論理文法におけるギャップの扱い†

徳永健伸†† 岩山 真†† 田中穂積††

英語における関係節やWh疑問文に現れる構成素の移動現象を論理文法上で扱う枠組がPereiraのXGをはじめとして，いくつか提案されている．これらはいずれも構成素が移動した後には仮想的なカテゴリがギャップとして残るという考え方に基づき，ギャップの存在を文法記述中に許している．このような文法記述では，一般に記述すべき文法規則の数が少なくてすむという利点があるが，これまでの枠組では，移動した構成素と後に残されたギャップとの同一性を十分に記述できない場合がある．本論文では，DCGに支配制約という概念を導入することによって，構成素の移動現象を自然に記述する枠組を提案する．この文法記述形式をLG[3]と呼ぶ．構成素の移動を記述するためにスラッシュ記法と呼ばれる記法がいくつかの枠組で提案されているが，本論文ではスラッシュ記法の制約を緩めた支配制約という記法を提案し，両者を組み合わせることによって柔軟な文法記述力が得られることを示す．一般にスラッシュ記法は親カテゴリがあるカテゴリを支配し，そのカテゴリが必ずギャップを支配することを要請するものであるが，支配制約は親カテゴリの下に，あるカテゴリが存在することしか要請しない．また，PereiraのDCGの実装法を拡張することによって，支配制約がProlog上で容易に実装できることを示す．

1. はじめに

Colmeraure の Metamorphosis Grammars[1] によって論理プログラミングの枠組と自然言語処理との整合性のよさが明確に示されて以来，文法記述を論理プログラムによっておこなう論理文法の枠組がいくつか提案されている[2]．特に Pereira が DCG (Definite Clause Grammars)[3] を提案して以来，DCG の拡張がさまざまな形でおこなわれてきた．DCG に端を発する研究には大きく2つの流れを認めることができる．その1つは DCG で記述された文法を効率のよい統語解析プログラムに変換する研究で，例として松本らの BUP[4]，SAX[5]，Nilsson らの AID[6]，林らの YAPX[7] などがある．もう1つは DCG に新しい記法を導入して，文法記述能力を強化する研究である．この2つの方向性は，DCG が文法記述形式であると同時に，容易に Prolog プログラムに変換可能であるという事実に起因している．本論文は後者の研究の流れに属し，特に構成素の移動現象に関する記述能力を強化するために DCG に新しい記法を導入することを提案する．

英語における関係節や Wh 疑問文に現れる構成素の移動現象は言語学の分野では古くから研究されてきた．最近では，文法カテゴリを素性の束と考え，素性同士の単一化といくつかの原則によって文を分析する単一化文法[17]の考え方が研究のひとつの流れになっている．単一化文法では素性の伝播の仕方を原則という形で抽象化しているため，(拡張) 文脈自由文法のように，文法カテゴリの書き換えによって文を分析する方法に比べ，一般に文法規則の数が少くてすむという特徴がある．もちろん，特定の単一化文法の枠組を論理文法上で実現することも可能である．しかし，処理をすべて素性の単一化に押し込めてしまうと，文法カテゴリの書き換えを Prolog インタプリタの推論機能を利用して容易におこなえるという論理文法の特徴が十分に活かせなくなる．本論文では，言語学の成果として提案されている特定の文法理論をそのまま計算機上で実現するのではなく，言語学の成果を参考にしながら文脈自由文法の記述力，特に構成素の移動に関する記述力を豊かにしようとする立場をとる．

構成素の移動現象を論理文法上で扱う枠組は，Pereira の XG (Extraposition Grammars)[8] や Dahl の DG (Discontinuous Grammars)[9] をはじめとして，いくつか提案されている．このうち，文法の生成能力という点では DG がもっとも強力であるが，DG を効率的に実現する方法はまだ提案されていない．XG は DCG[3] を，英語などに現れる左外置現象を扱えるように拡張したもので，文法記述能力に関しては DG のサブセットと考えることができる[10]．また，XG を Chomsky の GB 理論[11]に基づいて拡張した枠組として RLG (Restricted Logic Grammars) がある[12]．RLG では，XG で導入された手法を右方移動も扱え

† Handling Gaps in Logic Grammars by TAKENOBU TOKUNAGA, MAKOTO IWAYAMA and HOZUMI TANAKA (Department of Computer Science, Faculty of Engineering, Tokyo Institute of Technology).

†† 東京工業大学工学部情報工学科

るように拡張しているが，基本的な機構は同じである．われわれもXGと同等の文法記述能力を持つ枠組XGS(Extraposition Grammars With Slash Category)を提案し，上昇型解析器上に実現している[13),14)]．林らはXGSをさらに拡張した文法記述形式を提案している[7)]．これらはいずれも構成素が移動した後には仮想的なカテゴリがギャップとして残るという考え方に基づき，ギャップの存在を文法記述中に許している．このような文法記述では，一般に記述すべき文法規則の数が少なくてすむという利点がある．

しかしながら，これらの枠組では，移動した構成素と後に残されたギャップとの同一性を十分に記述できない場合がある．たとえば，所有格の関係代名詞を支配する構成素が移動する場合や，構成素が“and”や“but”などの等位接続詞によって接続された構造（等位構造）において構成素がギャップを含む場合に，XGやXGSのような枠組では，移動した構成素とギャップとの対応を正しく記述することが困難である．

