべて展開すると次の 8 通りの結合価パタンになる.

[HUM:ガ, HUM:ニ*, ABS:ヲ]　[HUM:ガ, ORG:ニ*, ABS:ヲ]
[HUM:ガ, HUM:カラ*, ABS:ヲ]　[HUM:ガ, ORG:カラ*, ABS:ヲ]
[ORG:ガ, HUM:ニ*, ABS:ヲ]　[ORG:ガ, ORG:ニ*, ABS:ヲ]
[ORG:ガ, HUM:カラ*, ABS:ヲ]　[ORG:ガ, ORG:カラ*, ABS:ヲ]

また, “*” は任意要素を表し, 結局, このエントリは 10 種類の結合価パタンを持つことになる. ただし, この実験では任意要素と必須要素の区別はおこなわない. すなわち, “*” は単に無視する.

4.3.2　結合価行列と優先度行列

まず, IPAL 動詞辞書から見出し語, 同音意義語番号, サブエントリ番号, 結合価パタン情報を抽出する. 前項で述べたように, 結合価パタンの交替要素について結合価パタンを展開し, 展開した結合価パタンに 0 から順に結合価パタン番号をふる. したがって, 各動詞は統語的, 意味的情報の違いも含めて, 見出し語, 同音意義語番号, サブエントリ番号, 結合価パタン番号の 4 つ組で表現できる. ただし, ここでいう「動詞」は 4.2 節で述べたように統語的, 意味的な違いも考慮したものである. こうして 8829 個の動詞が得られるが, ここで, ガ－ガ, ニ－ニのように同じ格を複数持つ結合価パタンを持つ動詞については結合価ベクトルが構成できないので削除する. これを除くと 8062 個の動詞が残る.

これらの動詞のいくつかは同一の結合価パタンを共有している. 異なる結合価パタンの数は 2225 である. 表 4-2は各結合価パタンを持つ動詞の数の分布を表している. 我々のモデルでは, 動詞の推定に関して同じ結合価パタンを持つ動詞は区別できない. しかし, 表 4-2より結合価パタンが決まれば候補の動詞がほとんどの場合 20 以下に抑えられることがわかる.

表 **4-2**　動詞数別の結合価パタンの頻度

動詞の数			頻度
100	～		3
80	～	100	4
60	～	80	5
40	～	60	16
20	～	40	35
	～	20	2162

以上のようにして得られた 8062 の動詞から 4.2 節で述べたように, 結合価ベクタ, 結合価行列を構成する. IPAL 動詞辞書の意味素性の数は 20 なので 20 行 8062 列の意味素性行列を作り, それを結合価行列に左からかけると 20 行 8 列の

優先度行列を得る．この優先度行列の要素の値について降順に整列したときの上位 20 の意味素性と格助詞の組を表 4-3に示す．

表 **4-3**　初期状態の優先度行列の要素の一部

順位	意味素性：格	動詞数
1	HUM:ガ	3898
2	ABS:ヲ	1654
3	ORG:ガ	1515
4	LOC:ニ	954
5	CON:ヲ	756
6	ABS:ニ	660
7	ABS:ガ	603
8	HUM:ニ	600
9	LOC:カラ	588
10	HUM:ヲ	566
11	PRO:ヲ	501
12	LOC:ヲ	497
13	PRO:デ	479
14	ANI:ガ	436
15	PRO:ガ	422
16	LOC:ヘ	401
17	ORG:ニ	397
18	CON:ガ	350
19	ACT:ヲ	345
20	ABS:デ	310

4.3.3　格の優先順序

かかりの広さが広い, すなわち優先度行列の要素の値が大きいかかり要素ほど先行するという佐伯の仮説にしたがえば, かかりの広さという観点から各動詞について最適な後置詞句配置が決定できる．各動詞について以下の手続きにより, 最適な後置詞句の配列を計算する．

(1) 動詞を 1 つ取り出し, 以下の処理を結合価パタンのすべての非 NON 要素についておこなう．

(2) その動詞の結合価パタンに含まれる後置詞句 (意味素性と格の組) の中から現在の優先度行列で最大の値を持つものを選択する．

(3) 選択した後置詞句に基づいて結合価行列を縮退し, 新しい優先度行列を計算する.

(4) 新しい優先度行列と結合価パタン中の残りの後置詞句について (2) から繰り返す.

このようにして得られた各動詞の最適な後置詞句配列から 2 つの格の優先関係を計算する. たとえば, 結合価パタンの最適配列は次のような形式で得られる.

443　28　HUM:ガ HUM:ニ ACT:ヲ

ここで, 最初の数字は結合価パタンの識別番号, 次の数字はその結合価パタンを持つ動詞の数, そして残りが後置詞句の配列である. この項目からは, ガ $>$ ニ, ガ $>$ ヲ, ニ $>$ ヲという優先関係にそれぞれ 28 点ずつを与える. このようにして得られた結果を表 4-4に示す. 表 4-4から次のような格の優先関係が読みとれる.

ガ $>$ ヲ $>$ ニ $>$ カラ $>$ デ

しかしながら, 1 つの格は一般に複数の役割を持つので各々の格がどのような意味素性の名詞句をとるときにどのような優先関係を持つのかという点を含めて議論しないと意味がない. そこで, 4.1 節で紹介した佐伯の語順に関する観察に沿って実験結果を検討する.

表 **4-4**　格の優先関係

X	Y	X > Y	X < Y	X	Y	X > Y	X < Y
ガ	ヲ	4384	1266	ニ	カラ	395	45
ガ	ニ	2824	804	ニ	ヘ	0	9
ガ	カラ	986	259	ニ	ト	56	5
ガ	ヘ	475	106	ニ	デ	300	77
ガ	ト	409	0	カラ	ヘ	287	13
ガ	ヨリ	32	1	カラ	ト	0	2
ガ	デ	1386	105	カラ	デ	102	98
ヲ	ニ	1408	790	ヘ	デ	35	46
ヲ	カラ	601	284	ト	デ	18	41
ヲ	ヘ	305	113	ヨリ	デ	0	17
ヲ	ト	204	31				
ヲ	ヨリ	11	0				
ヲ	デ	836	281				

4.3.4　検討

以下, 表 4-4の結果を 4.1 節で述べた佐伯の成分条件に基づく語順傾向に照らして検討する.

(傾向 **1**) 位格は他の格に先行する

佐伯は位格の格助詞としてニ, デ, カラ, ヲを考察の対象としている. 後述する (傾向 3) とも関係するが, ガに先行できるのは一般に位格だけである. したがって, ここでは, ガと位格の関係について調べる.

まず, ニについて, ニがガに先行するのは動詞が所動詞であり, 主格がモノである傾向が強く, 逆にガが先行するのは動詞が能動詞であり, 主格がヒトである傾向が強いという佐伯の観察がある. 我々の手法では, 格のとる意味素性を手がかりにこの分析をおこなうことができる. 実際, ニ $>$ ガ, ガ $>$ ニとなるそれぞれの場合について意味素性の分布を調べると, ニ $>$ ガの場合, ガの意味素性は ABS, PRO, CON, PHE, ANI がいずれも 10%以上を占め, これら 4 つの合計で 72%となる. ヒトも含まれる可能性のある ANI (動物) は単独では約 13%を占めるに過ぎない. また, この時のニは LOC と ABS で 77%以上を占める. 一方, ガ $>$ ニの場合, ガの意味素性は HUM と ORG で 86%以上を占めている. 格がとる名詞句の性質という点では佐伯の観察と同様の結果が得られた.

次にデについて考えよう. デはニに比べるとガに先行しにくいという佐伯の主張は, 表 4-4のガとニ, ガとデの優先関係の比率に現れている. ガ $>$ ニとなるのは, 4384 : 1266 (3.5 : 1) であるが, ガ $>$ デとなるのは, 1386 : 105 (13.2 : 1) という比率である. また, 意味素性についてニと同様に調べてみると, ガ $>$ デの場合のガは HUM と ORG で 82%を占める. また, この時のデは PRO と ABS で 50%以上となり, ガ $>$ デの場合はデが具格を表す場合が多いと推定できる. 逆に, デ $>$ ガの場合にガが HUM をとる例はなく, ANI が 24%をとるだけである.

カラについて, 佐伯はガ $>$ カラとなる場合は, 着格 (ニ, ヘ) も同時にとりやすいが, カラ $>$ ガの場合は着格をとりにくいという観察を述べているが, 我々の実験結果からはこのような観察は得られなかった.

最後に, ヲはガに先行しにくいという傾向は表 4-4から読みとれる. 特に, 意味素性に LOC をとる場所の位格については, ガに先行するものはわずか 15%しかなかった.

(傾向 **2**) トキの位格はトコロの位格に先行する

位格内の優先順位については, トキとトコロの位格を両方含む結合価パタ

ンがないために評価できなかった. これは, 結合価パタンが主に必須格を対象としているのに対して, 位格は必ずしも動詞の必須格にならないためであると考えられる.

(傾向 **3**) ガは位格を除く他の格に先行する
ガの優先性は表 4-4においていずれもガが他の格に先行していることから読みとれる.

(傾向 **4**) 与格のニは対格のヲに先行する
ニ $>$ ヲ, ヲ $>$ ニそれぞれの場合について, ニのとる意味素性について調べたところ, ニ $>$ ヲの場合はニ格に与格補語となりやすいと思われる意味素性 HUM をとる場合が全体の 25%であるのに対し, ヲ $>$ ニの場合は 16%であった. この差は 9%であり, この実験からは (傾向 4) について明確な傾向は観察できなかった.

(傾向 **5**) 発格 (カラ) は着格 (ニ, ヘ) に先行する
意味素性を LOC に限定して, カラとニ, への組合せの優先関係を調べたところ, カラとニについては LOC : ニ $>$ LOC : カラが 98%を占め, カラとへについては LOC : カラ $>$ LOC : へが 100%を占めていた. この結果からは, 発格が着格に先行するとはいえない. 発格, 着格はかかりの広さという観点からはどちらも同等であり, その順序関係は別の制約によって規定されると考えられる.

4.4 推定モデルの自然言語処理への応用

我々の推定モデルは日本語の後置詞句の並びと動詞からなる単文について, 文要素 (後置詞句または動詞) が次々に入力されると仮定して, 次要素の優先度を計算するものである. この点においては, 入力の意味解釈を漸次的におこなうアプローチ[2,83]と同様であるが, 我々のモデルは各要素が入力された時点で, 意味素性と格の組合せに対する優先度という形で次の要素を推定できる点が特徴である. 以下, この推定モデルの自然言語処理への応用について考察する.

4.4.1 名詞の語義のあいまい性解消

日本語の文の統語的, 意味的な主辞は文末の動詞であり, その動詞にかかる名詞句の意味によって動詞の意味, ひいては文全体の意味が決まる. 我々の推定モデルでは, 4.2 節で述べたように動詞の意味のあいまい性に関しては, 動詞の結合価ベクタの充足度という尺度で優先度を与えることができる. しかし, 一般には名詞句の意味にもあいまい性があるので, 理想的には名詞句と動詞が互いに相

手の意味のあいまい性を解消するようなモデルが望ましい．我々の推定モデルは後置詞句が入力された時点で, その格をとる名詞句の意味素性の優先度を与えることができる．したがって, 名詞句の意味のあいまい性を部分的に解消していることになる．

また, すでに入力された後置詞句に含まれる名詞句が複数の意味素性を持つ場合を考えよう．入力が進むにつれ, 動詞の候補が絞り込まれ, 残った動詞候補の結合価パタンの制約からすでに入力された名詞句のあいまい性が漸次的に解消できる．たとえば, 最初に後置詞句 (np_1, p_1) が入力されたとして, np_1の意味素性が s_1, s_2, s_3であったとしよう．この時点で優先度行列により s_1, s_2, s_3の間に優先度を付けることができる．さらに入力が進み, i 番目の後置詞句 (np_i, p_i) が入力された時点で, 候補として残っている動詞の p_1格に付く意味素性の分布から (np_i, p_i) が入力された時点での np_1の意味素性の優先度を計算することができる．これは, まさに動詞と名詞句が互いに相手の曖昧性を漸次的に解消していることを意味している．

4.4.2　音声認識における候補の絞り込み

音声認識では認識結果の候補をさまざまな情報を利用して絞り込むことが重要となる．我々のモデルが与える優先度情報は候補を絞り込むための情報のひとつとして利用できる．特に日本語の音声認識では, 名詞句などの自立語に比べ, セグメント数の小さい助詞などの附属語の認識率が低いという傾向がある.[4] 現在の日本語の言語処理, 特に意味解析が助詞の情報に大きく依存していることを考えると, 音声から意味構造を抽出する音声理解のためには, 助詞の認識率を向上させる必要がある．我々の推定モデルは名詞句の意味素性と格の組合せに対して優先度を与えるので, 名詞句の認識結果を手がかりに助詞を推定することもできる．

4.4.3　文生成における語順の決定

文生成の立場からすると, 出力すべき内容はすべて与えられているので, 文を解析する場合と異なり名詞句や動詞の意味のあいまい性は存在しない．生成ではどのような順序で後置詞句を並べるかという語順が重要な問題となる．我々の推定モデルは佐伯のかかりの広さという概念に基づき, 日本語ではかかりの広さの広い要素が先行するという仮説の上に構築したものである．したがって, 4.3 節で格の優先順位を算出するときに用いた手続きにより, 出力すべき動詞の結合価パタンから最適な後置詞句配列を計算することができる．もちろん最終的な語順を決定するには, 文脈の焦点や聞き手のモデルなど, さまざまな情報も考慮しなければならないが,[72] 我々のモデルが与える情報は文を生成するとき

の語順に関するもっとも基本的な情報として利用することができる.

