自然言語処理 44－2
（1984. 7. 26）

ホーン節を内部表現とする自然言語理解システム

西原典孝　森田憲一
（阪大・基礎工）

1　はじめに

計算機による自然言語理解には色々なアプローチがありうるが、ここでは、“自然言語の文を何らかの内部表現に変換し、知識として蓄え、疑問文に対して、その知識を元に推論を行ない答を出す”ことである、と定める。そして、本研究では、その内部表現として一階述語論理式のsubsetであるホーン節を用い、prolog上でこの全過程を実行するシステムを実現した。対象言語としては、英語のsubsetをとった。

ホーン節を内部表現に用いて自然言語を処理するやり方は、近年よく用いられる手法である。その代表的なものとしてPereiraによるChat-80システムがある［1］。しかし、Chat-80は、知識ベース（データベース）に対する英語の問合せ文を処理することに主眼を置いたシステムであり、平叙文から知識ベース表現への変換、蓄積を必要としていない。また、内部表現の中で取扱われる個体対象はすべて固有名詞か数値であった。結局、このシステムでは、自然言語理解における、より一般的かつ基本的な問題の多くが省られることなく残されていたといえる。

筆者らの目的は、自然言語理解に基づく応用システムを作成することではなく、むしろ、自然言語理解自体の問題点を明確にし、その解決を試みることにある。そして、そのような試みを実行するためのツールとして、自然言語から知識ベース表現への変換、蓄積、および疑問文に対して得られた解からの適切な答文の生成等を含めた自然言語理解システムを作成した。

また本論文では、副詞および指示的な冠詞“the ”の取扱いについても論じ、本システムでのそれらの解決方法を述べた。

2　システムの構成

システムは、次のような5つの部門からなる。

1）翻訳部門
2）ホーン節化部門
3）登録部門
4）推論部門
5）答文生成部門

入力された文は 1）により一階述語論理式に翻訳される。対象言語としては英語のsubsetを用いる。生成された一階述語論理式は 2）により、スコーレム変換され、ホーン節に直される。そして平叙文に対するホーン節は 3）に送られ知識ベースとして蓄えられる。その表現は全くprologのプログラムと同一である。一方、疑問文に対するホーン節は 4）を起動させる。 4）は、送られてきたホーン節を入力とし、知識ベースに基づき論理的推論を行ない答を出す。 5）は、 4）で生成された論理的答と元の疑問文の構文情報に基づき、英語の答文を生成し、出力する。

全システムは、前述したように、prolog上で実現されている。このprologはＡＣＯＳ－１０００ＬＩＳＰ１．９上でインプリメントしたものである。

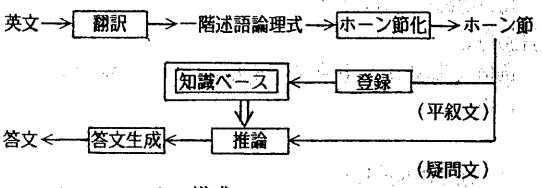

図1　システムの構成

3　統語規則と翻訳規則

統語規則と翻訳規則により、それぞれ、本システムで取扱い可能な自然言語の範囲とその論理式への翻訳方法を定める。また、ここでは補助的な規則として、統語制限と解釈仮説を設けた。全く同一の文でも、曖昧性を含む文は複数の読みを持つ。統語制限と解釈仮説は、入出力をスムーズに行なうために、このような曖昧性を制限し一つの読みにするためのものである。

3．1　統語規則

統語規則を文脈自由文法的に略式に表現すると次のようになる。実際には、動詞が三人称単数形であるか原形であるか等の文脈的情報も取扱える。これにより定められる英文には、形容詞、副詞、代名詞、接続詞、関係代名詞等が許され、時制は現在形のみで、複数形や否定文は許されていない。疑問文はＹＮ疑問文（Yes, No 疑問文）および、主語ないし目的語に相当するものを尋ねるWH疑問文（What 疑問文）である。　なお、WH疑問文の疑問詞はWhat のみであり、Who、Whom も全て What で代行される。

統語規則（一部）

s1)　文－＞平叙文
s2)　文－＞YN疑問文
s3)　文－＞WH疑問文
s22)　普通名詞句－＞普通名詞句´＋関係代名詞節
s23)　普通名詞句－＞普通名詞句´
s24)　普通名詞句´－＞形容詞＋普通名詞
s25)　普通名詞句´－＞普通名詞