本論文では，DCGに支配制約という概念を導入することによって，構成素の移動現象を自然に記述する枠組を提案する．この文法記述形式をLG²（Logic Grammars for handling Gap）と呼ぶ．支配制約は，XGSをはじめとするいくつかの枠組で導入されているスラッシュ記法の制約を緩めたものである．一般にスラッシュ記法は親カテゴリがあるカテゴリを支配し，そのカテゴリが必ずギャップを支配することを要請するものであるが，支配制約は親カテゴリの下に，あるカテゴリが存在することしか要請しない．支配制約とスラッシュ記法を組み合わせることによって柔軟な文法記述力が得られることを具体例によって示す．

以下，2章では，スラッシュ記法とその記述力の限界について考察し，支配制約について説明する．3章では，スラッシュ記法しか持たないXGSと対比させながら，支配制約を用いて，いくつかの英語の移動現象が自然に記述できることを示す．4章では，PereiraのDCGの実装法を拡張することによって，支配制約が計算機上に容易に実装できることを示す．最後に，5章では，本論文のまとめと今後の研究課題について述べる．

2. スラッシュ記法と支配制約

本章では，まず，スラッシュ記法を簡単に説明し，本論文で提案する支配制約について説明する．

2.1 スラッシュ記法

スラッシュ記法は，英語などに現れる左外置現象を簡潔に記述するために今野らがXGSで導入した記法である[13)]．この記法はGPSG[15)]のスラッシュ素性の考え方が基本になっている．たとえば，英語の関係節をXGSによって記述した例を文法1に示す．ただし，**relative**，**rel-pronoun**は，それぞれ関係節，関係代名詞を表す．

文法 1

s→np, vp. (1)
np→det, noun. (2)
np→det, noun(*X*), relative../np(*X*). (3)
vp→verb, np. (4)
relative→rel-pronoun, s. (5)

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

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

ことを表している．規則(3)の **relative../np(*X*)** は，名詞句をギャップとして持つ関係節を表している．文法1により，“the girl who loves me”や“the girl whom I love”などの関係節を含む名詞句を統一的に扱うことができる．また，先行詞 **noun** とスラッシュカテゴリ **np** の引数として同一の論理変数を与えることによって，ギャップとその先行詞の対応付けが容易に実現できる．この記法の利点は文法記述者がギャップが現れる位置を意識して文法規則を書く必要がない点である．スラッシュ記法を用いて文法を書き直し，文法規則の数が約3割減少した例が今野らによって報告されている[13)]．このほかにもXGSはギャップの現れる有効範囲を制御するための記法なども用意している．また，このスラッシュ記法を用いて英語の受動化，疑問文などの規則も記述できる[16)]．

2.2 スラッシュ記法の限界

XGやXGSによる関係節の記述では，関係代名詞は関係節の存在を示す単なる指標としてしか扱われていない[8),13)]．たとえば，規則(3)では先行詞 **noun** と関係節が支配するギャップは論理変数 ***X*** によって対応付けられているが，規則(5)の関係代名詞 **rel-pronoun** と，先行詞あるいはギャップは対応付けられていないので次のように関係代名詞の格が不適切な非文も受理してしまう．

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

一般に，論理文法では情報のやりとりを論理変数を介しておこなうが，論理変数の有効範囲は同一節内に限られているため，この例において先行詞，関係代名詞，ギャップの3者を正しく対応付けるためには次のいずれかの方法をとらなければならない．

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

（2） 引数を設けて情報を伝播する．

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

np→det, noun(*X*), rel-pronoun(*X*), s../np(*X*). (6)

ここで，変数 ***X*** は格に関する情報だけを持っていると仮定すると，単一化により，先行詞，関係代名詞，ギャップの格の一致は保証される．しかし，このような規則の展開をおこなうと，**relative** を規則の右辺に持つ規則数と **relative** を規則の左辺に持つ規則数の積に等しい数の文法規則が必要となる．これは文法規則数を減少させるというスラッシュ記法の利点を損なうものである．さらに，Pied-piping のように，関係代名詞を含む構造が移動する場合には，このように規則を展開することが不可能な場合がある．これらの具体的な例については3章で説明する．

一方，(2)の方法をとるならば，Pereira が DCG で左外置を扱う場合にとった方法[8]と同様に専用の引数を設けて情報を伝播しなければならない．たとえば，文法1は文法2のように修正できる．

文法 2

s→np, vp. (1)
np→det, noun. (2)
np→det, noun(*X*), relative(*X*)../np(*X*). (3)′
vp→ved, np. (4)
relative(*X*)→rel-pronoun(*X*), s. (5)′

しかしながら，このような方法をとると，どの文法規則を経由して情報が伝播するかを意識して文法を記述しなければならないので文法記述者の負担が増える．このように，移動した構成素とギャップの間の同一性を正確に記述するためには，論理変数とスラッシュ記法だけを使用したので限界があることがわかる．重要なことは規則を越えて，いかに情報を簡潔に伝播するかという点である．より柔軟な情報の受け渡しを実現するために支配制約という概念を導入する．

2.3 支配制約

支配制約はスラッシュ記法を一般化した概念である．スラッシュ記法が「スラッシュカテゴリが親カテゴリの下でギャップを支配すること」を要請するのに対し，支配制約は「あるカテゴリが親カテゴリの下に存在する」ことを要請する．例として文法1に対応する次の英語の文法規則を考えよう．

文法 3

s→np, vp. (1)
np→det, noun. (2)
np→det, noun(*X*), relative@rel-pronoun(*X*). (7)
vp→verb, np. (4)
relative→np(*X*), s//np(*X*). (8)
np→rel-pronoun. (9)

