Prologを使ったSDRT表記の実現

吉 武 春 光

1．まえがき

前稿(吉武,2007)では，Segmented Discourse Representation Theory（以下SDRT と略す)を使って，コンピュータ処理向けに，談話の意味記述を行った．本稿では，前稿で使用した複数の例文の SDRT 表記を実際にコンピュータにプログラムし，前稿の理論が正しいかどうかの検証を行った．

2．SDRT を使った談話の意味表現

図１が，大まかな，処理の流れである．前稿では，図１を手動でシミュレーションしたことになる．前稿では，回答文を分析する際に，形態素処理プログラム Chasen を使用し,その後で日本語の構文解析プログラム KNP を使用した．本来は，全処理を連ねて行うようにプログラムを書くべきであるが，今回は，構文解析が終わった段階から先の処理のみのプログラムを書くことにした．

プログラミングは，筆者個人の Linux マシンにて行った．

OS: Fedora 8
プログラミング言語：SWI-Prolog (Version 5.6.57)

2.1 Prolog の基本

Prolog (Programming in Logic) は論理型プログラミング言語である．一般に使われているＣ言語や Perl 等は手続き型プログラミング言語と呼ばれている．Prolog は第１階述語論理に基づいており，物事の間に成り立つ関係を定義していくことでプログラミングを行う仕組みになっている．

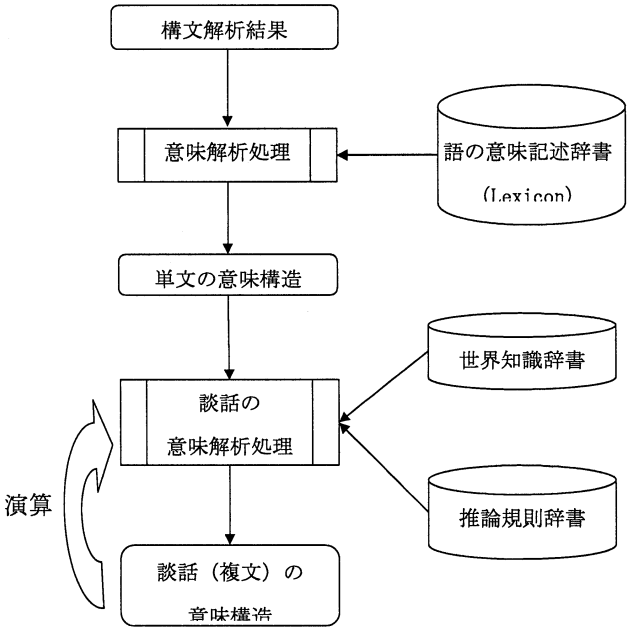

図１　SDRT における意味処理の流れ

Prolog は，複数の規則(rule) の集合からなっている．各規則は，次の形式をしている．

```
head  :-  goal-1, goal-2, ... goal-n  .
```

ここで，head と goal は，節(clause) と呼ばれ，第１階述語論理における述語に相当するものである．規則の行末にはピリオド . を付ける．この規則の意味は，head が成立するためには，goal-1, goal-2, ... goal-n の各 goal が全て成立しておかなければならない，ということである．各 goal は，論理積 (AND) の関係で，各規則は，論理和(OR)の関係になっている．規則の中で goal が全くないもの(つまり head だけの場合)を事実と呼ぶ．事実の集まりをデータベースと呼ぶこともある．一方，規則の中で head が全くないもの(つまり

goal だけの場合)を質問と呼び.Prolog はインタープリタと呼ばれる対話型の状態で稼働するのが基本で，その状態は，質問を受け付けている状態(つまり，goal を受け付ける状態)になっている．

Prolog の動作は,質問の各 goal とパターンマッチングする head を探すことになる．一般のプログラミング言語のパターンマッチングとは違い，Prolog のパターンマッチングでは変数同士のパターンマッチングも成功するようになっている．変数同士のパターンマッチングが成功した場合は，その変数の間の関連付けが行われる．規則の検索は自動的に再帰的に行われる．これをバック・トラッキングと呼んでいる．つまり，プログラミングにおいて，プログラムの制御を書かなくて済むのである．そのため，規則と事実の記述のみを行えば良いので，自然言語処理の理論を確かめる実験ではしばしば使われている．

2.2 SWI-Prolog

過去に，Prolog 処理系は幾つも開発され(黒川,1985)，方言も多かった．しかし，最近では，ISO standard for Prolog (http://www.logic-programming.org/prolog_std.html) が規定されたので，ISO 準拠の Prolog 処理系ならば，差がほとんどない．また，日本語処理に関しては，コンピュータシステムの文字コードが UTF-8 になっており，更に Prolog 処理系も UTF-8 対応の場合には，全く問題なく使用できる．現在，入手できるものとしては，IF/Prolog, SICStus Prolog, K-Prolog, SWI-Prolog などがある．最近では，無料で利用でき，SWI-Prolog が用いられていることが多いようだ．本研究でも SWI-Prolog を使用することにした．

SWI-Prolog (http://www.swi-prolog.org/)は，オランダの Amsterdam 大学が開発している Prolog 処理系であり，次の特徴がある．

・GNU Public License に従った Free Software
・色々な OS をサポートしている．
・Web からダウンロードにより容易に入手できる．
・他の C 言語, Java 言語, ODBC, CGI などへのインターフェースを備えている．
・グラフィカル表示のための関数を持っている．

・グラフィカル・デバッガと，キャラクタ・デバッガが提供されている．

なお,SWI-Prolog で使用される変数は必ず大文字で始まることになっている．もし大文字で始まる定数や日本語などを使用する場合は，両側をクオート’で挟んで使用する．また，他では参照されていない変数を無名変数(Singleton variables) と呼び，変数の先頭に下線 _ を付けて表す．

2.3　使用した回答文

以下では，前稿で SDRT による意味記述を行った e-Learning 回答文２文について，プログラミング言語 Prolog による記述を行った．但し，前稿でも前提として明記したが,語の GL としての意味記述は，後で必要に応じて行うことにして，基本的には割愛することにした．

なお，学生の回答は HTML 文章になっているので，解析処理のために HTML コードを除去した．また，回答文には全て異なる固有番号３桁が付いている．これを回答文番号と呼ぶことにする．

以下に，使用した２つの e-Learning 回答文を再掲しておく．

原文のままの回答文番号 039
ASCII コードにカタカナをつけたものを JISX0201 と言う，JIS による日本の漢字コードを JIS X　0208 と言い，実用的なコードは三つシフト JIS，EUC-JP，ISO-２０２２-JP が存在する．JISX0201 は半角カタカナを格納する ASCII コード部分にヨーロッパでは他の固有の文字が入っており，これが文字化けを起こす．

原文のままの回答文番号 183
◆日本の文字コードの種類◆ シフト JIS;日本のパソコン用の文字コードとして作られた． EUC-JP;UNIX 用 ISO-2022-JP;電子メール用（最上位ビットは０，つまり７bit でおさまる） ◆半角カタカナと呼ばれるコードの問題点◆

電子メールに，JISX0201 の８単位符号表の右半分が入っていると，表示の際に文字化けを起こすものが多い．
※インターネットの世界では使用禁止．

2.4 世界知識

図１の［談話の意味解析処理］は，[単文の意味構造］以外に，[世界知識］や［推論規則］が必要になる．これらは，いわば，人間の知識に該当する部分であるが，その量は膨大である．本実験では，談話の意味構造を生成するために必要最低限な世界知識と推論規則を記述することにした．

2.5 助詞「の」の意味記述

助詞「の」の意味については，数多くの解釈が存在することが知られている．(菊池，白井,2004)においては，助詞「の」の意味として７つの可能性を示している．しかしながら，筆者は，本研究を進める段階で，(菊池，白井,2004)で示されている「文脈によって決まる任意の関係」のままでは意味記述に無理があることを感じ，細分化する方が良いと判断するに至った．結局，前稿の解釈に新たに２つの可能性を追加して，以下の合計９つの解釈があるとした．

表１　「A の B」の解釈

項番	解　釈	説　明	例
1	名詞 B で表される関係	名詞 A がその項	太郎の兄
2	名詞 B で表される事象	名詞 A がその項	言葉の理解
3	名詞 B に固有的な関係・事象	名詞 A がその項	トヨタの車
4	名詞 A と名詞 B が独立に個体，事象のタイプを限定	文脈から名詞 B が個体と解釈される場合，名詞 A は付加的情報をもたらす	美人の母，日課の散歩
5	所有・所属	名詞 A が個体や組織の固有名で，名詞 B は人間・事物	花子の本

6	属性的	名詞Aが時間・場所・順序	東京のビル
7	集合の一部	B という集合の一部が A である	テニスの趣味
8	一部分	A の一部分が B である	JISX0201 の右半分
9	文脈によって決まる任意の関係		太郎の車

これを Prolog 表記すると次の記述が得られる.

