ICOT Technical Memorandum: TM-0878

TM-0878

オペレーション推定のための KLI言語による 並列帰納推論プログラム

April, 1990

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

オペレーション推定のための KL1 言語による並列帰納推論プログラム

滝　寛和、寺崎　智

概要: 本報告は、ある制限された形式の知識を帰納的に求める帰納推論プログラムに関するものである。本帰納推論プログラムは、知識獲得支援システム EPSILON/One のオペレーション推定のために開発された。アルゴリズムの特徴は、オペレーションタイプの記述制限を利用して、効率的な仮説生成を行う点である。なお、本プログラムは、並列論理型言語 KL1 で記述されており、Pseudo Multi-PSI および Multi-PSI 上で実行可能である。

1　はじめに

知識獲得支援システム EPSILON/One の機能拡張として、オペレーションのエバリュエータを帰納推論により求めるプログラムを開発した。本報告書では、アルゴリズムとプログラムリストを紹介する。開発の背景などは、ICOT TM-866 を参照してください。

2　オペレーション推定アルゴリズム

2.1　選択オペレーションの推定

2.1.1　選択アルゴリズム

ある要素が与えられたとき、その要素の属性と属性値が、選択基準を満たすとき選択される。選択基準は、ある属性の値の取り得る集合（または、複数の属性についての可能な属性集合の連言 (conjunction)）で表される。選択基準を満たすとは、評価基準が規定している属性に注目して、評価される要素のその属性値が、選択基準の可能な属性値の集合に含まれることをいう。

2.1.2　選択基準

単一属性から成る選択基準は、次のように限定する。

（属性名、属性値の集合）

複数属性から成る選択基準は、単一属性の選択基準の連言形式とする。

（属性名１、属性値の集合）∩（属性名２、属性値の集合）∩・・・・

2.1.3　例の扱い

例としては、複数の要素をオペレーションの入力および、出力要素グループとして与える。選択オペレーションで、選択された要素は、選択基準の正例と言える、また、選択されなかった要素は、負例と言える。要素は、オペレーションとは、独立の知識としてグローバルに定義できる。また、要素は、属性と１つの属性に対して１つの属性値の対の連言で表わされるものとする。

2.1.4　選択基準の推定アルゴリズム

Step1: まず、正例、負例を問わず、例すべてに含まれる属性 a を求める。

例：a1, a2, ... , am （ただし、aj (j=1～m) は、属性を表す）

Step2: 正例／負例毎に、各要素に対して、上記の属性集合の各属性に対して値を求める、要素がある属性を持たない場合には、その属性値を undefined とする。

Step3: 求める選択基準が単一属性であると仮定して、属性単位で正例／負例の可能な属性値の集合を求める。これは、属性間の従属性に関する一般化である。

例：要素を ek(k=1～n)、 属性を aj(j=1～m)、属性値を vjk(k=1～n, j=1～m) とすると正例／負例毎に要素と属性に関するマトリックスができる。

```
正例：    a1     a2     a3     ...     am
     e1 : v11    v21    v31    ...     vm1
     e2 : v12    v22    v32    ...     vm2
                  :
     ei : v1i    v2i    v3i    ...     vmi

負例：
   ei+1: v1i+1 v2i+1 v3i+1 ...    vmi+1
   ei+2: v1i+2 v2i+2 v3i+2 ...    vmi+2
                  :
    en : v1n    v2n    v3n    ...    vmn
```

正例と負例について属性ごとに属性値の集合を作る。 属性 aa に関する正例／負例の属性値集合をそれぞれ P(aa)、N(aa) とする。

```
P(a1) = {v11, v12, ..., v1i}
P(a2) = {v21, v22, ..., v2i}
                  :
P(am) = {vm1, vm2, ..., vmi}

N(a1) = {v1i+1, v1i+2, ..., v1n}
N(a2) = {v2i+1, v2i+2, ..., v2n}
                  :
N(am) = {vmi+1, vmi+2, ..., vmn}
```

Step4: 正例と負例で積集合を持つ属性は、単一属性では選択に利用できないので除く。

if $P(aj) \wedge N(aj) = \phi$ (空集合) then 属性 aj を選択する

else 属性 aj は選択しない, (j = 1 ～ m)

選ばれた 属性の集合が、aa, ab, ac であれば、 P(aa)、P(ab) または、 P(ac) が選択基準となる。

注) $P(aj) \wedge N(aj) = \phi$ から $\neg N(aa) \wedge \neg N(ab) \wedge \neg N(ac)$ も選択基準である。

ただし、$\neg N(aa)$ は、属性 aa が、属性値として、 N(aa) の要素を取らないことである。

Step5: 求められた選択基準を提示する。複数ある場合には、順に提示する。

上記の例では、 P(aa), P(ab), P(ac), $\neg N(aa) \wedge \neg N(ab) \wedge \neg N(ac)$ を順に提示する。

$P(aj) \wedge N(aj) = \phi$ となる属性 aj がない場合は、属性間を独立にすることは、失敗となり、複数の属性の関係を考慮した判定基準が必要となる。

2.2 順序付けオペレーションの推定

2.2.1 順序付けアルゴリズム

与えられたすべての要素について、ある属性に注目して、その属性値の全順序基準に従って要素を並べ替える。この順序は、オペレーション（選択や出力オペレーションなど）での要素の評価順序を意味する。

2.2.2 順序付け基準

単一属性から成る順序付け基準は、次のように限定する。 （属性名、属性値リスト）または、（属性名、昇順）または、（属性名、降順）属性値リストの記号順序が、上位から下位への順序を表すものとする。また、昇順と降順は、属性値が数値であるものを表す。

2.2.3 例の扱い

例としては、複数の要素をオペレーションの入力および、出力要素グループとして与える。 順序付けオペレーションでは、出力要素グループの要素の順序が順序基準の正例と言える、また、入力要素グループでの2要素の順序関係が、出力要素グループでの異なる順序となる2要素間の順序は、負例と言える。

(例題)

入力要素グループと出力要素グループを次のように与えると

入力要素グループ ={e1, e2, e3, e4}

出力要素グループ ={e1, e3, e4, e2}

a よりも b を優先する関係を ≫ で表すと、正例は、e1 ≫ e3 ≫ e4 ≫ e2 である。 また、負例は、e2 ≫ e3 と e2 ≫ e4 となる。

2.2.4 順序付け基準推定アルゴリズム

Step1: 入力要素グループと出力要素グループのそれぞれについて、要素間の優先順 序に関する2項関係を求める。 出力要素グループについての要素の2項関係は、すべて、正例である。 また、入力要素グループについての要素の2項関係で、出力での関係と 矛盾するものは、負例である。順序の関係に矛盾があるとは、 a ≫b かつ b ≫a の関係が同時に成立することをいう。

Step2: 属性値に undefined 含まない属性を1つを選び、出力要素グループの要素の 対について、その属性値の順序を決める。これをその属性に関する順序基準 と呼ぶ。推定対象となる属性がない場合には、推定を終了する。

Step3: 順序の関係に、矛盾がないかを調べる。矛盾があれば、属性を変更して、 Step2 を実行する。

Step4: 属性値が、数値のときは、昇順か、降順かを調べる。その他の関係 (偶数を優 先するなど) であれば、数値であっても、文字と同じように扱う。

Step5: このような条件を満足する属性が、複数ある場合には、属性とその順序 基準を順に提示する。

なお、順序付けオペレーションでの負例は、正例と共通部分がない (反対の関係にある) ため、正例から求められた評価基準を制限できないので、順序基準の推定には、使用されない。

2.3 属性値入れ替えオペレーションの推定

2.3.1 属性値入れ替えアルゴリズム

ある要素が与えられたとき、ある1つの属性に注目して、入れ替え基準に従って、元の属性値に対応する別の値に入れ替える。入れ替え基準は、変換テーブル形式で表され、元の値と変換後の値の対からなる集合である。

2.3.2 属性値入れ替え基準

単一属性から成る入れ替え基準 (規則) は、次のように限定する。

(属性名、(元の属性値、変換後の属性値) の集合)

2.3.3 例の扱い

例としては、複数の要素をオペレーションの入力および、出力要素グループとして与える。属性値の入れ替えオペレーションで、入力と出力で値の異なる要素は正例と言える、また、値の変わらない要素は負例と見ることができる。

2.3.4 属性値入れ替え基準推定アルゴリズム

Step1: 値が変更されている属性を1つ探す。これを注目属性とする。

Step2: 正例について、注目属性の元の属性値と変更後の属性値の対を作る。

Step3: 元の属性値が同じで、変更後の属性値が異なるものを探す。 この変更後の属性値の候補が複数あるものを不明確な基準とする

Step4: 負例について、注目属性の属性値を取り出す。

Step5: Step2 で得られた属性値の対の中で、元の属性値が Step4 で得られた属性値と等しい属性値の対を不正確な基準とする。

Step6: 不明確でなく、不正確でない属性値の対を属性値入れ替え基準として提示する。

Step7: さらに、不明確な基準と不正確な基準を参考情報として、提示する。

3 プログラムの実行方法

3.1 選択オペレーションの推定

メインのモジュール名： ind1 （ファイル ind1.kl1 ）
例を記述するモジュール名： index （ファイル index1.kl1 ）
例の形式： [[要素], [要素], [要素], ...]
[要素] = [要素名, [属性], [属性], ...]
[属性] = [属性名, 属性値]
input_group と output_group が必要。
実行ゴール： ind1:goal(N).
ただし、Nは、1以上の整数であり、生成候補の属性の数を指定する。

実行結果は、マウスで領域を指定したウインドウに表示される。
実行例は、図1参照。

3.2 順序付けオペレーションの推定

メインのモジュール名： ind2 （ファイル ind22.kl1 ）
例を記述するモジュール名： ind2（メインモジュール内に書く）
例の形式： [[要素], [要素], [要素], ...]
[要素] = [要素名, [属性], [属性], ...]
[属性] = [属性名, 属性値]
output_group が必要。
実行ゴール： ind2:goal(K).
ただし、Kは、ウインドウに表示された結果のストリームが渡される。

実行結果は、マウスで領域を指定したウインドウに表示される。
実行例は、図2参照。

3.3 属性値入れ替えオペレーションの推定

メインのモジュール名： ind3 （ファイル ind33.kl1 ）
例を記述するモジュール名： ind3（メインモジュール内に書く）
例の形式： [[要素], [要素], [要素], ...]
[要素] = [要素名, [属性], [属性], ...]
[属性] = [属性名, 属性値]
input_group と output_group が必要。
実行ゴール： ind3:goal(K).
ただし、Kは、ウインドウに表示された結果のストリームが渡される。

実行結果は、マウスで領域を指定したウインドウに表示される。
実行例は、図3参照。

3.4 負荷分散の検討

プログラムの構造は、図4 のようなプロセスで表される。そこで、2種類の負荷分散を行った。

（方式1） p-i1, p-i2, ..., p-in を1つのプロセッサに割り付ける。（ただし、i= 1 ～ m ）

（方式2） p-1j, p-2j, ..., p-mj を1つのプロセッサに割り付ける。（ただし、j= 1 ～ n ）

現在のプログラムは、次のような記述で書かれている。

```
p-i(Instream, Outstream) :- true |
        p-i(Instream, Outstream2),
        p-i+1(Instream, Outstream3),
        Outstream=[Outstream2|Outstream3].
```