4.5 モデルの拡張

本節では, これまでの議論で無視していた文脈情報, 省略語, 係り助詞などを我々のモデルでどのように扱うかについて, 基本的な考え方を述べる.

4.5.1 文脈情報の扱い

4.2 節で導入した結合価行列では, 各動詞がすべて等確率で出現することを前提としており, この前提に基づいて優先度行列を計算している. しかしながら, 実際には文脈的な情報や対象領域の性質から動詞の出現確率には偏りがあると考えるのが現実的である. 我々の推定モデルにこのような文脈あるいは対象領域による動詞の偏りを導入するひとつの考え方として, 各動詞, すなわち結合価行列の列に重みを付加し, 優先度行列を計算する時にその重みを考慮した計算をおこなうことが考えられる. 具体的には, 意味素性同士の積を以下のように再定義する. ここで w_vはその意味素性を持つ動詞の重みである.

$$s_i \times s_j = \begin{cases} w_v & \text{if } s_i = s_j \\ 0 & \text{otherwise} \end{cases}$$

さらに, 文脈によっては焦点や題目化によって特定の要素が文の前方に位置しやすいことも考えられる. このような名詞句の前置されやすさに関する偏りも動詞の出現頻度の偏りと同様に, 意味素性行列の各行に重みを付加することによってモデルに導入することができる. 具体的には, 意味素性同士の積を以下のように再定義する. ここで w_sは当該の意味素性の重みである.

$$s_i \times s_j = \begin{cases} w_v \times w_s & \text{if } s_i = s_j \\ 0 & \text{otherwise} \end{cases}$$

4.5.2 省略語の推定

実際の文では, 文要素の省略が頻繁に起こる. 省略が起こると本推定モデルで高い優先度で推定されている要素が入力されないで, より低い優先度の要素が入力されることになる. ある段階で高い優先度で推定された要素はその後の段階でも優先度の上位に位置する傾向がある. したがって, 入力と各時点での優先度の履歴を調べれば, 過去に高い優先度で推定されているにも関わらず, 実際には入力されていない要素は省略された可能性があると判断できる. しかしながら, 久野が指摘しているように, 省略は文を越えた談話による制約に大きな影響を受けるので,[12] 具体的にどのような判断基準で省略要素を推定するかは実際のデータをもとにさらに詳しく調べる必要がある.

4.5.3　係り助詞の扱い

これまでの議論では格助詞のみを考察の対象としてきたが, 日本語の解析では, 係り助詞, 特に頻繁に使用される「ハ」は無視できない存在である. 三上はハの機能を次のように分類している.[19]

$$\begin{cases} \text{代行} & \begin{cases} 1. & \text{連用 (ガ, ヲ, ニ)} \\ 2. & \text{連体 (ノ)} \end{cases} \\ \text{先行} & \begin{cases} 3. & \text{名詞の反復} \\ 4. & \text{状況や結果の提示} \\ 5. & \text{特殊語法} \end{cases} \end{cases}$$

このうち我々が関心があるのは, 連用の代行機能をはたすハである. このハにより題目化された後置詞句は格に関して無標化されるので, 結合価行列を縮退する際に, どの格の列を削除するかが一意に定まらない. ハを持つ後置詞句は題目化によって先行するのであり, 先行の原因はかかりの広さとは直接関係ないので, 名詞句の意味素性と優先度行列からハによって代行された格助詞を推定するのは適切でない. この点において音声認識への応用で述べたような助詞の推定とは状況が異なっている. この問題に対するひとつの解決策として, 結合価行列の縮退において列 (格) は削除せず, 行 (動詞) の削除に関してはハをともなう名詞句の意味素性とハが代行する可能性のある格助詞 (ガ, ヲ, ニ) の組を持つ行以外を削除することが考えられる. すなわち, これは格助詞に関する決定を遅延していることになる. 入力が進むにつれハに代行される格助詞が明示的に現れることもあるので, ハがどの格助詞を代行しているかは徐々に明らかになる.

4.6　要約

本章では佐伯の提案したかかりの広さという概念を用い, 後置詞句の並びと動詞という構造を持つ日本語の単文の次入力を推定するモデルを提案した. 基本的な考え方は, 意味素性と格の各組合せについて, その組合せをとりうる動詞の数を動詞の結合価パタンから計算し, もっとも多くの動詞がとる組合せがかかりの広さが広いと考え, 文の前方に配置されるという仮定に基づいている. また, 実際に IPAL 基本動詞辞書から結合価パタンを抽出し, IPAL 中の各動詞についてかかりの広い順に後置詞句を配列し, その語順傾向が佐伯の分析に一致することを確かめた. このモデルは音声認識における認識結果の候補の絞り込み, 名詞句の意味のあいまい性の解消, 文生成における基本語順の決定などに利用できる.

第 5 章

概念階層における視点の役割

本章と次章では自然言語処理に有用な概念体系について述べる. まず, 本章では概念体系における視点の重要性[34,37]を述べ, 次章では個々の概念をどのように設定するかについて論じる.

5.1 背景

計算機で自然言語を処理するためには, 人間が持っていると思われる常識を計算機上に表現することが必要である. しかしながら, 常識を明確に定義することが難しい上, その量が膨大なものであるなど問題が山積している. 人間の常識は大量で多種多様な概念が複雑に絡み合っているため, 一部の知識を他の知識とは独立に切り出すことが難しい. さらに, 大量の知識でないと常識とはいえない. このように質の問題とともに量の問題も解決しなければならない.

近年, 自然言語処理の分野において, 大容量な常識的知識の重要性が改めて認識され, 実際に計算機上にこれを構築する試みもなされている .[40,76] これらの試みでは, 常識を構成する大量の概念と概念間の関係 (概念体系) を計算機上に表現することが主な目的である. この概念体系が自然言語処理に重要な役割を果たすことはすでに多くの研究者によって指摘されている.[6,32,43] 特に, 概念間の上位/下位 (isa) 関係や概念間の部分/全体 (hasa, part-of) 関係に基づいて体系化された概念体系は自然言語処理に有用である. このような概念体系はシソーラスと呼ばれることもある. たとえば, Roget のシソーラス,[59] 分類語彙表,[50] 類義語新辞典[30]などがあるが, これらはもともと人間向けに作られたものであり, 概念というよりむしろ表層の語の分類体系となっている. また, 概念 (語) 間の関係の根拠が不明確であることが多く, 自然言語処理用の知識としては不十分である.[43]

本章では, 概念間の上位/下位関係に基づいた概念体系について議論する. 以

下では, このような概念体系を概念階層と呼ぶ. ただし, ここでは概念という語を 5.2 節で定義するように非常に限られた意味で使っている.

概念階層を自然言語処理に利用する場合は, 処理のどこかの段階で, 表層の文に現れる語と概念階層中の概念との対応をとらなければならない. しかし, 一般に語と概念の対応関係はその語を取り巻く文脈から十分な情報が得られないと一意に決めることはできず, あいまい性を残すことになる. 5.3 節ではこの種のあいまい性, 特にある概念が性質の異なる複数の上位概念を持つ場合に生じるあいまい性について考察する. そして, どの上位概念によってその概念を特徴づけるかということが重要であることを指摘し, 視点という考え方を導入する.

知識表現において視点を扱ったこれまでの研究[5,17,47,57]では, 概念を下位概念に分類する基準として視点を導入していた. ここでは, このような視点を下位方向の視点と呼び, 我々が提案する上位方向の視点と区別する. 5.3 節では上位方向の視点の重要性について述べ, 5.4 節ではこれを表現するための道具として概念の視点表現を提案する. 視点表現を使って概念を表現することにより, 複数の視点を持つ概念や特定の視点から見た概念を自然に表現することができる. また, 視点表現で表現された概念の単一化は文脈解析における照応処理などに応用することができ, 自然言語処理では重要な役割を果たすことを指摘する. 5.5 節ではこの単一化の定義を与える. 最後に, 5.6 節では視点表現の単一化の自然言語処理への応用について述べる.

5.2 概念階層

本節では, 上位/下位関係を唯一の概念間の関係とする概念階層について述べる.

5.2.1 概念階層の定義

まず, 概念階層を以下のように定義する.

定義 5-1 概念
性質の集合を概念という. 性質は属性と属性値の対である. 概念を参照するために概念に名前をつけることがある. 以後, 概念を表す場合は概念名を括弧で囲み, その前に$*$を付ける.

定義 5-2 個体
ある概念の性質を継承する実体をその概念の個体いう.

定義 5-3　複合概念
複数の概念を “,” で区切り, “{” と “}” で囲むことによって { } 内の概念の持つ性質をすべて持つ概念を表現する. この記法で表現した概念を複合概念という.

定義 5-4　概念階層
概念階層は概念の集合 C と概念間を結ぶ有向リンクの集合 L からなるループを含まない有向グラフ ⟨C, L⟩ である. 概念 a から概念 b に向かう有向リンクが存在する時, 概念 a は概念 b の性質を継承できる. このリンクを直接リンクといい, a < b と表す. また, 概念 a から概念 b へ 1 個以上の直接リンクを経て到達可能な時, この直接リンクの系列を間接リンクと呼び, a ≪ b と表す. いずれの場合も, 概念 b を概念 a の上位概念, 概念 a を概念 b の下位概念という.[16]

5.2.2　排他的概念

ある概念が複数の下位概念を持つ時, 下位概念が互いに排他的な関係になることがある. 概念階層には, このような概念間の排他的な関係も記述しなければならない.[17]

定義 5-5　排他的概念
複数の概念が共通の属性で属性値が異なるような性質を少なくとも 1 つ持つ場合には, これらの概念は互いに排他的な概念であるといい, x ⊗ y と表す. 複合概念の { } 内の概念は互いに排他的であってはならない.

たとえば, ∗(人間) の属性「性別」は値として「男」または「女」をとるとしよう. 「性別」の値に基づいて∗(人間) は∗(男) と∗(女) の 2 つの排他的な下位概念を持つと考えられる. この場合, ∗(人間) は属性「性別」によって 2 つの下位概念 ∗(男) と∗(女) に分類されたという.

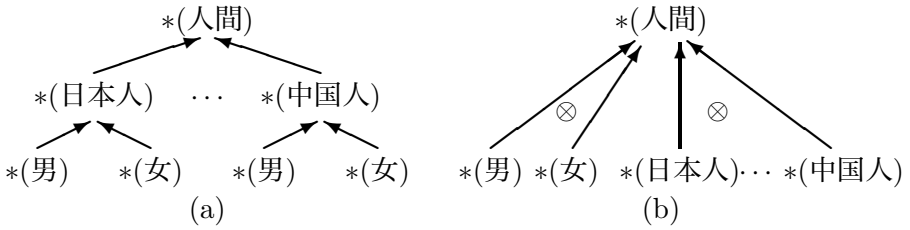


図 5-1　下位概念の分類例

概念の属性により下位概念を分類する場合, 次の 2 つの方法ある (図 5-1).

(a) 各概念において下位概念の分類に使う属性を 1 つに限る方法

(b) 各概念において下位概念の分類に使う属性を複数許す方法

(a) の方法では, 概念の下位概念を必ず 1 つの属性によって分類するため, 分類の基準となる属性の情報を明示しなくても, 下位概念同士はすべて排他的な関係であることがわかる. このような概念階層は弁別ネットワークとなる.[69] 一方, (b) の方法では複数の属性によって概念が下位概念に分類されるため, 排他的な下位概念を明示的に示す必要がある.[17,47]

(b) では, 下位概念が互いに排他的であるということをリンクの間に⊗を置いて表す. ⊗には概念を下位概念に分類するための基準となる属性が対応している. たとえば, (b) において∗(人間) を∗(日本人) や∗(中国人) に分類している基準は, ∗(人間) の持つ属性「国籍」の値の違いによると考えられる.

概念階層を考える上で排他性を明示的に記述する必要がないという点では (a) の方法が優れているが, この方法には次の欠点がある.

- 概念を分類する基準となる属性が複数ある場合, 分類のために適用する属性の順番を決めなければならないが, この順番を決めるのは難しい. たとえば, 図 5-1 (a) において属性「国籍」が属性「性別」よりも先に (階層の上のほうで) 適用されなければならないという根拠はない.

- 概念を 1 つの属性についてしか分類できないために階層の深さが深くなる. すなわち, 性質が垂直方向に分散される.

以下の議論では, 図 5-1 (b) の分類方法をとる.

これまでの知識表現の研究では, このような分類の基準となる属性を perspective,[57] 視点,[5,17] 見方[47] と呼んでいるが, 我々はこれを下位方向の視点と呼ぶ. これに対して, 概念が複数の上位概念を持つ場合に, どの上位概念によってその概念を特徴付けるか, という意味での視点を考え, これを上位方向の視点と呼んで, 下位方向の視点と区別する. このような視点の考え方は石崎らの提案するもの [24] と似ているが, 彼らの視点が概念から導出した個体を対象とし, 動的に作られるのに対し, 我々の提案する視点は概念を対象とし, 概念階層中に静的に存在する多くの視点 (多重継承) を動的に選択するという点で異なっている. 以下, 特に断らない限り上位方向の視点という意味で視点という語を使う.

