知識工学と人工知能 31－8
（1983. 9. 27）

ADLによる機械設計システム

長澤 勲* 古川 由美子* 荒牧 重登**
*九州大学中央計数施設 **九州大学情報処理教育センター

1. はじめに

機械設計プログラムは、FORTRANなどで記述され、教育や設計の現場で使用されている。[1] しかし、これらのプログラムはデータフローが固定されており、使用目的ごとに多数のプログラムが用意されている。

一方、筆者らが開発したprologを基礎とした設計記述言語ADL[4]を用いて記述した設計プログラムは、設計規約や設計公式などの設計知識や試行錯誤的な設計過程が自然に、統一的に表現され、FORTRANなどで書かれた現存の設計システムに比べ、次のような優れた性質を持っている。

(1)設計者は、設計知識や仕様を拘束条件の形で設計システムに与え、これを満足する解をシステムの支援のもとに探索することができる。このため設計者は、設計、設計の検証、部分変更など多目的に設計システムを利用できる。

(2)拘束条件解法はリダクション手法を用いており、種々の設計計算が統一的に表現できる。

(3)設計知識はシステム内で対象指向に管理され、高度にモジュール化されており、可用性、拡張性、保守性が良い。

(4)知識ベースエディタや解探索時におけるデバッガなどプログラミング環境が整備されている。

(5)設計知識は対象単位で階層化されているので設計情報の検索が容易に実現できる。

本報告では、機械設計プログラムの例、モジュール性、可用性を高めるためのプログラミング手法、解探索効率化の手法について述べる。

2. 設計プログラムの例

本節では、歯車減速機設計における平歯車システムの設計プログラムおよび実行結果の一例を示す。平歯車システムの拘束条件は前出資料の図3のように表わされる。図1は平歯車システムのフレーム表現とリダクションルールである。平歯車システムは、歯車(gear)と歯車システム(gearsys)の二つのフレームによって表わされている。フレームのスロットは、歯車や歯車システムの属性を、手続部は各々の拘束条件を表わす。ここでは、拘束条件を便宜的に回転数と速度比(＝)，歯数対と速度比(reduction_ratio)，歯車形状(gear_shape)，歯の曲げ強さ(lewis)に分けている。また、gear_sysは部分構造としてgearを選択又は構成する。拘束条件集合の解法には生成検証法を用いており、歯数対、モジュール、歯車材料をそれぞれ、reduction_ratio，gear_shape，lewisの各ルールによって仮定し、歯の曲げ強さの検証はlewisのルールによっている。

設計問題は図2(a)のように与えられる。solveでCRSが呼出され、スロット値の結合，拘束条件集合のリダクションが行われる。同図(b)はこの問題を解く過程のトレースである。図中リダクションされる拘束条件のみを示している。レベル番号625のgear_shapeで歯車のモジュールを3.0mmに仮定すると、レベル番号959のlewisで失敗しバックトラックしている。モジュールを4.0mmに仮定すると一つの解同図(c)が得られる。

```
(frame gear
   (Obj (ako gear)
        (rpm N)
        (power_kw L)
        (module M)
        (no_of_teeth Z)
        (pitch_circle_diameter D)
        (material Mat)
        (torque T)
        (pitch_circle_force Fs))
   (proc (gear_shape M Z D B)
         (lewis M Z N D L Mat T Fs)))

(frame gearsys
   (Obj (reduction_ratio U)
        (power_kw L)
        (driving_gear G1)
        (drived_gear G2))
   (proc (fget
          (G1 (ako gear)
              (module M)
              (no_of_teeth Z1)
              (rpm N1)
              (power_kw L))
          (G2 (ako gear)
              (module M)
              (no_of_teeth Z2)
              (rpm N2)
              (power_kw L)))
         (N2 = N1 * U)
         (reduction_ratio U Z1 Z2)))

(reduction_ratio ?u Z1 ?z2):- !! ,
   (Z2 * U = Z1),(13 < Z1),(Z1 < 101),
   (13 < Z2),(Z2 < 101)
(reduction_ratio U ?z1 ?z2):- !! ,
   (Z2 * U = Z1),(13 < Z1),(Z1 < 101),
   (13 < Z2),(Z2 < 101)
(reduction_ratio ?u Z1 Z2):- !! ,
   (for Z1 14 100),(Z2 * U = Z1),
   (13 < Z2),(Z2 < 101)

(gear_shape M ?Z D B):-
   (module M),
   (D = M * Z),
   (B = 10 * M).

(lewis ?M ?Z ?N ?D ?L Mat T Fs):-
   (60000 * V = 3.141592E+00  * D * N),
   (toothshape_coef Z Kz),
   (velosity_coef V Kv),
   (gear_material Mat _ _ Sigwb _ _),
   (N * T = 9.739995E+00  * 10000 * L),
   (Fs * D = 2 * T * 10),
   (M ** 3 * Kv * 10 * Z * Kz * Sig =
    2 * 1.250000E+00 * T * 10),
   (Sig <= Sigwb).
```