この記述では、もし、p-i+1 の処理が長いと Outstream の決定が遅れる。そのために、p-i のプロセスは終了しないで、p-i のプロセスは、どんどん増加する。つまり、p-i と p-i+1 のプロセス間に張られる Stream が増えることになる。このような記述では、方式1では、プロセス間通信が頻繁となり、処理速度は低下する。また、例から作られる仮説が多いとプロセス間の変数参照テーブルがあふれる。このような場合には、方式2の方が良い。

実際の実行でも、方式1は、プロセス間の変数参照テーブルがあふれる場合があり実行が中断された。

なお、本プログラムの使用目的は、対話型なので、実行速度の比較は行っていない。

4 プログラムリスト

```
ind1.kl     ---- 選択オペレーションの推定
index1.kl1 ---- 選択オペレーションの推定の例1
index2.kl1 ---- 選択オペレーションの推定の例2
index3.kl1 ---- 選択オペレーションの推定の例3
ind22.kl1   ---- 順序付けオペレーションの推定
ind33.kl1   ---- 属性値入れ替えオペレーションの推定
```

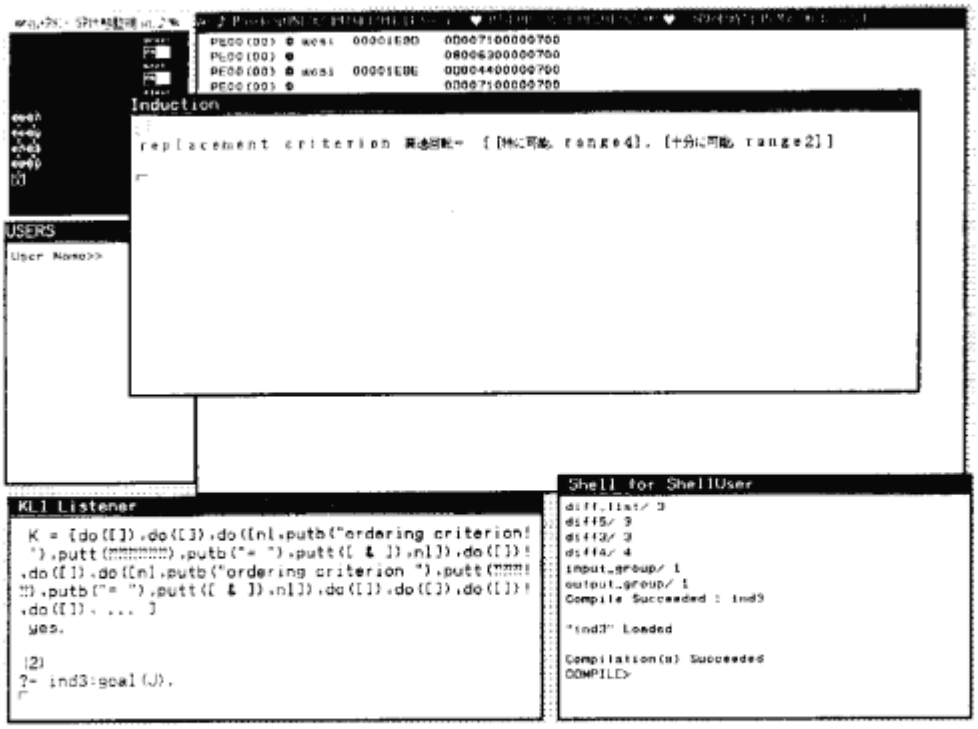

図3　　属性値入れ替えオペレーションの推定の実行例

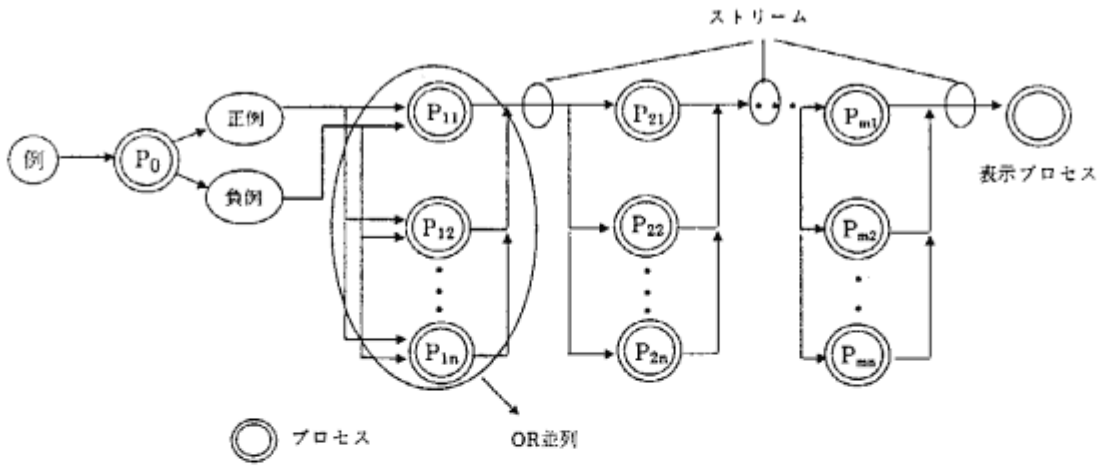

図4　　プロセスの関係

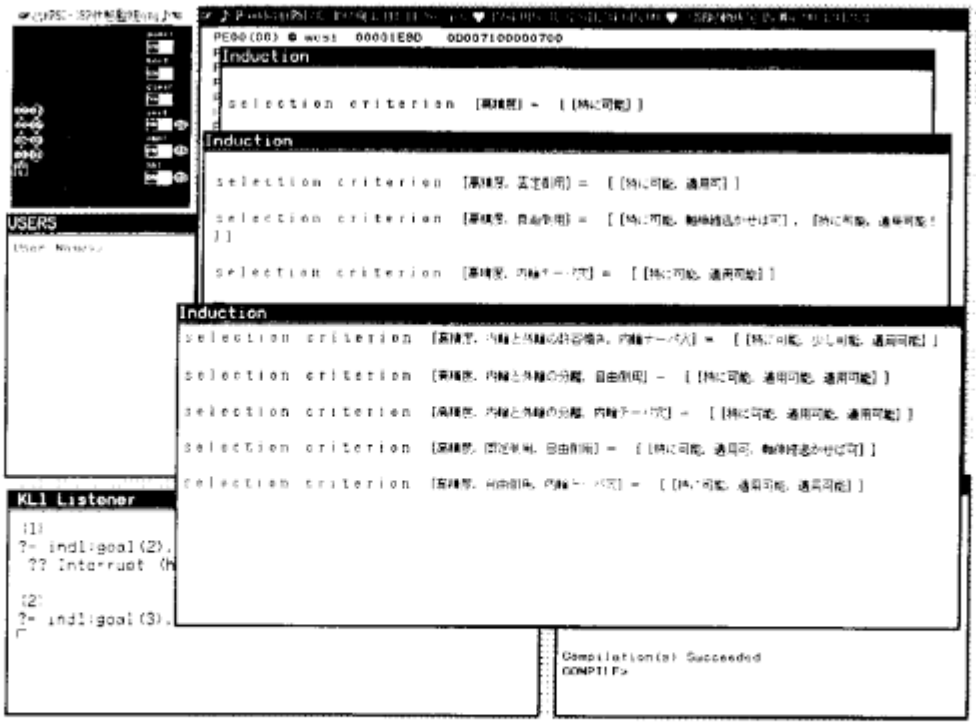

図1　　選択オペレーションの推定の実行例

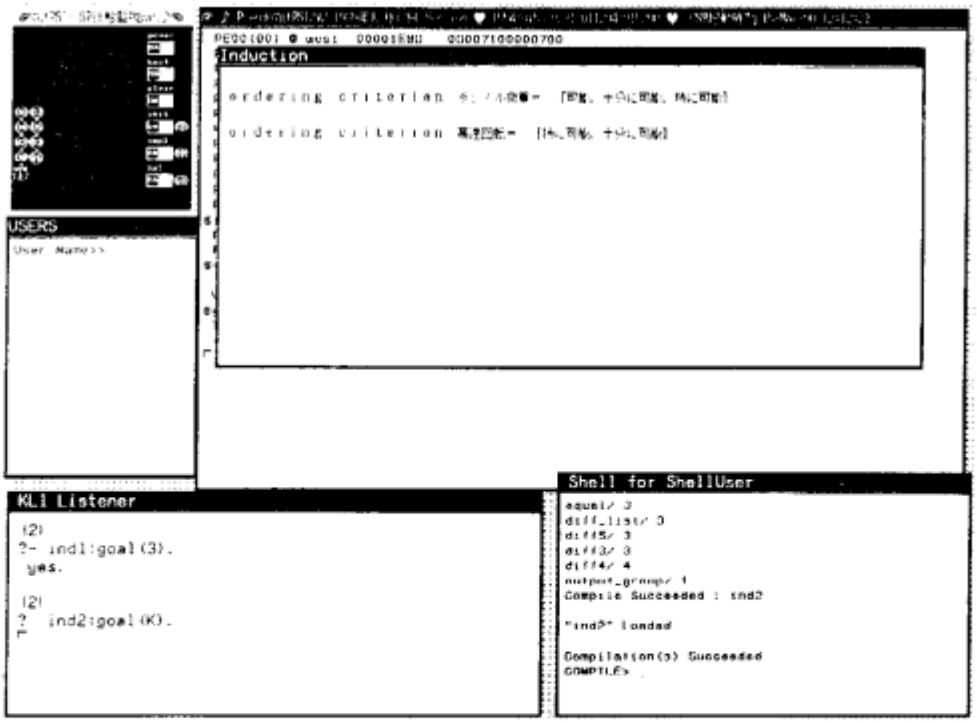

図2　　順位付けオペレーションの推定の実行例

```
% 1990.2.14 version
%
%    ind1.kl1 : Simple Induction Argolithm(1)
%                   選択オペレーションの推定
%
:- with_macro pimos.
:-module ind1.
:-public goal/1, wd/2, common/3, subgoal/3,
     subst/3, equal/3, pairlist/2, multlist/3,
     muul/3, mkelc0/2, app2/3, flat/2, lister/2, car/2,
     pos_att_list/1, make_neg/1, neg_att_list/2, make_pos_value/2,
     make_neg_value/3, analysis/2, efface/2,
     make_neg_mult/4, make_pos_mult/3, mpm/3, mpm2/3, mpm3/3,
     refm/3, refm2/2, display_empty/3.

wd(L, S) :- true | w_create(L, S).

w_create(Label, Str) :- true |
     shoen:raise(pimos_tag#task, general_request, GRD),
     GRD=[window(WRD)],
     w_create1(WRD, Label, Str).

w_create1(normal(Window, _, _), Label, Str) :- true |
     Window=[create(WD)],
     w_create2(WD, Label, Str).
otherwise.
w_create1(_, _, _) :-
     display_console('*** window create fail (no. 1) ***') |
     true.

w_create2(normal(W2, _, _), Label, Str) :- true |
   W2=[reshape(mouse, mouse, normal),
     set_font(string#"font:kanji_16", normal),
     set_title(Label, normal),
     activate(normal), W3],
   buffer:interaction_filter(Str, W3).
otherwise.
w_create2(_, _, _) :-
     display_console('*** window create fail (no. 2) ***') |
     true.

%
%         Main Program induction 1
%

subgoal(Att_list2, Result, Num) :- true | pos_att_list(Att1),
     make_neg(Att2),
     neg_att_list(Att2, Att3), Att4=[Att1|Att3],
     flat(Att4, Flat_att), efface(Flat_att, Att_list),
     make_att_list(Att_list, Att2, Result, Num, Att_list2).

% make_att_list(Att, Att2, Result, 1, Att) :- true |
%     single_induction(Att, Att2, R), Result=R.
% otherwise.
make_att_list(Att, Att2, R, Num, Matt) :- true |
     multlist(Att, Matt, Num),
     make_pos_mult(Att, Num, V1),
     make_neg_mult(Att, Att2, Num, V2), Mat=[V1, V2],
     analysis(Mat, R2), R=R2.

% single_induction(Att, Att2, R) :- true |
%     make_pos_value(Att, V1),
```

```
%     make_neg_value(Att, Att2, V2),
%     Mat=[V1,V2], analysis(Mat, Result), R=Result.

%
%    make_pos_mult
%    make_neg_mult
%
make_pos_mult(Att, Num, R) :- true |
     Index:output_group(E),
     mpm(Att, E, R2),
     refm(R2, Num, R3),
     refm2(R3, R4), R=R4.

make_neg_mult(Att, E, Num, R) :- true |
     mpm(Att, E, R2),
     refm(R2, Num, R3),
     refm2(R3, R4), R=R4.

mpm(Att, [], R) :- true | R=[].
mpm(Att, [E|Rest], R) :- true |
     mpm2(Att, E, R2), mpm(Att, Rest, R3),
     R=[R2|R3].
mpm2(Att, [E|Vj], R) :- true | mpm3(Att, V, R).
mpm3(_, [], V, R) :- true | R=[].
mpm3(_, [At1|AR], V, R) :- true |
     assocm(At1, V, R1), mpm3(AR, V, R2),
     R=[R1|R2].

assocm(X, [], Z) :- true | Z=[].
assocm(X, [[Y,V]|Rest], Z) :- diff(X,Y) | assocm(X, Rest, Z).
otherwise.
assocm(X, [[Y,V]|Rest], Z) :- true | Z=V.

refm([], Num, R) :- true | R=[].
refm([M|V], Num, R) :- true |
     multlist(M, M2, Num), refm(V, Num, R2),
     R=[M2|R2].

%
refm2([[]|Rest], R) :- true | R=[].
otherwise.
refm2(Org, R) :- true | refm4(Org, Org, R1), cdrlist(Org, O2),
     refm2(O2, R2), R=[R1|R2].

refm4([[]|Rest], L, R) :- true |
     R=[], % [[],[],[],....] nara stop
refm4(Org, [], R) :- true | R=[].
refm4(Org, [A|M], R) :- true | car(A, CA),
     refm4(Org, M, R2), R=[CA|R2].

%


make_neg_value(Att, Att2, V2) :- true |
     pick_value(Att, Att2, V3), V2=V3.
make_pos_value(Att, V) :- true | index:output_group(P),
     pick_value(Att, P, V2), V=V2.

pick_value([], P, V) :- true | V=[].
pick_value([A1|Rest], P, V) :- true |
     assocv(A1, P, V2), pick_value(Rest, P, V3),
     V=[V2|V3].
```

```
assocv(A1, [], V) :- true | V=[].
assocv(A1, [E|R], V) :- true |
    assocv2(A1, E, V2), assocv(A1, R, V3), V=[V2|V3].
assocv2(A1, [Ename|Att], V) :- true |
    assv3(A1, Att, V2), V=V2.

assv3(A1, [], V) :- true | V= undefined.
assv3(A1, [[H|Value]|R], V) :- diff(A1, H) | assv3(A1, R, V2), V=V2.
otherwise.
assv3(A1, [[H,Value|K]|R], V) :- true | V= Value.

make_neg(Att) :- true | index:input_group(In),
    index:output_group(Out),
    element_select(In, Out, Att2), Att=Att2.

element_select([], Y, Z) :- true | Z=[].
element_select([Elem|Rest], Y, Z) :- true | elems(Elem, Y, Z2),
    element_select(Rest, Y, Z3), without_nil(Z2, Z3, Z).

elems([Ename|Econt], Y, Z) :- true |
    elems2(Ename, Econt, Y, Z2), Z=Z2.
elems2(Ename, Econt, [], Z) :- true |
    without_nil(Ename, Econt, Z2), Z=Z2.
elems2(Ename, Econt, [[H|T]|R], Z) :- diff(Ename, H) |
    elems2(Ename, Econt, R, Z2),
    Z=Z2.
otherwise.
elems2(Ename, Econt, Y, Z) :- true | Z =[].

neg_att_list(Att1, Att2) :- true | pick_up(Att1, Att3), Att2=Att3.

pos_att_list(Att1) :- true | index:output_group(P_element),
    pick_up(P_element, Att_list), Att1=Att_list.

pick_up([], Y) :- true | Y=[].
pick_up([X|Y], Z) :- true | pick_up2(X, X2), pick_up(Y, Z2), Z=[X2|Z2]
%
pick_up2([X|X2], Y) :- true |
    pick_up3(X2, Y2), Y=Y2. % X is element name
pick_up3([], Y) :- true | Y=[].
pick_up3([X|Y], Z) :- true |
    pick_up4(X, Z1), pick_up3(Y, Z2), Z=[Z1|Z2].

pick_up4([A|V], Z) :- true | Z=A.

flat([], Y) :- true | Y=[].
flat([Org|Rest], Dest) :- list(Org) | flat(Org, D2),
            flat(Rest, D3),
            append(D2, D3, D4), Dest= D4.
otherwise.
flat([Org|Rest], Dest) :- true | Dest=[Org|D2], flat(Rest, D2).

append([], Y, Z ) :- true | Z = Y.
append([A|X], Y, Z) :- true | Z=[A|Z2], append(X, Y, Z2).

efface([], Y) :- true | Y=[].
efface([Check|Rest], Dest) :- true | assoce(Check, Rest, C2),
      efface(Rest, C3), without_nil(C2, C3, Dest).

without_nil([], Y, Z) :- true | Z = Y.
otherwise.
without_nil(X, Y, Z) :- true | Z = [X|Y].
```

```