5.3 語のあいまい性と概念階層

自然言語の意味処理において, 語のあいまい性の解消は大きな問題の 1 つである. 語のあいまい性は語が指し示すことのできる概念が複数あることに起因して

いる. したがって, 語のあいまい性を解消することは, 文章中に現れる語に対応する概念を決定することに他ならない. 本節では語のあいまい性を大きく 2 つに分類し, それぞれについて概念をどのように設定すべきかについて考察する.

5.3.1　語の多義性によるあいまい性

語の多義性によるあいまい性とは, 同一の語がまったく異なる概念を指し示すことによって生じるあいまい性である. たとえば, 次の例文において “さくら” (以下, 語を表す場合は 2 重引用符で囲む) という語はいずれも異なる概念を指している.

(5-1) 太郎は庭に桜を植えた. (桜の木)

(5-2) 先日九州で桜を食べた. (馬肉)

(5-3) あの男はさくらにちがいない. (人間)

概念階層において, 異なる概念は別概念として表現できなければならない. これらの例文において, それぞれの “さくら” が指す概念は同一の個体となりえないので各概念は別の概念として設定すべきである.

多義語に対応する概念を設定する場合に既存の国語辞典などが参考になる. たとえば, 三省堂の新明解国語辞典[11]で「さくら」の項目を引くと,

1. (さくら)
   (a) 金をもらって、劇場の客席から俳優に声をかけてほめる人.
   (b) 縁日や盛り場の露店などで、客のふりをして品物をほめたり進んで買ったりしてお客の買い気をそそる役をする・人（仲間）.
2. (さくら) [桜]
   (a) 堤・道わき・公園・庭に植える落葉高木. 春、一面に美しく咲く薄紅色の花は国花。材は建築・家具に使う. 〔バラ科〕
   (b) 〔←桜色 1448〕薄い赤色。淡紅色.
   (c) 〔←桜肉 1451〕「馬の肉」のえんきょく表現. 〔桜色をしているから〕
   (以下, 派生語が続く)

となっている. 新明解国語辞典では例文 (5-3)の “さくら” は他の 2 文の “さくら” とは別項目になっているが, 例文 (5-1)と例文 (5-2)の “さくら” は同一項目

の下位分類として区別されている. これは馬の肉という意味の “さくら” の語源が, 馬の肉が桜色をしているという事実に基づいているためである. しかしながら, この 2 つの “さくら” に対応する概念の個体は同一のものとなりえないので別概念として設定すべきである. また, 岩波書店の広辞苑[28]では “さくら” の項目はただ 1 つである. このように, 国語辞典は多義性のある語に対応する概念を設定する時に役立つが, 国語辞典の内容の構造から機械的に概念を設定するのでは十分でないことがわかる.

5.3.2 視点によるあいまい性

視点によるあいまい性とは, 1 つの概念を異なる視点から眺めることによって異なる概念と考えることができるというあいまい性である. このようなあいまい性は, ある概念が互いに性質の異なる複数の上位概念を持つ場合に起こる. 次のような例文を考える.

(5-4) この野菜は害虫に強い. (植物の野菜)

(5-5) 太郎は野菜をたくさん食べる. (食物の野菜)

たとえば, 例文 (5-4)の “野菜” は∗(植物としての野菜) を, 例文 (5-5)の “野菜” は∗(食物としての野菜) に対応すると考えられる. これらを別概念として設定することももちろん可能であるが, 適当な文脈を与えれば両概念の個体が同一のものとなりうるから, 必ずしも別概念とする必要はない. 逆に, 文脈によってはどちらか一方の概念の個体しか指せない場合もあるから, それぞれ別概念の個体としても表現できなければならない.

1 つの方法は∗(野菜) を∗(野菜 1) (植物としての野菜という概念) と∗(野菜 2) (食物としての野菜という概念) に分けて, 一方の概念の性質しか継承しない個体はその概念だけを継承し, 両方の概念の性質を継承できる個体は多重継承で両方の概念を継承することが考えられる. しかしながら, 概念を分割するとその下位概念も分割しなければならなくなり, 概念の数が多くなるという問題が生じる. たとえば, ∗(野菜) を∗(野菜 1) と∗(野菜 2) に分割すると, ∗(野菜) の下位概念である∗(にんじん), ∗(レタス), ∗(かぼちゃ), ... についてもすべて, ∗(にんじん 1), ∗(にんじん 2), ∗(レタス 1), ∗(レタス 2), ... に分割する必要がある. さらに, ∗(にんじん) に下位概念があれば, それらもすべて分割しなければならない. また, 概念を分割すると∗(野菜 1), ∗(野菜 2) が共通に持つ性質を両方の概念に記述する必要があり, 不経済である.

この問題は, 概念を考える時には様々な視点があるということに起因している. たとえば, ∗(野菜) も植物としてみた野菜と食物としてみた野菜を考えることができ, 文脈によってその視点が変化する. このような動的な視点の変化を静

的な概念階層で表現しようとするところに問題がある. したがって, 概念階層中では様々な視点を含む概念を静的な形で表現し, 概念階層を使って推論をおこなう時に視点を含めた表現ができるような枠組を与えることが必要である. 概念が様々な視点を持つことは, 概念階層中では概念が複数の上位概念を持つことに対応する. また, 特定の視点からみた概念は, 上位概念から注目している概念へ至る経路を含めて概念を表現することに対応する.

“野菜” の例について図 5-2の概念階層を考えよう. ここで, ∗(野菜) は上位概念として∗(植物) と∗(食物) をもつと仮定している. ∗(野菜) を∗(植物) から∗(野菜) へ至る経路を含めて考えれば, 植物としての野菜という概念を表すことができるし, ∗(食物) からの経路を含めて考えれば, 食物としての野菜という概念を表すことができる. 文中の “野菜” がどちらの概念に対応するか, 文脈上決定できないような場合は, これを∗(野菜) と対応付ければあいまい性を含んだ表現ができる.

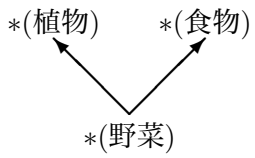


図 **5-2**　∗**(野菜)** の上位概念

このように上位概念からある概念へ至る経路を含めて概念を眺めることは, すなわち視点を考慮して概念を解釈していることになる. 概念階層を作成, あるいは利用する時にこのように様々な視点を表現できることが望ましい. 5.4 節では概念階層において視点を表現するための道具について考察する.

5.4　概念の視点表現

本節では, 概念階層において視点を表現するための道具として概念の視点表現を提案する. 視点表現は, 概念を上位概念からの継承経路を含む形で表現する方法であり, 特に注目している概念が複数の上位概念を持つときに意味を持つ. 視点表現を表現するためのオペレータとして “\” を使う.

たとえば, 図 5-3において, ∗(野菜) はすべての上位概念の性質を継承するが, 視点表現∗(野菜)\∗(植物) は上位概念の∗(植物) の性質しか継承しない概念を意味する. すなわち, 植物という視点からみた野菜という概念を表現していることになる. 視点表現には複合概念を含むことができる. また, 複合概念は展開できる. たとえば, ∗(野菜)\{∗(食物), ∗(植物)} は∗(食物) と∗(植物) の性質しか継承しない野菜の概念を表わす. また, ∗(野菜)\ {∗(食物), ∗(植物)} と {∗(野菜)\∗(食物), ∗(野菜)\ ∗(植物)} は等価である.

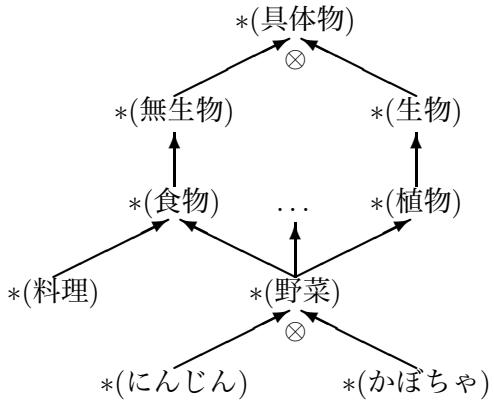


図 5-3　概念階層の一部

視点表現の形式的な定義を以下に示す. まず, いくつかの記号を定義する.

$Prop(x)$	概念 x が持つすべての性質
$SelfProp(x)$	概念 x 自身が持つ性質
$InheritProp(x)$	概念 x が継承によって持つ性質

ただし, $x_i < x_j$という関係があるとき, 上位概念が x_j であるという性質は $InheritProp(x_i)$ に含まれるものとする. ここで,

$$Prop(x) = SelfProp(x) \cup InheritProp(x)$$

が成り立つ.

定義 5-6　視点表現で表わされた概念の持つ性質
視点表現で表わされた概念が持つ性質を以下のように定義する.

$$Prop(x_1\backslash x_2\backslash \ldots \backslash x_n) = \bigcup_{i=1}^{n-1} SelfProp(x_i) \cup Prop(x_n)$$

唯一の上位概念を持つ概念については, 視点表現で表現しても, その概念の持つ性質は変わらない. たとえば, 図 5-3において∗(にんじん) と∗(にんじん)\∗(野菜) は等価である. すなわち,

定義 5-7　視点表現の縮退
視点表現 $x_1\backslash x_2\backslash \cdots \backslash x_n$において, x_i (x_i $i = 1 \ldots n$) が唯一の上位概念を持つならば, 視点表現 $x_1\backslash x_2\backslash \cdots \backslash x_n$と x_1は等価である.

5.5　視点表現の単一化

本節では視点表現で表現された概念の個体の単一化を定義する. 個体の単一化の可能性は文脈解析における照応処理などに応用できる.[42,47] 2 つの個体が単一化できるかどうかは, 個体の属す概念の概念階層中の位置関係によって決まる. したがって, 視点表現を含む概念の単一化を定義しなければならない. 厳密にいえば, 単一化は個体に対しておこなわれるものなので, 概念の単一化という表現は正確ではないが, 以下では, 簡単のため, 概念 a の個体と概念 b の個体の単一化の代わりに概念 a と概念 b の単一化という表現を使う.

前提として, 視点表現中に複合概念がある場合は 5.4 節 で述べたように展開できるので, 以下の定義では視点表現中には複合概念はないものとする. また, 複合概念の単一化については, 複合概念の各要素を単一化して得られた結果によって構成される複合概念を複合概念の単一化の結果とする.

定義 5-8　視点表現の単一化
概念 $x = x_1\backslash x_2\backslash\cdots\backslash x_m$ と概念 $y = y_1\backslash y_2\backslash\cdots\backslash y_n$ $(m, n > 0)$ の単一化を次のように定義する.

(1) $x_m \ll y_1$の場合
⇒ $x_1\backslash\cdots\backslash x_m\backslash path(x_m, y_1)\backslash y_1\backslash\cdots\backslash y_n$
ただし, $path(p, q)$ は両端に概念 p, qを含まない p から q への経路を表す.

(2) $x_{m-k+1}\backslash\cdots\backslash x_m = y_1\backslash\cdots\backslash y_k$ なる k が存在する場合
⇒ $x_1\backslash\cdots\backslash x_m\backslash y_{k+1}\backslash\cdots\backslash y_n$

(3) (1), (2) 以外で, x と yが排他的な概念でない場合
⇒ $\{x, y\}$

(4) (1), (2) 以外で, x と yが排他的な概念の場合
⇒ 失敗

図 5-3を使って単一化の例を説明する. ここで, ∼ は単一化を表す.

(1) (a) ∗(にんじん) ∼ ∗(野菜)⇒ ∗(にんじん)\∗(野菜)

(b) ∗(野菜) ∼ ∗(植物)⇒ ∗(野菜)\∗(植物)

(c) ∗(にんじん)\∗(野菜) ∼ ∗(生物)
⇒ ∗(にんじん)\∗(野菜)\∗(植物)\∗(生物)

(a) の例は, 通常の概念階層における概念同志の単一化と同じで, 概念は下位概念の方向に特殊化される.[47] この場合, ∗(にんじん) は唯一の上位概

念を持つので, 定義 5-7より, 単一化の結果である∗(にんじん)\∗(野菜) は ∗(にんじん) と等価である. (b) の例は, ∗(野菜) が複数の上位概念を持つので, 単一化の結果, 上位概念の方向に特殊化される.

(2) ∗(野菜)\∗(食物) ∼ ∗(食物)\∗(無生物)⇒ ∗(野菜)\∗(食物)\∗(無生物)
単一化される概念のパスが一部重複している場合である.

(3) ∗(料理) ∼ ∗(野菜) ⇒ {∗(料理), ∗(野菜)}
単一化される概念間に上位/下位関係がなく, 両概念が排他的でない場合, 両概念から構成した複合概念を単一化の結果とする.

(4) ∗(にんじん) ∼ ∗(かぼちゃ) ⇒ 失敗
単一化される概念が排他的なので単一化は失敗する.

5.6 視点表現の自然言語処理への応用

視点表現の応用として, 文脈解析における漸次的なあいまい性の解消[3,83]や照応参照の解決が考えられる. たとえば,

(5-6) 花子はにんじんが好きである.
彼女はその野菜を毎日食べている.

