知識工学と人工知能　46－2
（１９８６．５．９）

Relation Management System

Kiyoshi AKAMA and Masami TAKIKAWA
(Faculty of Letters, Hokkaido University, Sappro-shi, 060, Japan)

Abstract : Relation management system (RMS) is an inductive learning system that receives successively positive and negative relational information (stating that certain objects have a certain relation), selects important information from them, generates new relational knowledge and constructs its knowledge base that reflects the structure of the input information. We constructed a limited version of RMS, which deals with only binary relational input. RMS can improve the information processing for the knowledge acquisition of inductive learning system LS/1, when it works as the intelligent subsystem of LS/1.

関係管理システム

赤間　清，　滝川　雅巳
（北海道大学　文学部）

内容梗概：関係管理システム（ＲＭＳ：Relation Management System）は，いくつかの対象がある関係にある／ないという形の関係情報を逐次的に大量に受け取って，重要な関係情報を選択し，新しい関係知識を生成して，入力情報の構造を表現する知識を作る帰納的学習システムである．我々は，扱う関係情報を２項関係に限定してＲＭＳを実現した．それは，関係知識に付与された重要度を管理することによって，諸アルゴリズムの協調をとり，また，知識の構造化における探索の組合わせ爆発を抑制している．ＲＭＳは，帰納的学習システムＬＳ／１のサブシステムとして用いることによって，ＬＳ／１の知識獲得の情報処理を強力に牽引しうることが確認された．

１．　まえがき

関係管理システム（ＲＭＳ：Relation Management System）は，いくつかの対象がある関係にある／ないという形の関係情報を逐次的に大量に受け取って，重要な関係情報を選択し，新しい関係知識を生成して，入力情報の構造を表現する知識を作る帰納的学習システムである．

一般に，帰納的学習の研究では，正の例の集合Ｐと負の例の集合Ｎが与えられて，

$$P \subset S \subset \overline{N} \qquad \cdots☆$$

なる集合Ｓをもとめる形の定式化がなされることが多い．このときＳは，あらかじめ決められた表現の枠組みＥＸＰの元でなければならない．

$$S \in EXP \qquad \cdots★$$

我々の扱う帰納的学習の問題も基本的にはそのような問題［１］であり，ＰとＮにあたる集合の元が逐次的に与えられる設定である．

しかし☆はＳに対する最低限の条件でしかない．ＥＸＰの選び方によっては，☆と★の問題は，学習対象とかけ離れた構造を無理に汲み取る試みになりかねない．広い範囲の構造を許容しうる表現力の大きいＥＸＰの開発と，その表現の自由度の中からＰとＮに照らして，より良いＳを発見する強力で柔軟なアルゴリズムの探求が必要なのである．そしてさらに，☆と★の問題設定を越える新たな観点と枠組みを発見せねばならない．

ＲＭＳはそのような方向への１つの試みであり，データベースの自動生成にあたる処理を行うサブシステムを帰納的学習システムから切出す形で定式化したものである．データベースは，質問応答システムの中核に位置し，対象世界の意味に直結している．質問応答の会話から自動的にそのエッセンスとなる意味情報をデータベースとして取込むことができれば，それは帰納的学習システムの情報処理を促進する上で非常に有用であろう．これは既存の帰納的学習の設定では未だ取り扱われていない重要な観点である．

2．　帰納的学習システムとデータベース

図１のようなデータベースがあるとする．ただし，ここでは関係名や属性名は省略してある．我々は，このデータベースを利用してさまざまな質問に答えることができる．図２は可能な質問応答の例である．図１の＊印はそのときに使用しないデータである．

```
A:                B:                C:              D:
 DOG    ANIMAL      DOG    RUN        BIRD   FLY       BIRD   WING
 CAT    ANIMAL      CAT    RUN        FISH   SWIM      ANIMAL LEG
 PIG    ANIMAL      DOVE   FLY      * ANIMAL RUN
 HAWK   BIRD     * PIG    RUN
 DOVE   BIRD     * HAWK   FLY
 SALMON FISH     * SALMON SWIM
```