assoce(Check, [], Result) :- true | Result = Check.
assoce(Check, [Head|Rest], Result) :- list(Check) |
    equal(Check, Head, J),
    assoce2(Check, Head, Rest, J, Result).
otherwise.
assoce(Check, [Head|Rest], Result) :- true |
    assoce3(Check, Head, Rest, Result).

assoce3(Check, Head, Rest, Result) :- diff(Check, Head) |
    assoce(Check, Rest, R2), Result = R2.
otherwise.
assoce3(Check, Head, Rest, Result) :- true | Result = [].

assoce2(Check, Head, Rest, [], R) :- true | assoce(Check, Rest, R).
otherwise.
assoce2(Check, Head, Rest, J, R) :- true | R=[].

%
%
%
% Common(A, B, L) : L is common elements list of A and B
%
common([], B, L) :- true | L=[].
common([A|R], B, L) :- true | common2(A, B, L2), common(R, B, L3),
    without_nil(L2, L3, L). % L2 atom
common2(A, [], L) :- true | L=[].
common2(A, [B|R], L) :- list(A) | equal(A, B, C), common3(A, R, C, L).
otherwise.
common2(A, Y, L) :- true | common5(A, Y, L).

common3(A, R, [], L) :- true | common2(A, R, L).
otherwise.
common3(A, R, C, L) :- true | L=A.

common5(A, [B|R], L) :- diff(A, B) | common2(A, R, L2), L=L2.
otherwise.
common5(A, [B|R], L) :- true | L=A.
%
%
%  Substring List
%  Subst(A, B, L) : A B -L
subst([], B, L) :- true | L=[].
subst([A|R], B, L) :- true | subst2(A, B, L2), subst(R, B, L3),
    without_nil(L2, L3, L).
subst2(A, [], L) :- true | L=A.
subst2(A, Y, L) :- list(A) | subst3(A, Y, L2), L=L2.
otherwise.
subst2(A, Y, L) :- true | subst5(A, R, L).

subst5(A, [B|R], L) :- diff(A, B) | subst2(A, R, L).
otherwise.
subst5(A, [B|R], L) :- true | L=[].
%
subst3(A, [], L) :- true | L=A.
subst3(A, [B|R], L) :- list(B) | equal(A, B, C), subst4(A, R, C, L).
subst4(A, R, [], L) :- true | subst2(A, R, L).
otherwise.
subst4(A, R, C, L) :- true | L=[].

analysis([Pos, Neg], R) :- true | ana(Pos, Neg, R2), R=R2.
ana([], [], R) :- true | R=[].
```

```
ana([Pos|Posr], [Neg|Negr], R) :- true | ana2(Pos, Neg, R2),
     ana(Posr, Negr, R3), R=[R2|R3].
ana2(Pos, Neg, R) :- true |
     common(Pos, Neg, CL), subst(Pos, CL, P2),
     subst(Neg, CL, N2), R=[P2, CL, N2].

%
% Num attribute combination
%
goal(Num) :- true
     w_create("Induction", Str),
     subgoal(Att, Mat, Num),
     display_empty(Att, Mat, Str).

display_empty([], [], St) :- true | St=[nl, flush(_), getc(_)].

display_empty([Att|AR], [Mat|MR], St) :- true | dispe2(Att, Mat, St1),
     display_empty(AR, MR, St2), St = [do(St1)|St2].

dispe2(Att, [P, [], N], St) :- true |
     efface(P, P2),
     negrige(P2, P3), dispe3(Att, P3, St).
otherwise.
dispe2(Att, M, St) :- true | St=[].

dispe3(Att, [], St) :- true | St=[].
dispe3(Att, P, St) :- true |
     St=[nl, puib(string#"selection criterion "),
     putt(Att), putb(string#" = "), putt(P), nl].
%
negrige([], R) :- true | R = [].
negrige([X|Rest], R) :- true | neg2(X, X, R2), negrige(Rest, R3),
     without_nil(R2, R3, R).

neg2(X, [], R) :- true | R = X.
neg2(X, [undefined|Rest], R) :- true | R= [].
neg2(X, [[]|Rest], R) :- true | R= [].
otherwise.
neg2(X, [A|Rest], R) :- true | neg2(X, Rest, R).

%
%

% List equality check if equal then true, else [].

equal(X, Y, R) :-true | diff_list(X, Y, R).

diff_list(X, Y, R) :- list(X) | diff5(X, Y, R2), R=R2.
diff_list(X, Y, R) :- diff(X, Y) | R=[].
otherwise.
diff_list(X, Y, R) :- true | R=true.

diff5(X, Y, R) :- list(Y) | diff3(X, Y, R2), R=R2.
otherwise.
diff5(X, Y, R) :- true | R=[].

diff3([], [], R) :- true | R=true.
diff3([X|R1], [Y|R2], R) :- true | diff_list(X, Y, R3),
     diff4(R3, R1, R2, R4), R=R4.
otherwise.
diff3(X, Y, R) :- true | R= [].
```

```
diff4(R3, X, Y, R) :- diff(R3, [ ]) | diff3(X, Y, R2), R=R2.
otherwise.
diff4(R3, X, Y, R) :- true | R=[ ].

%
% make pair list  multi list
pairlist(Org, Pair) :- true | mul1(Org, Pair).

mul1([ ], R) :- true  R=[ ].
mul1([Head|Tail], R) :- true | mul12(Head, Tail, R2),
     mul1(Tail, R3), append(R2, R3, R4), R=R4.

mul12(H, [ ], R) :- true | R=[ ].
mul12(H, [Can|Tail], R) :- true | R=[[H, Can]|R2],
     mul12(H, Tail, R2).

% multi-list version
multlist(Org, Mlist, Num) :- muul(Org, Mlist2, Num),
     mkele0(Mlist2, R2), Mlist=R2.

muul([ ], R, Num) :- true | R = [ ].
muul( X, R, 0) :- true | R = [ ].
otherwise.
muul( X, R, Num) :- true | Num2 := Num - 1,
     R=[X|R2], muul2(X, R2, Num2).

muul2( [X|Y], R, Num) :- true | muul(Y, R2, Num), R=R2.

mkele(Head, [ ], Next, Result) :- true| Result = [ ].
mkele(Head, [Fele|Self], [ ], Result) :- true |
     app2( Head, [Fele], H2), mkele(Head, Self, [ ], R2),
     Result=[H2|R2].
otherwise.
mkele(Head, [Fele|Self], Next, Result) :- true |
     app2( Head, [Fele], H2), mkele2(H2, Next, R2),
     cdrlist(Next, Next2), mkele(Head, Self, Next2, R3),
     conn(R2, R3, Result).
%     Result=[R2|R3].


mkele2(Head, [ ], R ) :- true | R = Head.
mkele2(Head, [Next1|Next2], R) :- true |
     mkele(Head, Next1, Next2, R2), R=R2.

mkele0( Mlist, R ) :- true| mkele2([ ], Mlist, R2), R=R2.

app2(X, Y, Z) :- true |
     lister(X, X2), lister(Y, Y2), append(X2, Y2, Z2), Z=Z2.

% [ ] is not list.
lister([ ], R) :- true | R= [ ].
lister(X, R) :- list(X) | R=X.
otherwise.
lister(X, R) :- true | R=[X].

cdrlist([ ], R) :- true | R= [ ].
cdrlist([X|Ta], R) :- true | R=[ R1 | R2],
 cdr(X, R1), cdrlist(Ta, R2).

cdr([ ], R) :- true | R=[ ]. % Why X is [ ] ?
cdr([X|R1], R) :- true | R=R1.
```

```
car([ ], R)  :- true | R = [ ].
car([X|R1], R)  :- true | R=X.

conn([ ], [ ], Z)  :- true | Z=[ ].
conn([ ], Y, Z)  :- diff(Y, []) | Z= Y.
conn(X, [ ], Z)  :- true | Z= X.
otherwise.
conn(X, Y, Z)  :- true | append(X, Y, Z2), Z=Z2.

reform([], Y)  :- true | Y=[ ].
reform([X|Y], Z)  :- true | Z=[Y|Z2], reform(X, Z2).
otherwise.
reform(X, Z)  :- true | Z=[X].
```

```
%  1990.2.21  version
%
%     ind22.kl1:  Simple  Induction  Argolithm(2)
%                    順位付けオペレーションの推定
%
:- with_macro pimos.
:-module  ind2.
:-public  goal/1,
          wd/2,  mk_v_list/3,  ck_order/2,  output_group/1,  pick_up/2,
          efface/2,  tonari/2, flat/2.

wd(L, S)  :-  true  |  w_create(L, S).

w_create(Label,  Str)  :- true  |
     shoen:raise(pimos tag=task,  general_request,  GRD),
     GRD=[window(WRD)],
     w_create1(WRD,  Label,  Str).

w_create1(normal(Window,  _,  _),  Label,  Str)  :- true  |
     Window=[create(WD)],
     w_create2(WD,  Label,  Str).
otherwise.
w_create1(_, _, _)  :-
     display_console('***  window  create  fail  (no.  1)  ***')  |
