動的な日本語解析モデルと計算機上での実装

小林昌博

1 はじめに

本稿では，文を動的に解析する計算モデルを提案し，数量詞遊離構文を用いてそのモデルの妥当性を検討する．また，その計算機モデルについて検討する．

言語は線状的な性質を持つため，我々は文を読んだり聞いたりする際に左から右へと単語を一語ずつ読み込み，逐次的に文を解釈していく[1]．一方，文の生成の観点から考えても生成された発話列は線状的な性質を持つ．このような言語の線状性は，文の情報構造のみならず，コミュニケーションの構造にも大きな影響を与えている．本稿では，数量詞の遊離現象などを例にとり，人間が一次元的な単語列を読み込み，動的に文の意味を解釈していく過程を Dynamic Syntax (Kempson, Meyer-Viol and Gabbay 2001) を用いて定式化し，計算機上に実装することで言語理解のメカニズムについて考察する．

本稿の構成は以下の通りである．次節では，言語の線状性という概念を概観し，日本語を処理する場合の問題点を指摘し，数量詞遊離構文について述べる．3節では，本稿で採用する枠組みである Dynamic Syntax (Kempson et al. 2001) について解説する．4節では，本稿で提案するモデルがどのように数量詞遊離構文を扱えるかを示す．5節では，Perl を用いた理論の実装について解説する．6節は結論である．

2 言語の線状性と問題の所在

前節でも述べたように，我々は単語を一語ずつ読み込みながら文の意味を動的に解釈していく．つまり，我々は一次元的な記号列から脳内で三次元的な意味の世界を構築している．人間の言語能力が線状的な記号列を処理する計算機構だとすると，言語理解のモデリングは，この逐次的な言語解析の過程をシミュレートすることでなされるはずである．例えば (1) の例を見てみる．

(1) 太郎が本を買ったと花子が言った．

(1) の例を我々は左から右へと逐次的に発話する．(1) において，文頭の名詞「太郎」が発話された段階で，我々は「太郎」が主語なのか目的語なのか知ることができない．「太郎が」と

[1] アラビア語、ヘブライ語等の言語では日本語や英語とは逆に右から左へ読まれる．

発話された段階で，この名詞句は主語であると理解できるが，主文の主語なのか，連体修飾節の主語なのか，従属節の主語なのかまではわからない．実際,「太郎が本を買った」までの発話を考えれば，これ自体で完全に文法的な文となり,「太郎が」は主文の主語ということになる．しかし次の補文標識の「と」が発話された段階で，文頭の「太郎が」は主文の主語ではなく，従属節の主語であることが初めてわかる．日本語用いてコミュニケーションをとる際に，我々はこのような複雑な作業を無意識のうちに行っていることになる．これは，英語等とは異なり，日本語がヘッド・ファイナル言語であることとも大きく関係している．

英語は語順が比較的厳密な言語であるが，日本語は語順がある程度ゆるやかな言語なため，線状性に関係する問題が多く研究されてきた．例として数量詞遊離構文を見てみる．数量詞遊離とは (2a) に見られる数量表現「3 冊」が (2b) のように右方に移動する現象を指す．

(2) a. 3 冊の本を花子が買った．

b. 本を花子が 3 冊買った．

(2b) を逐次解析する場合，数量表現である「3 冊」を読み込んだときに,「3 冊」がすでに読み込まれたどの名詞句と修飾関係にあるかをコンピュータに処理させるのは易しいことではない．Bond, Kurz and Shirai (1998) は，機械翻訳や言語処理の立場からこの問題を扱っている．さらに，数量表現の遊離はいかなる場合も可能というわけではない．

(3) a. 300 人の警官でこの暴動を鎮圧した．

b. * 警官でこの暴動を 300 人鎮圧した．

文頭に「*」マークがついている文は，その文が非文法的であることを示している．(3) の例において，数量表現「300 人」が修飾している名詞句「警官で」は動詞の項ではなく付加詞である．Miyagawa (1989) は樹形図上の c-command という概念を用いて (3b) のような遊離が不可能であることを説明している．さらに数量詞の遊離は (4) のように目的語からの遊離と主語からの遊離では明らかに文法性に差がある．

(4) a. * 学生がコンピュータを 5 人買った．

b. コンピュータを学生が 5 台買った．

(4) では，遊離数量詞とそれが修飾する名詞句の両方に下線を引いてある．(4a) では，主語である「学生が」から数量表現「5 人」が遊離しており，かつ名詞句がその間に挿入されている．(4b) では，目的語から数量表現「5 台」が遊離しており，その間に名詞句が挿入されている．(4) のように，主語からの遊離か，目的語からの遊離かで文法性に差が見られる．Saito (1985) は移動の概念を用いて (4) の例を説明できているように見えるが，高見 (1998) のように (4) に見られる主語，目的語対称性への反例も報告されている．(5) は高見 (1998) からの例である．

(5) 今朝，学生さんがその新刊雑誌を 5 人買って行きましたよ．

(5) は (4a) と同じ統語パターンであるが，(5) の方が文法の許容性が高いであろう．高見 (1998) は，(5) では主語と数量詞の間の名詞句が定名詞句であるのに対して，(4a) では不定名詞句であることが文法性に差がある原因であると主張している．つまり，高見 (1998) によると文の情報構造が数量詞遊離の解釈に関わっていると述べている．Miyagawa (1989) や Saito (1985) の分析では，(5) の文法性は説明できない．

上記の先行研究は本稿のような言語理解のモデルを意識したものではない．次節以降，本稿の動的な文解析モデルが上記のような問題も解決する言語理解モデルであることを示す．

3　Dynamic Syntax による逐次的文解析

Dynamic Syntax (Kempson et al. 2001) は，文を文頭から一語ずつ読み込み，逐次的に文の解釈を組み立てていく理論である．Dynamic Syntax では，樹形図に相当するものはノードの集合として表現され，ノードの支配関係は tree modality により表される．樹形図に相当するノードの集合は，解析が進むにつれて情報が増え，成長していく．解析の初期状態は *axiom* と呼ばれ，(6) のような一つの単純なノードである (Kempson et al. 2001: pp.76).