図１　データベース（ A,B,C,Dの４つの関係からなる）

```
(what is dog) ---------------- (it is animal)
(say example of animal) ---------------- (cat)
(salmon is fish) ---------------------- (yes)
(dog can run) ------------------------- (yes)
(hawk is bird) ------------------------ (yes)
(what can bird) ------------------- (can fly)
(dove can fly) ------------------------ (yes)
(what is dove) ------------------ (it is bird)
(say example of bird) ----------------- (dove)
(what is hawk) ------------------ (it is bird)
(what can fish) ------------------- (can swim)
(what is pig) ---------------- (it is animal)
(bird has wing) ----------------------- (yes)
(say example of animal) ---------------- (cat)
(cat can run) ------------------------- (yes)
(animal has leg) ---------------------- (yes)
(which is animal : dog or dove) ------- (dog)
(which is bird : hawk or cat) -------- (hawk)
(which is animal : eagle or pig) ------ (pig)
(which is fish : dove or salmon) --- (salmon)
```

図２　質問応答列

逆に，図２のような質問応答だけが次々に与えらるとき，図１のようなデータベースを帰納的に獲得する学習システムは構築できるだろうか？　ただし，質問応答に用いる言語に対する予備知識や，データベースを構成するときなにが重要な関係かについての知識はまったく仮定しない．何の予備知識もなしに図２のような問答を繰り返して，その文字面から図１のようなデータベースを作り上げねばならない．この問題は，特にデータベースの構築という視点から帰納的学習システム設計問題を再提示したものである．帰納的学習システムの厳密な設定は［３］に述べたものを採用する．

図２の質問応答列と関係 A,B,C,D の間のつながりを考えてみる．質問応答ペアで，

(what is $1)－－－－－(it is $2)

という形を持つものを考えると，$1と$2に入りうる単語の列のペア全体の集合は，

S1={((dog), (animal)),((pig), (animal)),((hawk), (bird)),((dove), (bird))}

となる．これは少々わずらわしい．以下では簡単のために，(dog)をDOG，(animal)をANIMAL…という

ように記述することにする．そのとき，S1は

S1={(DOG, ANIMAL), (PIG, ANIAML), (HAWK, BIRD), (DOVE, BIRD)}

となる．また同様に，

(say example of $1)－－－－($2),

($1 is $2)－－－－－－－－－(yes)

の場合には，それぞれ，

S2={(ANIMAL, CAT), (BIRD, DOVE), (ANIMAL, PIG)}

S3={(SALMON, FISH), (HAWK,BIRD)}

となる．集合S2の各ペアの順序を逆転して集合S2'を得る．

S2'={(CAT, ANIMAL), (DOVE, BIRD), (PIG, ANIMAL)}

関係 A は，(集合として見れば) 集合S1と集合S2'と集合S3の和集合と同一視できる．

A = S1 ∪ S2' ∪ S3

簡潔に記述するためにいくつかの記号を導入する． 質問応答列 QA や関係 A,B,C,D などは，見方によって，関係とも，単語列のペアの集合とも，対応とも見なしうることに注意せよ．本論文では，同じ対象をそのときの取扱方法に応じて，関係と呼んだり，集合と呼んだり，対応と呼んだりするので注意が必要である．まず，単語列のペアの集合Xを単語列のペアの集合Yに変換する対応を定義する．$ で始まるアトムを単語列に対応する変数として，

Y = { ($1, $2) | ((what is $1), (it is $2)) ∈ X },

Y = { ($1, $2) | ((say example of $1), ($2)) ∈ X }

で決まる対応を，それぞれ，

[WHAT_IS_$1→IT_IS_$2],

[SAY_EXAMPLE_OF_$1→$2]

と書くことにする．他の類似の対応についても同様とする．ただし，

[WHICH_IS_$1_:_$2_OR_$3→$2],

[WHICH_IS_$1_:_$3_OR_$2→$2]

