論文 / 著書情報
Article / Book Information

題目(和文)	日本語理解のための計算モデルに関する研究
Title(English)	
著者(和文)	奥村学
Author(English)	Manabu Okumura
出典(和文)	学位:工学博士, 学位授与機関:東京工業大学, 報告番号:甲第2048号, 授与年月日:1989年3月26日, 学位の種別:課程博士, 審査員:田中　穂積
Citation(English)	Degree:Doctor of Engineering, Conferring organization: Tokyo Institute of Technology, Report number:甲第2048号, Conferred date:1989/3/26, Degree Type:Course doctor, Examiner:
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

日本語理解のための計算モデルに関する研究

1989年　1月

指導教官　　田中穂積 教授

東京工業大学大学院理工学研究科

情報工学専攻

奥村 学

概要

コンピュータで自然言語を理解するということは，文の並びからなる文章を解析し，文章の意味する表現を得ることである．従って，自然言語理解では，自然言語の解析が重要な問題になる．一般に，自然言語理解のための解析過程は，形態素解析，構文解析，意味解析，文脈解析の４つのステップに大別される．

コンピュータによる自然言語理解で最も困難な問題は，文に含まれる様々な曖昧性をどのように解消するかということである．自然言語に曖昧性が遍在するのは，文を構成する各部分の情報がどれも部分的で，文脈情報や，さらには我々が持っている一般常識を用いないと十分に曖昧性を解消できないことに起因している．上で述べた４つの解析ステップは，様々なレベルの曖昧性を解消するステップであると考えられる．これまでの研究のように，文を解析する際に，４つのステップを独立に実行する手法を採ると，各ステップでの曖昧性解消はそのステップとそのステップに先行する情報だけを用いて実行せざるをえず，例えば，意味解析結果を構文解析過程にフィードバックすることができないので，構文解析過程では無駄な解析を重ね，曖昧性が十分に解消できないだけでなく，効率のよい解析が不可能になり，解析速度の低下を招いてしまう．

一方，人間の言語理解過程を考えてみると，これら４つの解析ステップを１つ１つ順に実行しているわけではなく，融合して実行しているように思われる．そして，文を左から右へ読み進みながら，文の部分を部分的に解析した結果をそれ以後の解析に利用して，次第に曖昧性を解消し，妥当なものに絞り込む処理を行っていると思われる．

このような自然言語理解の計算モデルとして，本論文では，上で述べた４つの解析ステップを融合して実行し，文を解析する過程で得られる情報（制約）を増進的に蓄積し，増進的に曖昧性を解消していく計算モデルを提案する．我々は，これを増進的曖昧性解消モデルと呼ぶ．増進的曖昧性解消モデルを実現する際には，以下に示す３つの問題を解決しなければならない：

- 曖昧性を含んだ表現をどのように形式化するか
- その表現を用いてどのように増進的に曖昧性を解消していくか
- 曖昧性解消にどのような情報を用いるか

文の解析中途では，解析を終えた部分の情報は利用できるが，これから解析しようとする文の部分の情報は利用できない．そのため，文の解析中途では，曖昧性を解消するのに十分な情報が得られないことが多く，解析の中途では未決定の部分が残された解析結果を得ることになる．この未決定の部分を含む解析結果は，解析が進み，後から得られる情報により，未決定の部分が新しく決定される表現でなければならない．本研究では，不定項という概念を提案し，このような表現を実現している．

解析中途で得られる不定項は，文を左から右へ読み進む過程で徐々に未決定の部分が決定されていく．これはBUP-UTIと呼ばれる機構によって実現する．BUP-UTIは，左から右へ情報を伝播する機構であり，文を読み進む過程で伝播された不定項に対して，増進的に曖昧性を解消する．曖昧性解消は不定項上での拡張ユニフィケーション機構を導入して実現する．

曖昧性解消を行うための情報は，最終結果を絞り込むための制約とみなすことができる．新しいプログラミング・パラダイムとして最近登場した制約プログラミングは，「制約を能動的に用いて」，探索空間を段階的に小さくしていく手法であり，次の特徴を持っている：

- バックトラックしない
- 実行が進むにつれ，探索空間が徐々に小さくなる

以上のことから，増進的曖昧性解消モデルは制約プログラミングの枠組みで捉えることができる．なぜなら，増進的曖昧性解消モデルの次の特徴，

- 左から右への後戻りのない解析，
- 得られた制約による増進的曖昧性解消

は，上に示した制約プログラミングの特徴とよく整合するからである．従って，本論文で提案する計算モデルは，論理型プログラム言語Prologの拡張として，制約プログラミングを実現したものと考えることができる．

目 次

図目次

表 目 次

第 1 章

序論

1.1 本研究の目的と背景

コンピュータで自然言語を理解するということは，文の並びからなる文章を解析し，文章の意味する表現を得ることである．従って，自然言語理解では，自然言語の解析が重要な問題になる．本研究の目的は，この問題に対する１つの新しい解決策を提案し，それをコンピュータ上にプログラムとして具体化し，有効性を実証するとともに，問題点を明らかにすることである．

一般に，自然言語理解のための解析過程は次の４つのステップに大別される：

1. 形態素解析
2. 構文解析
3. 意味解析
4. 文脈解析

形態素解析では，文を構成する単語の認定を行なう．単語の認定は，それ以後の解析のベースとなるものであるから重要である．特に，単語間の区切りを空白で明示しない日本語では，形態素解析は重要なステップである．構文解析では，認定した単語がその言語で許容される並びをしているかどうかを文法を用いて検査する．意味解析では，文の意味的な妥当性を検査し，文の意味表現を抽出する．文脈解析では，１文単位であった前出の３つの解析ステップと異なり，先行する文に含まれる情報（文脈情報）を用いたより深い意味解析を行なう．

これまでの多くの自然言語解析の研究では，これら４つの解析ステップを独立なものとして順に実行する手法を用いていた．形態素解析が終了して始めて構文解析を開始し，構文解析が終了して始めて意味解析を開始するわけである．このような４つのステップを逐次的に実行する手法が採られたのは，これら４つの解析ステップを融合させる技術が十分でなかったこと，未成熟な意味解析，文脈解析技術とは独立に，形態素解析，構文解析を研究する傾向が強かったためであ

る．しかし，この手法では，各解析ステップ間の相互作用は，1 → 2 → 3 → 4の一方向に限られる．従って，例えば，意味解析結果を構文解析過程にフィードバックすることができない．

コンピュータによる自然言語理解で最も困難な問題は，文に含まれる様々な曖昧性をどのように解消するかということである．自然言語に曖昧性が遍在するのは，文を構成する各部分の情報がどれも部分的で，文脈情報や，さらには我々が持っている一般常識を用いないと十分に曖昧性を解消できないことに起因している．上で述べた4つの解析ステップは，2章で述べる様々なレベルの曖昧性を解消するステップであると考えられる．これまでの研究のように，文を解析する際に，4つのステップを独立に実行する手法を採ると，各ステップでの曖昧性解消はそのステップとそのステップに先行する情報だけを用いて実行せざるをえず，先に述べたように，意味解析結果を構文解析過程にフィードバックすることができないので，構文解析過程では無駄な解析を重ね，曖昧性が十分に解消できないだけでなく，効率のよい解析が不可能になり，解析速度の低下を招いてしまう．

一方，人間の言語理解過程を考えてみると，これら4つの解析ステップを1つ1つ順に実行しているわけではなく，融合して実行しているように思われる．そして，文を左から右へ読み進みながら，文の部分を部分的に解析した結果をそれ以後の解析に利用して，次第に曖昧性を解消し，妥当なものに絞り込む処理を行っていると思われる．

このような自然言語理解の計算モデルとして，本論文では，上で述べた4つの解析ステップを融合して実行し，文を解析する過程で得られる情報（制約）を増進的に蓄積し，増進的に曖昧性を解消していく計算モデルを提案する．我々は，これを増進的曖昧性解消モデルと呼ぶ．増進的曖昧性解消モデルを実現する際には，以下に示す3つの問題を解決しなければならない：

- 曖昧性を含んだ表現をどのように形式化するか
- その表現を用いてどのように増進的に曖昧性を解消していくか
- 曖昧性解消にどのような情報を用いるか

文の解析中途では，解析を終えた部分の情報は利用できるが，これから解析しようとする文の部分の情報は利用できない．そのため，文の解析中途では，曖昧性を解消するのに十分な情報が得られないことが多く，解析の中途では未決定の部分が残された解析結果を得ることになる．この未決定の部分を含む解析結果は，解析が進み，後から得られる情報により，未決定の部分が新しく決定される表現でなければならない．本研究では，2章で述べる不定項という概念を提案し，このような表現を実現している．

解析中途で得られる不定項は，文を左から右へ読み進む過程で徐々に未決定の部分が決定されていく．これは3章で述べるBUP-UTIと呼ばれる機構によって実現する．BUP-UTIは，左から

右へ情報を伝播する機構であり，文を読み進む過程で伝播された不定項に対して，増進的に曖昧性を解消する．曖昧性解消は不定項上での拡張ユニフィケーション機構を導入して実現する．

曖昧性解消を行うための情報は，最終結果を絞り込むための制約とみなすことができる．新しいプログラミング・パラダイムとして最近登場した制約プログラミング[73]は，「制約を能動的に用いて」，探索空間を段階的に小さくしていく手法であり，次の特徴を持っている：

- バックトラックしない
- 実行が進むにつれ，探索空間が徐々に小さくなる

以上のことから，増進的曖昧性解消モデルは制約プログラミングの枠組みで捉えることができる．なぜなら，増進的曖昧性解消モデルの次の特徴，

- 左から右への後戻りのない解析，
- 得られた制約による増進的曖昧性解消

は，上に示した制約プログラミングの特徴とよく整合するからである．従って，本論文で提案する計算モデルは，論理型プログラム言語 Prolog の拡張として，制約プログラミングを実現したものと考えることができる．

本論文で提案する増進的曖昧性解消モデルと関連する研究として，Hirst[80]の研究があるが，2章で説明するように，本論文で提案する増進的曖昧性解消モデルの計算機構の一部が実現されているに過ぎない．拡張ユニフィケーションに関する研究としては，Sowa[108]，CIL[63]，赤間[30]があるが，これらは本論文で提案する拡張ユニフィケーションのサブセットである．

本論文で提案する増進的曖昧性解消モデルの概要およびシステムの構成を図 1.1に示す．図 1.1に示したモデルでは，BUP-UTI を用いた構文解析ステップで，形態素解析も同時に実行し，構文解析で用いる文法中に付加されたプログラムにより，意味解析ステップを呼び出す．そして，意味解析ステップから必要に応じて文脈解析ステップを呼び出す．このとき，不定項として表現した曖昧性を含む解析の中途経過が，左から右へ（文頭から文末へ）BUP-UTI を用いて伝播され，その過程で様々な情報の相互作用が起こり，曖昧性が徐々に解消していく．

1.2 本論文の構成

2章では，自然言語理解で解消しなければならない曖昧性のタイプを分類するとともに，曖昧性のある解析の中途経過の表現形式として不定項を提案する．また，曖昧性解消の基本計算機構として拡張ユニフィケーション機構を提案する．また，以上を用いて，意味解析で扱う曖昧性の１つである語義的曖昧性を解消する方法について述べる．

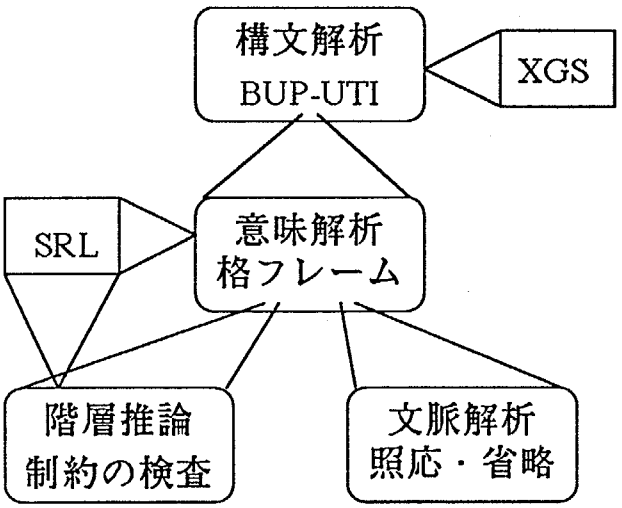


図 1.1: システム構成

3章では，増進的曖昧性解消を行うための情報伝播機構である BUP-UTI について述べる．また，構文解析レベルの曖昧性として，構造的曖昧性の解消方法について述べる．

4章では，増進的曖昧性解消の基礎となる意味解析用の辞書（フレーム）の記述言語として SRL を提案する．また，それがどのように意味解析に応用されるかについても説明する．

5章では，4章の辞書エントリ中に記述された選択制限（制約）の検査を行なう時に必要な，概念階層上での推論手続きについて説明する．また，この推論を効率的に行なう新しい手法を提案する．この手法は，並列に推論を行うことを可能にする．

6章では，2章で分類した曖昧性のタイプうち，文脈解析レベルの曖昧性である照応，省略の問題を増進的曖昧性解消モデルを用いてどのように解決するかについて述べる．そこでは，統語的な制約や談話に関する制約を用いて，照応，省略の候補集合から候補をもっともらしさの順に選択する手法について説明する．

最後に7章では，本論文の結論及び今後の研究課題について述べる．

本論文では，中学校の理科の教科書から引用した実験に関する文章を修正を加えず用いて解析と実験を行った．これらは付録 Aに示した．

第 2 章

増進的曖昧性解消モデル

2.1 はじめに

自然言語解析の中心的課題は，文に含まれる様々な曖昧性をどのように解消するかということである．自然言語解析に曖昧性が遍在するのは，文を構成する各部分の情報がどれも部分的で，他の部分の情報や，文脈などの文外の情報が得られないと曖昧性なく決定できない場合があることに起因している．文を解析する際に，その文に含まれる曖昧性を文末まで解消しない方法を採ると，文全体の曖昧性は組合せ的に爆発してしまう．そのため，文末まで読み進んだ後，その文全体の曖昧性を解消しようと試みるシステムは非現実的である．また，文に含まれる曖昧性がその文の終りまで読み進めば十分に解消できるという保証はない．なぜなら，文脈などの文外の情報を考慮しなければ解消できない曖昧性が存在するからである．従って，残された手法は，文を解析する過程で得られる情報（制約）を増進的に蓄積し，できるかぎり曖昧性を解消していく増進的曖昧性解消しかないのである．

また，曖昧性を人間がどのようにして解消しているかを考えてみると，曖昧性が文の終りなどのある時点で一気に解消されるのではなく，文を左から右へ読み進む過程で徐々に曖昧性が解消されていくと考えるのが自然である．我々はこのような計算モデルを増進的曖昧性解消モデルと呼ぶ．

新しいプログラミング・パラダイムとして最近登場した制約プログラミング[73]は，論理プログラミングの次のような問題点を解決する試みである．

論理プログラミングにおけるプログラムは，制約プログラミングの場合と同様，変数間の依存性（制約）を表現したものと考えることができ，表現された制約集合を満足するように変数集合に値を与えることが問題の解決となる．論理プログラミングでは（論理型プログラミング言語 Prolog を例に考えると），この問題解決をバックトラック探索に基づいて行なう．そのため，制約の処理過程は，数え上げ手続きが変数に値を与え，その値が制約集合を満足するかどうかを調べることになる（ generate-and-test ）．これは，満足するかどうかの検査にしか制約を用いないことから，

「制約の受動的使用」と呼ばれる．この手法では，同じ失敗を何度も繰り返すことから，効率が悪いことは直感的にも明らかである．

これに対し，「制約を能動的に用いて」，変数の可能な値集合を制限し，探索空間を段階的に小さくしていくのが，制約プログラミングの処理である．この手法は，

- バックトラックすることがない
- 処理が進むにつれ，探索空間が徐々に小さくなる

ことから，効率面で大幅な改善が期待できる．Prologの拡張システムの中には，この制約プログラミング・パラダイムとして考えられるものも存在する．本論文で述べる増進的曖昧性解消モデルのように，拡張ユニフィケーションを用いたシステムはその1例である．

次節では，日本語に出現すると考えられる曖昧性を概観する．

2.2 日本語の曖昧性についてのサーベイ

本節では，日本語に出現すると考えられる曖昧性を階層的に分類する[8,40,97,38]．まず大きく以下のように分類する．

- 漠然性(vagueness)
- 曖昧性(ambiguity)
  - 指示的(referential)
  - 言語的(linguistic)
    - 語彙的(lexical)
    - 文的(sentential)

どのあたりまでがその適用範囲として許容されるかが明らかでない場合を漠然性といい，1つの表現に関して2つ以上の別個の可能性が存在し，どちらとも決められないような場合を曖昧性という．「数年」という表現が実際にどれぐらいの年数を表すかは明らかでなく，これは漠然性の1例である．これは，後述する多義語のような曖昧性と対照をなす．漠然性について本論文では特に言及しない．以後は，曖昧性についてのみ分類を進める．

曖昧性は，指示的なものと言語的なものに分類する．言語表現に関する曖昧性を言語的といい，言語表現が指示する指示対象に関する曖昧性を指示的という．言語的曖昧性は，語彙的なものと文的なものに分類する．指示的，語彙的，文的曖昧性については，2.2.1, 2.2.2, 2.2.3節でそれぞれ細分類する．

2.2.1 指示的曖昧性

指示的曖昧性には，次の４つのものが考えられる．

1. Indexicals

   「私」,「昨日」などのように，発話状況(utterance situation)[1][65]が異なると，その指示するものが異なる言語表現

2. 名詞句の指示対象に関して

   次の例文を考えてみよう．

   恐竜は絶滅した．

   太郎はりんごが好きだ．

   太郎はフランス人と結婚した．

   下線の３つの名詞句の指示する対象は以下のようにその性質が異なる．

   「恐竜」「恐竜」というクラス（種類）

   「りんご」「りんご」の個体すべて

   「フランス人」「フランス人」のある個体

   このように日本語の名詞句は次の３種類の指示対象を指示する可能性がある．

   - クラス
   - クラス中の個体すべて
   - ある個体

   また，これとは別に，名詞句を内包的(intensional)に解釈するのか，外延的(extensional)に解釈するのかということから，次の２通りの読みが存在することがある．

   - *de dicto* reading
   - *de re* reading

   次の文を考えてみよう．

   にわとりは飛ぶと太郎は信じている．

[1]「いつ，どこで，誰が，誰に，どのようなことを」発話したかを記述した状況

この文の解釈として，次の２つの論理式が得られる．

$$\text{信じている}(\text{太郎}, \hat{}\exists x[\text{にわとり}(x) \wedge \text{飛ぶ}(x)]) \tag{2.1}$$

$$\exists x[\text{にわとり}(x) \wedge \text{信じている}(\text{太郎}, \hat{}\text{飛ぶ}(x))] \tag{2.2}$$

2.1 式が *de dicto* 読みと言われるもので，「太郎はにわとりと呼ばれるものが飛ぶと信じている」という解釈を表す．また，2.2 式は *de re* 読みと言われ，「太郎はあるものが飛ぶと信じており，そのものはにわとりと呼ばれる」という解釈を表す．

3. 照応参照（指示語の指示対象の同定）に関して
   「彼」，「それ」，「その男」など，人称代名詞，指示代名詞，指示形容詞を含む名詞句は，文脈中，あるいは「発話の場」に存在するものを指示する[50,60]．文脈中，あるいは「発話の場」には，この指示対象となる候補が一般に複数存在しうるので，その同定には曖昧性が派生する．

4. 省略補完（省略されている格要素を補う）に関して
   照応参照の場合と同様，省略されている格要素は，文脈中，あるいは「発話の場」から補う必要があるため，その補完には曖昧性が派生する．

2.2.2 語彙的曖昧性

語彙的曖昧性は，形態論的なものと語義的なものに分類する．

形態論的曖昧性としては，用言の活用形の曖昧性がある．例えば，「行く」という単語は，終止形であるのか連体形であるのか，それだけでは判別できない．

語義的曖昧性にはつぎの２つのものがある．

1. 多義性(polysemy)

2. 同音異義性(homonymy)

「子供」という単語に，「(親に対する) 子」の意味と，「(おとなに対する少年・少女」の意味の（少なくとも）２通りの意味があることを多義といい，「はし」という単語に，「橋」と「箸」の（少なくとも）２通りの意味が考えられるということを同音異義性という．語の意味が全くわからないという点で，未知語は最も多義性が顕著である例と考えられる．また，「ある」（連体詞，動詞）などの多品詞語は，同音異義語の特殊な場合と考えられる．

用言が多義の場合，その格フレームにも曖昧性が生じる．例えば，「指す」という単語には，（少なくとも）２つの意味が考えられるが，それらはそれぞれ対応する格フレーム[2]を持つ（図 2.1,

[2] 本論文中での格フレームの記述は，4 章で説明する SRL に基づく．

```
意味：指やはりなどをあるものに向ける
格フレーム：
  [ 指す ::
    [ が $ Hum ; Pro -> 動作主 ]
    [ を $ Div -> 目標 ]]
```

Hum,Proなどは，[62]で設定された名詞句の意味素性を表す．

図 2.1:「指す」の意味と格フレーム（1）

```
意味：あることを意味する
格フレーム：
  [ 指す ::
    [ が $ Lin -> 対象 ]
    [ を $ Div -> 目標 ]]
```

図 2.2:「指す」の意味と格フレーム（2）

2.2[3]参照)．図 2.1, 2.2の格フレーム中,「が」格スロットの記述が異なることに注意してほしい．

付属語の多義性は，格スロットの曖昧性に反映される．例えば，助詞「で」には,「場所」,「時間」,「方法」,「道具」,「材料」などの意味が考えられる[15]が，これらは，図 2.3のような複数の格スロットとして記述される．

2.2.3 文的曖昧性

文的曖昧性には，次の４つのものが考えられる．

1. 分かち書きに関して

   日本語は，単語間に空白などの区切りをおかない膠着言語であるため，単語を切り出し認定する分かち書きが不可欠である．しかし，この単語の切り出し方が常に１通りであるとは限らない．「天井のねずみ」という文字列は,「「天井」「のねずみ」」,「「天井」「の」「ねずみ」」

[3] [62]より引用．

```
[ で $ isa:場所 -> 場所 ]
[ で $ isa:時 -> 時間 ]
[ で $ isa:方法 ; isa:理論，体系 -> 方法 ]
[ で $ isa:道具 ; isa:精神的活動の道具 -> 道具 ]
[ で $ isa:部品，材料 -> 材料 ]
```

図 2.3: 曖昧性のある「で」格スロットの記述

の２通りの区切り方が可能である．

2. 取りうる格に関して

通常，用言に係る名詞句は，その用言に対して何らかの格に位置していると考えられる．この格が文中に陽に表現されていない場合には，その格の決定に際して曖昧性が生じる．「書く鉛筆」の「鉛筆」のように，関係節化 (relativization)された名詞句や，「花子は太郎が好きだ」の「花子」のように，主題化 (topicalization)された名詞句は，その例である．

3. スコープの範囲

助動詞「ない」で表される否定や「すべての」とか「ある」などの連体詞で表現される限量子が，文中のどの範囲まで及ぶかが曖昧である場合がある．例えば，

パン屋に行って，パンを買わなかった．

では，下線部で表される否定の及ぶ範囲は，

(a) 文全体（「パン屋に行かず，従ってパンを買うこともしなかった」ことを意味する）

(b) 読点より後ろの節（「パン屋に行きはしたが，パンは買わなかった」ことを意味する）

のどちらなのか曖昧である．

4. 係り受け(並列) などの構造的曖昧性

複数の句（節）が存在すると，それらのうち，どれとどれが関係しているかが曖昧になる．例えば，「大きな森にある家」では，「大きな」が「森」に係るのか，「家」に係るのか曖昧である．この構造的曖昧性は，解析結果として構文木が複数得られることに反映される（図 2.4 参照）．

2.2.4 まとめ

本節では，日本語に出現すると考えられる曖昧性を階層的に分類した．次節では，指示的曖昧性および語彙的曖昧性を統一的に処理することが可能な増進的曖昧性解消モデルについて述べる．また，2.4 節では，文的曖昧性の増進的解消方法について述べる．

2.3 増進的曖昧性解消の計算モデル

本節ではまず，増進的曖昧性解消方法を取っていると考えられる研究を概観する（2.3.1 節）．2.3.2 節では，曖昧性を処理するシステムを論理型プログラミング言語(Prolog)で実現する際の問題点を指摘する．2.3.3 節では，問題点の解決策として提案されている拡張ユニフィケーションに関する他

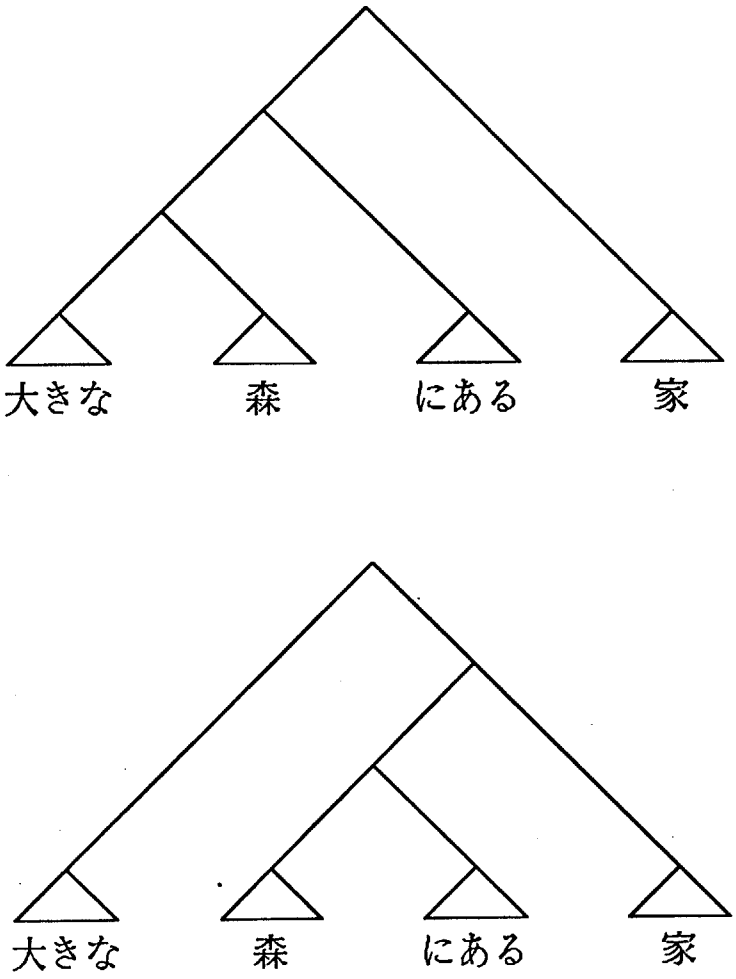


図 2.4:「大きな森にある家」の構文木

```
[ operate:
    [ cause-to-function
          agent          SUBJ
          patient        SUBJ,OBJ
          instrument     SUBJ,with
          method         by
          ... ]
    [ perform-surgery
          agent          SUBJ
          patient        upon,on
          instrument     with
          method         by
          ... ]]
```

図 2.5: 動詞'operate'のポラロイド語

の研究を概観する．そして，2.3.4 節では，本研究で提案する増進的曖昧性解消モデルについて述べる．

2.3.1 増進的曖昧性解消に関するいくつかの研究

本節では，増進的曖昧性解消方法を取るいくつかの自然言語理解研究について概観し，その問題点を指摘する．

1. Hirst[80]は，曖昧性として語彙的曖昧性および構造的曖昧性について論じ，曖昧性を解消するのに用いることができる知識を挙げている．また，ポラロイド語(Polaroid Words) という概念を用いて曖昧性を解消する手法を提案している．ポラロイド語は，3で述べる Mellish の指示実体や 2.3.3 節で述べる CIL の部分項と同じく，曖昧な単語の意味をそのまま表現することが可能なデータであり，それらがお互いに情報を交換し合うことで，曖昧性が解消されていくことになる．図 2.5[4]に，ポラロイド語の例を示す．

   ポラロイド語には次のような問題点が存在する．

   (a) 図 2.5のように，ポラロイド語は，いわば辞書そのものであるため，動詞のようにその情報量の多いものでは，解析中それらのデータを持ち回ることになり効率的でない．

   (b) ポラロイド語は各単語ごとに 1 つ必ず存在し，それらの関連により解析が進む．そのため，動詞の格フレームは，図 2.5の動詞のポラロイド語と図 2.6の前置詞のポラロイド

[4] [80]より引用．

```
[ SUBJ:
   agent          animate
   patient        thing
   instrument     physobj
   source         physobj
   destination    physobj ]

[ OBJ:
   patient        thing
   transferee     physobj ]
```

図 2.6: SUBJ, OBJ 格のポラロイド語

語の組合せによって表現される．この場合には，格フレームを通常記述する場合のように，スロットに対するその動詞に固有の選択制限が記述できず，選択制限に関する記述が非常に一般的にならざるをえない．

(c) ポラロイド語は，2.3.2 節で述べる曖昧性の表現のタイプのうち，1の方にしか対処できない．この問題点については，2.3.4 節で述べる．

2. コネクショニスト・モデル(Connectionist Models)を用いた自然言語解析の研究[112,71]では，曖昧性は抑制性リンクでそれぞれのノードが結ばれたネットワークとして表現され，解析の過程で他の部分からの情報によりそのうちのどれかが活性化されると残りのものの重みが下がり，最終的に最も妥当なものの重みが最大になるメカニズムになっている．しかし，このモデルでは，曖昧性の解消はノードの重みの順序づけ（優先順位）をすることに対応しており，曖昧性を表現したネットワークの構造は変化しない．そのため，曖昧性をネットワーク形式で表現したまますべて保持していることと等しく，曖昧性が組合せ的になると記憶容量の爆発が考えられる．

3. Mellish は,「文を読み進みながら部分的に意味解析を行ない，文全体を読み終った時点で意味解析結果が得られる」という，人間に近い意味解析手法のモデルを提案している[93]（Mellish はこの手法を早期意味解析と呼んでいる）．[93]では，この早期意味解析を用いた名詞句の意味解析について述べている．名詞句の意味解析が難しいのは，その名詞句が指示している対象がどのようなものなのかがその名詞句の情報だけでは明確でなく，従ってその対象の内部表現としてどのようなものを作ればよいかが明確でないからである．この問題点を解決するため，Mellish は，

指示実体(**reference entity**) 名詞句の指示対象の内部表現

依存性リスト(**dependency list**) 指示実体間の依存性（制約）を表現したデータ

候補集合(**candidate set**) 指示対象の候補となる実体の集合

などの概念を導入している．そして，解析の過程で依存性リストが徐々に評価され，その結果，候補集合の濃度が小さくなり，指示実体の表現が明確になっていくというメカニズムが実現されている．

2.3.2 Prolog上での曖昧性の表現

曖昧性を処理するためには，その曖昧性を表現することがまず必要になる．2.2 節で述べた指示的，語彙的曖昧性を表現するためには，次の２つのタイプの曖昧性の表現が必要である．

1. 値として可能な選択肢が複数存在し，そのうちの１つに決定できないという曖昧性の表現
   例えば，求める解は述語'p'を満足するものであり，'p'を満足する値は，1, 3, 5 であるというような場合である．2.2.2で述べた多義性などはこのタイプで表現される．このタイプで表現される曖昧性の解消は，その選択肢の数が減少することに相当する．

