ICOT Technical Memorandum: TM-0879

TM-0879

知識コンパイラの出力の並列化

寺崎　智

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

知識コンパイラの出力の並列化

寺崎 智

概要: 本報告は，機械設計向けの知識コンパイラ MECHANICOT の出力として KL1 プログラムを得るための変換規則について述べる。変換規則が対象とする入力データは，知識コンパイラでの解析で得られた，制約式間の依存関係を表すデータフローである。設計においては複数の解を比較することが必要なため，OR 並列的に探索出来るデータ構造を定め，このデータ構造を用いて全解探索を行うプログラムを出力するための変換規則を定めた。

1 はじめに

制約知識に着目した知識コンパイラ [溝口他 88] を制約指向知識コンパイラと呼んでいる。制約指向知識コンパイラは，宣言的な知識表現である「制約」を手続き的な知識表現に変換し，問題解決機構が効率良く実行可能な設計手順を自動的に生成することを目的としている [永井他 89a]。

我々は，このような制約指向知識コンパイラを機械設計問題に適用した MECHANCIOT [寺崎他 88] を開発し，制約をはじめとしたさまざまな知識を有効に活用したシステムの構築支援ならびに問題解決の効率化について取り組んできた。

MECHANICOT における知識コンパイラは，入力として宣言的に記述された設計知識（仕様や設計公式など）を受け取り，図 1.1 に示す処理フローに従って制約を解析して設計手順を生成し，設計手順に従ったプログラムを出力する [永井他 89b]。

しかしながら，MECHANICOT における知識コンパイラの出力は，ESP で記述された逐次処理を行うプログラムであった。

そこで，知識コンパイラの出力の並列化を目的として，各制約間の変数の依存関係に基づくデータフロー表現から KL1 プログラムを生成する変換規則の設計を行った。これは，図 1.1 において太字で示した「制約解析部 (Phase 2)」および「プログラム生成部」の改良に相当する。

ESP プログラムを出力とする知識コンパイラ（以後 ESP 版と呼ぶ）では，ESP の継承，スロット（グローバルデータ），およびバックトラックの機能を積極的に活用していた。一方，出力を KL1 プログラムにするとこれらの機能が全て使えない。設計においては複数の解を比較出来る環境が必要であるので，グローバルデータとバックトラック機能を用いずに OR 並列的な探索が行えるようなデータ構造を設計し，このデータ構造に基づいて全解探索を行うプログラムを出力する変換規則を定めた。

以下，2 章では，機械設計の例題を用いて知識コンパイラの概要と，本報告で扱うデータフローについて述べる。3 章では，全解探索を行うための構造体データを示す。4 章および 5 章で，データフローから KL1 プログラムを生成するための変換規則について述べる。なお，付録として変換結果である KL1 プログラムの例を示す。

2 対象とするデータフロー

具体例として「2 段変速ギアユニット」を用いて，対象とするデータフローについて述べる。この例題は，工作機械の主軸設計問題 [井上他 88] の部分問題となっている。「2 段変速ギアユニット」は，図 2.1 に示すように，2 本の軸 (in_shaft, out_shaft) と 2 組のギアユニット (highspeed_gear, lowspeed_gear) から成り立っている。

これは仕様として，

- 入力の最高回転数，最大トルク，振りモーメント
- 出力の最高回転数，最大トルク，振りモーメント，等

を与え，出力として，

- 入力軸の軸径，出力軸の軸径
- 各ギアのピッチ円径，歯数，およびギア比，等

を求める問題である。

これを知識コンパイラへ入力する時の知識表現を付録 1 に示す。付録 1 において，class_name, inherit_from,

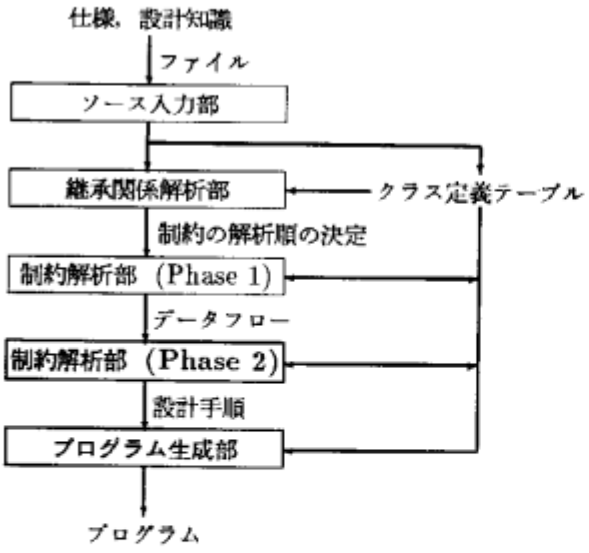

図 1.1　知識コンパイラの処理フロー

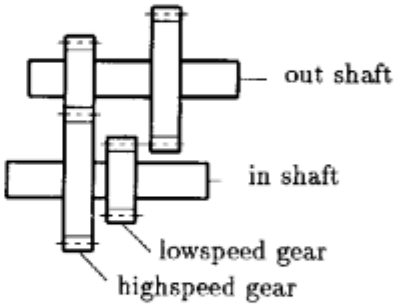

図 2.1　2 段変速ギアユニット

constraint, design_method, generator, tester, def_method は予約語である。これらの予約語の意味，および文法については，[寺崎 88] を参照のこと。知識コンパイラはこれらの知識が記述されたファイルを受け取ると，generator,tester,design_method に記述された制約をそれぞれユニークなサブゴールと考える。そして，各々のサブゴールにユニークな名前を与え，クラス単位で，これらのサブゴール間の依存関係を解析してデータフローを生成する。このデータフローの生成は，図 1.1 における「制約解析部 (Phase 1)」でなされる。なお，データフローの生成アルゴリズムについては，[永井他 89b, Nagai 89, 永井他 90] を参照のこと。

図 2.2 に，付録 1 を入力とした場合のデータフローを示す。図でノードは各制約で求めるべき出力変数（の名前）を表し，エッジは制約間の変数依存関係を示す。また，変数名の前にある Gn,Tn,Cn はそれぞれ generator, tester, および design_method の各制約の役割を表している。

3　基本データ構造

設計においては，複数の解を比較出来ることが必要となる。ESP 版では複数の解を求めるのに，バックトラック機能を利用していた。一方，KL1 は AND 並列の環境であり，バックトラック機能がないため（AND 並列では意味を持たないため），OR 並列と等価な探索が出来るデータ構造を用意する必要がある。解の組が何個生成されるかは，設計が終了するまで分からず可変長のデータとなるが，ある 1 組の解についての属性の数（求めるべきパラメータの数）は，制約解析時に分かり固定長のデータとして扱える。そこで下に示すようなデータ構造を用いることとする。リストの要素であるベクタがある 1 組の解（属性値の組）を表している。このデータ構造により OR 並列的な探索が可能となり，かつ，常にある 1 組の解に着目してベクタ構造 1 つを生成検査ループの入力とすることにより階層的な生成検査を行うことが容易に出来る。

```
[ [{Attr111, ..., Attr11N}, ..........., {Attr1M1, ..., Attr1MN} ],
  ......................,
  [{Attr111, ..., Attr11N}, ..........., {AttrLM1, ..., AttrLMN} ],
  ......................,
  [{AttrL11, ..., AttrL1N}, ..........., {AttrLM1, ..., AttrLMN} ] ]

      AttrMN (M=1 ～ M, N=1 ～ N) : 属性値
      リストのネストが generate & test の階層（この例では 2 階層）を表す
```

4 各予約語の変換規則

4.1 class_name

入力知識表現における class_name 定義は，KL1 におけるモジュールに対応させる。

4.2 inherit_from

入力知識表現で，inherit_from 定義により継承すべきクラスが指示されていた場合には，継承先のクラス定義を全て継承元のクラスに吸収することとする。例えば，付録 1 に示すように 2 段変速ギアユニットの定義において，クラス out_shaft が line_shaft を継承していたとする。この場合のコンパイル結果は，モジュール out_shaft に入力知識表現でクラス line_shaft に記述されている全ての parameter, constraint, design_method 等の定義を含めた KL1 プログラムを出力するものとする。（付録 2 参照）

4.3 constraint

ESP 版では，constraint にかかれた制約（値の伝播）についてもそれぞれ 1 つのサブゴールを割り当てていた。しかし，KL1 版では全解探索が可能なように出力変数は全て構造体データとした。従って，値の伝播に関して陽にサブゴールとせずに，知識コンパイラの解析結果を用いてベクタ要素へのアクセスとして実現する必要がある。

4.4 design_method

design_method 定義において述語定義が他のクラスに存在する場合には，Kl1 におけるモジュール間呼び出しとする。

4.5 generator

基本的には，design_method の場合と同じように KL1 におけるモジュール呼び出しとする。但し，generator は複数の解候補を生成するため，出力パラメータは構造体データであると解釈する。変換規則の詳細については次節で述べる。

4.6 tester

tester は，generator で生成された解候補を入力として，tester に定義された関係を満たす解候補を出力する。従って，入出力データが共に構造体データとなる。入力知識表現には構造体データであることが陽には記述されていないので，図 4.6 に示すような tester のテンプレートを用意する。このテンプレートをもとに，入力知識表現に対応する KL1 プログラムを出力する。モジュール名は，入力知識表現におけるメソッド名とする。変換規則の詳細については次節で述べる。なお，テスタは 2 項関係に関する制約を処理する。3 項以上の関係については 2 項関係に分解して処理する。

4.7 def_method

def_method 宣言以降に現れたメソッド定義については，そのまま KL1 プログラムであると解釈する。

5 データフローから KL1 プログラムへの変換

5.1 変換規則

データフローを基に，いくつかのサブゴールをまとめてゴールを生成する。ここで，注意しなければならないのは，design_method に定義された制約はアトミックなデータを入出力とするのに対して，generator は常に構造体データを出力すること，および，tester は構造体データを入力として構造体データを出力することである。さらに，generator や tester が階層的な呼び出し関係にある時には，この階層関係を保つように呼び出し形式および構造体データの扱いを決める必要がある。