は特別で，それぞれ，

Y = { ($1, $2) | ((which is $1 : $2 or $3), ($2)) ∈ X },

Y = { ($1, $2) | ((which is $1 : $3 or $2), ($2)) ∈ X }

とする（これは便宜的な定義である）．また，

Y = { ($2, $1) | ($1, $2) ∈ X }

で決まる対応を，REVで表わす．次に，対応と見て操作するために，対応Xと対応Yの合成がZである（対応Xで写像し，さらに対応Yで写像した結果が対応Zで写像した結果と同じになる）ことを，

Z = Y * X

と表わす．このとき，A, B, C, D と QA の関連（の1例）は次のように書ける．

(1) A = [WHAT_IS_$1→IT_IS_$2](QA)
∪ REV ([SAY_EXAMPLE_OF_$1→$2](QA))
∪ [$1_IS_$2→YES](QA)

(2) B = C * A

(3) C = [$1_CAN_$2→YES](QA) * (REV (A))

(4) D = [$1_HAS_$2→YES] (QA)

質問応答列からデータベースを作り出す問題は，QAが与えられて，(1)－(4)式（または，それに類似の式）を発見して，集合A, B, C, D （またはそれに類似の集合）を求める問題である．この問題には与えられるものに比較して「未知数」が非常に多い．集合QAからその他すべてを発見する必要がある．帰納的学習システムとしては，さらに，得られたデータベースから「推測値」 [QA]（これが一般論のS式にあたる）を求める式をも得ておく必要がある．例えば，

[QA] = [WHAT_IS_$1→IT_IS_$2]$^{-1}$(A)
∪[SAY_EXAMPLE_OF_$1→$2]$^{-1}$(REV(A))
∪[$1_CAN_$2→YES]$^{-1}$(B)
∪[$1_IS_$2→YES]$^{-1}$(A)
∪[WHAT_CAN_$1→CAN_$2]$^{-1}$(C)

∪[WHICH_IS_$1_:_$2_OR_$3→$2]$^{-1}$(REV(A))
∪[WHICH_IS_$1_:_$3_OR_$2→$2]$^{-1}$(REV(A))
がその１例である．

３．　ＲＭＳの定式化と主要な問題点

帰納的学習の枠組みでデータベース獲得の問題に効率的に取り組むために，我々は，帰納的学習システムから，特にデータベース部分を扱う情報処理システムを切出し，それを一般的なレベルで定式化する．それは関係管理システム（ＲＭＳ:Relation management system）と呼ばれる．

(1) ＲＭＳに対して情報（signとrelationとn-tupple）が逐次的に与えられる．
(2) 情報は次の形をしている．
　（p　r　t1　t2　…）　　対象t1，対象t2…がｒの関係にある．
　（n　r　t1　t2　…）　　対象t1，対象t2…がｒの関係にない．
(3) ＲＭＳは，逐次的にそれらの情報に対応するような「データベース」を構成する．
(4) ＲＭＳは次の形の質問を受け付ける．
　対象t1，対象t2…はｒの関係にあるか？
　対象t1，対象t2…の関係はなにか？
　対象t1，対象t2…とｒの関係にある対象tiはなにか？
(5) ＲＭＳは獲得した「データベース」を利用して質問に答える．

ＲＭＳの設定は帰納的学習システムＬＳ／１の設定と相似度が高い．ＲＭＳはそれ自身１つの（自立した）帰納的学習システムであるといえる．ＲＭＳの受け取る情報の組(tupple)の項数(arity)は，いろいろなものが混在していてもよい．関係や成分(compornent)は構造（例えばＳ式で表現される構造）を持つ場合もある．(3)で作るデータベースは，もちろん，既存の関係データベースの形と違って構わない．学習にとってどんな形の表現が必要かを見出して行く研究が重要である．

我々が既に実現した関係管理システムＲＭＳは，帰納的学習システムＬＳ／１のサブシステムとして動いている．ＲＭＳは，帰納的学習システムＬＳ／１のサブシステムとして用いるとき，ＬＳ／１にとって極めて重要な役割を果たす．そのときの両者の結合方式を例で示す．例えば，外界からの質問（what is dog）に対してＬＳ／１が(bird) と答え，(bird)は誤りであって，正解の１つは（it is animal)だと教えられるとき，ＬＳ／１からＲＭＳへの入力情報は，

```
(p ((what is $1) (it is $2))       (dog) (animal))
(p ((what $1 dog) ($2 is animal))  (is) (it))
(n ((what is $1) ($2))             (dog) (bird))
```

