自然言語処理 67－2
（1988. 7. 22）

論理型文法における制約解析

杉村領一
(財)新世代コンピュータ技術開発機構

本論文は、論理型文法によって示される制約と、談話構造の基本的な制約を、ロジックにより関係づける。これにより、自然言語を解析する際に生じる文の複数の解釈を、談話において処理する事が可能になる。つまり、文解釈で成り立つ制約を、制約論理プログラミングによって談話処理へ伝搬し解く。制約は命題論理式として表現される。文解釈にあい昧さがある場合、伝搬された命題論理式の中に、解釈の一意に決定されない命題論理変数が存在する。また、多環境による環境共有の問題も、本論の枠組みによってとらえ直され、その一部は解決される。また、談話処理における本モデルの問題点や、制約生成系としての解析システムを具体的に構築する際の問題点についても考察を行う。

Constraint Analysis of Logic Grammars

Ryôichi Sugimura
Institute for New Generation Computer Technology (ICOT)
4-28, Mita 1-chome, Minato-ku, Tokyo 108 Japan

This paper presents a tiny model to connect constraints on logic grammars like DCG with basic constraints in discourse. In this model, we can handle disjunctive interpretations derived from natural language analysis on logic grammars in constraint logic programming. To put in the concrete, the model presented here enables us to obtain logical core constraints between disjunctive interpretations explicitly in propositional logic. Consequently, the major advantage of our method is that we can propagate logical constraints between disjunctive interpretations on discourse analysis. The computational complexity of disambiguation is improved from Π number of disjunctions to Σ number of disjunctions. Practically, a structure which has partially solved with logical variables can easily be shared with disjunctive interpretations. Problems of this model in discourse analysis and problems in designing a parser as generator of constraints are also discussed.

1. はじめに

自然言語の解析において、いわゆる文脈(談話)情報を用いずに、一文の解釈を一意に決定することは困難である。これは、基本的に、一つの文が種々の談話状況で異なる解釈を持つ[Barwise 83]ためであり、当然である。しかし、一文の処理と談話処理を結び付ける事も難かしい[向井 88]。例えば、単純に文脈依存言語により文脈処理規則を記述すると、この規則に従った解析処理が停止しない事は十分に予想できる。このため、従来は文解釈において談話の制約や情報を用いず、種々のヒューリスティクスにより一文の解釈を決定している[Nakamura 86]。

本論は、これらヒューリスティクスや談話構造を、曖昧さの処理を通じて、談話において研究するための基本的な枠組みを提供する。つまり、DCG等の論理型文法による一文の処理と談話処理を、曖昧さの処理を通じて結び付ける一手法を提案する[sugimura 87]。結び付ける道具は制約論理プログラミング[Jaffar 87][Dincbas 86]である。文解析の複数の解釈間に成り立つ制約は、制約論理プログラミングによって談話処理へ伝搬される。伝搬された制約は、談話における制約と共に順次蓄積され、解釈を順次決定する事を可能にする。

一般に、自然言語処理において扱える情報は部分的な情報であるが、部分情報を談話における制約として用いて、順次解釈を絞りこむ事を制約論理プログラミングは可能にする。また、制約の計算は、例えば、本論で用いる手法のように、命題論理の制約を正規形[向井 88]に変換する手続が必ず停止する事が保証されているものがある。このため、制約論理プログラミングは今後の有望な談話処理のための計算メカニズムになるだろう。

本論は以下の5つの部分から構成される。

まず、2.で論理型文法の制約と談話構造の基本的な制約を定式化し、関係づける。

3.では、一文の解析処理の役割を示す。具体的には、同一の部分木を重複して作らない構文解析を行なう。これは、計算機の中で、同一の部分木を idempotent law($T \wedge T \Leftrightarrow T$) により変換する事を避けたいためで、計算効率を得るために必要である。

解析では、複数の解釈の談話における制約を生成する。つまり、解析処理を制約生成系とみなす。この考え方は「文の意味は、状況と状況の関係(制約)である」とする状況意味論の立場に適合する。

現在、構文解析にはSAX[Matsumoto 87]を用いているが、同一ゴールの重複計算の無い解析が、インプリメントにおいて、解釈間の制約を有効にするための必要条件である。

4.では、命題論理式の制約処理機構について述べる。制約の解決方法としては2つの手法を示す。

第一の手法は、遅延実行制御による制約の受動的な解法である。本手法では、制約式の書き換えを能動的には行わない。

第二の手法は、能動的に制約式を書き換える。制約式の論理値が決定できない場合には、この制約を文脈処理へ伝搬する。第二の手法は、項変換プログラムで実行可能である。この手法によれば、多環境における論理変数共有の問題[Matsumoto 87]は部分的に解決される。計算量等についても考察する。

5.では、談話処理について述べる。具体的には、一文の解析において得られた複数の解釈間の制約の談話における意味を定式化し、どの様な制約が談話における制約として有効かを考察する。