という文脈において, “にんじん” は図 5-3の概念階層の∗(にんじん) に対応すると考えられる. 次に, 第 2 文を解析すると “その野菜” に対応する概念は “食べる” との選択制限から∗(野菜)\∗(食物) であることがわかる. この 2 つの概念は単一化できて, ∗(にんじん)\∗(野菜)\∗(食物) が得られる. すなわち, 第 1 文を処理した時点では∗(にんじん) であったものが第 2 文を処理した時点では∗(にんじん)\∗(野菜)\∗(食物) に変化するわけであるから, 上位方向の視点によるあいまい性が解消されたことになる. また, 見方を変えれば第 1 文の “にんじん” は第 2 文を処理した時点では, もはや植物のにんじんではありえない, という意味で視点表現は制約の蓄積にもなっている. このように, 文を読み進むにつれて徐々にあいまい性が解消されるモデルを奥村は増進的あいまい性解消モデルと呼んでいる.[3]

また, 例文 (5-6)の “にんじん” と “その野菜” は照応関係にあるが, 2 つの語が照応関係にあるかどうかはそれぞれの語に対応する概念が単一化できるかどうかで判定できる場合が多い. 本章で定義した単一化は照応関係の解決に対する部分的な解を与えている. 我々の単一化の定義では, 単一化される概念が互いに排他的な場合, 単一化は失敗するが, 必ずしもこれは適切でない.

(5-7) 太郎は庭で野菜を栽培している. (植物の野菜)

(5-8) 太郎は毎日野菜を食べている. (食物の野菜)

たとえば, 例文 (5-7)の “野菜” は “栽培する” の選択制限から∗(野菜)\∗(植物) に対応し, 例文 (5-8)の “野菜” は “食べる” の選択制限から∗(野菜)\∗(食物) に対応すると考えるのが自然である. しかしながら, たとえば,

(5-9) 太郎は自分の庭で野菜を栽培している.
彼はその野菜を毎日食べている.

という文脈では, これらの “野菜” は同一の個体を指していると考える方が自然である. 我々の単一化の定義ではこのような単一化は失敗する. 視点表現を用いず, 文脈から得られる個体に関する制約を蓄積しないで, 両方の “野菜” を単に∗(野菜) と考えれば, “栽培する”, “食べる” のいずれの選択制限も満足するのでこの単一化は成功することになる. しかしながら, “栽培して”, “食べる” 間には, なんらかのイベント (たとえば, 収穫など) が起こり, 同一の個体といえどもその野菜には, なんらかの変化が起こっているはずである. 我々は, 単一化の失敗をこのような変化を検出するきっかけとして扱うべきで, これを単一化のレベルで扱うべきではないと考えている. つまり, この例の単一化はむしろ失敗する方が, より深い解析をおこう必要性を検出できるという意味で望ましい. 今後, このような単一化の失敗を受けて, 個体の変化を扱うメカニズムを考える必要がある.

5.7　要約

本章では, 概念階層における概念に関して, これまでの下位方向の視点とは異なる上位方向の視点の存在を明らかにし, その表現方法として概念の視点表現を提案した. また, 視点表現で表現された概念同志の単一化も定義した. 視点表現を用いることによって, 自然言語を処理する際に問題となるあいまい性を表現することができる. また, その単一化は一部の照応参照の解消に用いることができる.

第 6 章

対訳辞書からの概念の自動抽出

前章では自然言語処理に必要な概念体系における概念間の関係, 特に視点の役割を中心に述べた. 本章では, 概念体系に含まれる個々の概念をどのように設定するかについて論じる.

6.1 背景

自然言語処理, 特に意味解析においては, 言語の意味に関する知識として概念体系が重要な役割を果たす. 前章では, 概念体系中の個々の概念同士の関係をどうとらえるかに重点を置いて論じた. 概念同士の関係とともに重要なのは, 個々の概念をどのように設定するか, すなわち, 概念の認定に関する問題である.

この問題に対するひとつの考え方として, 機械翻訳の分野で盛んに研究されている中間言語という考え方がある.[45] 中間言語の考え方とは, 言語に依存しない概念の集合を仮定し, その概念の組合せによって言語表現の意味を表現しようとするものである. 理想的な中間言語が設計できれば, 多言語間の機械翻訳システムを作成する上で大きな利点となる. なぜなら, 現存の機械翻訳システムのほとんどが原言語の構造を目標言語の構造に変換するためにトランスファ規則を使用しているため, 1 つのシステムは 1 言語対の, しかも 1 方向の翻訳しかできないが, 中間言語を経由して翻訳する方式をとれば, 各言語と中間言語の間の双方向の変換システムを用意すれば, n 言語間の翻訳システムが構築できるからである.

中間言語の目標は言語に依存しない普遍的な概念の集合を求めることであるが, 目標をもう少し限定して特定の言語に依存する概念の集合を設定する問題を考えよう. ひとつの方法として, 既存の辞書, たとえば国語辞典などから語の意味を表す語義を抽出し, これを概念として認定することが考えられる. 最近では機械可読な辞書も入手可能なので, このアプローチは実現可能性という点では有

望である. 本章では, このアプローチを一歩進めて 1 組の機械可読な対訳辞書を用い, 2 言語に共通な概念を機械的に抽出する手法について論じる.

中間言語における概念に対する考え方には, Schank の概念依存理論[94]のように言語に完全に独立な概念の原子要素をあらかじめ用意し, その組合せによってすべての概念を表現しようとする立場もあるが, 各言語が持っている概念の総和をそのまま中間言語の概念とする立場[67]も考えられる. 理想的には概念の原子要素を求めることが望ましいが, そこに至る過程として各言語の持つ概念の総和を求めることも重要である. 特に, 言語に共通な概念は, 概念の原子要素を求める上で重要な手がかりとなる. 我々は 2 言語を対象とするが, これを n 言語に拡張して得られる結果は中間言語を構築する上の基礎データとして重要な役割を果たすと考えている. その基本的な考え方は 6.6 節で説明する. 以下, 本章では, 具体的な応用として 2 言語間の機械翻訳を想定し, 機械翻訳システムのための概念辞書の自動構築について述べる.[39]

6.2 翻訳に必要な知識

機械翻訳システムにおいて翻訳の質を向上させるためには, 表層語から概念の世界に踏み込んだ, より深い解析が必要となる. 表層語だけを考慮したのでは, 語の多義性のために適切な訳語が選択できないからである.[25,48] また, 概念にも言語に共通な概念と言語の文化的背景に依存する言語固有の概念があるので, 概念の世界を考える場合に原言語と目標言語の両方の概念を考慮することが重要である.

概念レベルの解析をおこなう機械翻訳では, 少なくとも以下の情報が必要となる.

(1) 概念の集合

(2) 表層語と概念の対応関係

(3) 原言語の概念と目標言語の概念の対応関係

概念レベルの解析を目指した機械翻訳用の辞書に関する大規模な研究として, 日本電子化辞書研究所 (EDR) がおこなっている概念辞書の開発があげられる. EDR では上述の 3 つの要素に対応して, それぞれ, (1) 概念辞書, (2) 単語辞書, (3) 言語間対訳辞書の開発を目指している.[40] EDR の概念辞書は概念の単なる集合ではなく, 概念と概念の間にどのような関係が許されるかも記述されている. また, 言語間の対訳辞書として日本語－英語間の辞書を考えている.

EDR では, これらの辞書の構築を計算機の支援によって, すべて人手でおこなうというアプローチをとっている.[40,46] また, Carnegie Mellon 大学の Nirenburg

らも計算機の支援により概念を人手によって組織的に構築することを提案している.[88] このようなアプローチの問題点として作業者の主観によって概念の設定にゆれが生じることが挙げられる.[23] また, 各言語の概念間の対応関係を設定する作業では, 両言語の概念数の積に等しい対応関係の検査が必要となる. これらの作業を人手でおこなうことを考えると, その量は膨大なものとなる.

これに対して, 我々は機械可読な対訳辞書の対からこれらの 3 つの情報を機械的に抽出する手法を提案する. ここでいう対訳辞書とは市販の英和辞典, 和英辞典などの辞書のことである. もちろん, 本論文で提案する手法によって完全な概念の集合や概念の対応関係が自動的に抽出できるわけではない. 最終的には人間による精密化や修正が必要となるが, 重要なことは, 機械的に処理できる部分はできるだけ計算機でおこなうという点である. たとえば, 概念間の対応関係を設定する場合に, 作業者の負担という観点から考えると, 人間が辞書を参照しながら一から設定するよりは, 計算機で自動的に抽出した対応関係について, それが正しいかどうかを判断する方が負担ははるかに軽減される.

上述した 3 つの情報に対する我々の基本的な考え方を以下に述べる. まず, (1) 概念の集合については, 対訳辞書中で定義されている語義を概念の候補として考え, 語義の集合で近似する. 対訳辞書中の語義は, その見出し語が持つ意味を表しており, 各語義には適切な目標言語の訳語が割り当てられている. したがって, 機械翻訳における訳し分けを考える場合に, 語義を概念として設定し, それを経由した訳語選択をおこなうことには意味がある. また, 対訳辞書の語義は見出し語に対する語義であるから, 語義を概念として設定すれば, (2) 表層語と概念の対応関係は自動的に求めることができる. 以下では, 特に断らない限り語義と概念を同義として使う.

もっとも困難なのは (3) 原言語の語義と目標言語の語義の対応関係をどのようにして抽出するかという問題である. 本論文では, 2 言語に関する双方の対訳辞書を使うことによってこの問題に対する一つの解を与える. このために, 6.3 節では対訳辞書を翻訳グラフとしてモデル化し, 翻訳回路という概念を導入する. 翻訳回路の直感的な意味は, 両言語の見出し語の対について両方向の辞書でこの見出し語をそれぞれ引いた時, お互いにいずれかの語義が訳語として相手の見出し語を含むことである. 翻訳回路中に含まれる語義の対が対応関係にある, つまり, 意味がほぼ等しいというのが我々の仮定である. 6.4 節では, 既存の英和辞典, 和英辞典を用いた語概念の抽出実験について, 6.5 節では, EDR の対訳辞書を用いた実験について述べる.

6.3 対訳辞書の構造

6.3.1 対訳辞書と語義間の対応

本節では, 対訳辞書から 2 言語間の語義対応を抽出するための準備として対訳辞書の構造について考察する. 2 つの言語 L^a と L^bについて, L^aから L^bへの対訳辞書 $D_{a\to b}$と L^bから L^aへの対訳辞書 $D_{b\to a}$を考える. 今, $D_{a\to b}$の見出し語 $a_i \in L^a$が m 個の語義 $a_i/1, \ldots, a_i/m$ を持ち, $D_{b\to a}$の見出し語 $b_j \in L^b$が n 個の語義 $b_j/1, \ldots, b_j/n$ を持つとする. そして, a_iの語義 $a_i/2$ の訳語が b_jであったとしよう (図 6-1). ここで, 1 つの語義が複数の訳語に翻訳されることもあるので, 一般に各語義からは目標言語の複数の見出し語に向けた有向辺が張られる.

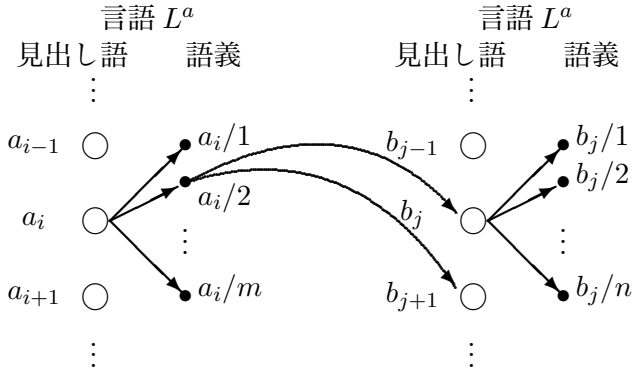


図 **6-1**　対訳辞書の構造

図 6-1は, $D_{a\to b}$の見出し語 a_iの語義 $a_i/2$ の訳語が b_jであることを示している. この時, 訳語 b_jは n 個の語義を持つが, その内のどれが $a_i/2$ に対応する語義であるかはわからない. これは, 対訳辞書が語義と訳語 (見出し語) との間の対応を示しているだけで, 語義間の対応を示していないことに起因している. この対応を機械的に抽出するのが我々の目的である.

6.3.2 翻訳回路

我々は, 対訳辞書を 1 つの有向グラフと見なし (図 6-1), このグラフを翻訳グラフと呼ぶ. 言語の対を決め, 言語 L^aから言語 L^b(ただし, $a \neq b$) への翻訳グラフを TG_{ab}と書く. 添字の順序は翻訳の方向を表す. TG_{ab}は 4 つ組 $\langle H_a, S_a, T_b, E_{ab}\rangle$ で構成される. ここで, H_a, S_a, T_bは節点の集合, E_{ab}は辺の集合である. $H_a \subset L^a$ は TG_{ab}の見出し語の集合, S_aは H_aの持つ語義の集合, $T_b \subset L^b$は TG_{ab}の訳語の集合を表す. T_bの中には TG_{ba} の見出し語に含まれないものも存在する.

TG_{ab}を用いた翻訳は, H_aの要素 a_iに対して 1 つの語義 a_i/kを選択し, a_i/k の持つ訳語 b_jを選択するというプロセスになる. この翻訳プロセスには $a_i \to$

$a_i/k \rightarrow b_j$という経路が対応する. これを翻訳経路と呼ぶ.