2. 値が十分に限定されていないという曖昧性の表現
   例えば，変数 X の値が現時点では「人間」であるが，他の情報が得られた結果，「人間」より限定された「太郎」に変化する場合である．この場合，変数 X の値は，より限定された値に変化しうるという点で曖昧性を持つ．そして，より限定された値に変化していくことが曖昧性解消を意味する．語義が不明である未知語(語義を表現する変数の値が束縛されていないことに相当する）の処理や，照応参照による指示対象の決定はこのタイプの曖昧性解消である．

この２つのタイプの曖昧性が Prolog 上でどのように表現できるかを見ていくことにする．選択肢が複数あるタイプの曖昧性の場合，Prolog では，１つの選択肢を１つの節に対応させ，複数の節として表現する．例えば，上で述べた述語'p'の場合，値として可能性のある 1, 3, 5 それぞれについて，

$$p(1).$$

$$p(3).$$

$$p(5).$$

のように記述する．求める解が曖昧である（複数存在する）ことは，ゴール

$$?-p(X).$$

を実行し，Prologのバックトラック機構によりこれらの節を順々に探索し，値として1, 3, 5を順々に得ることで判断される．しかし，この方法では，解として3を吟味している時には，それ以外の値の可能性(1, 5)についてはその存在さえわからないため，求める解が曖昧であるかどうかということは，その解析中には判断できない．従って，このような表現では，解析中に出現した曖昧性を表現しておき，それを増進的に解消していくモデルは実現できない．また，2.1節で述べたように，バックトラックを用いた論理プログラミングの問題点が指摘され，新しいプログラミング・パラダイムとして制約プログラミングが登場している．

増進的曖昧性解消を行なうためには，解析中に出現した曖昧性を陽に表現できなければならない．例えば，述語'p'の例では，

$$p((1;3;5)).$$

のようにである[5]．この表現では，'p'の値として1, 3, 5の3通りが考えられることが明示されている．このような表現はProlog上で複合項(compound term)として可能である．このように表現された曖昧性の解消は以下のような過程である．求める解の条件（制約）が'p'かつ'q'の場合を考える．

$$q((3;5;7)).$$

であるとすると，この場合，求める解は(3;5)である．ゴール

$$?-p(X),q(X).$$

を実行するとまず，'p'を満足する集合として(1;3;5)を得，この集合が次の制約'q'を満足するかどうか検査することになる．この検査は，集合(1;3;5)と(3;5;7)の積集合(intersection)を取ることに相当する．この検査はユニフィケーションにより実現されるものであるが，残念なことにPrologのユニフィケーションは，これほど強力なものではない．そのため，この表現を用いて曖昧性解消を行なうためには，ユニフィケーションを拡張する必要がある．

値が増進的に限定されていくタイプの曖昧性をPrologで表現できないことは明らかである．Prologの論理変数は1度値が束縛されると，2度とその値を書き換えられないからである[6]．このため，Prologに増進的に値が変更可能な項を導入する必要がある．

以上，述べたように，2つのタイプの曖昧性を表現し，それを用いて増進的に曖昧性を解消していくためには，Prologの項およびユニフィケーションを拡張する必要がある．次節では，Prologの項およびユニフィケーションの拡張を試みているいくつかの研究を概観する．

---

[5] ';'は選言(OR)を表現している．

[6] 値が束縛された時点までバックトラックし，値を他の選択肢に変更することは可能であるが，これは今述べている値を限定する場合にはあたらない．

2.3.3 項，ユニフィケーションの拡張に関するいくつかの研究

本節では，Prologの項，ユニフィケーションを拡張したいくつかの研究について概観し，その問題点を指摘する．

1. CIL[63]では，Prolog上に部分項という概念を導入し，Prologの項の拡張を行っている．部分項は，ラベルと値の対の集合として定義される．具体的には，

$$\{a_1/b_1, ...a_n/b_n\}$$

の形式で記述する．ここで，a_iがラベルであり，b_iが値である．部分項はその名前の通り，対象の持っている情報の一部を表現したものであり，解析が進むにつれ，ラベル-値の対が追加・蓄積されていく．しかし，部分項は，単調に増加するデータ構造にすぎないため，新しい情報が得られることにより，構造が単純になるような場合（選言，否定的データ構造）に対処できない．

2. PALにおける集合束縛変数[30]は，変数に制約としてユニファイ可能な集合を指定し，

$$X\hat{}<\text{制約}>$$

のような記述を許すものである．例えば，

$$X\hat{}\text{動物}$$

は，Xが「動物」の下位のクラスの篠体としか束縛されないことを意味する．また，

$$X\hat{}\{\text{犬,猫,豚,鶏}\}$$

のように，いくつかの要素からなる集合を記述するデータ型もある．Xが「犬」,「猫」,「豚」,「鶏」のどれかであることを記述している．問題点は，

- 3つの型の集合束縛変数がカバーする範囲が離散的である．
- 異なる型の変数同士のユニフィケーションが実行できない．

ことである．

3. [6]の条件付単一化は，変数が満足すべき条件（制約）をパターン[7]に直接付加し，それらの条件の間のユニフィケーションに基づいて解析を行ない，解となるパターンを得るメカニズムである．解は抽象的なパターンとして得られるので，解となるパターンの数え上げを行う

[7] Prologにおける項を[6]ではパターンと呼んでいる．

ためのバックトラックは一切行わない．また，ユニフィケーションの際に蓄積される条件はコンパイルされるので，条件間に矛盾があるようなパターン同士のユニフィケーションは失敗する．この手法の問題点は，解析時に述語を実行するのではなく，述語の書き換えを行い，新しい述語の定義を行う[8]ので手間がかかることである．

4. [33]は項記述という概念により，2で述べた[30]と同様，項に制約を付加できるようにして，

$$< \text{項} >:< \text{制約} >$$

のような記述を許すように，項の拡張を行っている．< 項 >は< 制約 >を満足しなければならない．この手法では，制約として記述するものが通常の述語呼び出しのため，制約の評価は通常のゴールの実行と変わらない．そのため，項記述として記述力を拡張しても，解の探索はバックトラックを用いて実行されるため，記述力の拡張の利点を生かしきれていない．また，変数を具体化する時点でしか，制約を評価しないため，変数同士のユニフィケーションでは制約を連言として結合することしか行わず，従って矛盾した制約を持つ対象間のユニフィケーションが成功してしまうことがある．

5. [89,20]は，等号に関する言明(assertion)を導入することにより，ユニフィケーションの拡張を行っている．

$$A + B = C :- C \; is \; A + B.$$

のような等式を定義することにより，

$$5 = 2 + 3$$

のようなユニフィケーションを成功させる手法である．これは，上のように定義した等式を項書き換え[44]とみなして初理を行っていると考えられる．すなわち，上の例でいえば，$2+3$を，それから計算できる（等式の右辺の条件を満足する）Cに書き換えてからユニフィケーションを行うことと等価である．

6. LFG(Lexical Functional Grammar) [86], HPSG(Head-driven Phrase Structure Grammar) [100], PATR [106] などの「ユニフィケーションに基づく文法形式」(unification-based grammar formalism)では，カテゴリなどの言語的情報をすべて素性(feature)の束として表現し，解析はそれらの素性の束の間のユニフィケーションにより実行される．この素性の束は 1で述べた CIL における部分項と同等のものである．従って，この素性の束にも選言，否定の情報を記述することはできなかった．しかし，その必要性は[87,88]に述べられており，そのアイデアも提案されている．

[8] このような手続きをプログラム変換[7]という．

2.3.4 拡張ユニフィケーションを用いた増進的曖昧性解消モデル

2.3.2 節では，指示的，語彙的曖昧性を増進的に解消するモデルを実現するには，2つのタイプの曖昧性の表現が必要であることについて述べた．また，論理型プログラミング言語上では，そのためには，項，ユニフィケーションの拡張が不可欠であることを述べた．本節では，本研究で提案する拡張ユニフィケーションを用いた増進的曖昧性解消モデルについて述べる．

2.3.1 節で述べた増進的曖昧性解消に関する研究のうち，Hirst のポラロイド語は，他の2つの研究と比較して，

- 扱える曖昧性の一般性
- 効率

の点で優れていると考えられる．本節で提案するモデルは，部分的にこの研究に基づいており，ポラロイド語をユニフィケーションの概念を用いて統一的に実現したものとみなすことができる．しかし，ポラロイド語は，2.3.2 節で述べた曖昧性のタイプのうち，1の方にしか対処できない．そのため，2のタイプの曖昧性解消である照応参照処理などはこのモデルでは扱うことができない．本論文で提案するモデルは，2.3.2 節で述べた2つのタイプの曖昧性をユニフィケーションを用いて統一的に扱うことが可能な増進的曖昧性解消モデルであり，その点でポラロイド語を包含するものである．

本モデルでは，まず項の拡張として，部分的に得られた（曖昧性のある）情報の表現形式として，不定項を導入する．増進的曖昧性解消は，部分情報が解析の過程で他の部分からの情報により洗練・付加され，情報量が増加する過程である．従って，解析の途中経過を表現する項は，情報の変化過程を十分表現できなければならない．不定項はそのような表現形式である．不定項の詳細については後述する．また，これに伴い，不定項を扱えるようにユニフィケーションを拡張する[9]．

以下では，まず増進的曖昧性解消に必要なユニフィケーションの機能について例を示す．続いて，本モデルでの表現形式である不定項および拡張されたユニフィケーションについて説明する．最後に，それらを用いた曖昧性解消の例として，選択制限を用いた曖昧性解消の例を示す．

ユニフィケーションとは2つの構造を同一視しようとすることである．例えば，「庭に紫のバラが咲いている．私はその甘いにおいの花が好きだ．」という文では，「その花」は「バラ」を指す．これは照応関係と呼ばれるが，この照応関係は「その花」と「バラ」をユニファイすることである．

図 2.7はユニフィケーションの例である．「花」と「バラ」をユニファイすると，より具体的な「バラ」がユニフィケーションの結果として得られる．これは，意味するものがより限定されたということで，タイプ2の曖昧性解消を行なっていることになる．色は「紫」と，「紫」を要素とし

[9] 拡張されたユニフィケーションについても後述する．

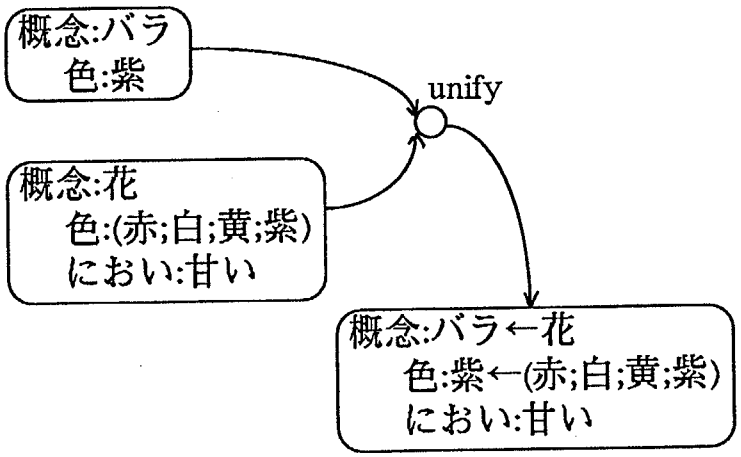


図 2.7:「紫のバラ」と「甘いにおいの花」のユニフィケーション

て持つ集合[10]をユニファイすることにより，結果として「紫」が得られる．これは，「色」という属性の値が様々なものの中から１つに決定されたということで，タイプ１の曖昧性解消を行なっていることになる．また，「色」以外に新しく得られた「におい」に関する制約が表現に付加される．新しく付加された制約により表現している対象がより限定されたわけなので，これはタイプ２の曖昧性解消を行なっていることになる．

ユニフィケーションの結果として得られた表現中，矢印は値の変化の過程を表現している．値の変化は，曖昧性解消の過程で新しく得られた制約により引き起こされるので，この値の変化過程は，それらの制約の蓄積過程を表現している．また，「におい」のように，表現をより限定するように付加された制約（属性）は，表現中に増進的に蓄積される．

本モデルにおける表現形式である不定項について述べる．不定項は，次の２種類に分類される．

1. 単純不定項

2. 複合不定項

複合不定項は，その表現するものが対象（名詞句）であるのか，出来事(動詞句）であるのかにより，次のような表現形式を取る．

1. 名詞句　　$object(Qua, Con, Slots)$

ここで，

Qua 限量子に関する情報[11]

[10] この色に関する集合は，「花」の色はこのうちのどれかであるという常識から得られる．

[11] 本論文では，限量子に関する曖昧性については扱わないので，この情報は用いないで議論を進める．そのため，以後この引数位置には無名変数('_')しか現れない．

```
unify(object(_,(バラ,_),(色:(紫,_),_)),
      object(_,(花,_),(色:((赤;白;黄;紫),_),においt:(甘い,_),_)))
                                  ↓
      object(_,(花,バラ,_),(色:((赤;白;黄;紫),紫,_),においt:(甘い,_),_))
```

図 2.8: 不定項間のユニフィケーション

Con 表す概念

Slots 持っている属性

である.

2. 動詞句　$event(Con, Slots, Pol)$

ここで,

Con 表す概念

Slots 取っている格スロット

Pol 極性[12]

である.

Slots は,

$$SlotName_1 : SlotValue_1, ..., SlotName_n : SlotValue_n$$

であり, 2.3.3 節で述べた CIL における部分項, 6の素性の束と等価であり, 情報が追加されると単純に増加していくデータ構造である.. 図 2.7に示したユニフィケーションは, 図 2.8のような不定項間のユニフィケーションとして実現される. ユニフィケーション後の不定項の属性を表す第 3 引数に「においt」に関する制約が付加されていることに注意して欲しい. 複合不定項の概念を表す部分(Con), スロット値を表す部分($SlotValue_i$)には, 単純不定項を記述する[13]. 単純不定項は,

$$(SI_1, SI_2, ..., SI_n, \_)$$

の形式であり,項の最初の値が SI_1であり,その後新しい情報が得られた結果,項の値が $SI_2, ..., SI_{n-1}$ と変化し, 現在は SI_nであることを表現する. この単純不定項により, 2.3.2 節で述べた 2のタイプの曖昧性解消が実現できる. この単純不定項により, 図 2.7の矢印で示した値の変化過程の表現を実現している. 図 2.8のユニフィケーション後の不定項の概念および色のスロット値を表す部分

[12] そのイベントが実際に成り立っている(1), または成り立っていない(0)を記述する.
[13] スロット名($SlotName_i$)にはアトムしか記述できない.

に注意して頂きたい．2.3.2 節で述べた 2のタイプの曖昧性解消はこの単純不定項により実現される．また，SI_i の値として，アトムだけでなく，

- 選言(disjunction)

$$(v_1; v_2; ...; v_n)$$

ここで，v_iはアトムである．

- 否定(negation)

$$-v$$

ここで，vはアトムまたは選言である．

を導入することで，1のタイプの曖昧性も表現できる．図 2.7のユニフィケーション後の不定項は，「花」であった概念が「バラ」に限定され，また，色も 4つの選択肢のうちから紫に決定されて，「その花」が「紫のバラ」であることを表現している．このように，不定項を用いることで，CIL などで実現されている従来の意味での（情報が追加されると項も単純に増加していく）制約蓄積だけでなく，情報が新しく得られることにより，表現しているものの構造がより簡潔になる（限定される）ような場合を統一的に扱うことができる．

次に，複合不定項を扱えるようにユニフィケーションを拡張する．複合不定項上でのユニフィケーションは，以下のようなものである．

概念部 概念階層を検索し，それらの概念が同一化可能かどうか調べる．同一化可能な場合，より具体的な概念に同一化する．

属性（スロット）部 同一スロット名の属性は同一スロット値を持たなければならない

例えば，「犬」と「猫」のユニフィケーションは，それらが背反な概念であることから失敗し，「人間」と「動物」のユニフィケーションは，「人間」が「動物」の下位概念であることから，結果として「人間」を得る．このような概念の同一化メカニズムは，Sowa[108]や Dahlgren[72]などでも用いられている．概念階層上での効率的な推論方法については，5章で述べる．

属性（スロット）の集合間のユニフィケーションは，それらの集合の和集合(union)を取る演算とみなされる．例えば，属性集合

{名前：太郎,性別：男,年齢：27}

と

{名前：太郎,趣味：音楽観賞}

のユニフィケーションの結果としては，属性集合

{名前：太郎,性別：男,年齢：27,趣味：音楽観賞}

を得る．しかし，属性集合

{名前：太郎,性別：男,年齢：27}

と

{名前：花子,趣味：編みもの}

のユニフィケーションは,「名前」スロットの値が同一化できないため失敗する．

また，単純不定項の要素として選言，否定を導入したことにより，ユニフィケーションも選言，否定を扱えるように拡張する必要がある．選言，否定を含む項上でのユニフィケーションを図 2.9[14]に示す[88,87]. 図 2.9で，矢印の右辺はユニフィケーションの結果を，*if*の右側はユニフィケーションが成功するための条件を表す．また，'\+'は Prolog の否定を表し，$(a_i \mid P(a_i))$は，Pを満足する a_iを選言でつないだ集合を表す．$union(a,b)$は，a,b の和集合を表す関数とする．

最後に，本モデルによる選択制限を用いた曖昧性解消の例を示す．

選択制限は，動詞の格フレーム中の各スロットを満たすもの[15]に関する条件であり，これが満足されない文は非文法的であると判断される．従って，選択制限は，動詞の格スロットを満たす名詞句の指示対象に制約を課する．この制約により不適格な意味は排除され，その名詞句の意味に関する曖昧性は,（部分的に）解消される．同様に，未知語の意味も，その未知語が満たす格スロットの選択制限により（部分的に）特定できる．照応参照処理においても同様に選択制限の情報が用いられるが，これに関しては 6 章で述べる．

また，動詞の格フレームは，その意味と 1 対 1 対応をしているという前提に立てば，解析の過程で名詞句が動詞の格スロットを満たす際，選択制限を用いて，複数ある格フレームのうち妥当なもののみを選択することにより，動詞の意味の曖昧性を解消することができる．

この選択制限の検査を上で述べた拡張ユニフィケーションで実現することにより，曖昧性解消を実現できる．例えば，図 2.10のような格フレームを考えてみよう．ここで，各スロットの$と→の間の部分が選択制限の記述である．図 2.10のように，SRL 形式で記述された辞書は，図 2.11のような形式のプログラムに変換され実行される．SRL 形式の辞書およびそれを用いた意味解析に関しては，4 章で述べる．図 2.11の下線部の'*unify*'が上で述べた拡張ユニフィケーションを実行する述語であり,「掛ける」の「が」格スロットの選択制限（Hum または Org または Ani である）をフィラー(V)が満足するかどうか検査する．この検査をユニフィケーションで行なうことにより，

[14] ⊗はユニフィケーションを表す演算記号とする．
[15] フィラーと呼ばれる．

$A \otimes B$:

1. $a \otimes b$(アトム同士)
   $\longrightarrow a \; if \; a == b$
   例:$a \otimes a \longrightarrow a$
   $a \otimes b \longrightarrow$ 失敗

2. $a \otimes -b$(アトムと否定)
   $\longrightarrow a \; if \; \backslash + \; unify(a, b)$
   例:$a \otimes -b \longrightarrow a$
   $a \otimes -a \longrightarrow$ 失敗

3. $a \otimes (b_1; ...; b_n)$(アトムと選言)
   $\longrightarrow a \; if \; \exists j \; unify(a, b_j)$
   例:$a \otimes (c; a; t) \longrightarrow a$
   $a \otimes (d; o; g) \longrightarrow$ 失敗

4. $(a_1; ...; a_n) \otimes -b$(選言と否定)
   $\longrightarrow (a_i \mid \backslash + \; unify(a_i, b))$
   例:$(a; b; c) \otimes -b \longrightarrow (a; c)$

5. $(a_1; ...; a_n) \otimes (b_1; ...; b_n)$(選言同士)
   $\longrightarrow (a_i \mid \exists j \; unify(a_i, b_j))$
   例:$(a; b; c) \otimes (b; c; d) \longrightarrow (b; c)$

6. $-a \otimes -b$(否定同士)
   $\longrightarrow -union(a, b)$
   例:$-a \otimes -b \longrightarrow -(a; b)$

図 2.9: 選言，否定のために拡張されたユニフィケーション

```
かける
  [ 欠ける::
    [ が $ Con -> 対象 ]]
  [ 賭ける::
    [ が $ Hum -> 動作主 ]
    [ に $ Abs -> 目標 ]
    [ を $ Qua -> 対象 ]]
  [ 掛ける::
    [ が $ Hum;Org;Ani -> 動作主 ]
    [ に $ Loc -> 場所 ]
    [ を $ Pro -> 対象 ]]
  [ 駆ける::
    [ が $ Hum;Ani -> 動作主 ]
  ( を $ Loc -> 経由 )]
```

図 2.10: 動詞「かける」の複数の格フレーム

$$sem(\text{掛ける}, \text{が} : V, In, Out) : -$$
$$\underline{unify(V, I'(\_, ((hum; org; ani), \_), \_))} \rightarrow$$
$$addSlot(\text{動作主} : V, In, Out).$$

図 2.11: 動詞「掛ける」の「が」格スロットの内部表現

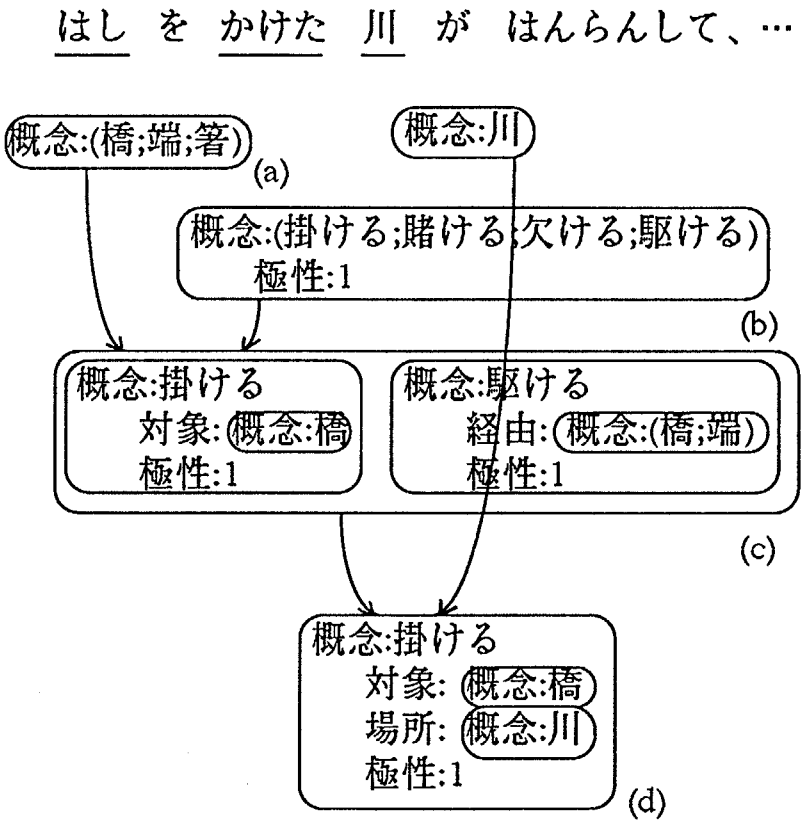


図 2.12:「はしをかけた川がはんらんして … 」の意味解析過程

選択制限がフィラーに対する制約として課され，フィラーの不定項はより限定された表現になる（概念やスロット値がより限定された値になるか，または新しいスロット名の属性が付加される）.

拡張ユニフィケーションを実行するこの述語をプリミティブとして用いた解析の例を図 2.12[16]に示す．図 2.12では，「はし」は３通りの曖昧性があり（表現(a)参照），「かける」には４通りの曖昧性がある（表現(b)参照）.「はしを」の段階では「はし」の曖昧性は解消されないが，「かけた」の時点で「はし」は「橋」と「端」のどちらかであると限定することができる（表現(c)参照）．また，同時に「かける」の意味の曖昧性も「掛ける」と「駆ける」のどちらかであると限定される．これは，図 2.10に示した「かける」の格フレームから，

1.「欠ける」は，「を」格スロットを持たないので不適格である．

[16] 本図では，値の変化過程は省略し，最新の値のみを記述することにする．

2.「賭ける」の「を」格スロットの選択制限を「はし」の3つの可能な意味がいづれも満足しないので,「賭ける」も不適格である.

3.「掛ける」の「を」格スロットの選択制限を「橋」は満足するが,「箸」と「端」は満足しない.

4.「駆ける」の「を」格スロットの選択制限を「橋」と「端」は満足するが,「箸」は満足しない.

ことが,ユニフィケーションにより判断された結果である.また,その後解析が進み,「川」が解析されると,依然曖昧であった「かける」の2通りの意味のうち,「駆ける」の意味は排除され,「掛ける」の方に意味が決定される(表現(d)参照).「川」が「駆ける」の残された格スロットの選択制限を満足しないからである.その結果,「はし」の曖昧性も,依然残っていた2通りの意味のうち,「橋」の方に決定される.

しかし,選択制限の情報を用いた曖昧性解消には限界がある.比喩や

赤ちゃんがビー玉を飲んだ.

のように,異常事態を記述する文は,選択制限に違反するにもかかわらず,意味的に妥当であるからである.この問題点に関しては 7.2.1 節で述べる.

2.4 文的曖昧性の増進的解消

本節では,文的曖昧性の増進的解消方法について述べる.2.2.3 節で述べた文的曖昧性のうち,2.4.1 節では構造的曖昧性について,2.4.2 節では格の曖昧性についてそれぞれ述べる.分かち書きおよびスコープの範囲に関する曖昧性については,7.2.1 節で述べる.

2.4.1 構造的曖昧性の増進的解消

構造的曖昧性の定量的考察についてまず述べる.名詞と格助詞「の」が連続する

$$A_1のA_2の.....のA_n \qquad n \geq 2$$

のような句を考えてみる.nが3の場合,この句の構造は,

$$(A_1のA_2)のA_3 \tag{2.3}$$

$$A_1の(A_2のA_3) \tag{2.4}$$

の2通り存在する.ここで,2.3, 2.4は,図 2.13の構文木(a), (b)に対応し,括弧は2つの語の結び付きの優先順位を表す.nが4,5と変化すると,句の構造は5,14と増加する.この句の構造の

$$1, 2, 5, 14, ...$$

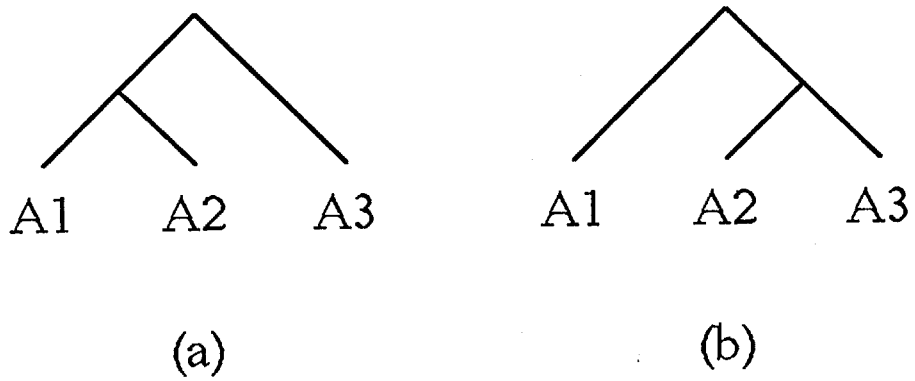


図 2.13: A_1の A_2の A_3の２つの構文木

という数列はカタラン数と呼ばれ，その n 番目の数は，

$$Cat(n) = {}_{2n}C_n/(n+1)$$

で表される． ${}_nC_k$は，n 個のものから k個を取る組合せの数を表す．この数は級数的に増加し，n が２１になると，６５億にも達する[69,3,23].

１文の解析が終了するまで構造的曖昧性を解消せずに放置すると，上に述べたようにその数は組合せ的に爆発し，その解消は非常に計算量を必要とし困難になる．このため，構造的曖昧性の増進的解消が要請される．しかし，このような構造的曖昧性の増進的解消を批判する指摘もある．丸山ら[23,5]は，構造的曖昧性の解消を文末の時点で行なう方法の問題点を指摘しているが，それと同時に，部分的な情報による局所的な（増進的な）曖昧性解消は，

- 曖昧性解消に用いられる
  - 「格助詞を持つ文節は用言に係る」
  - 「係り受け非交差の原則」
  - 「１文１格の原則」

  などの統語的制約が絶対的なものではなく，必ず例外を含む
- 選択制限を用いた曖昧性解消には，2.3.4 節で述べたように問題点が存在する

ため，解析の精度を犠牲にしていると批判している．しかし，自然言語理解研究において，丸山が提案しているような「人間との対話に基づく曖昧性解消方法」を採ることはできない．また，我々は，

- 曖昧性解消に用いる制約の記述を厳密にする

- 7.2.1 節で述べるように，選択制限の記述に優先性(preference)を導入する

ことにより，解析の精度を向上させることは可能であると考える．

構造的曖昧性の増進的解消の研究としては，杉村ら[109]によるものがある．これは，上で述べたような原則や選択制限などを制約として記述し，それらの制約を満足する係り受けの可能な候補を行列の形で得るものであり，制約プログラミング的手法を取っている．また，ワードエキスパートパーサ(word-expert parser)[64]は，各単語に構文的，意味的，実世界知識などをもたせ，構文解析や意味解析における曖昧性を解消しようとしたパーサであるが，各単語をワードエキスパートと呼ばれる動作主体とみなし，各単語内部に記述された動作および単語間の動作の相互作用によって構文解析の制御を行なう点で，構造的曖昧性の増進的解消モデルと考えられる．

本論文で提案する構造的曖昧性の増進的解消方法は，

- 構造的曖昧性による複数の構文木の導出を最小限に抑え，バックトラックしない文法体系を用いる．
- 係り受けの曖昧性を内包する表現として，未解析の係り句を登録したスタックを用いる．
- 曖昧性を増進的に解消するための，左から右への情報伝播メカニズムとして，BUP-UTIを用いる．この BUP-UTI は，2.3.4 節で述べた増進的曖昧性解消モデルを用いた解析を実現するための情報伝播メカニズムである．

に基づく．詳細については 3 章で述べる．

2.4.2 格の曖昧性の増進的解消

本節では，格の曖昧性を増進的に解消する際に問題となる曖昧性の表現方法について述べる．

2.2.3 節で述べたように格が省略された場合や，2.2.2 節で述べたように助詞などの付属語が多義である場合には，格の曖昧性が生じる．また，2.2.2 節で述べたように，動詞の意味が曖昧な場合には，その格フレームにも曖昧性が生じる．このような格に関する曖昧性は，可能な候補をリストの形式ですべて保持することで表現している．例えば，図 2.12では，「はしをかける」の（曖昧な）意味解析結果は，「掛ける」と「駆ける」の２つの格フレームを要素とするリストとして表現されている．

しかし，このようにすべての可能性を展開して保持する手法では，曖昧性が増大すると記憶量組合せ的爆発が爆発することが考えられる．この問題に対しては，

- 複数の格フレームに共通な格スロットは，それらの格フレームの上位の格フレームに記述するなどの「格フレームの階層的記述」を行なう

- 格の曖昧性を圧縮して表現するデータ構造を考える

などの対策が考えられる.