助詞「の」に関する Prolog 表記

```
'Relation'(1,A,B) :- is_list(B),H =.. B,memberchk(H,A).
'Relation'(2,_A,_B).
'Relation'(3,A,B) :- is_list(B),H =.. B,memberchk(H,A).
'Relation'(4,A,B) :- is_list(A),H =.. A,memberchk(H,B).
'Relation'(5,A,B) :- is_list(A),H =.. A,memberchk(H,B).
'Relation'(6,_A,_B).
'Relation'(7,A,B) :- subset(A,B).
'Relation'(8,A,B) :- mem(B,A).
'Relation'(9,_A,_B).
list-sum( [], 0 ).
list-sum( [_X|Y], Z):- list-sum( Y, Z2), Z is 1 + Z2.
mem(A,B) :- is_list(A),member(M,A),!,H =.. [M],memberchk(H,B).
```

但し，解釈２，解釈６，解釈９については本例文には出現していないので，今回は記述していない.

2.6　意味記述の正当性の確認方法

記述した文の意味が正しいかどうかを確認する手段としては，従来から，質問応答という手段が使われてきた．すなわち，記述した文の意味に対して，世

界常識を用いて質問し，それに対するシステムの返答を得，その返答の内容が正しいかどうかによって記述した文の意味が正しいかどうかを判断するというものである．以下の Prolog による意味記述についても，この質問応答を行うことにより正当性を確かめた．

2.7 SWI-Prolog の実行の様子

SWI-Prolog を使って，回答文番号(039-1)の Prolog 表記を実行させている様子を次に示す．

?- の右側が，SWI-Prolog に対するユーザ（つまり，筆者）の入力である．

```
[yositake@ai Prolog]$ pl -s wk.pl  ⇐ 注1
% /home/yositake/.plrc compiled 0.00 sec, 236 bytes
% wk.pl compiled 0.00 sec, 17,420 bytes
Welcome to SWI-Prolog (Multi-threaded, 32 bits, Version 5.6.57)
Copyright (c) 1990-2008 University of Amsterdam.
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?- [s039].  ⇐ 注2
% s039 compiled 0.01 sec, 2,056 bytes
true.  ⇐ 注3

?- 's039-1'.  ⇐ 注4
true .  ⇐ 注3

?- 'ASCII コード'(X),'カタカナ'(Y),'つける'(_E,X,Y,Z),'もの'(Z),'JISX0208'(Q),'言う'(Z,Q).  ⇐ 注5
false.  ⇐ 注6

?- 'ASCII コード'(X),'カタカナ'(Y),'つける'(_E,X,Y,Z),'もの'(Z),'文字コード'(Q),'言う'(Z,Q).  ⇐ 注7

X = ['ASCII コード'],
Y = ['半角カタカナ'],
_E = ['つける'],
Z = ['ASCII コード', '半角カタカナ'],
Q = ['JISX0201', 'ASCII コード', '半角カタカナ'] .  ⇐ 注8

?-
```

図2　回答文番号(039-1) に対するコンピュータ・プログラムの動作の様子

注1：pl は SWI-Prolog の起動コマンドである．オペランド -s で指定したファイルを，SWI-Prolog 起動時に読み込むようになっている．世界知識を wk.pl というファイルに記述したので，ここでは，SWI-Prolog 起動

時に wk.pl を読み込ませている.

注2：SWI-Prolog に対する質問として［s039］を入力している．カギ括弧のペア［ と ］は，consult 関数を表している．［ と ］で指定したファイルを SWI-Prolog に読み込んでから実行する．この例では，s039.pl というファイルを読み込ませている．その際，拡張子 .pl は省略できる．

注3：SWI-Prolog からの応答で，true は質問が旨くパターンマッチングに成功したことを表している．

注4：SWI-Prolog に対する質問として s039-1 を入力している．s039.pl というファイル中に s039-1 という名前の head を持つ規則を書いているので，その 規則とパターンマッチングし，更に，s039-1 という名前の head を持つ規則の各 goal を次々にパターンマッチングした結果，全てが旨くパターンマッチングを成功し，true. と表示されている．

注5：SWI-Prolog に対する質問として，複数の goal を連ねたものを入力している．この場合，「ASCII コードにカタカナをつけたものを JISX0208 と言うか？」という意味的に間違った質問にしている．

注6：SWI-Prolog からの応答で，false は質問がパターンマッチングに失敗したことを表している．

注7：SWI-Prolog に対する質問として，複数の goal を連ねたものを入力している．この場合，「ASCII コードにカタカナをつけたものは何という文字コートか？」という質問にしている．

注8：SWI-Prolog からの応答で，パターンマッチングの結果として変数に何が関連付けられたかを表示している．ここで，変数 Q に注目すると，旨く JISX0201 がパターンマッチングしているのが分かる．

3．回答文番号 039 の SDRT による意味記述

2.3 で挙げた回答文 039 を最小限の後編集を行った結果，次の 3 文に分かれる．

分析に使用した回答文番号 039

(039-1) ASCII コードにカタカナをつけたものを JISX0201 と言う. (039-2) JIS による日本の漢字コードを JISX0208 と言い，実用的なコードは 3 つシフト JIS，EUC-JP，ISO-2022-JP が存在する. (039-3) JISX0201 は半角カタカナを格納する ASCII コード部分にヨーロッパでは他の固有の文字が入っており，これが文字化けを起こす.

これらの文の SDRT による意味記述は次の通りになる.

3.1 回答文番号(039-1)

回答文番号(039-1)
(039-1) ASCII コードにカタカナをつけたものを JISX0201 と言う.

前稿に示したように，次の通りに意味記述できる.

回答文番号(039-1) の SDRT による意味記述
π039-1: [x, y, e1, z | ASCII コード(x) ∧ カタカナ(y) ∧ つける(e1, x, y, z) ∧ もの(z) ∧ JISX0201(q) ∧ 言う(z, q)]

ここで，言う(z, q) は，発言するという意味ではなく，そのように呼んでいるという意味で使われているので，=(z, q) に置き直すことにする.

回答文番号(039-1) の SDRT による意味記述
π039-1: [x, y, e1, z, q | ASCII コード(x) ∧ カタカナ(y) ∧ つける(e1, x, y, z) ∧ もの(z) ∧ JISX0201(q) ∧ =(z, q)]

その際，'もの'(Z). という形として'もの'の引数を変数のままにしておくと Prolog の unify が旨く出来る. つまり，'もの'は変項であると捉える. その結果，次の Prolog 記述が得られる.

回答文番号(039-1) に関する世界知識の Prolog 表記

```
'ASCII コード'(['ASCII コード']).
'カタカナ'(X) :- '半角カタカナ'(X);'全角カタカナ'(X).
'半角カタカナ'(['半角カタカナ']).
'全角カタカナ'(['全角カタカナ']).
'JISX0201'(['JISX0201','ASCII コード','半角カタカナ']).
```

回答文番号(039-1) の Prolog 表記

```
's039-1' :- 'ASCII コード'(X),'カタカナ'(Y),
    'つける'(_E,X,Y,Z),'もの'(Z),'JISX0201'(Q),'言う'(Z,Q).
```

3.1.1 実行の様子

前出の図２が実行の様子である．以下，該当個所を抜粋しておく．

回答文番号(039-1) の実行の様子

```
[yositake@ai Prolog]$ pl  -s  wk.pl
... 省略 ...
?- [s039].
% s039 compiled 0.01 sec, 2,056 bytes
true.
?- 's039-1'.
true .
?-
```

回答文番号(039) 全体の Prolog 記述を読み込ませてから，回答文番号(039-1)である 's039-1' を実行させている．この結果として true が表示されているので，回答文番号(039-1) が正しいことが分かる．

3.1.2 誤った内容を検出させた例

前出の図２における注５は,意図的に次の誤った内容を入力したものである.