     true.

w_create2(normal(W2,  _,  _),  Label,  Str)  :- true  |
   W2=[reshape(mouse, mouse, normal),
     set_font(string#"font:kanji_16", normal),
     set_title(Label,  normal),
     activate(normal)|W3],
   buffer:interaction_filter(Str,  W3).
otherwise.
w_create2(_, _, _)  :-
     display_console('***  window  create  fail  (no.  2)  ***')|
     true.

%
%          Main  Program  induction  2
%
goal(Str)  :- true  |
     w_create("Induction",  Str),
     subgoal(Att,  Mat),
     display_list(Att,  Mat,  Str).
%

display_list([], [], St)  :- true  |  St=[nl,  flush(_), getc(_)].

display_list([Att|AR], [Mat|MR],  St)  :- true  |  dispe2(Att,  Mat,  St1),
     display_list(AR,  MR,  St2),   St  =  [do(St1)|St2].

dispe2(A,  [],  St)  :- true  |  St =[].
dispe2(A,  [M],  St)  :- true  |  St =[].
% M includes undefined and element is only one
% will be negrected 1990.2.20
%
otherwise.
dispe2(A, M,  St)  :- true  |  revmember(undefined,  M,  R),
     dispe3(A, M, St, R).

dispe3(A, M, St, [])  :- true  |  St =[].
```

```
otherwise.
dispe3(A, M, St, R)  :- true |
     St=[nl,  putb(string#"ordering criterion "),
     putt(A),  putb(string#" = "),  putt(M), nl].
%
%
subgoal(Att_list,  Result) :- true |
     output_group(P_element),
     pick up(P_element,  Att1),
     flat(Att1,  Att0),
     efface(Att0,  Att_list),
% made_up all attribute list= Att_list
%    make_value list_for_each_attribute
     mk_v_list(Att_list,  P_element,  Att2),
% Att2 { [value_list], [,], [,],...}
     tonari(Att2,  Att3),   % tonariau monoga onajidearuto sorewo nozoku
     ck_order(Att3,  Candidate),  Result=Candidate.
%
tonari([ ], R)  :- true | R=[ ].
tonari([A|Rest], R)  :- true |
     tonari2(A, R2),  tonari(Rest, R3),  R=[R2|R3].
tonari2([ ],  R)  :- true | R = [ ].
tonari2([A, A|Rest], R)  :- true |
     A2=[A|Rest], tonari2(A2, R2),  R=R2.
otherwise.
tonari2([A|Rest], R)  :- true | tonari2(Rest, R2),  R=[A|R2].
%

mk_v_list([ ], P, A2)  :- true | A2=[ ].
mk_v_list([A_attribute|R_attribute], P, A2) :- true |
     mk_vlist2(A_attribute,  P,  A3),
     mk_v_list(R_attribute, P, A4),  A2=[A3|A4].

mk_vlist2(A, [ ],  A2)  :- true | A2=[ ].
mk_vlist2(A, [[Name|Plist]|Rest], A2)  :- true |
     assoca(A,  Plist,  Value),
     mk_vlist2(A,  Rest,  A3),  A2=[Value|A3].
%
assoca(A, [ ],  C)  :- true | C= [ ].
assoca(A, [[A, B]|R], C)  :- true | C=B.
otherwise.
assoca(A, [B|R], C)  :- true | assoca(A, R, C).
%
ck_order([ ], R)  :- true | R=[ ].
ck_order([A|Rest], R)  :- true | ck_o2(A, R2),
     ck_order(Rest, R3),  R=[R2|R3].              % include [ ]

ck_o2([ ], R)  :- true | R = [ ].
ck_o2([A|Rest],  R)  :- true | revmember(A, Rest, Result),
          % Result = [ ] : Rest contains A
     ck_o3(Result, Rest, R).
otherwise.
ck_o2(_, R)  :- true | R = [ ].

ck_o3([ ],  R1,  R)  :- true | R=[ ].
otherwise.
ck_o3(A, B, R)  :- true | R=[A|R2],  ck_o2(B, R2).
%
revmember(A, [ ],  C)  :- true | C = A.
revmember(A, [A|B], C)  :- true | C= [ ].
otherwise.
revmember(A, [B|D], C)  :-true | revmember(A, D, C).
```

```
pick_value([], P, V) :- true | V=[].
pick_value([A1|Rest], P, V) :- true |
     assocv(A1, P, V2), pick_value(Rest, P, V3),
     V=[V2 V3].

assocv(A1, _, V) :- true | V=[].
assocv(A1, [E|R], V) :- true |
     assocv2(A1, E, V2), assocv(A1, R, V3), V=[V2|V3].
assocv2(A1, [Ename|Att], V) :- true |assv3(A1, Att, V2), V=V2.

assv3(A1, [], V) :- true | V= undefined.
assv3(A1, [[H|Value]|R], V) :- diff(A1, H) |
     assv3(A1, R, V2), V=V2.
otherwise.
assv3(A1, [[H,Value K] R], V) :- true | V= Value.

pick_up([], Y) :- true | Y=[].
pick_up([X Y], Z) :- true |
     pick_up2(X, X2), pick_up(Y, Z2), Z=[X2|Z2].
%
pick_up2([X X2], Y) :- true |
     pick_up3(X2, Y2), Y=Y2. % X is element name
pick_up3([], Y) :- true | Y=[].
pick_up3([X|Y], Z) :- true |
     pick_up4(X, Z1), pick_up3(Y, Z2), Z=[Z1|Z2].

pick_up4([A|V], Z) :- true | Z=A.

flat([], Y) :- true | Y=[].
flat([Org|Rest], Dest) :- list(Org) | flat(Org, D2),
               flat(Rest, D3),
               append(D2, D3, D4), Dest= D4.
otherwise.
flat([Org,Rest], Dest) :- true | Dest=[Org|D2], flat(Rest, D2).

append([], Y, Z) :- true | Z = Y.
append([A|X], Y, Z) :- true | Z=[A|Z2], append(X, Y, Z2).

efface([], Y) :- true | Y=[].
efface([Check|Rest], Dest) :- true | assoce(Check, Rest, C2),
       efface(Rest, C3), without_nil(C2, C3, Dest).

without_nil([], Y, Z) :- true | Z = Y.
otherwise.
without_nil(X, Y, Z) :- true | Z = [X|Y].

assoce(Check, [], Result) :- true | Result = Check.
assoce(Check, [Head|Rest], Result) :- list(Check) |
     equal(Check, Head, J),
     assoce2(Check, Head, Rest, J, Result).
otherwise.
assoce(Check, [Head|Rest], Result) :- true |
     assoce3(Check, Head, Rest, Result).

assoce3(Check, Head, Rest, Result) :- diff(Check, Head) |
     assoce(Check, Rest, R2), Result = R2.
otherwise.
assoce3(Check, Head, Rest, Result) :- true | Result = [].

assoce2(Check, Head, Rest, [], R) :- true | assoce(Check, Rest, R).
otherwise.
```

```
assoce2(Check, Head, Rest, J, R) :- true | R=[ ].

%
%
%
% List equality check if equal then true, else [ ].

equal(X, Y, R) :-true | diff_list(X, Y, R).

diff_list(X, Y, R) :- list(X) | diff5(X, Y, R2), R=R2.
diff_list(X, Y, R) :- diff(X, Y) | R=[ ].
otherwise.
diff_list(X, Y, R) :  true | R=true.

diff5(X, Y, R) :- list(Y) | diff3(X, Y, R2), R=R2.
otherwise.
diff5(X, Y, R) :- true | R=[ ].

diff3([ ], [ ], R) :- true | R=true.
diff3([X|R1], [Y|R2], R) :- true | diff_list(X, Y, R3),
     diff4(R3, R1, R2, R4), R=R4.
otherwise.
diff3(X, Y, R) :- true | R= [ ].

diff4(R3, X, Y, R) :- diff(R3, [ ]) | diff3(X, Y, R2), R=R2.
otherwise.
diff4(R3, X, Y, R) :- true | R=[ ].

%
%

output_group(E) :- true |
E=[
[深溝玉軸受,
      [ラジアル荷重, 可能],
      [アキシアル荷重, 可能],
      [アキシアル荷重方向, 両方向],
      [合成荷重, 可能],
      [高速回転, 特に可能],
      [高精度, 特に可能],
      [低雑音と低トルク, 特に可能],
      [剛性, undefined],
      [内輪と外輪の許容傾き, 十分に可能],
      [調心作用, undefined],
      [内輪と外輪の分離, undefined],
      [固定側用, 適用可],
      [自由側用, 軸伸縮逃がせば可],
      [内輪テーパ穴, undefined]],
[アンギュラ玉軸受,
      [ラジアル荷重, 十分に可能],
      [アキシアル荷重, 十分に可能],
      [アキシアル荷重方向, 一方向],
      [合成荷重, 十分に可能],
      [高速回転, 特に可能],
      [高精度, 特に可能],
      [低雑音と低トルク, undefined],
      [剛性, undefined],
      [内輪と外輪の許容傾き, 少し可能],
      [調心作用, undefined],
      [内輪と外輪の分離, undefined],
      [固定側用, undefined],
      [自由側用, undefined],
```