などである．これはＬＳ／１からＲＭＳへの教示である．またＬＳ／１は，

```
(p ((what is $1) (it is $2))       (dog) (*what))
```

などと変数（＊で始まるアトム）を用いてＲＭＳに質問し，変数に入るべきものをＲＭＳに答えさせる．もしＲＭＳの性能が良ければ，ＲＭＳは教えられた情報をうまく整理，構造化して，いい情報をＬＳ／１に返すであろう．このようにＬＳ／１はＲＭＳに対して教師／利用者の役割を果たす．それは，外界がＬＳ／１に対して教師／利用者の役割を果たすのと相似である．２つの帰納的学習システムがこのような関係にあることを，ＲＭＳはＬＳ／１の知的サブシステムである，と言う．以下では簡単のために，ＬＳ／１の知的サブシステムとしてのＲＭＳに限定して，関係管理システムの説明を与える．

作成されたＲＭＳの設定は，上記の一般的な定義よりはるかに制限されている．それは項数２だけの組を受け入れる．また，成分は長さ１の単語列だけに限定する．関係や成分の構造に入り込んで分析することもないし，負の例の扱いも十分ではない．

関係管理システムを実現する上での最大の問題点は，重要な関係をどのように発見するかである．ＲＭＳがあまり「意味のない」関係知識を大量に受け取ることに注意せよ．１つの質問と応答のペアに限っても，ＬＳ／１がＲＭＳへ送る関係知識は非常に多い．例えば，

```
(what is dog) －－－－－－－－ (it is animal)
```

という質問と正解のペアから得られる正の例は，成分が長さ１の単語列になるものだけ（現在のＲＭＳはそれしか扱わない）を考えるとしても，次のように${}_6C_2$＝１５個も存在する．

```
(($1 $2 dog) (it is animal))     (what)      (is)
(($1 is $2) (it is animal))      (what)      (dog)
        …省略…
((what is dog) (it $1 $2))       (is)        (animal)
```

一方，我々が常識から考えて「意味のある」関係は，図２の質問列全部に対してでも次の７通りぐらいが考えられるだけである．

(1) ((what is $1)(it is $2)) (2) ((say example of $1)($2)) (3) (($1 is $2)(yes))
(4) (($1 can $2)(yes)) (5) ((what can $1)(can $2)) (6) ((which is $1 : $2 or $3)($2))
(7) ((which is $1 : $2 or $3)($3))

有用なデータベースを構築するためには，「意味がある」とは限らない大量の関係情報から，「意味のある」と思われる関係情報を効果的に絞り込んでいく知識管理が最も重要である．うまく絞らなければそれらの相互関係を見つける探索は爆発するであろう．

４．　ＲＭＳの基本的な情報処理

図２の質問応答列から得られる入力列を処理する場合を例にして，ＲＭＳの基本的な情報処理の概要を記述する．

(a)　恒等関係（包含関係）の発見

２つの集合の同一または包含関係を，それらの共通元の存在を根拠にして推測する．例えば，

```
[WHAT_IS_$1→IT_IS_$2](QA) = {(DOG, ANIMAL),(DOVE, BIRD),(HAWK, BIRD),(PIG, ANIMAL)}
[$1_IS_$2→YES](QA)        = {(SALMON, FISH), (HAWK, BIRD)}
```

が得られた場合，

$$[WHAT\_IS\_\$1 \rightarrow IT\_IS\_\$2](QA) \cap [\$1\_IS\_\$2 \rightarrow YES](QA) = \{ (HAWK, BIRD) \}$$

を手がかりにして，２つの和集合を作り，新たな関係を生成する．

(b)　逆転関係の発見

２つの集合の逆転関係を，それらの共通元の存在を根拠にして推測する．例えば，

```
[WHAT_IS_$1→IT_IS_$2](QA) = {(DOG, ANIMAL),(DOVE, BIRD),(HAWK, BIRD),(PIG, ANIMAL)}
[SAY_EXAMPLE_OF_$1→$2](QA)= {(ANIMAL, CAT), (BIRD, DOVE)}
```