ここで, 2 つの翻訳グラフ TG_{ab}と TG_{ba}の和 TG_{ab+ba} $(= TG_{ab} \cup TG_{ba})$ を考え, これを双方向翻訳グラフと呼ぶ. TG_{ab+ba}において, 閉路 $h_a(\in H_a) \rightarrow s_a(\in S_a) \rightarrow h_b(\in H_b) \rightarrow s_b(\in S_b) \rightarrow h_a$を翻訳回路と呼ぶ. ここで, 見出し語 $h_a \in H_a$ と $h_b \in H_b$を含む回路が存在するとき, 見出し語 h_aと h_bの間に回路が存在するという.

翻訳回路の例を図 6-2に示す. 翻訳回路に含まれる語義の組によって翻訳回路を表現する. たとえば, 図 6-2の翻訳回路は $\langle a_i/2, b_j/1\rangle$ と表現する.

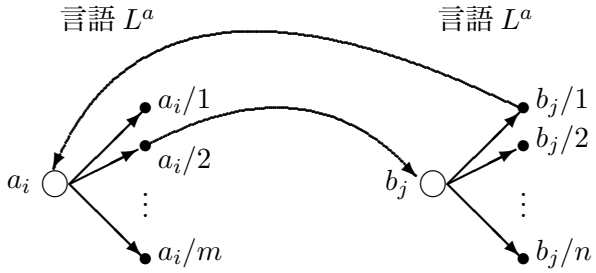


図 6-2　A 型の翻訳回路

次に, 見出し語対を固定した時に, その見出し語の間にできる翻訳回路を 3 種類に分類する.

> **定義 6-1　A 型の翻訳回路**
> 両言語の見出し語 h_a, h_bの間に唯一の翻訳回路が存在するとき, この回路を A 型の翻訳回路という.

たとえば, 図 6-2の見出し語 a_iと b_jの間に存在する翻訳回路は $\langle a_i/2, b_j/1\rangle$ のみである. したがって, これは A 型の翻訳回路である.

> **定義 6-2　B 型の翻訳回路**
> 両言語の見出し語 h_a, h_bの間に複数の翻訳回路が存在し, これらの翻訳回路に含まれる語義の集合をそれぞれの言語について考えた時, 一方の言語の語義の集合が唯一つの要素しか持たないならば, これらの回路を B 型の翻訳回路という.

たとえば, 図 6-3の見出し語 a_iと b_jの間には 2 つの翻訳回路 $\langle a_i/k, b_j/p\rangle$ と $\langle a_i/k, b_u/q\rangle$ が存在し, いずれの回路も言語 L^aの語義については a_i/kしか含まない. したがって, $\langle a_i/k, b_j/p\rangle$ と $\langle a_i/k, b_u/q\rangle$ は B 型の翻訳回路である.

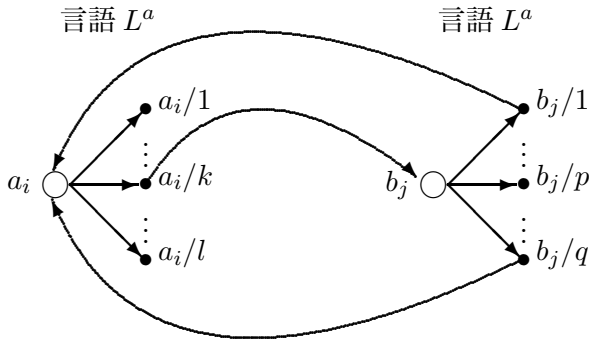


図 6-3　B 型の翻訳回路

定義 6-3　C 型の翻訳回路
両言語の見出し語 h_a, h_bの間に複数の翻訳回路が存在し, これらの翻訳回路に含まれる語義の集合をそれぞれの言語について考えた時, いずれの言語の語義の集合も複数の要素を持つならば, これらの回路を C 型の翻訳回路という.

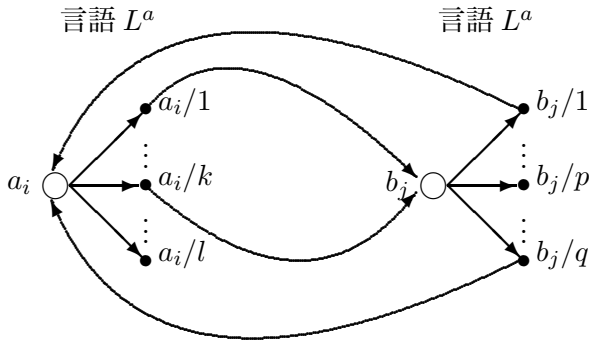


図 6-4　C 型の翻訳回路

たとえば, 図 6-4の例は, 見出し語 a_iと b_jの間に 4 つの翻訳回路 $\langle a_i/k, b_j/p\rangle$, $\langle a_i/k, b_j/q\rangle$, $\langle a_i/l, b_j/p\rangle$, $\langle a_i/l, b_j/q\rangle$ が存在し, これらの回路は両言語の語義を 2 つずつ含んでいる. したがって, これらは C 型の翻訳回路である.

我々は翻訳回路中に含まれる語義は互いに意味が等しいと仮定している. このように意味の等しい語義の対を語義対応と呼ぶ. 6.4 節, 6.5 節の実験によってこの仮定の妥当性を検証する.

6.3.3　同言語内の語義対応

翻訳回路から得られる語義対応は, 異なる 2 言語の見出し語の語義に対して与えられるものである. これに対して, 同言語の見出し語が持つ語義の中にも語義対応を考えることができる. たとえば, 翻訳回路から図 6-5に示すような語義対応が得られたとしよう. $a_i, a_j, \ldots$, $b_i, b_j, \ldots$ はそれぞれ言語 L^a, L^bの語義を表し, 実線は語義対応を表す.

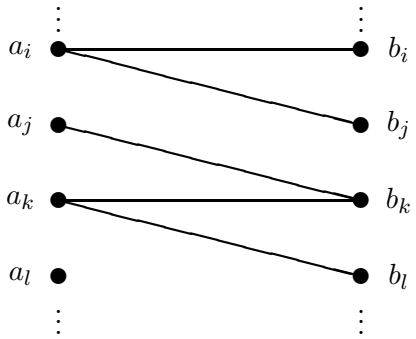


図 6-5　語義対応の例

ここで, 語義対応に推移律を仮定すると相手言語の語義を介して語義対応で結ばれた同言語の語義間にも語義対応を設定することができる. たとえば, 図 6-5 の b_iと b_j, a_jと a_k, b_kと b_lの間にそれぞれ対応関係を設定できる. したがって, 語義対応で結ばれたすべての語義を 1 つの概念としてまとめれば, 概念の数を減らすことも可能である. しかし, 語義対応の推移律が実際にどの程度成り立つかは, 実験によって調べる必要がある. これについては 6.5 節で述べる.

6.4　実験 1 (市販の対訳辞書を用いた実験)

本節では, 市販の対訳辞書の一部を用いておこなった語義対応の抽出実験について説明し, その結果について考察する. 実験用のデータとしては, 研究社のライトハウス英和辞典[33]とライトハウス和英辞典[20]を用いた. これらの辞書を選んだのは, 以下の理由による.

- 2 つの辞書の編者が同じで, 同じ編集方針に基づいて編集している.
- 語釈文による語義の定義が明確で, 語義を機械的に抽出しやすい.

6.4.1 準備

まず, 6.3 節で述べた語義対応の定義を実際の対訳辞書に適用するために, 記法上の準備をする.

語義の定義

実際の対訳辞書を観察すると, 英和辞典では, bank (土手) と bank (銀行) のように, 同じつづりで異なる意味を表す語は別の見出し語として扱われている. 一方, 和英辞典では, 1 つのひらがな表記の見出し語に複数の漢字表記が割り当てられている場合がある.

英和, 和英のいずれの辞書も, 語義は以下のような階層構造を持っている.

- 同一見出し語の中は意味の違いによっていくつかの項目に分けられ, 数字が割り当てられている. これを大項目と呼ぶ.
- 大項目の中は, 意味の違いによっていくつかの訳語が “;” または “,” で区切られている. この内, “;” の方が意味の違いが大きいことを表しており, “,” は交換可能な程度の意味の違いを表している. “;”, “,” で区切られたものをそれぞれ中項目, 小項目と呼ぶ.

小項目のレベルに現れる訳語はほとんど同義と考えられるので, 実験では中項目のレベルを 1 つの語義として定義する. したがって, 1 つの語義に複数の訳語が割り当てられることもある. また, いずれの辞書においても, 訳語中で, 省略可能な要素 (語) は ( ) で, 代替要素は [ ] で囲まれている. いずれの辞書についても省略可能要素を省略したものと省略しないものを訳語とした. また, 代替要素については, 和英辞典の場合は代替要素を置き換えたすべての場合を訳語としたが, 英和辞典の場合は代替要素を省略した. これは, 英和辞典では, 訳語 (日本語) が分かち書きされていないので, どの部分が代替要素と置き換わるかが機械的に判別できないからである.

訳語の中には 1 つの語ではなく, 句や, 訳語に注釈文が付いた形で記述されているものがある. しかしながら, このような句や注釈付きの語が反対方向の辞書の見出し語として現れることはほとんどありえない. 訳語に対する注釈は語義の説明と考えられるので削除した. 語義の説明は語義を区別することによって反映されるものと仮定する.

英和辞典の訳語中に「(助詞+) サ変名詞+する」の形でサ変名詞が現れることが多いが, 和英辞典の見出し語にはサ変名詞の形でしか現れない. 英和辞典の訳語をプログラムによって抽出する際に, 「(助詞+) サ変名詞+する」の形はサ変名詞に変換した. その他の複数語からなる訳語については, このような訳語がど

れくらいの頻度で存在するかを調べるのも実験の目的のひとつなので, そのまま実験データとして含めた.

語義の表現

実験では, 語義を大項目番号, 品詞, 中項目番号の 3 つ組で表現する. 大項目番号は辞書中で大項目に振られている番号をそのまま使用する. 中項目番号は, 大項目中での中項目の出現順に番号を 0 から割り当てる. 実験の対象とした品詞は, 名詞, 動詞, 形容詞, 副詞の 4 つである. 品詞については, 英和辞典と和英辞典で若干扱いが異なっており, 和英辞典では, すべての訳語の品詞が機械的に抽出できるわけではない. このような場合は品詞不明として, いずれの品詞にもなる可能性があるものとして扱った.

6.4.2　実験方法および結果

ライトハウス英和辞典, 和英辞典はいずれも機械可読な形式になっていないので, 手作業で辞書を計算機に入力した. 理想的には両方の辞書を全部入力することが好ましいが, 経済的, 時間的資源の制約上, これは不可能であった. そこで, 対訳付きの英語のエッセイ集[77]から 6 編のエッセイを選び, その中に現れる自立語について辞書項目を入力した. 日本語の場合は, 単語の分かち書きが必要であるが, これも手作業でおこなった. このエッセイを選んだのは, 1 つのエッセイが 300 語程度の比較的容易な英語で書かれているので, 一般的な語が多いであろうと予想したからである. 最終的に, 英和辞典については 562 見出し語, 和英辞典については 779 見出し語を入力した. 入力に際しては, 大項目を 1 行として入力する程度の加工はおこなったが, 基本的に辞書の字面をそのまま入力した. ただし, 例文は省略した.

次に簡単なプログラムを作成し, これらの見出し語について見出し語, 語義, 訳語の組を機械的に抽出した. これは, 6.3 節で述べた翻訳経路に相当する. 英和, 和英についてそれぞれ 7,824 と 4,809 の翻訳経路を抽出することができた. 品詞別の翻訳経路数を表 6-1に, 各見出し語が持つ語義数を表 6-2に, 各語義が持つ訳語を表 6-3に示す.

表 **6-1**　翻訳経路数 (品詞別)

	名詞	動詞	形容詞	副詞	不明	合計
英和辞典	3,145	2,939	1,061	679	0	7,824
和英辞典	1,240	1,784	471	232	1,082	4,809

表 **6-2**　見出し語あたりの語義数

	英和辞書		和英辞書	
総見出し語数	562		779	
語義数	頻度	累積 [%]	頻度	累積 [%]
1	60	10.7	121	15.5
2	72	23.5	149	34.7
3	54	33.1	102	47.8
4	60	43.8	92	59.6
5	39	50.7	80	69.8
6	31	56.2	54	76.8
7	32	61.9	37	81.5
8	32	67.6	28	85.1
9	24	71.9	25	88.3
10	18	75.1	12	89.9
⋮	⋮	⋮	⋮	⋮
	平均	8.33	平均	5.16

表 **6-3**　語義あたりの訳語数

	英和辞書		和英辞書	
総語義数	4,681		4,022	
訳語数	頻度	累積 [%]	頻度	累積 [%]
1	2,243	47.9	3,266	81.0
2	1,844	87.3	623	96.0
3	496	97.9	80	98.6
4	87	99.8	43	99.7
5	6	99.9	3	99.8
⋮	⋮	⋮	⋮	⋮
	平均	1.67	平均	1.24

表 **6-4**　翻訳回路の抽出結果

型	正	誤	総数	正当率
A 型	215	3	218	98.6%
B 型	86	35	121	71.0%
C 型	31	73	104	29.8%
合計	332	111	443	74.9%

次に, 抽出した翻訳経路を Prolog の単位節に変換し, Prolog インタプリタによって翻訳回路の数を数えた. 得られた翻訳回路について回路中の語義の組が正しい対応関係にあるかどうかをすべて手作業で検査した. その結果を表 6-4に示す.