2.5 おわりに

拡張ユニフィケーションを用いて2つのタイプの意味的曖昧性を統一的に処理できる増進的曖昧性解消モデルを提案した. 本モデルは, 曖昧性解消過程で新しく得られた制約の蓄積過程を計算機上に表現することで, 曖昧性を増進的に解消するメカニズムを実現している. 制約を増進的に蓄積しておくことの利点は次のようなものである.

1. 7.2.1 節で述べるように, 選択制限は優先性として考える必要がある. そのように選択制限を記述した場合も, 優先度があるしきい値を超えた意味しか妥当なものでないとして, 選択制限を満足しない意味を排除することで, 曖昧性の組合せ的爆発を回避し, 増進的曖昧性解消を可能にする. また, そのようにして選択制限を満足しない意味を排除した結果, 解析に失敗した場合にも, 値の変化の経過として蓄積していた意味のうち, 排除したものを再び考慮することにより, 選択制限に違反する意味的に妥当な文の処理を容易に実現することができる.

2. 2.3.2 節で述べたように, 新しく得られた制約の蓄積として値の変化を記録しておくことにより, 最初は全くわからない未知語の意味や,「それ」などの指示語の指示する対象の意味の特定が容易に実現できる.

第 3 章

構造的曖昧性の増進的解消メカニズム

本章では，2 章で述べた曖昧性のうち，構造的曖昧性を増進的に解消するメカニズムについて述べる．本論文で提案する手法は，

- 構造的曖昧性による複数の構文木の導出を最小限に抑え，バックトラックしない文法体系を用いる．
- 係り受けの曖昧性を内包する表現として，未解析の係り句を登録したスタックを用いる．
- 曖昧性を増進的に解消するための，左から右への情報伝播メカニズムとして，BUP-UTI を用いる．この BUP-UTI は，2.3.4 節で述べた増進的曖昧性解消モデルを用いた解析を実現するための情報伝播メカニズムである．

ものである．

3.1 節では，本論文で採用する文法体系について述べる．また，3.2 節では，BUP 系解析システム上でトップダウンな（左から右への）情報伝播を可能にするメカニズム BUP-UTI について述べる．最後に，3.3 節では，BUP-UTI を用いた構造的曖昧性の増進的解消の例を示す．

3.1 構文木の導出を最小限に抑える文法体系

文法は言語が許容する文の構造を記述するものである．構文解析は，その文法を用いて，文の構造を解析する処理である．構造的曖昧性を反映した形で文法を記述し，構文解析により構造的曖昧性を反映した複数の構文木を得る手法が従来採られてきた．この手法では，構造的曖昧性は導出された構文木の数で表現される．縦型探索を用いてこの構文解析を実行すると，2.3.2 節で述べたように，曖昧性が陽には表現できず，増進的に曖昧性を解消することができない．また，文が複雑になり，含まれる構造的曖昧性が増加すると，得られる構文木の数は爆発的に増加し，構文解析に要する時間も増加させてしまう．我々は，日本語の埋め込み文に関して，構造的曖昧性を反映した

複数の構文木を導出する左方枝分かれ構造と，構造的曖昧性を内包した構文木を得る右方枝分かれ構造を比較し，

- 左方枝分かれ構造を得る文法では，文が複雑になり，埋め込まれる文の数が多くなればなるほど，導出される構文木の数が増加する．

のに対し，

- 右方枝分かれ構造を得る文法の場合には，構文木が1つしか導出されない．

ことを考察した[46,2]．この右方枝分かれ構造の構文木は，左方枝分かれ構造の複数の構文木の曖昧性を内包する表現と考えられる．このように構造的曖昧性を内包した数少ない構文木を導出し，その内包された構造的曖昧性を，解析過程で得られる他の情報により増進的に解消する方法を本論文では採用する．この手法を用いると，

1. 得られる構文木を少数に抑えることができる．
2. 複数の構文木を得るための構文解析過程での深いバックトラックを行わない．
3. 係り受けの曖昧性は，得られる少数の構文木上での意味解析過程における受け句の選択に帰着される．

本研究で用いた文法を付録 Bに示す．

次節では，増進的曖昧性解消に不可欠である，左から右への情報伝播を実現する BUP-UTI について述べる．また，3.3 節では，上述した文法を用いた構造的曖昧性の増進的解消の例を示す．

3.2 増進的曖昧性解消過程を表現するための左から右への情報伝播

本節では，増進的曖昧性解消過程を表現するメカニズムとして，BUP 系解析システム上でトップダウンな情報伝播を実現する BUP-UTI について述べる．

3.2.1 はじめに

近年，Prolog を用いた自然言語解析に関する研究が活発である．Prolog を自然言語解析に用いることの利点の1つは，DCG(Definite Clause Grammar)[99] とよばれる文法記述形式で記述した文法が，Prolog プログラムに変換され，そのままパーザとして動作させることが可能な点にある．我々は，作成する文法中に左再帰的な文法規則が含まれている場合のことを考慮して，DCG で記述した文法を変換して，ボトムアップにパージングを行う Prolog プログラムを生成するシステム

BUP[91]を用いてきた．その後BUPは，いくつかの改良がなされ[26,16,25]，現在は自然言語解析システムLangLAB[48]の中核になっている．

純粋にボトムアップなパージングを行うシステムでは，解析木を下から上に向けて生長させるにつれて，情報を下から上へと自然に送ることができるが，その解析メカニズムによる制約で，上から下へは情報を送ることができない．そのため，解析途中での構文的，意味的チェックも，情報が下から上がってきて初めて実行されることになり，上から下に情報を流すことが可能な場合に較べると，実行を行うタイミングが遅れ，不要な計算をすることがある．

また，3.2.3節で述べるように，自然言語解析システムにおいては，下から上，上から下の双方向に情報を流せることが望ましい．文を左から右に読み進みながら理解している人間のモデルを考えるなら，読み終った（理解した）部分の情報を左から右に（解析上は上から下に）流しながら文を解析していき，文全体を読み終った時点で，理解した結果が解析木の最上部に得られるシステムが自然であると思われるからである．

既存のボトムアップ解析システムの中には，下から上への情報を用いて一度パージングを終了し，その後，得られた（完全な）解析木をたどる過程で上から下に情報を流すものは存在する[61,45]．しかし，本論文で述べるように，文を読み進む過程で，部分的な解析結果を計算し，それらをダイナミックに双方向に流せるものは存在しなかった．

BUPは，ボトムアップベースの解析システムではあるが，詳細に検討すると，純粋なボトムアップ法ではなく，3.2.2節で述べるgoal節を介して，トップダウンな予測を行っており，また，上から下へ情報を流すことが可能である．左外置処理のためBUPを拡張したBUP-XG[17]では，この特徴を生かし，goal節を拡張するなどして，左外置処理に必要なスタックを実現している．

本研究では，BUP-XGにおけるスタックがトップダウンな情報として用いられていることに着目し，ボトムアップベースの自然言語解析システムでは困難であると思われていた，上から下へ情報を流す機構を開発した．これをBUP-UTI(BUP Using Top-down Information)とよぶ．このBUP-UTIにより，ボトムアップ解析システム上で，解析途中に上下双方向へダイナミックに情報を流すことが可能になることを示す．また，それを応用して，効率よく自然言語解析を行うことができることを示す．

まず3.2.2節で，BUP-UTIの基礎をなすBUP-XGの概要を述べるとともに，BUP-UTIの基本原理を説明する．3.2.3節では，簡単な日本語解析システムを例にあげ，トップダウンな情報が具体的にどのように有効に用いられるか，また，解析中それらがどのように伝わっていくかを述べる．また，3.2.4節では，トップダウンな情報の流れを生かした解析の例を示し，BUP-UTIを用いることにより，従来のボトムアップ解析システムに較べ，早期に非文法的な箇所を検出でき，効率よく文の解析を行うことが可能なことを示す．

3.2.2 BUP-UTIの基本原理

BUP-XGは，左外置処理をBUP上で実現したものである．このため，文法の記述形式にも拡張を施したもの（XGS）を用いている．本節では，トップダウンな情報の流れを実現する基礎となる，BUP-XGにおけるスタックの実現法および，文法記述形式XGSについて簡単に述べるとともに，BUP-UTIの基本原理を説明する．なお，BUP-XGの詳細は文献[17]を参照して頂きたい．

BUP-XGにおけるスタックの実現法

Prolog上で左外置処理をトップダウンに実行する方法として，XG (extraposition grammar)[98]がある．この方法では，左外置処理実現のため，スタックを用いている．ボトムアップ法をベースにするBUP-XGでは，このスタックを実現するため，後述するように，goal節を用いたBUPの動作特性を利用している．本節では，このスタックの実現法について簡単に説明する．

BUP-XGでは，スタックの状態を構造体で表し，それを後続する文法カテゴリに次々受け渡していくことで，スタックを実現している．例えば，下の文法規則3.1に対して，3.2のように2変数を付加したものを考える．3.2の各文法カテゴリに付加された2変数（イタリック体）のうち，左の変数は，その文法カテゴリの解析を始める時点でのスタック（入力スタック）を表し，右の変数は，解析が終了した時点でのスタック（出力スタック）を表す．

$$s \rightarrow np, vp. \tag{3.1}$$

$$s(X0, X2) \rightarrow np(X0, X1), vp(X1, X2). \tag{3.2}$$

3.2はトランスレータにより次の3.3に変換され，実際の解析は3.3のBUP節を用いて行う．3.3を用いた解析では，

$$np(\mathrm{G}, [], \mathrm{I}, X0, X1, \mathrm{XR}) \rightarrow \{s(\mathrm{G})\}, goal\_x(vp, [], X1, X2), s(\mathrm{G}, [], \mathrm{I}, X0, X2, \mathrm{XR}). \tag{3.3}$$

npの解析が成功すると，*X0*,*X1*にスタックの内容が返される．そして，npの出力スタック*X1*が，次の文法カテゴリvpの入力スタックに渡され，vpの解析が成功すると，*X2*に新しいスタックが返される．その後，npの入力スタック*X0*，vpの出力スタック*X2*をそれぞれsの入力スタック，出力スタックとして解析を続ける．

ここで注意したいのは，npの出力スタック*X1*をvpの入力スタックに受け渡す点である．このvpに対する入力スタック*X1*は，npを解析した結果得られた情報であり，goal節を用いてvpを解析する時に，goal節のボディで辞書引きされる文法カテゴリまで伝わる（図3.1参照）．これは，npから得られた情報（スタック）が，文法カテゴリvpを経て，次に辞書引きされる文法カテ

ゴリまで，トップダウンに流れることを意味している．このスタック（トップダウンな情報）の流れを実現するメカニズムが，BUP-UTIの基本原理となっている．次節では，この基本原理について詳しく述べる．

BUP-UTIの基本原理

前節では，BUP-XGにおけるスタックの実現法について述べた．BUP-XGでは，変数を介して次々とスタックの状態を受け渡している．その際，各文法カテゴリに対して入力，出力スタックが存在し，ある文法カテゴリの出力スタックは，文法規則中のその文法カテゴリの直後にある文法カテゴリの入力スタックになっている．そして，この入力スタックは，goal節を用いて解析することにより，トップダウンな情報として，次に辞書引きされる文法カテゴリまで伝わる．

以上のような情報（スタック）の受け渡しに関する考察から，スタックの実現に用いた2変数をトップダウンに情報を流すための変数（この変数を以後，TI(Top-down Information)用変数とよぶ）として利用し，この変数で情報を左から右に（上から下に）流すことで，効率の良い解析システムの実現を可能にしたのがBUP-UTIである．

なお，BUP-UTIでは，BUP-XGと同様に，文法記述形式としてXGS（後述）を利用し，また，トランスレータにはBUP-XGトランスレータ（後述）を用いている．

以下では，このBUP-UTIの基本原理について述べる．

BUP-UTIでは，前節に示した3.1 → 3.3のように，ユーザがXGSで記述した文法は，BUP-XGトランスレータによりBUP節等に変換される．そして，それらが図 3.1のgoal節とともにボトムアップ構文解析システムとして直接動作する．ここで，述語'goal'は，トップダウンな情報を用いないで解析を行うgoal節であり，一方，述語'goal_x'は，トップダウンな情報を解析に用いるgoal節である．

解析は，次のようなgoal節の呼び出しにより始まる．

$$?-goal(s,A,[she,smiled],[]).$$

このゴールの実行により，(4)のボディが実行される．(4)のボディでは，入力文の先頭の単語の辞書引きを行い（述語'dict'），その単語の文法カテゴリをヘッドとするBUP節を選択する．今の場合，先頭の単語'she'の辞書引きの結果，その文法カテゴリであるnpをヘッドとするBUP節3.3を選択する．そして，そのボディの実行において，文法カテゴリvpを次のゴールとして解析を行う(goal_x(vp,[],X1,X2))．これは，入力文の残りの部分に対して，文法カテゴリvpの部分が存在するはずだという予測をし，goal節を通じて，その予測（文法カテゴリ名vp）をトップダウンに流していることに相当する．さらにgoal節のボディで次の単語を辞書引きし，その文法カテゴリ

```
goal(G,A,X,Z) :-
    dict(C,A1,X,Y),
    Link =.. [C,G],call(Link),
    P =.. [C,G,A1,AA,[],[],[],Y,Z],call(P),
    A = AA.                                    (4)
goal_x(G,A,XI,XO,X,Z) :-
    dict(C,A1,X,Y),
    Link =.. [C,G],call(Link),
    P =.. [C,G,A1,AA,XI,XI,XOO,Y,Z],call(P),
    A = AA,XO = XOO.                           (5)
```

図 3.1: BUP-UTI における goal 節

をヘッドとする BUP 節を選択することで解析が進む．このように BUP は，純粋にボトムアップに解析を行っているわけではなく，レフトコーナーから解析を始め，そこから得た情報を用いて，次に来るべき文法カテゴリについて予測をし，その予測をトップダウンに用いることによって解析を進めている．

従って，この BUP の動作特性を生かし，従来の goal 節を修正し，TI 用変数を導入して（図 3.1 参照），情報をトップダウンに流すことを考える．前節で述べたように，ある文法カテゴリを解析して得られた情報が，次に辞書引きする文法カテゴリに（上から下へ）伝わるのは，図 3.1の述語'goal_x'において，直前の文法カテゴリから伝わった入力 TI 用変数 XI が，次の単語を辞書引きして呼び出される述語 C の入力 TI 用変数に受け渡されることによる（イタリック体の箇所参照）．図 3.2 に，文法規則 $M \rightarrow H, G.$が適用され，H を頂点とする部分木の解析が成功し，G をゴールとして解析を行う際の情報の流れを示す．従って，BUP-UTI におけるトップダウンな情報は，トップダウンという用語から通常想起するように，支配している文法カテゴリ（上）から，支配されている文法カテゴリ（下）へと段階的に流れるわけではなく，goal 節で呼び出される文法カテゴリ(G)から，辞書引きされる単語のレベル（レフトコーナーの一番ボトム）の文法カテゴリ(C)にまで一気に伝わる．それから，次の goal 節の呼び出しまでは，部分木がボトムアップに成長するとともに下から上に伝わる．そして，次の goal 節が呼び出されると，再び一番ボトムのレフトコーナーの文法カテゴリへと一気に流れる（図 3.2参照）ことを繰り返す．

既存のボトムアップ解析システムの中には，トップダウンな情報を解析に利用しているものもある．これらと BUP-UTI は，以下の点で異なる．

1. 拡張 LINGOL[61]や属性文法における属性計算[45]などのように，ボトムアップな情報だけを用いて一度パージングを終了した後，完成した解析木上を上から下に情報を流すものと異なり，BUP-UTI では，解析木を作成する過程でダイナミックに情報を上から下に流すことができる．

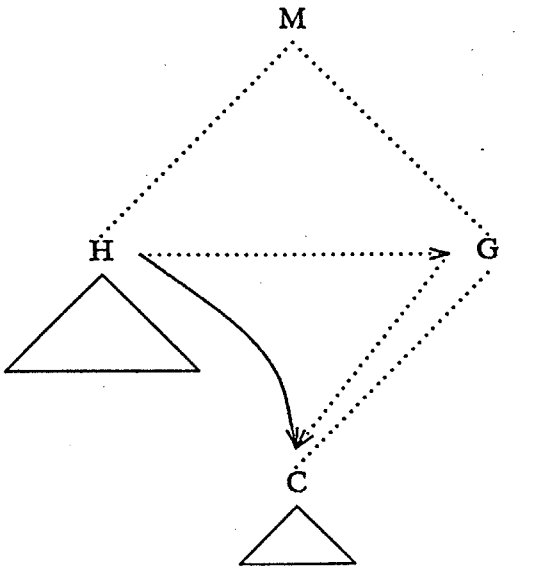


図 3.2: BUP-UTI における情報の流れ

2. 属性文法の１つのクラス（LR属性文法;LR-attributed grammar）では，LR構文解析の過程で双方向に情報（属性）を流し，属性計算をすることが可能である．しかし，このクラスでは，相続属性(上から下へ伝えられる属性）の使用には以下の条件が課されている．構文解析過程のすべての状態において，適用可能なすべての文法規則に対して相続属性は唯１つに決定できることこの条件により，以下のような文法規則を含む文法はこのクラスには入らない．

$$A \rightarrow X, B, X.\ \ with \downarrow B = \downarrow A + 1 \tag{3.4}$$

$$A \rightarrow X, B, Y.\ \ with \downarrow B = \downarrow A + 2 \tag{3.5}$$

（ここで，with 以後は属性計算式，↓ B は B の相続属性を表わす.）

なぜなら，B の相続属性は，↓ B = ↓ A+1,↓ B = ↓ A+2 の２通りの計算式があり，唯１つに決定できないからである[45]．この例から明らかなように，相続属性に対する条件は強力で，LR 属性文法では，限られた場合にしか上から下への情報（属性）を利用できない．これに対し，BUP-UTI では，上から下へ流す情報として任意のものを記述できる．

XGS および BUP-XG トランスレータ

BUP-XG における拡張された文法記述形式 XG S で記述した文法は，BUP-XG トランスレータにより BUP 節に変換される．その際，上で述べたトップダウンな情報を受け渡すための変数（TI 用変数）を トランスレータが自動的に付加する．従って，通常ユーザはその変数について気にする必要はない．しかし，ユーザが直接変数を操作したいこともある．そのために，XGS では，文

$$s_1[X0,X1] \Longrightarrow s_2[X0,X1],[and],s_3[X0,X1].$$

↓変換

$$s_2(G,[],I,X0,X1,XR) \rightarrow \{s(G)\},$$
$$[and],$$
$$goal\_x(s_3,[],X0,X1),$$
$$s_1(G,[],I,X0,X1,XR).$$

図中の文法カテゴリsに対する添字は，理解しやすさのためにつけたもので，実際の解析には用いられない．

図 3.3: BUP-XG トランスレータによる変換例

法規則の中にTI用変数を陽に記述できるようになっている．図 3.3は，XGSでのその記述例および変換結果である（この例は，文が'and'で等位接続された場合には，その接続された2文に対するギャップは同じでなければならないことを記述するためのBUP-XGの文法規則である[17]）．

図 3.3からわかるように，XGSの記述における文法カテゴリの直後の[ ]内の2つの変数が，TI用変数としてBUP節にそのまま付加される．

3.2.3 BUP-UTIを用いた簡単な日本語解析システム

解析メカニズム

図 3.4に示すXGSで記述した簡単な日本語文法を用いて，「花子は学校へ行った.」という文を解析してみよう．解析は文を左から右に読み進みながら行う．

図 3.4の文法を用いると，最終的には図 3.5の解析木を得るが，その解析過程で情報は概略矢印のように流れる．この情報の流れを利用し，この日本語解析システムでは，TI用変数を後置詞句の意味情報のスタックとして用いて解析を行っている．すなわち，後置詞句を解析した意味解析結果を次々にスタックに付加して文末の方向（右方）に流し，述語（動詞）が現われた直後にそのスタックと動詞の意味を用いて全体の文の 意味を決定している．また，TI用変数を後置詞句の意味情報のスタックとして用いることで，日本語に対して一般に成り立つとされる「係り受け非交差の原則」を簡潔に実現することができる．この「係り受け非交差の原則」の実現法に関しては，3.3 節で述べる．

以下，実際のBUP-UTIの実行過程を見ることで，具体的にスタック情報の流れを追い，トップダウンな情報の流れをどのように利用するかを説明する．解析は，

$$?-goal(文,A,[花子,は,学校,へ,行った],[]).$$

の呼び出しで始まる．

1. goal節(4)を用いて，最初の単語「花子」の辞書引きを行い，その文法カテゴリ「名詞」を

ヘッドにもつ文法規則を選択する.

2. 文法規則(g3)を適用する. そのボディにおいて, goal(助詞,P) を呼び出す. ……これは「は」で成功する.

3. 意味解析プログラム interp を実行する.「花子は」の意味 (X:花子) を入力 TI 用変数 X0 (内容は[]) の先頭に付加し X1 に返す.

4. 後置詞句の解析が終了する. X1 には[X:花子]がバインドされている.

5. 文法規則(g1)を適用する. そのボディにおいて, goal_x(文,S,[X:花子],X2)を呼び出す.

6. 次の単語の辞書引きを行う. 以下, 1 ～ 5 と同様の解析を繰り返し, その結果, goal_x(文,S',[へ:学校,X:花子],X2')を呼び出す.

7. 次の単語「行った」の辞書引きを行う. そして, その文法カテゴリ「述語」をヘッドにもつ文法規則を選択する. この時点で, スタックの内容として, [へ:学校,X:花子]という情報が,「述語」をヘッドにもつ文法規則までトップダウンに伝わってきており, 辞書引き直後に直ちに利用可能なことに注意してほしい.

8. 文法規則(g2)を適用する.

9. 意味解析プログラム interp を実行する.「行った」の意味(Vp)と入力 TI 用変数 X0 ([へ:学校,X:花子]) を用いて意味解析を行う. そして, 解析結果

$$(\text{行った}, (\text{行為者} : \text{花子}, \text{目標} : \text{学校}, \_), 1)$$

を S に返す (X1 には[]がバインドされる).

10. (トランスレータにより付加された) 停止条件節を使って S が次々と上のレベルの「文」に上がり, 最終的にトップレベルの「文」の呼び出し時の変数 A まで上がっていき, 解析が終了する. そして, 解析結果として,

$$A = (\text{行った}, (\text{行為者} : \text{花子}, \text{目標} : \text{学校}, \_), 1)$$

を得る.

以上の例から明らかなように, BUP-UTI を用いると,「述語」として「行った」が得られた時点で, 意味解析が終了し, 全文に対する解析結果が得られることに注意してほしい (9 参照). BUP-UTI を用いる場合と, 純粋なボトムアップ解析システムを用いる場合とで, 解析過程が異なり, どのような利点が得られるかについては 3.2.4 節で述べる.

```
文(S) --> 後置詞句(_) , 文(S) .
文(S)[X0,X1] ==> 述語(Vp) ,
                 {interp((v,t),Vp,X0,S,X1)} .
後置詞句(_)[X0,X1] ==> 名詞(N) , 助詞(P) ,
                       {interp((n,v),P:N,X0,_,X1)} .
                        ↓変換
後置詞句(G,_,I,X0,X1,XR) --> {文(G)},
     goal_x(文,S,X1,X2),
     文(G,S,I,X0,X2,XR).                    (g1)
述語(G,Vp,I,X0,X0,XR) --> {文(G)},
     {interp((v,t),Vp,X0,S,X1)},
     文(G,S,I,X0,X1,XR).                    (g2)
名詞(G,N,I,X0,X0,XR) --> {後置詞句(G)},
     goal(助詞,P),
     {interp((n,v),P:N,X0,_,X1)},
     後置詞句(G,_,I,X0,X1,XR).              (g3)
```

図 3.4: 簡単な日本語解析用文法

3.2.4 BUP-UTIを用いた構文的，意味的チェック

本節では，従来のボトムアップ解析システム BUP と，BUP-UTI における構文的，意味的チェックの実行法を比較することで，その実行のタイミングの違いを明らかにし，BUP-UTI の有効性を示す．なお，構文的チェックの例としては，主語と動詞の呼応のチェックを取り上げる．

構文的チェック

1. 従来の BUP の場合

   従来の BUP では，下に示すような文法規則を用いてチェックを行う．

$$s(S\_A) \rightarrow np(NP\_A), vp(VP\_A), \{concord(NP\_A, VP\_A)\}. \tag{3.6}$$

$$vp(V\_A) \rightarrow v(V\_A), that\_clause(THAT\_A). \tag{3.7}$$

   v（動詞）の各単語の辞書項目に記述されている人称・数に関する情報(V_A)は，文法規則3.7を用いて述語 vp の引数まで伝わり，その結果 3.6で，np の人称・数に関する情報(NP_A)との間で呼応のチェックが述語'concord'を用いて行われる．これは，呼応のチェックが，3.6の vp の解析終了まで遅延されることを意味する（図 3.6参照）．

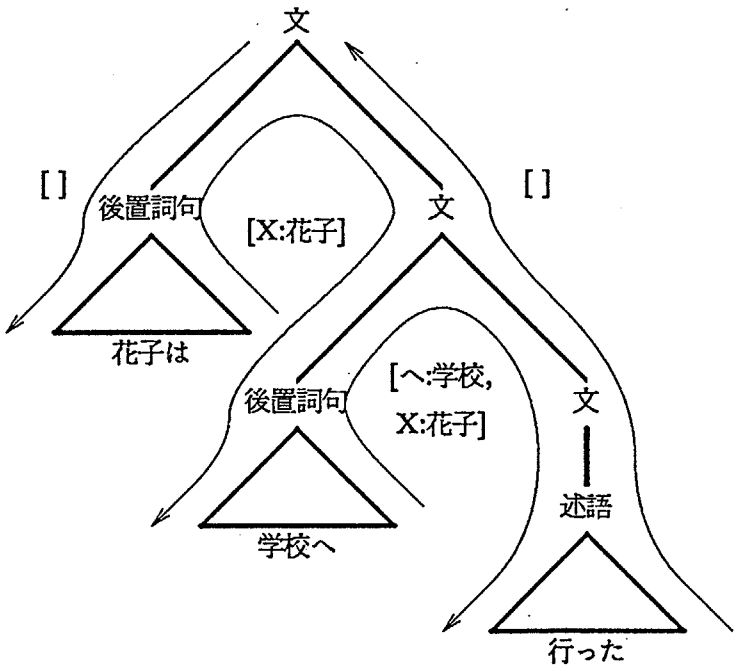


図 3.5: BUP-UTIを用いた解析の流れ

2. BUP-UTIの場合

BUP-UTIを用いると，次に示すような文法規則でチェックを行うことができる．

$$s(S\_A)[X0, X1] \Longrightarrow np(NP\_A), vp(VP\_A)[[NP\_A|X0], X1]. \tag{3.8}$$

$$vp(V\_A)[[NP\_A|X0], X0] \Longrightarrow v(V\_A), \{concord(NP\_A, V\_A)\}, that\_clause(THAT\_A). \tag{3.9}$$

BUP-UTIでは，3.8の文法規則で，npの人称・数の情報(NP_A)をTI用変数に付加してvpに流し，それを用いて3.9の文法規則中で，vの人称・数の情報(V_A)との間の呼応のチェックを直ちに行うことができる（図 3.7参照）．

この例からわかるように，従来のBUPでは，vpの解析が終るまで呼応のチェックが遅延されるのに対し，BUP-UTIの場合には，vpの解析中，vが現れた時点で，すなわち，チェックに必要な情報が得られたらすぐに呼応のチェックが行える．この違いは，vpが単純な構造の場合には，さほど問題にならないが，vpが複雑で大きな構造をもつ場合（動詞の目的語として不定詞句，that節など，文に近いような構造のものがくる場合）には，この非文法的な箇所のチェックが遅延されることによる計算コストは大きくなる．

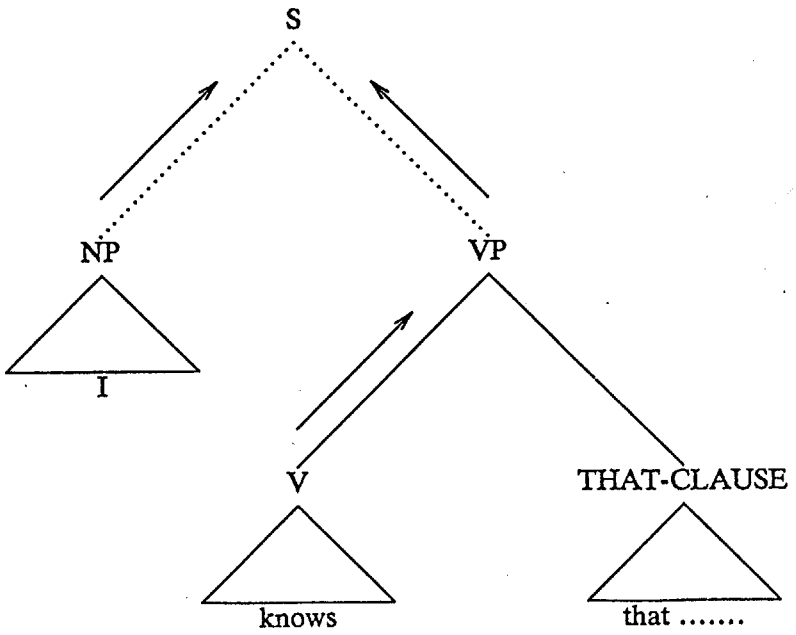


図 3.6: 従来の BUP における情報の流れ

	BUP	BUP-UTI
1	450(8)	100(2)
2	600(8)	116(2)

実験は，Quintus-Prolog インタープリタ上で，文法規則数 13 で行った．従来の BUP と BUP-UTI では，(6),(7) ↔ (8),(9)のように，s,vp に関する文法規則だけが異なる．括弧の中の数字は辞書引きされた単語数，解析時間の単位は msec. である．

表 3.1: 従来の BUP と BUP-UTI を用いた解析の比較

文１，２を，従来の BUP と BUP-UTI を用いて解析した時の，解析に要した時間及び辞書引きされた単語数を以下に示す．

1. I opens the door with a key.

2. I opens not only the door but also the window.

この表から，呼応のチェックの実行のタイミングの違いによる，従来の BUP と BUP-UTI の解析に要する時間の差は明らかであろう．また，従来の BUP では，動詞句に対応する部分が長くなると，解析時間が増加するのに対し，BUP-UTI ではほとんど変わらないことに注意してほしい．これは，従来の BUP の場合には，動詞句の最後まで解析を進めないと非文法的な箇所を検出できないのに対し，BUP-UTI の場合には，主語の'I'と動詞の'opens'を辞書引きした時点で，非文法的な箇所を検出し，それ以後の無駄な解析を行わないからである（辞書引きされた単語数参照）．