誤った内容を含んだ文(039-1-bad)
(039-1-bad) ASCII コードにカタカナをつけたものを JISX0208 と言う.

誤った内容を含んだ文(039-1-bad) の Prolog 表記
's039-1-bad' :- 'ASCII コード'(X),'カタカナ'(Y), 'つける'(_E,X,Y,Z),'もの'(Z),'JISX0208'(Q),'言う'(Z,Q).

誤った内容を含んだ文(039-1-bad) の実行の様子
[yositake@ai Prolog]$ pl -s wk.pl ... 省略 ... ?- [s039]. % s039 compiled 0.01 sec, 2,056 bytes true. ?- 'ASCII コード'(X),'カタカナ'(Y),'つける'(_E,X,Y,Z),'もの'(Z),'JISX0208'(Q),'言う'(Z,Q). false. ?-

この結果として false が表示されているので,この誤った内容の文が,システムによって誤っていると判別されているのが分かる.

3.1.3　質問をしてシステムに回答させた例

前出の図２における注７は,質問として,ASCII コードにカタカナをつけた文字コードを問うたものである.

質問文(039-1-q)
(039-1-q) ASCII コードにカタカナをつけた文字コードを何と言うか？

質問文(039-1-q)の Prolog 表記
's039-1-q' :- 'ASCII コード'(X),'カタカナ'(Y), 'つける'(_E,X,Y,Z),'もの'(Z),'文字コード'(Q),'言う'(Z,Q).

質問文(039-1-q)の実行の様子
[yositake@ai Prolog]$ pl -s wk.pl ... 省略 ... ?- [s039]. % s039 compiled 0.01 sec, 2,056 bytes true. ?- 'ASCII コード'(X),'カタカナ'(Y),'つける'(_E,X,Y,Z),'もの'(Z),'文字コード'(Q),'言う'(Z,Q). X = ['ASCII コード'], Y = ['半角カタカナ'], _E = ['つける'], Z = ['ASCII コード', '半角カタカナ'], Q = ['JISX0201', 'ASCII コード', '半角カタカナ'] . ?-

この結果として，変数 Q の値として (Q = ['JISX0201', 'ASCII コード', '半角カタカナ'])が求まっている．変数 Q の値はリスト形式になっており，第1番目の項('JISX0201')が，求まった文字コードの名前であり，正しく求められているのが分かる．

3.2 回答文番号(039-2)の意味記述

分析に使用した回答文番号(039-2)
(039-2) JIS による日本の漢字コードを JISX0208 と言い，実用的なコードは3つシフト JIS，EUC-JP，ISO-2022-JP が存在する．

前稿に示したように，次の通りに意味記述できる．

回答文番号(039-2) の SDRT による意味記述
π039-2: [m, n, o, q, a, b, c, h, i, j | JIS(m) ∧ R2(m, o) ∧ 日本(n) ∧ 漢字コード(o) ∧ R(n, o) ∧ JISX0208(q) ∧ =(o, q) ∧ 実用的(a) ∧ コード(b) ∧ R3(a, b) ∧ 3つ(c) ∧ =(b, c) ∧ SJIS(h) ∧ EUC-JP(i) ∧ ISO-2022-JP(j) ∧ a-part-of(h, b) ∧ a-part-of(i, b) ∧a-part-of(j, b) ∧ =(sum(h, i, j), 3)]

その結果，次の Prolog 記述が得られる．

回答文番号(039-2) に関する世界知識の Prolog 表記
'言う'(Z, Q):- subset(Z, Q). 'JIS'(['JIS']). '日本'(['日本']). '漢字コード'(X) :- 'JISX0208'(X). '漢字コード'(X) :- 'SJIS'(X). '漢字コード'(X) :- 'EUC-JP'(X). '漢字コード'(X) :- 'ISO-2022-JP'(X). 'JISX0208'(['JISX0208','JIS','日本']). '実用的'(['実用的']). 'コード'(Z) :- '漢字コード'(Z). '3つ'(3). 'SJIS'(['SJIS','8bit','パソコン','実用的','日本']). 'EUC-JP'(['EUC-JP','8bit','UNIX','実用的','日本']). 'ISO-2022-JP'(['ISO-2022-JP','7bit','電子メール','実用的','日本']).

回答文番号(039-2)の Prolog 表記

```
's039-2' :-
   'JIS'(M),'漢字コード'(O),'Relation'(5,M,O),
   '日本'(N),'Relation'(5,N,O),'JISX0208'(L),'言う'(L,O),!,
   '3つ'(C),findall(S,'s039-2-sub'(S),List),list-sum(List,C).

's039-2-sub'(B) :- '実用的'(A),'コード'(B),'Relation'(4,A,B).

list-sum( [], 0 ).
list-sum( [_X|Y], Z):- list-sum( Y, Z2), Z is 1 + Z2.
```

3.2.1 実行の様子

次は，実行の様子である．

回答文番号(039-2)の実行の様子

```
[yositake@ai Prolog]$ pl  -s  wk.pl
... 省略 ...
?- [s039].
% s039 compiled 0.01 sec, 2,056 bytes
true.
?- 's039-2'.
true .
?-
```

回答文番号(039) 全体の Prolog 記述を読み込ませてから，回答文番号(039-2)である 's039-2' を実行させている．この結果として true が表示されているので，回答文番号(039-2) が正しいことが分かる．

3.2.2 誤った内容を検出させた例

回答文番号(039-2)の内容と矛盾する次の内容を入力してみよう．

誤った内容を含んだ文(039-2-bad)
(039-2-bad) 実用的なコードは４つ存在する.

誤った内容を含んだ文(039-2-bad) の SDRT による意味記述
π039-2-bad: [a, b, c | 実用的(a) ∧ コード(b) ∧ R3(a, b) ∧ ４つ(c) ∧ =(b, c)]

誤った内容を含んだ文(039-2-bad)の Prolog 表記
's039-2-bad' :- 　findall(S,'s039-2-sub'(S),List),list-sum(List,4).

誤った内容を含んだ文(039-2-bad) の実行の様子
[yositake@ai Prolog]$ pl -s wk.pl ... 省略 ... ?- [s039]. % s039 compiled 0.01 sec, 2,056 bytes true. ?- findall(S,'s039-2-sub'(S),List),list-sum(List,4). false. ?-

この結果として false が表示されているので, この誤った内容の文が, システムによって誤っていると判別されているのが判る.

3.2.3 質問をしてシステムに回答させた例

ここでは, 実用的な文字コードは何かと質問を行う.

質問文(039-2-q)
(039-2-q) 実用的なコードは何か？

質問文(039-2-q)の Prolog 表記

```
's039-2-q' :- '実用的'(A),'コード'(B),'Relation'(4,A,B).
```

実は，これは，回答文番号(039-2)の Prolog 表記の中で用いていた一時的な記述 's039-2-sub' と同じである．

質問文(039-2-q)の実行の様子

```
[yositake@ai Prolog]$ pl  -s  wk.pl
... 省略 ...
?- [s039].
% s039 compiled 0.01 sec, 2,056 bytes
true.
?- findall(S,'s039-2-sub'(S),List).
List = [['SJIS', '8bit', 'パソコン', '実用的', '日本'], ['EUC-JP', '8bit',
'UNIX', '実用的', '日本'], ['ISO-2022-JP', '7bit', '電子メール', '実用的
', '日本']].

?-
```

この結果として，変数 List に，リスト形式にて，次の３つが求まっているのが分かる．

['SJIS', '8bit', 'パソコン', '実用的', '日本'],
['EUC-JP', '8bit', 'UNIX', '実用的', '日本']
['ISO-2022-JP', '7bit', '電子メール', '実用的', '日本']

各々，第１番目の項（つまり，'SJIS'と'EUC-JP'と'ISO-2022-JP'）が文字コードの名前になっているので，正しく求まっているのが分かる．

3.3 回答文番号(039-3)の意味記述

分析に使用した回答文番号(039-3)
(039-3) JISX0201は半角カタカナを格納するASCIIコード部分にヨーロッパでは他の固有の文字が入っており，これが文字化けを起こす.