(6) *Axiom*: $\{Tn(a), ?Ty(t), \Diamond\}$

「$Tn(a)$」は，樹形図におけるノードの位置を示す．「$?Ty(t)$」における「?」は requirement と呼ばれ，$?Ty(t)$ はこのノードが解析の終了時までにタイプ t のノードになることを意味する．タイプ t とは真偽値を問えることを意味し，通常，文のレベルを表す．つまり「$?Ty(t)$」は解析の初期段階では，今から読み込まれる記号列が文であることを要求している．「$\Diamond$」はポインタと呼ばれ，実際の解析ではポインタのあるノードに規則が適用されていく．単語の語彙情報は「IF α, THEN β, ELSE γ」という規則の形式をとる。これは「もしポインタを含むノードに α が存在するならば，β という操作を実行する．存在しなければ γ を実行する」という意味である．この語彙の構造は Dynamic Syntax の特徴の一つである．

Dynamic Syntax による実際の文解析の例として，単純な文 (7) を例にとり解説する．なお，Dynamic Syntax を用いた日本語の解析例は，Kempson et al. (2001: pp.67-75) を参照されたい．ここでの日本語の解析方法も Kempson et al. (2001: pp.67-75) の解説に従っている．

(7) 太郎がミカンを食べた．

(7) を実際に解析する際の樹形図の初期状態は (6) である．まず最初に名詞「太郎」が読み込まれる．名詞「太郎」は単語の情報として (8) のような情報を設定する．

(8) *Taro*
　IF　　$\{?Ty(t)\}$,
　THEN　make($\langle\downarrow_*\rangle$); put($Fo(Taro)$, $Ty(e)$, $Def(+)$)
　ELSE　ABORT

「太郎」の辞書情報である規則 (8) はポインターのあるノード，ここでは「太郎」は最初に読み込まれる単語であるため，(6) に適用される．(8) の意味は，もし (6) に「$?Ty(t)$」が存在

すれば，**THEN** 以下の操作を実行せよ，という意味である．もし「$?Ty(t)$」がなければ，解析は失敗する．ここで**THEN** 以下の操作について解説する．まず，述語である make であるが，これは新たなノードを作成し，樹形図を拡張することを意味している．この場合，具体的にはポインタの下に点線でつながれたノードを作成せよ，という指示である．この make の操作により，ポインタは新しく作成されたノードに移動する．make の次に適用される操作は，put である．この put により，ポインタのあるノードに put が引数でとっている要素が挿入される．ここでは $Fo(Taro)$ と $Ty(e)$ および $Def(+)$ である．$Fo(Taro)$ は，このノードの意味表示が $Taro$ であることを表している．ここでは，名詞「太郎」の意味は $Taro$ である．これらの表示は，最終的に一階の述語論理式に変換される．$Ty(e)$ は，このノードのタイプが e であることを表している．なお，通常の形式意味論の扱いとは異なり，Dynamic Syntax では，量化表現や固有名詞などすべての名詞のタイプは e となる．$Def(+)$ は，この名詞が定の名詞であることを表している．(8) を読み込んだ後の樹形図は (9) となる．

(9)
$$\{Tn(0), ?Ty(t)\}$$
$$\vdots$$
$$\{Fo(Taro), Ty(e), Def(+), \Diamond\}$$

(9) に見られるように，文頭の名詞「太郎」を読み込むことによって (6) の初期状態が拡張されている．ここで一番上のノードをルートノードと呼ぶ．「太郎」を表す新しいノードが点線によりルートノードと結ばれている理由は $\texttt{make}(\langle\downarrow_*\rangle)$ の操作によるものである．Dynamic Syntax では，最終的な樹形図におけるノードの位置が不確定な (underspecified) 場合，そのノードは暫定的に点線で結合される．ここでは，文頭の名詞句「太郎」を読み込んだ時点では「太郎」が主節の主語なのか，従属節の主語なのかがわからない．したがって，「太郎」が最終的な樹形図において占める位置が判明するまではルートノードに点線でつながれている．

次に格助詞「が」が読み込まれる．本稿では，格助詞「が」の語彙情報を (10) のように定義する[2]．

(10) *ga*

IF $\{Ty(e)\}$,
THEN IF $\{Def(+)\}$,
THEN $\texttt{put}(\langle\uparrow_0\rangle Ty(t)); \texttt{gofirst}_\uparrow(Tn(0)); \texttt{put}(SubjDef(+))$
ELSE $\texttt{put}(\langle\uparrow_0\rangle Ty(t)); \texttt{gofirst}_\uparrow(Tn(0)); \texttt{put}(SubjDef(-))$
ELSE ABORT