6.では、制約生成系として、具体的にどのようなメカニズムが解析処理に必要かを考察する。特に、並列実行の可能な構文解析システムSAX(AX)を中心に問題点を整理する。

2.文文法と談話の関係

2.1.論理型文法

DCG等の論理型文法では、解析は、文法として与えられた推論規則と入力文から、述語"文"を証明する証明過程と考える事が出来る。解析過程で得られた種々の部分木は、入力文と文法から得られる定理である。先ず、推論規則を単純化して、以下の例のようになっているとする。

$$h(Sh,A,An) \Leftarrow b1(S1,A,A1) \wedge \ldots \wedge bn(Sn,An\text{-}1,An) \wedge rel(Sh,(S1,\ldots,Sn)) \quad (1)(2)$$

$$bj(Sj,A,B) \Leftarrow c(A,B) \wedge B = A + 1. \quad (3)$$

(1)(2)は文法として予め与えられている推論規則である。論理変数のS_h, $S_j(1 \leqq j \leqq n)$は、それぞれの述語h, bj$((1 \leqq j \leqq n)$を真とする意味情報がアサインされる変数である。論理変数のA, $A_j(1 \leqq j \leqq n)$は、それぞれの述語h, bj$((1 \leqq j \leqq n)$を真とする文中の単語並びの範囲を示す変数である。ただし、各単語には自然数が文の左端の単語から順に対応させられているとする。例えば、$h(S_h,A,A_n)$は、位置AからA_nの手前までをhとして解釈できる事を示す。

(2)の述語$rel(S_h,(S_1,\ldots,S_n))$は、引数$S_1,\ldots,S_n$と、$S_h$の間に何らかの関係が成り立つ場合に真となる。述語relの詳細は、6で検討する。

(3)の$b_j(S_j,A,B)$は辞書として与えられる述語である。$c(A,B) \wedge B = A + 1$は、入力文によって、その真偽が決定される。cは表層を示す。

2.2. 複数の解釈と文脈

文解析とは、上記の推論規則(文法と辞書)を用いて入力文(長さをLとする)で与えられた解釈から証明できる、述語"文(S文,1,L+1)"を得る事だと考えよう。一般に、構文解析処理においては、入力文1つに対して、複数の述語"文"が証明できる。複数の解釈を文脈で扱うために、述語"文"については、上記の文法へ以下のように下線で示した述語を追加する。

$$文(S_{文},A,A_n) \wedge \underline{S_{文} \in I} \Leftarrow b1(S_1,A,A_1) \wedge \ldots \wedge bn(S_n,A_{n-1},A_n) \wedge rel(S_{文},(S_1,\ldots,S_n))$$

Iは述語"文"を真とする意味情報を要素とする集合である。

次に、意味情報を談話を考慮したものに拡張する。文内でも、例えば敬語などのように、談話状況が異なる[杉村 86]。そして、文内での談話の切り替わりは、形態的に分かる場合が多い。例えば、文「『私』が正しいと彼が言ったのを、私は聞いた」では、私と『私』とは別人であり、私と『私』の意味が成り立つ談話状況は異なっている。そこで、意味情報を談話を示すパラメータと意味情報の組で示す。例えば、形態的に明らかに談話が変わる場合には、文法規則は以下の様に示される。

$$h((DS_b,S_h),A,A_2) \Leftarrow b1((DS_a,S_1),A,A_1) \wedge b2((DS_b,S_2),A_1,A_2) \wedge rel((DS_b,S_h),((DS_a,S_1),(DS_b,S_2)))$$

上記の例では、述語b1とb2の間で談話状況がDS_aからDS_bへ切り替わり、全体としては、談話状況DS_bになる事を示している。

また、辞書規則については、発話において、cなる単語があり、かつ、この状況から"c"の意味Sjが得られるので、以下のように規則を記述する。述語⊨は、発話状況DUにおいてcなる単語が発声された事を意味する。

$$h((DS,S_j),A,B) \wedge rel((DS,S_j),(DU,c)) \wedge DU \models c \Leftarrow c(A,B) \wedge B = A + 1.$$

以上の拡張の結果、述語"文"は以下のように定義される。

$$文((DS_{文},S_{文}),A,A_n) \wedge (\underline{DS_{文},S_{文}}) \in I \Leftarrow b1((DS_1,S_1),A,A_1) \wedge .. \wedge bn((DS_n,S_n),A_{n-1},A_n) \wedge rel(S_{文},(S_1,...,S_n))$$

2.3. 談話の制約

2.1,2.2で示した規則に従い文の解釈を行ったとする。すると、述語$(\underline{DS_{文},S_{文}}) \in I$に付いては、以下の制約が成立する。