そこで，データフローに基づいてサブゴールからゴールにまとめる時に，制約のタイプ（generator,tester）を利用する。変換規則は次の4ステップからなる。

Step 1: サブゴールがtester（入出力データが共に構造体）の場合，tester単独でゴールとする。

Step 2: サブゴールがgenerator（出力データが構造体）の場合，generatorとその入力パラメータを求めているdesign_methodをまとめて1つのゴールとする。

Step 3: サブゴールがdesign_method（入出力データが共にアトミック）の場合，次の入力パラメータの依存関係に従い分類して，それぞれの集合をゴールにまとめる。

- testerの出力データを含む
- generatorの出力データを含む
- 他クラスのゴールの出力データを含む
- 仕様のみ

Step 4: 階層的なgenerate & testが可能なように，入出力のデータ構造を揃える。

これらの変換規則の基本的な考え方は，『ゴールは常に「ある1組」の入力データに対する解候補の組を返す』というものである。

また，以上のようなゴールを順次呼びだしていくに従い構造体データは属性値の数（即ちベクタの要素数）が増えていく。この時，構造体データのベクタの要素の並べ方は，『先行するゴール群で求められた属性値の並びに対して，新たに求めた属性値の並びを後側からアペンドする』ものとする。

図5.1に図2.2のデータフローとこれらの変換規則を対応付けて示す。

5.2 Step 1: サブゴールがtesterの場合

サブゴールがtesterの場合には，generatorで生成された解候補に対して，test & mergeを行う。従って，testerは入出力変数が共に構造体データとなる。そこでtesterの場合には単独でゴールとして扱う。

また，testerでの制約はgeneratorで生成された変数のみを用いているとは限らない。そこで，generatorで生成された変数以外の入力データを必要とする場合には，これらの入力データをベクタの形式で渡すこととする。呼び出し形式を以下に示す。変換例は，付録2を参照のこと。

```
goal(Wm1,Wm2,Vpara, ~Wm3),  または,  goal (Wm1,Wm2, ~Wm3)
      Wm1 : [{In111,In112,..,In11K},...,{In1J1,In1J2,..,In1JK}]
      Wm2 : [{In211,In212,..,In21M},...,{In2L1,In2L2,..,In2LM}]
      Vpara: {Para1,...,ParaN} (ベクタ)
      Wm3 : [{In111,In112,..,In11K,In211,In212,..,In21M},
               .....,
             {In111,In112,..,In11K,In2L1,In2L2,..,In2LM},
               .....,
             {In1J1,In1J2,..,In1JK,In211,In212,..,In21M},
               .....,
             {In1J1,In1J2,..,In1JK,In2L1,In2L2,..,In2LM}]
```

5.3 Step 2: サブゴールがgeneratorの場合

サブゴールがgeneratorの場合には，入力データはアトミックなデータであるが出力データは構造体となる。そこで，以下のような呼び出し形式のゴールとする。変換例は，付録2を参照のこと。

```
goal(In1,In2,..,InJ, ~Wm)
      InJ (J=1～J) : 入力パラメータ (アトミックデータ)
      Wm : [{Out11,Out12,..,Out1K},.....,{OutJ1,OutJ2,..,OutJK}]
```

5.4 Step 3: サブゴールがdesign_methodの場合

入出力データは共にアトミックなデータであるが，generatorでの呼び出し形式とあわせるために，ゴールの呼び出し形式を以下のように定め，出力データを構造体とする。

```
goal(In1,In2,..,InJ, ^Wm)
      InJ (J=1 ～ J) : 入力パラメータ (アトミックデータ)
      Wm : [{Out1,Out2,..,OutN}]
```

なお，出力データを構造体とするために，述語create_wm/3を用意する。create_wm/3は，第1引数と第2引数にリストおよびその要素の個数を与えると，与えられたリストをベクタに変換して，第3引数に構造体データを返すものである。変換例は，付録2を参照のこと。

5.5 Step 4 : ゴール間のデータ構造を揃える

Step 1 ～ Step 3 で生成されたゴールに続いて，design_method からなるゴール (Step 3 で生成されたゴール) を呼び出したり，generator のゴールから generator を続いて呼び出す時には，入出力データの形式があわない。そこで，『ゴールは常に「ある1組」の入力データに対する解候補の組を返す』という原則に従い，階層的な generate & test を行うためのゴール呼び出しのインタフェースメソッドとして以下の呼出し形式を用意する。変換例は，付録2を参照のこと。

```
goal_if (Wm1, ^Wm2)
goal_if (Wm1, In1,In2,..,InJ, ^Wm2)
    In1 ～ InJ : 入力データ (アトミックデータ)
    Wm1 : [{In11,In12,..,In1K},..., {InJ1,InJ2,..,InJK}]
     Wm3 : [
              [{In11,..,In1K,Out11,..,Out1N},
               {In11,..,In1L,Out21,..,Out2N},
                 ..................,
               {In11,..,In1L,OutM1,..,OutMN}],
                 ..................,
              [{InJ1,..,InJK,Out11,..,Out1N},
                 ..................,
               {InJ1,..,InJK,OutM1,..,OutMN}]
           ]
```

参考文献

[井上他 88] 井上 克巳，永井 保夫，藤井 裕一，今村 聡，小島 俊雄，“工作機械の設計手法の解析－旋盤の回転機能部品の設計”, ICOT Technical Memorandum, TM-494, ICOT, (1988)

[永井他 89a] 永井 保夫，滝 寛和，寺崎 智，横山 孝典，井上 克巳，“設計問題向けツール・アーキテクチャ”，人工知能学会，Vol.4, No.3, (1989) pp.297-303.

[永井他 89b] 永井 保夫，寺崎 智，“設計向け制約指向知識コンパイラにおける制約解析および手順生成について”，第3回人工知能学会全国大会，(1989) pp.693-696.

[永井他 90] 永井 保夫，生駒 憲治，“グラフ理論による制約解析および手順生成”，第40回情報処理全国大会，(1990).

[寺崎他 88] 寺崎 智，永井 保夫，横山 孝典，井上 克巳，堀内 英一，滝 寛和，“機械設計支援システム構築ツール－ MECHANICOT–”，人工知能学会，知識ベースシステム研究会資料，SIG-KBS-8803, (1988).

[溝口他 88] 溝口 理一郎，山口 高平，角所 収，“エキスパートシステム構築方法論について”，人工知能学会研究会資料，SIG-KBS-8801-2, (1988).

[Nagai 89] Y. Nagai and K. Ikoma, “Design Plan Generation Through Constraint Compilation”, Italian-Japanese-Swedish Workshop on Concurrent Logic Programming and Constraint Logic Programming, ICOT Technical Memorandum, ICOT (1989).

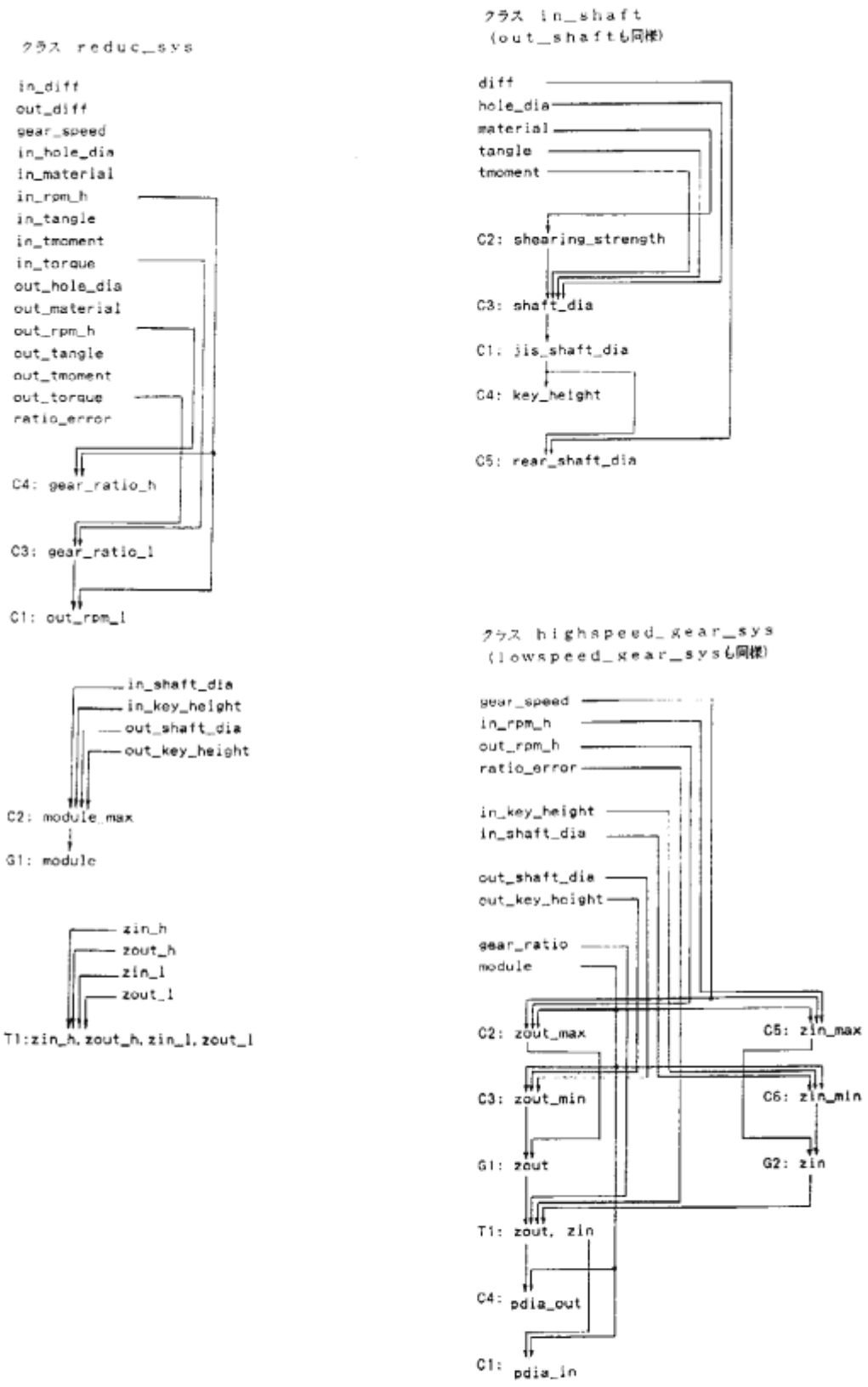
クラス reduc_sys
in_diff
out_diff
gear_speed
in_hole_dia
in_material
in_rpm_h
in_tangle
in_tmoment
in_torque
out_hole_dia
out_material
out_rpm_h
out_tangle
out_tmoment
out_torque
ratio_error
C4: gear_ratio_h
C3: gear_ratio_l
C1: out_rpm_l
クラス in_shaft
(out_shaftも同様)
diff
hole_dia
material
tangle
tmoment
C2: shearing_strength
C3: shaft_dia
C1: jis_shaft_dia
C4: key_height
C5: rear_shaft_dia
in_shaft_dia
in_key_height
out_shaft_dia
out_key_height
C2: module_max
G1: module
zin_h
zout_h
zin_l
zout_l
T1:zin_h, zout_h, zin_l, zout_l
クラス highspeed_gear_sys
(lowspeed_gear_sysも同様)
gear_speed
in_rpm_h
out_rpm_h
ratio_error
in_key_height
in_shaft_dia
out_shaft_dia
out_key_height
gear_ratio
module
C2: zout_max
C5: zin_max
C3: zout_min
C6: zin_min
G1: zout
G2: zin
T1: zout, zin
C4: pdia_out
C1: pdia_in