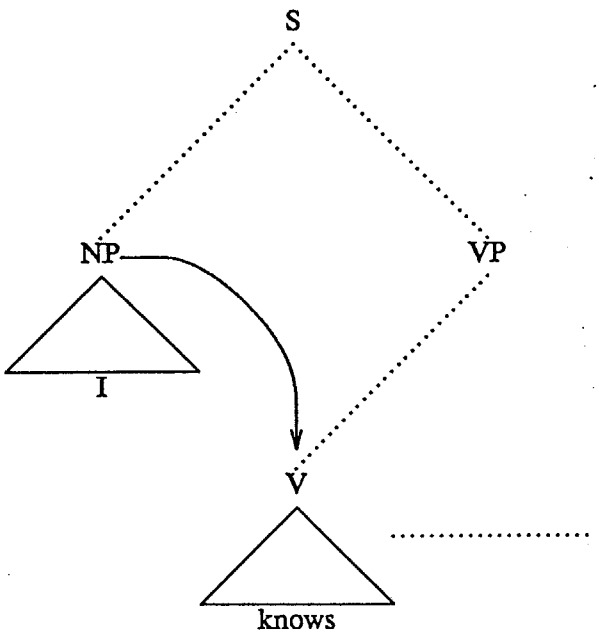


図 3.7: BUP-UTIにおける情報の流れ

この呼応のチェックと同様に，日本語における用言（動詞，形容詞，助動詞等）の相互承接関係のチェックや，係り結び的な現象のチェックにも，BUP-UTIが利用可能であり，我々はそれにより早期に非文法的な箇所の検出を行っている．

意味的チェック

1. 従来の BUP の場合

   従来の BUP では，下のような文法規則を用いて意味解析を行う．

$$文(S) \rightarrow 後置詞句(Pp), 文(S1), \{interp(Pp, S1, S)\}. \tag{3.10}$$

$$文(Vp) \rightarrow 述語(Vp). \tag{3.11}$$

$$後置詞句(P : N) \rightarrow 名詞(N), 助詞(P). \tag{3.12}$$

   具体的には，3.10を再帰的に適用して次々と「後置詞句」の部分木を生成し，最終的に「述語」が現れた時点で 3.11を適用する．この点に関しては，BUP-UTIを用いた意味解析（3.2.3節参照）と変わらないが，従来の BUP の場合，意味解析を実行するプログラム interp が文法規則 3.10の最後にあるため，その実行のタイミングは，「後置詞句」の右側の述語「文」の意味 S1 が得られて初めて実行されることになる．従って，「花子は学校へ行った」という文で，後置詞句「花子は」に対して意味解析を実行するのは，その右にある「文」の部分木が完成した後，すなわち，後置詞句「学校へ」と文「行った」から文「学校へ行った」の意味

を解析した後になってしまう．以上まとめると，従来のBUPの場合には，全体の文に対する意味解析結果が得られるのは，図3.5における一番上の「文」に対する解析木（全体の文に対する解析木）が形成される時点まで延期される．これは，文献[93]で有用であると論じられている早期意味解析(early semantic analysis)の考え方に反する．

2. BUP-UTIの場合

BUP-UTIを用いる意味解析の詳細については，3.2.3節で説明したが，要約すると，後置詞句を解析した（部分的な）意味解析結果を次々にスタックに付加して文末の方向に流し，述語（動詞）が現れた直後にそのスタックと動詞の意味を用いて全体の文の意味を決定することができる．そして，「述語」として「行った」が得られた時点で，意味解析が終了し，全文に対する解析結果が得られる．この時点では，図3.5のような全体の文に対する解析木が完成しているわけではなく，完成している部分木は，そのうちの，2つの「後置詞句」の部分木および「述語」の部分木だけであることに注意してほしい．

この例から明らかなように，従来のBUPでは，BUP-UTIと較べ，意味解析の実行のタイミングが遅れる．そのため，意味的に異常な箇所の検出が遅くなり，不要な計算を実行する可能性がある．例えば，3.2.3節の例文の「花子は」が「花子を」であるような文「花子を学校へ行った.」に対して，BUP-UTIの場合には，文末の「行った」の直後の意味解析で，「花子を」が「行った」に係りえないことから，意味的に異常であると判断することができるのに対し，従来のBUPを用いると，「花子を」が先頭の後置詞句であることから，全体の文に対する解析木が形成される時点で実行される意味解析プログラムにより初めて，「行った」に係りえないことに気付くことになる．

以上述べてきたように，BUP-UTIを用いれば，構文的，意味的チェックを，従来のBUPに較べ早期に実行できることから，不要な計算を回避することが可能である．

3.3 BUP-UTIを用いた構造的曖昧性の増進的解消

本節では，BUP-UTIを用いて係り受けの曖昧性を増進的に解消する方法を例により説明する．「花子は橋で泳いでいる少年を見た.」という文を解析することを考える．解析には，以下に示す文法を用いる．

```
文(S) --> 後置詞句(_) , 文(S) .                          (g4)
文(S)[X0,X2] ==> 述語(Vp) ,
                 {interp((v,n),Vp,X0,_,X1)} ,
                 文(S)[X1,X2] .                          (g5)
文(S)[X0,X1] ==> 述語(Vp) ,
```

{interp((v,t),Vp,X0,S,X1)} . (g6)

後置詞句(_)[X0,X1] ==> 名詞(N) , 助詞(P) ,

{interp((n,v),P:N,X0,_,X1)}. (g7)

この例文には,(左方枝分かれ構造を得る文法で解析すると,) 図 3.8のような3通りの構造的曖昧性（構文木）が存在する. 一方, 上に示した文法を用いて解析すると, 図 3.9のような1通りの構文木を得る. 係り受けの曖昧性は, TI用変数を用いたスタックに保存された係り句がどの句を受け句とするかという選択に帰着される. 出現した係りの句（この例の場合には, 後置詞句のみ）の意味は, 次々スタックの先頭に付加される(g7). すなわち, 文「花子は橋で泳いでいる少年を見た.」を解析していく過程で, スタックは,

[]	（文の先頭）
[X:花子]	（「花子は」の解析後）
[で:橋,X:花子]	（「橋で」の解析後）

のように変化する. このスタック中の要素は, 未解析で受け句が決定されていない係り句を表しており, これらより後ろに出現する任意の受け句に係りうる.「橋で」の解析後のスタック中の要素[で:橋,X:花子]は, 後ろに現れる「泳いでいる」,「見た」の両方に係りうることから, 図 3.8に示した3通りの構造的曖昧性を内包していると考えることができる[1]. ここで,「花子は」が「泳いでいる」に係り,「橋で」が「見た」に係る場合が考慮されていないのは, 構造的曖昧性を解消する情報として用いている統語的制約の1つである「係り受け非交差の原則」により, その係り受けが排除されたことによる.

「係り受け非交差の原則」は, 日本語の係り受けに関して一般的に成り立つとされている.

「係り受け非交差の原則」 係りの句から受けの句への矢印を用いて係り受け関係を表わすと, その矢印は交差しない.

すなわち, 次のようなことはない.

この原則は, TI用変数をスタックとして用いることにより簡潔に実現できる. すなわち, 左から右へと係り句の情報をスタックとして伝播し, 受けの句（この例文では「泳いでいる」）が現れた

[1]「花子は」,「橋で」が両方「泳いでいる」に係る場合が構文木(a),「橋で」だけが「泳いでいる」に係る場合が構文木(b), 両方「見た」に係る場合が構文木(c)にそれぞれ対応する.

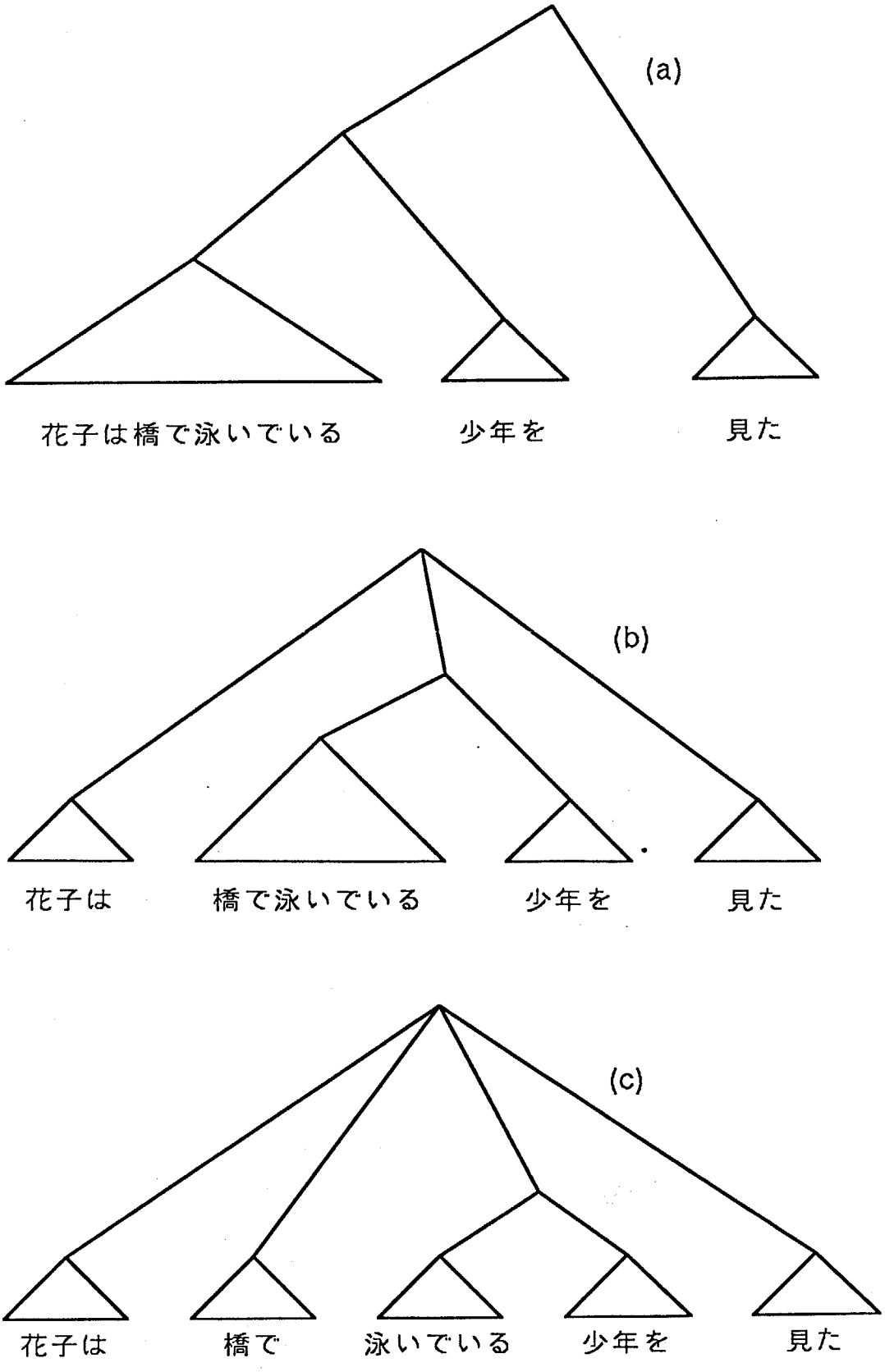


図 3.8:「花子は橋で泳いでいる少年を見た.」の左方枝分かれ構造の構文木

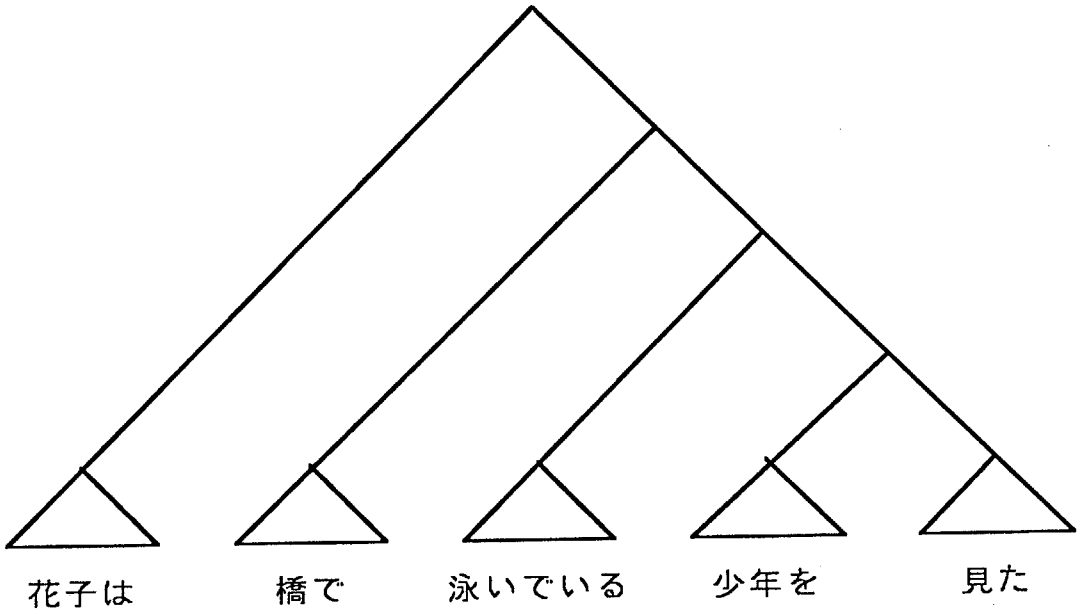


図 3.9:「花子は橋で泳いでいる少年を見た.」の右方枝分かれ構造の構文木

時点で，係り受け関係の決定を行なう(g5). この際，係り句の要素を取り出す順番は，スタックとして先頭からしか許さないことにする．この例の場合,「橋で」が「泳いでいる」に係らないのであれば,「花子は」は決して「泳いでいる」には係りえないことになる．この「係り受け非交差の原則」以外に，本研究では，構造的曖昧性を解消する情報として，

1.「1文1格の原則」

2. 選択制限

   これらは，2.3.4 節で述べたユニフィケーションにより容易に実現される．

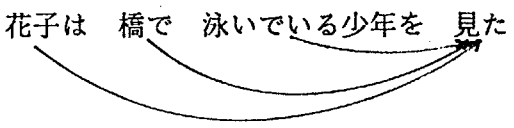
3.「名詞句」＋「助詞」は用言に，連体形の用言は体言に係るなど，係る句の情報により受ける句は制限される．

を用いている．

この例では，その後解析が進むと,「橋で」が選択制限により「泳いでいる」に係りえないことから,「橋で」,「花子は」がそのままスタック中に残る．そして，その後現れる「少年を」がスタックに付加され,「見た」を解析する時点でのスタックは，

$$[\text{を}:\text{少年},\text{で}:\text{橋},X:\text{花子}]$$

のようになっている．そして,「見た」との係り受け関係の決定を試みた結果，解析は成功して終了する(g6). その結果，得られるこの例文全体に対する係り受け関係は，次のような妥当なものただ1つとなる．



3.4　おわりに

BUP-UTIの基本原理，その実現法について述べ，実際の解析例を用いてその解析メカニズムを説明した．

BUP-UTIを用いることにより，ボトムアップ解析システム上で，解析途中に上下双方向へダイナミックに情報を流すことが可能になった．それにより，従来のボトムアップ解析システムに較べ，早期に非文法的な箇所や意味的な異常さを検出できるため，効率の良い自然言語解析が可能になることを示した．

3.2.3節で述べた日本語解析システムは，文献[93]の早期意味解析法にそったものであり，3.2.1節で述べた人間の文理解過程に近いと考えられる．

第 4 章

辞書記述用言語 SRL

本章では，本システムにおける意味解析用の辞書を記述するための高水準言語 SRL について述べる．また，SRL 形式の辞書を用いた意味解析の例を示す．

4.1 はじめに

近年，自然言語理解に関する研究が活発に行われている．その中で，意味解析は困難な問題の 1 つである．自然言語の意味解析では，どのような情報をどのような形式で，辞書に記述しておくかが重要になる[36,9]．辞書記述に要求される条件は以下のものである．

1. ユーザが辞書を簡潔かつ容易に記述できる．
2. 入力文の意味的妥当性を検査するために，辞書項目中に，その単語と意味的に共起しうる単語についての情報（格フレームにおける，格を満たしうるものに関する制約条件はこれにあたる）を記述できる．
3. 辞書項目中に，宣言的知識だけでなく，手続き的知識も記述できる．
4. 辞書の記述量の節約を図るために，知識全体を階層構造として記述しておき，知識の継承を利用して，上位の知識を，下位の辞書項目から参照できる．
5. ある単語が他の単語との間に持ちうる意味的な関係が不可欠なものか，なくてもよいものか（格フレームにおいて，格が必須であるか任意であるか[75]）を記述できる．

これまでの研究では，フレーム[95]形式を用いて意味解析用の辞書を記述することが多かった．典型的なフレーム形式による辞書記述例を図 4.1 に示す．フレームは，スロットの集合から構成されている（例えば，図 4.1 の(1-5)は 1 つのスロットを表現している）．一般に，スロットは，スロット名(obj)，制約条件([$OR,event,thingOpen])，アクション(=object)の 3 つ組から構成される．

これまでのフレーム形式を用いた辞書は，辞書記述に必要な条件を以下の点で満足している．

```
[open,
    [subj,[[human,=agent],                          ……(1-1)
           [[$OR,event,thingOpen],
                  =object],                         ……(1-2)
           [instrument,=instrument],                ……(1-3)
           [wind,=reason]]],                        ……(1-4)
    [obj,[$OR,event,thingOpen],=object],            ……(1-5)
    [with,instrument,=instrument],                  ……(1-6)
    [self,act]]                                     ……(1-7)
```

図 4.1: 動詞'open'の辞書記述

(a) 1つの単語に対する情報をまとめて記述できる（条件1を満たす）.

(b) アクションを用いることで，宣言的知識だけでなく，手続き的知識も記述できる（条件3を満たす）.

(c) (1-7)のように，上位項目を記述しておくことにより，知識の継承を実現できる（条件4を満たす）.

しかし，フレーム形式辞書を用いたこれまでの研究には，次の2つの問題点がある.

[a] スロットの制約条件として，そのスロットを満たしうるフレーム（これを以下ではフィラーとよぶ）についての制約条件しか記述できないため，フレーム中に存在する複数のスロット間の共起関係について制約条件が記述できない.

[b] スロットの必須性・任意性に対する扱いが十分ではない.

[a]についていえば，自然言語の辞書記述において，スロット相互間に存在する共起関係についての情報を記述できることは重要である．なぜなら，スロット相互間に共起関係が存在する（動詞の）フレームは多数存在する[32]からである．たとえば,「劣る」という動詞を考えてみよう.「劣る」のフレームには,「が」格および「より／に」格のスロットがある．この2つのスロットのフィラーに対する制約条件としては，個々のフィラーに対する意味制約条件とともに，それらが互いに同じ範疇に入る対象であるという（スロット間の）制約条件が必要になる．このような情報を辞書中に記述することは，これまでのフレーム形式の辞書記述では十分考慮されていなかった．従って，これまでのフレーム形式辞書を用いていたのでは，スロット間の共起関係をも考慮した意味解析が行えないことになる.

[b]についていえば，これまでのフレーム形式辞書を用いた研究では，スロットの必須性・任意性に関して，必須スロットのみを格フレーム中に記述し，任意スロットに関しては，スロットの

フィラーに関する制約条件を用いないで，意味解析を行うという手法を用いていた[35,53]. [35]では，任意スロットは無条件に満たしうるとしているが，これは厳密な意味解析を行う立場からは問題がある．以上から，これまでの研究は，スロットの必須性・任意性に関して記述を行っているものの，任意スロットに関する意味解析法が不十分であるといわざるをえない．

本研究では，これまでのフレーム形式辞書の上記した問題点を解決し，上に述べた５つの条件を満たす辞書記述用言語 SRL(Structured object Representation Language)を提案する．

4.2節では，これまでのフレーム形式辞書で問題となっている，辞書記述に要求される条件 2,5，言い換えると，

2' スロット間の共起関係に関する制約条件の記述

5' スロットの必須性・任意性に関する記述

の必要性について具体的に説明する．

4.3節では，本研究で提案する辞書記述用言語 SRL について述べる．SRL で記述した知識を実際に計算機上で実行する際の内部表現形式としては，本研究室で開発された DCKR(Definite Clause Knowledge Representation)[47] を採用することにする．4.3.2節では，4.2節で述べた問題点を SRL でどのように解決するか述べる．また，4.3.3節では，SRL で記述した辞書を用いた意味解析の実行例を示す．

4.2 これまでのフレーム形式辞書の問題点

フレームは，意味解析を行うための辞書記述の形式としてよく用いられてきた．フレームは，スロットの集合から構成されている．一般に，スロットは，スロット名，制約条件，アクションの３つ組から構成される．スロット名，制約条件としてどのようなものを記述するかについては，辞書作成者に委ねられているが，議論を具体的にするために，スロット名，制約条件は，以下のものであるとする．制約条件には，そのスロットを満たしうるフレーム（これをフィラーとよぶ）の意味的性質を記述する．これを意味制約条件とよぶ．スロット名は，フィラーが文中で果たす統語的役割（統語制約条件）を表す．アクションには，フィラーが意味制約条件を満たしたとき実行するプログラムを記述する．

4.1節で述べたように，フレームは辞書記述形式として３つの条件を満足している．しかし，これまでのフレーム形式による辞書記述には，4.1節で述べた[a],[b]の問題点が存在する．本節では，これまでのフレーム形式辞書で問題となっている，辞書記述に関する２つの条件（4.1節の 2',5'）の必要性について述べる．この２つの条件に関して十分考慮されていない辞書記述を用いて意味解析を行うと，意味的に異常な文も，正常な文として受理してしまうことがある．これを実例によ

り説明する．ただし，以下では，比喩的な文を含め，選択制限に反する文はすべて，意味的に異常な文と考えることにする．解析には，図4.1のopenに関する典型的なフレーム形式の辞書を用いることにする．

ケース1　例）"*We opened a debate with a key."
'a debate'は，(1-5)を使って解析され，また，'a key','we'はそれぞれ，(1-6),(1-1)を用いて解析され，意味解析に成功する．

ケース1の文が意味的に異常な文であるにもかかわらず，意味解析に成功する理由は，以下のようである．

(1-6)のwithスロットの制約条件として，フィラーがinstrumentでなければならないという意味制約条件が書かれているだけで，(1-6)と共起する(1-5)のスロットのフィラーがthingOpenでなければならない（'debate'のようなeventではない）という制約条件が記述されていない．

従って，ケース1の問題点を解決するには，(1-6)のスロットの制約条件中で，他のスロット（(1-5)）の参照が可能でなければならない．そのためには，現在の時点での意味解析の途中結果を参照して，その内容を検査する機構が必要である．

ケース2　例）"*I opened."
'I'が，(1-1)を用いて解析され，意味解析に成功する．

この場合に，"*I opened." という意味的に異常な文を受理してしまうのは，

'I'がopenの主語の位置にくる他動詞としてのopenのフレームに，「主語のスロットと目的語のスロットは必ず満たされなければならない」ということが記述されていない．

からである．従って，ケース2の問題点の解消のためには，openに関するフレームにおいて，どのスロットは必須で，どのスロットは任意であるという記述が必要である．

次節では，以上の問題点が，本研究で提案する辞書記述形式SRLにより，どのように解決されるかを述べる．

4.3　辞書記述用言語 SRLとその Prologプログラムへの変換

4.3.1　辞書記述用言語 SRL

4.1節で述べた辞書記述形式としてのフレームの利点（(a)-(c)）を損なうことなく，前節で述べたフレーム形式の辞書記述の問題点を解決した辞書の記述形式として，辞書記述用言語SRL(Structured

```
open ::
  [ subj   $ isa:event ;
             isa:thingOpen          => object ]
    ::
  [ subj   $ isa:instrument        => instrument ;
             isa:wind              => reason     ]
  [ obj    $ isa:thingOpen         => object ]
    ::
  [ subj   $ isa:human             => agent      ]
  [ obj    $ isa:thingOpen ;
             isa:event             => object     ]
  ( with   $ isa:instrument
             where
             object!instance isa:thingOpen …(2-1)
                                   => instrument )
    ::
  [ isa    : act ]
```

図 4.2: 動詞'open'の SRL 形式の辞書記述

object Representation Language)を提案する. SRLで記述した知識は, SRL トランスレータ (現在, Unix4.2BSD 上で稼働中) により, DCKR形式に変換され, 変換されたプログラムが直接 Prolog 上で動作する. 図 4.2 に open について SRL で記述した例を示す. この記述がトランスレータにより図 4.3 の DCKR形式に変換されることになる. 以下, SRL のシンタックスについて説明する.

1. '::'の間に記述されているスロットの並びが 1 つのフレームを表す. 図 4.2 の open の 3 つのフレームは, 図 4.3 の節集合に変換される. 図 4.3 では, ヘッドの述語の第 2 引数の先頭の数字が同じ節集合は 1 つのフレームを表現する.

2. スロットの記述における区切り記号の意味は以下の通りである.

   '$' スロット名と意味制約条件の区切りを表し, スロットが「満たされていないスロット」(unsatisfied slot)であることを示す. '$'の後に, そのスロットのフィラーに関する意味制約条件を記述する. 1 つのスロットは, 図 4.3 の 1 つの節に変換される. 意味制約条件は, 節のボディの述語 sem の呼び出しに変換される.

   '→' 意味制約条件とアクションの区切りを表す. '→'の後に, フィラーが制約条件を満足した時起動されるプログラムを記述する.

   ':' スロット名とスロット値の区切りを表す. スロットが「満たされたスロット」(satisfied slot)であることを示す (図 4.4 の mammal のフレーム参照. (4-1)は,「ほ乳類(mammal)は温血(bloodTemp:warm)である」ことを表現する).

```
sem(open,1~SubIn~SubOut~subj:N~In~Out,_) :-
   ( ( sem(N,isa:event,_) ;
       sem(N,isa:thingOpen,_) ) ->
          addProp(object:N,In,Out) ),
     delete1(subj,SubIn,SubOut).
sem(open,2~SubIn~SubOut~subj:N~In~Out,_) :-
   ( ( sem(N,isa:instrument,_) ->
          addProp(instrument:N,In,Out) ) ;
     ( sem(N,isa:wind,_) ->
          addProp(reason:N,In,Out) ) ),
     delete1(subj,SubIn,SubOut).
sem(open,2~SubIn~SubOut~obj:N~In~Out,_) :-
   ( sem(N,isa:thingOpen,_) ->
       addProp(object:N,In,Out) ),
     delete1(obj,SubIn,SubOut).
sem(open,3~SubIn~SubOut~subj:N~In~Out,_) :-
   ( sem(N,isa:human,_) ->
       addProp(agent:N,In,Out) ),
     delete1(subj,SubIn,SubOut).                    (3-1)
sem(open,3~SubIn~SubOut~obj:N~In~Out,_) :-
   ( ( sem(N,isa:event,_) ;
       sem(N,isa:thingOpen,_) ) ->
          addProp(object:N,In,Out) ),
     delete1(obj,SubIn,SubOut).                     (3-2)
sem(open,3~SubIn~SubIn~with:N~In~Out,_) :-
   ( member(sem(V,object:X,_),In) ->
     ( sem(X,isa:thingOpen,_),
       sem(N,isa:instrument,_),
       addProp(instrument:N,In,Out) ) ;
     addProp1(demon(sem(V,object:X,_) ~
        sem(V,3~TSubIn~TSubIn~with:N~TIn~TOut,_)),In,Out) ).
                                                    (3-3)
sem(open,_,Prop,N) :-
   isa(act,Prop,N).                                 (3-4)
```

図 4.3: 動詞'open'の DCKR形式の辞書記述

```
mammal ::
 [bloodTemp:warm] → sem(mammal,bloodTemp:warm,_).
                                              ……(4-1)
 [isa:animal].    → sem(mammal,Prop,N) :-
                        isa(animal,Prop,N).

creature ::
 [age $ { is 1988 - birthYear!instance }].
                  → sem(creature,age:X,N) :-
                        sem(N,birthYear:Y,_),
                        X is 1988-Y.    ……(4-2)
```

図 4.4: SRL形式のオブジェクトの表現およびその DCKR への変換結果

3. AND,OR,NOT の記法はすべて，Prolog におけるものと同じ','，';'，'not' を用いており，制約条件にはこれらの任意の組合せを記述することができる．

4. 必須スロットは，そのスロット全体を'[',']'で，任意スロットは，そのスロット全体を'(',')'で囲む．4.2 節で述べた問題点に対する，この記法を用いた解決法は 4.3.2 で詳述する．

5. スロット間の共起関係を記述する記法として，

   *where*スロット名!*instance*制約条件

   という表現を，フィラーに関する制約条件に付加して記述することを許す．この表現は，スロット名が示すスロットのフィラーに関する意味制約条件を記述する．ここで，where,instance は予約語であり，

   スロット名!*instance*

   という表現により，他のスロットのフィラー（値）を参照する[96]．4.2 節で述べた問題点に対する，この記法を用いた解決法は 4.3.2 節で詳述する．

6. スロットの値（フィラー）に対する意味制約条件として，DCG (Definite Clause Grammar)[99] の場合と同じ記法を用いて，Prolog プログラム（図 4.4 の(4-2)における","で囲まれた部分）を直接記述することができる（(4-2)は，生物の年齢をその生まれた年(birthYear)から計算する付加手続きである）．

4.3.2 これまでのフレーム形式辞書の問題点の SRL による解決法

本節では，4.2 節で述べたこれまでのフレーム形式辞書の問題点を SRL でどのように解決するかについて述べる．なお，以下では，SRL の記述形式だけでなく，その内部表現（図 4.3 参照）も説

明に用いることにする．この内部表現は，図2の形式でユーザが記述した辞書をトランスレータにより変換することで得られる実際に動作するプログラムである．ただし，内部表現として我々は，本研究室で開発されたDCKR形式をとりあえず用いるが，本節の説明は，この内部表現に依存するものではないことに注意してほしい．

ケース1の場合に，意味的に異常な文を排除するためには，withスロットについて，次の意味制約条件が必要となる．

> フィラーがinstrumentであり，
> かつ
> 深層格がobjectであるスロットのフィラー（objスロットのフィラーは，解析の結果，
> 深層格がobjectとなる）はthingOpenでなければならない

この制約条件の後半部は，withスロットと共起する他のスロット（objスロット）のフィラーに対する意味制約条件である．この制約条件を，我々は，図4.2の(2-1)のように記述する．上のような意味制約条件を辞書に記述し，実際に解析に用いるには，ケース1で述べたように，現在の意味解析の途中結果を参照・検索する機構が必要となる．また，いわゆるデモン[68]と同等の動作をする機構が，意味解析プログラムに必要になる．これは，関係節などの場合に，用言を修飾する前置詞句を解析する際には，目的語となる名詞句が左外置されるために，まだobjスロットが満たされていないという状況に対処するため，デモンを作ってobjスロットが満たされるのを監視する必要があるからである．この意味制約条件の内部表現は図4.3の(3-3)である．(3-3)は，およそ次のような処理を行う．

> 解析の途中結果(In)の中に，深層格objectをもつフレームがあるかどうか検索し（objスロットが満たされているかどうか検査し），深層格objectをもつフレームが存在するなら，上で述べた制約条件を，withスロットのフィラー(N)およびobjスロットのフィラー(X)が満たすかどうか調べる．objスロットが満たされていないなら，上の制約条件の実行は不可能であるから，withスロットの解析をいったん中断し，objスロットが満たされるのを監視するデモン(「深層格としてobjectをもつフレームが現われたら（objスロットが満たされたら），自分自身（中断する解析）を起動せよ」というプログラム）を作る．

(3-3)のようなwithスロットの記述により，ケース1のような意味的に異常な文を排除することができる．

ここで述べたデモンは，CIL(Complex Indeterminates Language)[63]のように，Prologを拡張することで実現したものではなく，融合方式の機械翻訳[19]で用いられた関数WAIT-AND-SEE