３．２　統語制限

統語的曖昧性に対する統語制限として次の２つを設けた。

統語制限１
副詞は直前の動詞のみに係り、前置詞句による副詞句は動詞句全体に係る。
統語制限２
関係代名詞に続く動詞句は可能なだけ、広い範囲をとる。

３．３　基本語句の翻訳

翻訳規則については、基本語句に対する翻訳方法のみを説明しておく。

翻訳例

	英語表現	翻訳
普通名詞・・一項述語	man	man (x)
形容詞　・・一項述語	red	red (x)
固有名詞・・個体定数	john	john
副詞　・・副詞定数	slowly	slowly
前置詞　・・副詞関数	in	in (x)
自動詞　・・二項述語	run	run (x , ad)
	第１引数－－主語に相当する個体変数	
	第２引数－－副詞リスト	
他動詞　・・三項述語	love	love (x , y , ad)
	第１引数－－主語に相当する個体変数	
	第２引数－－目的語に相当する個体変数	
	第３引数－－副詞リスト	

英文から論理式の変換における従来の方法では、自動詞を一項、他動詞を二項述語に翻訳している。本システムでは、これに副詞リストという引数を追加し、副詞情報を動詞に対応する述語に埋め込むようにした。これは、副詞により生じる論理の高階性を避けるためである。なお、冠詞“a ”は存在限量化を行なうもの、“every ”は全称限量化を行なうものとして翻訳する。冠詞“the ”については後述する。

３．４　解釈仮説

統語的曖昧性と同じく、解釈的曖昧性を含む文も複数の読みを持つ。この解釈的曖昧性を取除くために、次のような解釈仮説を与える。この解釈仮説は、一般的にみて最も納得できる読みを選び出せるようなものとした。

解釈仮説１
代名詞が指示しているものは、その性と対応し、その代名詞よりも前でかつ最も近くにある文中の名詞句とする。
解釈仮説２
冠詞による限量化の範囲は、文全体とし、左優先とする。

３．５　疑問文の解釈

YN疑問文
YN疑問文に対応する肯定文の論理式をPとする。このPが、今ある知識ベース、すなわち論理式群から導けるかどうかを推論するわけである。この推論は導出原理に基づいて行なわれる。導出原理は背理法を使用しているので、この疑問文の最終的な翻訳は～Pとすればよい（ここで、～は否定記号）。

WH疑問文
WH疑問文において、What に相当するものを個体変数 xとおいた時の翻訳をP (x) とする。この疑問文に対して行なうべき推論は“P (x) を成り立たせるようなx に対するすべてのインスタンスを見つけよ”であると解釈する。同じく、この推論は導出原理に基づくので、WH疑問文の最終的な翻訳は、
(∀x) ～P (x)　　{＝～ (∃x) P (x)) }
とすればよい。

３．６　冠詞the を伴う名詞句（［the ］＋普通名詞句）の解釈

ｉ）その普通名詞句が関係代名詞節を伴う場合
冠詞“the ”は冠詞“a ”と全く同一なものとして解釈する。

ii）その普通名詞句が関係代名詞節を伴わない場合

冠詞“the ”は指示的な“the ”であり、その名詞句は、それ以前に蓄えられた知識中の、ある個体表現を直接指示しているものと見なす。例えば、“John loves a woman.”という事実が知識の中に存在する時、

Bill loves the woman.

の“the woman ”は“John loves a woman.”中の“a woman ”と同一のものを指示しているものとする。このような取扱いは次のような手続きにより実現される。

普通名詞句の翻訳をC（x ）とすると、（∃x ）C（x ）が知識ベースから導けるかどうかを導出原理を適用し推論する。もし導けるならば、その時の xに対するインスタンスを“the woman ”の翻訳とする。そのようなインスタンスが複数個あるならば、最新に入力された知識に対応するものを採用する。

C（x ）が導けない場合は、この“the ”は“a ”と解釈的に同一であると見なし、かつその“the ”による限量化は最優先であるとする。

4　翻訳部門

翻訳部門では、次のことを行なう。入力された文に対して、まず、それに適合する統語規則を見つけ出し、次々に文を分解していき文構造を調べる。そして、その統語規則に対応する翻訳規則を適応して、論理式への変換を行なう。