図2.2　データフロー

```
tester([V1|Tail1],Wm2,Wm3) :- true |
        test_sub1(V11,Wm2,Wm2New,Wm3,Wm3Tail),
        tester(Tail1,Wm2New,Wm3Tail).
tester([],_,Wm3) :- true |  Wm3 = [].

test_sub1(V1,[V2|Tail],Wm2New,Wm3,Wm3Tail) :- true |
        test_and_merge(V1,V21,V11,V22,Status),
        check_result(Status,V11,V22,V12,V23,Wm3,Wm3New),
        Wm2New = [V23|Tail2New],
        test_sub1(V12,Tail2,Tail2New,Wm3New,Wm3Tail).
test_sub1(_,[],Wm2New,Wm3,Wm3Tail) :- true | Wm2New   = [], Wm3Tail  = Wm3.

check_result(Status,V1,V2,V1New,V2New,Wm3,Wm3New) :- Status = ok |
        wm_util:expand_element(V1,V2,V1New,V2New,V3),
        wm_util:add_element(V3,Wm3,Wm3New).
otherwise.
check_result(Status,V1,V2,V1New,V2New,Wm3,Wm3New) :- true |
        V1New = V1, V2New = V2, Wm3New = Wm3.
```

(a) パラメータなしの場合

```
tester([V1|Tail1],Wm2,Vpara,Wm3) :- true |
        test_sub1(V1,Wm2,Vpara,Wm2New,VparaNew,Wm3,Wm3Tail),
        tester(Tail1,Wm2New,VparaNew,Wm3Tail).
tester([],_,_,Wm3) :- true |  Wm3 = [].

test_sub1(V1,[V2|Tail],Vpara,Wm2New,VparaNew,Wm3,Wm3Tail) :- true |
        test_and_merge(V1,V2,Vpara,V11,V21,Vpara1,Status),
        check_result(Status,V1,V2,V12,V22,Wm3,Wm3New),
        Wm2New = [V22|Tail2New],
        test_sub1(V12,Tail2,Vpara1,Tail2New,VparaNew,Wm3New,Wm3Tail).
test_sub1(_,[],Vpara,Wm2New,VparaNew,Wm3,Wm3Tail) :- true |
        VparaNew = Vpara, Wm2New   = [], Wm3Tail  = Wm3.

check_result(Status,V1,V2,V1New,V2New,Wm3,Wm3New) :- Status = ok |
        wm_util:expand_element(V1,V2,V1New,V2New,V3),
        wm_util:add_element(V3,Wm3,Wm3New).
otherwise.
check_result(Status,V1,V2,V1New,V2New,Wm3,Wm3New) :- true |
        V1New = V1,  V2New = V2,  Wm3New = Wm3.
```

(b) パラメータ付きの場合

図4.6 testerのテンプレート

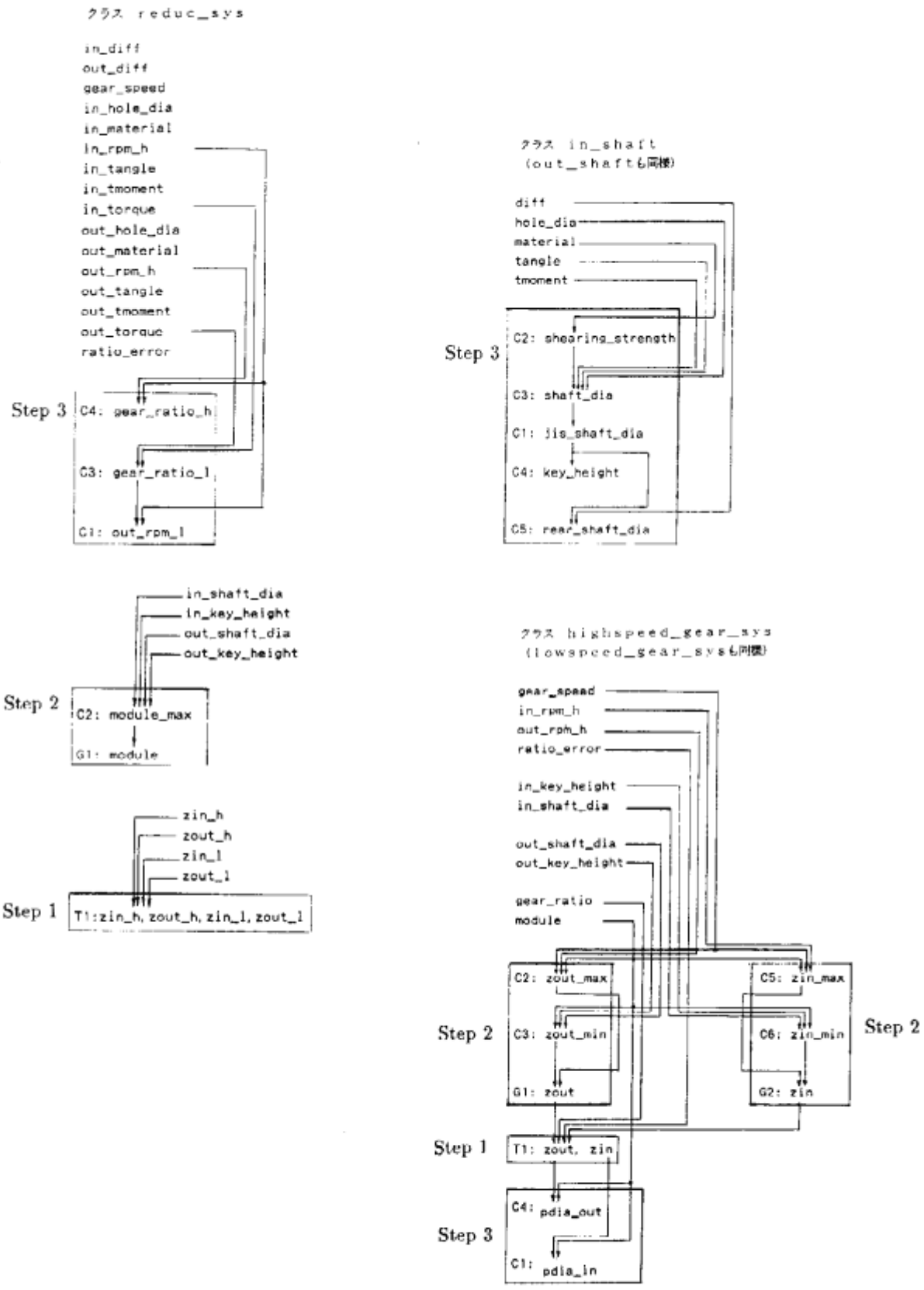
クラス reduc_sys
in_diff
out_diff
gear_speed
in_hole_dia
in_material
in_rpm_h
in_tangle
in_tmoment
in_torque
out_hole_dia
out_material
out_rpm_h
out_tangle
out_tmoment
out_torque
ratio_error
Step 3
C4: gear_ratio_h
C3: gear_ratio_l
C1: out_rpm_l
クラス in_shaft
(out_shaftも同様)
diff
hole_dia
material
tangle
tmoment
Step 3
C2: shearing_strength
C3: shaft_dia
C1: jis_shaft_dia
C4: key_height
C5: rear_shaft_dia
in_shaft_dia
in_key_height
out_shaft_dia
out_key_height
Step 2
C2: module_max
G1: module
zin_h
zout_h
zin_l
zout_l
Step 1
T1:zin_h, zout_h, zin_l, zout_l
クラス highspeed_gear_sys
(lowspeed_gear_sysも同様)
gear_speed
in_rpm_h
out_rpm_h
ratio_error
in_key_height
in_shaft_dia
out_shaft_dia
out_key_height
gear_ratio
module
Step 2
C2: zout_max
C3: zout_min
G1: zout
C5: zin_max
C6: zin_min
G2: zin
Step 2
Step 1
T1: zout, zin
Step 3
C4: pdia_out
C1: pdia_in