公理D1

$$\exists DS \models S \text{ where } (DS, S) \underline{\in I}$$

この制約は非常に重要である。つまり、文には最低一つの意味があるということを、この制約は述べている。この制約は、例えば集合Iに2つの要素がある場合には、以下の様に書き下せる。

$$(DS \models S_1) \text{ or } (DS \models S_2)$$

論理和を"or"で示したのは、文法上の制約と談話上の制約を識別するためである。

なお、本公理は、文解析が失敗することは考えにいれていない。解析が失敗した場合までも含めたモデルの作成は今後の課題である。

次に、本論で重要な位置をしめる推論規則を示す。

公理D2

$$(DS \models S_h) \Rightarrow (DS_1 \models S_1) \text{ and,...,and } (DS_n \models S_n) \text{ where } rel(S_h, ((DS_1,S_1), ... ,(DS_n,S_n)))$$

公理D2は、ある状況DSである意味が成り立ち、この意味が他の状況で成り立つ部分意味から構成される場合、この部分意味もある状況で成り立つ事を示す。直感的には、文の意味が談話状況で成立するかどうかは、その部分意味がある状況で成立するかどうかを調べればよい事を示している。

ただし、この公理の意図は、述語論理の域をはみ出ている。というのは、例えば、動詞「走る」が一つの引数「誰か」を持つ述語だとすると、「誰か」が「走る」ことが成立するかを決定するには、引数の決定されない(つまり、適当な引数への値の割り当てが決まっていない)述語「走る」が成り立つかどうかを調べねばならず、この推論を、適当な引数への値の割り当てが決まっていなくても行えるという立場であるためである。これは、状況理論の基本的な立場である。

以上の公理D1,D2と解析のための推論規則を用いて如何に、文が処理されるかを、3以降で述べる。

3.構文解析

構文解析は同一の部分木を重複して生成しない方法で行う。つまり、同一の定理を複数回証明しない事が望ましい。これは、計算機上で同一の述語が複数真であると、無意味な曖昧さが述語"文"について作られるためである。また、此れをidemporent lawにより宿退する手間も無駄である。まず、以下のようにSTを構文解析中にできた部分木(定理)の集合とする。

$$ST = \{Sti \mid Stiはiをインデクスとする部分木\}$$

必要な条件は以下の通りである。

$$\forall i \forall j(Sti = Stj \Rightarrow i = j) \quad \text{where } Sti \in ST \wedge Stj \in ST$$

同一の部分木を重複して生成しない構文解析は、論理型構文解析システムSAX、BUP[Matsumoto 84]、LangLAB[今野 86]等で既に実現されている。

SAXでは、DCGを並列処理可能なPrologのプログラムに変換する際に、同一の部分木を複数の解釈間で共有するように変換が行われる。

BUP、LangLABでは、生成に成功した部分木をPrologのデータベースを用いて記憶しておき再計算を行わない。

4. 談話における制約処理

以上、3で示した条件を満たす構文解析手法により、例えば英文、"A saw B on C"を、2で示した形式の文法と辞書を用いて解析し、図1の様な解析結果を得たとする。図中では、構文解析で成り立つ述語を、(XY)で以下の規則を用いて簡略に示す。

$$(DX \models Y) \Leftrightarrow YX$$

例えば、$(DS \models S_{16})$は$S_{16}S$と示す。

結果としては、以下の述語が公理D1、公理D2により、文脈内部で成立つ。

公理D1より

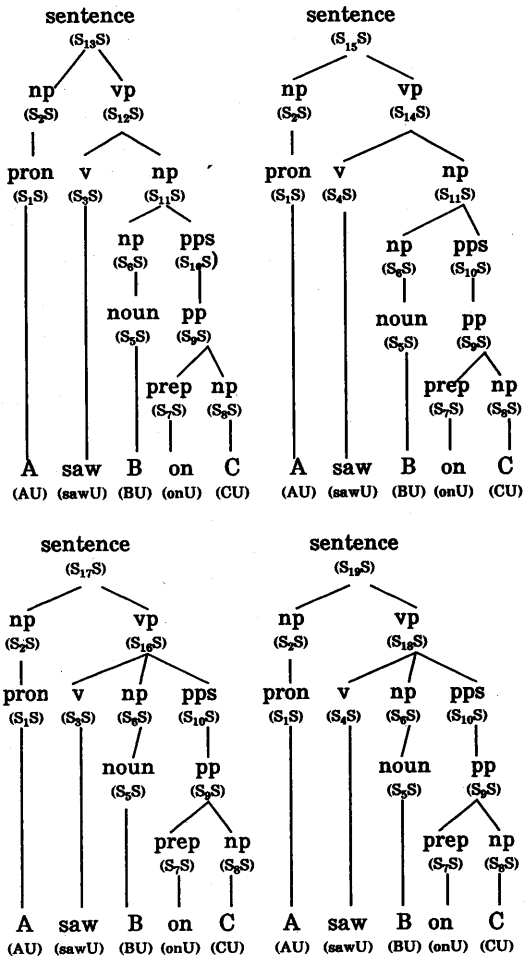


図1 "You saw Jon on that"

$(DS \models S_{13})$ or $(DS \models S_{15})$ or
$(DS \models S_{17})$ or $(DS \models S_{19})$