(039-3)は重文になっており，「おり」という接続助詞が用いられている．「おり」の解釈としては，表４における命題間の関係の Result に該当すると思われるが，ここでは，単に論理積∧で結合することにする．その結果，(039-3)の意味記述は，次の通りになる．

回答文番号(039-3) の SDRT による意味記述
π039-3: [r, s, e3, t, u, e4, v, w, e5 | 半角カタカナ(r) ∧ ASCII コード(s) ∧ 格納する(e3, r, s) ∧ 他の(t) ∧ R(t, u) ∧ 固有名詞(u) ∧ 入っている(e4, u, s) ∧ 文字化け(v) ∧ JISX0201(w) ∧ 起こす(e5, w, v)]

その結果，次の Prolog 記述が得られる．

回答文番号(039-3) に関する世界知識の Prolog 表記
'JISX0201'(['JISX0201','ASCII コード','半角カタカナ']). '半角カタカナ'(['半角カタカナ']). '格納する'(['格納する'],A,B) :- mem(A,B). 'ASCII コード'(['ASCII コード']). '部分'(_P). 'ヨーロッパ'(['ヨーロッパ']). 'では'(_Y,_E). '固有の文字'(['固有の文字']). '入っている'(['入っている'],U,S) :- mem(U,S). '文字化け'(['文字化け']).

```
'起こす'(['起こす'],_U,_V).
list-sum( [], 0 ).
list-sum( [_X|Y], Z):- list-sum( Y, Z2), Z is 1 + Z2.
mem(A,B) :- is_list(A),member(M,A),!,H =.. [M],memberchk(H,B).
```

回答文番号(039-3)の Prolog 表記

```
's039-3' :-
    'JISX0201'(_W),'半角カタカナ'(R),'ASCII コード'(_S),'部分'(P),
    '格納する'(_E3,R,P),
    'ヨーロッパ'(Y),'では'(Y,E4),
    '固有の文字'(U),'入っている'(E4,U,P),
    '文字化け'(V),'起こす'(_E5,U,V).
```

3.3.1 実行の様子

次は，実行の様子である．

回答文番号(039-3)の実行の様子

```
[yositake@ai Prolog]$ pl  -s  wk.pl
... 省略 ...
?- [s039].
% s039 compiled 0.01 sec, 2,056 bytes
true.
?- 's039-3'.
false .
?-
```

回答文番号(039) 全体の Prolog 記述を読み込ませてから，回答文番号(039-3)である 's039-3' を実行させている．今までの回答文とは異なり，結果として false が表示されている．これは，回答文番号(039-3) が間違った記述であることを意味している．誤っているのは下記の部分である．「半角カタカナ」を格納しているコードは，「ASCII コード」ではなく「JISX0201」である．

回答文番号(039-3)の誤りの個所
(039-3-bad)半角カタカナを格納する ASCII コード部分に

(039-3-bad)の Prolog 表記
's039-3-bad' :- '半角カタカナ'(R),'ASCII コード'(A),'部分'(A,G,P),'格納する'(_E3,R,P).

3.3.2　正しい内容を検出させた例

(039-3-bad)を修正して次の正しい内容を入力してみる.

正しい内容である(039-3-good)
(039-3-good) 半角カタカナを格納する JISX0201 部分に

正しい内容である(039-3-good)の Prolog 表記
's039-3-good' :- '半角カタカナ'(R),'JISX0201'(A),'部分'(A,G,P),'格納する'(_E3,R,P).

正しい内容である(039-3-good) の実行の様子
[yositake@ai Prolog]$ pl -s wk.pl ... 省略 ... ?- [s039]. % s039 compiled 0.01 sec, 2,056 bytes true. ?- '半角カタカナ'(R),'ASCII コード'(A),'部分'(A,G,P),'格納する'(_E3,R,P). false. ?- '半角カタカナ'(R),'JISX0201'(A),'部分'(A,G,P),'格納する'(_E3,R,P).

```
R = ['半角カタカナ'],
A = ['JISX0201', 'ASCII コード', '半角カタカナ'],
G = ['JISX0201', 'ASCII コード', '半角カタカナ'|_G414],
P = ['JISX0201', 'ASCII コード', '半角カタカナ'],
_E3 = ['格納する'] .

?-
```

この結果として，正しい内容である（039-3-good）に対して，'部分' の変数 G に値が求まっており，（039-3-good）がシステムによって正しいと判断されているのが分かる．

4．回答文番号 183 の SDRT による意味記述

まず，記号◆と※を除去した．そして，箇条書きの１行が１つの文であると仮定した．全角数字と半角数字が混在しているので，数字を全て半角数字に変更した．また，セミコロン ； は，主語を示す助詞の代りに使われていると仮定した．括弧で囲まれた文は，括弧から取り出して単独な文として取り扱った．その結果，分析に使用した回答文は，次の８つの単文から構成された談話となる．

分析に使用した回答文番号 183
(183-1) 日本の文字コードの種類
(183-2) シフト JIS;日本のパソコン用の文字コードとして作られた．
(183-3) EUC-JP;UNIX 用
(183-4) ISO-2022-JP;電子メール用
(183-5) 最上位ビットは 0，つまり 7bit でおさまる．
(183-6) 半角カタカナと呼ばれるコードの問題点
(183-7) 電子メールに，JISX0201 の 8 単位符号表の右半分が入っていると，表示の際に文字化けを起こすものが多い．
(183-8)インターネットの世界では使用禁止．

これらの文の SDRT による意味記述は次の通りになる.

4.1 回答文番号(183-1)の意味記述

分析に使用した回答文番号 (183-1)
(183-1) 日本の文字コードの種類

日本語の構文的には, [日本の]が[文字コード]に係るのか, それとも, [種類]に係るのか, という曖昧さがあるのだが, 「読点が入っていない場合は直近の語に係る」という暗黙の了解を適用することにした. つまり, [日本の]は[文字コード]に係るとした.

回答文番号(183-1) の SDRT による意味記述
π 183-1: [x, y, z | 日本(x) ∧ 文字コード(y) ∧ R1(x, y) ∧ 種類(z) ∧ R2(y, z)]

次に, [の種類] の意味記述であるが, 直前の項を列挙していると捉えることにした. その結果, 次の Prolog 記述が得られる.

回答文番号(183-1) に関する世界知識の Prolog 表記
'日本'(['日本']). '文字コード'(I) :- 'コード'(I). 'コード'(Z) :- 'ASCII コード'(Z). 'コード'(Z) :- 'JISX0201'(Z). 'コード'(Z) :- 'ISO-8859-1'(Z). 'コード'(Z) :- '漢字コード'(Z). 'の種類'(Z) :- writeq(Z).

回答文番号(183-1)の Prolog 表記

```
's183-1' :-
    '日本'(X),'Relation'(6,X,Y),'文字コード'(Y), 'の種類'(Y).
```

4.1.1 実行の様子

次は，実行の様子である．

回答文番号(183-1)の実行の様子

```
[yositake@ai Prolog]$ pl  -s  wk.pl
... 省略 ...
?- [s183].
% s039 compiled 0.01 sec, 2,056 bytes
true.
?- 's183-1'.
true.

?- '日本'(X),'文字コード'(Y),'Relation'(6,X,Y),'の種類'(Y).
X = ['日本'],
Y = ['JISX0201', 'ASCII コード', '半角カタカナ', '日本'] ;
X = ['日本'],
Y = ['JISX0208', 'JIS', '日本'] ;
X = ['日本'],
Y = ['SJIS', '8bit', 'パソコン', '実用的', '日本'] ;
X = ['日本'],
Y = ['EUC-JP', '8bit', 'UNIX', '実用的', '日本'] ;
X = ['日本'],
Y = ['ISO-2022-JP', '7bit', '電子メール', '実用的', '日本'].

?-
```

回答文番号(183) 全体の Prolog 記述を読み込ませてから，回答文番号(183-1)である 's183-1' を実行させている．なお，上記では，この結果として true が表示されただけである．そこで，得られた具体的な種類を見るために，'s183-1' の Prolog 表記を直接，入力し，マッチングした変数を表示させている．変数 Y に種類としての答えが得られているのが分かる．