図 5.1　変換規則の適用

付録 1 : 2段変速ギアユニットの設計知識記述

```
%%%%%%%%
%%%%%%%%  spec_input : 設計仕様定義
%%%%%%%%
class_name
    spec_input;

design_object   % 設計対象宣言
    my_reduc_sys;

parameter
    gear_speed      := 2000.0,  % ギア許容最大周速 [m/min]
    in_diff         := 0.0,     % 入力軸径の前側と後側の差 [mm]
    in_hole_dia     := 0.0,     % 入力軸穴径 [mm]
    in_material     := carbide, % 入力軸材質
    in_rpm_h,                   % 入力軸最高回転数 [rpm]
    in_tangle       := 0.25,    % 入力軸許容捩じれ角 [degree/m]
    in_tmoment,                 % 入力捩じりモーメント [kg・mm]
    in_torque,                  % 入力トルク [kg・m]
    out_diff        := 10.0,    % 出力軸径の前側と後側の差 [mm]
    out_hole_dia,               % 出力軸穴径 [mm]
    out_material    := carbide, % 出力軸材質
    out_rpm_h,                  % 出力軸最高回転数 [rpm]
    out_tangle      := 0.025,   % 出力軸許容捩じれ角 [degree/m]
    out_tmoment,                % 出力捩じりモーメント [kg・mm]
    out_torque,                 % 出力トルク [kg・m]
    ratio_error     := 0.01;    % ギア比許容誤差

constraint
    #my_reduc_sys!gear_speed     := gear_speed,
    #my_reduc_sys!in_diff        := in_diff,
    #my_reduc_sys!in_hole_dia    := in_hole_dia,
    #my_reduc_sys!in_material    := in_material,
    #my_reduc_sys!in_rpm_h       := in_rpm_h,
    #my_reduc_sys!in_tangle      := in_tangle,
    #my_reduc_sys!in_tmoment     := in_tmoment,
    #my_reduc_sys!in_torque      := in_torque,
    #my_reduc_sys!out_diff       := out_diff,
    #my_reduc_sys!out_hole_dia   := out_hole_dia,
    #my_reduc_sys!out_material   := out_material,
    #my_reduc_sys!out_rpm_h      := out_rpm_h,
    #my_reduc_sys!out_tangle     := out_tangle,
    #my_reduc_sys!out_tmoment    := out_tmoment,
    #my_reduc_sys!out_torque     := out_torque,
    #my_reduc_sys!ratio_error    := ratio_error;

end.
```

```
%%%%%%%%%%
%%%%%%%%%%  my_reduc_sys : 2段変速ギアユニット全体の定義
%%%%%%%%%%
class_name
    my_reduc_sys;

inherit_from   % 継承宣言
    reduction_sys0;

consist_of     % 構成部品の定義
    highspeed_gear_sys is standard_spur_gear_sys,
    lowspeed_gear_sys  is standard_spur_gear_sys,
    out_shaft          is line_shaft,
    in_shaft           is line_shaft;

end.


%%%%%%%%%%
%%%%%%%%%%  reduction_sys0
%%%%%%%%%%
class_name
    reduction_sys0;

consist_of
    highspeed_gear_sys,
    lowspeed_gear_sys,
    in_shaft,
    ou_shaft;

parameter
    gear_ratio_h     := void,    % 高速側ギア比
    gear_ratio_l     := void,    % 低速側ギア比
    gear_speed       := 2000.0,  % ギア許容最大周速 [m/min]
    in_diff          := 0.0,     % 入力軸径の前側と後側の差 [mm]
    in_hole_dia      := void,    % 入力軸材質
    in_key_height    := void,    % 入力軸のキー高さ [mm]
    in_material      := carbide, % 入力軸穴径 [mm]
    in_rpm_h         := void,    % 入力軸最高回転数 [rpm]
    in_shaft_dia     := void,    % 入力軸径 (規格化) [mm]
    in_tangle        := 0.25,    % 入力軸許容捩じれ角 [degree/m]
    in_tmoment       := void,    % 入力捩じりモーメント [kg・mm]
    in_torque        := void,    % 入力トルク [kg・m]
    module           := void,    % ギアモジュール [mm]
    module_max       := void,    % ギアモジュールの最大値 [mm]
    out_diff         := 10.0,    % 出力軸径の前側と後側の差 [mm]
    out_hole_dia     := void,    % 出力軸穴径 [mm]
```

```
    out_key_height  := void,     % 出力軸のキー高さ [mm]
    out_material    := carbide,  % 出力軸材質
    out_rpm_h       := void,     % 出力軸高速側最高回転数 [rpm]
    out_rpm_l       := void,     % 出力軸低速側最高回転数 [rpm]
    out_shaft_dia   := void,     % 出力軸径（規格化）[mm]
    out_tangle      := 0.025,    % 出力軸許容捩じれ角 [degree/m]
    out_tmoment     := void,     % 出力捩じりモーメント [kg・mm]
    out_torque      := void,     % 出力トルク [kg・m]
    ratio_error     := void,     % ギア比許容誤差
    zin_h           := void,     % 入力ギアの歯数（高速側）
    zin_l           := void,     % 入力ギアの歯数（低速側）
    zout_h          := void,     % 出力ギアの歯数（高速側）
    zout_l          := void;     % 出力ギアの歯数（低速側）

constraint
    in_key_height   := #input_shaft!key_height,
    in_shaft_dia    := #input_shaft!jis_shaft_dia,
    out_key_height  := #output_shaft!key_height,
    out_shaft_dia   := #output_shaft!jis_shaft_dia,
    zin_h   := #highspeed_gear_sys!zin,
    zin_l   := #lowspeed_gear_sys!zin,
    zout_l  := #lowspeed_gear_sys!zout,
    zout_h  := #highspeed_gear_sys!zout,
    #input_shaft!diff       := in_diff,
    #input_shaft!hole_dia   := in_hole_dia,
    #input_shaft!material   := in_material,
    #input_shaft!tangle     := in_tangle,
    #input_shaft!tmoment    := in_tmoment,
    #highspeed_gear_sys!gear_ratio  := gear_ratio_h,
    #highspeed_gear_sys!gear_speed  := gear_speed,
    #highspeed_gear_sys!in_key_height   := in_key_height,
    #highspeed_gear_sys!in_rpm_h    := in_rpm_h,
    #highspeed_gear_sys!in_shaft_dia:= #input_shaft!jis_shaft_dia,
    #highspeed_gear_sys!module      := module,
    #highspeed_gear_sys!out_key_height  := out_key_height,
    #highspeed_gear_sys!out_rpm_h   := out_rpm_h,
    #highspeed_gear_sys!out_shaft_dia   := #output_shaft!jis_shaft_dia,
    #highspeed_gear_sys!ratio_error := ratio_error,
    #lowspeed_gear_sys!gear_ratio   := gear_ratio_l,
    #lowspeed_gear_sys!gear_speed   := gear_speed,
    #lowspeed_gear_sys!in_key_height:= in_key_height,
    #lowspeed_gear_sys!in_rpm_h     := in_rpm_h,
    #lowspeed_gear_sys!in_shaft_dia := #input_shaft!jis_shaft_dia,
    #lowspeed_gear_sys!module       := module,
    #lowspeed_gear_sys!out_key_height   := out_key_height,
    #lowspeed_gear_sys!out_rpm_h    := out_rpm_l,
    #lowspeed_gear_sys!out_shaft_dia:= #output_shaft!jis_shaft_dia,
    #lowspeed_gear_sys!ratio_error  := ratio_error,
```

```
    #output_shaft!diff      := out_diff,
    #output_shaft!hole_dia  := out_hole_dia,
    #output_shaft!material  := out_material,
    #output_shaft!tangle    := out_tangle,
    #output_shaft!tmoment   := out_tmoment;

generator
    {[module], get_module(#self, module_max,module) };

tester
    { shaft_distance_tester(#self, zin_h,zout_h,zin_l,zout_l) };

design_method
    {[out_rpm_l], output_rpm_low(#self,in_rpm_h,gear_ratio_l,out_rpm_l) },
    {[module_max], module_max(#gearsDM, gear_speed, in_rpm_h, out_rpm_h,
       in_shaft_dia, out_shaft_dia,in_key_height,out_key_height, module_max) },
    {[gear_ratio_h], rpm_ratio(#self, in_rpm_h, out_rpm_h, gear_ratio_h) },
    {[gear_ratio_l], torque_ratio(#self, in_torque, out_torque, gear_ratio_l) };

def_method
get_module(ModuleMax,Wm) :- floating_point(ModuleMax) |
        gear_module:start_server(module,STR),
        STR = [serach(all,2.0,ModuleMax,Num,Module)],
        wm_util:merge_vector_and_catalog(0,{},Num,1,Module,Wm).

shaft_dsitance_tester(Zin1,Zout1,Zin2,Zout2,Status) :-
    X $:= Zin1 + Zout1, Y $:= Zin2 + Zout2, X = Y |  Status = ok.
otherwise.
shaft_distance_tester(_,_,_,_,Status) :- true |  Status = no.

output_rpm_low(In_rpm_h, Gear_ratio_l, Out_rpm_l) :-
    floating_point(In_rpm_h), floating_point(Gear_ratio_l) |
        Out_rpm_l $:= In_rpm_h / Gear_ratio_l.

rpm_ratio(Rpm_input_shaft, Rpm_output_shaft, Gear_ratio) :-
    floating_point(Rpm_input_shaft), floating_point(Rpm_output_shaft) |
        Gear_ratio $:=  Rpm_input_shaft / Rpm_output_shaft.

torque_ratio(Input_torque, Output_torque, Gear_ratio) :-
    floating_point(Input_torque), floating_point(Output_torque) |
        Gear_ratio $:= Output_torque / (Input_torque * 0.75).

end.


%%%%%%%%%%
%%%%%%%%%%  standard_spur_gear_sys
%%%%%%%%%%
```

```
class_name
    standard_spur_gear_sys;

parameter
    gear_ratio      := void,      % ギア比
    gear_speed      := 2000.0,    % ギアの許容最大周速 [m/min]
    in_key_height   := void,      % 入力軸のキー高さ [mm]
    in_rpm_h        := void,      % 入力軸最高回転数 [rpm]
    in_shaft_dia    := void,      % 入力軸径 [mm]
    module          := void,      % ギアモジュール [mm]
    out_key_height := void,       % 出力軸のキー高さ [mm]
    out_rpm_h       := void,      % 出力軸最高回転数 [rpm]
    out_shaft_dia   := void,      % 出力軸径 [mm]
    pdia_in         := void,      % 入力側ギアピッチ円径 [mm]
    pdia_out        := void,      % 出力側ギアピッチ円径 [mm]
    ratio_error     := 0.01,      % ギア比の許容誤差率
    zin             := void,      % 入力側ギア歯数
    zin_min         := void,      % 入力側ギア歯数の最小値
    zin_max         := void,      % 入力側ギア歯数の最大値
    zout            := void,      % 出力側ギア歯数
    zout_min        := void,      % 出力側ギア歯数の最小値
    zout_max        := void;      % 出力側ギア歯数の最大値

generator
    {[zin], generator(#generator, zin_min, zin_max, 1.0, zin) },
    {[zout], generator(#generator, zout_min, zout_max, 1.0, zout) };

tester
    { gear_ratio_tester(#self, zin, zout, gear_ratio,ratio_error) };

design_method
    {[pdia_out], get_pitch_dia(#gearsDM, zout, module, pdia_out) },
    {[zout_max], get_tnum_max(#gearsDM, gear_speed, out_rpm_h, module, zout_max) },
    {[zout_min], get_tnum_min(#gearsDM, out_shaft_dia,out_key_height,module,zout_min) },
    {[pdia_in],  get_pitch_dia(#gearsDM, zin, module, pdia_in) },
    {[zin_max],  get_tnum_max(#gearsDM, gear_speed, in_rpm_h, module, zin_max)  },
    {[zin_min], get_tnum_min(#gearsDM, in_shaft_dia, in_key_height, module, zin_min) };

def_method
gear_ratio_tester(Zin,Zout,Ratio,RatioError,Status) :-
       R $:= Zout / Zin,  Error $:= Ratio*RatioError,
       Rmin $:= Ratio - Error,  Rmax $:= Ratio + Error,
       Rmin $=< R,R $=< Rmax |  Status = ok.
otherwise.
gear_ratio_tester(_,_,_,_,Status) :- true | Status = no.

end.
```

```
%%%%%%%%%%
%%%%%%%%%%  line_shaft
%%%%%%%%%%
class_name
    line_shaft;

parameter
    diff              := 10.0,  % 軸径の前側と後側の差 [mm]
    hole_dia          := void,  % 軸穴径 [mm]
    jis_shaft_dia     := void,  % 前側の軸径（規格化）[mm]
    key_height        := void,  % 軸のキー高さ [mm]
    material          := carbide, % 軸材質
    rear_shaft_dia    := void,  % 後側の軸径 [mm]
    shaft_dia         := void,  % 前側の軸径（計算値）[mm]
    shearing_strength := void,  % 旋断弾性係数 [kg/mm2]
    tangle            := 0.025, % 軸の許容振じれ角 [deg/m]
    tmoment           := void;  % 振じれモーメント [kg・mm]


design_method
    {[jis_shaft_dia], jis_shaft_dia(#jis_shaft_dia, shaft_dia, jis_shaft_dia) },
    {[shearing_strength], shearing_strength(#material_kb, material, shearing_strength) },
    {[shaft_dia], calc_shaft_dia(#shaftDM, tmoment, tangle, shearing_strength,
          hole_dia,shaft_dia) },
    {[key_height], key_height(#key_height, jis_shaft_dia, key_height) },
    {[rear_shaft_dia],  rear_shaft_dia(#self, jis_shaft_dia, diff, rear_shaft_dia) };

def_method
rear_shaft_dia(Shaft_dia, Diff, Rear_shaft_dia) :-
    floating_point(Diff),
    floating_point(Shaft_dia)| Rear_shaft_dia = Shaft_dia - Diff.

end.
```