公理D2より

$(DS \models S_{13}) \Rightarrow (DS \models S_2)$ and $(DS \models S_{12})$
$(DS \models S_2) \Rightarrow (DS \models S_1)$
$(DS \models S_{12}) \Rightarrow (DS \models S_3)$ and $(DS \models S_{11})$
$(DS \models S_{11}) \Rightarrow (DS \models S_6)$ and $(DS \models S_{10})$
$(DS \models S_6) \Rightarrow (DS \models S_5)$
$(DS \models S_{10}) \Rightarrow (DS \models S_9)$
$(DS \models S_9) \Rightarrow (DS \models S_7)$ and $(DS \models S_8)$
$(DS \models S_{15}) \Rightarrow (DS \models S_2)$ and $(DS \models S_{14})$
$(DS \models S_{14}) \Rightarrow (DS \models S_4)$ and $(DS \models S_{11})$
$(DS \models S_{17}) \Rightarrow (DS \models S_2)$ and $(DS \models S_{16})$
$(DS \models S_{16}) \Rightarrow (DS \models S_3)$ and $(DS \models S_6)$ and $(DS \models S_{10})$
$(DS \models S_{19}) \Rightarrow (DS \models S_2)$ and $(DS \models S_{18})$
$(DS \models S_{18}) \Rightarrow (DS \models S_4)$ and $(DS \models S_6)$ and $(DS \models S_{10})$

発話より

$(DS \models S_1) \Rightarrow (DU \models A)$
$(DS \models S_3) \Rightarrow (DU \models saw)$
$(DS \models S_4) \Rightarrow (DU \models saw)$
$(DS \models S_5) \Rightarrow (DU \models B)$
$(DS \models S_7) \Rightarrow (DU \models on)$
$(DS \models S_8) \Rightarrow (DU \models C)$
$DU \models A$ $DU \models saw$ $DU \models B$
$DU \models on$ $DU \models C$

以下、上記制約の処理方法について述べる。

まず、以降の議論では、上記の各述語を図1で用いた簡略化により、例えば一つの述語$DU \models C$をCUとして示される一つの論理変数として用いる。これにより、構文解析で得られた、談話における制約は全て以下の様に命題論理式として扱える。

$S_{13}S$ or $S_{15}S$ or $S_{17}S$ or $S_{19}S$
$S_{13}S \Rightarrow S_2S$ and $S_{12}S$ $S_2S \Rightarrow S_1S$
$S_{12}S \Rightarrow S_3S$ and $S_{11}S$
$S_{11}S \Rightarrow S_6S$ and $S_{10}S$
$S_6S \Rightarrow S_5S$ $S_{10}S \Rightarrow S_9S$
$S_9S \Rightarrow S_7S$ and S_8S
$S_{15}S \Rightarrow S_2S$ and $S_{14}S$
$S_{14}S \Rightarrow S_4S$ and $S_{11}S$
$S_{17}S \Rightarrow S_2S$ and $S_{16}S$
$S_{16}S \Rightarrow S_3S$ and S_6S and $S_{10}S$
$S_{19}S \Rightarrow S_2S$ and $S_{18}S$
$S_{18}S \Rightarrow S_4S$ and S_6S and $S_{10}S$
$S_1S \Rightarrow AU$ $S_3S \Rightarrow sawU$
$S_4S \Rightarrow sawU$ $S_5S \Rightarrow BU$
$S_7S \Rightarrow onU$ $S_8S \Rightarrow CU$
AU sawU
BU onU
CU

(1)受動的な制約処理

第一の手法は、遅延実行制御(CIL, Prolog-IIのfreeze, PSIのbind_hook)による制約の受動的な解法である。本手法では、制約式の書き換えを能動的には行わない。具体的には、ある論理変数に、それが取るべき値についての制約を記述すると、論理変数に値が代入されるまでは、この制約は評価されない。

まず、論理型言語CIL[Mukai 87]で実現されている遅延実行制御による命題論理の制約解決機構について説明する。表1にCILの真理値表を示す。Uは値

表1 CIL真理値表

∧	1	0	U	∨	1	0	U	¬	
1	1	0	U	1	1	1	1	1	0
0	0	0	0	0	1	0	U	0	1
U	U	0	U	U	1	U	U	U	U

の定まらない変数である。例えば、CILでは以下の

ように命題論理の制約を解決する。記号＞は、CILの督促記号である。

```
＞constr(and(A, B), false), A = true.        (1)
A = true,
B = false,
＞constr(and(A, B), false).                  (2)
A = Unbound
B = Unbound
```

(1)の場合、論理変数AとBの論理積が偽であるという制約を述語constrで定義し、かつ、Aが真であるとしている。すると、自動的にB＝偽が得られる。

(2)の場合、論理変数AとBの論理積が偽であるという制約だけを述語constrで定義している。すると、AもBもその値は決定されずUnboundのまま残る。

ちなみに、A⇒Bは、¬A∨BとしてCILで定義されている。

以上の機能を用いて、上記命題論理で表現された制約を定義できる。

ただし、上記の公理1から得られる制約、つまり、$S_{13}S$ and $S_{15}S$ and $S_{17}S$ and $S_{19}S$＝真からは、何ら論理的な帰結は自動的には得られない。ここに、受動的な解法と、能動的な解法の異なりが現れる。ただし、例えば$DS \models S_3$が偽であることが分かれば、$S_{13}S$ or $S_{17}S$＝偽となる。