この回答文番号(183-1)は，単に列挙させているだけであるので，質問応答による正解チェックは行っていない．

4.2 回答文番号(183-2)の意味記述

分析に使用した回答文番号 (183-2)
(183-2)　シフト JIS;日本のパソコン用の文字コードとして作られた．

(183-2)では，「A 用の B」という日本語の意味解釈を行わなければならない．ここでは，前稿で議論したように，表１の所有・所属という解釈と同様だと解釈する．これに従って，(183-2)の意味記述は，次の通りになる．

回答文番号(183-2) の SDRT による意味記述
π183-2:［f,g,h,i ｜シフト JIS (f) ∧ 日本(g) ∧ パソコン(h) ∧ R3(g,h) ∧ 文字コード(i) ∧ 用の(h,i) ∧ 作る(e1,f,i)］

その結果，次の Prolog 記述が得られる．

回答文番号(183-2) に関する世界知識の Prolog 表記
'シフト JIS'(F) :- 'SJIS'(F). '日本'(['日本']). 'パソコン'(['パソコン']). '文字コード'(I) :- 'コード'(I). '用の'(H,I) :- 'Relation'(5,H,I). 'として'(F,I) :- mem(F,I). '作る'(_E1,_NULL,_I).

```
list-sum( [], 0 ).
list-sum( [_X|Y], Z):- list-sum( Y, Z2), Z is 1 + Z2.
mem(A,B) :- is_list(A),member(M,A),!,H =.. [M],memberchk(H,B).
```

回答文番号(183-2)の Prolog 表記

```
's183-2' :- 'シフトJIS'(F),'日本'(G),'パソコン'(H),
'Relation'(6,G,H),'文字コード'(I),'用の'(H,I),'として'(F,I),'作る
'(_E1,_NULL,I).
```

4.2.1 実行の様子

次は，実行の様子である．

回答文番号(183-2)の実行の様子

```
[yositake@ai Prolog]$ pl -s wk.pl
... 省略 ...
?- [s183].
% s039 compiled 0.01 sec, 2,056 bytes
true.
?- 's183-2'.
true .
?-
```

回答文番号(183) 全体の Prolog 記述を読み込ませてから，回答文番号(183-2)である 's183-2' を実行させている．この結果として true が表示されているので，回答文番号(183-2) が正しいことが分かる．

4.2.2 誤った内容を検出させた例

回答文番号(183-2)の内容と矛盾する次の内容を入力してみよう．

誤った内容を含んだ文(183-2-bad)

(183-2-bad) ISO-8859-1は日本のパソコン用の文字コードとして作られた.

誤った内容を含んだ文(183-2-bad) の SDRT による意味記述
π183-2-bad: [f, g, h, i | ISO-8859-1(f) ∧ 日本(g) ∧ パソコン(h) ∧ R3(g, h) ∧ 文字コード(i) ∧ 用の(h, i) ∧ 作る(e1, f, i)]

誤った内容を含んだ文(183-2-bad)の Prolog 表記
's183-2-bad' :- 'ISO-8859-1'(F),'日本'(G),'パソコン'(H), 'Relation'(6,G,H),'文字コード'(I),'用の'(H,I),'として'(F,I),'作る'(_E1,_NULL,I).

誤った内容を含んだ文(183-2-bad) の実行の様子
[yositake@ai Prolog]$ pl -s wk.pl ... 省略 ... ?- [s183]. % s039 compiled 0.01 sec, 2,056 bytes true. ?- 's183-2-bad'. false. ?-

この結果として false が表示されているので, この誤った内容の文が, システムによって誤っていると判別されているのが判る.

4.2.3 質問をしてシステムに回答させた例

ここでは, 日本のパソコン用の文字コードは何かと質問を行う.

質問文(183-2-q)
(183-2-q) 日本のパソコン用の文字コードは何か？

質問文(183-2-q)の Prolog 表記
's183-2-q' :- 'パソコン'(H),'文字コード'(I),'用の'(H,I), '日本'(G), 'Relation'(6,G,I).

質問文(183-2-q)の実行の様子
[yositake@ai Prolog]$ pl -s wk.pl ... 省略 ... ?- [s183]. % s183 compiled 0.01 sec, 2,056 bytes true. ?- 'パソコン'(H),'文字コード'(I),'用の'(H,I), '日本'(G), 'Relation'(6,G,I). H = ['パソコン'], I = ['SJIS', '8bit', 'パソコン', '実用的', '日本'], G = ['日本'] . ?-

この結果として,変数 I に,リスト形式にて,次が求まっているのが分かる.

['SJIS', '8bit', 'パソコン', '実用的', '日本'],

つまり, 文字コードの名前である第1番目の項に 'SJIS' が表示されており, 正しく求まっているのが分かる.

4.3 回答文番号(183-3)の意味記述

分析に使用した回答文番号 (183-3)
(183-3) EUC-JP;UNIX 用

(183-3)では, 前稿とは違い,「EUC-JP」の属性の中に「UNIX」があると捉え

ることにする．その結果，(183-3)の意味記述は，次の通りになる．

回答文番号(183-3) の SDRT による意味記述
π183-3: [j,k ｜ EUC-JP(j) ∧ UNIX(k) ∧用の(k,j)]

その結果，次の Prolog 記述が得られる．

回答文番号(183-3) に関する世界知識の Prolog 表記
'EUC-JP'(['EUC-JP','8bit','UNIX','実用的','日本']). 'UNIX'(['UNIX']). '用の'(H,I) :- 'Relation'(5,H,I).

回答文番号(183-3)の Prolog 表記
's183-3' :- 'EUC-JP'(J),'UNIX'(K),'用の'(K,J).

4.3.1　実行の様子

次は，実行の様子である．

回答文番号(183-3)の実行の様子
[yositake@ai Prolog]$ pl -s wk.pl ... 省略 ... ?- [s183]. % s183 compiled 0.01 sec, 2,056 bytes true. ?- 's183-3'. true . ?-

回答文番号(183) 全体の Prolog 記述を読み込ませてから，回答文番号(183-3)である 's183-3' を実行させている．この結果として true が表示さ

れているので，回答文番号(183-3) が正しいことが分かる．

4.3.2 誤った内容を検出させた例

回答文番号(183-3)の内容と矛盾する次の内容を入力してみよう．

誤った内容を含んだ文(183-3-bad)
(183-3-bad) シフト JIS は UNIX 用である．

誤った内容を含んだ文(183-3-bad) の SDRT による意味記述
π183-3-bad: [j,k ｜ シフト JIS(j) ∧ UNIX(k) ∧用の(k,j)]

誤った内容を含んだ文(183-3-bad)の Prolog 表記
's183-3-bad' :- 'シフト JIS'(J),'UNIX'(K),'用の'(K,J).

誤った内容を含んだ文(183-3-bad) の実行の様子
[yositake@ai Prolog]$ pl -s wk.pl ... 省略 ... ?- [s183]. % s039 compiled 0.01 sec, 2,056 bytes true. ?- 'シフト JIS'(J),'UNIX'(K),'用の'(K,J). false. ?-

この結果として false が表示されているので，この誤った内容の文が，システムによって誤っていると判別されているのが判る．

4.3.3 質問をしてシステムに回答させた例

ここでは，UNIX用の文字コードは何かと質問を行う．

質問文(183-2-q)
(183-3-q) UNIX用の文字コードは何か？

質問文(183-3-q)の Prolog 表記
's183-3-q' :- 'UNIX'(K),'文字コード'(I),'用の'(K,I).