6.4.3　考察

A 型の翻訳回路

A 型の翻訳回路は 218 個抽出できた. このうち, 3 つについては回路中に含まれる語義に対応関係はなかったが, 残りの 215 はいずれも正しい対応関係を含んでいた. したがって, 98.6%の正しさで語義対応が抽出できたことになる. A 型の翻訳回路については, 機械的に抽出したものをそのまま語義対応として使用できると考えられる.

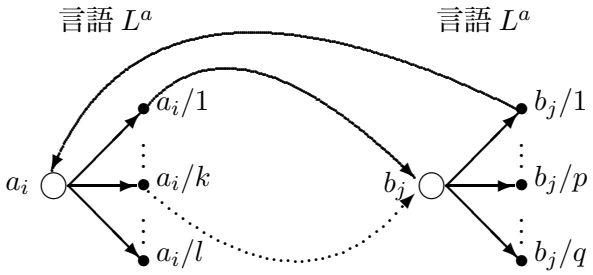


図 **6-6**　正しい語義対応を与えない例

語義対応の誤りは, 3 例とも図 6-6に示す状況で起こっていた. すなわち, 語義 $b_j/1$ に対応するのは a_i/kであるが, a_i/kに訳語 b_jが与えられておらず (点線で表示), たまたま $b_j/1$ とは意味の異なる a_iの語義 $a_i/1$ に訳語 b_jが与えられている場合である. このような状況では $a_i/1$ と $b_j/1$ の語義対応が抽出されてしまう. また, 一方の辞書の語義に対応する語義が, もう一方の辞書に用意されていない場合もあった.

B 型の翻訳回路

B 型の翻訳回路は 121 個抽出できた. このうち, 正しい語義対応を与える回路が 86 あった. B 型の翻訳回路のうち, 約 70%が正しい語義対応を与えている. B 型の翻訳回路では見出し語対の間の語義対応の候補が 1 対多の関係になる. 1 つの語義がいくつの語義に対応しているかを表 6-5に示す. 正答率が約 70%と低いので, B 型の翻訳回路から語義対応を選別するためには, 人間の判断に頼らざるを得ないが, 表 6-5からもわかるように, ほとんどの場合, 語義が 1 対 2 の関

係であるから, 適切なインターフェイスを用意すれば人間の負担はさほど大きくないと予想できる.

表 **6-5**　**B** 型の翻訳回路の語義対応

組合せ	1 対 2	1 対 3	1 対 4	1 対 5	1 対 8
頻度	86	9	8	10	8

語義の対応関係が 1 対多になる主な理由は, 一方の辞書の語義が他方の辞書の複数の語義を含む場合があるためである. 特に多くの語義を持つ見出し語は, より特殊な意味を持つ語義の前に, それらをすべて含むような一般的な語義を持つ場合がある. 例として, 見出し語 “write” と「書く」の間で抽出できた 5 つの B 型の語義対応を図 6-7に示す. この例では, “write” の語義 (1) は “write” の一般的な意味を表し, “write” の他の語義も含んでいると考えられる. これに対して, 「書く」の語義 (i) は一般的な「書く」を意味を表し, “write” の語義 (1) と対応する. 「書く」の方には “write” のように細かい語義の定義がないため, 語義 (i) と語義 (2), . . . , (4) の対応関係もそれぞれ抽出されてしまう. 実験 1 では, この例の場合 (1)-(i) の対応のみを正しいものとして数えた.

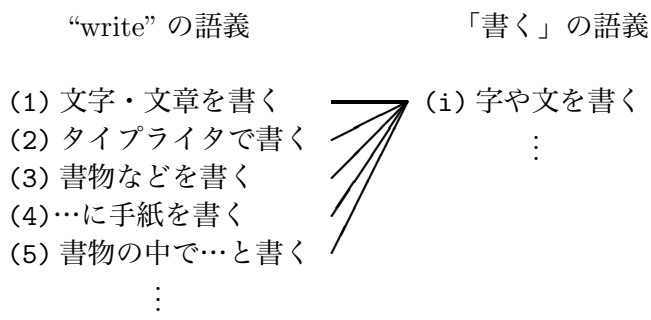


図 **6-7**　**“write”** と「書く」の語義対応

C 型の翻訳回路

C 型の翻訳回路は 104 個抽出できた. このうち, 正しい語義対応を与える回路が 31 あった. C 型の翻訳回路のうち, 約 30%が正しい語義対応を与えている. C 型の翻訳回路を作る見出し語の間には一般に複数の語義対応が得られる. 対応する語義の組合せを表 6-6に示す.

表 **6-6**　**C** 型の翻訳回路の語義対応

組合せ	2 対 2	2 対 3	2 対 4	2 対 6	3 対 3	3 対 4	4 対 8
頻度	12	27	8	12	6	7	32

より一般的な語ほどより多くの語義対応が得られる傾向がある．たとえば, 見出し語 “take” と「取る」の間には, 32 の C 型の翻訳回路が抽出できた．これらの回路に関与した語義は “take” が 4 つ, 「取る」が 8 つである．また, 正しい語義対応は 4 つであった．この例に関していえば, “take” 側の語義すべてについて「取る」側の対応する語義が見つかったことになる．問題は 32 の中から正しい 4 つをどうやって選別するかである．この選別は B 型の翻訳回路同様, 人間の判断に頼らざるをえないが, 判断を容易にするようなインターフェースを与えることが重要となる．

辞書の対称性

実験結果からわかるように, 翻訳回路を構成しない翻訳経路の数が非常に多い．この実験では, 辞書の見出し語の 1 部しか使っていないことも原因の 1 つであるが, その他にもいくつかの理由が考えられる．Byrd は, この理由として以下の 4 つを挙げている.[58] ただし, Byrd の例は英語–イタリア語である．

(1) 語がほとんど派生形で使われる場合

(2) 特殊な語の訳語として一般的な語をあてている場合

(3) 一般的な語の訳語としてより特殊な語をあてている場合

(4) 辞書編集上の洩れによる場合

今回の実験に関して例を挙げれば, 次のようなものがあった．矢印は翻訳の方向を表す．

dirt → わいせつな文章	(2) の例
fox → きつねの毛皮 (全体で部分を指す)	(3) の例
machine → 自動車 (上位で下位を指す)	(3) の例

我々は, 双方向に語の翻訳ができる場合に限り, 語義対応を考えている．1 方向に語の翻訳できれば逆方向の翻訳も必ず可能であるという考え方もあるが, ここに挙げた例からわかるように, 翻訳のためには特殊な文脈が必要な場合がある．我々の目的は語義の対応を抽出することであるから, 文脈に依存する情報を使うことは好ましくない．しかし, 1 方向にしか語の翻訳ができない例は, 言語に依存した概念の手がかりを与える場合がある．このような例はここでは対象としていないが, 最終的に翻訳システムを考える上では検討しなければならない問題である．

我々の手法は語義に割り振られた訳語が相手の対訳辞書の見出し語として現れることを前提としている．しかしながら, 訳語は対訳辞書の見出し語として現

れるような単一の語でなく, 複数の語からなる句や文となることも多い. 表 6-7 は, この実験で使用した和英辞典のデータに関して訳語が何単語からなっているかを調べたものである.

表 **6-7**　和英辞典の各訳語の単語数

訳語数	3,265	
単語数	頻度	[%]
1	1,940	62.2
2	702	22.5
3	307	9.8
4	131	4.2
5	32	1.0
6	7	0.2
7	2	0.1
	平均	1.60

表 6-7から和英辞典では約 62%の訳語が 1 単語からなっていることがわかる. これらの訳語のみが英和辞典の見出し語として現れる可能性がある. 実際, これらの 1,940 の訳語のうち, 実験に使用した英和辞典の見出し語として現れるのは約 20%の 387 語である. 参考のために, これらの訳語を電子化された約 5 万見出しのコンサイス英和辞典の見出し語と比較した結果, 1,820 語が見出し語に含まれることがわかった. これは, 約 93%に相当する. 訳語が複数語からなる場合を含めると全体の約 60%がコンサイスの見出しに現れることになる.

一方, 英和辞典については, 訳語がわかち書きされていないので, 訳語が和英辞典の見出し語としてどれくらいの割合で現れるかを調べることは容易ではない. 実験で使用した英和辞典のデータに現れる訳語全体のうち, 和英辞典の見出し語として現れるのは 306 語であった. 訳語の数が和英の約 2 倍の 6,191 であることを考えると, 英和辞典の訳語を和英辞典で引くには訳語に何らかの処理が必要であることがわかる. たとえば, “associate” の語義の 1 つに「仲間に加える」という訳語が割り当てられているが, 和英辞典には「仲間に加える」という見出し語はないので翻訳回路はできない. このような訳語を扱うためには, まず, 訳語の形態素解析をおこない, 見出し語として引ける単語に分割する必要がある. その上で対応がとれれば, 概念レベルで構造変換を必要とする概念の対応関係に関する情報が抽出できる可能性がある. 複数からなる訳語をどのように扱うかは今後さらに検討を加える必要がある.

6.5　実験 2 (EDR の対訳辞書を用いた実験)

前節では, 市販の英和, 和英辞典の一部を用いて, 語義対応の抽出実験をおこなった. 本節では, EDR が作成した比較的大規模な英和, 和英対訳辞書を用いて同様の実験をおこなう.

6.5.1　実験方法および結果

まず, EDR の対訳辞書から必要な情報を抽出する. 我々の目的には, 見出し語, 語義, 訳語の 3 つ組があれば十分である. EDR の対訳辞書には, 概念 ID と呼ばれる語義レベルの識別子が含まれているので, この識別子をそのまま語義の識別子として使用する. EDR の辞書は計算機処理を前提として作成されているので, 実験 1 で用いた市販の辞書に比べ, 必要な項目を計算機で抽出するのは容易である. ただし, 計算機資源の制約からすべての見出し語について実験することが困難であったので, 一番数が多く, 手作業で概念を設定する場合に問題となると考えられる名詞概念を抽出するために, 名詞の見出し語のみについて実験をおこなった.

対訳辞書の名詞の見出し語数は, 英和辞典 72,596, 和英辞典 127,730 である. この辞書から抽出できた翻訳経路は, 英和辞典 153,520, 和英辞典 325,055 である. 実験 1 と同様に, 各見出し語が持つ語義数, 各語義が持つ訳語数を表 6-8, 表 6-9に示す. また, 翻訳回路の抽出結果を表 6-10に示す.

表 **6-8**　見出し語あたりの語義数

	英和辞典		和英辞典	
見出し語数	72,596		127,730	
語義数	頻度	累積 [%]	頻度	累積 [%]
1	45,558	62.8	88,733	69.5
2	16,684	85.7	27,385	90.9
3	5,166	92.9	6,351	95.9
4	1,995	95.6	2,981	98.2
5	1,007	97.0	992	99.0
⋮	⋮	⋮	⋮	⋮
	平均	1.76	平均	1.48

表 **6-9**　語義あたりの訳語数

	英和辞典		和英辞典	
語義数	127,908		189,104	
訳語数	頻度	累積 [%]	頻度	累積 [%]
1	104,154	81.4	109,673	58.0
2	22,051	98.7	47,445	83.1
3	1,574	99.9	18,139	92.7
4	114	99.9	7,910	96.9
5	5	99.9	3,162	98.5
⋮	⋮	⋮	⋮	⋮
	平均	1.20	平均	1.72

表 **6-10**　翻訳回路の抽出結果

型	A 型	B 型	C 型	合計
	11,336	7,819	3,964	23,119

6.5.2　考察

翻訳回路

抽出できた 23,119 の翻訳回路のうち, A 型は約半数の 11,336 であった．このうち 2,450 を標本として抽出し, 手作業で回路中の語義対応を検査した結果, 98.6%が正しい対応であった．2 つの実験から, A 型の翻訳回路については正しい語義対応を与えると結論できる．対応関係が正しくない理由として以下の 2 つが観察できた．

(1) 両語義の間に上位/下位関係が成立する場合

(2) 一方が比喩的な意味を持つ語義の場合

(1) の例として, “art” (the making or expression of what is beautiful or true, especially in a manner that can be seen, as in a painting) と「美術」(色や形などを手段として, 美を表現する芸術) という対応関係があげられる．英語の “art” の語義は芸術一般を指しているのに対し, 日本語の「美術」の語義は範囲がより限定されている．(2) の例として, “mier” (a troublesome or intractable situation) と「泥沼」(どろ深い沼) の対応関係があげられる．この例では, 英語の “meir” の語義が比喩的な意味を持っているのに対し, 日本語の「泥沼」の語義は字義通りの意味である．

辞書の対称性

実験 2 では, 対訳辞書の名詞の見出し語をすべて使用したにもかかわらず, 10 万以上の語義のうち, 正しい対応が自動的にとれたものはわずかに 1 万あまりであった. この原因として, EDR の和英辞書の訳語の多くが複数語の訳語であることが考えられる. 表 6-11は実験 1 の表 6-7に対応するもので, 和英辞書の訳語の単語数を示している.