(2)能動的な制約処理

前記の命題論理の制約の解法には種々の手法が適応可能である。

第1の手法としては、命題論理式の各論理変数に総当たり的に値を割り当てる手法があるが、これは、計算量が$O(2^n)$(nは変数の数)になるので、用いたくない。

第2の手法は、上記の命題論理の制約が、Prologと同様のクローズのみから与えられている事を利用する。例えば、以下の様なプログラムで実行可能である。

```
go(V,U):-
    assert(':-'(s,s13s)),assert(':-'(s,s15s)),
    assert(':-'(s,s17s)),assert(':-'(s,s19s)),
    assert(AU),assert(sawU),
    assert(BU),assert(onU),
    assert(':-'(s1s,AU),assert(':-'(s3s,sawU)),
    assert(':-'(s4s,sawU),assert(':-'(s5s,BU)),
    assert(':-'(s8s,CU)),
    assert(':-'(s13s,(s2S,s12s))),assert(':-'(s2s,s1s)),
    ＜中略＞
    go([s1S,s2,s3,s4,s5,s6,...,s16s,s17s,s18s,s19s],
            V,U).
go([],[],[]).
go([H|R],V,U):-retract(':-'(H,B)),
            (s,assert(':-'(H,B)),U＝[H|UR],V＝VR
            ;assert(H),V＝[H|VR],U＝UR),
            go(R,VR,UR).
```

本プログラム実行により、述語goの第1引数Vへ真となる論理変数の名前が入り、第2引数Uへ値の決まらない論理変数の名前が入る。本手法の欠点は、例えばs2s⇒s1sでs2s＝真のとき、自動的にs1s＝真とできない点である。

そこで、上記のassert部分をCILの命題論理制約記述に置き換える方法が、インプリメントでは有効である。具体的なプログラムは以下のようになる。

以下の場合、値の決まらない変数だけが、リストとして返される。

```
go(U):-
    constr(_s13s∨_s15s∨_s17s∨_s19s,true),
    constr(AU,true),constr(_sawU,true),
    constr(BU,true),constr(_onU,true),
    constr(_s1s⇒AU,true),
    constr(_s3s⇒sawU,true),
    constr(_s4s⇒sawU,true),
    constr(_s5s,⇒BU,true),
    constr(_s8s,⇒CU,true),
    ＜中略＞
    go1([_s1S,_s2,_s3,...,s16s,_s17s,_s18s,_s19s],
            U).
go1([],[]).
go1([H|R],U):-
            (H = false,assert(flag),fail
            ;(flag,     U＝[H|UR],
                        retract(flag);
                        U＝UR)),
            go1(R,UR).
```

計算量は詳細は省くがPrologの場合$O(n\times m)$(nは命題論理変数の総数、mは命題論理式の総数)で、CILの場合$O(n\times m')$($m' \leqq m$)である。現在の所、本手法を我々は用いている。

第3の手法はブーリアングレブナーベースのソルバを用いる手法である。本手法でも計算時間は$O(n\times m)$になる。詳細は[Sakai 88]を参照されたい。

5. 談話における制約処理

以下では、前記の能動的な制約処理方法によって解かれた制約の意味を考察する。

まず、4.で示した命題論理の制約を解くと、以下の帰結が得られる。値の決まらない変数間の関係を図2へ示す。

$(DS \models S_1)$and$(DS \models S_2)$and$(DS \models S_5)$and
$(DS \models S_6)$and$(DS \models S_7)$and$(DS \models S_8)$and
$(DS \models S_9)$and$(DS \models S_{10})$＝真

①値の決まった命題論理値に対応する述語

文文法の制約と、公理D1,D2だけから、上記の値が決まる事に注意して欲しい。談話処理は、これら、値の真となる命題論理値に対応した述語を真とするように、談話を形成する必要がある。また、対応した述語内部の意味構成を行っても多重環境による論理変数共有は起こらない。例えば、SAXでは以下の様に意味構成を行っている。

$rel(S,(S_1,..,S_n)) \wedge (DS \models S) \Rightarrow S = make(S_1,..,S_n)$

makeはボトムアップに意味を組み上げる関数である。relを枝刈りに用いる場合は、解析実行中のコントロールを考慮する必要があるが、詳細は6に示す。

多重環境による論理変数共有が起こらないのは、シンタクスにおいて、値の決まった命題論理値に対応する述語は全ての多重環境によって共有されている為である。

②値の決まらない命題論理値に対応する述語

談話処理において、順次これらの値を決定する必要がある。まず、図2を例に考えると、公理2より、図の上の部分の真偽値は、図の下の部分から順次真偽を決定してゆく事により、決定できる。