(10) において，まずポインタのあるノードのタイプが e であれば，**THEN** 以下を実行する．$\texttt{gofirst}_\uparrow(Tn(0))$ はポインタを現在の位置から上の方向に一番最初に $Tn(0)$ があるノードに移動せよ，という意味である．ここではルートノードに相当する．この定義によると，直前に読み込んだ名詞が定か不定かによりルートノードに $SubjDef(+)$ か $SubjDef(-)$ が挿入

[2]Kempson et al. (2001: pp.70) の提案する「が」の辞書情報と (10) は異なっている．本稿では (10) に見られるように，名詞の定，不定を扱うための新しい格助詞「が」に関する辞書情報の定義を提案する．

される．$\langle\uparrow_0\rangle Ty(t)$ は，最終的な樹形図における位置において，上に上がるとタイプ t のノードがあることを意味している．つまり，タイプ t であるルートノードに直接支配されることを表している．下付の 0 は当該のノードが関数の引数であることを表している．つまり，フレーゲの構成性原理 (compositionality) によると名詞句「太郎が」は動詞句の引数となる．関数適用の際の関数子 (functor) は下付の 1 で表される．(10) の適用後の樹形図は (11) となる．

(11)　　$\{Tn(0),\ ?Ty(t),\ SubjDef(+),\ \Diamond\}$

$\{Fo(Taro),\ Ty(e),\ Def(+),\ \langle\uparrow_0\rangle Ty(t)\}$

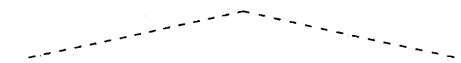
(9) と (11) と比較すると，解析が進むと樹形図の情報が増えていることがわかる．次の単語「ミカン」の辞書情報を (12) のように定義する．(12) のような普通名詞の辞書情報は，(8) のような固有名詞の辞書情報とは異なり，少し複雑になっている．これは，普通名詞の前に定冠詞がつく場合があるため，定，不定の区別をつけるためである．

(12) *mikan*

IF　　$\{?Ty(t)\}$,
THEN make($\langle\downarrow_*\rangle$); put($Fo(\epsilon,\ x,\ Orange(x))$, $Ty(e)$)
ELSE　**IF**　　$\{?Ty(e)\}$,
　　THEN　**IF**　　$\{Def(+)\}$,
　　THEN　put($Fo(\iota,\ x,\ Orange(x))$)
　　ELSE　put($Fo(\epsilon,\ x,\ Orange(x))$)
　　ELSE　ABORT

名詞「ミカン」を読み込んだ後の樹形図は (13) となる．

(13)



$\{Fo(Taro),\ Ty(e),\ Def(+),\ \langle\uparrow_0\rangle Ty(t)\}$　　$\{Fo(\epsilon,\ x,\ Orange(x)),\ Ty(e),\ \Diamond\}$

(13) に見られる $Fo(\epsilon,\ x,\ Orange(x))$ は，エプシロン計算に基づく表記であるが，一階の述語論理式 $\exists x[Orange(x)]$ と等価である．次の格助詞「を」の辞書情報は (14) のようになる．

(14) *o*

IF　　$\{Ty(e)\}$,
THEN IF　　$\{Def(+)\}$,
　　THEN　put($\langle\uparrow_0\rangle Ty(e\rightarrow t)$); gofirst$_\uparrow$($Tn(0)$); put($ObjDef(+)$)
　　ELSE　put($\langle\uparrow_0\rangle Ty(e\rightarrow t)$); gofirst$_\uparrow$($Tn(0)$); put($ObjDef(-)$)
ELSE　ABORT

(14) の適用により，樹形図 (13) は (15) となる．

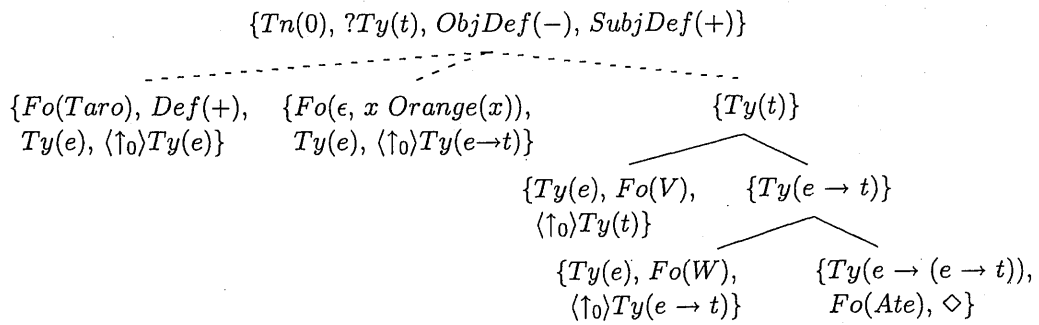
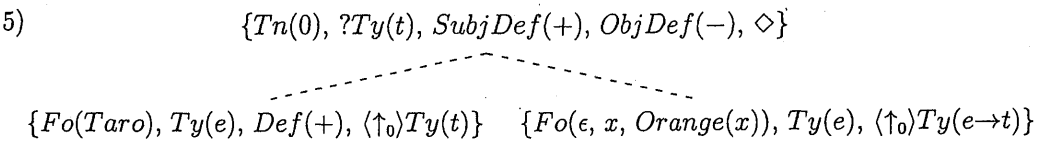


図 1: 「食べた」を読み込んだ後の樹形図

(15)



最後に読み込まれる単語は他動詞の「食べた」である.「食べた」の辞書情報は (16) のように定義される.