質問文(183-3-q)の実行の様子
[yositake@ai Prolog]$ pl -s wk.pl ... 省略 ... ?- [s183]. % s183 compiled 0.01 sec, 2,056 bytes true. ?- 'UNIX'(K),'文字コード'(I),'用の'(K,I). K = ['UNIX'], I = ['EUC-JP', '8bit', 'UNIX', '実用的', '日本'] . ?-

この結果として,変数 I に,リスト形式にて,次が求まっているのが分かる．

['EUC-JP', '8bit', 'UNIX', '実用的', '日本']

つまり，文字コードの名前である第１番目の項に 'EUC-JP' が表示されており，正しく求まっているのが分かる．

4.4 回答文番号(183-4)の意味記述

分析に使用した回答文番号 (183-4)

(183-4) ISO-2022-JP;電子メール用

(183-4)の意味記述は，(183-3)と同様に，次の通りになる．

回答文番号(183-4) の SDRT による意味記述
π183-4: [j,k | ISO-2022-JP (j) ∧電子メール(k) ∧用の(k,j)]

その結果，次の Prolog 記述が得られる．

回答文番号(183-4) に関する世界知識の Prolog 表記
'ISO-2022-JP'(['ISO-2022-JP','7bit','電子メール','実用的','日本']). '電子メール'(['電子メール']). '用の'(H,I) :- 'Relation'(5,H,I).

回答文番号(183-4)の Prolog 表記
's183-4' :- 'ISO-2022-JP'(J),'電子メール'(K),'用の'(K,J).

4.4.1 実行の様子

次は，実行の様子である．

回答文番号(183-4)の実行の様子
[yositake@ai Prolog]$ pl -s wk.pl ... 省略 ... ?- [s183]. % s183 compiled 0.01 sec, 2,056 bytes true. ?- 's183-4'. true . ?-

回答文番号(183) 全体の Prolog 記述を読み込ませてから，回答文番号(183-4)である 's183-4' を実行させている．この結果として true が表示されているので，回答文番号(183-4) が正しいことが分かる．
なお,回答文番号 (183-4)は,回答文番号 (183-3)と似た文になっているので,(183-4)の,誤った内容を検出させた例と,質問をしてシステムに回答させた例については，割愛する．

4.5 回答文番号(183-5)の意味記述

分析に使用した回答文番号 (183-5)
(183-5) 最上位ビットは0，つまり7bitでおさまる．

(183-5)は重文になっており，「つまり」という接続助詞が用いられている．「つまり」の解釈としては，表4における命題間の関係の Elaboration 又は Explanation に該当すると思われるが，ここでは，単に論理積∧で結合することにする．その結果，(183-5)の意味記述は，次の通りになる．

回答文番号(183-5) の SDRT による意味記述
π183-5: [q,r,s,e2 |最上位ビット(q) ∧ 0(r) ∧ =(q,r)∧ 7bit(s) ∧ おさまる(e2,q,s)]

その結果，次の Prolog 記述が得られる．

回答文番号(183-5) に関する世界知識の Prolog 表記
'最上位ビット'(T) :- '0'(T);'1'(T). '0'([0]). '1'([1]).

```
'7bit'(['7bit',0]).
'8bit'(['8bit',1]).
'おさまる'('おさまる',Q,S) :- subset(S,Q).
```

回答文番号(183-5)の Prolog 表記

```
's183-5' :- '7bit'(S),'最上位ビット'(Q),
   'Relation'(1,S,Q),'0'(R),subset(R,Q).
```

4.5.1 実行の様子

次は，実行の様子である．

回答文番号(183-5)の実行の様子

```
[yositake@ai Prolog]$ pl  -s  wk.pl
... 省略 ...
?- [s183].
% s039 compiled 0.01 sec, 2,056 bytes
true.
?- 's183-5'.
true .
?-
```

回答文番号(183) 全体の Prolog 記述を読み込ませてから，回答文番号(183-5)である 's183-5' を実行させている．この結果として true が表示されているので，回答文番号(183-5) が正しいことが分かる．

4.5.2 誤った内容を検出させた例

回答文番号(183-5)の内容と矛盾する次の内容を入力してみよう．

誤った内容を含んだ文(183-5-bad)

(183-5-bad) 最上位ビットは0，つまり 8bit でおさまる．

誤った内容を含んだ文(183-5-bad) の SDRT による意味記述
π183-5-bad: [q, r, s, e2 |最上位ビット(q) ∧ 0(r) ∧ =(q, r)∧ 8bit(s) ∧ おさまる(e2, q, s)]

誤った内容を含んだ文(183-5-bad)の Prolog 表記
's183-5-bad' :- '8bit'(S),'最上位ビット'(Q), 'Relation'(1,S,Q),'0'(R),subset(R,Q).

誤った内容を含んだ文(183-5-bad) の実行の様子
[yositake@ai Prolog]$ pl -s wk.pl ... 省略 ... ?- [s183]. % s039 compiled 0.01 sec, 2,056 bytes true. ?- '8bit'(S),'最上位ビット'(Q),'Relation'(1,S,Q),'0'(R),subset(R,Q). false. ?-

この結果として false が表示されているので，この誤った内容の文が，システムによって誤っていると判別されているのが判る．

4.5.3　質問をしてシステムに回答させた例

ここでは，7bit の最上位ビットは何かと質問を行う．

質問文(183-5-q)
(183-5-q)　7bit の最上位ビットは何か？

質問文(183-5-q)の Prolog 表記

```
's183-5-q' :- '7bit'(S),'最上位ビット'(Q),'Relation'(1,S,Q).
```

質問文(183-5-q)の実行の様子

```
[yositake@ai Prolog]$ pl  -s  wk.pl
... 省略 ...
?- [s183].
% s183 compiled 0.01 sec, 2,056 bytes
true.
?- '最上位ビット'(Q).
Q = [0] ;
Q = [1].

?- '7bit'(S),'最上位ビット'(Q),'Relation'(1,S,Q).
S = ['7bit', 0],
Q = [0] ;
false.

?-
```

まず，'最上位ビット'(Q) と入力することにより，最上位ビットとして取り得る値 Q を調べている．変数 Q の値として，最初に 0 が求まり，そこでセミコロン ; を入力して別の解を探させると 1 が求まっている．次に，'7bit'(S),'最上位ビット'(Q),'Relation'(1,S,Q). と入力することにより，7 bit の最上位ビットは何かを調べている．その結果として，変数 Q に 0 が求まっているのが分かる．そこで，そこでセミコロン ; を入力して別の解を探させると false になっている．これにより，変数 Q が 0 のみであることが分かる．

4.6 回答文番号(183-6)の意味記述

分析に使用した回答文番号 (183-6)
(183-6) 半角カタカナと呼ばれるコードの問題点

(183-6)では「呼ばれる」つまり「呼ぶ」の受身形の意味記述を考える必要がある.

回答文番号(183-6) の SDRT による意味記述
π183-6: [t,u,v,e3 |半角カタカナ(t) ∧ コード(u) ∧ 呼ぶ(e3,u,t) ∧ 問題点(v) ∧ R4(u,v)]

ここで, 前稿では,呼ぶ(e3,u,t)は=(u,t)と同じ解釈が可能だと仮定したが, 考察の結果, より正確には t の一部が u であると解釈すべきだと判断した. 次に,[の問題点] の意味記述であるが, 直前の項の一部を列挙していると捉えることにした. というのが, この段階では, 問題点の具体的な内容は記述されていないので, 単に, 半角カタカナと呼ばれるコードの性質を列挙することしか出来ないのである.

その結果, 次の Prolog 記述が得られる.

回答文番号(183-6) に関する世界知識の Prolog 表記
'半角カタカナ'(['半角カタカナ']). '言う'(Z,Q):- subset(Z,Q). 'コード'(Z) :- 'ASCII コード'(Z). 'コード'(Z) :- 'JISX0201'(Z). 'コード'(Z) :- 'ISO-8859-1'(Z). 'コード'(Z) :- '漢字コード'(Z). '問題点'(_D).

回答文番号(183-6)の Prolog 表記
's183-6' :- '半角カタカナ'(T),'コード'(U),'言う'(T,U), 'の問題点'(V,U).