表 **6-11**　和英辞典の各訳語の単語数

訳語数	180,401	
単語数	頻度	[%]
1	34,504	19.1
2	79,698	44.2
3	26,420	14.7
4	16,733	9.3
5	10,295	5.7
⋮	⋮	⋮
28	1	−
	平均	2.69

表 6-11を表 6-7と比較すると, 1 単語の訳語がライトハウス和英辞典では, 約 60%を占めているのに対し, EDR の辞書では, 20%弱であることがわかる. したがって, 和英辞書の訳語を英和辞書で引く時点で訳語の 80%は辞書引きに失敗することになる. さらに, 1 単語の訳語 34,504 について, EDR の英和辞書の見出し語として現れるものを調べると約 63%であった. EDR の英和辞書の見出し語として現れない残りの 37%についてコンサイス英和辞典で調べると, そのうちの約 30%は見出し語として引けることがわかった. したがって, EDR の英和辞典は見出し語数は名詞のみで約 7 万と多いが, コンサイスの見出し語にある多くのものを含んでいないということがわかる. コンサイスの見出し語数はすべての品詞を含んで約 5 万である. また, EDR の和英辞書の 1 単語の訳語全体をコンサイスで引くと, 約 58%の訳語しか引けなかった. これは, ライトハウス和英辞典の 1 単語の訳語の 93%がコンサイスの見出し語に現れるのとは大きな違いである. 以上から EDR の対訳辞書は以下のような特徴を持つと推測できる.

- 和英辞書の訳語が説明的, または特殊な語である
- 英和辞書の見出し語に特殊な語が多い

EDR では, 対訳辞書を人間が概念間の対応を考える時の参照用として使用している. したがって, そのための目的としては訳語が説明的な方が概念の違いがわ

かりやすいので有効である. しかし, 我々の目的にはこのような性質はそぐわない. 実験 1, 実験 2 から, 市販の対訳辞書の方が我々の目的には合致しているということが結論できる.

同言語内の語義対応

6.3 節で述べたように, 語義対応に推移律を仮定すれば, 抽出した語義対応をさらにまとめることができる. 実験 2 で抽出できた 11,336 の語義対応について語義をまとめた結果, 表 6-12のような結果が得られた. ただし, ここでは図 6-5 の実線で結ばれた語義を 1 グループとして数えている.

表 **6-12**　語義のグループ化

語義数	3	4	5	6	7	8	9
頻度	1,225	285	65	19	10	4	2
正答率 [%]	97	95	92	84	90	50	50

表 6-12からグループ内の語義数が 5 以下では 90%以上の正答率が得られている. 語義対応を経由するたびに語義の微妙な違いが蓄積され, 異なる語義が同一のグループに入ってしまう. したがって, グループ内の語義数が増えるにしたがって推移律を仮定しにくくなり, 正答率が低くなるのは自然な結果である.

6.6　多言語への拡張

これまでの議論では, 対象を 2 言語に限定したが, この手法を複数の言語間に適用すれば, 言語に共通な中間言語を構築するための基礎データとなる可能性がある. 今, $L^c, L^1, \ldots, L^n$ の $n+1$ 個の言語について, 以下のような手順を考える.

(1) 中心言語 L^c と L^1 の間の語義対応を抽出する.

(2) (1) で得られた語義対応に含まれる L^c の各語義に $L^2, \ldots, L^n$ の対応する語を割り当てる.

(3) (2) で割り当てた情報と辞書 $D_{2\to c}, \ldots D_{n\to c}$ を用いて $n-1$ 個の 2 言語間語義対応を抽出する. ここで, $D_{a\to b}$ は, 言語 L^a から L^b への対訳辞書である.

このうち, (1) と (3) は計算機による自動化が可能であるが, (2) については, 人間の判断に頼らざるをえない. (2) については, 中心言語の語義に人手で語を割り当てるのではなく, 既存の辞書 $D_{c\to i}$ $(i = 2, \ldots n)$ を用い, 機械的に語義対応を抽出することも考えられる. しかし, この場合, 辞書 $D_{c\to i}$ の語義の定義はす

べて異なるので, 辞書間の語義の対応関係を人手で求める必要がある. どちらの方法がよいかは一概には言えない. いずれにしても, このようにして作成した中心言語の語義とその他の言語の語義の対応関係は言語に共通な中間言語を構築する上で重要な役割を果たすであろう. 我々はこうして得られた語義とその間の対応関係をもとに, 中間言語の概念項目の集合の核が構築できるのではないかと考えている.

6.7　要約

本章では, 対訳辞書で定義されている語義を概念の近似として用い, 2 言語間の語義の対応関係を機械可読な対訳辞書の対から機械的に抽出する方法を示し, その実現可能性について検討した. 本手法では, まず, 2 言語間の対訳辞書を翻訳グラフでモデル化し, 3 種類 (A 型, B 型, C 型) の翻訳回路を抽出する. そして, 抽出した翻訳回路に基づき語義対応を求めるという手順をとる. 本手法を市販の英和, 和英辞典の一部と EDR の対訳辞書の組にそれぞれ適用し, 語義対応の抽出実験をおこなった結果, いずれの実験においても A 型の翻訳回路からは 98%以上の正答率で正しい語義対応を抽出することができた. また, 最後に本手法の多言語への拡張についても述べた.

第 7 章

結 論

本論文では, 自然言語処理における問題を, 主として処理に必要な知識の側面から検討し, いくつかの解決策を与えた.

第2章では, 自然言語解析において基礎が固まっていると考えられる形態素解析, 統語解析のためのソフトウェアツールである LangLAB について, その特徴と処理速度の高速化手法を中心に述べた. LangLAB を使えば文法と辞書を用意するだけで統語構造を得るための解析システムを自動的に得ることができる. 我々は, LangLAB が自然言語処理システムを開発するための有力なツールになると信じている. 残された問題としては, 文法規則開発用のデバッガの開発が挙げられる. 現在, 文法規則のデバッグは Prolog のデバッガを使っているため, 不必要に細かい部分までトレースをしてしまうなど, 柔軟性に欠ける面がある. また, さまざまな情報を統語木などの 2 次元的な方法によって提示するなどの機能も必要である. 現在, これら点については改良が加えられている.

第3章では, LangLAB で採用している文法記述形式 XGS の限界を克服するために新たな文法記述形式 LG^2を提案した. LG^2では, 構成素の移動にともなう長距離依存関係をより柔軟に記述できるように支配制約という概念を導入した. XGS などでは記述できなかった言語現象を LG^2では容易に記述できることを英語の具体例によって示した. 我々の最終的な目標は, 入力文からその文の意味構造を抽出することであり, 統語解析はその手段にすぎない. 単に構成素の移動を文法上で記述できるだけでは不十分であり, 移動した構成素と後に残されたギャップの同一性を宣言的に正しく記述できることが重要である. 支配制約はこの問題に対するひとつの解を与えている. 第3章の主な目的は文法記述における支配制約の有効性を示すことなので, 3.4 節で述べた実装法は必ずしも最良のものとはいえない. この実装法は下降型深さ優先解析器なので, Pereira の DCG の実装と同様に左再帰規則が扱えないという問題がある. また, BUP[79]や LangLAB[36]のように再計算を回避する機構も持っていない. 今後, 支配制約を

効率のよい解析アルゴリズム上に実現し, 大規模な文法の記述により, その有効性をさらに検討する必要がある.

第4章では, 佐伯の提案したかかりの広さという概念を用い, 後置詞句の並びと動詞という構造を持つ日本語の単文の次入力を推定するモデルを提案した. 基本的な考え方は, 動詞の結合価情報から意味素性と格のすべての組合せについて, 各組合せを結合価パタン中に持つ動詞の数を計算し, その数でかかりの広さを定量化するものである. また, 実際に IPAL 動詞辞書から結合価パタンを抽出し, IPAL 動詞辞書の各動詞についてかかりの広い順に後置詞句を配列し, その語順傾向が佐伯の分析によく一致することを確かめた. 我々のモデルは音声認識における認識結果の候補の絞り込み, 名詞句の意味の推定, 文生成のための語順の基礎的情報として利用できることも述べた. このモデルの拡張としては文脈情報の扱い, 省略要素の推定, 係り助詞の扱いなどが考えられるが, その基本的な考え方についても説明した. ただし, これらの拡張に際しては具体的なデータを用いてさらに推定精度の検証をおこなう必要がある. また, 本論文ではもっとも単純な構造である単文しか対象としなかったが, 埋め込み文を含む場合, 受動化や使役化によって格交替を起こす場合, ニーニ, ガーガのように同じ格を複数とる動詞の扱いなども含めてモデルの拡張を検討する必要がある.

第5章では, 概念階層に関して, これまでの下位方向の視点とは異なる上位方向の視点の存在を明らかにし, その表現方法として概念の視点表現を提案した. また, 視点表現で表現された概念同士の単一化も定義した. 視点表現を用いると, 自然言語を処理する際に問題となるあいまい性を含む表現が可能となる. また, 視点表現の単一化は一部の照応参照の解消に利用できる. 本論文で提案した視点は, 概念間に上位/下位関係があることを前提としている. たとえば, 次の 2 つの文中に現れる “ホテル” はそれぞれ右に併記した視点表現で表現できる.

ホテルで少女と会う	∗(ホテル)\∗(場所)
ホテルを建てる	∗(ホテル)\∗(建築物)

ここで, ∗(ホテル) は∗(場所), ∗(建築物) の両方を上位概念として持つことを仮定しており, 視点表現によってどちらの上位概念の性質を継承するかを選択している. そして, この場合に選択した上位概念の性質はすべて継承されることになる. 現在, 視点表現の考え方は, 上位/下位関係を持たない概念間にも視点表現を考え, これを比喩的な表現の理解に利用しようという研究に発展している.[7,9,73]

第6章では, 概念体系の各概念をどのように設定するかという問題を論じた. 具体的には対訳辞書で定義されている各語義を概念の近似として用い, 2 言語間の語義の対応関係を機械可読な対訳辞書の対から機械的に抽出する方法を示し, その実現可能性について検討した. 本手法では, まず 2 言語間の対訳辞書を翻訳グラフでモデル化し, 3 種類 (A 型, B 型, C 型) の翻訳回路を抽出する. そして,

抽出した翻訳回路に基づき語義対応を求めるという手順をとる．この手法を市販の英和, 和英辞典の一部と EDR の対訳辞書の組にそれぞれ適用し, 語義対応の抽出実験をおこなった．その結果, いずれの実験においても, A 型の翻訳回路からは 98%以上の正答率で正しい語義対応を抽出することができた．今回の実験では, 対訳辞書中の訳語が 1 語だけからなる場合を対象に実験をおこなったが, 実際には, 訳語として句や節が与えられている場合が非常に多い．このようなものをどのように扱うか, 今後さらに検討する必要がある．

謝辞

本論文をまとめるにあたりご指導, ご助言いただきました東京工業大学工学部田中穂積教授に深く感謝申し上げます. また, 東京工業大学工学部当麻喜弘教授, 志村正道教授, 片山卓也教授, 米崎直樹教授, 佐伯元司助教授, 渡辺治助教授からは論文に対する貴重なご意見をいただきました. 深く感謝申し上げます.

多くの方々のおかげでこの論文を完成することができました. 第6章 の実験で使用した辞書データを提供してくださいました日本電子化辞書研究所の横井俊夫所長, 辞書データに関する技術的な助言をしていただきました日本電子化辞書研究所の三池誠司氏, LangLAB, LG^{2}の実装に関して協力していただきました東京工業大学田中研究室の岩山真氏, 論文を精読し, 不備な点を指摘していただきました東京工業大学田中研究室の乾健太郎氏, Chomsky の GB 理論についてご教授いただきました東京工業大学工学部の長谷川宏講師, Cornell 大学の柳田優子氏, 日頃熱心に討論していただく横浜国立大学工学部の田村直良助教授, ソニーコンピュータサイエンス研究所の大澤英一氏, 長尾確氏, 東京工業大学工学部の奥村学氏. そして, 東京工業大学田中研究室の学生諸氏ならびに秘書の方々には公私にわたりお世話になりました. みなさんどうもありがとうございました.

参考文献

[1] 井上和子 (編). 日本語の基本構造. 三省堂, 1983.

[2] 奥村学, 田中穂積. 自然言語解析における意味的曖昧性を増進的に解消する計算モデル. 人工知能学会誌, 4(6), 1989.

[3] 奥村学. 日本語理解のための計算モデルに関する研究. 博士論文, 東京工業大学, 1989.

[4] 岡田美智男, 伊藤彰則, 牧野正三, 城戸健一. 構文駆動型連続 DP 法による連続音声中からの活用語のスポッティング. 電子情報通信学会論文誌, J70-D(12), 1987.

[5] 荻野綱男. シソーラスについて. ソフトウェア文書のための日本語処理の研究 –5, pp. 1–61, 情報処理振興事業協会, 1983.

[6] 荻野綱男. シソーラスの作成の問題点. 言語, 6(5):64–71, 1987.

[7] 岩山真, 徳永健伸, 田中穂積. 比喩を含む言語理解における視点の役割. 情報処理学会 自然言語処理研究会, NL73-7, 1989.

[8] 岩山真, 徳永健伸, 田中穂積. *LangLAB User's Manual.* 東京工業大学 工学部 情報工学科 田中研究室, 第 1 版, 1989.

[9] 岩山真, 徳永健伸. 「リンゴのような頬」は赤いか? — 自然言語理解における顕現性の役割—. 認知科学会 学習と対話研究分科会, SIGLAL 90-2:1–12, 1990.

[10] 岩山真, 徳永健伸, Quek Chee Huei, 田中穂積. 自然言語処理のための英語文法. 情報処理学会第 37 回全国大会, pp. 1100–1101, 1988.