(16) *tabeta*

IF $\{?Ty(t)\}$,
THEN make($\langle\downarrow_*\rangle$); put($Ty(t)$);
make($\langle\downarrow_0\rangle$); put($Fo(V)$, $Ty(e)$, $\langle\uparrow_0\rangle Ty(t)$}; go($\langle\uparrow\rangle$);
make($\langle\downarrow_1\rangle$); put($Ty(e\rightarrow t)$);
make($\langle\downarrow_0\rangle$); put($Fo(W)$, $Ty(e)$, $\langle\uparrow_0\rangle Ty(e\rightarrow t)$); go($\langle\uparrow\rangle$);
make($\langle\downarrow_1\rangle$); put($Fo(Ate)$, $Ty(e\rightarrow e\rightarrow t)$); gofirst$_\uparrow$($Ty(t)$)
ELSE ABORT

(16) は Kempson et al. (2001: pp.72) における他動詞「食べた」の定義とほぼ同じである．日本語では，動詞の前に生じる名詞句は動詞が読み込まれるまで最終的な樹形図の位置がわからない．その不確定性を解消するのは動詞の役目であるため，(16) に見られるように，動詞は 'verb-frame' (Kempson et al. 2001: pp.72) と呼ばれる部分木を導入する．(16) を読み込んだ後の樹形図は図 1 になる．図 1 において，$Fo(V)$ や $Fo(W)$ などの素性があるが，これらは meta-varible (Kempson et al. 2001: pp.34) と呼ばれ，意味的な情報は何もないが，引数としての場所 (slot) を提供していることを表す．

図 1 の樹形図ができた段階で，これ以上読み込む単語が存在しないので，残された処理は点線で結ばれたノードの最終的な位置を決める作業である．まず，主語と目的語のノードは $Fo(V)$ や $Fo(W)$ のあるノードに置き換えられる．この置き換え作業は，それぞれのノードのタイプのマッチングに基づき行われる．たとえば，$Fo(V)$ があるノードには $\langle\uparrow_0\rangle Ty(t)$ の

素性があるため，$Fo(Taro)$ のあるノードとマッチする．このような置き換え作業後の樹形図は図 2 となる．ルートノードはその下の $Ty(t)$ のノードと置き換えられる．

点線がなくなって最終的な樹形図ができた段階で，意味の計算が始まる．意味計算は，β-reduction によりボトムアップに行われる．最終的な意味表示は (17) になる．

(17) $\exists x[Orange(x) \wedge Ate(x)(john)]$

(17) は通常の一階述語論理式である．

次節では，数量詞の遊離構文がどのように線状性に基づく解析で処理されるかを解説する．

4　言語の線状性と数量詞遊離構文

本節では，Miyagawa (1989) と高見 (1998) により扱われた一見相反するような例が本稿での提案で解決できることを示す．まず，Miyagawa (1989) で主張された mutual c-command で説明できる例を Dynamic Syntax を用いて解析し，次に高見 (1998) の例を扱う．

4.1　Miyagawa (1989) の例と Dynamic Syntax

Miyagawa (1989) の例では，数量詞とそれが修飾する名詞句が mutual c-command の関係になければならない．(18) を例にとり考えてみる．

(18) 本を$_i$ 太郎が t_i 3 冊買った．

(18) における t_iは目的語「本を」が移動した痕跡 (trace) である．Miyagawa (1989) によると，(18) の名詞句「本を」と数量詞「3 冊」は離れているが，数量詞と痕跡が mutual c-command の関係にあるので，文法性に問題はない．Miyagawa (1989) の説明に見られるように，生成文法は移動を用いる言語生成モデルであるため，痕跡の概念を用いる必要が生じる．Dynamic Syntax を用いた分析では，上記の 2 つの問題を特別な規定なしに解決できる．(18) の例を Dynamic Syntax を用いて解析してみる．(19) は「本を太郎が」まで解析した樹形図である．解析方法は前節で解説した方法に基づいている．

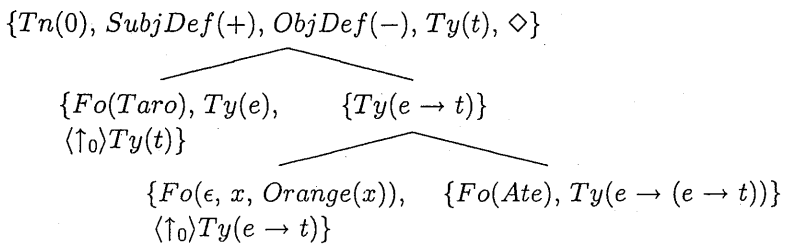


図 2: (7) の最終的な樹形図

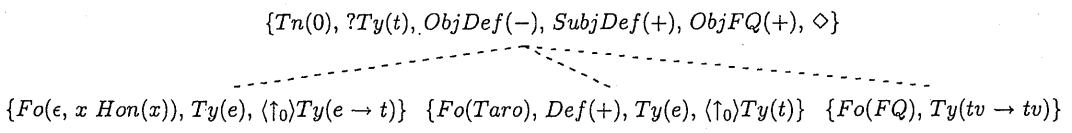
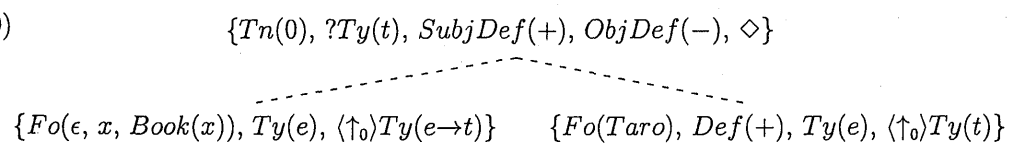