本システムでは、これを行なうための構文解析法としてDCG（definite clause grammar ）を採用した［２］。ここでのDCGは統語規則に翻訳規則を埋め込んだ表現をしており、構文解析と翻訳を同時に実行できる。

この部門に英文を入力すると、それは論理式に翻訳され、その結果と入力文の構文情報（平叙文、疑問文の区別、および主語、動詞の内容等）が続く部門に送られる。構文解析に失敗した場合、すなわち、入力文が統語規則に適合しない場合、システムは“I can´t understand . ”というメッセージを出力する。

構文解析の際に、文中に代名詞が現れた場合、それが指示しているものを知るために、代名詞変換リスト［M, F, N］を用意しておく。M, F, Nには、その性がそれぞれ男性，女性，中性である名詞句に対応する最新の個体表現がユニファイされている。この代名詞変換リストは、新しい名詞句が現れる毎に、その性に対応した要素の値が更新される。各要素の初期値はnil としておく。

なお、論理式中の、whatに相当する個体変数は“ans ”で表わされ、それ以外のすべての変数は、同一変数の識別のために、ナンバリングされたアトム　v1,v2...で表わされている。

翻訳例

(i) john runs slowly.

run（john, ［slowly］）

(ii) john eats the fish that a woman finds.

（∃v1）（∃v2）（ eat（john, v1, nil ）∧ fish（v1）∧find（v2, v1, nil ）∧ woman（v2））

ここで“eat ”、“find”には副詞が係っていないので、副詞リストは nil（=空リスト）となっている。

(iii) what does john eat ?

（∀ ans）～ eat（john, ans , nil ）

5　ホーン節化部門

この部門では、生成された一階述語論理式を、まずスコーレム変換して、スコーレム標準形に直し、さらに節表現に直す。そして、その結果であるホーン節リストは、入力文が平叙文ならば登録部門に送られ、入力文が疑問文ならば推論部門に送られる。

スコーレム定数およびスコーレム関数名は、論理式P中の存在限量化されている各変数ごとに、別の名前を与えなければならないため、ナンバリングされたアトム c1,c2,..を使用する。このナンバリングは、知識ベース全体にわたって行なっている。ただし、疑問文に対するホーン節は知識ベースに蓄えておく必要がないため、この部門の起動ごとにリセットされるナンバリングアトム a1,a2,..を使用する。

さらに、prologにおいては、変数は大文字で表わすので、論理式中の全称限量化されているすべての変数は大文字に変え、prologプログラムの変数と同一化する。

注：スコーレム変換して作成した節集合の中にホーン節でないものが存在した場合、システムはエラーメッセージを出し、その節を削除する。　平叙文において、“every ”を伴う名詞句の関係代名詞節中以外で接続詞“or”が使われた場合、非ホーン節が発生する。本来、このような文は統語規則で制限すべきであったが、現段階では成されていない。

文からホーン節への翻訳例

(i)　john eats the fish that a woman finds.

fish（c0）.
woman（c1）.
eat（john, c0, nil ）.

find (c1, c0, nil) .

(ii) the woman loves john.

love (c1, john) .

"the woman " が指示しているものが c1 となっている。

(iii) does john eat a fish?

← eat (john, V1 , nil) , fish (V1) .

(iv) what does john eat?

← eat (john, ANS, nil) .

6 登録部門

登録部門では、ホーン節化部門から送られてきた平叙文に対するホーン節、および答文生成のための名詞句情報を知識ベースへ登録する。すべての登録が終了すると、システムは"I see. "というメッセージを出力する。

6.1 ホーン節の登録

平叙文に対するホーン節集合は知識として蓄えられる。否定文が許されていないため、このホーン節集合は、宣言節ないし規則節から成り、ゴール節は含まれていない。これらのホーン節は、prologプログラムと全く同一であり、事実、システムはprologのプログラムとしてこれらを登録する。