[11] 金田一京助, 他 (編). 新明解国語辞典. 三省堂, 第 3 版, 1982.

[12] 久野すすむ. 談話の文法. 大修館書店, 1978.

[13] 今野聡, 奥村学, 田中穂積. ボトムアップ構文解析システム BUP の高速化. 日本ソフトウエア科学会第 1 回大会論文集, pp. 3A–2, 1984.

[14] 今野聡, 田中穂積. 左外置を考慮したボトムアップ構文解析. コンピュータソフトウエア, 3(2):115–125, 1986.

[15] 佐伯哲夫. 現代日本語の語順. 笠間書院, 1975.

[16] 堺和宏, 徳永健伸, 田中穂積. シソーラス作成支援ツールに関する基礎的考察. 情報処理学会第 35 回全国大会予稿集, pp. 1801–1802, 1987.

[17] 堺和宏. 自然言語の意味処理のための辞書に関する研究. 修士論文, 東京工業大学, 1988.

[18] 児玉徳美. 語順の普遍性. 山口書店, 1987.

[19] 三上章. 象は鼻が長い. くろしお出版, 1960.

[20] 小島義郎, 竹林滋 (編). ライトハウス和英辞典. 研究社, 1984.

[21] 松本裕治, 杉村領一. 論理型言語に基づく構文解析システム SAX. コンピュータソフトウェア, 3(4):4–11, 1986.

[22] 上脇正, 田中穂積. 辞書の TRIE 構造化と熟語処理. Logic Programming Conference'85, pp. 329–340, ICOT, 1985.

[23] 清野正樹. 概念辞書における概念の安定化の方法. 第 3 回人工知能学会全国大会, pp. 383–386, 1989.

[24] 石崎俊, 井佐原均, 橋田浩一, 内田ユリ子, 横山晶一. 文脈理解のための概念記述法. 情報処理学会 自然言語処理研究会, NL64-7, 1987.

[25] 石崎俊, 井佐原均. 文脈情報翻訳システム CONTRAST. 情報処理, 30(10):1240–1249, 1989.

[26] 石川彰. 言語理論の新しい動向. 自然言語の基礎理論, pp. 1–50, 共立出版, 1986.

[27] 石綿敏雄, 荻野孝野. 結合価からみた日本文法. 文法と意味 I, 朝倉書店, 1983.

[28] 新村出 (編). 広辞苑. 岩波書店, 1976. 第 2 補訂版.

[29] 大塚高信, 中島文雄. 新英語学辞典. 研究社, 1982.

[30] 大野晋, 浜西正人. 角川類義語新辞典. 角川書店, 1981.

[31] 中島文雄. 日本語の構造 – 英語との対比 –. 岩波新書第 373 巻, 岩波書店, 1987.

[32] 鶴丸弘昭, 内田彰, 日高達, 吉田将. 国語辞典からの情報抽出とその構造化. 情報処理学会 自然言語処理研究会, NL43-6, 1984.

[33] 竹林滋, 小島義郎 (編). ライトハウス英和辞典. 研究社, 1984.

[34] 徳永健伸, 奥村学, 田中穂積. 概念階層への視点の導入. 情報処理学会論文誌, 30(8):970–975, 1989.

[35] 徳永健伸, 岩山真, 乾健太郎, 田中穂積. 日本語語順の推定モデルとその応用. 情報処理学会 自然言語処理研究会, NL81-2, 1991.

[36] 徳永健伸, 岩山真, 田中穂積, 上脇正. 自然言語解析システム LangLAB. 情報処理学会論文誌, 29(7):703–711, 1988.

[37] 徳永健伸, 岩山真, 田中穂積. 視点を考慮した概念の同一化とその応用. 情報処理学会 自然言語処理研究会, NL71-8, 1989.

[38] 徳永健伸, 岩山真, 田中穂積. 論理文法におけるギャップの扱い. 情報処理学会 自然言語処理研究会, NL76-3, 1990.

[39] 徳永健伸, 田中穂積. 対訳辞書からの概念項目の自動抽出. 人工知能学会学会誌, 6(2):228–235, 1991.

[40] 内田裕士. 電子化辞書の開発. 「自然言語処理技術」シンポジウム論文集, pp. 89–98, 情報処理学会, 1988.

[41] 田村直良, 田中穂積. 意味解析に基づく並列名詞句の構造解析. 情報処理学会 自然言語処理研究会, NL59-2, 1987.

[42] 田中穂積, 奥村学, 小山晴夫. オブジェクトの同一性判定および同一化アルゴリズムとその応用. 日本ソフトウエア科学会第 2 回大会論文集, pp. 2A–2, 1985.

[43] 田中穂積, 仁科喜久子. 上位下位関係シソーラス ISAMAP1 の作成. 情報処理学会 自然言語処理研究会, NL64-4, 1987.

[44] 田中穂積. 自然言語解析の基礎. 産業図書, 1989.

[45] 田中穂積, 他. 機械翻訳における中間言語方式をめぐって. 人工知能学会誌, 4(6):49–58, 1989.

[46] 電子化辞書研究所. 単語辞書 *(第 2版)*. TR-006, 電子化辞書研究所, 1988.

[47] 木下聡, 佐野洋, 浮田一男, 天野真家. 文脈理解のための知識の表現と推論. Logic Programming Conference '88, pp. 205–212, ICOT, 1988.

[48] 野村浩郷, 田中穂積 (編). 機械翻訳 *bit* 別冊. 共立出版, 1988.

[49] 北原保雄. 日本語動詞の研究. 大修館書店, 1981.

[50] 林大. 分類語彙表. 秀英出版, 1966.

[51] 林達也, 宮俊司, 坂巻利哉, 吉田健一. 横型トップダウン文解析システムの実現と評価. 情報処理学会 自然言語処理研究会, NL74-9, 1989.

[52] 林達也. 拡張 CFG とその構文解析法 YAPX について. 情報処理学会論文誌, 29(5):480–487, 1988.

[53] 計算機用日本語基本動詞辞書. 情報処理振興事業協会, 1986.

[54] H. Abramson and V. Dahl. *Logic Grammars*. Springer-Verlag, 1989.

[55] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, 1983.

[56] A. V. Aho and J. D Ullman. *The Theory of Parsing, Translation, and Compiling*. Volume I & II, Prentice Hall, 1972.

[57] D. G. Bobrow and T. Winograd. An overview of KRL. *Cognitive Science*, 1:3–46, 1977.

[58] R. J. Byrd, N. Calzolari, M. S. Chodorow, and M. S. Klavans, J. L. Neff. Tools and methods for computational lexicology. *Computational Linguistics*, 13(3-4):219–240, 1987.

[59] L. R. Chapman. *Roget's International Thesaurus (Fourth Edition)*. Harper & Row, 1984.

[60] N. Chomsky. *Lectures on Government and Binding*. Foris Publications, 1981.

[61] A. Colmeraure. Metamorphosis grammar. In *Natural Language Communication with Computers*, pp. 133–190, Springer-Verlag, 1978.

[62] V. Dahl. *Discontinuous Grammars.* Technical Report TR88-26, CSS/LCCR, 1988.

[63] V. Dahl and H. Abramson. On gapping grammars. In *Proceedings of the Second International Logic Programming Conference*, pp. 77–88, 1984.

[64] V. Dahl and P. Massicotte. *Meta-programming for Discontinuous Grammars.* Technical Report TR88-25, CSS/LCCR, 1988.

[65] V. Dahl and M. C. McCord. Treating coordination in logic grammars. *American Journal of Computational Linguistics*, 9(2):69–91, 1983.

[66] V. Dahl and P. Saint-Dizier. *Constrained Discontinuous Grammars : A linguistically motivated tool for processing language.* Technical Report, INRIA, 1986.

[67] EDR. *Concept Dictionary.* Technical Report TR-009, Japan Electronic Dictionary Research Institute, 1988.

[68] S. Fong and R. C. Berwick. New approach to parsing conjunctions using Prolog. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 118–126, 1985.

[69] T. Y. Galloway. TAXI: A taxonomic assistant. In *the Proceedings of the National Conference on Artificial Intelligence*, pp. 416–420, 1987.

[70] G. Gazdar and A. F. Pullum. *Generalized Phrase Structure Grammar: A Theoretical Synopsis.* Indiana University Linguistics Club, 1982.

[71] L. Hirschman. Conjunction in meta-restriction grammar. *The Journal of Logic Programming*, 3(4):299–328, 1986.

[72] E. H. Hovy. *Generating Natural Language under Pragmatic Constraints.* Lawrence Erlbaum Associates, 1988.

[73] M. Iwayama, T. Tokunaga, and H. Tanaka. A method of calculating the measure of salience in understanding metaphors. In *the Proceedings of the National Conference on Artificial Intelligence*, 1990.

[74] M. B. Kac and T. C. Rindflesch. Coordination in reconnaissance-attack parsing. In *the Proceedings of the International Conference on Computational Linguistics*, pp. 285–290, 1988.

[75] N. Kimura. Right node raising: A null anaphor analysis. *English Linguistics*, 3:118–133, 1986.

[76] D. Lenat, M. Prakash, and M. Shepherd. CYC: Using common sense knowledge to overcome brittleness and knowledge acquisition bottlenecks. *AI Magazine*, 6(4):65–85, 1986.

[77] C. D. Lummis. *The Last Badger*. Syobunsya, 1988.

[78] J. Lyons, 國廣哲彌監訳. 理論言語学. 大修館書店, 1973.

[79] Y. Matsumoto. *Natural Language Parsing Systems based on Logic Programming*. PhD thesis, Kyoto University, 1989.

[80] Y. Matsumoto, H. Tanaka, H. Hirakawa, H. Miyoshi, and H. Yasukawa. BUP: A bottom-up parser embedded in Prolog. *New Generation Computing*, 1(2):145–158, 1983.

[81] M. McCord. Natural language processing in prolog. In Adrian Walker, (ed), *Knowledge Systems and Prolog*, chapter 5, pp. 291–402, Addison-Wesley, 1987.

[82] M. C. McCord. Using slots and modifiers in logic grammars. *Artificial Intelligence*, 18(3):327–367, 1982.

[83] C. S. Mellish. *Computer Interpretation of Natural Language Descriptions*. Ellis Horwood, 1985.

[84] K. Mukai and H. Yasukawa. Complex indeterminates in Prolog and its application to discourse models. *New Generation Computing*, 3(4):441–466, 1985.

[85] H. Musha. A new predictive analyzer of English. In *the Proceedings of the International Conference on Computational Linguistics*, pp. 470–472, 1986.

[86] K. Nagao. Constraints and preferences: Integrating grammatical and semantic knowledge for structural disambiguation. In *the Proceedings*

of Pasific Rim International Conference on Artificial Intellegence '90, 1990.

[87] U. Nilsson. AID: An alternative implementation of DCGs. *New Generation Computing*, 4(4):383–399, 1986.

[88] S. Nirenburg and V. Raskin. The subworld concept lexicon and the lexicon management system. *Computational Linguistics*, 13(3-4):276–289, 1989.

[89] T. Okunishi, Y. Sugimura, R. Matsumoto, N. Tamura, T. Kamiwaki, and H. Tanaka. Comparison of logic programming based natural language parsing systems. In V. Dahl and P. Saint-Dizier, (eds), *Natural Language Understanding and Logic Programming, II*, pp. 1–14, North-Holland, 1988.

[90] F. C. N. Pereira. Extraposition Grammars. *American Journal of Computational Linguistics*, 7(4):243–256, 1981.

[91] F. C. N. Pereira and D. H. D. Warren. Definite Clause Grammars for language analysis – A survey of the formalism and a comparison with Augmented Transition Networks. *Artificial Intelligence*, 13(3):231–278, 1980.

[92] R. Quirk, S. Greenbaum, G. Leech, and Jan Svartvik. *A Grammar of Contemporary English*. Longman, 1972.

[93] I. Sag, G. Gazdar, T. Wasow, and S. Weisler. *Coordination and How to Distinguish Categories*. Technical Report CSLI-84-3, CSLI, 1984.

[94] R. C. Schank. *Conceptual Information Processing*. Volume 3 of *Fundamental Studies in Computer Science*, North-Holland, 1975.

[95] C. Sedogbo. A meta grammar for handling coordination in logic grammars. In *Proceedings of the Conference on Natural Language Understanding and Logic Programming*, pp. 137–150, 1984.

[96] P. Sells. *Lectures on Contemporary Syntactic Theories*. CSLI, Stanford University, 1985.

[97] S. M. Shieber. *An Introduction to Unification-based Approaches to Grammar*. CSLI, Stanford University, 1986.

[98] E. P. Stabler, Jr. Restricting logic grammars with Government-Binding theory. *Computational Linguistics*, 13(1-2):1–10, 1987.

[99] T. Tokunaga, M. Iwayama, T. Kamiwaki, and H. Tanaka. LangLAB: A natural language analysis system. In *the Proceedings of the International Conference on Computational Linguistics*, 1988.

[100] T. Winograd. *Language as a Cognitive Process.* Volume 1:Syntax, Addison-Wesley, 1983.

[101] T. Winograd. *Understanding Natural Language.* Academic Press, 1972. PhD Thesis.

[102] W.A. Woods. An experimental parsing system for transition network grammar. In R. Rustin, (ed), *Natural Language Processing*, pp. 145–149, Algorithmic Press, 1973.