に近いものと考えることができる．すなわち，デモン管理プログラムは，意味解析用プログラムの一部としてインプリメントしている．このデモンは，解析過程で絶えず起動の可能性を調べ，可能になった時点で即座に起動する，という点で，通常の意味でのデモン[68]をシミュレートしたものに近いと考える．デモン管理プログラムについては，次節で述べる．

次にケース２の問題点をどのように解決するかを述べる．図 4.1 のフレーム形式辞書は，図 4.2 の３つのフレームをまとめて記述したものであった．より厳密な意味解析を行うためには，我々は，図 4.2 に示す３つのフレームに分けて記述すべきであると考える．図 4.2 では，必須スロットは'['，任意スロットは'('でスロットを囲むことで，スロットの必須性・任意性を表現している．すなわち，

フレーム１について，subj スロットが必須，
フレーム２について，subj,obj スロットが必須，
フレーム３について，subj,obj スロットが必須，
with スロットが任意

である．英語の場合には，文を解析し終えた時点で，必須スロットのフィラーが満たされぬまま残っていてはならないわけである．そこで，記述した辞書を実行可能な内部表現，すなわち DCKR 形式に変換する際に，必須スロットのリスト（図 4.3 の各節における SubIn,SubOut）を付加し，必須スロットの解析が終了するごとに，そのスロットを必須スロットのリストから削除していく（図 4.3 の必須スロットを表現する各節の最後の述語'delete1'によって行う）ことにする．この必須スロットのリストは，HPSG[100]における SUBCAT feature と同等のものである．具体的には，たとえばフレーム３を用いて解析を開始する時点で，必須スロットのリストとして，[subj,obj]を SubIn にユニファイし，その後，必須スロットを解析するたびに，そのスロット名をリストから削除していく．そして，文の解析終了時に，必須スロットのリストにまだ要素が残っている場合（リストが[]でない場合）には，文脈から補完できないことを確認してから，そのフレームについての解析を失敗させる．必須スロットの要素を文脈から補完する省略補完処理については，6 章で述べる．任意スロットの解析は，必須スロットのリストとは無関係に実行され，リストの内容は変わらない（図 4.3 の節(3-3)のヘッドの SubIn 参照）．

4.3.3 SRL 形式の辞書を用いた意味解析

本節では，SRL で記述した辞書を用いた意味解析について述べる．例として，図 4.2 の SRL 形式の辞書を用いることにする．この辞書はトランスレータにより図 4.3 の DCKR 形式に変換され，変換されたプログラムが直接 Prolog 上で動作する．

DCKR 形式の辞書を用いた意味解析には，次のような特徴がある．図 4.3 では，

1. 1つのスロットを，述語 sem をヘッドとする1つのホーン節で表現している．これを以下では DCD節とよぶ．例えば，(3-2)は，obj スロットを表現している．(3-2)では，obj スロットを満たしうるフィラーは，" 事象" (event) もしくは" 開くもの" (thingOpen)でなければならず，また，フィラーがこの意味制約条件を満たすなら，アクションとして addProp という述語を呼び出し，フィラーの深層格が対象格(object)であるという意味を抽出する，ということが記述されている．

2. スロットの並びは，DCD 節の並びとして表現している．それにより，スロットの選択が，Prolog のバックトラック機構により自動的になされる．

3. スロットの意味制約条件が，DCD 節のボディに直接記述されている．従って，意味制約条件の検査は，Prolog プログラムを直接実行することに置き換えられる．

4. 知識の継承は，DCD 節(3-4)で行う．それにより，それより前のスロット(subj , obj , with)をフィラーが満たしえなかった時には，Prolog のバックトラックおよびユニフィケーション機構をそのまま利用して，上位の act にあるスロットを調べることができる．

5. Prolog の基本計算機構をそのまま利用して，DCD 節で記述した辞書項目を検索し，応答することができる．辞書がホーン節形式で記述されているわけであるから，辞書項目の検索は，検索事項をプログラムとして Prolog 上で実行することに置き換えられる．

このように，DCKR 形式を用いると，意味解析に必要なプログラムの大部分を，Prolog に組み込みの基本計算機構で代用することができる．また，計算効率も，従来のように，フレームを1つの大きなデータ構造として表現する場合に較べ，改善されることが期待できる．

意味解析は，DCG[99]形式で記述した文法規則中の補強項で行う．平叙文を解析する文法規則の例を次に示す．

```
sdec(SynVp,SemSdec) -->
    np(SynNp,SemNp),
    vp(SynVp,SemVp),
   {concord(SynNp,SynVp),
    interp(SemVp,subj:SemNp,SemSdec)}.
```

","で囲まれた部分が補強項である．述語 concord は，主語と動詞の人称・数の呼応を調べる述語である．述語 interp は，SemNp 中の名詞をフィラーとして，スロット名が subj の，SemVp に含まれる動詞（たとえば open）の辞書項目（たとえば(3-1)）を呼び出す．得られた意味解析結果は

SemSdec に返される．4.3.2節で述べたデモンの管理は，述語 interp で行い，デモンが作られている場合には，動詞の辞書項目を呼び出した後で，デモンの起動を試みる．述語 interp の定義を付録 Cに示す．以下，句"the door which I opened with a key"の意味解析過程を説明することにする．

1. 動詞句'opened with a key'で，動詞 open の with スロットの解析を行う．図 4.3 のフレーム3の with スロット(3-3)を満たそうとするが，obj スロットが満たされていないので，デモンを作成して with スロットの解析を中断する．

2. 埋め込み文'I opened with a key'に対して，動詞 open の subj スロットの解析を行う．'I'は意味制約条件（human であること）を満足するので，深層格を agent として解析に成功する（(3-1)参照)．そして，必須スロットのリストから subj が削除され，必須スロットのリストは[obj]となる．1で作られたデモンの起動を試みるが，obj スロットが満たされていないので，デモンは起動されない．

3. 動詞 open に対して，'door'をフィラーとして，obj スロットの解析を行う．'door'は意味制約条件を満足するので，深層格を object として解析に成功する（(3-2)参照)．そして，必須スロットのリストから obj が削除され，必須スロットのリストは[]となる．obj スロットが満たされたので，デモンが起動され，with スロットの解析を再開する．obj スロットのフィラー('door')が意味制約条件（thingOpen であること）を満足し，かつ with スロットのフィラー('key')が意味制約条件（instrument であること）を満足するので，深層格を instrument として解析に成功する（(3-3)参照)．

4.4 おわりに

本研究では，これまでの研究におけるフレーム形式を用いた辞書記述に関する問題点を具体的に指摘した．また，これまでのフレーム形式辞書の問題点を解決した辞書の記述形式として，辞書記述用言語 SRL を提案した．SRL で記述した知識は現在稼働中のトランスレータにより，DCKR 形式に変換され，変換されたプログラムが直接 Prolog 上で動作する．計算機上での辞書の内部表現として DCKR 形式を採用したのは，

> 自然言語の意味解析に，DCKR を用いると，辞書項目の柔軟な意味記述が可能になるとともに，それを利用した意味解析用プログラムの中核が，Prolog に組み込みの機能で代用可能になる

からである．

第 5 章

概念階層上での効率的な推論

本章では，概念階層[4]上で効率的な推論を行なうための，概念階層の内部表現形式および推論アルゴリズムを提案する．

5.1 Yet Another DCKR – 知識表現形式 DCKRの高速化

5.1.1 はじめに

我々が開発した知識表現形式 DCKR(Definite Clause Knowledge Representaion)[47]は，Prologのユニフィケーション機構をそのまま利用して知識の継承，手続き付加などが容易に行えるが，知識の量が増えるにつれて速度が低下するという問題がある[47,29]．この問題を解決するため，我々はDCKRの持つ知識検索に関する一般性と整合性を保ち，しかも高速に動作可能な知識の新しい内部表現形式を開発した．この新しい内部表現形式はPrologの宣言節として実現されており，それをどのように解釈して使うかを決める簡単なインタープリタと共に用いる．しかし，この内部表現形式で知識を直接記述することは必ずしも容易ではない．そこで高水準の知識表現言語 SRL（4章参照）を用いて記述することにする．SRLを用いて記述された知識はトランスレータにより，内部表現形式に変換される．

5.1.2 DCKRによる知識表現

オブジェクトの表現

本節ではDCKRを用いた知識表現の例を示す[47]．DCKRでは，オブジェクトを構成する各スロットを，semという単一の述語をヘッドとするホーン節として表現する．例えば，clyde#1は「象」で，「色は白」，象は「サイズが大きく」て「哺乳類」，哺乳類は「温血」であるという知識をDCKRで記述すると，(1a),(1b),(1c),(1d),(1e)のようになる．

```
:- op(100,xfx,:),
```

```
    op(90,xfx,#).
sem(clyde#1, color:white).                (1a)
sem(clyde#1, P) :-
        isa(elephant, P).                 (1b)
sem(elephant, size:big).                  (1c)
sem(elephant, P) :-
        isa(mammal, P).                   (1d)
sem(mammal, bloodTemp:warm).              (1e)
```

述語 isa の定義は以下のようである.

```
isa(Upper,P) :-
    P = isa:Upper ;
    sem(Upper,P).
```

述語 sem の第一引数は，オブジェクト名である．オブジェクトには大別して個体とプロトタイプがある．オブジェクト名のうち'#'の付いたものは個体名を，また'#'の付かないものはプロトタイプ名を表わす．例えば，(1a)(1b)の clyde#1 は個体名であり，(1c),(1d)の elephant はプロトタイプ名である．個体名（またはプロトタイプ名）が等しい述語 sem をヘッドとするホーン節の集合は，一つの個体（またはプロトタイプ）を表わす．例えば，(1a)と(1b)の記述は，clyde#1 という個体を表わし，(1c),(1d) の記述は，elephant というプロトタイプを表わす.

述語 sem の第二引数は，スロット名とスロット値の対である．たとえば(1a)の記述は「個体 clyde#1 の color が white である」という事実を表わす．color がスロット名で，white がスロット値である．スロット名とスロット値の対を以下では SV 対（属性）とよぶ.

知識継承と推論

(1b)は,「clyde#1 は elephant である」という事実を表すが，実は上位から下位への知識継承(inheritance of knowledge)の記述にもなっている．知識継承の観点からは，(1b)は「elephant が性質 P をもてば，clyde#1 も同じ性質 P をもつ」と読むが，この例のように，述語 isa でプロトタイプに連結された個体は，プロトタイプのインスタンスになる．例えば(1b)の記述により，個体 clyde#1 はプロトタイプ elephant のインスタンスである.

知識継承が Prolog のユニフィケーション機構によって自動的になされることを次に示す.

$$?-sem(clyde\#1, P).(2)$$

を実行すると，以下のようにclyde#1に関する知識を次々に（自動的に）得ることができる．

```
P = color:white;                (3a)
P = isa:elephant;               (3b)
P = size:big;                   (3c)
P = isa:mammal;                 (3d)
```

知識継承によりclyde#1の上位にある知識が全て得られていることに注意されたい．

$$?-sem(X,Y).(4)$$

を実行すると，XとYの対として全ての知識を出力し始める．また，

$$?-sem(X,size:big).(5)$$

を実行すると，Prologの特徴を生かして，

```
X = clyde#1;                    (6a)
X = elephant;                   (6b)
```

のように，mammalの下位に位置する個体とプロトタイプとを知ることができる．

DCKRの問題点

これまで説明してきたように，DCKRによる知識の記述は，Prolog組み込みの機能を極力そのまま推論に利用するという方針で設計されているため，極めて簡潔である．しかし，それ故に実行時の効率が悪い．すなわち，全ての知識がsemという単一の述語で記述されるため，知識の検索の際には全ての知識とマッチングを行わなくてはならない．前述の(5)のようにオブジェクト名が変数となっている検索の場合には特に効率の低下が著しい．以下にその理由を述べる．知識として(0a)-(0d)および(1a)-(1e)があるとする（図5.1参照）．

```
sem(taro#1, birthYear:1962).        (0a)
sem(taro#1, P) :-
        isa(monkey, P).             (0b)
sem(monkey, color:brown).           (0c)
sem(monkey, P) :-
        isa(mammal, P).             (0d)
```

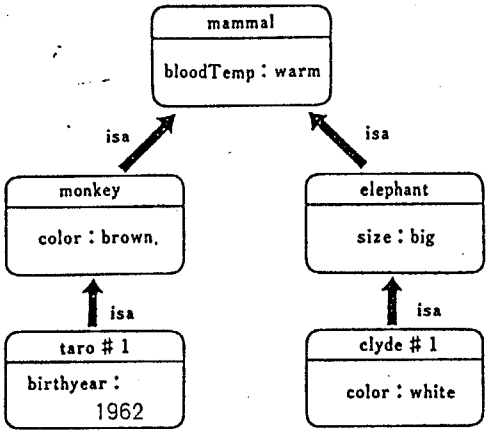


図 5.1: 概念階層の例

ここで(5)を実行すると，述語名はすべて sem であるから，(5)の述語 sem は(0a)-(0d),(1a)-(1e)全てとマッチングを行う．まず，(0a)とマッチするが，taro#1 の属性は size:big とは異なるので，これは失敗する．しかし，taro#1 自身が size:big という属性を持っていなくても，taro#1 の上位のオブジェクトがその属性を持っていれば taro#1 もそれを継承して，結局 size:big という属性を持つことになる．そこで，taro#1 の上位のオブジェクトがこの属性を持つかどうかを調べるために，(0b)の isa の知識によって概念階層を遡る．すると，taro#1 は上位オブジェクトとして monkey を持つことが分かるので，今度は monkey の属性を調べる．(0c)より，monkey の性質として color:brown が得られるが，これも size:big とは異なるので失敗する．しかし，monkey は(0d)より，mammal の下位オブジェクトであるから，mammal の属性も調べなくてはならない．(1e)より mammal の属性は bloodTemp:warm であるからこれも失敗する．

ここで注意すべきことは，これまでの検索で monkey や mammal の属性を調べているにもかかわらず，それらはあくまでも taro#1 の属性を調べるために継承階層を遡ったに過ぎず，後に monkey や mammal に対して，それ自身の属性として size:big があるかどうかを調べるために再度検索が行われることである．これがオブジェクト名が変数のときに DCKR が遅くなる主要な原因である．すなわち今の例で検索を続けると，オブジェクトとして taro#1 を考えた場合はすべて失敗したので，次はオブジェクトとして(0c)より monkey を考えることになる．monkey は属性として size:big を持たず，また monkey の上位のオブジェクトの mammal もそのような属性は持たないので，これもすべて失敗する．すなわち，ここまでの実行はすべて無駄であったことになる．実行のトレースを続けると，最終的にオブジェクトとして clyde#1 を選ぶことになる((1a)より)．

もちろん clyde#1 は属性として color:white しか持っていないのでこの段階では fail するが，clyde#1 の上位のオブジェクトの elephant がこの属性を持つので，clyde#1 は size:big という属性を持つことになり，(6a)の出力が得られる．先に述べたように，実際に size:big という属性を持っ

ているのは elephant であり，(6a)を出力するために elephant を調べに行っているにもかかわらず，elephant 自身が size:big という属性を持っているという出力(6b)がなされるのは，後に elephant をオブジェクトとしてマッチングを行うときである． このように，DCKR の実行ではあらゆるオブジェクトとマッチングを行い，その各々の場合に全ての概念階層を遡ろうとするので，速度の低下が起こる．今の場合は知識の量が極めて小さいのでこの程度であるが，知識が大きくなればこれは深刻な問題となる[29].

5.1.3 効率的な推論を実現する知識の内部表現形式

そこで我々は，DCKR の簡潔性を受け継ぎ，実行上のコンパチビリティを保ちながら，かつ高速で動作する方式を考案した．この方式では，知識はすべて内部表現として持ち，知識の検索は Prolog によって記述されたコンパクトなインタープリタによって行う．また，内部表現はそのまま記述するのではなく，記述言語を用いる．それぞれの内容について以下の節で説明する．

内部表現とインタープリタ

内部表現の記述例を示す．DCKR の(1a)に対応する知識は(8a1)-(8a2)，(1b)は(8b1)-(8b2)，(1c)は(8c1)-(8c2)のように表す．ここで，(8b1)-(8b2)は概念階層を表す内部表現である．

```
clyde1(color:white).          (8a1)
color(clyde1,white).          (8a2)

clyde1(isa:elephant).         (8b1)
elephant(subCon:clyde1).      (8b2)

elephant(size:big).           (8c1)
size(elephant,big).           (8c2)
```

後述するように，これは知識検索の道筋に冗長性を持たせたことに等しい．例えば，(8a1),(8a2)では，オブジェクト名が既知の場合には(8a1)の知識を，スロット名が既知の場合には(8a2)の知識を用いて検索を行う．(8b1),(8b2)では，上位のオブジェクト elephant が既知で，その下位にどのようなオブジェクトが存在するかを検索したい場合には，(8b2)の知識を用いる．逆に，下位のオブジェクト clyde#1 の上位にどのようなオブジェクトがあるかを知りたいときには(8b1)の知識を用いる．

このように，どの知識を用いて検索を行うべきかは，検索にあたり何が既知で何が未知であるかを知ることにより容易に判断することができる．それは Prolog で記述された小さな検索制御プログラム（後述する述語 ask）を介して行う．一つの知識に対して，オブジェクト名，スロット名を一つのファンクタ（述語）とした内部表現を用意しているので，そのいずれを未知にして検索する場合でも，既知のものをファンクタとしてもつ述語を選んで検索対象をすばやく取り出すことができる．それにより，検索の高速化を図ることができる．

具体的に(5)の実行では，スロット名が既知でオブジェクト名が未知（変数）であるから，スロット名 size をファンクタとする述語(8c2)を用いて，対応するオブジェクト名を取り出して，(8b2)により（上位から下位への）概念階層の知識を用いて階層を下りながら，そのスロット値を継承するオブジェクト名を求めていくことができる．検索制御を行うためのトップレベルの述語 ask は，オブジェクト名，スロット名のそれぞれをキーとして検索するための，２つの述語 prop,what を呼び出す．

```
ask(X,Y) :-
    nonvar(X) -> prop(X,Y) |                    (9a)
    nonvar(Y) -> what(X,Y).                     (9b)
```

prop は X，すなわちオブジェクト名が変数でないときに呼び出され，

1. オブジェクト名をヘッドとする内部表現を探して，マッチすればそのスロット名とスロット値の対を返して 2 に行く．

2. 概念階層を表す内部表現があれば，階層を遡って上位のオブジェクト名を取り出して 1 に行く，なければ fail で終了．

   what は X が変数で，Y が変数でないとき

3. スロット名をヘッドとする内部表現を探して，マッチすればオブジェクト名と，スロット値を返して 4 に行く．

4. オブジェクト名に対し概念階層を表す内部表現があれば，階層を下って（スロット値を引き継ぐ）下位のオブジェクト名を返して 4 を繰り返す，なければ fail で終了．

実行方法は DCKR とコンパチブルで，トップレベルにおいて

$$?-ask(clyde\#1, X).$$

とすると(3a)-(3d)が，

$$?-ask(X, size:big).$$

を実行すれば(6a),(6b)が得られる.

内部表現を用いた知識検索の具体的解析

5.1.2節で，DCKRは，オブジェクト名が変数の場合に特に効率が悪いことを述べたが，それでは本内部表現形式を用いると，それがどのように改善されるかを具体例で示す．5.1.2節の知識を本内部表現形式で表すと，(8a1)-(8c2)の知識のほか，以下のようになる.

```
taro1(birthYear:1962).
birthYear(taro1,1962).

        ..........

mammal(bloodTemp:warm).
bloodTemp(mammal,warm).
```

実行結果は次のようになる.

```
?- ask(X,size:big).        (10a)
  X = elephant;            (10b)
  X = clyde#1;             (10c)
```

(10a)を実行すると，オブジェクト名が変数(X)であるので，インタープリタはスロット名のsizeをヘッドとする述語を探しにいく．前述のように内部表現はオブジェクト名，スロット名のそれぞれをヘッドとした述語を持っているので，sizeをヘッドとする述語も存在するはずである．その結果(8c2)の表現とマッチする．(8c2)の第1引数より，sizeというスロット名を持つオブジェクトとしてelephantがあることが分かり，そのスロット値bigは，検索(10a)のスロット値と一致するので，これより(10b)の結果が得られる．elephantがsize:bigという属性を持つことが分かれば，elephantの下位のオブジェクトもその属性を引き継ぐので，あとは概念階層を順に下ってelephantの下位のオブジェクト名を出力していけばよい．すなわち，(8b2)よりelephantの下にclyde#1があることが分かるので即座に(10c)を得る．clyde#1の下にはもうクラスもインスタンスも無いのでfailする．さらに，sizeをヘッドとする述語は他には無いので実行を終了する．このように，内部表現形式を用いた方法では，size:bigという属性を持つオブジェクトを即座に捕まえることができ，あとは概念階層を下るだけでよいので，(余計な) 知識がたくさん記述されていても必要な知識のみを参照し，しかもDCKRのように同じ知識を何度も重複して参照することが無いため，かなりの

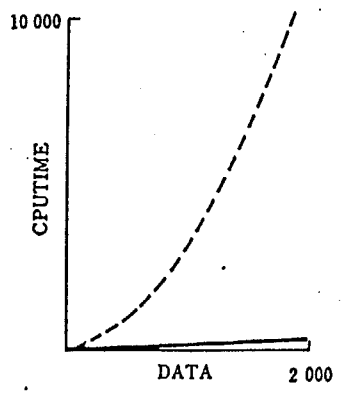


図 5.2: DCKR とインタープリタの速度比較（オブジェクト名が変数の場合）

高速化が期待できる．本節で述べた内部表現形式を用いた場合に，どの程度の検索速度の向上が見込まれるかを知るために実験を行った．それについては 5.1.4 節で述べる．

高水準知識表現言語 **SRL**

(8a1)-(8a2),(8b1)-(8b2)のような内部表現形式は，直接書くには煩雑であろう．より高水準の知識表現言語が必要である．我々は，4 章で述べた SRL を知識表現言語として用いる．SRL では，(8a1)-(8a2),(8b1)-(8b2)の知識を(11)のように書く．SRL 形式で書かれた知識(11)は，トランスレータにより(8a1),(8a2)の内部表現形式に変換される．

```
[clyde#1 ::
      [color:white]
      [isa:elephant]].                    (11)
```

5.1.4 検索速度の比較

DCKR とインタープリタの速度比較を図 5.2, 5.3 に示す．縦軸は 検索に要した CPUTIME，横軸は知識の数である．点線が DCKR で，実線が内部表現形式を用いた場合の実測値である．無論，継承の数やデータの種類によって速度は変化するが，どの場合も定性的な傾向はおおむねこのようになる．図 5.2, 5.3 から明かなように，DCKR では，知識の量が増えると検索時間が指数関数的に増えるのに対し，本方式では比例関係にある．したがって，知識の規模が大きいとき，もしくは(10a)のようにオブジェクト名を変数とする知識検索を行う場合には極めて有効であることが分かる．また，5.1.2 節でオブジェクト名が変数の場合 DCKR では検索に時間がかかることを述べたが，実際この場合（図 5.2）には他の場合と比べて検索のオーダーが一桁多くなっている．それに比べて本方式では他の場合とほとんど同じ時間で検索できる．

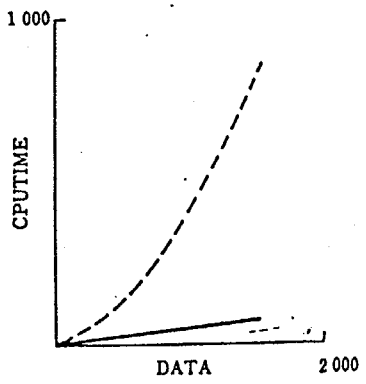


図 5.3: DCKR とインタープリタの速度比較（スロット値が変数の場合）

5.1.5 おわりに

概念階層上で効率的な推論を行なうための，概念階層の内部表現形式について述べた．次節では，この表現形式を用いた高速な推論アルゴリズムについて述べる．従来のように，記述した知識をそのまま推論に用いるのではなく，知識表現体系を，知識の記述，記述した知識のコンパイル，そしてそれを用いた推論の 3 つの段階として考え，記述した継承ネットワークを条件付きリンクや冗長なリンクを含まない等価な継承ネットワークに変換して推論を行う方法を提案する．

5.2 例外を含む多重継承ネットワークにおける継承アルゴリズム

5.2.1 はじめに

概念とそれらの間の関係をネットワーク形式で表現する手法は知識表現ではよく用いられる．ここで，概念はノード，それらの間の関係はリンクで表現される．概念体系を分類して階層構造に記述する際には，IS-A 関係が用いられる．この階層構造では，下位概念はその上位概念のもつ性質を継承することが原則であり，そのためこの表現形式を継承ネットワークと呼ぶ．継承ネットワークでは，性質の継承により知識の記述量に関して経済性が得られる．しかし，一般にこの性質継承には例外が存在する可能性がある．また，継承ネットワークが単純な木構造ではなく，下位概念が複数の上位概念をもつこともありうる．このような多重継承の場合には，複数の上位概念間の背反性を考慮しなければならない．このようなことから，例外を含む多重継承ネットワークでは，その継承アルゴリズムはかなり複雑になる．

例外を含む多重継承ネットワークに関する研究としては，[110,74,18]などがある．これらの最近の研究は，

1. 例外リンクや冗長なリンクを含む多重継承ネットワークにおいて正当な動作をする継承原理を議論する．

2. デフォルト論理[102]などの非単調論理を用いて例外を含む継承ネットワークおよびその継承アルゴリズムを定式化する.

3. 1の継承原理に基づいた並列アルゴリズムを開発する.

ものである. その中でいくつかの正当な動作をする継承アルゴリズムが提案されている[110,111]. しかし, これらの研究は, 人工的に作り出した複雑な継承ネットワークを扱うことができるような強力な継承アルゴリズムを提案するものであり, 提案された継承アルゴリズムは, 実際の知識からみると不要なほど強力であると考えられる.

これらの研究のように, 記述した知識をそのまま推論に用いるのではなく, 本研究では, 知識表現システムを, 知識の記述, 記述した知識のコンパイル, そしてそれを用いた推論の3つの段階として考える. そして, 記述した継承ネットワークを条件付きリンクや冗長なリンクを含まない等価な継承ネットワークに変換して推論を行う方法を提案する. この方法では, 実際に推論を行う継承アルゴリズムは比較的単純でよいことになる.

5.2.2 継承ネットワークの定義

継承ネットワークは非循環(acyclic)とする. また, リンクの始点を source, 終点を destination とよぶことにする. リンクとしては, IS-A,IS-NOT-A,exception リンクのみを考え, 性質名をラベルにもつリンクは説明を簡単にするため考えない (この性質名リンクを含むネットワークは, 性質名リンクを IS-A リンクに, そしてその destination ノードを「性質値」から「その性質名に対してその性質値をとる概念」に変更することにより, 等価なネットワークに置き換え可能である).

IS-A リンク　A $\longrightarrow$ B

実線矢印で表わす.

IS-NOT-A リンク　A $\not\longrightarrow$ B

横線付き実線矢印で表わす.

exception リンク　A ----→

点線矢印で表わす. IS-A,IS-NOT-A リンクと異なり, exception リンクは, ノードからリンクへのリンクであり, ノードがそのリンクに対する例外であることを表わす.

IS-A 関係は推移的であり, この IS-A 関係を通して性質継承は実行される. 単一継承ネットワークでは, 例外は, 下位概念が上位概念の性質と矛盾する性質をもつことに対応するが,

「矛盾する性質が存在しないかぎり, 下位概念は上位概念の性質を継承する」

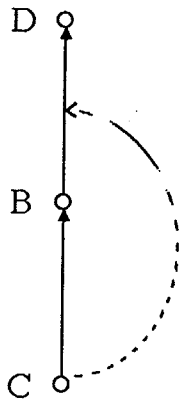


図 5.4: 下位ノードから上位リンクへの exception リンクを含むネットワーク

という原則に基づいて継承アルゴリズムを実現すれば，下位概念の性質が常に優先されるので，一般に例外は意識する必要がなく，従って陽に記述する必要もない．しかし，矛盾する性質をもつわけではなく，上位概念の性質の継承を単にブロックする記述を行いたいときや多重継承ネットワークでは，exception リンクを用いて例外を陽に記述しなければならないことがある．例を図 5.4, 5.5 に示す．ノード C からリンク B → D への点線のリンクが exception リンクであり，C が B から D への IS-A リンクに対する例外であることを表わす．従って，図 5.5 で，A は，C の性質を継承するため，B が D の性質を継承するのに対し，D の性質を継承しない．図 5.4 の継承ネットワークでは，exception リンクの source ノードが destination リンクからみて下位にあるのに対し，図 5.5 の継承ネットワークでは，exception リンクの source ノードは destination リンクと上位／下位関係にない．このため，図 5.4 の継承ネットワークでは，上に述べた継承の原則をそのまま用いて exception リンクが扱えるのに対し，図 5.5 の継承ネットワークでは，exception リンク及びその destination リンクである IS-A リンクに対し，（その IS-A リンクを用いて推論を行う際，exception リンクが成り立つかどうか調べるという）余分な推論を実行しなければならない．この推論の問題点及びその解決法は 5.2.3 節で述べる．

5.2.3 継承ネットワークのコンパイル

冗長なリンクの除去

最短パスをたどって継承を行う推論方式が，従来継承アルゴリズムとして用いられることが多かった．しかし，Touretzky ら[110,111]は，この最短パス・アルゴリズムを用いると，冗長なリンクを含むネットワークでは誤った結論を導く可能性があることを指摘し，この問題を解決する継承原理として，パスの長さではなく，継承の途中経路を考慮した推論距離順序(inferential distance ordering)という考え方を提案している．Touretzky らは，最短パス・アルゴリズムで問題となる

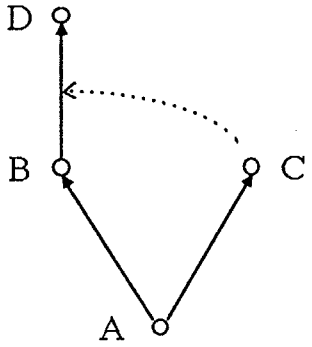


図 5.5: 下位ノードではないノードからの exception リンクを含むネットワーク

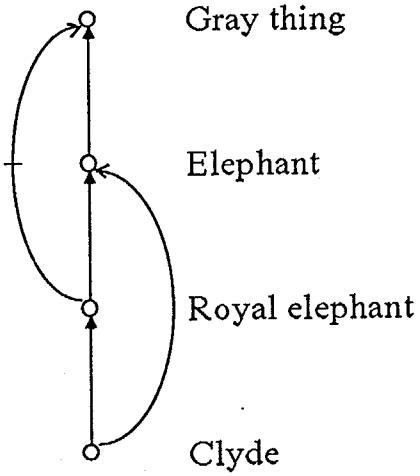


図 5.6: 冗長なリンクを含むネットワーク

ネットワークの例として図 5.6（[111]より引用.）に示すものを挙げている．最短パス・アルゴリズムでは，Clyde から Elephant への冗長な IS-A リンクがあるため，Clyde に関して，Gray でない（Royal elephant が Gray でないことに基づく），Gray である（Elephant が Gray であることに基づく）という矛盾した結論が導かれてしまう．

しかし，本来知識表現において冗長なリンクは何の役割も果たさないはずである．また，知識は整合的に保たれるべきであることを考えると，冗長なリンクは知識管理機構[59]により除去されるべきである．このような考え方に基づき，冗長なリンクを除去した継承ネットワークで推論を行う立場を主張する．

まず，冗長なリンクを定義する．ノード間を直接結ぶ IS-A リンクを直接リンク，1 つ以上のノードを経由して IS-A リンクで結ばれている時には間接リンクで結ばれているということにする．ノード間に 2 つ以上のリンクがあるのは次の 3 つの場合である（図 5.7 参照）．

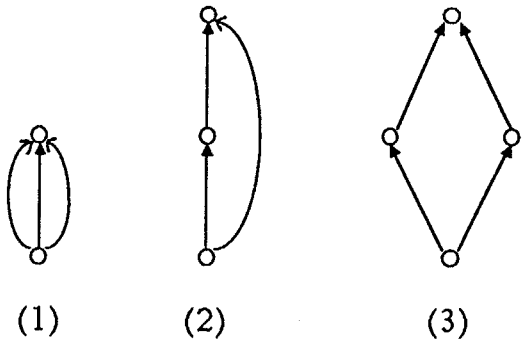


図 5.7: ノード間に複数のリンクがある場合

1. ノード間に 2 つ以上の直接リンクが存在する.