6.2 名詞句情報の抽出と登録

8章で詳述するが、答文生成のために、存在限量詞化されている変数に対して与えられたスコーレム関数が具体的に何を指すのかの情報を蓄えておかなければならない。ただし、このシステムにおいて現段階で取扱えるのは、スコーレム定数に関してだけである。蓄えておく情報（今後、名詞句情報と呼ぶ）は、

ci : Pi

ここでciはスコーレム定数、Pi はciに関するホーン節リスト（この定義は8章で与える）

であり、このような情報の抽出は、翻訳部門および本部門で行なわれる。そして、名詞句情報 ci : Pi は、述語名がci、その引数がPi である"ci (Pi) . "というホーン節の宣言節の形で登録される。すなわち、平叙文に対するホーン節と同様に、prologプログラムとして登録される。なお、疑問文に対するホーン節は、この部門に送られないため、このような操作は行なわれない。

7 推論部門

疑問文に対するホーン節はこの部門に送られてくる。推論部門は、このホーン節に対して、知識ベースを元に推論を行ない論理的答を導出する。

7.1 推論メカニズム

推論部門において行なうべきことは、(『知識ベース』→『疑問文が尋ねている事実』) を証明することであり、疑問文がWH疑問文ならば、この式を成り立たせしめるwhatに相当する変数のインスタンスを求めることである。一方、本システムでは、自然言語に対する内部表現としてホーン節を使用している。このため、ホーン節集合＝prologプログラム、ホーン節集合に対する推論＝prologプログラムの実行、という性質により、本部門での推論は、知識ベースと疑問文に対するホーン節集合をそのままprologプログラムとして実行することで実現できる。

疑問文に対するホーン節集合は、一般に次のようなものになる。

P1
:
Pn　　n ≧ 0　　Pi は宣言節ないし規則節
←G　　←Gはゴール節

これらのホーン節に対する推論の手続き（今後、推論手続きと呼ぶ）は、次のようなものである。

P1 , . . , Pn を知識ベースに追加登録する。この新たな知識ベースであるホーン節集合とゴール節"←G"をprologプログラムとして実行する。もし"←G"が成功すれば、推論手続きは成功したと呼ぶことにし、成功しなければ、推論手続きは失敗とする。ゴール節の評価を終えると、P1 , . . , Pn を知識ベースから削除し、推論手続きは終了する。

7.2 付加推論規則

推論部門では、導出原理の適用だけでは不十分な点を、付加推論規則でおぎない、推論能力を高めている。

例えば次の文について考えてみる。

john runs slowly.
does john run ?

これらの文に対するホーン節はそれぞれ、

run (john, [slowly]) .
← run (john, nil) .

となる。この場合、疑問文に対するゴール節は第2引数が異なるため成功しない。しかし、一般には、この疑問文に対してはYesと答えるのが妥当であろう。そこで次

の付加推論規則を設ける。

付加推論規則1

動詞名(x1,..,xn, 副詞リスト´)
→
動詞名(y1,..,yn, 副詞リスト)　　n = 1,2
が成り立つのは、
動詞名(x1,..,xn) → 動詞名(y1,..,yn)
が成り立ち、かつ
副詞リスト´⊇副詞リストの時である。

この付加推論規則を実現するには、疑問文に対するゴール節

←Q1,Q2,.., Qn .　　(n ≧1)

の述語Qi が、"動詞名(x1,..,xn, ad)"である場合、Qi の代りに2つの述語、"動詞名(x1,..,xn, AD1), include(AD1, ad)"をゴール節に挿入し、これを新たなゴール節とすればよい。ここで" include(x, y)"はx ⊇y を意味する述語である。

7.3 推論部門の動作

YN疑問文に対する推論

YN疑問文に対するホーン節集合に対して、推論手続きを行ない、この手続きが成功したならば、答はYesとなり、失敗したならばNo となる。このように推論は閉世界の立場で考え、疑問文が知識ベースから証明できない場合、すなわち、知識として知らない事柄を尋ねられた場合、No と答える。

WH疑問文に対する推論

WH疑問文に対するゴール節を"←G(ANS)"とおく。"ANS"はwhatに相当する変数である。prologにおいては、"←G(ANS)"が成功するような"ANS"に対するインスタンス(今後、これを解と呼ぶ)が最初に見つかった時点で、プログラムの実行は終了する。しかし、このような解が複数個存在する場合がある。WH疑問文に対する推論では、ゴール節を成功させるような"ANS"に対するインスタンスをすべて求めなければならない。