図 3: (18) の数量詞まで読み込んだ樹形図

(19)



次に，数量詞「3 冊」の辞書情報を (20) のように定義する．

(20) *san-satsu*
IF $\{?Ty(t), ObjDef(-)\}$,
THEN $\mathtt{put}(ObjFQ(+)); \mathtt{make}(\langle\downarrow_*\rangle); \mathtt{put}(Ty(tv \rightarrow tv), Fo(FQ)); \mathtt{gofirst}_{\uparrow}(Tn(0))$
ELSE **IF** $\{?Ty(t), SubjDef(-)\}$,
THEN $\mathtt{put}(SubjFQ(+)), \mathtt{make}(\langle\downarrow_*\rangle); \mathtt{put}(Ty(vp \rightarrow vp), Fo(FQ));$
$\mathtt{gofirst}_{\uparrow}(Tn(0))$
ELSE ABORT

(20) によると，数量詞はルートノード上にある主語と目的語に関する定，不定の情報を参照し，主語遊離の数量詞 ($SubjFQ(+)$) か目的語遊離の数量詞 ($ObjFQ(+)$) かを区別する．遊離数量詞のタイプを見ると主語遊離数量詞は $Ty(vp \rightarrow vp)$ であり，目的語遊離数量詞は $Ty(tv \rightarrow tv)$ である．(20) を読み込んだ後の樹形図は図 3 になる．

他動詞「買った」の辞書情報として，本稿では数量詞を扱うために，すでに定義してある (16) や Kempson et al. (2001: pp.72) の定義とは異なる定義を与える．「買った」の辞書情報は図 4 を参照されたい．図 4 の定義により，主語遊離の数量詞がある場合と目的語遊離の数量詞がある場合，さには遊離数量詞がない場合で異なる verb-frame が導入される．「買った」を読み込んだ後の樹形図は図 5 になる．図 5 に見られるように，遊離数量詞はルートノードにある目的語遊離数量詞 ($ObjFQ(+)$) を参照して，他動詞「買った」を引数としてとる遊離数量詞用の関数子の slot を提供する．この段階で読み込む単語は存在しないので，残る操作は点線で結ばれているノードに最終的な場所を提供することである．これはすでに説明してあるようにタイプのマッチングにより行われる．(18) の最終的な樹形図は図 6 になる．図 6 を見ると，(18) における表層の語順が反映されていない．つまり，(18) では，目的語と主語が倒置されているが，樹形図は主語，目的語の順になっている．これは，動詞の前の要素を不確定 (underspecified) な状態にして，最後に場所の指定を行うことにより実現される．図 6 の樹形図により得られる意味表示は (21) である．

(21) $\exists x[|x| = 3 \wedge Hon(x) \wedge Katta(x)(taro)]$

katta
IF $\{?Ty(t)\}$,
THEN IF $\{SubjFQ(+)\}$,
THEN make($\langle\downarrow_*\rangle$); put($Ty(t)$);
make($\langle\downarrow_0\rangle$); put($Ty(e)$, $Fo(V)$, $\langle\uparrow_0\rangle Ty(t)$); go($\langle\uparrow\rangle$);
make($\langle\downarrow_1\rangle$); put($Ty(e \rightarrow (e \rightarrow t))$;
make($\langle\downarrow_1\rangle$); put($Ty((e \rightarrow t) \rightarrow (e \rightarrow t))$, $Fo(W)$; go($\langle\uparrow\rangle$);
make($\langle\downarrow_0\rangle$); put($Ty(e \rightarrow t)$);
make($\langle\downarrow_0\rangle$); put($Ty(e)$, $Fo(X)$, $\langle\uparrow_0\rangle Ty(e \rightarrow t)$); go($\langle\uparrow\rangle$);
make($\langle\downarrow_1\rangle$); put($Ty(e \rightarrow (e \rightarrow t))$, $Fo(Katta)$)
ELSE IF $\{ObjFQ(+)\}$,
THEN make($\langle\downarrow_*\rangle$); put($Ty(t)$);
make($\langle\downarrow_0\rangle$); put($Ty(e)$, $Fo(V)$, $\langle\uparrow_0\rangle Ty(t)$); go($\langle\uparrow\rangle$);
make($\langle\downarrow_1\rangle$); put($Ty(e \rightarrow t)$);
make($\langle\downarrow_0\rangle$); put($Ty(e)$, $Fo(W)$, $\langle\uparrow_0\rangle Ty(e \rightarrow t)$); go($\langle\uparrow\rangle$);
make($\langle\downarrow_1\rangle$); put($Ty(e \rightarrow (e \rightarrow t))$);
make($\langle\downarrow_1\rangle$); put($Ty(tv \rightarrow tv)$, $Fo(X)$); go($\langle\uparrow\rangle$);
make($\langle\downarrow_0\rangle$); put($Ty(e \rightarrow (e \rightarrow t))$, $Fo(Katta)$)
ELSE make($\langle\downarrow_*\rangle$); put($Ty(t)$);
make($\langle\downarrow_0\rangle$); put($Ty(e)$, $Fo(V)$, $\langle\uparrow_0\rangle Ty(t)$); go($\langle\uparrow\rangle$);
make($\langle\downarrow_1\rangle$); put($Ty(e \rightarrow t)$);
make($\langle\downarrow_0\rangle$); put($Ty(e)$, $Fo(W)$); go($\langle\uparrow\rangle$);
make($\langle\downarrow_1\rangle$); put($Ty(e)$, $Fo(Katta)$)
ELSE ABORT