2. ノード間に直接リンクと間接リンクが存在する.

3. ノード間に 2 つ以上の間接リンクが存在する.

1 の場合, すべての直接リンクは等価であるから, 1 つを除いて残りのリンクはすべて冗長なリンクと判断し除去する. 2 の場合, 直接リンクから得られる情報は, 間接リンクを通して得られる情報の部分集合になっているので, 直接リンクは冗長なリンクとみなし除去する. 3 の場合は, それぞれの間接リンクから得られる情報の中にお互いに含まれない部分が存在するので, どのリンクも冗長なリンクとは考えず, 多重継承として扱う.

この冗長なリンクの検出・除去は, ユーザが記述する知識の一貫性を動的に保持する知識管理機構によって実行される. 我々は, 冗長なリンクの除去機能を含むシソーラス作成支援ツールを逐次型推論マシン PSI 上に作成している[58].

exception リンクの展開

図 5.4, 5.5 のような exception リンクを含む継承ネットワークをそのまま用いて推論を行う方法が従来用いられている[42]. しかし, [42]で述べられているように, この方法では, 図 5.5 のように source ノードと destination リンクの間に上位／下位関係のない exception リンクを含む継承ネットワークでは, 5.2.2 節で述べた継承の原則をそのまま用いることができず, 本来ごく少数であるはずの例外に対して, 継承の過程で例外であるかどうかを必ず調べなければならず, 推論の効率を著しく低下させる.

そこで, 記述された継承ネットワーク中の図 2 のタイプの exception リンクを図 5.4 のタイプの exception リンクに変換し, 変換後の図 1 のタイプの exception リンクしか含まない等価な継承

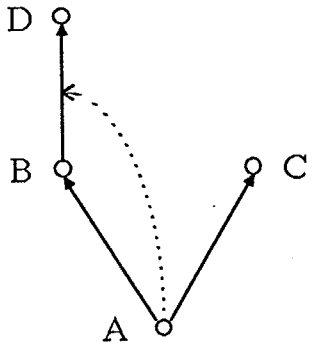


図 5.8: 図 5.5 のネットワークのコンパイル後のネットワーク

ネットワークを用いて推論を行う方法を提案する．図 5.4 のタイプの exception リンクしか含まない継承ネットワークでは，5.2.2 節で述べた継承の原則に基づいて容易に例外を扱うことができる．図 5.5 の継承ネットワークは図 5.8 の継承ネットワークに変換される．変換の原則は，

> exception リンクの source ノードを，exception リンクの source ノード，destination リンク両方の下位ノード中，最上位のノードに変更する

である．図 5.5 では，exception リンクはノード C からリンク B → D へ向かっているので，その source ノードを，C から，C,B 両方の下位ノード中最上位にある A に変更する (図 5.8 参照)．図 5.5 の例から明らかなように，変換後の exception リンクの source ノードである，変換前の exception リンクの source ノードと destination リンク両方の下位ノード中最上位のノードは，その exception リンクの destination リンクの例外となる最も一般的な(most generic)ノードであり，かつこのノードの性質を継承しないノードはすべて，例外とならないことに注意してほしい．すなわち，このノードの性質を継承するノードだけが例外となる．これは，例外となるノードをコンパイル時に circumscribe し[92]，推論時の負担を軽くしていることになる．

また，exception リンクは，[54]でいう制約リンクのうち，負のリンクに相当しており，その観点から見ると，この exception リンクの展開は，リンクに付加されている制約を推論実行前に部分計算[43]していることに相当する．

5.2.4 継承アルゴリズム

5.2.3 節で述べた継承ネットワークのコンパイル後に実際に推論を行う継承アルゴリズムについて説明する．5.2.3 節で述べたように，単一継承ネットワークの場合には，ネットワークコンパイル後の継承アルゴリズムは，5.2.2 節で述べた原則

> 「矛盾する性質が存在しないかぎり，下位概念は上位概念の性質を継承する」

に基づいていれば十分である．問題となるのは，多重継承ネットワークの場合である．5.2.1節で述べたように，多重継承ネットワークの場合には，複数の上位概念間で得られる結論に矛盾が生じる場合がある．この矛盾として以下の2通りを考える．

1. 背反な上位概念に到達する．ここで，概念の背反性とは次のようなものである．概念体系を分類する際には,必ずある見方[1]をとることにより下位概念に分類している．その1つの見方で分類した下位概念同士は互いに背反である．

2. 上位概念同士は互いに背反ではないが，それらのもつ性質の中に矛盾するものが存在する．

このため，多重継承ネットワークを扱う継承アルゴリズムは以下のようになる．

1. 多重継承するノードが現れたらその上位概念をパス名[41]として分岐する．

2. それらの上位概念が互いに背反であるかどうか検査する．

3. それぞれのパスを上位方向にたどる．

4. 拡張(extension)を作成する．

   (a) 2で上位概念が背反な場合
   得られる性質集合は，それぞれのパスにより得られる性質集合の選言(disjunction)をとったもの

   (b) 上位概念が無矛盾で，得られる性質にも矛盾するものが存在しない場合
   得られる性質集合は，それぞれのパスにより得られる性質集合の和集合

   (c) 上位概念は無矛盾であるが，得られる性質に矛盾するものが含まれる場合
   得られる性質集合は，無矛盾な性質集合と，矛盾する性質に関してその選言をとった集合の和集合

以下では，図5.9（[111]より引用.）の継承ネットワークに関して，上で述べた継承アルゴリズムを用いて推論を行う過程を示す．

1. Nixonから継承ネットワークを上に向けてたどる．

2. QuakerとRepublicanは互いに背反ではないので，それぞれのパスを上にたどる．その結果，それぞれのパスに対して，次の性質集合が得られる．

Q	P/Q	A/P/Q	
R	～P/R	F/R	～A/F/R

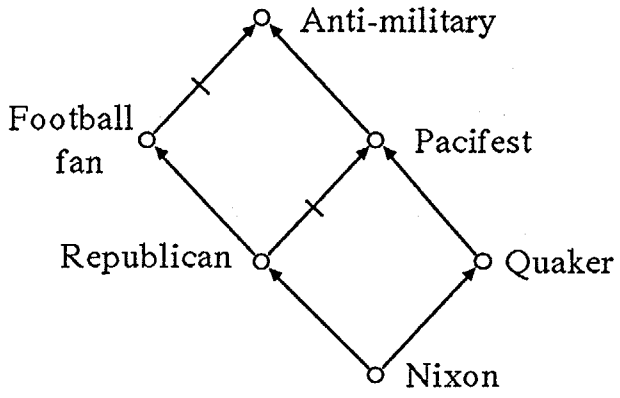


図 5.9: 曖昧なネットワーク

ここで，'～'は否定を表わしており，各概念はそのイニシャルのアルファベットで表現している．また，パスの履歴は，'/'を用いて表現する．このパスの履歴は，命題の成立条件を表わしていると考えられ（例えば，P/Q は，P if Q を表わしている），次の拡張の計算で用いられる．

3.

(a) 互いに矛盾しない Q,R を無矛盾な性質集合に加える．

$$\text{無矛盾な性質集合}\{Q, R\}$$

それぞれのパスの性質集合中の条件から Q,R を除去する．その結果，それぞれのパスに対する性質集合は次のようになる．

P　　A/P

～P　　F　　～A/F

(b) P,～P は互いに矛盾する性質が存在するので，無矛盾な性質集合には付加できない．次に，F が，Q,R と同様に，無矛盾な性質集合に付加され，性質集合中の条件から除去される．この時点で，それぞれのパスに対する性質集合は次のようである．

P　　A/P

～P　　～A

(c) ～A も矛盾する性質が存在するので，従ってこれ以上無矛盾な性質集合は拡張することができず，結果として

$$\text{無矛盾な性質集合}\{Q, R, F\}$$

矛盾する性質集合$\{P, A/P, \sim P, \sim A\}$

が得られる．

(d) 矛盾する性質集合から無矛盾な最大部分集合の集合を計算する．その結果，

$$\{\{P, A\}, \{P, \sim A\}, \{\sim P, \sim A\}\}$$

が得られる．そして，最終的に

$$\{Q, R, F\} \cup \{\{P, A\}, \{P, \sim A\}, \{\sim P, \sim A\}\} =$$

$$\{\{Q, R, F, P, A\}, \{Q, R, F, P, \sim A\}, \{Q, R, F, \sim P, \sim A\}\}$$

が性質集合として得られる．この集合の要素である3つの集合はそれぞれが1つの拡張を表わしている．

このように曖昧な継承ネットワークにおいて，可能な複数の拡張をすべて得る点で，信心的(credulous)[111]な立場をとることになる．

5.2.5 おわりに

従来のように，記述した知識をそのまま推論に用いるのではなく，知識表現体系を，知識の記述，記述した知識のコンパイル，そしてそれを用いた推論の3つの段階として考え，記述した継承ネットワークを条件付きリンクや冗長なリンクを含まない等価な継承ネットワークに変換して推論を行う方法を提案した．この方法では，実際に推論を行う継承アルゴリズムは，単一継承ネットワークに関しては，

「矛盾する性質が存在しないかぎり，下位概念は上位概念の性質を継承する」

という直観に合った原則に基づいていれば十分であり，多重継承ネットワークに関しても，5.2.4節で述べたような，複数の上位概念に対するパスから無矛盾な拡張集合を計算する手続きを付加するだけでよい．

第 6 章

照応，省略に関する曖昧性解消

6.1 はじめに

本章では，2 章で述べた曖昧性のうち，

- 指示語の指示対象を同定する照応参照
- 省略されている格要素を補う省略補完

に関する曖昧性解消方法について述べる．

照応および省略現象は，その指示する対象（補完される対象）が文脈中に存在するか，または「発話の場」に存在するかにより大きく 2 つに分けられる[50,60]．本論文では，文脈中から対象を決定する場合（前者）のみを考え，「発話の場」に存在する対象は考慮しない．この点に関しては，7.2.5 節でさらに述べる．

照応現象は，「これ」，「それ」などの指示代名詞や，「この」，「その」などの指示形容詞によって発生する．例として，指示形容詞による照応現象を説明する．照応現象を含む文の例として[49]には，以下のものが挙げられている．

1. (a) すずめがいる．
   (b) そのすずめは足に怪我をしていた．

2. (a) すずめがいる．
   (b) その鳥は足に怪我をしていた．

3. (a) 自転車を買った．
   (b) 花子はそのサドルの高さを調節した．

4. (a) 凸レンズを置く．
   (b) その左に凹レンズを置く．

5. (a) 自転車を売った.

(b) その金でラジオを買った.

6. (a) 地震が発生した.

(b) その時太郎は学校にいた.

1は，先行詞[1](「すずめ」) と同一単語を含む照応詞[2](「そのすずめ」) による参照の例である．2は，先行詞の上位概念を含む照応詞 (「その鳥」) による参照の例である．3の例は，1, 2とは異なり，既出の「自転車」を「そのサドル」が参照しているのではなく,「そのサドル」のうち,「その」によって「自転車」が参照されている．4も同様で,「その左」の「その」によって，前に現われた「凸レンズ」が参照されている．これら４つのタイプの照応現象は，先行詞と指示形容詞の修飾している単語の間の次のような意味的関係によって特徴づけられている[3].

1. 指示形容詞の修飾している単語が先行詞と同じ概念を表している

2. 指示形容詞の修飾している単語の表す概念が先行詞の表す概念の上位概念になっている

3. 指示形容詞の修飾している単語が先行詞の部分を表している

4. 指示形容詞の修飾している単語が先行詞からの相対的な位置関係を表している

5, 6では，文(a)全体が指示語により参照されている．5では,「金」が前文で記述されている出来事(行為) と因果関係を持っていると考えられる．6では,「その時」により，前文の出来事が発生した時間を表している．このような文を指示する照応現象はここでは扱わず，7.2.5 節で今後の課題として述べる．

以上述べたように，照応参照による指示語の指示対象の同定処理は，指示語との間にある関係が成り立つような適切な先行詞を文脈中から見つけ出すことである．

省略は，日本語には非常に頻繁に見られる現象である．付録 Aに示した例文にも省略は数多く見られる．省略されたものを明示化した形でその一部を以下に示す．

1 三角フラスコにアルミニウムを入れる.

2 (Xに) うすい塩酸を注ぐ.

3 3本の集気びんに発生する気体を水上置換で集め，ガラス板で (Xに) ふたをしたまま，(Yを) 机の上にさかさに立てる.

4 1本目の集気びんに石灰水を入れて，(Xを) ふり混ぜる.

[1] 指示語により参照されている既出の単語を先行詞(antecedent)と呼ぶことにする.
[2] 指示代名詞や，指示形容詞を含む名詞句などの照応現象を引き起こす句を総称して照応詞(anaphor)と呼ぶことにする.
[3] 1, 2のタイプの照応を限定指示，3, 4のタイプの照応を代行指示という[50,60].

5　2本目の集気びんに，草花を入れる．また，においも調べる．

6　3本目の集気びんを，口を下にして持ち，（xに）火のついた線香を入れる．

例文中括弧で囲まれた格要素が実際には省略されていたものである．これらの省略された格要素を文脈から正しく補う処理が省略補完処理である．

このように，照応，省略に関する曖昧性解消処理は，これら2つの現象を含む文を正しく理解するため，文脈中から必要な対象を見つけ出すことになる．これらの処理は大きく次の2つのステップに分けられる．

1. 文脈中に存在する対象の中から候補となる対象を選択する．

2. 選択された対象の候補が妥当かどうかを検査する．

2では，

- その候補と指示語との間に上述した適切な意味的関係が成り立っているかどうか
- その候補が動詞の選択制限を満足しているかどうか

の検査を行なう．これは2.3.4節で述べた拡張ユニフィケーションにより実行される．図2.7はその例である．このユニフィケーションにより，指示する（または省略されている）対象の意味するものが同定される．

文脈中には通常複数の対象が存在するため，その中から1つを選択する1のステップが，本章で述べる照応，省略に関する曖昧性解消処理の主眼となる．ステップ1で用いるような対象に固有の制約（情報）以外に，照応，省略にはそれに付随する様々な制約（情報）が存在する．文脈中に存在する対象の中には，後述する言語的制約などにより対象の候補になりえないものが存在する．また，候補集合の中の対象には，後述する様々な文脈的情報により，候補としての適切性に順序（優先順位）付けが可能である．このような情報を用いて，文脈中に存在する対象の中から可能な候補のみを集め，適切と判断される候補から順番に選択することは，対象の候補の曖昧性の解消を意味する．

状況意味論(Situation Semantics)，談話表示理論(Discourse Representation Theory;DRT)などの最近の形式意味論を用いた照応現象のモデルはいずれも，この「複数存在する候補の中から適切な対象を1つ選択する」というステップを重要視せず，考慮に入れていないため，非常にナイーブなものになっている．DRTでは，文に対応するDRS(Discourse Rerepsentation Structure)を構成し，出現した名詞句は参照マーカ(reference marker)としてそのDRS中に記録しておく．そして，照応現象は，その中から「適切な」(suitable)要素を選択することにより説明する．この「適

切な」要素をどのようにして決定するかについては全く述べられない[84,22,31]. しかし，談話表示理論は，限量子のスコープと照応との間の相互関係による言語現象（例えば，ロバ文(donkey sentence)）を明確に説明する理論として興味深い[4]. 状況意味論に基づく照応モデルでは，インデキシングにより同一のインデックスを持った要素同士は同じ指示対象を持ち，照応関係にあると考える[56,94]. しかし，このインデキシングの戦略はこのモデルの範囲外であるとし，適当な指示対象を選択する機構を仮定している.

6.2 節以後，このステップ 1 における

- 候補集合の縮小
- 候補集合からの選択の順序付け

について述べる. これらの処理に用いられる主な情報として，

- 統語的制約
- 談話に関するヒューリスティクス
  1. 焦点
  2. 談話構造

を取り上げ，それぞれについて述べるとともに，6.5 節では，これらの情報を統合した本モデルにおける照応，省略に関する曖昧性解消アルゴリズムについて述べる. また，上で述べた

- 指示語の指示するものの同定
- 省略語の補完

以外に，文脈解析において重要であると考えられる

- 主題や焦点の抽出
- 常識的推論

の本システムにおける実現方法にもふれる.

次節ではまず，本論文では特に扱わない情報を用いたいくつかの研究を概観する.

[4] [84]で参照マーカを，限量子のスコープと照応現象の関係を同様に扱っている Webber[113]（6.2 節に後述する.）の談話実体(discourse entity)と同様のものであるとコメントしているのはこのためである.

6.2 照応処理に関するいくつかの研究

照応処理に関する研究に関しては，いくつかのサーベイが存在する[79,67,107]．それらを参考にして，本論文では特に言及せず，今後の課題とするいくつかの情報（知識）を用いた研究について述べる．

1. Webber[113]の研究

   Webber は，限量子のスコープの範囲と照応現象の間の関係についていくつかの考察をしている．また，その考察に基づき，論理式を表現形式として照応現象の説明を行なっている．残念なことに，この Webber の研究も，上述した談話表示理論の場合と同様，「複数存在する候補の中から適切な指示対象を１つ選択する」手法については全く言及していない．

2. Hobbs[81]の研究

   Hobbs は，談話中の文の間には，何らかの整合的関係(coherence relation)が成り立っていることが不可欠であると主張し，それらの関係としていくつかのプリミティブを定義している．そして，照応は文間の整合的関係による副次的な現象であるとする．すなわち，文間の整合的関係が成り立つために必要な，「異なる２つの名詞句が同一の指示対象を持つ」という制約により照応現象を説明する．具体的な処理方法としては，文を表現する内部表現中，代名詞には変数を対応づけ，文間の整合的関係を検査する過程で，その変数に束縛された値がその代名詞の指示対象であるとする．文間の整合的関係により談話にはある構造が与えられるという点で，この研究は後述する談話構造に関する研究と関連がある．

3. 日本語の指示語は，「これ」，「それ」，「あれ」の例から明らかなように，「コ・ソ・ア」の３つの体系に分類される．この３つの体系はその用法が微妙に異なるとされている[28,13]．この用法の違いを風斗[52]は，共有知識(mutual knowledge)の概念を用いて説明している．共有知識とは，話者と聴者の両方が持っている知識のことであり，概略，「ア」系は共有知識内の対象を，「ソ」系は共有知識外の対象を指示するとしている．

これら

1. 限量子のスコープに関する情報（制約）
2. 文間の整合的関係に関する情報
3. 話者（書き手）と聴者（読み手）の共有知識の概念

は，本論文では考慮していない．2, 3を用いた処理では常識的推論を考慮せねばならず，本モデルの曖昧性解消アルゴリズムが目指す，後述するような比較的浅いレベルでの処理とは相反するからである．

6.3 統語的制約

本節では，照応，省略に関する曖昧性解消に用いる情報の１つ目として，統語的な制約について述べる．

１文内での照応や省略現象に関しては，理論言語学の分野で数多く研究がなされている．c-統御(c-command)[101]の概念はその１つである．

Hobbs[82]は，この c-統御などの概念を用いた照応処理の統語的なアルゴリズムを提案している．Hobbs は，この統語的な情報のみを用いた手法で驚くほど良い結果が得られると述べている．Hobbs が用いた統語的制約は，以下のものである．

1. 再帰代名詞以外は，その先行詞が同一文中に現れてはいけない．
2. 代名詞の先行詞は，代名詞より前にあり，代名詞を c-統御しなければならない．

   ここで，ノード NP_1がノード NP_2を c-統御するとは，

   > NP_1と NP_2が互いに支配関係（上下関係）になく，NP_1を支配する最も近い Sノードが NP_2を間接的に（直接ではなく）支配する

   ことをいう．

Carbonell[66]は，照応，省略現象の処理には，文間の様々なレベルでの並列性に関する情報が重要であることを指摘している．そして，

- 等位構造などのように，２つの文が対照されている場合には，対応する位置にある要素が照応詞の指示対象として優先する．
- 照応詞の指示対象として，以前の文の対応する格スロットを満たしていた要素を優先する．

のような優先性を挙げている．

本論文でもこれと同様の統語的制約を用いている．それについては，6.5 節で述べる．

6.4 談話に関するヒューリスティクス

本節では，照応，省略に関する曖昧性解消の際，指示対象としての適切性の順序（優先順位）付けをするのに用いる文脈的情報について述べる．

6.4.1 焦点

本節では，談話の中心にあるものとしての焦点(focus)の概念を用いたいくつかの照応処理研究を概観する．これらはいずれも，

焦点は照応参照で指示されやすく，また，指示された対象は焦点になりやすい．

という仮説に基づいている．焦点は談話の進行に伴い変化するため，その変化過程を表現する何らかのデータが必要になる．

1. Grosz[77]は焦点空間(focus space)を用いて現在の焦点を表現する．また，焦点と関連する対象も'implicit focus'として登録しておくことにより，それらの対象への焦点の移動を可能にする[5]．また，焦点空間の階層は，作業(task)に関する談話の構造に対応するとしている．すなわち，1つの作業は1つの空間に対応し，ある作業Aの部分となる作業はAの空間の副空間を表現する．この研究は，6.4.2節で述べるGroszらの談話構造の研究[78]へと発展する．

2. Sidner[107]の焦点モデルは，焦点を扱った研究の中で体系的なものとして最初のものである．6.5節で述べるアルゴリズムでは，焦点の情報に関してこのモデルを参考にしている．以下では，このモデルに関して簡単に述べる．

   Sidnerのモデルでは以下の6つのレジスタを用いて焦点の状態を表す．

   - DF(Discourse Focus), AF(Actor Focus)
     現在，焦点となっている要素を登録する．
   - PDFL(Potential Discourse Focus List), PAFL(Potential Actor Focus List)
     前文に出現した要素を新しい焦点の候補として蓄積する．
   - DFS(Discourse Focus Stack), AFS(Actor Focus Stack)
     過去の焦点の履歴を保存する．

   AFは動作主格(agent)の位置に現れる「生物」(animate)のみを扱う焦点として，DFと区別される．

   照応の処理アルゴリズムは以下のようになる．

   (a) 最初の文に対して焦点の候補を設定する．

   (b) 焦点を使って照応の解釈を行なう．

   (c) 解釈した照応により焦点の状態 (各レジスタの値) を更新する．

[5] [34]では，KM-スタック（仮説名詞スタック）を用いてこの機能を実現している．

(a)を最初の文に適用したのち，(b)と(c)を文中に照応表現が現れるたびに適用する．(b)で焦点が照応の解釈に用いられた場合には，焦点の移動は起きず，焦点は変化しない．焦点を照応の解釈に用いることができない場合，焦点の候補を蓄積している'Potential Focus List'中の要素を用いて解釈を試みる．解釈に成功した場合，今度は焦点の移動が起きたと考え，古い焦点を'Focus Stack'に積み，解釈に用いた'Potential Focus List'中の要素を新しい焦点とする．

焦点の候補は次の優先順位によって順序づける．

(a) 対象格

(b) 動作主格を除く要素

(c) 動作主格

(d) 動詞句

3. 亀山[85]の'centering'理論では，英語だけでなく，日本語における焦点と照応現象との間の関連も考察している[6]．'centering' 理論における'backward center'はSidnerの焦点モデルにおける'discourse focus'に，'forward center'は'potential discourse focus'にそれぞれ対応する．また，亀山は，焦点を決定する際には，'empathy'の概念[90]も考慮する必要があることを指摘している．

4. Carter[67]は，Sidnerのモデルの拡張を試みている．Carterは，Sidnerのモデルの問題点として，

   - 同一文中の指示対象に対処できない．
   - 指示対象の候補の確認にc-統御などの統語的制約を用いていない．

   などを挙げ，それらの点の拡張を行なっている．また，焦点の候補の優先性として，

   (a) 新しく現れた要素ほど指示対象として指示されやすい(recency)．

   (b) 主節よりも埋め込まれた節の要素を指示対象として優先する．

   などを導入している．さらに，前文の動作の結果出現した対象を指示対象とする場合などのように，常識的推論を用いないと解消できない照応があることを指摘している．

焦点の他に，日本語では，係助詞「は」により示される主題(topic)の情報も得られる．主題については，久野[12]，三上[21]で詳しく述べられている．

[6] [85]では，'focus'ではなく，'center'という用語を用いている．

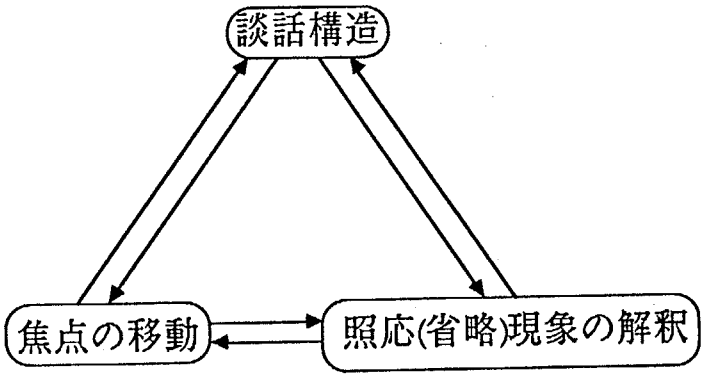


図 6.1: 談話構造，焦点，照応（省略）現象の関係

6.4.2 談話構造

本節では，談話構造が照応処理において果たす役割について述べる．

一連の文の集まりとしての文章には，文にその統語構造があるように，談話構造(discouse structure)があるとされる[7]．すなわち，文章は，いくつかの文の並びが１つのノードに対応し，それらのノードが何らかの関係により階層をなしているものとして表現する．談話構造は，談話の進行過程を表現するものなので[8]，談話中の焦点の移動や，それに付随した照応表現の解釈，省略語の補完を制約する．逆に，照応表現の解釈や焦点の移動の仕方により，談話の構造には制約が課される[78,37,39,51]．すなわち，これら３つのものは，図 6.1に示すような相互関係にあると考えられる．

このような興味深い考察がなされているにもかかわらず，談話構造をどのように構築するかという研究では，あまり成果が得られていない．[78]などでは，談話構造の構築は研究の範囲外であるとされている．

談話構造の構築に関する最近の研究としては，

1. スクリプト(Script)[103]のような，典型的な談話の構造をあらかじめ用意しておく

2. ある限定した領域に関して，それらの領域に特有の情報を用いて談話構造の構築を行なう

ものがある．2の研究としては，

1. Grosz[77]のように，話者の目標(goal)に依存した談話について，その構造の構築を目指した研究

[7] DRT の DRS も談話構造の表現の１つであると考えることができる．
[8] 談話の進行過程を表現するものとして，談話構造中現在の談話の位置（ノード）を示すポインタが通常用意されている

2. 議論の談話を取り上げ，命題間には主張とその証拠の関係があるとして，その関係により木構造の構築を試みた Cohen[70]の研究

などがある．残念なことに，談話の領域を限定せず，その談話構造を構築しようという研究はなく，本論文でも領域を限定した談話についての考察しか行なっていない．これは今後の課題である．

上述した研究で談話構造の構築に主に用いられてきたのは，文章中に出現する'clue word'の情報である．英語における'and'などの連結詞(connective)や，日本語の「そして」などの接続詞は，文間の関係を表現するために用いられるので，談話構造の構築にはこの情報が手がかりとして利用可能であるとの考察に基づいている[76,55,70]．言い換えると，'clue word'は，談話構造を陽に示すために用いられるものである．Cohen[70]は，英語の連結詞をいくつかに分類し，それぞれの表現する文間の関係を示している．また，それ以外に，談話の以前の箇所へ話題を戻すための'clue word'が存在することも述べている．これから述べる内容を予告するような表現（「以下では，…について述べる.」など）も'clue word'の一種と考えられる．

また，桃内[39]は，主題と談話構造の間の関係について述べている．

6.5 照応，省略に関する曖昧性解消アルゴリズム

本節では，6.3, 6.4 節で述べた情報を統合して用いた照応，省略の曖昧性解消方法について述べる．

6.5.1 節では，本モデルで曖昧性解消に用いた情報について述べる．6.5.2 節では，それらの情報を用いたアルゴリズムを例文を用いて説明する．

6.5.1 照応，省略現象の説明に用いる情報

6.2, 6.3, 6.4 節で，照応，省略に関する曖昧性解消に用いることができる様々な情報について述べた．これらの情報は，統語的制約のように，浅い解析で処理可能なものから，複雑な常識的推論を用いなければ処理できないものまで様々である．本モデルでは，照応の多くは浅い解析で解釈できるという Carter[67]の主張に基づき，比較的浅いレベルでの解析のみにより，照応，省略の曖昧性解消を実現する．

以下では，本モデルで照応，省略の曖昧性解消に用いる情報について述べる．

比較的浅い処理で実現が可能な統語的制約，優先性として，次のものを導入する．

1. 再帰代名詞以外は，その先行詞が同一文中に現れてはいけない．省略語（ギャップ）が同一文中の要素と co-index されることはない．

   太郎が ϕ 殴った．

ここで，ϕ[9]は，太郎ではない．

2. 新しく現れた要素ほど指示対象として指示されやすい(recency).

3. また，主題化，関係節化，'scrambling'に関する統語的制約，優先性[57,11,10]として次のものを考慮する．これらは，GB(Government-Binding)理論[105]におけるバリアー(barrier)の概念に基づいている．

(a) 埋め込み文(embedded sentence)[10]，付加句(adjunct)[11]には主題は存在しない．

*太郎は上着を脱ぐと，花子は洋服掛けに掛けた．

下線部の付加句に主題（「太郎は」）が存在するため，この文は意味的に異常である[12]．

(b) 日本語においては，

太郎は学校へ行った．

学校へ太郎は行った．

のように，語順に自由性がある．この語順の自由性を説明する概念が'scrambling'である．すなわち，「学校へ太郎は行った.」は，「太郎は学校へ行った.」を'scramble'した結果得られると考える．この'scrambling'には，「文のバリアーを出て'scramble'することはできない」という制約がある．

*学校に，太郎は顔を洗い，靴をはき，ϕ 行った．

ここで，ϕの位置から「学校に」が'scramble'されて，前方へ移動することはない．

(c) 付加句中のギャップは，その文の主題または埋め込み文の主名詞と co-index しやすい．

太郎が ϕ 殴ったので，花子は泣いた．

この文の下線部の付加句中のギャップϕは，「花子」であるという解釈をしやすい．

(d) 埋め込み文中のギャップは，主文の主題と co-index しやすい．

(e) 主文中のギャップは前文までの主題と co-index されやすい．

また，比較的浅いレベルで処理できる情報として，文間の並列性に関する情報[66]も考慮し，

前に出現した同一格スロットの要素とギャップは co-index しやすい．

[9] 以後，ϕでギャップの箇所を明示することにする．
[10] 関係節化されて名詞句を修飾する文のことを埋め込み文と呼ぶことにする．
[11] 文中の述語を修飾する連用修飾句を付加句と呼ぶことにする．
[12] 非文を言語学上の記法に従って，'*'をつけて表現する．

という優先性を導入した.