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

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

ことを表している．スラッシュ記法と異なり，カテゴリ **C** は必ずしもギャップを支配する必要はない．規則(7)の **relative@rel-pronoun(*X*)** は **relative** が **rel-pronoun** を支配していることを表している．この場合もスラッシュ記法と同様に，論理変数によって情報を受け渡すことができる．支配制約は被支配カテゴリ (**rel-pronoun**) とそれを支配するカテゴリ (**relative**) の間で情報の受け渡しをおこなう手段として使うことができる．具体的な記述例については次章で述べる．

3. 記述例

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

3.1 関係節

スラッシュ記法と支配制約を用いて，移動した構成素とギャップの対応関係も含めて関係節が正しく記述できることを示す．文法3に所有格の関係代名詞を扱うための規則(10)を追加したのが文法4である．

文法 4

s→np, vp. (1)
np→det, noun. (2)

np→det, noun(*X*), relative@rel-pronoun(*X*). (7)

vp→verb, np. (4)

relative→np(*X*), s//np(*X*). (8)

np→rel-pronoun. (9)

det→rel-pronoun. (10)

文法4を用いて所有格の関係代名詞を含む名詞句 "the girl whose eyes I loved" を解析して得られる統語木を**図1**に示す．ここで，実線は構成素の支配関係を，点線は論理変数によって構成素が対応付けられていることを表している．記号 "ε" は構成素が移動した後のギャップを表す．図1から先行詞 "girl" が **noun, relative@rel-pronoun, rel-pronoun** を経て "whose" と対応付けられていることがわかる．したがって，意味構造を構成することを考えると，"whose eyes" から "*girl's eyes*" という意味が構成できる．また，"whose eyes" から構成される **np** も **s//np** を経てギャップと正しく対応付けられている．これによって埋め込み文から "*I loved the girl's eyes*" という意味が構成できる．

一方，同じ例文を XGS の枠組を用いて解析する場合を考えよう．文法1に所有格の関係代名詞を扱うために，規則(10)，(11)，(12)を追加する．

文法 5

s→np, vp. (1)

np→det, noun. (2)

np→det, noun(*X*), relative../np(*X*). (3)

np(*X*)→det, noun(*X*), relative. (11)

vp→verb, np. (4)

relative→rel-pronoun, s. (5)

relative→np(*X*), s../np(*X*). (12)

det→rel-pronoun. (10)

規則(3)，(5)は主格，目的格の関係代名詞を扱う規則，規則(11)，(12)は所有格の関係代名詞を扱う規則であるが，両者を比べると，スラッシュカテゴリの位置が異なっている．主格，目的格の関係代名詞を扱う規則では，スラッシュカテゴリが **relative** に付いているが，所有格の関係代名詞を扱う規則では，**relative** ではなく **s** に付いている．XGS では，移動するカテゴリは，そのカテゴリを支配するカテゴリと同一規則の右辺に現れなければならない．これは，移動したカテゴリと後に残されるギャップの対応関係を論理変数を使っておこなうためである．所有格の関係代名詞では関係代名詞を含む構造が移動する．この例では移動するのは "girl's eyes" であって "girl" ではないから，規則(3)は使うことができない．文法5を用いて "the girl whose eyes I loved" を解析した結果を**図2**に示す．図2では，左に移動した "whose eyes" と埋め込み文の目的語のギャップは対応付けられているが，先行詞 "girl" と関係代名詞 "whose" との対応が付かないので "whose eyes" が "*girl's eyes*" であることが解析できない．これを解決するには，2章でも述べたように規則を展開するか，特別な引数を設けて情報を伝播しなければならない．しかし，これが規則数を増やす原因となることはすでに2章で指摘した．

変形を用いない言語理論の1つである GPSG では，関係節を **WH** 素性を用いて次のように分析している[15]．

[N′ N$_j$[S′ NP$_i${+R$_j$}[S⋯ε$_i$⋯]]]

ここで，**+R** は **[WH NP[WHMOR R]]** の省略形で，関係代名詞を表す．**NP$_i$** は関係代名詞を含む名詞句で **S** 中のギャップ ε$_i$ と同一の指標 i が付与されている．また，先行詞 **N$_j$** と **NP$_i$** 中の関係代名詞も同一の指標 j が付与されている．GPSG による分析によれば，**S** の主格，目的格の位置がギャップとなる場合は，指標 i と j が同一となる特殊な場合である．一方，所有格の関係代名詞の場合は i と j が異なる．

支配制約を使って記述した文法4と GPSG の関係

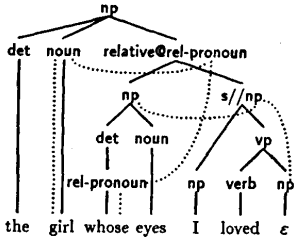

図 1 支配制約を用いた関係節の解析例
Fig. 1 An analysis of a relative clause with domination constraint.

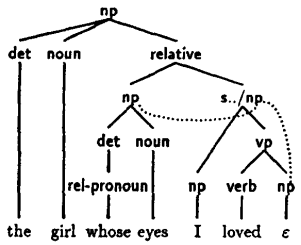

図 2 XGS による関係節の解析例
Fig. 2 An analysis of a relative clause with XGS.