ところが、ここで、重要な事実にぶつかる。例えば、公理1と

$((DS \models S_{3\,(to\ saw)}) \Rightarrow (DU \models saw))$,

$((DS \models S_{4\,(to\ see)}) \Rightarrow (DU \models saw))$,

$(DU \models saw)$ より、$(DS \models S_3)$ or $(DS \models S_4)$ となるが、$(DS \models S_3)$と$(DS \models S_4)$のどちらが正しいかを単純に決めるには、$not(DS \models S_3)$か$not(DS \models S_4)$を導く必要がある。談話処理において、このような否定的な情報を部分的な談話情報から導く事には非常な困難が予想される。

例えば、談話中でAがBを見た(to see)事が真であっても、$S_{3\,(to\ saw)}$を否定する事は論理的にはできない。では、どのように解決すべきか、以下考察する。

(A) 語彙の曖昧さについて

語彙の曖昧さについては、例えば、談話がどの言語で営まれ、どの分野の談話であるかなどの情報、つまりテキストタイプや、プレサポジションなどを用いる事が必要であろう。

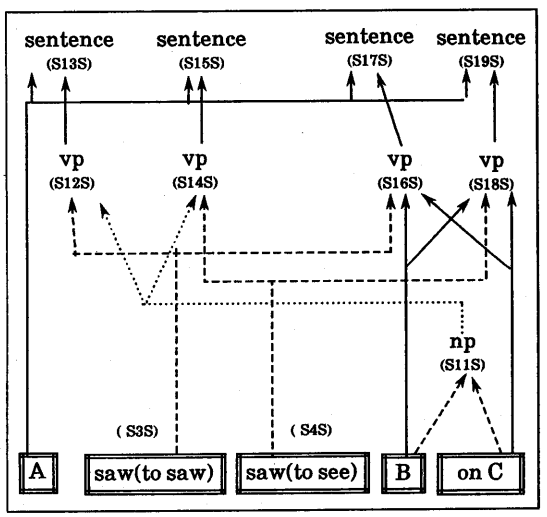


図2 談話上の論理的帰結

具体的には、例えば、語義に尤度情報を付ける事なども実際に動くシステムを作成する上では必要であろう。つまり、談話状況をタイプ分けして、ある状況下で用いられる頻度の低い単語については、例えば$(DU \models saw) \Rightarrow \neg(DS \models S_{3(to\ saw)})$とする方法が有効であろう。これは、ワードプロセッサで、頻度情報や、直前使用単語情報が有効に働くことからも、妥当であると思う。$\neg(DS \models S_3)$が分かれば、$DS \models S_{13}$と$DS \models S_{17}$は、論理的な帰結として偽になる。

(B)談話依存の曖昧さについて

談話状況の間の規則については、まだまだ多くの課題があり、分かっていない事のほうが遥かに多いが、最低限以下の公理系を用いても良いと考えている[Fenstad 85]。

$DS \models A \wedge B \Leftrightarrow (DS \models A)$ and $(DS \models B)$ (DI1)

$DS \models A \vee B \Leftrightarrow (DS \models A)$ or $(DS \models B)$ (DI2)

$not(DS \models A) \Leftrightarrow (DS \models \neg A)$ (DI3)

語彙の曖昧さと異なり、複雑な構造を持つ意味が、談話で成立するかどうかを、尤度情報で計算することには、多大な困難があるように思われる。そこで、論理の中でまず何をすべきか考えてみる。

②で述べた問題を以下にまとめる。

α) 談話において、$DS \models A$の真偽を決定する問題

β) 推論規則における前提と帰結の問題。

γ) 論理和の制約(曖昧さ)縮退の問題。

前述したように、本枠組みではDUのレヴェルから順に図2の上方向に各述語の真偽を決定してゆける。そこで、Aには何らかの部分情報が与えられているとしよう。

まずα)について整理する。Aと談話状況との関係には以下のようなものが考えられる。

1) AかnotAが談話状況から証明できる場合

$DS \models A \wedge B$のように、Aを含むような状況があれば、(DI1)より、$DS \models A$を真とする事ができる。(DI1)と(DI3)より、¬Aを含むような状況があればnot(A)とできる。例えば、図2で$S_{11}S$が偽である事が分かると、更に、$S_{17}S$と$S_{19}S$が残り、$S_{13}S$と$S_{15}S$は偽となる。

2) AもnotAも談話状況から証明できない場合

3値論理でいうならばUnknownである場合であるが、以下のような分類ができる。

2-1)Aに含まれるSがある場合

例えば、$DS \models A1 \wedge A2 \wedge A3$の真偽を決定する際に、$DS \models A2$は証明できる場合である。自然言語の場合このような場合が多い事が予想される。この場合、部分情報から仮説を生成して推論を働かせる方法などが有効かもしれない。