4.6.1 実行の様子

次は，実行の様子である．

回答文番号(183-6)の実行の様子

```
[yositake@ai Prolog]$ pl  -s  wk.pl
... 省略 ...
?- [s183].
% s039 compiled 0.01 sec, 2,056 bytes
true.
?- 's183-6'.
true.

?- '半角カタカナ'(T),'コード'(U),'言う'(T,U),'の問題点'(V,U).
T = ['半角カタカナ'],
U = ['JISX0201', 'ASCII コード', '半角カタカナ', '日本'],
V = 'JISX0201' ;
T = ['半角カタカナ'],
U = ['JISX0201', 'ASCII コード', '半角カタカナ', '日本'],
V = 'ASCII コード' ;
T = ['半角カタカナ'],
U = ['JISX0201', 'ASCII コード', '半角カタカナ', '日本'],
V = '半角カタカナ' ;
T = ['半角カタカナ'],
U = ['JISX0201', 'ASCII コード', '半角カタカナ', '日本'],
V = '日本' ;
false.

?-
```

回答文番号(183) 全体の Prolog 記述を読み込ませてから，回答文番号(183-6)である 's183-6' を実行させている．なお，上記では，この結果として true が表示されただけである．そこで，得られた具体的な問題点を見るために，'s183-6' の Prolog 表記を直接，入力し，マッチングした変数を表示させている．変数 V に半角カタカナに関係する項目が得られているのが分かる．

但し，この段階では問題点ではなく，あくまで，半角カタカナに関係する項目が表示されているだけである．

この回答文番号(183-6)は，単に列挙させているだけであるので，質問応答による正解チェックは行っていない．

4.7 回答文番号(183-7)の意味記述

分析に使用した回答文番号 (183-7)
(183-7) 電子メールに，JISX0201 の 8 単位符号表の右半分が入っていると，表示の際に文字化けを起こすものが多い．

(183-7)は重文になっており，「と」という接続助詞が用いられている．「と」の解釈としては，表４における命題間の関係の Result に該当すると思われるが，ここでは，単に論理積∧で結合することにする．また，量子化の問題は取り扱わないことにしているので，「ものが多い」の意味記述は無視する．その結果，前稿で得た(183-7)の意味記述は，次の通りであった．

回答文番号(183-7) の SDRT による意味記述 (前稿)
π 183-7: [a, b, c, d, e4, ff | 電子メール(a) ∧ JISX0201(b) ∧ 8 単位符号表(c) ∧ 右半分(d) ∧ R5(b, c) ∧ R6(c, d) ∧ 入っている(e4, d, a) ∧ 起こす(e4, a, ff) ∧ 文字化け(ff)]

「JISX0201 の 8 単位符号表」の「の」は，前出の７つの「の」の解釈には該当しない．例えば，『テニスの趣味』に似ている．ここでの「A の B」は，B という代表名の中の１つの具体例を A を指している，と考える．つまり，B の一部が A である．これを，新たに，Relatin7 として定義した．「JISX0201…の右半分が」の「A の右半分」は，「A の一部」と解釈する．「表示の際に」の意味記述は，「文字化け」の意味に完全に含まれるので，ここでは「表示の際に」の

意味記述をしていない．この結果，(183-7)の意味記述は，次の通りになる．

回答文番号(183-7) の SDRT による意味記述 (改訂版)
π183-7: [a, b, c, d, e4, f | 電子メール(a) ∧ JISX0201(b) ∧ 8単位符号表(c) ∧ 右半分(d) ∧ Relation(4, b, c) ∧ R6(c, d) ∧ 入っている(e4, d, a) ∧ 起こす(e4, a, ff) ∧ 文字化け(f)]

その結果，次の Prolog 記述が得られる．

回答文番号(183-7) に関する世界知識の Prolog 表記
'電子メール'(['電子メール']). 'JISX0201'(['JISX0201','ASCII コード','半角カタカナ']). '8単位符号表'(C) :- 'JISX0201'(C);'ISO-8859-1'(C). 'の右半分'(B,P) :- member(P,B). '適合する'('文字化けする',A,P) :- mem(A,P). list-sum([], 0). list-sum([_X|Y], Z):- list-sum(Y, Z2), Z is 1 + Z2. mem(A,B) :- is_list(A),member(M,A),!,H =.. [M],memberchk(H,B).

回答文番号(183-7)の Prolog 表記
's183-7' :- 'JISX0201'(B),'8単位符号表'(C),'Relation'(7,B,C), 'の右半分'(B,P), '電子メール'(A),not('適合する'(_E,A,P)).

4.7.1 実行の様子

次は，実行の様子である．

回答文番号(183-7)の実行の様子

```
[yositake@ai Prolog]$ pl -s wk.pl
... 省略 ...
?- [s183].
% s039 compiled 0.01 sec, 2,056 bytes
true.
?- 's183-7'.
true .
?-
```

回答文番号(183) 全体の Prolog 記述を読み込ませてから，回答文番号(183-7)である 's183-7' を実行させている．この結果として true が表示されているので，回答文番号(183-7) が正しいことが分かる．

4.7.2 誤った内容を検出させた例

回答文番号(183-7)の内容で，「JISX0201」を「シフト JIS」に変えた文を入力してみよう．

誤った内容を含んだ文(183-7-bad)
(183-7-bad) 電子メールに，シフト JIS の 8 単位符号表の右半分が入っていると，表示の際に文字化けを起こすものが多い．

誤った内容を含んだ文(183-7-bad) の SDRT による意味記述
π183-7-bad: [a, b, c, d, e4, f |電子メール(a) ∧シフト JIS(b) ∧ 8 単位符号表(c) ∧右半分(d) ∧ Relation(4, b, c) ∧ R6(c, d) ∧ 入っている(e4, d, a)∧ 起こす(e4, a, ff) ∧ 文字化け(f)]

誤った内容を含んだ文(183-7-bad)の Prolog 表記
's183-7-bad' :- 'シフト JIS '(B),'8 単位符号表'(C),'Relation'(7,B,C), 'の右半分'(B,P), '電子メール'(A),not('適合する'(_E,A,P)).

誤った内容を含んだ文(183-7-bad) の実行の様子

```
[yositake@ai Prolog]$ pl -s wk.pl
... 省略 ...
?- [s183].
% s039 compiled 0.01 sec, 2,056 bytes
true.
?- 'シフト JIS'(B),'8 単位符号表'(C),'Relation'(7,B,C),'の右半分'(B,P),'
電子メール'(A),not('適合する'(_E,A,P)).
false.
?-
```

この結果として false が表示されているので，この誤った内容の文が，システムによって誤っていると判別されているのが判る．

4.7.3 質問をしてシステムに回答させた例

ここでは，電子メール用の文字コードは何かと質問を行う．

質問文(183-7-q)

```
(183-7-q) 電子メール用の文字コードは何か？
```

質問文(183-7-q)の Prolog 表記

```
's183-7-q' :- '文字コード'(P),'電子メール'(A),'適合する'(_E,A,P).
```

質問文(183-7-q)の実行の様子

```
[yositake@ai Prolog]$ pl -s wk.pl
... 省略 ...
?- [s183].
% s183 compiled 0.01 sec, 2,056 bytes
true.
?- '文字コード'(P),'電子メール'(A),'適合する'(_E,A,P).
```

```
P = ['ISO-2022-JP', '7bit', '電子メール', '実用的', '日本'],
A = ['電子メール'],
_E = '文字化けする'.

?-
```

この結果として，変数 P に 'ISO-2022-JP' が求まっているのが分かる．

4.8 回答文番号(183-8)の意味記述

分析に使用した回答文番号 (183-8)
インターネットの世界では使用禁止.

(183-8)では，助詞「では」の意味記述が必要になる．ここでは，単に論理積∧で結合することにする．その結果，(183-8)の意味記述は，次の通りになる．

回答文番号(183-8) の SDRT による意味記述
π183-8: [a1, b1, c1, e5 | インターネット(a1) ∧ 世界(b1) ∧ R7(a1, b1) ∧ 使用(c1) ∧ 禁止する(e5, d1, c1)]

その結果，次の Prolog 記述が得られる．

回答文番号(183-8) に関する世界知識の Prolog 表記
'半角カタカナ'(['半角カタカナ']). 'インターネット'(['インターネット']). '世界'(B) :- 'コード'(B). '禁止する'(_E, H, A) :- not(subset(H, A)).