付録２：出力プログラム例

```
%%%
%%%  トップレベルモジュール：仕様入力
%%%
:- module reduc_sys_design_system.
:- public goal/2.

% Spec:{GearSpeed,InDiff,    InHoleDia,  InMaterial,
%       InRpmH,   InTangle,  InTmoment,  InTorque,
%       OutDiff,  OutHoleDia,OutMaterial,OutRpmH,
%       OutTangle,OutTmoment,OutTorque,  RatioError}
%
% WmOut :[[
%           [{OutRpmL,RatioH,RatioL,
%             InKeyHeight, InShaftDia, InRearShaftDia,
%             OutKeyHeight,OutShaftDia,OutRearShaftDia,Module1,
%             ZinH1,ZoutH1,PdinH1,PdoutH1,ZinL1,ZoutL1,PdinL1,PdoutL1},
%             ..............................,
%            {OutRpmL,RatioH,RatioL,
%             InKeyHeight, InShaftDia, InRearShaftDia,
%             OutKeyHeight,OutShaftDia,OutRearShaftDia,Module1,
%             ZinHN,ZoutHN,PdinHN,PdoutHN,ZinLN,ZoutLN,PdinLN,PdoutLN}
%           ],
%             ..............................,
%           [{OutRpmL,RatioH,RatioL,
%             InKeyHeight, InShaftDia, InRearShaftDia,
%             OutKeyHeight,OutShaftDia,OutRearShaftDia,ModuleK,
%             ZinH1,ZoutH1,PdinH1,PdoutH1,ZinL1,ZoutL1,PdinL1,PdoutL1},
%             ..............................,
%            {OutRpmL,RatioH,RatioL,
%             InKeyHeight, InShaftDia, InRearShaftDia,
%             OutKeyHeight,OutShaftDia,OutRearShaftDia,ModuleK,
%             ZinHN,ZoutHN,PdinHN,PdoutHN,ZinLN,ZoutLN,PdinLN,PdoutLN}
%           ]
%         ]]

goal(Spec,WmOut) :- true |
% Spec:{GearSpeed,InDiff,    InHoleDia,  InMaterial,
%       InRpmH,   InTangle,  InTmoment,  InTorque,
%       OutDiff,  OutHoleDia,OutMaterial,OutRpmH,
%       OutTangle,OutTmoment,OutTorque,  RatioError}
       set_vector_element(Spec  , 0,GearSpeed,  GearSpeed,  Spec1),
       set_vector_element(Spec1 , 1,InDiff,     InDiff,     Spec2),
       set_vector_element(Spec2 , 2,InHoleDia,  InHoleDia,  Spec3),
       set_vector_element(Spec3 , 3,InMaterial, InMaterial, Spec4),
       set_vector_element(Spec4 , 4,InRpmH,     InRpmH,     Spec5),
       set_vector_element(Spec5 , 5,InTangle,   InTangle,   Spec6),
       set_vector_element(Spec6 , 6,InTmoment,  InTmoment,  Spec7),
```

```
        set_vector_element(Spec7 , 7,InTorque,   InTorque,   Spec8),
        set_vector_element(Spec8 , 8,OutDiff,    OutDiff,    Spec9),
        set_vector_element(Spec9 , 9,OutHoleDia, OutHoleDia, Spec10),
        set_vector_element(Spec10,10,OutMaterial,OutMaterial,Spec11),
        set_vector_element(Spec11,11,OutRpmH,    OutRpmH,    Spec12),
        set_vector_element(Spec12,12,OutTangle,  OutTangle,  Spec13),
        set_vector_element(Spec13,13,OutTmoment, OutTmoment, Spec14),
        set_vector_element(Spec14,14,OutTorque,  OutTorque,  Spec15),
        set_vector_element(Spec15,15,RatioError, RatioError, _),

        % Wm1 : [{OutRpmL,RatioH,RatioL}, ....]
        reduc_sys:goal2(InRpmH,OutRpmH,InTorque,OutTorque,Wm1),
        % Wm2 :[{InDiff,InKeyHeight,InShaftDia,InRearShaftDia}, ...]
        reduc_sys:goal4(InDiff,InHoleDia,InMaterial,InTangle,InTmoment,Wm2),
        % Wm3 :[{OutDiff,OutKeyHeight,OutShaftDia,OutRearShaftDia}, ...]
        reduc_sys:goal5(OutDiff,OutHoleDia,OutMaterial,OutTangle,OutTmoment,Wm3),
        % Wm4 :[{InKeyHeight, InShaftDia, InRearShaftDia,
        %              OutKeyHeight,OutShaftDia,OutRearShaftDia}, ...]
        wm_util:merge_wm(3,Wm2,3,Wm3,Wm4),
        % Wm5 : [{OutRpmL,RatioH,RatioL,
        %            InKeyHeight, InShaftDia, InRearShaftDia,
        %            OutKeyHeight,OutShaftDia,OutRearShaftDia}, ...]
        wm_util:merge_wm(3,Wm1,6,Wm4,Wm5),
        reduc_sys:goal_if1(Wm5,GearSpeed,InRpmH,OutRpmH,RatioError,
                                              RatioH,RatioL,WmOut).

%%%
%%%  2段変速ギアユニット
%%%
:- module reduc_sys.
:- public goal_if1/8, goal_if2/6,
          goal1/6, goal2/4, goal3/3, goal4/5, goal5/5, goal6/11, goal7/11.

% WmIn : [{OutRpmL,RatioH,RatioL,
%           InKeyHeight, InShaftDia, InRearShaftDia,
%           OutKeyHeight,OutShaftDia,OutRearShaftDia}, ...]
% WmOut :[[
%            [{OutRpmL,RatioH,RatioL,
%              InKeyHeight, InShaftDia, InRearShaftDia,
%              OutKeyHeight,OutShaftDia,OutRearShaftDia,Module1,
%              ZinH1,ZoutH1,PdinH1,PdoutH1,ZinL1,ZoutL1,PdinL1,PdoutL1},
%              ..............................,
%             {OutRpmL,RatioH,RatioL,
%              InKeyHeight, InShaftDia, InRearShaftDia,
%              OutKeyHeight,OutShaftDia,OutRearShaftDia,Module1,
%              ZinHN,ZoutHN,PdinHN,PdoutHN,ZinLN,ZoutLN,PdinLN,PdoutLN}
%           ],
%              ..............................,
```

```
%            [{OutRpmL,RatioH,RatioL,
%              InKeyHeight, InShaftDia, InRearShaftDia,
%              OutKeyHeight,OutShaftDia,OutRearShaftDia,ModuleK,
%              ZinH1,ZoutH1,PdinH1,PdoutH1,ZinL1,ZoutL1,PdinL1,PdoutL1},
%              ..............................,
%             {OutRpmL,RatioH,RatioL,
%              InKeyHeight, InShaftDia, InRearShaftDia,
%              OutKeyHeight,OutShaftDia,OutRearShaftDia,ModuleK,
%              ZinHN,ZoutHN,PdinHN,PdoutHN,ZinLN,ZoutLN,PdinLN,PdoutLN}
%            ]
%          ]]
goal_if1([In|InTail],GearSpeed,InRpmH,OutRpmH,
                      RatioError,RatioH,RatioL,WmOut) :- true |
        set_vector_element(In ,0,OutRpmL,        OutRpmL,        In1),
        set_vector_element(In1,1,RatioH,         RatioH,         In2),
        set_vector_element(In2,2,RatioL,         RatioL,         In3),
        set_vector_element(In3,3,InKeyHeight,    InKeyHeight,    In4),
        set_vector_element(In4,4,InShaftDia,     InShaftDia,     In5),
        set_vector_element(In5,5,InRearShaftDia, InRearShaftDia, In6),
        set_vector_element(In6,6,OutKeyHeight,   OutKeyHeight,   In7),
        set_vector_element(In7,7,OutShaftDia,    OutShaftDia,    In8),
        set_vector_element(In8,8,OutRearShaftDia,OutRearShaftDia,In9),
        % Wm1 :[{Module1},{Module2}, ....,{ModuleK}]
        goal1(GearSpeed,InRpmH, InKeyHeight, InShaftDia,
                    OutRpm,OutKeyHeight,OutShaftDia,Wm1),
        % Wm2  : [{OutRpmL,RatioH,RatioL,
        %          InKeyHeight, InShaftDia, InRearShaftDia,
        %          OutKeyHeight,OutShaftDia,OutRearShaftDia,Module1},
        %            ..............................,
        %         {OutRpmL,RatioH,RatioL,
        %          InKeyHeight, InShaftDia, InRearShaftDia,
        %          OutKeyHeight,OutShaftDia,OutRearShaftDia,ModuleK}]