焦点，談話構造に関する情報については，6.4.2 節の図 6.1に示したような，照応，省略との間に相互関係があるとの立場に立つことにする．すなわち，

- 照応，省略処理に用いられたことにより焦点となった対象は，再び照応，省略処理に用いられやすい．しかし，用いられなかった場合には，焦点は照応，省略処理に用いられた対象に移動する．

- 談話の木構造中現在のノードから上位方向に到達可能なノードに存在する対象しか，照応，省略処理に用いることはできない．現在のノードまでのそのパスが，談話の進行過程を表現しているからである．もし，それらの対象により照応，省略処理が失敗した場合には，現在のノードを表現するポインタの位置を変更しなければならない．

また，上の２点から明らかなように，

- 焦点となる対象は，現在のノードから到達可能な談話の進行過程上のノードに出現する対象でなければならず，現在のノードを表現するポインタの位置が変化することにより，焦点も移動する．また逆に，焦点が移動することにより，ポインタの位置も変更される．

本研究では，談話構造を構築する際に，領域に依存した情報を用いている．本研究では，例文として，１章で述べたように，化学の実験文を対象にしていることから，

- 実験過程における容器中の状態変化を談話の進行過程としてとらえ，談話構造中の１つのノードに１つの状態を対応させる．

- 談話構造の構築の手がかりとなる'clue word'として,「そして」などの接続詞の他に，数量詞「もう１つの」,「他の」などを用いる．複数の容器を同時に用いて，いくつかの実験を並列に行なうことがよくあるからである．

これらの情報を用いて，実際に談話構造を構築していく例は次節に示す．

焦点のモデルとしては，Sidner のもの[107]を用いる．しかし，Sidner のモデルは，本論文で扱うような状態変化が頻繁に起きる領域では問題がある．Sidner のモデルでは,「状態変化による対象の消滅，発生」を考慮していないからである．現在焦点である対象が化学反応した結果消滅し，他の対象が発生した場合，消滅した対象はもはや焦点ではありえず，新しく発生した対象が代わりに焦点となる傾向がある．Carter[67]も，これに関して，

> 動詞がある実体の変化を記述する場合，その変化した実体は' theme'[13]となる．

[13] 'theme'である要素が焦点に最もなりやすいと Carter は考えている．

と述べている．

このような考察により，Sidnerの焦点モデルの拡張を行なった．文によって記述される出来事（行為）から連想される対象のリスト

連想リスト(*AssociatedList*)

を新しく導入し，状態変化の結果発生した対象を記録する．また，現在焦点となっている対象が他の対象に変化すると推論される場合には，新しい対象を焦点の状態を表現するレジスタに記録する．また，状態変化の結果消滅した対象は，焦点の状態を表現するレジスタから削除する．この拡張により，上述した「状態変化による対象の消滅，発生」に対処できる．この連想リストは，Grosz[77]の研究における'implicit focus'や，長尾ら[34]のKM-スタック（仮説名詞スタック）と同等のものである．この拡張したモデルを用いた解析例は次節に示す．

本研究では，上述したような状態変化を記述するため，（領域にかなり依存した）常識的知識を推論規則として用いている．本研究で比較的浅いレベルの解析しか用いなかったのは，

> 膨大な量の常識的知識を用いた推論（常識的推論）では必ずフレーム問題[24]に直面する．

との考察に基づいている．本研究では，フレーム問題に対して，

> 変化した最小の部分しか記述せず，変化しなかった部分については，時間をさかのぼって推論する（記述量の解決）．

という方法を採っている．しかし，この方法は，フレーム問題に対する十分な解決策ではなく，知識の量が増加するとともに，推論時間が増大し，効率の低下を招くものと予想される．

6.5.2 照応，省略の処理手続き

本節では，6.5.1節で説明した情報を用いた，照応，省略に関する曖昧性解消アルゴリズムについて述べる．焦点の移動，談話構造と照応，省略の解釈の間の相互関係を例を用いて示す．統語的制約は，文中に出現した要素のうち，それらの制約を満足したもののみ焦点の候補となりうるという形でアルゴリズム中に取り入れられている．

このアルゴリズムは，

- 「それ」や，「その」を含む名詞句が出現した時点
- 4章で述べた必須スロットが満たされていないと判断された時点

で起動される.

以下に示す付録中の例文の解析過程を見ていくことにする.

1 試験管に塩素酸ナトリウムをとり，熱してとかす.

2 この試験管を熱し，発生する気体を水上置換で2本の集気びんに集める.

3 集気びんに集めた気体の中に，火のついた木炭を入れる.

4 次に，この集気びんに石灰水を入れ，ふり混ぜる.

5 もう1つの集気びんの気体の中に，火のついたイオウを入れる.

6 次に，この集気びんに色のついた草花を入れる.

7 また，水を入れてふり混ぜ，この水をリトマス紙につける.

各文を解析した結果得られる意味表現，その意味表現[14]と推論規則[15]により得られる推論結果[16]および焦点のレジスタの内容[17]を示す. また，談話構造の構築過程も図示する. 意味表現中'*'がついている対象は，照応，省略処理によって決定されたものである.

1.「試験管に塩素酸ナトリウムをとり,」

意味表現 : [取る，目標:試験管 1，対象:塩素酸ナトリウム 1]

推論結果 : [試験管 1，内容物:塩素酸ナトリウム 1] R1

DF : 塩素酸ナトリウム 1　　PDFL : [試験管 1]

DFS : []　　AL : []

焦点の候補の優先順位により，対象格の「塩素酸ナトリウム 1」が DF になる.

「熱してとかす.」

意味表現 : [溶かす，対象:塩素酸ナトリウム 1*,

道具:[熱する，対象:塩素酸ナトリウム 1*]]

推論結果 : [試験管 1，内容物:塩素酸ナトリウム 2]

[塩素酸ナトリウム 2，isa:液体] R2

[14] 対象名に付けられた数字は，同一名の対象が複数出現する際それらを区別するための識別子である.
[15] 付録 Dにこの例文を解析するのに用いた推論規則を示す.
[16] 推論結果に付けられた記号は，推論に用いた推論規則の記号（付録 D中）を表す.
[17] 対象とする例文中には動作主が現れないので，'actor focus'に関するレジスタは省略した.

DF ：塩素酸ナトリウム 2　　PDFL ：[試験管 1]

DFS ：[]　　AL ：[]

「熱する」,「とかす」の「を」格（対象格）が省略されているが，DF の「塩素酸ナトリウム 1」により補完される．また，推論の結果，試験管中の塩素酸ナトリウムは液体に変化（塩素酸ナトリウム 2）するため，DF もその新しい対象に変化する．

2.「この試験管を熱し,」

意味表現 ：[熱する，対象:試験管 1*]

推論結果 ：[熱する，対象:塩素酸ナトリウム 2] R3

[発生する，対象:酸素 1] R4

DF ： 試験管 1　　PDFL ：[]

DFS ：[塩素酸ナトリウム 2]　　AL ：[]

「発生する気体を水上置換で２本の集気びんに集める.」

意味表現 ：[集める，対象:[気体 1，対象![発生する]]，道具:水上置換 1，

目標:[集気びん 1，数量:2]]

推論結果 ：[集気びん 1，内容物:気体 1] R1

[eq，酸素 1，気体 1]

[気体 1，可溶性:0] R5

DF ： 試験管 1　　PDFL ：[気体 1,集気びん 1,水上置換]

DFS ：[塩素酸ナトリウム 2]　　AL ：[]

3.「集気びんに集めた気体の中に，火のついた木炭を入れる.」

意味表現 ：[入れる，目標:中([気体 1*，対象![集める，目標:集気びん 2]])，

対象:[木炭 1，目標![つく，対象:火 1]]]

推論結果 ：[集気びん 2，内容物:気体 1] R1

[入れる，目標:集気びん 2，

対象:[木炭 1，目標![つく，対象:火 1]]] R6

[集気びん 2，内容物:(気体 1,木炭 1)] R1

[燃える，対象:木炭 1] R7

[発生する，対象:二酸化炭素 1] R8

[集気びん 2，内容物:二酸化炭素 1] R1

DF ： 二酸化炭素 1　　PDFL ： [集気びん 2]

DFS ： [気体 1,試験管 1,塩素酸ナトリウム 2]

AL ： []

埋め込み文を含んだ名詞句（「集気びんに集めた気体」による照応で,「気体 1」が焦点になる．しかし，その後の状態変化により,「気体 1」,「木炭 1」などは消滅し，代わりに発生した「二酸化炭素 1」が焦点となる．

4.「次に，この集気びんに石灰水を入れ,」

意味表現 ： [入れる，目標:集気びん 2*，対象:石灰水 1]

推論結果 ： [集気びん 2，内容物:(二酸化炭素 1,石灰水 1)] R1

DF ： 集気びん 2　　PDFL ： [石灰水 1]

DFS ： [二酸化炭素 1,気体 1,試験管 1,塩素酸ナトリウム 2]

AL ： []

「ふり混ぜる.」

意味表現 ： [ふり混ぜる，対象:(二酸化炭素 1,石灰水 1)*]

推論結果 ： [集気びん 2，内容物:[液体 1，混合物:(炭酸カルシウム 1,水 1)，
色:白]] R9

DF ： 集気びん 2　　PDFL ： []

DFS ： [二酸化炭素 1,気体 1,試験管 1,塩素酸ナトリウム 2]

AL ： [液体 1]

この文が解析された時点までで構築された談話構造を図 6.2に示す．最初の文からこの文までは順接的につながれてきたので，談話構造も直線的である．

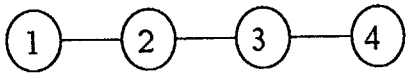


図 6.2: 文４までの談話構造

5.「もう１つの集気びんの気体の中に，火のついたイオウを入れる.」

意味表現 ： [入れる，目標:中([気体 1*，容器:集気びん 3])，
対象:[イオウ 1，目標![つく，対象:火 2]]]

推論結果 ： [集気びん 3，内容物:気体 1] R1
[入れる，目標:集気びん 3，
対象:[イオウ 1，目標![つく，対象:火 2]]] R6
[集気びん 3，内容物:(気体 1,イオウ 1)] R1
[燃える，対象:イオウ 1] R7
[発生する，対象:二酸化イオウ 1] R10
[集気びん 3，内容物:二酸化イオウ 1] R1

DF ： 二酸化イオウ 1　　PDFL ： [集気びん 3]
DFS ： [気体 1,試験管 1,塩素酸ナトリウム 2]
AL ： []

「もう１つの」という談話構造の構築に関する'clue word'が現れたので，談話構造をさかのぼり，「複数の集気びん」が出現したノードを探索する．そして，ノード２を発見する．このノードから今度は下向きにその「複数の集気びん」のうちの「１本の集気びん」に言及するノードを探し，ノード２を見つける．その結果ノード２から分岐する談話構造を構築する（図 6.3参照）．談話の進行過程が１，２，３，４となっていたのが，１，２，５と変わったので，それに伴い焦点のレジスタの内容を変更する．文５を解析する前のレジスタの内容として，文４の解析後のレジスタではなく，ノード２の時点でのレジスタの内容を用いる．そして，その中の PDFLの「気体１」が新しく焦点となる．その後，状態変化により，「二酸化イオウ１」に焦点が移動する．

6.「次に，この集気びんに色のついた草花を入れる.」

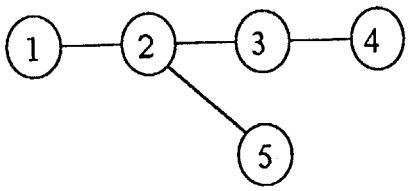


図 6.3: 文５までの談話構造

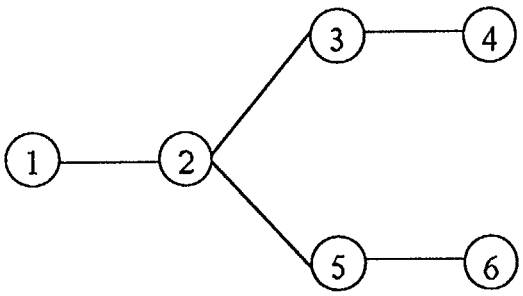


図 6.4: 文６の時点の談話構造

意味表現 ： [入れる，目標:集気びん 3*，
対象:[草花 1，目標![つく，対象:色 1]]]

推論結果 ： [入れる，目標:集気びん 3，
対象:[草花 1，色:(X \== 無)]] R11
[集気びん 3，内容物:(二酸化イオウ 1,草花 1)] R1
[集気びん 3，内容物:[草花 2，色:無]] R12

DF ： 集気びん 3　　PDFL ： [草花 2]

DFS ： [二酸化イオウ 1,気体 1,試験管 1,塩素酸ナトリウム 2]

AL ： []

PDFL 中の「草花」は，状態変化により，「草花 2」に変わっている．この時点で談話構造は図 6.4のようになる．

7.「また，水を入れて」

意味表現 ： [入れる，目標:集気びん 3*，対象:水 2]

推論結果 ： [集気びん 3，内容物:(二酸化イオウ 1,水 2)] R1

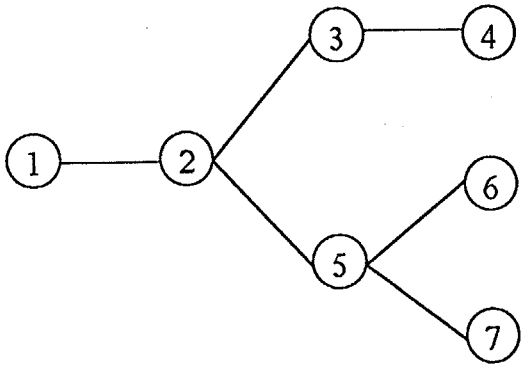


図 6.5: 文７の時点の談話構造

DF ： 集気びん 3　　PDFL ： [水 2]

DFS ： [二酸化イオウ 1,気体 1,試験管 1,塩素酸ナトリウム 2]

AL ： []

'clue word'「また」により，文７のノードは，談話構造上ノード６と並列される (図 6.5). このため，文７の解析の際には，文５の時点のレジスタの内容を用いる.

「ふり混ぜ,」

意味表現 ： [ふり混ぜる，対象:(二酸化イオウ 1,水 2)*]

推論結果 ： [集気びん 3，内容物:[益体 2，混合物:(二酸化イオウ 1,水 2)，
性質:酸性]] R13

DF ： 集気びん 3　　PDFL ： []

DFS ： [二酸化イオウ 1,気体 1,試験管 1,塩素酸ナトリウム 2]

AL ： [液体 2]

「この水をリトマス紙につける.」

意味表現 ： [つける，対象:液体 2*，リトマス紙 1]

推論結果 ： [リトマス紙 1，色:赤] R14

DF ： 液体 2　　PDFL ： [リトマス紙 1]

DFS ： [集気びん 3,二酸化イオウ 1,気体 1,試験管 1,塩素酸ナトリウム 2]

AL : []

「この水」の指示対象は，前に現れた「水 2」の状態変化後の「液体 2」に AL リストから決定される．また，「液体につける以前の」「リトマス紙 1」に対する前提条件(presupposition)として，

[リトマス紙1,色 : 青]

であることも推論により得られる．

第 7 章

結論と今後の研究課題

本論文の結論および今後の研究課題について述べる.

7.1 結論

本論文では, 自然言語理解の計算モデルとして, 形態素, 構文, 意味, 文脈解析の 4 つの解析ステップを融合して実行し, 文を解析する過程で得られる情報（制約）を増進的に蓄積し, 増進的に曖昧性を解消していく計算モデルを提案した.

増進的曖昧性解消モデルは制約プログラミングの枠組みで捉えることができる. なぜなら, 増進的曖昧性解消モデルの次の特徴,

- 左から右への後戻りのない解析
- 得られた制約による増進的曖昧性解消

は, 制約プログラミングの特徴とよく整合するからである. 従って, 本論文で提案する計算モデルは, 論理型プログラム言語 Prolog の拡張として, 制約プログラミングを実現したものと考えることができる.

本モデルは, 曖昧性解消過程で新しく得られた制約の蓄積過程を計算機上に表現することで, 曖昧性を増進的に解消するメカニズムを実現している. 制約を増進的に蓄積しておくことの利点は次のようなものである.

1. 意味解析で用いる選択制限は絶対的なものではなく, 選択制限に違反した場合には意味解析結果の優先順位が相対的に低下するとして考えなければならない. ただし, 優先順位がある しきい値を超えた意味解析結果しか妥当なものでないとして, その意味解析結果を排除しないと, 曖昧性の組合せ的爆発が生じ, 増進的曖昧性解消を効率よく実現することができない. しかし, 優先度が満たない意味解析結果を排除した結果, 解析に失敗した場合には, 過去に遡るとどこかに排除される以前の状態が蓄積されており, それを再び考慮することにより,

一般には，選択制限に違反し排除された意味解析結果を復活し，選択制限とは別の基準の意味解析を行い，最終的に意味解析を成功させることが可能である．

2. 2.3.2 節で述べたように，制約が蓄積するとともに，最初は意味が不明な未知語の意味や，指示語（「それ」など）の指示する対象指示対象を特定することができる．

また，増進的曖昧性解消に不可欠である，左から右へ情報を伝播する機構 BUP-UTI を提案した．

BUP-UTI を用いることにより，ボトムアップ解析システム上で，解析途中に上下双方向へダイナミックに情報を流すことが可能になった．それにより，従来のボトムアップ解析システムに較べ，早期に非文法的な箇所や意味的な異常さを検出できるため，効率の良い自然言語解析が可能になることを示した．

3.2.3 節で述べた日本語解析システムは，文献[93]の早期意味解析法にそったものであり，3.2.1 節で述べた人間の文理解過程に近いと考えられる．

4 章では，増進的曖昧性解消の基礎となる意味解析用の辞書（フレーム）の記述言語として SRL を提案した．また，これまでの研究におけるフレーム形式を用いた辞書記述に関する問題点を具体的に指摘した．

5 章では，4 章の辞書エントリ中に記述された選択制限（制約）の検査を行なう時に必要な，概念階層上での推論手続きについて述べ，この推論を効率的に行なう新しい手法として，従来のように，記述した知識をそのまま推論に用いるのではなく，知識表現体系を，知識の記述，記述した知識のコンパイル，そしてそれを用いた推論の 3 つの段階として考え，記述した継承ネットワークを条件付きリンクや冗長なリンクを含まない等価な継承ネットワークに変換して推論を行う方法を提案した．

7.2 今後の課題

今後の課題としては，以下のようなものが挙げられる．

7.2.1 増進的曖昧性解消モデルに関する今後の課題

1. 選択制限は曖昧性解消に利用できる重要な情報である．2.3.4 節では，その有効性について述べた．しかし，その選択制限の利用には，以下に示す比喩文，否定文，疑問文，異常事態を記述する文を解析する際に問題がある：

   比喩文「アイデアが開花する．」

   否定文「ねずみは空を飛ばない．」

疑問文「あひるは本を読みますか？」

異常事態を記述する文「赤ちゃんがビー玉を飲む.」

上述した文中の下線部の句が動詞の持つ選択制限に違反するにもかかわらず，意味的には妥当であるからである．この問題は次のようにして解決できる：

> 選択制限の記述に優先性(preference)を導入し，曖昧性解消時に選択制限に違反する文を排除するのではなく，その優先度を下げるにとどめ，排除しない．

2. 分かち書きの曖昧性に関しては，現在はバックトラックで可能なすべての分かち書きの候補を１つずつ生成する手法を用いている．従って，形態素解析の中途で複数の曖昧性が得られず，増進的曖昧性解消が実現できない．一方，文を左から右に読み進む過程で分かち書きを並列に実行し，得られた複数個の結果（曖昧性）を圧縮した形式で得るもの[27]がある．この手法では，文を最後まで解析し，すべての分かち書きの候補を得てから次の解析ステップに進むことになるので，本論文で提案する「左から右へ文を読み進む過程での増進的曖昧性解消モデル」とは相入れない．

3. 限量子や否定のスコープの曖昧性解消は本論文では扱っていない．限量子に関するスコープの曖昧性解消に関しては，最近研究が行なわれており[83,93]参考になると思われる．

4. 2.3.4 節で述べたように，指示対象に対する制約は，その複合不定項のスロットに単に蓄積しているだけである．それらの制約は，他の制約と矛盾する可能性がある．従って，スロットのユニフィケーションには，無矛盾性検査の機能を付加する必要がある．

5. 本論文では，１文を越えての曖昧性解消としては，照応参照処理，省略補完処理などについて述べたが，文以外の制約，例えば，常識を用いた曖昧性解消については今後の課題である．

6. 曖昧性のある文を聞いても，通常人間は，可能性のある解釈すべてを保持しているわけではなく，ただ１つの解釈を選択しているといわれている．また，希な場合曖昧性を認識したとしても，ある１つの解釈に他の解釈と較べて相対的な優先性を感じ，その優先性に基づいて１つの解釈を行い，後でそれがなんらかの矛盾を引き起こすようになったら，別の解釈を考え直すという手法を用いているように思われる[1][97]．このような人間の認知モデルに関する心理学的考察を反映したモデルを実現していく必要がある．

[1] これと関連する現象として袋小路文(garden path sentence)がある．

7.2.2 構文解析メカニズム BUP-UTI に関する今後の課題

3.2.3, 3.2.4 節の例からわかるとおり，BUP-UTI における文法記述は XGS を用いていることもあり，やや書きにくい．さらに高水準な文法記述言語を XGS の上に設計する必要がある．

7.2.3 辞書記述言語 SRL に関する今後の課題

1. 4.2 節で述べた選択制限に違反する文（比喩）も，特別な文脈が与えられれば，正常な文として解釈できることが多い．現在の辞書記述の枠組みでは，このような状況には対処できず，選択制限に違反する文はすべて意味的に異常な文として排除してしまう．今後は，比喩をも考慮に入れた辞書記述を検討していかなければならない．

2. SRL を含めて，辞書の開発環境を整備していく必要があり，その中でも特に，辞書項目の修正，追加に伴う辞書の保守・管理を行うプログラムを開発する必要がある[36,9]．

7.2.4 概念階層上での推論に関する今後の課題

1. 5.2.3 節で述べた継承ネットワークのコンパイル・アルゴリズムの定式化，妥当性の検討，及びインプリメントを行う．

2. 5.2.4 節で述べた無矛盾な拡張集合の計算手続きの定式化を行う．

3. 5.2.4 節で述べた継承アルゴリズムを並列に実行するプログラムを開発する．現在，並列論理型言語 GHC[14]によるインプリメントを検討中である．

4. 継承ネットワークのコンパイル結果に，時間変化などを含む新たな情報を追加する問題を解決しなければならない．これを動的な継承ネットワークの問題と呼ぶ[54]．5.2.4 節で述べたように，'/'を用いたパスの履歴を条件とみなせば，本論文で提案した継承アルゴリズムで動的な継承ネットワークを扱えると思われる．

7.2.5 照応，省略の曖昧性解消に関する今後の課題

- 「発話の場」（状況）を考慮した照応，省略処理
- 文（出来事）を指示する照応

以上の問題に関連して，[65,104]の研究が参考になるかもしれない．

- 6.2 節で述べた情報を用いた照応，省略処理の実現

- 領域に依存しない談話構造の構築

以上を実現するためには，6.5.1 節で述べたように，膨大な量の常識的知識を用いた推論が必要になる．しかし，膨大な量の知識を計算機上で扱う研究は現在活発化しているが，未解決の問題が山積している．今後は，領域に依存しない常識的知識をどのように計算機上で扱うかを研究していく必要がある．

謝辞

本研究を行なうにあたり，終始変わらず暖かい御助言，御指導を頂きました田中穂積教授に深く感謝致します．田中先生には，自然言語処理について一から御教授頂いただけでなく，科学研究を進める上で何が重要であるかを教えて頂きました．

ICOT 自然言語理解ワーキンググループの方々，京都大学の松本裕治助教授，京都大学長尾研究室の佐藤理史助手，東芝総合研究所の木下聡氏，横浜国立大学の田村直良講師，コーネル大学の柳田優子氏からは，この研究に対して大変有益なコメントを頂きました．感謝致します．

田中研究室のメンバーの方々には，長いようで短かかった学生生活を楽しく過ごさせて頂きました．メンバーの方々との討論はこの研究を進める上で大変有益でした．また，ソフトウェアの使い方がよく分からず困っている時にも親切に教えて頂きました．特に，徳永健伸助手には大変お世話になりました．感謝致します．

付録 A

本論文で対象とした例文

例文 1

試験管に塩素酸ナトリウムをとり，熱してとかす．
この試験管を熱し，発生する気体を水上置換で２本の集気びんに集める．
集気びんに集めた気体の中に，火のついた木炭を入れる．
次に，この集気びんに石灰水を入れ，ふり混ぜる．
もう１つの集気びんの気体の中に，火のついたイオウを入れる．
次に，この集気びんに色のついた草花を入れる．
また，水を入れてふり混ぜ，この水をリトマス紙につける．