節の分析を比べると規則(7)の中 **relative** が S′ に対応し，**rel-pronoun** が+**R**$_j$ に対応していることがわかる．GPSG では，素性を種々の原則によって制御しながらカテゴリ間で伝播させるが，支配制約はその伝播のショートカットを与えていると考えることができる．このように，文法5と比べると文法4の方が関係節の記述に一貫性がある．実際，次に述べる Pied-piping 現象を扱おうとすると，XGS の方法では破綻をきたす．

3.2 Pied-piping 現象

関係節を形成する時に，関係節中から節頭に移動する構成素が単に先行詞と同一の名詞句あるいは Wh を付与された名詞句だけでなく，その名詞句を支配する上位の名詞句が移動する場合がある．このような現象を Pied-piping という．この現象を説明するために Ross は Pied-piping convention という変形操作に対する制約を提案した[18]．たとえば，次の例を考えよう[18]．

(a) [$_s$ the government prescribes [$_{np_1}$ the height of [$_{np_2}$ the lettering on [$_{np_3}$ the covers of [$_{np_4}$ the reports]]]]]

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

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

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

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

例文(a)から np_1, np_2, np_3, np_4 をそれぞれ節頭に移動すると(e)，(d)，(c)，(b)の名詞句ができる．XGS の場合，文法5の規則(11)，(12)を使って任意の np_i を移動することができるが，すでに述べたようにこれらの規則では先行詞とギャップの対応付けができない．この場合，規則を展開して対応付けをしようとすると，一般には不定個の前置詞句を引き連れた名詞句が移動するので無限個の規則を用意しなければならなくなる．

支配制約を使うと，この例を文法4と規則(13)，(14)を用いて所有格の関係代名詞とまったく同様に解析できる．

np→det, noun, pp. (13)

pp→p, np (14)

名詞句(c)の解析例を**図3**に示す．先行詞 "reports" と関係代名詞 "which" の対応，移動した名詞句 "the covers of which" とギャップの対応がそれぞれとれていることがわかる．

3.3 等位構造

"and" や "but" などの等位接続詞によって任意の構成素が接続される構造を等位構造という．もちろん，等位接続される構成素の組合せには制約があるが[19]，ここでは言語学的な詳細には立ち入らない．

等位構造は自然言語解析において統語木のあいまい性やスコープのあいまい性など多くの問題を引き起こす．これまでの等位構造の扱いに関する研究は，等位構造の記述を文法記述者に透明にするか，文法記述者の手にゆだねるかによって大きく2種類に分類できる．前者のアプローチとしては，Dahl と McCord の MSG[20]，Sedogbo のメタ文法[21]，Fong と Berwick の RPM[22]，武舎の Predictive Analyzer[23]，Hirschman の Meta-Restriction Grammars (MRG)[24] などがある．これらの枠組では，等位構造に関する規則を文法記述者が記述する必要がなく，等位構造は解析器によって解析中に動的に，あるいはコンパイラによって事前に処理される．このアプローチには等位構造に関する記述が不要であるという利点はあるが，等位構造の処理に変更を加えることが困難であるという欠点がある．

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

もっとも単純な等位構造は，

Tom loves Mary and John loves Jane.

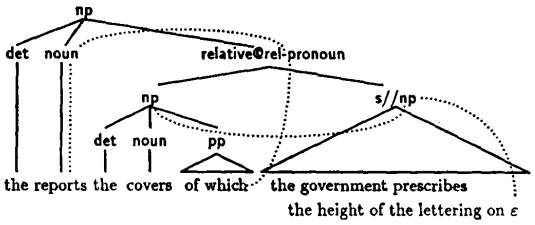

図3 Pied-piping の解析例
Fig. 3 An analysis of Pied-piping.

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

Mary saw and John heard the train.

(a) [s[s[np Mary][vp[v saw][np the train]]] and [s[np John][vp[v heard][np the train]]]]

(b) [s[s[s[np Mary][vp[v saw][np ε]]] and [s[np John][vp[v heard][np ε]]]][np the train]]

伝統的な変形文法では，このような文は，(a)の深層構造から Right Node Raising という変形を適用して(b)の表層構造が派生するという分析をおこない，ギャップを含む文が等位接続されることはないとしている．しかし，この分析にもいくつかの問題点が指摘されている[25]．以下の議論では，伝統的な変形文法の分析とは異なる木村の空照応分析[25]に近い方法を採用する．

XGS で，この例文を解析するためには，たとえば，次の文法が必要となる．

文法 6

s→sd. (15)

s→sd../np([case:obj]),[and],s. (16)

sd→np,vp. (17)

np→det,noun. (2)

vp→verb,np. (4)

スラッシュカテゴリ **np** の引数として **[case:obj]** を渡している点に注意して欲しい．これにより，**sd** の下でギャップとなっている **np** は目的格の素性を持たなければならないことを要請している．しかしながら，この規則では等位接続される最初の **sd** の目的語と次の **s** の目的語が同一であることを記述していない．XGS では，この規則中で2番目の **s** が支配する構成素に関する情報を参照する手段がないため，この記述は不可能である．したがって，"saw" の目的語が "the train" であるということが解析できない．

一方，支配制約を用いると規則(16)の代わりに規則(18)のように記述することができる．

文法 7

s→sd. (15)

s→sd//np([case:obj|X]),
[and],s@np([case:obj|X]). (18)

sd→np,vp. (17)

np→det,noun. (2)

vp→verb,np. (4)

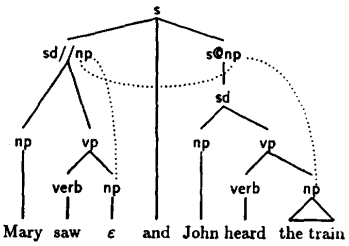

図 4 ギャップを含む等位構造の解析例
Fig. 4 An analysis of a coordinate structure including a gap.