        wm_util:merge_vector_and_wm(9,In9,1,Wm1,Wm2),
        % Wm3 : [
        %          [{OutRpmL,RatioH,RatioL,
        %            InKeyHeight, InShaftDia, InRearShaftDia,
        %            OutKeyHeight,OutShaftDia,OutRearShaftDia,Module1,
        %            ZinH1,ZoutH1,PdinH1,PdoutH1,ZinL1,ZoutL1,PdinL1,PdoutL1}
        %             ..............................,
        %           {OutRpmL,RatioH,RatioL,
        %            InKeyHeight, InShaftDia, InRearShaftDia,
        %            OutKeyHeight,OutShaftDia,OutRearShaftDia,Module1,
        %            ZinHN,ZoutHN,PdinHN,PdoutHN,ZinLN,ZoutLN,PdinLN,PdoutLN}],
        %             ..............................,
        %          [{OutRpmL,RatioH,RatioL,
        %            InKeyHeight, InShaftDia, InRearShaftDia,
        %            OutKeyHeight,OutShaftDia,OutRearShaftDia,ModuleK,
        %            ZinH1,ZoutH1,PdinH1,PdoutH1,ZinL1,ZoutL1,PdinL1,PdoutL1}
```

```
        %              ..............................,
        %            {OutRpmL,RatioH,RatioL,
        %             InKeyHeight, InShaftDia, InRearShaftDia,
        %             OutKeyHeight,OutShaftDia,OutRearShaftDia,ModuleK,
        %             ZinHN,ZoutHN,PdinHN,PdoutHN,ZinLN,ZoutLN,PdinLN,PdoutLN}]
        %       ]
        goal_if2(Wm2,GearSpeed,InRpmH,OutRpmH,RatioError,Wm3),
        wm_util:add_wm(WmOut,Wm3,OutTail),
        goal1_if1(InTail,GearSpeed,InRpmH,OutRpmH,OutTail).
goal1_if1([],_,_,_,WmOut) :- true | WmOut   = [].


% WmIn  : [{OutRpmL,RatioH,RatioL,
%           InKeyHeight, InShaftDia, InRearShaftDia,
%           OutKeyHeight,OutShaftDia,OutRearShaftDia,Module1},
%              ..............................,
%          {OutRpmL,RatioH,RatioL,
%           InKeyHeight, InShaftDia, InRearShaftDia,
%           OutKeyHeight,OutShaftDia,OutRearShaftDia,ModuleK},
%        ]
% WmOut : [
%           [{OutRpmL,RatioH,RatioL,
%             InKeyHeight, InShaftDia, InRearShaftDia,
%             OutKeyHeight,OutShaftDia,OutRearShaftDia,Module1,
%             ZinH1,ZoutH1,PdinH1,PdoutH1,ZinL1,ZoutL1,PdinL1,PdoutL1}
%              ..............................,
%            {OutRpmL,RatioH,RatioL,
%             InKeyHeight, InShaftDia, InRearShaftDia,
%             OutKeyHeight,OutShaftDia,OutRearShaftDia,Module1,
%             ZinHN,ZoutHN,PdinHN,PdoutHN,ZinLN,ZoutLN,PdinLN,PdoutLN}
%           ],
%              ..............................,
%           [{OutRpmL,RatioH,RatioL,
%             InKeyHeight, InShaftDia, InRearShaftDia,
%             OutKeyHeight,OutShaftDia,OutRearShaftDia,ModuleK,
%             ZinH1,ZoutH1,PdinH1,PdoutH1,ZinL1,ZoutL1,PdinL1,PdoutL1}
%              ..............................,
%            {OutRpmL,RatioH,RatioL,
%             InKeyHeight, InShaftDia, InRearShaftDia,
%             OutKeyHeight,OutShaftDia,OutRearShaftDia,ModuleK,
%             ZinHN,ZoutHN,PdinHN,PdoutHN,ZinLN,ZoutLN,PdinLN,PdoutLN}
%           ]
%         ]
goal_if2([In|InTail],GearSpeed,InRpmH,OutRpmH,RatioError,WmOut) :- true |
        set_vector_element(In ,0,OutRpmL,        OutRpmL,        In1),
        set_vector_element(In1,1,RatioH,         RatioH,         In2),
        set_vector_element(In2,2,RatioL,         RatioL,         In3),
        set_vector_element(In3,3,InKeyHeight,    InKeyHeight,    In4),
```

```
        set_vector_element(In4,4,InShaftDia,     InShaftDia,     In5),
        set_vector_element(In5,6,OutKeyHeight,   OutKeyHeight,   In6),
        set_vector_element(In6,7,OutShaftDia,    OutShaftDia,    In7),
        set_vector_element(In7,9,Module,         Module,         In8),
        % Wm1 : [{ZinH,ZoutH,PdinH,PdoutH}, ...]
        goal6(RatioH,InKeyHeight,InShaftDia,OutKeyHeight,OutShaftDia,Module,
                                  GearSpeed,InRpmH,OutRpmH,RatioError,Wm1),
        % Wm2 : [{ZinL,ZoutL,PdinL,PdoutL}, ...]
        goal7(RatioL,InKeyHeight,InShaftDia,OutKeyHeight,OutShaftDia,Module,
                                  GearSpeed,InRpmH,OutRpmL,RatioError,Wm2),
        % Wm3 : [{ZinH,ZoutH,PdinH,PdoutH,ZinL,ZoutL,PdinL,PdoutL}, ...]
        goal2(Wm1,Wm2,Wm3),
        % Wm4 : [{OutRpmL,RatioH,RatioL,
        %          InKeyHeight, InShaftDia, InRearShaftDia,
        %          OutKeyHeight,OutShaftDia,OutRearShaftDia,Module1,
        %          ZinH1,ZoutH1,PdinH1,PdoutH1,ZinL1,ZoutL1,PdinL1,PdoutL1}
        %              ..............................,
        %        {OutRpmL,RatioH,RatioL,
        %          InKeyHeight, InShaftDia, InRearShaftDia,
        %          OutKeyHeight,OutShaftDia,OutRearShaftDia,Module1,
        %          ZinHN,ZoutHN,PdinHN,PdoutHN,ZinLN,ZoutLN,PdinLN,PdoutLN}]
        wm_util:merge_vector_and_wm(10,In8,8,Wm3,Wm4),
        wm_util:add_wm(WmOut,Wm4,OutTail),
        goal_if2(InTail,GearSpeed,InRpmH,OutRpmH,RatioError,OutTail).
goal_if2([],_,_,_,_,_,WmOut) :- true | WmOut = [].


% WmOut :[{Module1},{Module2}, ....,{ModuleK}]
goal1(GearSpeed,InRpmH, InKeyHeight, InShaftDia,
                    OutRpm,OutKeyHeight,OutShaftDia,WmOut) :- true |
        get_module_max(GearSpeed,InRpmH, InKeyHeight, InShaftDia,
                       OutRpm,OutKeyHeight,OutShaftDia,ModuleMax),
        get_module(ModuleMax,WmOut).


% Wm1 : [{ZinH,ZoutH,PdinH,PdoutH}, ...]
% Wm2 : [{ZinL,ZoutL,PdinL,PdoutL}, ...]
% WmOut : [{ZinH,ZoutH,PdinH,PdoutH,ZinL,ZoutL,PdinL,PdoutL}, ...]
goal2(Wm1,Wm2,WmOut) :- true |
        shaft_distance_tester:tester(WM1,WM2,WMOut).

% WmOut : [{OutRpmL,RatioH,RatioL}, ....]
goal3(OutRpmH,InTorque,OutTorque,WmOut) :-
        rpm_ratio(InRpmH,OutRpmH,RatioH),
        torque_ratio(InTorque,OutTorque,RatioL),
        out_rpm_low(InRpmH,RatioL,OutRpmL) ,
        wm_util:create_wm(3,[OutRpmL,RatioH,RatioL],WmOut).
```

```
% WmOut :[{InKeyHeight,InShaftDia,InRearShaftDia}, ...]
goal4(InDiff,InHoleDia,InMaterial,InTangle,InTmoment,WmOut) :- true
        in_shaft:goal1(InDiff,InHoleDia,InMaterial,InTangle,InTmoment,WmOut).

% WmOut :[{OutKeyHeight,OutShaftDia,OutRearShaftDia}, ...]
goal5(OutDiff,OutHoleDia,OutMaterial,OutTangle,OutTmoment,WmOut) :- true |
        out_shaft:goal1(OutDiff,OutHoleDia,OutMaterial,OutTangle,OutTmoment,WmOut).

% WmOut : [{ZinH,ZoutH,PdinH,PdoutH}, ...]
goal6(RatioH,InKeyHeight,InShaftDia,OutKeyHeight,OutShaftDia,Module,
                        GearSpeed,InRpmH,OutRpmH,RatioError,WmOut) :- true |
        % Wm1 : [{ZinH1},{ZinH2}, ..., ]
        highspeed_gear_sys:goal1(InKeyHeight,InShaftDia,Module,