図 4: 他動詞「食べた」の辞書情報

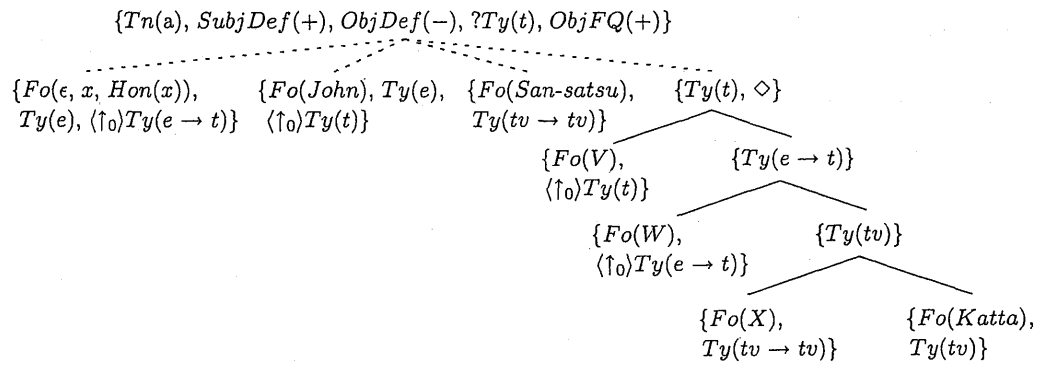


図 5: 「買った」まで読み込んだ樹形図

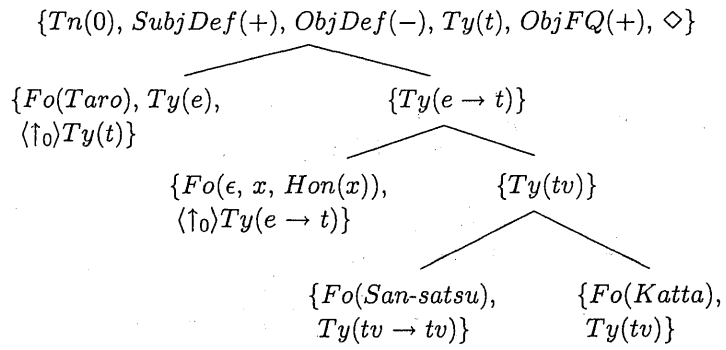


図 6: (18) の最終的な樹形図

(21) は (18) の意味を正しく表している．このように，Miyagawa (1989) で用いられた mutual c-command の規定などを想定しなくても，Dynamic Syntax で数量詞遊離構文の解析が可能である．次の小節では，Miyagawa (1989) の mutual c-command の説明を批判する高見 (1998) の規定も Dynamic Syntax で同様に扱えることを示す．

4.2 高見 (1998) の例と Dynamic Syntax

高見 (1998) は，Miyagawa (1989) の mutual c-command で説明できない例を挙げ，Miyagawa (1989) を批判すると同時に，数量詞の解釈には名詞句の定，不定がもたらす情報構造が関与する，と主張した．本節では，高見 (1998) の主張を Miyagawa (1989) と矛盾しない形で定式化できることを示す．例文 (22)(=(5)) をもう一度見てみる．(22) は高見 (1998) からの引用である．

(22) 今朝，学生さんが その新刊雑誌を 5 人 買って行きましたよ．

(22) の例では，数量詞「5 人」とそれが修飾する名詞句「学生さんが」が mutual c-command の関係にない．2 節で述べたように，高見 (1998) は数量詞は新情報，つまり不定の名詞句と修飾関係を持ちやすいので，(22) では数量詞と主語が隣接していなくても文法的になると主張している．この高見 (1998) の主張も Dynamic Syntax を用いると，比較的容易に定式化が可能である．(22) の動詞まで読み込んだ段階の樹形図は図 7 になる．図 7 において，他動詞「買った」の辞書情報の構造は，図 4 と同じであり，遊離数量詞「5 人」の辞書情報は，すでに定義した遊離数量詞「3 冊」の辞書情報 (20) と構造は同じである．図 7 に対して置き換え操作を適用した，(22) の最終的な樹形図は図 8 になる．最終的な樹形図 8 より導出される論理式は (23) になり，(23) は (22) の意味を正しく表示している．

(23) $\exists x[|x| = 5 \wedge Gakusei(x) \wedge \iota y[Zasshi(y) \wedge Katta(y)(x)]]$

本稿での定式化の独自性は以下の 2 点にまとめられる．1 点目は，高見 (1998) で議論されているような，mutual c-command の関係にない例も Miyagawa (1989) の例も最終的な樹形