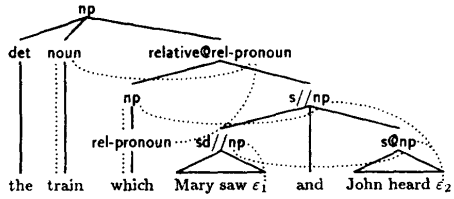

図 5 等位構造を含む関係節の解析例
Fig. 5 An analysis of a relative clause including a coordinate structure.

規則(18)は右辺の **sd** の下でギャップを支配している **np** の情報と，**s** が支配する **np** の情報がともに目的格の素性を持ち，かつ単一化可能であることを要請している．したがって，意味的にも **sd** と **s** の目的語が同一であることが正しく解析できる．等位接続詞が2つ以上ある場合も規則(18)を再帰的に適用することによって同様に解析できる．文法7による解析例を**図 4** に示す．

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

4. 実　　装

本章では2章で述べたスラッシュ記法と支配制約の1つの実装法について述べる．ここでは，多くの Prolog インタプリタに組み込まれている DCG トランスレータの存在を仮定し，スラッシュ記法と支配制約を含む文法規則を，トランスレータにより DCG に変

換するという方法をとる．DCG に変換された文法規則はスラッシュカテゴリや被支配カテゴリを管理する補助述語とともに Pereira の手法[3]により Prolog プログラムに変換され，Prolog インタプリタで直接実行される．

4.1 トランスレータ

スラッシュ記法と支配制約を含む文法規則を DCG に変換するトランスレータは2引数述語 **trans** として実現されている．述語 **trans** の完全な定義は文献 26) を参照されたい．

文法の変換例を**図 6** に示す．図 6 (a) は英語の等位構造を解析するための文法で，3章で例として使用した文法 7 を修正したものである．この文法により "Mary saw and John heard the train" が解析できる．各文法カテゴリは2つの引数を持ち，第1引数は格に関する情報を，第2引数は統語木の情報を保持する．解析結果として統語木が第2引数にリストの形で得られる．述語 **unify** は CIL[27] で採用されている部分項の単一化をおこなう述語である．部分項は素性と値の対の集合として考えることができ，部分項の単一化は直観的には集合の和集合を計算する操作となる．以下，部分項の単一化を Prolog の単一化と区別するために拡張単一化とよぶ．ここでは最後尾の要素として変数を持つ Prolog のリストを用いて部分項を実現している．

```
(a-1)  :- op(680,xfy,':').
(a-2)  s(_,s(T)) →
         sd(_,T).
(a-3)  s(_,s(T1,and,T2)) →
         sd(_,T1)//np([case:obj|N],T),
         [and],
         s(_,T2)@np([case:obj|N],T).

(a-4)  sd(_,sd(T1,T2)) →
         np(N,T1),
         {unify(N,[case:sbj|_])},
         vp(_,T2).

(a-6)  np(N,np(T)) →
         n(N,T).

(a-7)  vp(_,vp(T1,T2)) →
         v(_,T1),
         np(N,T2),
         {unify(N,[case:obj|_])}.
(a-8)  n(_,n(john)) → [john].
(a-9)  n(_,n(mary)) → [mary].
(a-10) n(_,n(train)) → [the,train].
(a-11) v(_,v(saw)) → [saw].
(a-12) v(_,v(heard)) → [heard].
```

(a) 変換前
(a) Before translation.

```
(b-1)  :- op(680,xfy,':').
(b-2)  s(_,s(T),A,X0,X1) →
         sd(_,T,A,X0,X1).
(b-3)  s(_,s(T1,and,T2),A,X0,X2) →
         sd(_,T1,A,[np([case:obj|N],T)|X0],X1),
         {found_gap([np([case:obj|N],T)|X0],X1)},
         [and],
         s(_,T2,[a(Found,np([case:obj|N],T))|A],X1,X2),
         {found_dom([a(Found,np([case:obj|N],T))|A])}.
(b-4)  sd(_,sd(T1,T2),A,X0,X2) →
         np(N,T1,A,X0,X1),
         {unify(N,[case:sbj|_])},
         vp(_,T2,A,X1,X2).
(b-5)  np(N,T,A,[np(N1,T1)|X0],X0) → [ ]
         {unify_args([N,T],[N1,T1])},
         {found(np(N,T),A)}.
(b-6)  np(N,np(T),A,X0,X1) →
         n(N,T,A,X0,X1),
         {found(np(N,np(T)),A)}.
(b-7)  vp(_,vp(T1,T2),A,X0,X2) →
         v(_,T1,A,X0,X1),
         np(N,T2,A,X1,X2),
         {unify(N,[case:obj|_])}.
(b-8)  n(_,n(john),_,X,X) → [john].
(b-9)  n(_,n(mary),_,X,X) → [mary].
(b-10) n(_,n(train),_,X,X) → [the,train].
(b-11) v(_,v(saw),_,X,X) → [saw].
(b-12) v(_,v(heard),_,X,X) → [heard].
```

(b) 変換後
(b) After translation.

図 6 文法の変換例
Fig. 6 An example of the grammar translation.