                                            GearSpeed,InRpmH,Wm1),
        % Wm2 : [{ZoutH1},{ZoutH2}, ..., ]
        highspeed_gear_sys:goal2(OutKeyHeight,OutShaftDia,Module,
                                            GearSpeed,OutRpmH,Wm2),
        % Wm3 : [{ZinH1,ZoutH1},{ZinH2,ZoutH2}, ..., ]
        highspeed_gear_sys:goal3(Wm1,Wm2,{RatioH,RatioError},Wm3),
        % WmOut : [{ZinH1,ZoutH1,PdinH1,PdoutH1}, ..., ]
        highspeed_gear_sys:goal_if1(Wm3,Module,WmOut).

% WmOut : [{ZinL,ZoutL,PdinL,PdoutL}, ...]
goal7(RatioL,InKeyHeight,InShaftDia,OutKeyHeight,OutShaftDia,Module,
                        GearSpeed,InRpmH,OutRpmL,RatioError,WmOut) :- true |
        % Wm1 : [{ZinL1},{ZinL2}, ..., ]
        lowspeed_gear_sys:goal1(InKeyHeight,InShaftDia,Module,
                                            GearSpeed,InRpmH,Wm1),
        % Wm2 : [{ZoutL1},{ZoutL2}, ..., ]
        lowspeed_gear_sys:goal2(OutKeyHeight,OutShaftDia,Module,
                                            GearSpeed,OutRpmL,Wm2),
        % Wm3 : [{ZinL1,ZoutL1},{ZinL2,ZoutL2}, ..., ]
        lowspeed_gear_sys:goal3(Wm1,Wm2,{RatioL,RatioError},Wm3),
        % WmOut : [{ZinL1,ZoutL1,PdinL1,PdoutL1}, ..., ]
        lowspeed_gear_sys:goal_if1(Wm3,Module,WmOut).

rpm_ratio(InRpm,OutRpm,Ratio) :-
    floating_point(InRpm),
    floating_point(OutRpm) | Ratio $:= InRpm / OutRpm .

torque_ratio(InTorque,OutTorque,Ratio) :-
    floating_point(InTorque),
    floating_point(OutTorque) | Ratio $:= OutTorque / (InTorque*0.75) .

out_rpm_low(InRpmH,LowSpeedRatio,OutRpmL) :-
    floating_point(InRpmH),
    floating_point(LowSpeedRatio) | OutRpmL $:= InRpmH / LowSpeedRatio .
```

```
get_module_max(InRpm, InShaftDia, InKeyHeight,
               OutRpm,OutShaftDia,OutKeyHeight,GearSpeed,ModuleMax) :-
    floating_point(InRpm),
    floating_point(InShaftDia),
    floating_point(InKeyHeight),
    floating_point(OutRpm),
    floating_point(OutShaftDia),
    floating_point(OutKeyHeight) |
        InMax $:= (((1000.0*GearSpeed)/(3.1416*InRpm))
                        - InShaftDia - InKeyHeight) / 7.5,
        OutMax $:= (((1000.0*GearSpeed)/(3.1416*OutRpm))
                        - OutShaftDia - OutKeyHeight) / 7.5,
        math_util:minimum(InMax,OutMax,ModuleMax).

get_module(ModuleMax,WmModule) :-
    floating_point(ModuleMax) |
        gear_module:start_server(module,STR),
        STR = [search(all,2.0,ModuleMax,Num,Module)],
        wm_util:merge_vecotr_and_catalog(0,{},Num,1,Module,WmModule).


%%%
%%%  出力軸
%%%
:- module out_shaft.
:- public goal1/8.

goal1(Diff,Hdia,Material,Tangle,Tmoment,WmOut) :- true |
        shearing_strength(Material,G),
        calc_shaft_dia(Tmoment,Tangle,G,Hdia,Dia_old),
        jis_shaft_dia:jis_shaft_dia(Dia_old,ShaftDia),
        key_height:key_height(ShaftDia,KeyHeight),
        rear_shaft_dia(ShaftDia,Diff,RearShaftDia),
        wm_util:create_wm(3,[KeyHeight,ShaftDia,RearShaftDia],WmOut).

shearing_strength(Material,G) :- true |  G $:= 8200.0 .

calc_shaft_dia(Tmoment,Tangle,G,Hdia,Shaft_dia) :-
    floating_point(Tmoment),
    floating_point(Tangle),
    floating_point(G),
    floating_point(Hdia)   |
        D4 $:= (5760000.0*Tmoment)/
                    (G*9.869604*Tangle),
        Hdia4 $:= Hdia*Hdia*Hdia*Hdia,
        D4_dh $:= D4 + Hdia4,
        math_func:sqrt(D4_dh,D2_dh),
        math_func:sqrt(D2_dh,Shaft_dia).
```

```
rear_shaft_dia(Shaft_dia,Diff,Rear_shaft_dia) :-
    floating_point(Shaft_dia),
    floating_point(Diff)     |  Rear_shaft_dia $:= Shaft_dia - Diff.


%%%
%%%  入力軸
%%%
:- module in_shaft.
:- public goal1/8.

goal1(Diff,Hdia,Material,Tangle,Tmoment,WmOut) :- true |
        shearing_strength(Material,G),
        calc_shaft_dia(Tmoment,Tangle,G,Hdia,Dia_old),
        jis_shaft_dia:jis_shaft_dia(Dia_old,ShaftDia),
        key_height:key_height(ShaftDia,KeyHeight),
        rear_shaft_dia(ShaftDia,Diff,RearShaftDia),
        wm_util:create_wm(3,[KeyHeight,ShaftDia,RearShaftDia],WmOut).

shearing_strength(Material,G) :- true |    G $:= 8200.0 .

calc_shaft_dia(Tmoment,Tangle,G,Hdia,Shaft_dia) :-
    floating_point(Tmoment),
    floating_point(Tangle),
    floating_point(G),
    floating_point(Hdia)   |
        D4 $:= (5760000.0*Tmoment)/
                    (G*9.869604*Tangle),
        Hdia4 $:= Hdia*Hdia*Hdia*Hdia,
        D4_dh $:= D4 + Hdia4,
        math_func:sqrt(D4_dh,D2_dh),
        math_func:sqrt(D2_dh,Shaft_dia).

rear_shaft_dia(Shaft_dia,Diff,Rear_shaft_dia) :-
    floating_point(Shaft_dia),
    floating_point(Diff)     |   Rear_shaft_dia $:= Shaft_dia - Diff.


%%%
%%%  高速側ギアユニット
%%%
:- module highspeed_gear_sys.
:- public
        goal1/7, goal2/7, goal3/4, goal4/4, goal5/4, goal6/3.

goal1(InKeyHeight,InShaftDia,Module,GearSpeed,InRpm,WmOut):- true |
        get_tnum_min(InShaftDia, InKeyHeight, Module,ZinMin),
```

```
        get_tnum_max(GearSpeed,InRpm, Module,ZinMax) ,
        generator:generator(ZinMin,ZinMax,1.0,WmOut).

goal2(OutKeyHeight,OutShaftDia,Module,GearSpeed,OutRpm,WmOut):- true |
        get_tnum_min(OutShaftDia,OutKeyHeight,Module,ZoutMin),
        get_tnum_max(GearSpeed,OutRpm,Module,ZoutMax),
        generator:generator(ZoutMin,ZoutMax,1.0,WmOut).

%  WM1 = InGear   WM2 = OutGear
goal3(WM1,WM2,Para,WmOut) :- true | gear_ratio_tester:tester(WM1,WM2,Para,WmOut).

goal_if1([In|InTail],Module,WmOut) :-  ture |
        set_vector_element(In, 0, Zin, Zin, In1),
        set_vector_element(In1,1, Zout,Zout,In2),
        goal4(Zin,Zout,Module,Wm1),
        wm_util:reshape_wm(Wm1,V1),
        wm_util:expand_element(In2,V1,_,_,V2),
        wm_util:add_element(V2,WmOut,WmOut1),
        goal_if1(InTail,Module,WmOut1).
goal_if1([],_,WmOut) :- true | WmOut = [].

goal4(Zin,Zout,Module,WmOut) :- true |
        get_pitch_dia(Zin, Module,PdiaIn),
        get_pitch_dia(Zout,Module,PdiaOut),
        wm_util:create_wm(2,[PdiaIn,PdiaOut],WmOut).

get_tnum_min(ShaftDia,Key_height,Module,Tnum_min):-
    floating_point(ShaftDia),
    floating_point(Key_height),
    floating_point(Module)    |
        Tnum_float $:= (ShaftDia+Key_height)/Module+7.0,
        math_util:cut_up_float(Tnum_float,Tnum_min).

get_tnum_max(GearSpeed,ShaftRpm,Module,Tnum_max) :-
    floating_point(GearSpeed),
    floating_point(ShaftRpm),
    floating_point(Module)    |
        Tnum_float $:= (1000.0*GearSpeed)/
                            (3.14159*ShaftRpm*Module),
        math_util:cut_away_float(Tnum_float,Tnum_max).

get_pitch_dia(Tnum,Module,Pdia) :-