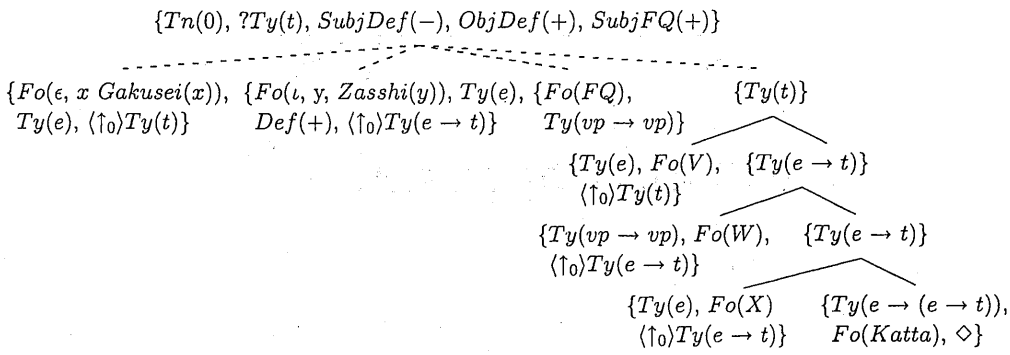


図 7: (22) の最後まで読み込んだ樹形図

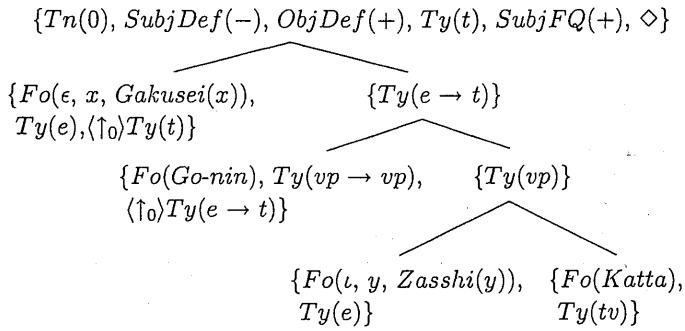


図 8: (22) の最終的な樹形図

図では，ともに遊離数量詞と名詞句が隣接する位置関係にある，ということである．図 8 を見てみると，数量詞「5 人」は表層の語順とは異なり，名詞句「学生が」と隣接関係にある．このように，一見相反する Miyagawa (1989) と高見 (1998) の例も抽象的な樹形図のレベルで同一の構造として処理可能である．

2 点目は，Miyagawa (1989) と高見 (1998) の例を同じ枠組みで処理するために，高見 (1998) の提案された，名詞句の定，不定の情報を Dynamic Syntax の辞書情報として定式化している点である．本稿の定式化を用いると，Miyagawa (1989) の例も高見 (1998) の例もアドホックな規定なしに扱えることがわかる．

5　計算機上での実装

前節までで述べた定式化を計算機上に実装し，現在データ解析を行っている[3]．本節ではその計算機モデルの解説を行う．実装には Perl を用いており，文法開発を容易に行うために Perl/Tk を用いたインタフェースを備えている．パーサのコンソール画面は図 9 に示される．

[3]データを用いた検証に関しては，Kobayashi and Yoshimoto (2003) を参照されたい．

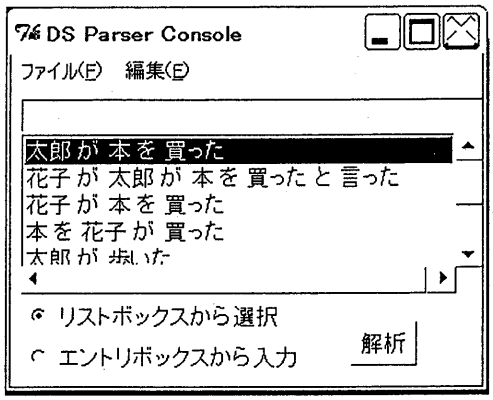


図 9: パーサのコンソール画面

```
#katta
sub katta {
     if (check_if("Ty(t)")) {
          make("⟨↓*⟩"); put("Ty(t)");
          make("⟨↓0⟩"); put("Ty(e)", "Fo(V)", "⟨↑0⟩Ty(t)"); go("Ty(t)");
          make("⟨↓1⟩"); put("Ty(e->t)");
          make("⟨↓0⟩"); put("Ty(e)", "Fo(W)", "⟨↑0⟩Ty(e->t)"); go("Ty(e->t)");
          make("⟨↓1⟩"); put("Ty(e->e->t)", "Fo(Katta)");
          go_first("Ty(t)");
     } else {
          abort();
     };
};
```

図 10: 「買った」を表すサブルーチン

図9に見られるように，リストボックスに解析対象となる文が表示される．別のファイルに書いてある文を読み込むことも可能である．またエントリボックスから直接入力することもできる．辞書の情報はPerlのサブルーチンとして記述される．たとえば，他動詞「買った」の辞書情報は図10のようになる．

Dynamic Syntaxでは，樹形図はノードの集合として表現され，各ノードの位置関係はtree modalityにより表される．本稿のモデルでは可能な限りDynamic Syntaxの操作を忠実に再現しているが，解析の効率を考慮し，点線で表されていた位置が固定してないノードと固定しているノードを別な配列で扱っている．解析アルゴリズムの概略は図11になる．例として「本を花子が買った」の解析結果を図12及び図13に示す．

```
initialize   @unfixed_nodes, @fixed_nodes;   #各ノードを格納する配列の初期化
foreach   words
      apply each lexical information;        #各単語の辞書情報を適用
      specify nodes;                         #ノード位置の指定
apply substitution;                          #点線のすべてのノードを位置を固定
foreach   nodes
      apply thinning;                        #各ノードにおいて重複する情報を削除
build semantics;                             #論理式を導出
```