例文２

三角フラスコにアルミニウムを入れる．

うすい塩酸を注ぐ．

３本の集気びんに発生する気体を水上置換で集め，ガラス板でふたをしたまま，

机の上にさかさに立てる．

１本目の集気びんに石灰水を入れて，ふり混ぜる．

２本目の集気びんに，草花を入れる．また，においも調べる．

３本目の集気びんを，口を下にして持ち，火のついた線香を入れる．

例文３

アンモニア水を試験管に入れ，熱してできる気体を，かわいた３本の試験管に集め，ゴムせんをする．

気体を集めた試験管の口に，水で湿したリトマス紙をかざす．

また，塩酸をつけたガラス棒の先を近づける．

気体を集めたもう１つの試験管を，ビーカー中の水にさかさに入れる．

３本目の試験管に，燃えているマッチを入れる．

付録 B

本研究で用いた文法

```
文(文_A,文_S,文_T) -->
    後置詞句(_,_,_),
  ( [] ; 読点(_,_,_) ),
    文(文_A,文_S,文_T).

文(文_A,文_S,文_T) -->
    従属節(_,_,_),
  ( [] ; 読点(_,_,_) ),
    文(文_A,文_S,文_T).

文(文_A,文_S,文_T) -->
    接続詞(_,_,_),
  ( [] ; 読点(_,_,_) ),
    文(文_A,文_S,文_T).

文(文_A,文_S,文_T)[X0,X1] ==>
    副詞句(_,副詞句_S,副詞句_T),
  ( [] ; 読点(_,_,_) ),
    文(文_A,文_S,文_T)[[(v,[mod:副詞句_S,副詞句_T])|X0],X1].

文(文_A,文_S,文_T)[X0,X2] ==>
    名詞句(名詞句_A,名詞句_S,名詞句_T),
  { check(subcat(taigenDome),名詞句_A) },
```

```
 { interp((n,v),[[],[]]:[名詞句_S,名詞句_T],X0,_,X1) },
 ( [] ; 読点(_,_,_) ),
   文(文_A,文_S,文_T)[X1,X2].

文(文_A,文_S,文_T)[X0,X2] ==>
   述語(述語_A,述語_S,述語_T),
 { rentai(述語_A) },
 { interp((v,n),[述語_S,述語_T],X0,_,X1) },
 ( [] ; 読点(_,_,_) ),
   文(文_A,文_S,文_T)[X1,X2].

文(文_A,文_S,文_T) -->
   連体句(_,_,_),
 ( [] ; 読点(_,_,_) ),
   文(文_A,文_S,文_T).

文(文_A,文_S,文_T)[X0,X1] ==>
   主語句(_,主語句_S,主語句_T),
 ( [] ; 読点(_,_,_) ),
   文(文_A,文_S,文_T)[[(v,[主語句_S,主語句_T])|X0],X1].

文(文_A,文_S,文_T)[X0,X1] ==>
   連体詞(_,連体詞_S,連体詞_T),
   文(文_A,文_S,文_T)[[(n,[連体詞_S,連体詞_T])|X0],X1].

文(文_A,文_S,文_T)[X0,X1] ==>
   連体詞句(_,連体詞句_S,連体詞句_T),
   文(文_A,文_S,文_T)[[(n,[連体詞句_S,連体詞句_T])|X0],X1].

文(述語_A,宣言文(文_S),文_T)[X0,X1] ==>
   述語(述語_A,述語_S,述語_T),
 { shuushi(述語_A) },
```

```
  句点(_,_,_),
{ interp((v,t),[述語_S,述語_T],X0,[文1_S,文_T],X1) },
{ postP(文1_S,文_S) }.

文(述語_A,文_S,文(文_T,終助詞_T))[X0,X1] ==>
  述語(述語_A,述語_S,述語_T),
  終助詞(終助詞_A,終助詞_S,終助詞_T),
{ shousetsu(述語_A,終助詞_A) },
( 句点(_,_,_) ; 疑問符(_,_,_) ),
{ interp((v,t),[述語_S,述語_T],X0,[文1_S,文_T],X1) },
{ postP(文1_S,文2_S) },
{ modalinterp(終助詞_S,文2_S,文_S) }.

文(_,疑問文(文_S),文(文_T,終助詞(か)))[X0,X2] ==>
  名詞句(_,名詞句_S,名詞句_T),
  [か],
( 句点(_,_,_) ; 疑問符(_,_,_) ),
{ interp((n,t),[名詞句_S,名詞句_T],X0,[名詞句1_S,名詞句1_T],X1) },
{ interp((v,t),[名詞句1_S,用言(名詞句1_T,助動詞(です))],X1,[文1_S,文_T],X2) },
{ postP(文1_S,文_S) }.

後置詞句(_,_,_)[X0,X1] ==>
  名詞句(名詞句_A,名詞句_S,名詞句_T),
{ \+ check(subcat(taigenDome),名詞句_A) },
{ \+ check(subcat(fukushiTeki),名詞句_A) },
  助詞(_,助詞_S,助詞_T),
{ interp((n,v),[助詞1_S,助詞_T]:[名詞句_S,名詞句_T],X0,_:[名詞句1_S,_],X1) },
{ ppinterp(名詞句1_S,助詞_S,助詞1_S) }.

後置詞句(_,_,_)[X0,X1] ==>
  述語(述語_A,述語_S,述語_T),
{ shuushi(述語_A) },
```

```
  [と],
{ interp((v,v),[と,格助詞(と)]:[述語_S,述語_T],X0,_,X1) }.

後置詞句(_,_,_)[X0,X1] ==>
   述語(述語_A,述語_S,述語_T),
{ renyou(述語_A) },
   [に],
{ interp((v,v),[に,格助詞(に)]:[述語_S,述語_T],X0,_,X1) }.

従属節(_,_,_)[X0,X1] ==>
   述語(述語_A,述語_S,述語_T),
   接続助詞(接続助詞_A,接続助詞_S,接続助詞_T),
{ shousetsu(述語_A,接続助詞_A) },
{ interp((v,v),[接続助詞_S,接続助詞_T] wrt [述語_S,述語_T],X0,_,X1) }.

従属節(_,_,_)[X0,X1] ==>
   述語(述語_A,述語_S,述語_T),
{ renyou(述語_A) },
{ interp((v,v),[relation((coordinate;temporal)),[]] wrt [述語_S,述語_T],X0,_,X1) }.

副詞句(_,副詞_S,副詞句(副詞_T)) ==>
   副詞(_,副詞_S,副詞_T).

副詞句(_,名詞_S,副詞句(名詞_T)) ==>
   名詞(名詞_A,名詞_S,名詞_T),
{ check(subcat(fukushiTeki),名詞_A) }.

副詞句(_,形容詞_S,副詞句(形容詞_T)) ==>
   形容詞(形容詞_A,形容詞_S,形容詞_T),
{ renyou(形容詞_A) }.

副詞句(_,形容動詞_S,副詞句(形容動詞_T)) ==>
```

```
    形容動詞(形容動詞_A,形容動詞_S,形容動詞_T),
  { renyou(形容動詞_A) }.

述語(動詞_A,述語_S,述語(動詞_T)) ==>
    動詞(動詞_A,動詞_S,動詞_T),
  { henkou(動詞_S,[],述語_S) }.

述語(助動詞列_A,述語_S,述語_T)[X0,X1] ==>
    動詞(動詞_A,動詞_S,動詞_T),
    助動詞列(助動詞列_A,助動詞列_S,助動詞列_T)[[動詞_A|X0],X1],
  { henkou(動詞_S,助動詞列_S,述語_S) },
  { 述語_T=..[述語,動詞_T|助動詞列_T] }.

述語(形容詞_A,述語_S,述語(形容詞_T)) ==>
    形容詞(形容詞_A,形容詞_S,形容詞_T),
  { henkou(形容詞_S,[],述語_S) }.

述語(助動詞列_A,述語_S,述語_T)[X0,X1] ==>
    形容詞(形容詞_A,形容詞_S,形容詞_T),
    助動詞列(助動詞列_A,助動詞列_S,助動詞列_T)[[形容詞_A|X0],X1],
  { henkou(形容詞_S,助動詞列_S,述語_S) },
  { 述語_T=..[述語,形容詞 _T|助動詞列_T] }.

述語(形容動詞_A,述語_S,述語(形容動詞_T)) ==>
    形容動詞(形容動詞_A,形容動詞_S,形容動詞_T),
  { henkou(形容動詞_S,[],述語_S) }.

述語(助動詞列_A,述語_S,述語_T)[X0,X1] ==>
    形容動詞(形容動詞_A,形容動詞_S,形容動詞_T),
    助動詞列(助動詞列_A,助動詞列_S,助動詞列_T)[[形容動詞_A|X0],X1],
  { henkou(形容動詞_S,助動詞列_S,述語_S) },
  { 述語_T=..[述語,形容動詞_T|助動詞列_T] }.
```

```
述語(用言_A,述語_S,述語(用言_T)) ==>
    用言(用言_A,用言_S,用言_T),
 { henkou(用言_S,[],述語_S) }.

名詞句(名詞_A,名詞句_S,名詞句(名詞_T)) ==>
    名詞(名詞_A,名詞_S,名詞_T),
 { startn(名詞_S,名詞句_S) }.

名詞句(_,名詞句_S,名詞句(名詞句 1_T,並列詞_T,名詞句 2_T)) ==>
    名詞句(_,名詞句 1_S,名詞句 1_T),
    並列詞(_,並列詞_S,並列詞_T),
    名詞句(_,名詞句 2_S,名詞句 2_T),
 { cnpinterp(並列詞_S,名詞句 1_S,名詞句 2_S,名詞句_S) }.

連体句(_,_,_)[X0,X1] ==>
    名詞句(_,名詞句_S,名詞句_T),
    [の],
 { interp((n,n),[の,格助詞(の)]:[名詞句_S,名詞句_T],X0,_,X1) }.

連体句(_,_,_)[X0,X1] ==>
    名詞句(_,名詞句_S,名詞句_T),
    格助詞(格助詞_A,case(格助詞_S),格助詞_T),
 { shousetsu(格助詞_A,(格助詞,の)) },
    [の],
 { interp((n,n),[格助詞_S,[格助詞_T,格助詞(の)]]:[名詞句_S,名詞句_T],X0,_,X1) }.

連体句(_,_,_)[X0,X1] ==>
    名詞句(_,名詞句_S,名詞句_T),
    副助詞(副助詞_A,副助詞_S,副助詞_T),
 { shousetsu(副助詞_A,(格助詞,の)) },
    [の],
```

```
{ interp((n,n),[の,[副助詞_T,格助詞(の)]]:[名詞句_S,名詞句_T],X0,_:[名詞句1_S,_],X1) },
{ ppinterp(名詞句1_S,[副助詞_S],_) }.

連体句(_,_,_)[X0,X1] ==>
    名詞句(_,名詞句_S,名詞句_T),
    連体表現(_,連体_S,連体_T),
{ interp((n,n),[連体_S,連体_T]:[名詞句_S,名詞句_T],X0,_,X1) }.

主語句(_,が:名詞句_S,主語句(名詞句_T,格助詞(の))) ==>
    名詞句(_,名詞句_S,名詞句_T),
    [の].

連体詞句(助動詞_A,連体詞句_S,連体詞句(連体詞_T,助動詞_T)) ==>
    連体詞(_,連体詞_S,連体詞_T),
    助動詞(助動詞_A,_,助動詞_T),
 { check((助動詞,ようだ),助動詞_A) },
 { detinterp(連体詞_S,連体詞句_S) }.

助詞(_,[格助詞_S],助詞(格助詞_T)) ==>
    格助詞(_,格助詞_S,格助詞_T).

助詞(_,[副助詞_S],助詞(副助詞_T)) ==>
    副助詞(_,副助詞_S,副助詞_T).

助詞(_,[格助詞_S,副助詞_S],助詞(格助詞_T,副助詞_T)) ==>
    格助詞(格助詞_A,格助詞_S,格助詞_T),
    副助詞(副助詞_A,副助詞_S,副助詞_T),
 { shousetsu(格助詞_A,副助詞_A) }.

助詞(_,[副助詞1_S,副助詞2_S],助詞(副助詞1_T,副助詞2_T)) ==>
    副助詞(副助詞1_A,副助詞1_S,副助詞1_T),
    副助詞(副助詞2_A,副助詞2_S,副助詞2_T),
```

```
{ shousetsu(副助詞 1_A,副助詞 2_A) }.

助詞(_,[副助詞_S,格助詞_S],助詞(副助詞_T,格助詞_T)) ==>
    副助詞(副助詞_A,副助詞_S,副助詞_T),
    格助詞(格助詞_A,格助詞_S,格助詞_T),
 { shousetsu(副助詞_A,格助詞_A) }.

動詞(動詞 1_A,動詞_S,動詞(名詞_T,動詞 1_T)) ==>
    名詞(_,名詞_S,名詞_T),
 { sinterp(名詞_S,動詞_S) },
    動詞(動詞 1_A,_,動詞 1_T),
 { check((動詞,する),動詞 1_A) }.

動詞(補助動詞_A,動詞_S,動詞(動詞 1_T,補助動詞_T)) ==>
    動詞(動詞 1_A,動詞 1_S,動詞 1_T),
    補助動詞 1(補助動詞_A,補助動詞_S,補助動詞_T),
 { hinterp1(補助動詞_S,動詞 1_S,動詞_S) }.

動詞(補助動詞_A,動詞_S,動詞(動詞 1_T,接続助詞(て),補助動詞_T)) ==>
    動詞(動詞 1_A,動詞 1_S,動詞 1_T),
  (
  { shousetsu(動詞 1_A,(接続助詞,て)) },
    [て] ;
  { shousetsu(動詞 1_A,(接続助詞,て)&onbinKei) },
    [で]
  ),
    補助動詞 2(補助動詞_A,補助動詞_S,補助動詞_T),
  { hinterp2(補助動詞_S,動詞 1_S,動詞_S) }.

用言(助動詞_A,名詞句 1_S,用言(名詞句 1_T,助動詞_T))[X0,X1] ==>
    名詞句(_,名詞句_S,名詞句_T),
    助動詞(助動詞_A,_,助動詞_T),
```

```
( { check((助動詞,だ),助動詞_A) } ;
  { check((助動詞,です),助動詞_A) } ),
{ interp((n,t),[名詞句_S,名詞句_T],X0,[名詞句 1_S,名詞句 1_T],X1) }.

助動詞列(助動詞_A,[助動詞_S],[助動詞_T])[[用言_A|X0],X0] ==>
   助動詞(助動詞_A,助動詞_S,助動詞_T),
{ shousetsu(用言_A,助動詞_A) }.

助動詞列(助動詞列 1_A,[助動詞_S|助動詞列 1_S],
                [助動詞_T|助動詞列 1_T])[[用言_A|X0],X1] ==>
   助動詞(助動詞_A,助動詞_S,助動詞_T),
{ shousetsu(用言_A,助動詞_A) },
   助動詞列(助動詞列 1_A,助動詞列 1_S,助動詞列 1_T)[[助動詞_A|X0],X1].
```

付録 C

意味解析用述語 'interp'の定義

```
interp(Case:N,[[Rn::Time#(V,Slots,Pol)|D]|Rel],
              [[Rn::Time#VNew|NewD]|Rel]):-
   takeDaemon(Daemon,D),
 ( var(Daemon) -> sem(V,Rn,Case:N~[(V,Slots,Pol)]~[VNew|NewD]) ;
                  ( sem(V,Rn,Case:N~[(V,Slots,Pol)]~VTemp),
                    execDaemon(Daemon,VTemp,[VNew|NewD]) ) ).

takeDaemon(_,[]).
takeDaemon(daemon(X)&H,[daemon(X)|T]) :-
        takeDaemon(H,T).

execDaemon(V,A,A) :-
   var(V),!.
execDaemon(daemon(X)&T,A,C) :-
   wakeDaemon(daemon(X),A,B),
   execDaemon(T,B,C).
wakeDaemon(daemon(X^G),[Sem|D],VNew) :-
   Sem = (Con,Slots,Pol),
 ( memberSlot(X,Slots) ->
    ( G = sem(V,N,Case:Value~[Sem|D]~VNew),
      call(G) ) ;
   addSS1(daemon(X^G),[Sem|D],VNew) ).
```

付録 D

例文1を解析するのに用いた常識的知識

R1 ある容器にものを「取る」,「集める」,「入れる」といった行為を行なうと，その容器の中にそのものが存在することになる．また，容器中でものが「発生する」と，その容器の中にはそのものが存在することになる．

R2 ものを「溶かす」とそのものの状態は液体に変化する．

R3 容器を熱すると，その容器の中の内容物も熱せられる．

R4 塩素酸ナトリウムを熱すると，酸素が発生する．

R5 水上置換で集めることができるのは，水に溶けない気体に限られる．

R6 酸素中に火のついたものがあると，そのものは燃える．

R7 木炭が燃えると，二酸化炭素が発生する．

R8 二酸化炭素と石灰水を混ぜ合わせると，化学反応し，炭酸カルシウムと水ができる．

R9 イオウが燃えると，二酸化イオウが発生する．

R10 二酸化イオウには脱色作用がある．

R11 二酸化イオウと水を混ぜ合わせると，酸性の液体（亜硫酸）ができる．

R12 青色リトマス紙に酸性の液体をつけると，色が赤に変わる．

参考文献

[1] 井佐原均, 橋田浩一, 石崎俊, 内田ユリ子. 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報告, 64:53–59, 1987.

[2] 井佐原均, 田中穂積. 日本語埋め込み文の構文解析における諸問題. 情報処理学会自然言語処理研究会報告, 26, 1981.

[3] 奥津敬一郎. 「ボクハウナギダ」の文法 – ダとノ –. くろしお出版, 1978.

[4] 荻野孝野. 日本語の意味分類体系. 計量国語学, 16(3):95–112, 1987.

[5] 丸山宏, 渡辺日出雄. 制約伝播アルゴリズムを用いた対話的日本語解析システム. , 日本ソフトウェア科学会第 5 回大会論文集, 17–20 ページ, 1988.

[6] 橋田浩一, 白井英俊. 条件付単一化. コンピュータソフトウェア, 3(4):28–38, 1986.

[7] 玉木久夫. 論理型言語におけるプログラム変換. , 古川康一, 溝口文雄, 編集者, プログラム変換, 39–62 ページ, 共立出版, 1987.

[8] 吉田将, 首藤公昭, 藤田毅. 日本語の機械処理－日本語文における曖昧さと関係表現について. , 電気四学会連合大会, 79–82 ページ, 1978.

[9] 吉田将. 辞書構築における諸問題. 情報処理, 27(8):933–939, 1986.

[10] 吉本啓. 談話処理における日本語ゼロ代名詞の扱いについて. 情報処理学会自然言語処理研究会報告, 56, 1986.

[11] 吉本啓. 日本語の談話における主題の扱い. , 情報処理学会第 35 回全国大会講演論文集, 1407–1408 ページ, 1987.

[12] 久野すすむ. 談話の文法. 大修館書店, 1978.

[13] 久野すすむ. 日本文法研究. 大修館書店, 1973.

[14] 古川康一, 溝口文雄, 編集者. 並列論理型言語 *GHC* とその応用. 知識情報処理シリーズ, 共立出版, 1987.

[15] 国立国語研究所. 現代語の助詞・助動詞 -用法と実例 -. 国立国語研究所報告第 3 巻, 秀英出版, 1951.

[16] 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, 227-230 ページ, 1984.

[17] 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システム. コンピュータソフトウエア, 3(2):19-29, 1986.

[18] 坂間千秋, 奥村晃. 非単調並列継承ネットワーク. , *Proc. of the Logic Programming Conference'88*, 65-72 ページ, 1988.

[19] 糸井理人. 融合方式による英日機械翻訳システムにおける英語基本動詞の訳し分け. 修士論文, 東京工業大学, 1985.

[20] 柴山悦哉. 論理型言語におけるユニフィケーションの拡張とその応用. 情報処理学会ソフトウェア基礎論研究会報告, 10, 1984.

[21] 三上章. 象は鼻が長い. くろしお出版, 1960.

[22] 三藤博. ディスコース表示理論について. , 自然言語処理シンポジウム論文集, 35-52 ページ, 1988.

[23] 山内智恵子, 丸山宏. 機械翻訳の誤りと人間の誤り - 日本語の構造的多義 -. , 日本語国際シンポジウム「日本語教育の現代的課題」, 193-202 ページ, 1988.

[24] 松原仁, 山本和彦. フレーム問題に関する考察. 情報処理学会知識工学と人工知能研究会報告, 50, 1987.

[25] 松本裕治, 清野正樹, 田中穂積. BUP の高速化. 情報処理学会自然言語処理研究会報告, 39, 1983.

[26] 上脇正, 田中穂積, 沼崎浩明. LangLAB の構文処理アルゴリズムの高速化. , 情報処理学会第 *33* 回全国大会講演論文集, 1441-1442 ページ, 1986.

[27] 杉村領一, 赤坂宏二, 久保幸弘, 松本裕治, 佐野洋. 論理型形態素解析 LAX. , *Proc. of the Logic Programming Confenrece'88*, 213-222 ページ, 1988.

[28] 正保勇.「コソア」の体系.，日本語の指示詞，51–122ページ，国立国語研究所，1981.

[29] 赤間清. 継承階層 PROLOG と多重継承.，日本ソフトウェア科学会第 3回大会論文集，189–192ページ，1986.

[30] 赤間清. 集合束縛変数に基づく意味表現とユニフィケーション. 情報処理学会自然言語処理研究会報告，62:53–60，1987.

[31] 川森雅仁. 談話表示理論. 1988. 論理文法研究会「モンタギュ文法とその後の発展：ディスコース表示理論と状況意味論」チュートリアル資料.

[32] 村木新次郎, 青山文啓, 六条範俊, 村田賢一. 辞書における格情報の記述. 情報処理学会自然言語処理研究会報告，46，1984.

[33] 中島秀之. 項記述.，*Proc. of the Logic Programming Confenrece'84*，1984.

[34] 長尾真, 辻井潤一, 田中一敏. 意味および文脈情報を用いた日本語文の解析 – 文脈を考慮した処理. 情報処理，17(1):19–28，1976.

[35] 長尾真, 辻井潤一, 田中一敏. 意味および文脈情報を用いた日本語文の解析 – 名詞句・単文の処理. 情報処理，17(1):10–18，1976.

[36] 辻井潤一. 辞書の構成と機械翻訳. 情報処理，26(10):1174–1183，1985.

[37] 辻井潤一. 文脈処理.，自然言語処理シンポジウム論文集，75–87ページ，1988.

[38] 池上嘉彦. 意味の世界 – 現代言語学から視る –. NHK ブックス第 330巻，日本放送出版協会，1978.

[39] 桃内佳雄. 文章における主題構造と省略. 計量国語学，15(7):267–285，1986.

[40] 藤崎博也, 星合忠. 言語表現の曖昧性.，田中幸吉，編集者，知識工学，172–187ページ，朝倉書店，1984.

[41] 徳永健伸, 田中穂積. 視点を考慮した概念の同一化.，情報処理学会第 37回全国大会論文集，1284–1285ページ，1988.

[42] 徳永健伸, 田中穂積. 知識継承における例外の扱いとその応用について.，情報処理学会第 35回全国大会論文集，1837–1838ページ，1987.

[43] 二村良彦. 部分計算. , 古川康一, 溝口文雄, 編集者, プログラム変換, 63-80 ページ, 共立出版, 1987.

[44] 二木厚吉. 等式プログラムの等価変換. , 古川康一, 溝口文雄, 編集者, プログラム変換, 17-38 ページ, 共立出版, 1987.

[45] 田村直良, 高倉伸, 片山卓也. 自然言語処理を目的とした属性文法評価システム. コンピュータソフトウエア, 3(3):266-279, 1986.

[46] 田中穂積, 小山晴生, 奥村学. ボトムアップ構文解析システム BUP の拡張と日本語文法の試作. , *Proc. of the Logic Programming Confenrece'84*, 1984.

[47] 田中穂積, 小山晴生, 奥村学. 知識表現形式 DCKR とその応用. コンピュータソフトウェア, 3(4):12-20, 1986.

[48] 田中穂積, 上脇正, 奥村学, 沼崎浩明. 自然言語処理のためのソフトウエアシステム LangLAB. , *Proc. of the Logic Programming Conference'86*, 5-12 ページ, 1986.

[49] 田中穂積. 計算機による自然言語の意味処理に関する研究. 研究報告 797, 電子技術総合研究所, 1979.

[50] 田中望.「コソア」をめぐる諸問題. , 日本語の指示詞, 1-50 ページ, 国立国語研究所, 1981.

[51] 土井伸一, 永野三郎. 談話における焦点構造とその解析. , 情報処理学会第 *35* 回全国大会講演論文集, 1413-1414 ページ, 1987.

[52] 風斗博之. 名詞句の指示と談話処理. 情報処理学会自然言語処理研究会報告, 55, 1986.

[53] 平井誠. 日本語文の構文および意味構造の解析手法に関する研究. 博士論文, 豊橋技術科学大学, 1987.

[54] 毛受哲, 伊藤英則, 森田幸伯. 制約付き性質継承に関する並列アルゴリズム. , 人工知能学会第 *2* 回全国大会論文集, 145-148 ページ, 1988.

[55] 木村和広. スタック操作に基づく文脈処理について. , 情報処理学会第 *35* 回全国大会講演論文集, 1411-1412 ページ, 1987.

[56] 木村和広. 状況意味論に基づく照応モデル. 1986. 日本ソフトウェア科学会「論理と自然言語」研究会ワークショップ資料.

[57] 野田尚史. 複文における「は」と「が」の係り方. 日本語学, 5(2):31–43, 1986.

[58] 望月泰行. シソーラス作成支援ツールに関する研究. 卒業論文, 東京工業大学, 1988.

[59] 北上始. 知識獲得システム. , 古川康一, 溝口文雄, 編集者, 知識の学習メカニズム, 71–124 ページ, 共立出版, 1986.

[60] 林四郎,荻野綱男,田中幸子,樺島忠夫. 運用 *I*, 1–45 ページ. 朝倉日本語新講座第 5 巻, 朝倉書店, 1983.

[61] 拡張 *LINGOL*. 電子技術総合研究所, 1978.

[62] 計算機用日本語基本動詞辞書 *IPAL(Basic Verbs)* – 辞書編 –. 情報処理振興事業協会技術センター, 1987.

[63] *CIL* 言語マニュアル. 新世代コンピュータ技術開発機構, 第 3 版, 1988.

[64] G. Adriaens and S.L. Small. Word expert parsing revisited in a cognitive science perspective. In S.L. Small, G.W. Cottrell, and M.K. Tanenhaus, editors, *Lexical Ambiguity Resolution : Perspectives from Psycholinguitics, Neuropsychology, and Artificial Intelligence*, pages 13–43, Morgan Kaufmann Publishers, 1988.

[65] J. Barwise and J. Perry. *Situations and Attitudes.* MIT Press, 1983.

[66] J.G. Carbonell and R.D. Brown. Anaphora resolution : a multi-strategy approach. In *Proc. of the 12th International Conference on Computational Linguistics*, pages 96–101, 1988.

[67] D. Carter. *Interpreting Anaphoras in Natural Language Texts.* Ellis Horwood, 1987.

[68] E. Charniak. *Toward a Model of Children's Story Comprehension.* Technical Report 266, MIT AI Laboratory, 1972.

[69] K. Church and R. Patil. Coping with syntactic ambiguity or how to put the block in the box on the table. *American Journal of Computational Linguistics*, 8(3-4):139–149, 1982.

[70] R. Cohen. Analyzing the structure of argumentative discourse. *Computational Linguistics*, 13(1-2):11–24, 1987.

[71] G.W. Cottrell. Toward connectionist semantics. In *Theoretical Issues in Natural Language Processing 3 Position Papers*, pages 65–70, 1987.

[72] K. Dahlgren and J. McDowell. Kind types in knowledge representation. In *Proc. of the 11th International Conference on Computational Linguistics*, pages 216–221, 1986.

[73] M. Dincbas. Constraints, logic programming and deductive databases. In *Proc. of the France-Japan Artificial Intelligence and Computer Science Symposium'86*, pages 1–27, 1986.

[74] D.W. Etherington. *Reasoning with Incomplete Information. Research Notes in Artificial Intelligence*, Pitman, 1988.

[75] C. Fillmore. The case for case. In E. Bach and R. Harms, editors, *Universals in Linguistic Theory*, pages 1–88, Holt, Rinehart and Winston, 1968.

[76] M. Gerlach and M. Sprenger. Semantic interpretation of pragmatic clues : connectives, modal verbs, and indirect speech acts. In *Proc. of the 12th International Conference on Computational Linguistics*, pages 191–195, 1988.

[77] B.J. Grosz. Focusing and description in natural language dialogues. In A.K. Joshi, B.L. Webber, and I.A. Sag, editors, *Elements of discourse understanding*, pages 84–105, Cambridge University Press, 1981.

[78] B.J. Grosz and C.L. Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204, 1986.

[79] G. Hirst. *Anaphora in Natural Language Understanding : A Survey.* Volume 119 of *Lecture Notes in Computer Science*, Springer-Verlag, 1981.

[80] G. Hirst. *Semantic Interpretation and the Resolution of Ambiguity. Studies in Natural Language Processing*, Cambridge University Press, 1987.

[81] J.R. Hobbs. Coherence and coreference. *Cognitive Science*, 3(1):67–90, 1979.

[82] J.R. Hobbs. Resolving pronoun references. In B.J. Grosz, K.S. Jones, and B.L. Webber, editors, *Reading in Natural Language Processing*, pages 339–352, Morgan kaufman, 1986.

[83] J.R. Hobbs and S.M. Shieber. An algorithm for generating quantifier scoping. *Computational Linguistics*, 13(1-2):47–63, 1987.

[84] M. Johnson and E. Klein. Discourse, anaphora and parsing. In *Proc. of the 11th International Conference on Computational Linguistics*, pages 669–675, 1986.

[85] M. Kameyama. *Zero Anaphora : The Case of Japanese.* PhD thesis, Stanford University, 1985.

[86] R. Kaplan and J. Bresnan. Lexical-functional grammar : a formal system for grammatical representation. In J. Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages 173–281, MIT Press, 1982.

[87] L. Karttunen. Features and values. In *Proc. of the 10th International Conference on Computational Linguistics*, pages 28–33, 1984.

[88] R.T. Kasper and W.C. Rounds. A logical semantics for feature structures. In *Proc. of the 24th Annual Meeting of the Association for Computational Linguistics*, pages 257–266, 1986.

[89] W.A. Kornfeld. Equality for prolog. In *Proc. of the 8th International Joint Conference on Artificial Intelligence*, pages 514–519, 1983.

[90] S. Kuno. *Functional Syntax – Anaphora, Discourse and Empathy.* University of Chicago Press, 1987.

[91] 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.

[92] J. McCarthy. Applications of circumscription to formalizing common-sense knowledge. *Artificial Intelligence*, 28(1):89–116, 1986.

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

[94] A. Meulen. Processing pronouns – a comparison of situations semantics and discourse representation theory. In *Proc. of the 9th German Workshop on Artificial Intelligence*, Springer-Verlag, 1986.

[95] M. Minsky. A framework for representing knowledge. In P. Winston, editor, *The Psycology of Computer Vision*, pages 211–277, McGraw-Hill, 1975.

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

[97] P. Norvig. Dimensions of ambiguity. *Monthly Newsletter of the Center for Research in Language*, 1(6), 1987.

[98] F. Pereira. Extraposition grammar. *American Journal of Computational Linguistics*, 7(4):243–256, 1981.

[99] F. Pereira and D. Warren. Definite clause grammar for language analysis - a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence*, 13(3):231–278, 1980.

[100] C. Pollard and I.A. Sag. *HPSG : An Informal Synopsis.* Reports 79, CSLI, 1986.

[101] T. Reinhart. *Anaphora and Semantic Interpretation.* Croom Helm, 1983.

[102] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1-2):81–132, 1980.

[103] R.C. Schank and R.P. Abelson. *Scripts, Plans, Goals and Understanding.* Lawrence Erlbaum Associates, 1977.

[104] E. Schuster. Anaphora reference to events and actions : a representation and its advantages. In *Proc. of the 12th International Conference on Computational Linguistics*, pages 602–607, 1988.

[105] P. Sells. *Lectures on Contemporary Syntactic Theories : An Introduction to Government-Binding Theory, Generalized Phrase Structure Grammar, and Lexical-Functional Grammar.* Lecture Notes 3, CSLI, 1985.

[106] S.M. Shieber. *An Introduction to Unification-Based Approaches to Grammar.* Lecture Notes 4, CSLI, 1986.

[107] C.L. Sidner. Focusing in the comprehension of definite anaphora. In M. Brady and R.C. Berwick, editors, *Computational Models of Discourse*, pages 267–330, MIT Press, 1983.

[108] J.F. Sowa. *Conceptual Structures : Information Processing in Mind and Machine.* Addison-Wesley, 1984.

[109] R. Sugimura, H. Miyoshi, and K. Mukai. Constraint analysis on japanese modification. In *Proc. of the 2nd International Workshop on Natural Language Understanding and Logic Programming*, pages 5–18, 1987.

[110] D.S. Touretzky. *The Mathematics of Inheritance Systems. Research Notes in Artificial Intelligence*, Pitman, 1986.

[111] D.S. Touretzky, J.F. Horty, and R.H. Thomason. A clash of intuitions : the current state of nonmonotonic multiple inheritance systems. In *Proc. of the 10th International Joint Conference on Artificial Intelligence*, pages 476–482, 1987.

[112] D.L. Waltz and J.B. Pollack. Massively parallel parsing : a strongly interactive model of natural language interpretation. *Cognitive Science*, 9(1):51–74, 1985.

[113] B.L. Webber. So what can we talk about now? In M. Brady and R.C. Berwick, editors, *Computational Models of Discourse*, pages 331–371, MIT Press, 1983.

索引

論文リスト

査読付きの論文

- 田中穂積, 小山晴生, 奥村学, 知識表現形式 DCKR とその応用, コンピュータソフトウェア, Vol.3, No.4, pp.12-20, 1986.
- 奥村学, 田中穂積, BUP 系解析システム上でのトップダウンな情報の制御について, 情報処理学会論文誌, Vol.29, No.11, pp.1043-1050, 1988.
- 奥村学, 田中穂積, 制約蓄積による増進的曖昧性解消モデル, 情報処理学会論文誌, 投稿中.

査読なしの論文

- 田中穂積, 小山晴生, 奥村学, ボトムアップ構文解析システム BUP の拡張と日本語文法の試作, Proc. of the Logic Programming Conference'84, 12-3, 1984.
- 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, 日本ソフトウェア科学会第 1 回大会論文集, 3A-2, pp.227-230, 1984.
- 田中穂積, 池田光生, 奥村学, Definite Clause Dictionary – Prolog による辞書項目記述と意味処理, Proc. of the Logic Programming Conference'85, 12.1, pp.317-328, 1985.
- 田中穂積, 奥村学, 小山晴生, オブジェクトの同一性判定及び同一化アルゴリズムとその応用, 日本ソフトウェア科学会第 2 回大会論文集, 2A-2, pp.37-40, 1985.
- 田中穂積, 上脇正, 奥村学, 沼崎浩明, 自然言語処理のためのソフトウエアシステム LangLAB, Proc. of the Logic Programming Conference'86, 3.1, pp.5-12, 1986.
- 奥村学, 高倉伸, 田中穂積, 意味記述用言語 SRL/0 の設計と DCKR, 情報処理学会自然言語処理研究会報告, 54-5, 1986.
- 奥村学, 小池康晴, 田中穂積, 意味記述用言語 SRL/0 の設計と DCKR, 情報処理学会情報学基礎研究会報告, 1-2, 1986.
- 奥村学, 田中穂積, LangLAB における高水準辞書記述言語 SRL/0, 情報処理学会第 33 回全国大会講演論文集, 1N-3, pp.1431-1432, 1986.
- 奥村学, 田村直良, 徳永健伸, 田中穂積, BUP 系解析システム上でのトップダウンな情報の制御について, 日本ソフトウェア科学会第 3 回大会論文集, A-2-2, pp.17-20, 1986.

- 奥村学, 田村直良, 徳永健伸, 田中穂積, BUP 系解析システム上でのトップダウンな情報の制御について, 情報処理学会自然言語処理研究会報告, 59-5, 1987.
- 奥村学, 岩山真, 田中穂積, BUP 系解析システム上でのトップダウンな情報の制御について, 情報処理学会第 35 回全国大会講演論文集, 3T-8, pp.1359-1360, 1987.
- 山田基司, 奥村学, 田中穂積, 知識表現形式 DCKR の高速化と機能拡張, 日本ソフトウェア科学会第 4 回大会論文集, D-2-5, pp.223-226, 1987.
- 堺和宏, 徳永健伸, 奥村学, 田中穂積, 自然言語の意味処理用辞書の構成法, 情報処理学会自然言語処理研究会報告, 66-2, 1988.
- 奥村学, 田中穂積, 例外を含む多重継承ネットワークにおける継承アルゴリズムについて, 日本ソフトウェア科学会第 5 回大会論文集, A7-5, pp.313-316, 1988.
- 田中穂積, 山田基司, 奥村学, 知識表現形式 DCKR の高速化と機能拡張, 知識情報処理ハンドブック, 福村晃夫（編）, オーム社, pp.103-111, 1988.