「内輪テーパ穴，　ｕｎｄｅｆｉｎｅｄ］］，
［同筒ころ軸受，
［ラジアル荷重，　十分に可能］，
［アキシアル荷重，　不可］，
［合成荷重，　不可］，
［高速回転，　特に可能］，
［高精度，　特に可能］，
［低雑音と低トルク，　十分に可能］，
［剛性，　十分に可能］，
［内輪と外輪の許容傾き，　可能］，
［調心作用，　ｕｎｄｅｆｉｎｅｄ］，
［内輪と外輪の分離，　適用可能］，
［固定側用，　ｕｎｄｅｆｉｎｅｄ］，
［自由側用，　適用可能］，
［内輪テーパ穴，　ｕｎｄｅｆｉｎｅｄ］］，
［組合わせアンギュラ玉軸受，
［ラジアル荷重，　十分に可能］，
［アキシアル荷重，　十分に可能］，
［アキシアル荷重方向，　両方向］，
［合成荷重，　十分に可能］，
［高速回転，　十分に可能］，
［高精度，　特に可能］，
［低雑音と低トルク，　ｕｎｄｅｆｉｎｅｄ］，
［剛性，　十分に可能］，
［内輪と外輪の許容傾き，　少し可能］，
［調心作用，　ｕｎｄｅｆｉｎｅｄ］，
［内輪と外輪の分離，　ｕｎｄｅｆｉｎｅｄ］，
［固定側用，　適用可］，
［自由側用，　軸伸縮隙があれば可］，
［内輪テーパ穴，　ｕｎｄｅｆｉｎｅｄ］］，
［複列円筒ころ軸受，
［ラジアル荷重，　特に可能］，
［アキシアル荷重，　不可］，
［合成荷重，　不可］，
［高速回転，　十分に可能］，
［高精度，　特に可能］，
［低雑音と低トルク，　ｕｎｄｅｆｉｎｅｄ］，
［剛性，　特に可能］，
［内輪と外輪の許容傾き，　少し可能］，
［調心作用，　ｕｎｄｅｆｉｎｅｄ］，
［内輪と外輪の分離，　適用可能］，
［固定側用，　ｕｎｄｅｆｉｎｅｄ］，
［自由側用，　適用可能］，
［内輪テーパ穴，　適用可能］］

```
% 1990.2.27 Version
%
%    ind33.kl1: Simple Induction Argolithm(3)
%                  属性値入替えオペレーションの推定
%
:- with_macro pimos.
:-module ind3.
:-public goal/1.
     wd/2, mk_io_list/4, orcheck/2, output_group/1, pick up/2,
     input_group/1, efface/2, flat/2, orarrange/2, io_list/4,
     mk_io pair/4,  cklist/4, neg_arrange/2, mk_io_pair2/4.

wd(L,S) :- true | w_create(L,S).

w_create(Label, Str) :- true |
     shoen:raise(pimos_tag#task, general_request, GRD),
     GRD=[window(WRD)],
     w_create1(WRD, Label, Str).

w_create1(normal(Window, _, _), Label, Str) :- true |
     Window=[create(WD)],
     w_create2(WD, Label, Str).
otherwise.
w_create1(_,_,_) :-
     display_console('*** window create fail (no. 1) ***') |
     true.

w_create2(normal(W2, _, _), Label, Str) :- true |
   W2=[reshape(mouse,mouse,normal),
     set_font(string#"font:kanji_16",normal),
     set_title(Label, normal),
     activate(normal)|W3],
   buffer:interaction_filter(Str, W3).
otherwise.
w_create2(_,_,_) :-
     display_console('*** window create fail (no. 2) ***') |
     true.

%
%        Main Program induction 3
%
goal(Str) :- true |
     w_create("Induction", Str),
     subgoal(Att, Mat),
     display_list(Att, Mat, Str).
%

display_list([],[],St) :- true | St=[nl, flush(_),getc(_)].

display_list([Att|AR],[Mat|MR], St) :- true |
     dispe2(Att, Mat, St1),
     display_list(AR, MR, St2),
     St = [do(St1)|St2].

% if M contains [[a, uncertain]..] this attribute is no good
% if M contains [], [] must be eliminated.
dispe2(A,M, St) :- true |
     uncertain(M, Judge), dispe3(A,M,Judge, St).

dispe3(A,M, fail, St) :- true | St=[].
dispe3(A,[], _, St) :- true | St=[].
```

```
otherwise.
dispe3(A, M, _, St) :- true | neg_nil(M, M2), dispe4(A, M2, St).

dispe4(A, [ ], St) :- true | St=[ ].
otherwise.
dispe4(A, M, St) :- true |
     St=[nl, putb(string#"replacement criterion "),
     putt(A), putb(string#"= "), putt(M), nl].

uncertain([ ], J) :- true | J = true.
uncertain([[X, uncertain]|Rest], J) :- true | J= fail.
otherwise.
uncertain([X|Y], J) :- true | uncertain(Y, J).

neg_nil([ ], Y) :- true | Y= [ ].
neg_nil([[ ]|X], Y) :- true | neg_nil(X, Y).
otherwise.
neg_nil([A|X], Y) :- true | Y=[A|Y2], neg_nil(X, Y2).

%
%
subgoal(Att_list, Result) :- true |
     input_group(Pi),
     output_group(Po),
     pick_up(Po, Att1),
     flat(Att1, Att0),
     efface(Att0, Att_list), % same input and output about attribute
%
% made_up all attribute list= Att_list
%
% hypothesis = odering is same between input and output
%    make_value_list_for_each_attribute of input and output
     mk_io_list(Att_list, Pi, Po, Att2).
% Att2 : [ [input-value, output-value], [i o], [i o], ...]
%          [ [i o] [i o, ..] ...]
% top order is attribute order (at1, at2, at3)
% inside order is an element order in input group = output group
%
% if i = o , then [i, negative]
%
% [ [[i o] [i o], ... [..] .. ] -> if there is a same pair it will be [ ].
% i is same , but o is different , it will be [ ].
% After all, [x, negative] > [ ].
     orcheck(Att2, Att3). % Att3 has no inconsistent io.
     Result=Att3.
%
%
mk_io_list([ ], Pi, Po, Att2) :- true | Att2=[ ].
mk_io_list([A1|Att], Pi, Po, Att2) :- true | io_list(A1, Pi, Po, R),
     mk_io_list(Att, Pi, Po, R2), Att2=[R|R2].

io_list(A1, [ ], [ ], R) :- true | R = [ ].
io_list(A1, [Ei|Pi], [Eo|Po], R) :- true |
     mk_io_pair(A1, Ei, Eo, R2),
     io_list(A1, Pi, Po, R3), R=[R2|R3].

mk_io_pair(A1, [Name1|At1], [Name2|At2], R) :- true |
     mk_io_pair2(A1, At1, At2, R).

mk_io_pair2(A1, [ ], [ ], R) :- true | R = undefined.
mk_io_pair2(A1, [[A1, V1]|Rest1], [[A1, V2]|Rest2], R) :- diff(V1, V2) |
     R=[V1, V2].
```

```
mk_io_pair2(A1, [[A1,V1]|Rest1], [[A1,V1]|Rest2], R) :- true |
    R=[V1, negative].
% negative is not replaced value, it is negative example
otherwise.
mk_io_pair2(A1, [A2|Rest1], [A3|Rest2], R) :- true |
    mk_io_pair2(A1, Rest1, Rest2, R).

%
orcheck([], R) :- true | R=[].
orcheck([L_of_pair|Rest], R) :- true | orarrange(L_of_pair, O1),
    neg_arrange(O1, O2), orcheck(Rest, R2), R=[O2|R2].

orarrange([], R) :- true | R=[].
orarrange([[I,O]|Rest], R) :- true | cklist(I,O, Rest, R2),
    orarrange(Rest, R3), R=[R2|R3].
otherwise.
orarrange([A|Rest], R) :- true | R=[[A,negative]|R2],
    orarrange(Rest, R2).

cklist(I,O, [], R) :- true | R=[I,O].
cklist(I,O, [[I,O2]|Rest], R) :- diff(O,O2) | R=[I, uncertain].
cklist(I,O, [[I,O]|Rest], R) :- true | R=[].
otherwise.
cklist(I,O, [A|Rest], R) :- true | cklist(I,O, Rest, R).

neg_arrange([], R) :- true | R=[].
neg_arrange([[X,negative]|Rest], R) :- true| neg_arrange(Rest,R).
neg_arrange([Y|Rest], R) :- true | R=[Y|R2], neg_arrange(Rest, R2).


%
%

pick_up([],Y) :- true | Y=[].
pick_up([X|Y],Z) :- true |
    pick_up2(X,X2), pick_up(Y,Z2), Z=[X2|Z2].
%
pick_up2([X|X2],Y) :- true |
    pick_up3(X2,Y2), Y=Y2. % X is element name
pick_up3([],Y) :- true | Y=[].
pick_up3([X|Y],Z) :- true |
    pick_up4(X,Z1), pick_up3(Y,Z2), Z=[Z1|Z2].

pick_up4([A|V], Z) :- true | Z=A.

flat([],Y) :- true | Y=[].
flat([Org|Rest], Dest) :- list(Org) | flat(Org, D2),
    flat(Rest, D3),
    append(D2,D3,D4), Dest=D4.
otherwise.
flat([Org|Rest], Dest) :- true | Dest=[Org|D2], flat(Rest, D2).

append([],Y,Z) :- true | Z = Y.
append([A|X],Y,Z) :- true | Z=[A|Z2], append(X,Y,Z2).

efface([],Y) :- true | Y=[].
efface([Check|Rest], Dest) :- true |
    assoce(Check, Rest, C2),
    efface(Rest, C3), without_nil(C2, C3, Dest).

without_nil([],Y,Z) :- true | Z = Y.
otherwise.
```

```
without_nil(X, Y, Z)  :- true | Z = [X|Y].

assoce(Check, [], Result)  :- true | Result = Check.
assoce(Check, [Head|Rest], Result)  :- list(Check) |
     equal(Check, Head, J),
     assoce2(Check, Head, Rest, J, Result).
otherwise.
assoce(Check, [Head|Rest], Result)  :- true |
     assoce3(Check, Head, Rest, Result).

assoce3(Check, Head, Rest, Result)  :- diff(Check, Head) |
     assoce(Check, Rest, R2), Result = R2.
otherwise.
assoce3(Check, Head, Rest, Result)  :- true | Result = [].

assoce2(Check, Head, Rest, [], R)  :- true | assoce(Check, Rest, R).
otherwise.
assoce2(Check, Head, Rest, J, R)  :- true | R=[].

%
%
%
% List equality check if equal then true, else [].

equal(X, Y, R)  :-true | diff_list(X, Y, R).

diff_list(X, Y, R)  :- list(X) | diff5(X, Y, R2), R=R2.
diff_list(X, Y, R)  :- diff(X, Y)  | R=[].
otherwise.
diff_list(X, Y, R)  :- true | R=true.

diff5(X, Y, R)  :- list(Y)  | diff3(X, Y, R2), R=R2.
otherwise.
diff5(X, Y, R)  :- true | R=[].

diff3([], [], R)  :- true | R=true.
diff3([X|R1], [Y|R2], R)  :- true | diff_list(X, Y, R3),
     diff4(R3, R1, R2, R4), R=R4.
otherwise.
diff3(X, Y, R)  :- true | R=[].

diff4(R3, X, Y, R)  :- diff(R3, []) | diff3(X, Y, R2), R=R2.
otherwise.
diff4(R3, X, Y, R)  :- true | R=[].

%
%
%input_group(E)  :- true |
 %    E = [
 %   [name1, [at1, 1], [at2, 2], [at3, 3]],
 %   [name2, [at1, 1], [at2, 2], [at3, 3]],
 %   [name3, [at1, 2], [at2, 3], [at3, 4]],
 %   [name4, [at1, 2], [at2, 3], [at3, 4]],
 %   [name5, [at1, 3], [at2, 4], [at3, 5]],
 %   [name6, [at1, 3], [at2, 4], [at3, 5]] ].
%output_group(E)  :- true |
 %    E = [
  %   [name1, [at1, 2], [at2, 2], [at3, 4]],
    %  [name2, [at1, 2], [at2, 3], [at3, 5]],
%      [name3, [at1, 3], [at2, 10], [at3, 5]],
 %    [name4, [at1, 3], [at2, 4], [at3, 5]],
  %   [name5, [at1, 4], [at2, 5], [at3, 6]],
```

```
    %  [name6,  [at1,  4], [at2,  5], [at3,  6]] ].
%
%
input_group(E)  :-  true  |
E = [
 [深溝玉軸受,
        [ラジアル荷重,  可能],
        [アキシアル荷重,  可能],
        [アキシアル荷重方向,  両方向],
        [合成荷重,  可能],
        [高速回転,  特に可能],
        [高精度,  特に可能],
        [低雑音と低トルク,  特に可能],
        [剛性,  undefined],
        [内輪と外輪の許容傾き,  十分に可能],
        [調心作用,  undefined],
        [内輪と外輪の分離,  undefined],
        [固定側用,  適用可],
        [自由側用,  軸伸縮逃がせば可],
        [内輪テーパ穴,  undefined]],
 [アンギュラ玉軸受,
        [ラジアル荷重,  十分に可能],
        [アキシアル荷重,  十分に可能],
        [アキシアル荷重方向,  一方向],
        [合成荷重,  十分に可能],
        [高速回転,  特に可能],
        [高精度,  特に可能],
        [低雑音と低トルク,  undefined],
        [剛性,  undefined],
        [内輪と外輪の許容傾き,  少し可能],
        [調心作用,  undefined],
        [内輪と外輪の分離,  undefined],
        [固定側用,  undefined],
        [自由側用,  undefined],
        [内輪テーパ穴,  undefined]],
 [同筒ころ軸受,
        [ラジアル荷重,  十分に可能],
        [アキシアル荷重,  不可],
        [合成荷重,  不可],
        [高速回転,  特に可能],
        [高精度,  特に可能],
        [低雑音と低トルク,  十分に可能],
        [剛性,  十分に可能],
        [内輪と外輪の許容傾き,  可能],
        [調心作用,  undefined],
        [内輪と外輪の分離,  適用可能],
        [固定側用,  undefined],
        [自由側用,  適用可能],
        [内輪テーパ穴,  undefined]],
 [組合わせアンギュラ玉軸受,
        [ラジアル荷重,  十分に可能],
        [アキシアル荷重,  十分に可能],
        [アキシアル荷重方向,  両方向],
        [合成荷重,  十分に可能],
        [高速回転,  十分に可能],
        [高精度,  特に可能],
        [低雑音と低トルク,  undefined],
        [剛性,  十分に可能],
        [内輪と外輪の許容傾き,  少し可能],
        [調心作用,  undefined],
        [内輪と外輪の分離,  undefined],
        [固定側用,  適用可],
```

```
        [自由側用,　軸伸縮逃がせば可],
        [内輪テーパ穴,　undefined]],
    [複列同筒ころ軸受,
        [ラジアル荷重,　特に可能],
        [アキシアル荷重,　不可],
        [合成荷重,　不可],
        [高速回転,　十分に可能],
        [高精度,　特に可能],
        [低雑音と低トルク,　undefined],
        [剛性,　特に可能],
        [内輪と外輪の許容傾き,　少し可能],
        [調心作用,　undefined],
        [内輪と外輪の分離,　適用可能],
        [固定側用,　undefined],
        [自由側用,　適用可能],
        [内輪テーパ穴,　適用可能]]
];

output_group(E) :- true |
E = [
    [深溝玉軸受,
        [ラジアル荷重,　可能],
        [アキシアル荷重,　可能],
        [アキシアル荷重方向,　両方向],
        [合成荷重,　可能],
        [高速回転,　range4],
        [高精度,　特に可能],
        [低雑音と低トルク,　特に可能],
        [剛性,　undefined],
        [内輪と外輪の許容傾き,　十分に可能],
        [調心作用,　undefined],
        [内輪と外輪の分離,　undefined],
        [固定側用,　適用可],
        [自由側用,　軸伸縮逃がせば可],
        [内輪テーパ穴,　undefined]],
    [アンギュラ玉軸受,
        [ラジアル荷重,　十分に可能],
        [アキシアル荷重,　十分に可能],
        [アキシアル荷重方向,　一方向],
        [合成荷重,　十分に可能],
        [高速回転,　range4],
        [高精度,　特に可能],
        [低雑音と低トルク,　undefined],
        [剛性,　undefined],
        [内輪と外輪の許容傾き,　少し可能],
        [調心作用,　undefined],
        [内輪と外輪の分離,　undefined],
        [固定側用,　undefined],
        [自由側用,　undefined],
        [内輪テーパ穴,　undefined]],
    [同筒ころ軸受,
        [ラジアル荷重,　十分に可能],
        [アキシアル荷重,　不可],
        [合成荷重,　不可],
        [高速回転,　range4],
        [高精度,　特に可能],
        [低雑音と低トルク,　十分に可能],
        [剛性,　十分に可能],
        [内輪と外輪の許容傾き,　可能],
        [調心作用,　undefined],
        [内輪と外輪の分離,　適用可能],
        [固定側用,　undefined],
```