2-2)Aとある部分を共有するSがある。

これは、論理的には、2-1)と同等である。

2-3)Aに関係する情報は無い。

このような場合も非常に多い事が予想される。本モデルではこの場合、AはUnboundのまま残る。

β)については、帰納推論を行う必要があるだろうが、最低、談話においては、3値論理の方法に従って次のような制約を加えても良いだろう。

公理D3

$\neg(\sim(DS \models A1)\ or\ ,...,\ or \sim(DS \models An)) \Rightarrow (DS \models A)$
where $(DS \models A) \Rightarrow (DS \models A1)\ and\ ,...,\ and\ (DS \models An)$

～は、weak negationであり、真理値表は以下のようになる。

表2. ～の真理値表

～	1	0	U
	0	1	1

直感的な意味は、談話においてH⇒Bなる推論規則がある時、その論理的帰結部分Bが全て真であるならば、Hを真として良いというものである。

γ)については、公理D1を以下の形に変形しても良いかもしれない。

公理D1'

$\iota(DS \models S)$ where $(DS, S) \in I$

ιはIの要素の中の唯一の要素のみが⊨を満たす事をしめすとする。

以上、考察を進めたが、これらの公理については実証研究も含めて、研究を続けて行きたい。

6.制約生成系としての構文解析

6.1. 制約と多環境の関係

以上の議論を踏まえて、解析を捉えなおしてみる。特に、多環境による論理変数共有の問題との関係を考察する。

まず、2.で述べた論理型文法における述語relの意味を検討してみよう。2では、述語relは、その引数の間に何らかの関係が成り立つ場合に真となるとしたが、大別すると、述語relは、論理変数に全く影響を与えない制約の追加定義def__consと、ある種の制約の評価eval__consの2つの部分から成っている。ユニフィケーションは、def__consとeval__consを併せ持つ概念である。

$rel((DS_h,S_h),((DS_1,S_1),...,(DS_n,S_n)))$
$\Rightarrow def\_\_cons((DS_h,S_h),((DS_1,S_1),...,(DS_n,S_n)))$
$\wedge eval\_\_cons((DS_h,S_h),((DS_1,S_1),...,(DS_n,S_n)))$

上記の峻別は、SAXのように、複数の環境を並列にもちながら解析を進める場合には重要である。つまり、単純に制約をdef__consにより、追加定義するだけであれば、解析時点では環境共有の問題は生じない。これは、TreeをSAXで作成することができる事からも明らかである。def__consは制約を定義するだけなので、常に成功する。

さらに、本論で紹介したように、複数の環境すべてによって共有されている文法カテゴリ(図1の$S_1S,S_2S,S_5S,S_6S,S_7S,S_8S,S_9S,S_{10}S$)に対応するrelについては、制約の評価をeval__consにより行っても複数環境による論理変数共有の問題は生じない。

問題は、上記relを以下の形として解析実行中に用いる場合である。

$\neg eval\_\_cons((DS_h,S_h),((DS_1,S_1),...,(DS_n,S_n)))$
$\Rightarrow \neg rel((DS_h,S_h),((DS_1,S_1),...,(DS_n,S_n)))$

上記の場合、eval__consにより論理変数への代入が起こり、かつ、他の環境とこの変数を共有している可能性がある。relに渡る情報が全てグランドならば、問題は無いが、そうでなければ、環境をコピーする必要がある。

タームリライティングシステムの場合でも、制約の正規形への変換を変数を伴って行うならば、コピーをする必要がある。

では、実際にはとのような制約がrelについてはあるのか表3にまとめてみる。

表3. relにおける制約

	DS_h	S_h	DS_j	S_j	DS_i	S_i
DS_h		①	②	③		
S_h			④	⑤		
DS_j				⑥	⑦	⑧
S_j						⑨

①と⑥は、本論では⊨なる関係だけを考えている。def__consにのみ関与し、relそのものの計算には関係しない。⊨に関する制約は、公理D1,D2,D3であり、解析終了時点で定義される。

②と⑦は文内のコンテクストの切り替わり等の制約を定義し、def__consにのみ関与する。

③は、DS_hの部分状況DS_iでの意味S_iとDS_hとの関係に関与する。例えば、②と⑥ないしは⑧との関係にまたがるような制約を記述する事ができる。def__consにのみ関与する。

④と⑧は、どのような関係がありえるのか、判然としないが、def__consにのみ関与する。

⑤と⑨は、共に意味を定義するさいに、中心的な働きをすると考えられる。ユニフィケーション文法でも中心となる部分であり、eval__consにより、解析中に制約の評価を行う必要がある。したがって、SAXでは、環境のコピーが必要となる。