の２つの集合がわかっているとき，

$$[WHAT\_IS\_\$1 \rightarrow IT\_IS\_\$2](QA) \cap REV([SAY\_EXAMPLE\_OF\_\$1 \rightarrow \$2](QA)) = \{ (DOVE, BIRD) \}$$

を根拠に，前者∪REV（後者）を作り，新たな関係を生成する．

これら(a),(b)を合せると，３つの集合を母体として新しい集合 A を生成できる．

$$A \sim [WHAT\_IS\_\$1 \rightarrow IT\_IS\_\$2](QA)$$
$$A \sim [\$1\_IS\_\$2 \rightarrow YES](QA)$$
$$A \sim REV([SAY\_EXAMPLE\_OF\_\$1 \rightarrow \$2](QA))$$
$$A = [WHAT\_IS\_\$1 \rightarrow IT\_IS\_\$2](QA) \cup REV([SAY\_EXAMPLE\_OF\_\$1 \rightarrow \$2](QA)) \cup [\$1\_IS\_\$2 \rightarrow YES](QA)$$

が得られる．以下ではここで生成された A を用いる．

(c)　合成関係の発見

対応Ｘ１，Ｘ２…の合成が対応Ｙになることを，それらの元のレベルの到達関係より推測する．

$$(3)' \quad [WHAT\_CAN\_\$1 \rightarrow CAN\_\$2](QA) \sim [\$1\_CAN\_\$2 \rightarrow YES](QA) * (REV(A))$$

の合成関係を推測する根拠となる到達関係とは，

$$(BIRD, DOVE) \in (REV(A))$$
$$(DOVE, FLY) \in [\$1\_CAN\_\$2 \rightarrow YES](QA)$$
$$(BIRD, FLY) \in [WHAT\_CAN\_\$1 \rightarrow CAN\_\$2](QA)$$

である．(3)'をもとにして，[WHAT_CAN_$1→CAN_$2](QA)を母体にした新しい集合 C を作れば，

$$C \sim [WHAT\_CAN\_\$1 \rightarrow CAN\_\$2](QA)$$
$$C = [\$1\_CAN\_\$2 \rightarrow YES](QA) * (REV(A))$$

を得る．同様にして，

B ～ [$1_CAN_$2→YES](QA)
B ＝ C * A

(d) 集合の生成，管理

対応の定義域，値域，逆像などの演算を用いて集合を生成する．この例では定義域，値域により，

P={DOG, CAT, PIG, HAWK, DOVE}, Q={ANIMAL, BIRD}, R={RUN, FLY}, S={WING, LEG}

が，またそれらの特定の元の逆像から

T={DOG, CAT, PIG}, U={HAWK, DOVE}

が得られる．それらは，包含関係や排反関係を重視して管理される．

(e) 到達可能性判定の高速化

上で得られた排反な集合の族は，合成関係発見における組み合わせ爆発を抑制するために使われる．排反な集合全体の集合を，U＝｛Ｄ１，Ｄ２，Ｄ３，…｝とする．各々の 対応Ｒに，

Ｒ ⊂ Ｄｉ×Ｄｊ

によってペア（Ｄｉ，Ｄｊ）を対応させる．対応の間の合成関係を示す組の具体例が存在する場合，それに応じてＵ上の元の間の到達関係が存在する．従って逆に，合成関係発見のための組の探索を，Ｕ上での到達関係が存在する 対応の組み合わせの範囲だけに限定できる．