変換により文法中の Prolog プログラム，および規則中の "{" と "}" で囲まれた補強項はそのまま出力に複写する (図 6 (b-4)，(b-7) の補強項参照)．各文法カテゴリには，2つのXリスト[8),13)]と1つのAリストを引数として付加する．Xリスト，Aリストは，それぞれスラッシュカテゴリの情報と被支配カテゴリの情報を保持する引数である．以下では，文法規則の変換をスラッシュ記法の変換，支配制約の変換を中心に説明する．

4.2 スラッシュ記法の変換

スラッシュ記法の変換は，Pereira[8]，今野[13]の手法とほぼ同じである．各カテゴリに付加する2つのXリストはスタックで，それぞれ，そのカテゴリを解析する前（入力Xリスト）と解析した後（出力Xリスト）のスタックの状態を保持している．スラッシュカテゴリを含むカテゴリを変換する場合は，入力Xリストにスラッシュカテゴリを引数とともにプッシュする（図 6 (b-3) の **sd** を参照)．スラッシュカテゴリをXリストからポップするのはスラッシュカテゴリと同じカテゴリを解析する時であるから，文法規則中にスラッシュカテゴリとして出現するカテゴリについて次の規則を追加する．

Cat(Args1, A, [Cat(Args2)|X0], X0)→[ ]
{unify_args(Args1, Args2)}. (19)

この規則は ε 規則であり，入力文字列中にギャップが存在する時に使用する．図 6 の例では (b-5) がこの規則に相当する．ただし，この例では，**np** がスラッシュカテゴリであると同時に被支配カテゴリでもあるため，次節で述べる述語 **found** が追加されている．規則 (19) において，**Cat** はスラッシュカテゴリ名，**Args2** はスラッシュカテゴリが持つ情報，**A** は支配制約のためのAリストである．Aリストについては次節で述べる．下線を引いた部分がXリストである．入力Xリストの先頭（スタックトップ）にはスラッシュカテゴリ **Cat** が存在し，出力Xリストは入力Xリストの先頭を取り除いた残り部分になっている．すなわ

ち，この規則を使ってカテゴリ **Cat** の解析をおこなうと，入力文字列を消費するかわりに，入力Xリストの先頭にカテゴリ **Cat** があれば，これをポップし，残りの X リストを出力 X リストとして返すことになる．ただし，この規則を使う時点で検出したギャップの持つ情報（**Args1**）はスラッシュカテゴリに引数として指定された情報（**Args2**）と矛盾してはならない．この検査をおこなうために述語 **unify_args** を用いて **Args1** と **Args2** を拡張単一化する．また，スラッシュカテゴリを持つカテゴリを解析した直後には，入力文字列中でギャップを実際に検出している必要がある．これは，出力Xリスト中のスラッシュカテゴリがポップされていることを検査することで実現できる．述語 **found_gap** はこの検査をおこなう述語で，スラッシュカテゴリを持つカテゴリの解析後に補強項として挿入する．付録に述語 **found_gap**, **unify_args** の定義を示す．

4.3 支配制約の変換

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

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

  a(Found, Cat(Args))

  これをA構造と呼ぶ．第1引数の **Found** は被支配カテゴリを検出するとアトム **found** に束縛される変数である．支配制約を持つカテゴリの解析が終了した時点で，この変数の束縛状態を調べることにより支配制約を満足しているかどうかを検査する．この検査をおこなうのが以下で説明する述語 **found_dom** である．**Cat** は被支配カテゴリ名，**Args** はその引数である．例として図6（b-3）の右辺の **s** の変換結果を参照されたい．
- 被支配カテゴリとして文法中に現れるカテゴリを左辺に持つ文法規則の末尾には，述語 **found** を補強項として挿入する．述語 **found** は，第1引数に解析によって完成したカテゴリ，第2引数にAリストをとり，第1引数のカテゴリと引数を含めて拡張単一化できるカテゴリを含むA構造をAリストから探し，そのA構造の第1引数の変数をアトム **found** に束縛する．この変数の束縛によって，被支配カテゴリが検出されたことをAリスト中に記録する．付録に述語 **found** の定義を示す．
- 支配制約を持つカテゴリの直後には補強項として述語 **found_dom** を挿入する．**found_dom** は，直前の支配制約を持つカテゴリのAリストに挿入したA構造の第1引数がアトム（**found**）に束縛されているかどうか調べる．ここで，A構造の第1引数が変数であれば，支配制約を満たしていないことになるので **found_dom** は失敗する．もし支配制約を満たしていれば，その被支配カテゴリと残りのAリストに対して述語 **found** を再帰的に適用する．これによって，複数の被支配カテゴリを同一のカテゴリと対応付けることが可能となる．述語 **found_dom** の定義を付録に示す．

4.4 解析実行例

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

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

このゴールは（b-3）の左辺と照合し，右辺のゴールを呼び出す．（b-4）により等位接続された前半の文 “mary saw ε” を解析する．この時，主語の **np** の解析が（b-6）により成功すると **found** が呼び出されるが，この段階ではAリストは空なので **found** は単に成功する．その後，補強項により **np** の第1引数には主格の素性が追加される．次に，（b-7）により動詞句の解析をおこなうが，目的語はギャップとなっているので，入力文字列を消費するかわりに（b-3）でXリストにプッシュされたスラッシュカテゴリの **np** を（b-5）によりポップし **vp** の解析を成功させる．この時に，（b-5）の **unify_args** によってギャップの格情報（**N**）とスラッシュカテゴリの格情報（**N1**）を拡張単一化することにより，このギャップはスラッシュカテゴリで指定された素性 **case:obj** を持つことになる．（b-7）の補強項 **unify** によって目的語（ギャップ）の格情報が **case:obj** を持つことを検証するが，これは今述べた理由から成功する．