[自由側用，　適用可能]，
[内輪テーパ穴，　ｕｎｄｅｆｉｎｅｄ]]，
[組合わせアンギュラ玉軸受，
[ラジアル荷重，　十分に可能]，
[アキシアル荷重，　十分に可能]，
[アキシアル荷重方向，　両方向]，
[合成荷重，　十分に可能]，
[高速回転，　ｒａｎｇｅ２]，
[高精度，　特に可能]，
[低騒音と低トルク，　ｕｎｄｅｆｉｎｅｄ]，
[剛性，　十分に可能]，
[内輪と外輪の許容傾き，　少し可能]，
[調心作用，　ｕｎｄｅｆｉｎｅｄ]，
[内輪と外輪の分離，　ｕｎｄｅｆｉｎｅｄ]，
[固定側用，　適用可]，
[自由側用，　軸伸縮逃がせば可]，
[内輪テーパ穴，　ｕｎｄｅｆｉｎｅｄ]]，
[複列同荷こう軸受，
[ラジアル荷重，　特に可能]，
[アキシアル荷重，　不可]，
[合成荷重，　不可]，
[高速回転，　ｒａｎｇｅ２]，
[高精度，　特に可能]，
[低騒音と低トルク，　ｕｎｄｅｆｉｎｅｄ]，
[剛性，　特に可能]，
[内輪と外輪の許容傾き，　少し可能]，
[調心作用，　ｕｎｄｅｆｉｎｅｄ]，
[内輪と外輪の分離，　適用可能]，
[固定側用，　ｕｎｄｅｆｉｎｅｄ]，
[自由側用，　適用可能]，
[内輪テーパ穴，　適用可能]]
]．

```
%
%     Example File No.1 :  選択オペレーションの推定の例1
%
%
:-module index.
:-public input_group/1, output_group/1.

input_group(E) :- true |

E=[
[溝溝玉軸受,
        [ラジアル荷重,  可能],
        [アキシアル荷重,  可能],
        [アキシアル荷重方向,  両方向],
        [合成荷重,  可能],
        [高速回転,  特に可能],
        [高精度,  特に可能],
        [低雑音と低トルク,  特に可能],
        [剛性,  undefined],
        [内輪と外輪の許容傾き,  十分に可能],
        [調心作用,  undefined],
        [内輪と外輪の分離,  undefined],
        [固定側用,  適用可],
        [自由側用,  軸伸縮逃がせば可],
        [内輪テーパ穴,  undefined]],

[マグネット玉軸受,
        [ラジアル荷重,  少し可能],
        [アキシアル荷重,  少し可能],
        [アキシアル荷重方向,  一方向],
        [合成荷重,  可能],
        [高速回転,  十分に可能],
        [高精度,  undefined],
        [低雑音と低トルク,  undefined],
        [剛性,  undefined],
        [内輪と外輪の許容傾き,  少し可能],
        [調心作用,  undefined],
        [内輪と外輪の分離,  適用可],
        [固定側用,  undefined],
        [自由側用,  undefined],
        [内輪テーパ穴,  undefined]],

[アンギュラ玉軸受,
        [ラジアル荷重,  十分に可能],
        [アキシアル荷重,  十分に可能],
        [アキシアル荷重方向,  一方向],
        [合成荷重,  十分に可能],
        [高速回転,  特に可能],
        [高精度,  特に可能],
        [低雑音と低トルク,  undefined],
        [剛性,  undefined],
        [内輪と外輪の許容傾き,  少し可能],
        [調心作用,  undefined],
        [内輪と外輪の分離,  undefined],
        [固定側用,  undefined],
        [自由側用,  undefined],
        [内輪テーパ穴,  undefined]],

[複列アンギュラ玉軸受,
        [ラジアル荷重,  十分に可能],
        [アキシアル荷重,  十分に可能],
```