(f) 重要度の管理

ＲＭＳの扱うそれぞれの関係には重要度を示す整数を付与する．重要度を付加する狙いを３つ挙げる．第１は，「意味ある」関係を見つける諸アルゴリズムの協調をとることである．「意味のある」構造は単独で存在すると言うより，お互いに関連を持って存在していると考えられる．「意味のある」構造と関連のある構造には「意味がある」のである．従って，それぞれのアルゴリズムが関連を見つけたときに副作用として重要度を更新すれば，重要度は黒板（情報伝達の媒介者）としての役割を果たす．第２は，より「意味ある」関係を優先的に扱うことによって，探索の範囲を制限し，いろいろな対象の間の関係を見出すときに起こりがちな組合わせ爆発を回避することである．第３は，システムの安定性の改善である．重要度の適切な管理をすれば，入力情報の偏りを補い，システムの動作を安定にすることが予想される．入力が誤りを含む可能性は現在の設定では考慮していないが，重要度をうまく操作すれば，システムへの入力の誤りをふるいおとす上で役立つことが期待できる．重要度は，(1)応答生成への利用頻度，(2)関係に含まれる組（ペア）の数，(3)関係に関連する関係の数，(4)合成演算で関連がつく関係の重要度，などに基づいて計算されている．(1)以外の計算方法は，ＲＭＳによる関係の生成方法に依存している．典型的な関係Ｒは，（例えば付録の(3)のように）Ｒと等しい(EQL)と推測される関係と，Ｒと逆転の関係にある(REV)と推測される関係，それに他の関係の合成で作られる(COM)関係の和集合である．(2)はそうしてできる関係に含まれる組（ペア）の数，(3)は構成する２種(EQL, REV)の関係の数であり，(4)はCOMに関する評価である．

(g) 応答の生成

現バージョンは，無限ループを回避しつつ，PROLOGの縦型探索と原理的に同じやり方で応答を生成している．これは，ＬＳ／１で採用している最良優先探索［２，３］に改善すべきである．

５． むすび

データベースを構築する部分は，帰納的学習システムにとって極めて重要な部分である．データベースは意味世界の表現のエッセンスだからである．その部分の特質を生かした情報処理をすれば，帰納的学習システム全体の知識獲得を強力に牽引できる筈である．そのことはＬＳ／１での実験（例えば付録の学習過程や獲得知識）などで確認できた．

ＲＭＳの定式化と作成の試みにより，解くべき多くの問題が明らかになった．例えば，重要度をよりうまく管理する方式の開発，重要度を利用した学習のアルゴリズムが必要である．また，誤った応答の原因として最も可能性の大きい知識を効率的に見つける問題，負の例を的確に処理する枠組みへの拡張なども極めて重要である．

文献

1）赤間清：帰納的学習を行なうシステムの初歩的なモデル，HBSR M(S)，NO.7，P29（1984）
2）赤間清：帰納的学習システムの最良応答探索，
情報処理学会，知識工学と人工知能研究会資料，41-12,P89-96（1985）
3）赤間清：知識の構造化を重視した学習のモデル，
情報処理学会，知識工学と人工知能研究会資料，45-5，(1986)
4）ウルマン著，国井利泰ほか訳：データベース・システムの原理，p584（1985）

付録：図２の質問応答列によるＬＳ／１とＲＭＳの学習結果

⑴学習過程
各Ｓ式は，質問，（システムの）応答，正解からなる．

```
((what is dog) ??? (it is animal))
((say example of animal) ??? (cat))
((salmon is fish) ??? (yes))
((dog can run) ??? (yes))
((hawk is bird) (yes) (yes))
((what can bird) ??? (can fly))
((dove can fly) (yes) (yes))
((what is dove) (it is animal) (it is bird))
((say example of bird) (cat) (dove))
((what is hawk) (it is animal) (it is bird))
((what can fish) (can fly) (can swim))
((what is pig) (it is animal) (it is animal))
((bird has wing) ??? (yes))
((say example of animal) (cat) (cat))
((cat can run) (yes) (yes))
((animal has leg) ??? (yes))
((which is animal : dog or dove) (yes) (dog))
((which is bird : hawk or cat) (hawk) (hawk))
((which is animal : eagle or pig) (yes) (pig))
((which is fish : dove or salmon) (salmon) (salmon))
```