以上で，（b-3）の右辺の **sd** の解析が終了したので

補強項の **found_gap** を実行するが，目的語の解析でスラッシュカテゴリをポップしたので，これは成功する．次に後半の **s** の解析をおこなう．この例では2文が等位接続されているので，今度は（b-2）を使う．引続き（b-4）が呼び出され，（b-6）を用いて主語の **np** の解析をする．（b-6）のボディの最後で，**found** を呼び出し，Aリストの中のA構造の **np** と，ここで完成した **np**("john") の格情報を拡張単一化する．これで，この主語の **np** は目的格の素性を持つことになる．しかし，（b-4）の補強項では，主語の **np** は主格の素性を持つことを要請しているので，補強項の失敗によりバックトラックを起こす．結局，入力Aリストをそのまま出力Aリストに返すことにより **found** は成功する．

次に（b-4）の動詞句の解析に移る．動詞の解析を終ると（b-6）により目的語の **np** の解析をおこなう．ここで，先ほどと同様に **found** がAリスト中のA構造の **np** と目的語の **np** を拡張単一化し，A構造の第1引数をアトムに束縛する．次に補強項を実行するが，今度は格に関する素性が一致し，**vp** の解析，そして **s** の解析が成功する．次に（b-3）の最後の補強項 **found_dom** を実行する．（b-6）でAリスト中のAの構造の第1引数はアトムに束縛されているので，これは成功する．以上で解析が成功し，以下のような統語木の情報が得られる．

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

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

5. おわりに

本論文では，論理文法上でギャップを扱うために，支配制約という新しい概念を導入した文法記述形式 LG² を提案し，いくつかの英語の移動現象が自然に記述できることを示した．また，支配制約が Pereira の DCG の実装法を拡張することによって Prolog 上で容易に実現できることも述べた．

Pereira の DCG はすでに文脈自由文法の枠組をはみだしており，文法の生成能力という観点からは，本論文で提案した LG² は DCG を超えるものではない．しかしながら，Dahl も指摘しているように，DCG が文脈自由文法からはみだしているのは，補強項という手続き的な部分においてであり，これを宣言的な文法記述に反映させることは，文法記述の容易さという観点からは重要である[28]．支配制約の考え方はこの線に沿うものであり，移動した構成素とそのギャップを正しく対応付ける規則を宣言的に記述できる能力を与えている．

自然言語における移動現象を論理文法の枠組で扱おうとする最近の研究として林の YAPX[7]，RCSG[29] や Dahl の SDG[28] などがある．YAPX の文法記述形式は XGS[13] の拡張であり，RCSG は YAPX の文法記述形式に文脈依存性を導入したものである．林は RCSG で記述した文法を使って高速に統語解析をおこなうアルゴリズムも提案している．RCSG では，文法規則の右辺の各位置に指標を割り当て，特定の規則を特定の指標位置でのみ解析に使うことにより文脈依存性を実現している．このため高速な解析が可能となるが，どの位置でどの規則を適用するかは文法記述者にゆだねられている．これは文法記述という観点からは好ましくない．

一方，Dahl の SDG は DG[9] に制限を加えたものである．DG では，書き換え規則に関する制限がほとんどないので，文法の生成能力が強力である反面，文法記述の明確なガイドラインがなくては文法の見通しが悪くなり，効率も悪いという欠点があった．SDG は DG に制限を加え，文法記述のガイドラインとして Chomsky の GB 理論[11]を採用しようとしている．また，実際に，GB 理論の下接の条件を SDG の枠組で Prolog 上に実現している[30]．SDG の枠組で，GB 理論の各原理がどの程度実現できるかは，まだ明らかではない．

これらの研究に対して，本論文で提案した支配制約は，構成素の持つ情報の流れに重点を置いている．われわれの最終的な目標は，入力文からその文の何らかの意味構造を抽出することであり，統語解析はその手段にすぎない．単に構成素の移動を文法上で記述できるだけでは不十分であり，移動した構成素と後に残されたギャップの同一性を宣言的に正しく記述できることが重要である．支配制約はこの問題に対する1つの解である．

本論文の目的は，文法記述における支配制約の有効性を示すことなので，4章で述べた実装法は，必ずしも最良のものとはいえない．この実装法は，下降型深さ優先解析器なので，Pereira の DCG の実装法と同様に左再帰規則が扱えないという問題がある．また，

BUP[10] や LangLAB[14] が採用しているような再計算を回避する機構も持っていない．今後，支配制約を効率のよい解析アルゴリズム上に実現し，大規模な文法の記述により，その有効性をさらに検討する必要がある．

謝辞 論文の初版の誤りを指摘し，多くの有益なコメントをくださいました査読者の方々に感謝いたします．また，Chomsky の GB 理論についてご教授くださいました東京工業大学の長谷川宏講師に感謝いたします．