[アキシアル荷重方向，　両方向]，
[合成荷重，　十分に可能]，
[高速回転，　可能]，
[高精度，　ｕｎｄｅｆｉｎｅｄ]，
[低雑音と低トルク，　ｕｎｄｅｆｉｎｅｄ]，
[剛性，　ｕｎｄｅｆｉｎｅｄ]，
[内輪と外輪の許容傾き，　少し可能]，
[調心作用，　ｕｎｄｅｆｉｎｅｄ]，
[内輪と外輪の分離，　ｕｎｄｅｆｉｎｅｄ]，
[固定側用，　適用可]，
[自由側用，　軸伸縮逃がせば可]，
[内輪テーパ穴，　ｕｎｄｅｆｉｎｅｄ]]，

[組合わせアンギュラ玉軸受，
[ラジアル荷重，　十分に可能]，
[アキシアル荷重，　十分に可能]，
[アキシアル荷重方向，　両方向]，
[合成荷重，　十分に可能]，
[高速回転，　十分に可能]，
[高精度，　特に可能]，
[低雑音と低トルク，　ｕｎｄｅｆｉｎｅｄ]，
[剛性，　十分に可能]，
[内輪と外輪の許容傾き，　少し可能]，
[調心作用，　ｕｎｄｅｆｉｎｅｄ]，
[内輪と外輪の分離，　ｕｎｄｅｆｉｎｅｄ]，
[固定側用，　適用可]，
[自由側用，　軸伸縮逃がせば可]，
[内輪テーパ穴，　ｕｎｄｅｆｉｎｅｄ]]，

[自動調心軸受，
[ラジアル荷重，　可能]，
[アキシアル荷重，　少し可能]，
[アキシアル荷重方向，　両方向]，
[合成荷重，　少し可能]，
[高速回転，　十分に可能]，
[高精度，　ｕｎｄｅｆｉｎｅｄ]，
[低雑音と低トルク，　ｕｎｄｅｆｉｎｅｄ]，
[剛性，　ｕｎｄｅｆｉｎｅｄ]，
[内輪と外輪の許容傾き，　特に可能]，
[調心作用，　適用可能]，
[内輪と外輪の分離，　ｕｎｄｅｆｉｎｅｄ]，
[固定側用，　適用可能]，
[自由側用，　軸伸縮逃がせば可]，
[内輪テーパ穴，　適用可能]]，

[円筒ころ軸受，
[ラジアル荷重，　十分に可能]，
[アキシアル荷重，　不可]，
[合成荷重，　不可]，
[高速回転，　特に可能]，
[高精度，　特に可能]，
[低雑音と低トルク，　十分に可能]，
[剛性，　十分に可能]，
[内輪と外輪の許容傾き，　可能]，
[調心作用，　ｕｎｄｅｆｉｎｅｄ]，
[内輪と外輪の分離，　適用可能]，
[固定側用，　ｕｎｄｅｆｉｎｅｄ]，
[自由側用，　適用可能]，
[内輪テーパ穴，　ｕｎｄｅｆｉｎｅｄ]]，

[複列円筒ころ軸受，
[ラジアル荷重，　特に可能]，

[アキシアル荷重，　不可]，
[合成荷重，　不可]，
[高速回転，　十分に可能]，
[高精度，　特に可能]，
[低雑音と低トルク，　ｕｎｄｅｆｉｎｅｄ]，
[剛性，　特に可能]，
[内輪と外輪の許容傾き，　少し可能]，
[調心作用，　ｕｎｄｅｆｉｎｅｄ]，
[内輪と外輪の分離，　適用可能]，
[固定側用，　ｕｎｄｅｆｉｎｅｄ]，
[自由側用，　適用可能]，
[内輪テーパ穴，　適用可能]｝，

[片つば付円筒ころ軸受，
[ラジアル荷重，　十分に可能]，
[アキシアル荷重，　可能]，
[アキシアル荷重方向，　一方向]，
[合成荷重，　可能]，
[高速回転，　十分に可能]，
[高精度，　ｕｎｄｅｆｉｎｅｄ]，
[低雑音と低トルク，　ｕｎｄｅｆｉｎｅｄ]，
[剛性，　十分に可能]，
[内輪と外輪の許容傾き，　可能]，
[調心作用，　ｕｎｄｅｆｉｎｅｄ]，
[内輪と外輪の分離，　適用可能]，
[固定側用，　ｕｎｄｅｆｉｎｅｄ]，
[自由側用，　ｕｎｄｅｆｉｎｅｄ]，
[内輪テーパ穴，　ｕｎｄｅｆｉｎｅｄ]｝
]．

ｏｕｔｐｕｔ＿ｇｒｏｕｐ（Ｅ）　：－　ｔｒｕｅ　｜
Ｅ＝［
[深溝玉軸受，
[ラジアル荷重，　可能]，
[アキシアル荷重，　可能]，
[アキシアル荷重方向，　両方向]，
[合成荷重，　可能]，
[高速回転，　特に可能]，
[高精度，　特に可能]，
[低雑音と低トルク，　特に可能]，
[剛性，　ｕｎｄｅｆｉｎｅｄ]，
[内輪と外輪の許容傾き，　十分に可能]，
[調心作用，　ｕｎｄｅｆｉｎｅｄ]，
[内輪と外輪の分離，　ｕｎｄｅｆｉｎｅｄ]，
[固定側用，　適用可]，
[自由側用，　軸伸縮逃がせば可]，
[内輪テーパ穴，　ｕｎｄｅｆｉｎｅｄ]]，
[アンギュラ玉軸受，
[ラジアル荷重，　十分に可能]，
[アキシアル荷重，　十分に可能]，
[アキシアル荷重方向，　一方向]，
[合成荷重，　十分に可能]，
[高速回転，　特に可能]，
[高精度，　特に可能]，
[低雑音と低トルク，　ｕｎｄｅｆｉｎｅｄ]，
[剛性，　ｕｎｄｅｆｉｎｅｄ]，
[内輪と外輪の許容傾き，　少し可能]，
[調心作用，　ｕｎｄｅｆｉｎｅｄ]，
[内輪と外輪の分離，　ｕｎｄｅｆｉｎｅｄ]，
[固定側用，　ｕｎｄｅｆｉｎｅｄ]，
[自由側用，　ｕｎｄｅｆｉｎｅｄ]，
[内輪テーパ穴，　ｕｎｄｅｆｉｎｅｄ]]，

[組合わせアンギュラ玉軸受,
[ラジアル荷重, 十分に可能],
[アキシアル荷重, 十分に可能],
[アキシアル荷重方向, 両方向],
[合成荷重, 十分に可能],
[高速回転, 十分に可能],
[高精度, 特に可能],
[低雑音と低トルク, undefined],
[剛性, 十分に可能],
[内輪と外輪の許容傾き, 少し可能],
[調心作用, undefined],
[内輪と外輪の分離, undefined],
[固定側用, 適用可],
[自由側用, 軸伸縮逃がせば可],
[内輪テーパ穴, undefined]],
[同筒ころ軸受,
[ラジアル荷重, 十分に可能],
[アキシアル荷重, 不可],
[合成荷重, 不可],
[高速回転, 特に可能],
[高精度, 特に可能],
[低雑音と低トルク, 十分に可能],
[剛性, 十分に可能],
[内輪と外輪の許容傾き, 可能],
[調心作用, undefined],
[内輪と外輪の分離, 適用可能],
[固定側用, undefined],
[自由側用, 適用可能],
[内輪テーパ穴, undefined]],
[複列同筒ころ軸受,
[ラジアル荷重, 特に可能],
[アキシアル荷重, 不可],
[合成荷重, 不可],
[高速回転, 十分に可能],
[高精度, 特に可能],
[低雑音と低トルク, undefined],
[剛性, 特に可能],
[内輪と外輪の許容傾き, 少し可能],
[調心作用, undefined],
[内輪と外輪の分離, 適用可能],
[固定側用, undefined],
[自由側用, 適用可能],
[内輪テーパ穴, 適用可能]]

```
%
%     Example File No.2 :  選択オペレーションの推定の例2
%
%
:-module index.
:-public input_group/1, output_group/1.

input_group(K) :- true |

K=[
 [深溝玉軸受,
        [ラジアル荷重,  可能],
        [アキシアル荷重,  可能],
        [アキシアル荷重方向,  両方向],
        [合成荷重,  可能],
        [高速回転,  特に可能],
        [高精度,  特に可能],
        [低雑音と低トルク,  特に可能],
        [剛性,  undefined],
        [内輪と外輪の許容傾き,  十分に可能],
        [調心作用,  undefined],
        [内輪と外輪の分離,  undefined],
        [固定側用,  適用可],
        [自由側用,  軸伸縮逃がせば可],
        [内輪テーパ穴,  undefined]],

 [マグネット玉軸受,
        [ラジアル荷重,  少し可能],
        [アキシアル荷重,  少し可能],
        [アキシアル荷重方向,  一方向],
        [合成荷重,  可能],
        [高速回転,  十分に可能],
        [高精度,  undefined],
        [低雑音と低トルク,  undefined],
        [剛性,  undefined],
        [内輪と外輪の許容傾き,  少し可能],
        [調心作用,  undefined],
        [内輪と外輪の分離,  適用可],
        [固定側用,  undefined],
        [自由側用,  undefined],
        [内輪テーパ穴,  undefined]],

 [アンギュラ玉軸受,
        [ラジアル荷重,  十分に可能],
        [アキシアル荷重,  十分に可能],
        [アキシアル荷重方向,  一方向],
        [合成荷重,  十分に可能],
        [高速回転,  特に可能],
        [高精度,  特に可能],
        [低雑音と低トルク,  undefined],
        [剛性,  undefined],
        [内輪と外輪の許容傾き,  少し可能],
        [調心作用,  undefined],
        [内輪と外輪の分離,  undefined],
        [固定側用,  undefined],
        [自由側用,  undefined],
        [内輪テーパ穴,  undefined]],

 [複列アンギュラ玉軸受,
        [ラジアル荷重,  十分に可能],
        [アキシアル荷重,  十分に可能],
```