そこで、すべてのインスタンスを蓄えておくためのシンボルanslを用意する。そして、推論手続きに入る前に、シンボルanslの属性値にnil (=[])を登録し、推論手続きを開始する。"←G(ANS)"が成功するインスタンスが見つかると、その値はanslの属性値のリストに挿入され、そこで強制的にバックトラックを起し、別の解の探索にかかる。この手順が繰返され、最終的に推論手続きは失敗に終る。この時には、すべての解がanslの属性値として蓄えられている。

この2種類の推論を実現するためには、次のように行なえばよい。まず、シンボルanslの属性値にnil を登録する。そして推論手続きを実行する。ただし、各ゴール節は、次のように変更したものを用いる。

今、そのゴール節を"<−G"とする。このGに対して、付加推論規則で述べた変更を行なった結果をG´とし、このG´を構成している述語Q1,.., Qn を要素とするリストをP=[Q1,.., Qn]とおく。そして、最終的なゴール節を次のようなものにする。

←G´, ref-ans(SIGN, P)
ここでSIGN=nil　　YN疑問文の場合
=[ANS]　WH疑問文の場合
ref-ans(nil , P).
ref-ans([ANS], P) ← ins-put(ansl, [ANS, P]), false .
ここで ins-put(s , x)は、シンボル sの属性値として登録されているリストに xを挿入する述語である。

述語 ins-putが呼出された時点では、ANSには解であるインスタンスがユニファイされ、Pには、P中の各変数に具体的な値がユニファイされている。なお、このPは答文生成に必要とされる。

WH疑問文に対しては、推論手続きは最終的に失敗に終る。しかし、すべての解はanslの属性値として蓄えられている。

7.4 実行例

every man walks.
john loves mary.
john loves sarry.
これらの文に対するホーン節は、
walk(V1, nil) ← man(V1).
love(john, mary, nil).
love(john, sarry , nil).
である。今、これらのホーン節が知識ベースに蓄えられているとする。

(i)"does every man walk ?"に対するホーン節、
man(a1).
←walk(a1, nil).
が推論部門に送られてきた場合、推論手続きは成功し、

答えはYesとなる。

(iii) “what does john love ?” に対するホーン節、

←love (john, ANS, nil) .

が送られてきた場合、解のリストは、[mary, sarry] となる。

8 答文生成部門

答文生成部門では、推論部門の結果と、翻訳部門で生成された疑問文の構文情報を元に、疑問文に対する答えを正しい英文で出力する。YN疑問文に対しては、推論部門の推論手続きが成功したならばYes、失敗したならばNo と答える。WH疑問文に対しては、シンボルansl に蓄えられている解のリストを用いて、答文を生成する。ただし解のリストが nilならば、システムは “ I don´t know. ” と出力する。

8．1 スコーレム定数に対する名詞句生成

平叙文 “john eats a fish. ” に対するホーン節、

eat (john, c0, nil) .

fish (c0) .

が知識ベースに蓄えられている時、疑問文 “what does john eat?” に対する解のリストは [c0] である。ここでc0は、スコーレム定数である。

答文生成のためには、このc0が具体的に指す名詞句をまず生成しなければならない。スコーレム定数に対する名詞句生成には、そのスコーレム定数が引数に含まれている知識ベース中のホーン節を使用する。普通、そのようなホーン節は複数個存在し、そのどれを使用するかにより生成される名詞句も異なってくる。そこで次のような名詞句生成の仕様（すなわち、名詞句生成に使用するホーン節集合の選び方）を定める。

名詞句生成の仕様１

そのスコーレム定数が、元々、冠詞 “the ” を伴う名詞句に対応するものならば、その名詞句に対応するホーン節集合から生成する。

名詞句生成の仕様２

そのスコーレム定数が、元々、冠詞 “a ” を伴う名詞句に対応するものならば、その名詞句が含まれている文に対応するホーン節集合から生成する。

上のような条件の成り立つホーン節集合の内、そのスコーレム定数を含み、かつ宣言節であるものが、最終的に使用されるホーン節である。

8．2 名詞句生成の手続き

推論部門で導出された解に対する名詞句を生成する手続きは、次のようなものである。ここで、その解をq とし、それに対して生成される名詞句は、その名詞句を構成する単語のリストＮＬで表わすことにする。

q ＝固有名詞 の場合 ＮＬ＝[q]

q ＝スコーレム定数 の場合

まず、そのスコーレム定数に対するホーン節集合を取出す。このようなホーン節は、そのスコーレム定数を述語名とする述語の引数の形で、登録部門において、既に知識ベース中に蓄えられている。これらをそのまま利用すればよい。そして、それらを元に名詞句ＮＬを生成する。ここで、各ホーン節は宣言節、すなわち１つの述語表現となっている。ホーン節の中に述語名が動詞であるものが存在する時には、関係代名詞節で名詞句を構成する。さらにその述語の各引数に対しては、この名詞句生成手続きを再帰的に呼出し、各引数を名詞句に直す。ただし副詞情報は名詞句生成に用いない。

q ＝スコーレム関数 の場合 ＮＬ＝[q]