参考文献

1) Colmeraure, A.: Metamorphosis Grammar, *Natural Language Communication with Computers*, pp. 133-190, Springer-Verlag (1978).
2) Abramson, H. and Dahl, V.: *Logic Grammars*, Springer-Verlag (1989).
3) Pereira, F.C.N. and Warren, D.H.D.: Definite Clause Grammars for Language Analysis—A Survey of the Formalism and a Comparison with Augmented Transition Networks, *Artif. Intell.*, Vol. 13, No. 3, pp. 231-278 (1980).
4) Matsumoto, Y., Tanaka, H., Hirakawa, H., Miyoshi, H. and Yasukawa, H.: BUP: A Bottom-up Parser Embedded in Prolog, *New Generation Computing*, Vol. 1, No. 2, pp. 145-158 (1983).
5) 松本裕治，杉村領一：論理型言語に基づく構文解析システム SAX，コンピュータソフトウェア，Vol. 3, No. 4, pp. 4-11 (1986).
6) Nilsson, U.: AID: An Alternative Implementation of DCGs, *New Generation Computing*, Vol. 4, No. 4, pp. 383-399 (1986).
7) 林 達也：拡張 CFG とその構文解析法 YAPX について，情報処理学会論文誌，Vol. 29, No. 5, pp. 480-487 (1988).
8) Pereira, F.C.N.: Extraposition Grammars, *Am. J. Comput. Linguist.*, Vol. 7, No. 4, pp. 243-256 (1981).
9) Dahl, V. and Saint-Dizier, P.: Constrained Discontinuous Grammars: A Linguistically Motivated Tool for Processing Language, Technical Report, INRIA (1986).
10) Matsumoto, Y.: Natural Language Parsing Systems Based on Logic Programming, PhD thesis, Kyoto University (1989).
11) Sells, P.: Lectures on Contemporary Syntactic Theories, CSLI, Stanford University (1985).
12) Stabler, E.P., Jr.: Restricting Logic Grammars with Government-Binding Theory, *Computational Linguistics*, Vol. 13, No. 1-2, pp. 1-10 (1987).
13) 今野 聡，田中穂積：左外置を考慮したボトムアップ構文解析，コンピュータソフトウエア，Vol. 3, No. 2, pp. 115-125 (1986).
14) 徳永健伸，岩山 真，田中穂積，上脇 正：自然言語解析システム LangLAB，情報処理学会論文誌，Vol. 29, No. 7, pp. 703-711 (1988).
15) Gazdar, G. and Pullum, A.F.: Generalized Phrase Structure Grammar: A Theoretical Synopsis, Indiana University Linguistics Club (1982).
16) 岩山 真，徳永健伸，Quek Chee Huei，田中穂積：自然言語処理のための英語文法，第 37 回情報処理学会全国大会論文集，pp. 1100-1101 (1988).
17) Shieber, S.M.: An Introduction to Unification-based Approaches to Grammar, CSLI, Stanford University (1986).
18) 大塚高信，中島文雄：新英語学辞典，研究社 (1982).
19) Sag, I., Gazdar, G., Wasow, T. and Weisler, S.: Coordination and How to Distinguish Categories, Technical Report CSLI-84-3, CSLI (1984).
20) Dahl, V. and McCord, M.C.: Treating Coordination in Logic Grammars, *Am. J. Comput. Linguist.*, Vol. 9, No. 2, pp. 69-91 (1983).
21) Sedogbo, C.: A Meta Grammar for Handling Coordination in Logic Grammars, *Proceedings of the Conference on Natural Language Understanding and Logic Programming*, pp. 137-150 (1984).
22) Fong, S. and Berwick, R.C.: New Approach to Parsing Conjunctions Using Prolog, *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 118-126 (1985).
23) Musha, H.: A New Predictive Analyzer of English, *Proceedings of the International Conference on Computational Linguistics*, pp. 470-472 (1986).
24) Hirschman, L.: Conjunction in Meta-restriction Grammar, *The Journal of Logic Programming*, Vol. 3, No. 4, pp. 299-328 (1986).
25) Kimura, N.: Right Node Raising: A Null Anaphor Analysis, *English Linguistics*, Vol. 3, pp. 118-133 (1986).
26) 徳永健伸，岩山 真，田中穂積：論理文法におけるギャップの扱い，情報処理学会自然言語処理研究会，NL 76-3 (1990).
27) Mukai, K. and Yasukawa, H.: Complex Indeterminates in Prolog and Its Application to Discourse Models, *New Generation Computing*, Vol. 3, No. 4, pp. 441-466 (1985).
28) Dahl, V.: Discontinuous Grammars, Technical Report TR 88-26, CSS/LCCR (1988).
29) 林 達也，宮 俊司，坂巻利哉，吉田健一：横

型トップダウン文解析システムの実現と評価，情報処理学会自然言語処理研究会，NL 74-9 (1989).
30) Dahl, V. and Massicotte, P.: Meta-programming for Discontinuous Grammars, Technical Report TR 88-25, CSS/LCCR (1988).

付録 補助述語の定義

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

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

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

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

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

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

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

(平成2年2月13日受付)
(平成3年9月12日採録)

徳永 健伸（正会員）

1961年生．1983年東京工業大学工学部情報工学科卒業．1985年同大学院理工学研究科修士課程修了．同年(株)三菱総合研究所入社．1986年東京工業大学大学院博士課程入学．1987年より同大学工学部情報工学科助手．工学博士．自然言語処理に関する研究に従事．認知科学会，人工知能学会，計量国語学会各会員．

岩山 真（正会員）

1964年生．1987年東京工業大学工学部情報工学科卒業．1989年同大学院修士課程修了．現在，同大学院博士後期課程在学中．知識情報処理全般に興味を持つ．人工知能学会，日本ソフトウェア科学会，AAAI 各会員．

田中 穂積（正会員）

昭和39年東京工業大学理工学部制御工学科卒業．昭和41年同大学修士課程修了．同年電気試験所(現，電子技術総合研究所)入所．昭和58年東京工業大学工学部情報工学科助教授．昭和61年同大学教授となり現在に至る．工学博士．人工知能，自然言語処理の研究に従事．電子情報通信学会，認知科学会，日本ソフトウェア科学会，人工知能学会，計量国語学会各会員．