⑵上の最後の質問に対して，(salmon)という正解を生成した推論過程．

```
--------retrace-------      ;v-rootの問題がv00015の問題に展開される
(--value= 830000)
(---rule=
 ((p v-root (($00012 is $00013) ($00014))) (p v00015 (($00012) ($00013) ($00014)))))
(-weight= 83)
(---data= (v v-root ((which is fish : dove or salmon) ($)) i00016))
--------retrace-------      ;eagleとdoveがマッチされ類推がなされる
(--value= 288508)
(---rule= ((p v00015 ((which) ($-1 : eagle or $-2) ($-2))) (p v01183 (($-1) ($-2)))))
(-weight= 79)
(---data= (v v00015 ((which) (fish : dove or salmon) ($00014)) i01184))
--------retrace-------      ;ＲＭＳへの問い合せにより答えが得られる
(--value= 392370)
(---rule= ((p v01183 (((fish) (salmon))))))
(-weight= 68)
(---data= (r v01183 ((fish) (salmon)) RMS))
```

```
--- answer ---
(salmon)
```

(3)ＲＭＳが発見した重要な関係

```
rg   : rms_rg_29 : 53   (cat . run)  (dove . fly) (dog . run)   ;53は重要度
 eql : (v00119 ((&-1) (&-2)))
 eql : (v-root ((&-1 can &-2) (yes)))
 com : (rev . rms_rg_21) (eql . rms_rg_44)                      ;発見した合成関係

-----
rg   : rms_rg_21 : 52   (animal . pig) (fish . salmon) (bird . hawk)
       (animal . dog) (bird . dove) (animal . cat)
 eql : (v-root ((which is &-1 : hawk or cat) (&-2)))
        …省略…
 eql : (v-root ((say example of &-1) (&-2)))
 rev : (v-root ((&-1 is &-2) (yes)))
 rev : (v-root ((what is &-1) (it is &-2)))
 com : (eql . rms_rg_44) (rev . rms_rg_29)

-----
rg   : rms_rg_44 : 40  (fish . swim) (bird . fly)
 eql : (v01577 ((&-1) (&-2)))
        …省略…
 eql : (v00147 ((&-1) (&-2)))
 eql : (v-root ((what can &-1) (can &-2)))
 com : (eql . rms_rg_21) (eql . rms_rg_29)

-----
rg   : rms_rg_109 : 10   (dog . dove) (eagle . pig)        ;こういう「誤った」
 eql : (v01427 ((&-1) (&-2)))                              ;関係も抽出されている.
        …省略…                                             ;重要度は低く押さえ
 rev : (v-root ((which is animal : dog or &-1) (&-2)))     ;られている.

-----
rg   : rms_rg_63 : 4   (animal . leg) (bird . wing)
 eql : (v00892 ((&-1) (&-2)))
 eql : (v-root ((&-1 has &-2) (yes)))
```

(4)ＲＭＳが発見した重要な集合（排反分割を形成するもの）

```
set  : rms_set_3  { eagle, pig, salmon, hawk, cat, dove, dog }
 eql : (rms_rg_109 .1) (rms_rg_109 .2) (rms_rg_21 .2) (rms_rg_11 .1) (rms_rg_29 .1)
 link: (rms_set_3 eql rms_rg_109) (rms_set_2 eql rms_rg_11) (rms_set_4 eql rms_rg_29)
       (rms_set_3 rev rms_rg_109) (rms_set_2 rev rms_rg_21)

-----
set  : rms_set_2  { fish, bird, animal }
 eql : (rms_rg_63 .1) (rms_rg_44 .1) (rms_rg_11 .2) (rms_rg_21 .1)
 link: (rms_set_6 eql rms_rg_63) (rms_set_4 eql rms_rg_44) (rms_set_3 eql rms_rg_21)
       (rms_set_3 rev rms_rg_11)

-----
set  : rms_set_4  { swim, fly, run }
 eql : (rms_rg_44 .2) (rms_rg_29 .2)
 link: (rms_set_2 rev rms_rg_44) (rms_set_3 rev rms_rg_29)

-----
set  : rms_set_6  { leg, wing }
 eql : (rms_rg_63 .2)
 link: (rms_set_2 rev rms_rg_63)
```