回答文番号(183-8)の Prolog 表記

```
's183-8' :- 'インターネット'(A),'世界'(B),'Relation'(7,A,B),
            '半角カタカナ'(H),'禁止する'(_E,H,B).
```

4.8.1 実行の様子

次は，実行の様子である．

回答文番号(183-8)の実行の様子

```
[yositake@ai Prolog]$ pl  -s  wk.pl
... 省略 ...
?- [s183].
% s039 compiled 0.01 sec, 2,056 bytes
true.
?- 's183-8'.
true .
?-'インターネット'(A),'世界'(B),'Relation'(7,A,B),'半角カタカナ'(H),'禁
止する'(_E,H,B).
A = ['インターネット'],
B = ['ISO-8859-1', 'ASCII コード', 'インターネット', 'ヨーロッパ'],
H = ['半角カタカナ'] ;
A = ['インターネット'],
B = ['SJIS', '8bit', 'パソコン', '実用的', 'インターネット', '日本'],
H = ['半角カタカナ'] ;
A = ['インターネット'],
B = ['EUC-JP', '8bit', 'UNIX', '実用的', 'インターネット', '日本'],
H = ['半角カタカナ'] ;
A = ['インターネット'],
B = ['ISO-2022-JP', '7bit', '電子メール', '実用的', 'インターネット', '
日本'],
H = ['半角カタカナ'].

?-
```

回答文番号(183) 全体の Prolog 記述を読み込ませてから，回答文番号

(183-8)である　's183-8' を実行させると true が表示されているのが分かり，回答文番号(183-8) が正しいことが分かる．ここで，回答文番号(183-8)の Prolog 記述を直接，入力すると，変数 B に，インターネットの世界で使用可能なコードが得られているのが分かる．

4.8.2　誤った内容を検出させた例

ここでは，インターネットの世界で半角カタカナが使用可能である，という文を処理する．

誤った内容を含んだ文(183-8-bad)
(183-8-bad) インターネットの世界では半角カタカナが使用可能である．

誤った内容を含んだ文(183-8-bad) の SDRT による意味記述
π183-8-bad: [a1,b1,c1,e1 | インターネット(a1) ∧世界(b1) ∧ R7(a1,b1) ∧ 半角カタカナ(e1) ∧ 使用(c1)]

誤った内容を含んだ文(183-8-bad)の Prolog 表記
's183-8-bad' :- 'インターネット'(A),'世界'(B),'Relation'(7,A,B), '半角カタカナ'(H),subset(H,B).

誤った内容を含んだ文(183-8-bad) の実行の様子
[yositake@ai Prolog]$ pl -s wk.pl ... 省略 ... ?- [s183]. % s039 compiled 0.01 sec, 2,056 bytes true. ?- 'インターネット'(A),'世界'(B),'Relation'(7,A,B),'半角カタカナ'(H),subset(H,B). false. ?-

この結果として false が表示されているので，この誤った内容の文が，システムによって誤っていると判別されているのが判る．

4.8.3 質問をしてシステムに回答させた例

ここでは，インターネットの世界で使用可能なコードは何かと質問を行う．

質問文(183-8-q)
(183-8-q) インターネットの世界で使用可能なコードは何か？

(183-8-q)の Prolog 表記
's183-8-q' :- 'インターネット'(A),'世界'(B),'Relation'(7,A,B).

質問文(183-8-q)の実行の様子

```
[yositake@ai Prolog]$ pl  -s  wk.pl
... 省略 ...
?- [s183].
% s183 compiled 0.01 sec, 2,056 bytes
true.
?- 'インターネット'(A),'世界'(B),'Relation'(7,A,B).
A = ['インターネット'],
B = ['ISO-8859-1', 'ASCII コード', 'インターネット', 'ヨーロッパ'] ;
A = ['インターネット'],
B = ['SJIS', '8bit', 'パソコン', '実用的', 'インターネット', '日本'] ;
A = ['インターネット'],
B = ['EUC-JP', '8bit', 'UNIX', '実用的', 'インターネット', '日本'] ;
A = ['インターネット'],
B = ['ISO-2022-JP', '7bit', '電子メール', '実用的', 'インターネット', '
日本'].

?-
```

この結果として，変数 B に４種類の文字コードが求まっているのが分かる．

５．検討

前稿において挙げた問題点に関して，実際に SDRT の記述を行った結果を踏まえて考察を行う．

5.1　前稿で挙げた問題への対応

「述語の否定」と「反対の意味の述語」は，not 述語を使用した．「意味の包含関係」は，member 述語と subset 述語を用いた．全体としての意味記述については，世界知識を導入し，更に，処理する文の意味を全て蓄積することにより，全体としての矛盾が発生しないようにした．

5.2　原文の曖昧さ

今回，学生が回答した文を SDRT 表記として記述を行ったが，原文に曖昧さが多いことに驚いた．今回，使用した２文の内の，回答文番号 039 は，回答全体を通して文章となっているので，曖昧さは，ほとんど感じなかった．しかし，回答文番号 183 は，箇条書きの形式をとった回答であり，名詞の羅列に近いものであった．つまり，名詞間の関係については，漠然としたものであった．しかし，名詞の羅列であっても，SDRT 表記を行えば，適切な意味処理を行うことが出来るという感触を得た．前稿で挙げた「回答者が HTML コード に含めた意味は存在しないのか」という問題点は，特に，HTML コードによる部分を意味記述しなくても十分であると思えた．

逆に考えれば，きちんとした文章と箇条書きとの結果の差を感じていないということにより，述語に関する SDRT 記述が不十分なのか，述語に関する意味記述そのものが不要なのか，という検討が必要になろう．今回の SDRT 表記では，述語の意味表記は，名詞間の関係のみとしている．これで旨くいっているのは，扱っているテーマが静的なものであるので，述語の意味記述が簡素か，又は，全くない，としても，全体の意味処理が行えているのであろう．

5.3 原文の内容の誤り

回答文番号(039-3)は，人間が採点を行う場合は，正しい文と捉えてしまいがちである．実際，筆者も，回答文番号(039-3) を正しい内容の文と思ってSDRT 記述を行ったつもりであった．しかし，コンピュータ・プログラムからの出力結果が false になるので，熟考を加えたら，原文の内容に誤りがあることに気がついた．

しかし，ここで注意が必要である．本コンピュータ処理では，文字化された文章に対して，字面だけを捉えた処理を行っているが，現実世界で，実際に複数の文を処理する場合は，採点者が回答者の意図をくみ取って解釈することがある．つまり，字面だけの誤りをもって回答が間違っていると判断するのか，字面は間違っているが回答者は正しく理解していると判断するのか，という問題が生じる．後者は未だコンピュータ処理が難しいので，当面は，前者の取り扱いで行かざるを得ない．

5.4 MDC(Maximize Discourse Coherence) 処理

前稿で述べたように SDRT の考え方では，曖昧なものは後の MDC(Maximize Discourse Coherence) 処理に回す，としている．本研究では，MDC 処理は Prolog のバック・トラッキングに任せている．単語の意味記述の段階では詳細な意味を記述しておらず，多くの曖昧さを残しているが，後の MDC 処理としての Prolog のバック・トラッキングによって，曖昧さが解消されている．

5.5 量化子(quantifier)

本研究の前提条件として，量化子(quantifier) は取り扱わないことにしていた．しかし，「...が多い」や「の種類」や「の問題点」などの意味記述を考える必要があった．今回は，これらを簡単な包含関係で記述したが，旨く処理できているようである．

5.6 未処理の問題点

以下は未だ未着手のまま残った．

・時制の取り扱い.
・文の間の関係を Narration（語り）と仮定したのは荒すぎないのか？
・類似度の計算式は妥当か？

6．あとがき

本研究は，前項(吉武, 2007)で考察した SDRT 意味記述を，実際にコンピュータ・プログラムとして実現させたものである．今回，使用したサンプル文の範疇では，今回の実験は，旨く処理できていた．今後は，もっと複雑な文章や，多量の文章に対して SDRT 表記を行い，問題点の発見に努めたい．

参考論文

Asher Nicholas, Alex Lascarides(2005). Logics of Conversation. Cambridge University Press.

Pustejovsky James(1995). The Generative Lexicon. MIT Press.

菊池隆典，白井英俊(2004)．″文脈に応じた名詞句の意味解釈の検討.″第6回年次国際大会ハンドブック 言語科学会.

吉武春光(2002)．″電子メールからのナレッジ・リサイクルの試み.″ 西南学院大学商学論集 49巻 2号,

吉武春光(2007)：″ SDRT による談話の意味記述″，西南学院大学商学論集，53巻，3・4合併号.

黒川利明(1985).Prologのソフトウェア作法,岩波書店.