    floating_point(Tnum),
    floating_point(Module) |  Pdia $:= Tnum*Module.

shaft_distance(In_pdia,Out_pdia,Shaft_dis) :-
    floating_point(In_pdia),
    floating_point(Out_pdia) |  Shaft_dis $:= Out_pdia+In_pdia.
```

```
%%%
%%%  低速側ギアユニット
%%%
:- module lowspeed_gear_sys.
:- public
        goal1/7, goal2/7, goal3/4, goal4/4, goal5/4, goal6/3.

goal1(InKeyHeight,InShaftDia,Module,GearSpeed,InRpm,WmOut):- true |
        get_tnum_min(InShaftDia, InKeyHeight, Module,ZinMin),
        get_tnum_max(GearSpeed,InRpm, Module,ZinMax) ,
        generator:generator(ZinMin,ZinMax,1.0,WmOut).

goal2(OutKeyHeight,OutShaftDia,Module,GearSpeed,OutRpm,WmOut):- true |
        get_tnum_min(OutShaftDia,OutKeyHeight,Module,ZoutMin),
        get_tnum_max(GearSpeed,OutRpm,Module,ZoutMax).
        generator:generator(ZoutMin,ZoutMax,1.0,WmOut).

%  WM1 = InGear    WM2 = OutGear
goal3(WM1,WM2,Para,WmOut) :- true | gear_ratio_tester:tester(WM1,WM2,Para,WmOut).

goal_if1([In|InTail],Module,WmOut) :-  ture |
        set_vector_element(In, 0, Zin, Zin, In1),
        set_vector_element(In1,1, Zout,Zout,In2),
        goal4(Zin,Zout,Module,Wm1),
        wm_util:reshape_wm(Wm1,V1),
        wm_util:expand_element(In2,V1,_,_,V2),
        wm_util:add_element(V2,WmOut,WmOut1),
        goal_if1(InTail,Module,WmOut1).
goal_if1([],_,WmOut) :- true | WmOut = [].

goal4(Zin,Zout,Module,WmOut) :- true |
        get_pitch_dia(Zin, Module,PdiaIn),
        get_pitch_dia(Zout,Module,PdiaOut),
        wm_util:create_wm(2,[PdiaIn,PdiaOut],WmOut).

get_tnum_min(ShaftDia,Key_height,Module,Tnum_min):-
    floating_point(ShaftDia),
    floating_point(Key_height),
    floating_point(Module)    |
        Tnum_float $:= (ShaftDia+Key_height)/Module+7.0,
        math_util:cut_up_float(Tnum_float,Tnum_min).

get_tnum_max(GearSpeed,ShaftRpm,Module,Tnum_max) :-
    floating_point(GearSpeed),
    floating_point(ShaftRpm),
    floating_point(Module)    |
```

```
        Tnum_float $:= (1000.0*GearSpeed)/
                              (3.14159*ShaftRpm*Module),
        math_util:cut_away_float(Tnum_float,Tnum_max).

get_pitch_dia(Tnum,Module,Pdia) :-
    floating_point(Tnum),
    floating_point(Module) | Pdia $:= Tnum*Module.

shaft_distance(In_pdia,Out_pdia,Shaft_dis) :-
    floating_point(In_pdia),
    floating_point(Out_pdia) | Shaft_dis $:= Out_pdia+In_pdia.


%%%
%%%  ギア歯数のジェネレータ
%%%
:- module gen_type0.
:- public generator/4.

%   WM : [{ele1},{ele2},..{eleN}]
generator(Min,Max,Step,WM) :-
    Max $>= Min,
    floating_point(Step) |
        GenSizeFloat $:= ((Max -Min)/Step) + 1.0,
        floating_point_to_integer(GenSizeFloat,GenSize),
        gen(0,GenSize,Min,Step,WM).
otherwise.
generator(_,_,_,WM) :- true | WM = [].

gen(Pt,GenSize,Min,Step,WM) :-
    Pt < GenSize,
    floating_point(Min),
    floating_point(Step) |
        new_vector(Vsub,1),
        set_vector_element(Vsub,0,_,Min,Vsub1),
        Min1 $:= Min + Step,
        Pt1   := Pt + 1,
        WM = [Vsub1|WMNew],
        gen(Pt1,GenSize,Min1,Step,WMNew).
gen(Pt,GenSize,_,_,WM) :-  Pt >= GenSize |  WM = [].


%%%
%%%  軸間距離のテスタ
%%%
:- module   shaft_distance_tester.
:- public   tester/3.
```

```
% tester(WM1,WM2,WM3)
%  WM1 : [{Zin1,Zout1,Pdin1,Pdout1}, .... ]
%  WM2 : [{Zin2,Zout2,Pdin2,Pdout2}, .... ]
%  WM3 : [{Zin1,Zout1,Pdin1,Pdout1,Zin2,Zout2,Pdin2,Pdout2}, .... ]

tester([V1|Tail1],Wm2,Wm3) :- true |
        test_sub1(V1,Wm2,Wm2New,Wm3,Wm3Tail),
        tester(Tail1,Wm2New,Wm3Tail).
tester([],_,Wm3) :- true |   Wm3 = [].

test_sub1(V1,[V2|Tail],Wm2New,Wm3,Wm3Tail) :- true |
        test_and_merge(V1,V2,V11,V21,Status),
        check_result(Status,V11,V21,V12,V22,Wm3,Wm3New),
        Wm2New = [V22|Tail2New],
        test_sub1(V12,Tail2,Tail2New,Wm3New,Wm3Tail).
test_sub1(_,[],Wm2New,Wm3,Wm3Tail) :- true | Wm2New   = [], Wm3Tail  = Wm3.

check_result(Status,V1,V2,V1New,V2New,Wm3,Wm3New) :- Status = ok |
        wm_util:expand_element(V1,V2,V1New,V2New,V3),
        wm_util:add_element(V3,Wm3,Wm3New).
otherwise.
check_result(Status,V1,V2,V1New,V2New,Wm3,Wm3New) :- true |
        V1New = V1, V2New = V2, Wm3New = Wm3.

%%%%  !!! Interface for User Difined Method !!!
test_and_merge(V1,V2,V1New,V2New,Status) :- true |
        set_vector_element(V1, 0,Zin1,     Zin1,     V11),
        set_vector_element(V11,1,Zout1,    Zout1,    V1New),
        set_vector_element(V2, 0,Zin2,     Zin2,     V21),
        set_vector_element(V21,1,Zout2,    Zout2,    V2New),
        shaft_distance_tester(Zin1,Zout1,Zin2,Zout2,Status).

%%%%  !!! User Difined Method !!!
shaft_dsitance_tester(Zin1,Zout1,Zin2,Zout2,Status) :-
    X $:= Zin1 + Zout1,
    Y $:= Zin2 + Zout2,
    X = Y |  Status = ok.
otherwise.
shaft_distance_tester(_,_,_,_,Status) :- true |  Status = no.


%%%
%%%  ギア比のテスタ
%%%
:- module gear_ratio_tester.
:- public tester/4.

tester([V1|Tail1],Wm2,Vpara,Wm3) :- true |
```

```
        test_sub1(V1,Wm2,Vpara,Wm2New,VparaNew,Wm3,Wm3Tail),
        tester(Tail1,Wm2New,VparaNew,Wm3Tail).
tester([],_,_,Wm3) :- true |  Wm3 = [].

test_sub1(V1,[V2|Tail],Vpara,Wm2New,VparaNew,Wm3,Wm3Tail) :- true |
        test_and_merge(V1,V2,Vpara,V11,V21,Vpara1,Status),
        check_result(Status,V1,V2,V12,V22,Wm3,Wm3New),
        Wm2New = [V22|Tail2New],
        test_sub1(V12,Tail2,Vpara1,Tail2New,VparaNew,Wm3New,Wm3Tail).
test_sub1(_,[],Vpara,Wm2New,VparaNew,Wm3,Wm3Tail) :- true |
        VparaNew = Vpara, Wm2New   = [], Wm3Tail  = Wm3.

check_result(Status,V1,V2,V1New,V2New,Wm3,Wm3New) :- Status = ok |
        wm_util:expand_element(V1,V2,V1New,V2New,V3),
        wm_util:add_element(V3,Wm3,Wm3New).
otherwise.
check_result(Status,V1,V2,V1New,V2New,Wm3,Wm3New) :- true |
        V1New = V1, V2New = V2, Wm3New = Wm3.

%%%%  !!! Interface for User Difined Method !!!
test_and_merge(V1,V2,Vpara,V1New,V2New,Vpara1,Status) :- true |
        set_vector_element(V1, 0,Zin,     Zin,     V1New),
        set_vector_element(V2, 0,Zout,    Zout,    V2New),
        set_vector_element(Vpara, 0,Ratio,     Ratio,     Vpara1),
        set_vector_element(Vpara1,1,RatioError,RatioError,VparaNew),
        gear_ratio_tester(Zin,Zout,Ratio,RatioError,Status).

%%%%  !!! User Difined Method !!!
gear_ratio_tester(Zin,Zout,Ratio,RatioError,Status) :-
    R $:= Zout / Zin,
    Error $:= Ratio*RatioError,
    Rmin $:= Ratio - Error,
    Rmax $:= Ratio + Error,
    Rmin $=< R,R $=< Rmax | Status = ok.
otherwise.
gear_ratio_tester(_,_,_,_,Status) :- true | Status = no.
```