図 11: 解析処理のアルゴリズム

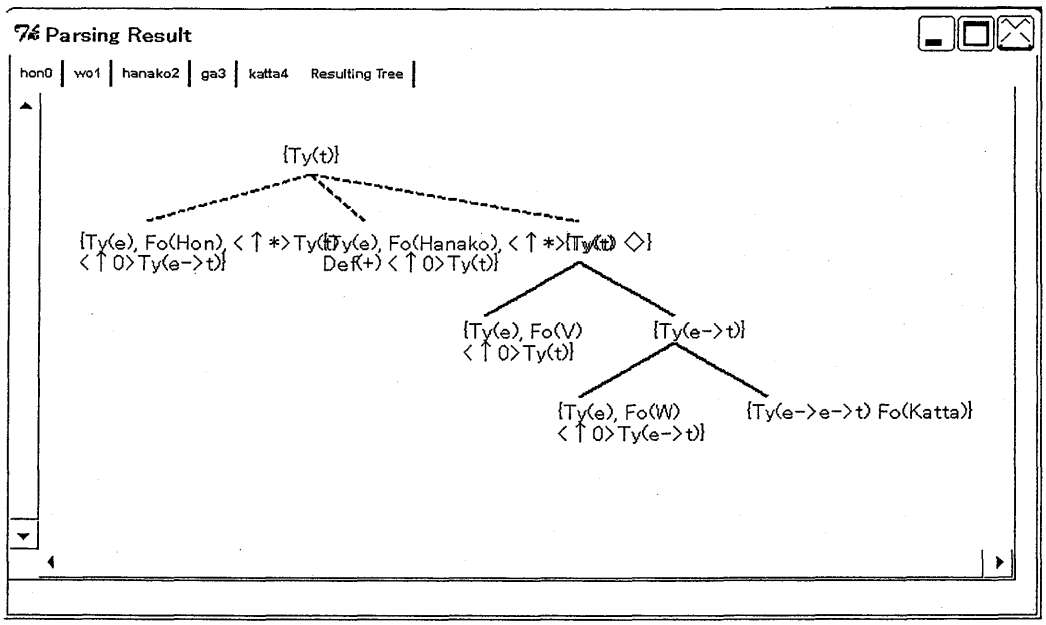


図 12: 解析結果のスナップショット

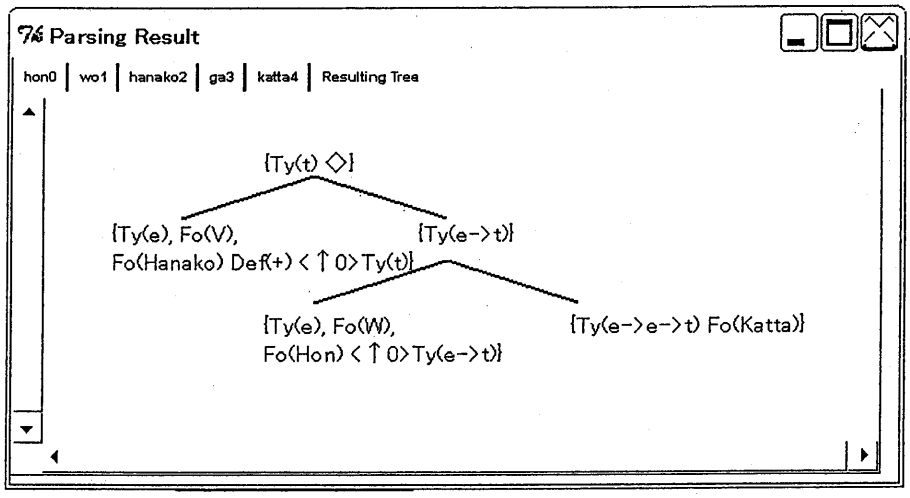


図 13: 解析結果のスナップショット

6 おわりに

本稿では，人間の言語理解をモデル化した，Dynamic Syntax (Kempson et al. 2001) に基づく逐次的な言語解析モデルを示した．数量詞の遊離構文に関しては，Miyagawa (1989) と高見 (1998) のデータを単一の枠組みで扱えることを示した．具体的には，高見 (1998) の提案する名詞句の定，不定と遊離数量詞の関係を Dynamic Syntax の辞書情報に定式化し，双方の例が抽象的な樹形図のレベルでは同一の形式になることを示した．今後の課題は，この枠組みが語順に関する他の構文にも有効かどうかを検討することである．

参考文献

Bond, F., D. Kurz and S. Shirai. 1998. Anchoring Floating Quantifiers in Japanese-to-English Machine Transltion. in *17th International Conference on Computational Linguistics: COLING-98.* pp.152-159.

Kempson, R., W. Meyer-Viol and D. Gabbay. 2001. *Dynamic Sytnax: The Flow of Language Understanding.* Oxford: Blackwell Publishers.

Kobayashi, M. and K. Yoshimoto. 2003. Association of Floating Quantifiers with NPs: A Linear Order Perspective. *Proceedings of The Ninth Annual Meeting of The Association for Natural Language Processing.* pp.171-174.

Miyagawa, S. 1989. *Syntax and Semantics 22: Structure and Case Marking in Japanese.* San Diego: Academic Press.

Saito, M. 1985. *Some Asymmetries in Japanese and Their Theoretical Implications.* Ph.D. dissertation. MIT.

高見健一. 1998. 日本語の数量詞遊離について：機能論的分析（下）.「言語」27: 3. pp.99-107.

(2004 年 11 月 15 日受理)