[アキシアル荷重方向, 両方向],
[合成荷重, 十分に可能],
[高速回転, 可能],
[高精度, undefined],
[低雑音と低トルク, undefined],
[剛性, undefined],
[内輪と外輪の許容傾き, 少し可能],
[調心作用, undefined],
[内輪と外輪の分離, undefined],
[固定側用, 適用可],
[自由側用, 軸伸縮逃がせば可],
[内輪テーパ穴, undefined]],

[組合わせアンギュラ玉軸受,
[ラジアル荷重, 十分に可能],
[アキシアル荷重, 十分に可能],
[アキシアル荷重方向, 両方向],
[合成荷重, 十分に可能],
[高速回転, 十分に可能],
[高精度, 特に可能],
[低雑音と低トルク, undefined],
[剛性, 十分に可能],
[内輪と外輪の許容傾き, 少し可能],
[調心作用, undefined],
[内輪と外輪の分離, undefined],
[固定側用, 適用可],
[自由側用, 軸伸縮逃がせば可],
[内輪テーパ穴, undefined]],

[自動調心軸受,
[ラジアル荷重, 可能],
[アキシアル荷重, 少し可能],
[アキシアル荷重方向, 両方向],
[合成荷重, 少し可能],
[高速回転, 十分に可能],
[高精度, undefined],
[低雑音と低トルク, undefined],
[剛性, undefined],
[内輪と外輪の許容傾き, 特に可能],
[調心作用, 適用可能],
[内輪と外輪の分離, undefined],
[固定側用, 適用可能],
[自由側用, 軸伸縮逃がせば可],
[内輪テーパ穴, 適用可能]],

[円筒ころ軸受,
[ラジアル荷重, 十分に可能],
[アキシアル荷重, 不可],
[合成荷重, 不可],
[高速回転, 特に可能],
[高精度, 特に可能],
[低雑音と低トルク, 十分に可能],
[剛性, 十分に可能],
[内輪と外輪の許容傾き, 可能],
[調心作用, undefined],
[内輪と外輪の分離, 適用可能],
[固定側用, undefined],
[自由側用, 適用可能],
[内輪テーパ穴, undefined]],

[複列円筒ころ軸受,
[ラジアル荷重, 特に可能],

```
        [アキシアル荷重,　不可],
        [合成荷重,　不可],
        [高速回転,　十分に可能],
        [高精度,　特に可能],
        [低雑音と低トルク,　undefined],
        [剛性,　特に可能],
        [内輪と外輪の許容傾き,　少し可能],
        [調心作用,　undefined],
        [内輪と外輪の分離,　適用可能],
        [固定側用,　undefined],
        [自由側用,　適用可能],
        [内輪テーパ穴,　適用可能]],

    [片方向性円筒ころ軸受,
        [ラジアル荷重,　十分に可能],
        [アキシアル荷重,　可能],
        [アキシアル荷重方向,　一方向],
        [合成荷重,　可能],
        [高速回転,　十分に可能],
        [高精度,　undefined],
        [低雑音と低トルク,　undefined],
        [剛性,　十分に可能],
        [内輪と外輪の許容傾き,　可能],
        [調心作用,　undefined],
        [内輪と外輪の分離,　適用可能],
        [固定側用,　undefined],
        [自由側用,　undefined],
        [内輪テーパ穴,　undefined]]
].

output_group(E)　:-　true　|
E=[
[深溝玉軸受,
        [ラジアル荷重,　可能],
        [アキシアル荷重,　可能],
        [アキシアル荷重方向,　両方向],
        [合成荷重,　可能],
        [高速回転,　特に可能],
        [高精度,　特に可能],
        [低雑音と低トルク,　特に可能],
        [剛性,　undefined],
        [内輪と外輪の許容傾き,　十分に可能],
        [調心作用,　undefined],
        [内輪と外輪の分離,　undefined],
        [固定側用,　適用可],
        [自由側用,　軸伸縮逃がせば可],
        [内輪テーパ穴,　undefined]],
[アンギュラ玉軸受,
        [ラジアル荷重,　十分に可能],
        [アキシアル荷重,　十分に可能],
        [アキシアル荷重方向,　一方向],
        [合成荷重,　十分に可能],
        [高速回転,　特に可能],
        [高精度,　特に可能],
        [低雑音と低トルク,　undefined],
        [剛性,　undefined],
        [内輪と外輪の許容傾き,　少し可能],
        [調心作用,　undefined],
        [内輪と外輪の分離,　undefined],
        [固定側用,　undefined],
        [自由側用,　undefined],
        [内輪テーパ穴,　undefined]]
```

```
%
%     Example File No.3 :  選択オペレーションの推定の例3
%
%
:-module index.
:-public input_group/1, output_group/1.

input_group(E) :- true !.

E=[
[深溝玉軸受,
        [ラジアル荷重,  可能],
        [アキシアル荷重,  可能],
        [アキシアル荷重方向,  両方向],
        [合成荷重,  可能],
        [高速回転,  特に可能],
        [高精度,  特に可能],
        [低雑音と低トルク,  特に可能],
        [剛性,  undefined],
        [内輪と外輪の許容傾き,  十分に可能],
        [調心作用,  undefined],
        [内輪と外輪の分離,  undefined],
        [固定側用,  適用可],
        [自由側用,  軸伸縮性があれば可],
        [内輪テーパ穴,  undefined]],

[マグネット玉軸受,
        [ラジアル荷重,  少し可能],
        [アキシアル荷重,  少し可能],
        [アキシアル荷重方向,  一方向],
        [合成荷重,  可能],
        [高速回転,  十分に可能],
        [高精度,  undefined],
        [低雑音と低トルク,  undefined],
        [剛性,  undefined],
        [内輪と外輪の許容傾き,  少し可能],
        [調心作用,  undefined],
        [内輪と外輪の分離,  適用可],
        [固定側用,  undefined],
        [自由側用,  undefined],
        [内輪テーパ穴,  undefined]],

[アンギュラ玉軸受,
        [ラジアル荷重,  十分に可能],
        [アキシアル荷重,  十分に可能],
        [アキシアル荷重方向,  一方向],
        [合成荷重,  十分に可能],
        [高速回転,  特に可能],
        [高精度,  特に可能],
        [低雑音と低トルク,  undefined],
        [剛性,  undefined],
        [内輪と外輪の許容傾き,  少し可能],
        [調心作用,  undefined],
        [内輪と外輪の分離,  undefined],
        [固定側用,  undefined],
        [自由側用,  undefined],
        [内輪テーパ穴,  undefined]],

[複列アンギュラ玉軸受,
        [ラジアル荷重,  十分に可能],
        [アキシアル荷重,  十分に可能],
```

[アキシアル荷重方向, 両方向],
[合成荷重, 十分に可能],
[高速回転, 可能],
[高精度, undefined],
[低雑音と低トルク, undefined],
[剛性, undefined],
[内輪と外輪の許容傾き, 少し可能],
[調心作用, undefined],
[内輪と外輪の分離, undefined],
[固定側用, 適用可],
[自由側用, 軸伸縮逃がせば可],
[内輪テーパ穴, undefined]],

[組合わせアンギュラ玉軸受,
[ラジアル荷重, 十分に可能],
[アキシアル荷重, 十分に可能],
[アキシアル荷重方向, 両方向],
[合成荷重, 十分に可能],
[高速回転, 十分に可能],
[高精度, 特に可能],
[低雑音と低トルク, undefined],
[剛性, 十分に可能],
[内輪と外輪の許容傾き, 少し可能],
[調心作用, undefined],
[内輪と外輪の分離, undefined],
[固定側用, 適用可],
[自由側用, 軸伸縮逃がせば可],
[内輪テーパ穴, undefined]],

[自動調心軸受,
[ラジアル荷重, 可能],
[アキシアル荷重, 少し可能],
[アキシアル荷重方向, 両方向],
[合成荷重, 少し可能],
[高速回転, 十分に可能],
[高精度, undefined],
[低雑音と低トルク, undefined],
[剛性, undefined],
[内輪と外輪の許容傾き, 特に可能],
[調心作用, 適用可能],
[内輪と外輪の分離, undefined],
[固定側用, 適用可能],
[自由側用, 軸伸縮逃がせば可],
[内輪テーパ穴, 適用可能]],

[同筒ころ軸受,
[ラジアル荷重, 十分に可能],
[アキシアル荷重, 不可],
[合成荷重, 不可],
[高速回転, 特に可能],
[高精度, 特に可能],
[低雑音と低トルク, 十分に可能],
[剛性, 十分に可能],
[内輪と外輪の許容傾き, 可能],
[調心作用, undefined],
[内輪と外輪の分離, 適用可能],
[固定側用, undefined],
[自由側用, 適用可能],
[内輪テーパ穴, undefined]],

[複列同筒ころ軸受,
[ラジアル荷重, 特に可能],

```
        [アキシアル荷重,　不可],
        [合成荷重,　不可],
        [高速回転,　十分に可能],
        [高精度,　特に可能],
        [低雑音と低トルク,　undefined],
        [剛性,　特に可能],
        [内輪と外輪の許容傾き,　少し可能],
        [調心作用,　undefined],
        [内輪と外輪の分離,　適用可能],
        [固定側用,　undefined],
        [自由側用,　適用可能],
        [内輪テーパ穴,　適用可能]],

    [片つば付円筒ころ軸受,
        [ラジアル荷重,　十分に可能],
        [アキシアル荷重,　可能],
        [アキシアル荷重方向,　一方向],
        [合成荷重,　可能],
        [高速回転,　十分に可能],
        [高精度,　undefined],
        [低雑音と低トルク,　undefined],
        [剛性,　十分に可能],
        [内輪と外輪の許容傾き,　可能],
        [調心作用,　undefined],
        [内輪と外輪の分離,　適用可能],
        [固定側用,　undefined],
        [自由側用,　undefined],
        [内輪テーパ穴,　undefined]]
    ].

output_group(E)　:-　true　|
E=[
    [組合わせアンギュラ玉軸受,
        [ラジアル荷重,　十分に可能],
        [アキシアル荷重,　十分に可能],
        [アキシアル荷重方向,　両方向],
        [合成荷重,　十分に可能],
        [高速回転,　十分に可能],
        [高精度,　特に可能],
        [低雑音と低トルク,　undefined],
        [剛性,　十分に可能],
        [内輪と外輪の許容傾き,　少し可能],
        [調心作用,　undefined],
        [内輪と外輪の分離,　undefined],
        [固定側用,　適用可],
        [自由側用,　軸伸縮逃がせば可],
        [内輪テーパ穴,　undefined]]],
    [複列円筒ころ軸受,
        [ラジアル荷重,　特に可能],
        [アキシアル荷重,　不可],
        [合成荷重,　不可],
        [高速回転,　十分に可能],
        [高精度,　特に可能],
        [低雑音と低トルク,　undefined],
        [剛性,　特に可能],
        [内輪と外輪の許容傾き,　少し可能],
        [調心作用,　undefined],
        [内輪と外輪の分離,　適用可能],
        [固定側用,　undefined],
        [自由側用,　適用可能],
        [内輪テーパ穴,　適用可能]]
```