スコーレム関数については、現システムでは取扱うことができないので、その値をそのまま返している。

名詞句生成の例

john eats the fish that he finds.

eat (john, c1) .

find (john, c1, nil) .

fish (c1) .

c1に対するホーン節集合、すなわち登録部門で登録されている名詞句情報は次のようなものである。

c1 ([find (john, c1, nil) , fish (c1)])

c1に対する名詞句ＮＬは次のようになる。

ＮＬ＝[the , fish, that, john, finds]

8．3 名詞句生成における冗長性の除去

８．２で例示した平叙文に対するホーン節が知識ベース中にあるとき、疑問文 “what does john finds?” （そのホーン節は、←find (john, ANS, nil) . ）に対する解はc1である。c1に対して生成される名詞句は “the fish that john finds” であり、 “that john finds ” の部分は、冗長なものになる。

さらに次の例について考えてみる。

john seeks the woman that finds a unicorn.

seek (john, c2, nil) .

```
find (c2, c3, nil ).
 woman (c2).
 unicorn (c3).
c2 ( [find (c2, c3, nil ), woman (c2) ] )
c3 ( [find (c2, c3, nil ), unicorn (c3) ] )
```

c2に対する名詞句NLは、

NL＝[the , woman , that, finds , (c3に対する名詞句)]

c3に対する名詞句NLは、

NL＝[the , unicorn , that, (c2に対する名詞句), finds]

となる。このように、c2に対する名詞句には、c3に対する名詞句を含み、かつその逆も成り立っているので、名詞句生成手続きは、無限ループに陥ってしまう。

このような冗長性は除去しなければならない。推論部門で導出された各々の解は、[解, P]の形で蓄えられている。冗長性を除去するために、このPを利用し、名詞句生成手続きを次のように変更する。

まず、除去すべきホーン節集合を表わすリストDLを用意し、初めにDL＝Pとおいて、名詞句生成手続きに入る。そして、q ＝スコーレム定数の場合、それに対するホーン節集合の中で、述語名が動詞でかつその述語と同一のものがDL中に存在するかをチェックし、もし存在すれば、その述語は名詞句生成に用いないようにする。

さらに、名詞句生成手続きを再帰的に呼出す際に、その名詞句生成手続きに対するDLには、このDLと、最初に選び出されたホーン節集合を連接したリストDL´を使用する。

このように変更した名詞句生成手続きにより生成される名詞句NLは、次のようになる。

c1に対して、

NL＝[a , fish]

c2 に対して、

NL＝[the , woman , that, finds , a , unicorn]

c3 に対して、

NL＝[the , unicorn , that, a , woman , finds]

8．4 代名詞変換

答文中の名詞句に、疑問文中の名詞句と同一のものがあれば、その名詞句は代名詞で置き変えるべきである。

この代名詞変換には、翻訳部門での最終的な代名詞変換リスト[M, F, N]を使用する。そして、名詞句生成手続きを次のように変更する。

名詞句生成手続きにおいて、まず最初に、解 qが代名詞変換リスト[M, F, N]中に存在するかをチェックし、

q ＝Mの場合　NL＝[he]ないし[him]

q ＝Fの場合　NL＝[she]ないし[her]

q ＝Nの場合　NL＝[it]

とする。代名詞の格をどちらにするかは、qが、主語なのか目的語なのかにより選択する。

8．5 答文生成例

(i) mary kills the man that loves her.

```
kill (mary, c0, nil ).
love (c0, mary, nil ).
 man (c0).
```

what does mary kill ?

←kill (mary, ANS, nil).

この場合、解はc0で、生成されるc0に対する名詞句と答文は、それぞれ次のようになる。

the man that loves her

she kills the man that loves her.