6.2. 制約生成系SAX

以上のように、枝刈りに用いる制約を、論理変

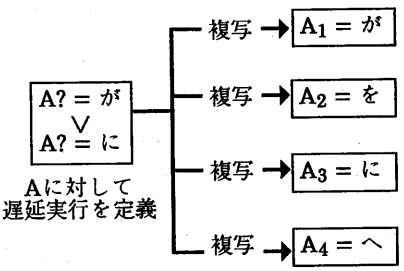
数の上に定義し、解析中に評価する場合には、環境のコピーが必要になる。

SAXでは、以上の問題点に鑑み、動的に制約を評価すべき部分と、文解析終了時点で評価すべき部分を識別して下のように記述する。

head -> body1,{ext1},..,bodyn,{extn} && {extL}

表3における、①②③④⑥⑦⑧は、文解析終了時点で評価すべき部分"{extL}"へ、記述して蓄積し、公理1と2を追加して、評価すれば良い。⑤と⑨については、{ext1}から{extn}へ記述する。宣言的な文法記述には、環境のコピーを必要とする。

ところが、遅延実行評価を用いて単純に{extj}へ制約を記述した場合には、変数に付けられた遅延実行プログラムが複写できないため、制約は動かない。例えば以下の場合を考えよう。



この場合、変数A_1からA_4へのユニフィケーションは全て成功してしまう。一般に、SAXのように、多環境を保持した構文解析において、Disjunctionを含む制約の宣言的な記述を実現するには、単純なユニフィケーションも、遅延実行メカニズムも使えない。そこで、現在、論理変数に対する制約の複写を行える以下のような、簡単な言語プリミティブの設計を考えている。これらは、制約に対する制約、つまり、メタな制約である。

制約の読み出しと定義	def_cons
制約の削除	del_cons
制約の評価	eval_cons

これらのプリミティブの正確な意味については、自己記述(Reflection)[田中88]との関係等の議論があり、後日報告をしたい。なおこの種の議論は正規形の研究に中心をおく制約論理の研究とは異なるものであることをお断わりしておく。

7. おわりに

文解析と談話構造の関係を曖昧さの処理の観点から整理した。まだまだ未熟な点が多いが、今後も、本手法をベースに談話構造の研究を行って行きたい。また、構文解析システムとしては、現在、将来並列化が可能なSAXを用いているが、制約生成系として、どのような言語プリミティブが必要なのか、更に検討を重ねたい。

[謝辞]

本研究にあたり、ICOT第2研究室向井主任研究員からは、多くの示唆を頂きました。本研究の機会を下さいました、ICOT渕所長、内田第2研究室室長に感謝いたします。

[参考文献]

[Barwise 83] Barwise, J. and Perry, J., Situations and attitudes, MIT Press, Cambridge, 1983

[Dincbas 86] Dincbas, M., Constraints, Logic Programming and Deductive Databases, Proceedings of France-Japan Artificial Intelligence and Computer Science Symposium 86, pp. 1-27, ICOT, 1986

[Fenstad 85] Fenstad, J.E., Halvorsen, P.K., Langholm, T., Benthem J., Equations, Schemata and Situations: A framework for linguistic semantics, CSLI Report No. CSLI-85-29, 1985

[Jaffar 87] Jaffar, J., and J.,L., Lassez, Constraint Logic Programming, Technical report from Dept. of Computer Science, Monash University.

[Kaplan 82] Kaplan, R.M., & Bresnan, J., Lexical-Functional Grammar: A Formal System for Grammatical Representation, in Bresnan, J., ed., The Mental Representation of Grammatical Relations, MIT press series on cognitive theory and mental representation, 1982

[今野 86] Konno, S. and Tanaka, H., Processing Left-extraposition in Bottom-up Parsing System, Computer Software Vol.3 No.2 (Tokyo, 1986) pp. 19-29

[Matsumoto 84] Matsumoto, Y., Kiyono, M. and Tanaka, H., Facilities of the BUP Parsing System, Proc. of 1st International Workshop on Natural Language Understanding and Logic Programming, Rennes, 1984

[Matsumoto 87] Matsumoto, Y., and Sugimura, R., A Parsing System based on Logic Programming, IJCAI 87

[Mukai 87] Mukai, K., A System of Logic Programming for Linguistic Analysis Based on Situation Semantics, Proceedings of the Workshop on Semantic Issues in Human and Computer Languages, Half Moon Bay, CSLI, 1987

[向井 88] 向井国昭, 談話理解とロジック,人工知能学会誌,vol.3, No.3, 1988, pp43-54

[Nakamura 86] Nakamura, J., Solutions for Problems of MT Parser -- Methods Used in Mu-Machine Translation Project, Coling '86, pp.133-135, 1986

[Sakai 88] Sakai. K, Sato Y., Boolean Gröbner Bases, ICOT Technical Memo No.488, 1988

[杉村 86] 杉村領一,日本語の待遇表現と状況意味論、ソフトウェア科学会、Vol.3 No.4 , pp46-54, 1986,

[Sugimura 87] Sugimura, R.,Miyoshi,H. Mukai,K., Constraint analysis on Japanese Modification, Proceeding of the Second international Workshop on Natural Language Understanding and Logic Programming, Northholland, 1987

[田中 88] 田中二郎,メタプログラミングとリフレクション,bit,vol.20,No5,pp41-50