(ii) the woman that finds a red fish eats it.

```
find (c1, c2, nil ).
 eat (c1, c2, nil ).
 woman (c1).
 red (c2).
 fish (c2).
```

what eats the fish?

← eat (ANS, c2, nil).

この場合、解はc1で、生成されるc1に対する名詞句と答文は、それぞれ次のようになる。

the woman that finds it .

the woman that finds it does.

9 まとめ

知識ベースへの知識の登録、スコーレム定数に対する名詞句生成等を含めた、英文の入力から出力までの首尾一貫した自然言語理解を行なう基本システムを作成できた。さらに、本来、ホーン節だけでは記述が困難であった対象、例えば、副詞や、冠詞“the ”を伴う名詞句に対して、付加推論規則を設けたり、手続き的な取扱いを織り混ぜることによって、その取扱いの実現に成功した。

本システムは、知識表現として論理式をとっているため、理論的評価、考察が容易であると共に、柔軟性と拡張性に富んだものとなっており、より能力のある自然言語理解システム開発のためのコアシステムになると考えられる。

次に現システムでの問題点をいくつかあげる。

本システムでは、次のような推論はできない。

john likes every dog.

like (john, x , nil) ← dog (x) .

この文は、複数形が許されるならば、 “john likes dogs.” となる。

what does john like ?

←like (john, ANS, nil) .

このゴール節は失敗する。すなわち、述語likeは成功するが、新しいゴール節← dog (x) は成功しない。(もちろん、YN疑問文 “does john like every dog?” に対しては、問題はなく、Yesと答える。)

また、この場合、

bill has a dog.

dog (c1) .

have (bill, c1,nil) .

という事実があるとすると、先の疑問文に対する解はc1となる。結局、答文は、

john likes the dog that bill has.

となってしまう。このようなことが起こるのは、推論を閉世界で考えているからである。すなわち、その世界では、知識として知っている事実がすべてであるととらえるので、確かにその世界では犬は一匹しかいなく、“every dog ” の中身がこの犬のみになるわけである。

さらに、prologにおいては、否定の表現ができない。否定の表現であるゴール節は、質問ととらえられるからである。このような問題点を含めて、より枠の広い自然言語の取扱いを実現するために、メタロジカルな機能の強化 [3] 、ないし内部表現として用いる論理体系自体の拡張等を現在検討中である。

[参考文献]

1)pereira,F. :Logic for Natural Language Analysis. PhD thesis, Univ. of Edinburgh (1982).
2)Pereira,F. and Warren,D. :Definite Clause Grammar for Language Analysis - A Survey of the Formalism and a Comparision with Augumented Transition Networks, Aritificial Intelligence,13, pp. 231-278(1980).
3)Bowen,K.A. and Kowalski,R.A. :Amalgamating Language and Metalanguage in Logic Programming, School of Computer and Information Sciences, Univ. of Syracuse (1981).

システムの実行例

```
+JOHN RUNS SLOWLY IN SUMMER.
 I SEE.
+DOES JOHN RUN SLOWLY?
YES.HE DOES.
+DOES JOHN RUN RAPIDLY?
NO.HE DOESN'T.
+JOHN LOVES MARY AND MARY LOVES BILL.
 I SEE.
+MARY IS A WOMAN.
 I SEE.
+DOES JOHN LOVE THE WOMAN THAT LOVES BILL?
YES.HE DOES.
+JOHN FINDS A RED LARGE UNICORN AND EATS IT.
 I SEE.
+THE WOMAN THAT TOM LOVES KILLS HIM.
 I SEE.
+WHAT DOES JOHN EAT?
 HE  EATS THE RED LARGE UNICORN THAT HE FINDS.
+WHAT FINDS A RED UNICORN.
  JOHN DOES.
+WHAT KILLS TOM?
  THE WOMAN THAT HE LOVES DOES.
+SARRY LOVES THE MAN THAT RUNS IN THE PARK.
 I SEE.
+THE MAN LOVES KATE.
 I SEE.
+BEN FINDS A BLUE FISH.
 I SEE.
+MARY EATS THE FISH.
 I SEE.
+WHAT DOES THE MAN THAT RUNS IN THE PARK LOVE?
 HE  LOVES KATE.
+WHAT DOES MARY EAT?
 SHE  EATS THE BLUE FISH THAT BEN FINDS.
+END.

CPUTIMES= 25571.MSEC
```