図1 平歯車システムのフレーム表現とリダクションルール

```
:- (solve (fget
      (gs1 (ako gearsys)
           (reduction_ratio 0.25)
           (power_kw 7.5)
           (driving_gear g1))
      (g1 (ako gear)
          (rpm 600)
          (material s45c)))))
```

伝達動力 7.5KW，回転数600rpm の原動機から1/4に減速する一組の平歯車を設計せよ。ただし小歯車の材質はS45C（鋳鋼）とする。

図2(a) 設計問題の一例とコード化

```
(g200002
  (ako gear)
  (rpm 150.0)
  (power_kw 7.5)
  (module 4.0)
  (no_of_teeth 56)
  (pitch_circle_diameter 224.0)
  (material sc49)
  (torque 4.869996E+03)
  (pitch_circle_force 4.348208E+02))
 (g1
  (ako gear)
  (rpm 600)
  (power_kw 7.5)
  (module 4.0)
  (no_of_teeth 14)
  (pitch_circle_diameter 56.0)
  (material s45c)
  (torque 1.217499E+03)
  (pitch_circle_force 4.348210E+02))
 (gs1
  (ako gearsys)
  (reduction_ratio 0.25)
  (power_kw 7.5)
  (driving_gear g1)
  (drived_gear g200002))
```

図2(c) 解の一例

```
(gear_material fc20 1.050000E+04  17 3.500000E+00  170 _)
(gear_material fc25 1.150000E+04  22 5.000000E+00  200 _)
(gear_material fc30 1.260000E+04  27 6.000000E+00  220 _)
(gear_material fc35 1.350000E+04  32 6.500000E+00  240 _)
(gear_material sc37 2.000000E+04  37 1.050000E+01  105 _)
(gear_material sc42 2.000000E+04  42 1.200000E+01  105 _)
(gear_material sc46 2.000000E+04  46 1.250000E+01  130 _)
(gear_material sc49 2.000000E+04  49 1.300000E+01  137 _)
(gear_material s35c 2.100000E+04  58 1.750000E+01  160 600)
(gear_material s40c 2.100000E+04  62 1.800000E+01  173 600)
(gear_material s45c 2.100000E+04  70 2.100000E+01  196 600)
(gear_material s50c 2.100000E+04  75 2.200000E+01  210 600)
```

```
(20) (call 0 0)
     (fget (gs1 (ako gearsys)
                (reduction_ratio 0.25)
                (power_kw 7.5)
                (driving_gear g1))
           (g1 (ako gear) (rpm 600)
               (material s45c)))
(20) (exit 0 0)
     (fget (gs1 (ako gearsys)
                (reduction_ratio 0.25)
                (power_kw 7.5)
                (driving_gear g1))
           (g1 (ako gear) (rpm 600)
               (material s45c)))
(233) (call 0 0)
      (fget (g1 (ako gear)(module  M.280)
                (no_of_teeth 14)
                (rpm  N1.284)
                (power_kw 7.5))
        (G2.278 (ako gear)(module  M.280)
                (no_of_teeth 56)
                (rpm  N2.288)
                (power_kw7.5)))
(233) (exit 0 0)
      (fget (g1 (ako gear)(module  M.280)
                (no_of_teeth 14)
                (rpm 600) (power_kw 7.5))
       (g200002 (ako gear)(module  M.280)
                (no_of_teeth 56)
                (rpm  N2.288)
                (power_kw 7.5)))
(456) (call 0 2)
      (reduction_ratio 0.25  Z1.282
                       Z2.286)
(456) (exit 0 2)
      (reduction_ratio 0.25 14 56)
(562) (call 0 2)
      (=  N2.288 (* 600 0.25))
(562) (exit 0 2)
      (= 150 (* 600 0.25))
(625) (call 0 3)
      (gear_shape  M.280 56  D.2948
                   B.2956)
(625) (exit 0 3)
      (gear_shape 3.0 56 168.0 30.0)
(857) (call 0 2)
      (gear_shape 3.0 14  D.661  B.669)
(857) (exit 0 2)
      (gear_shape 3.0 14 42.0 30.0)
(959) (call 0 36)
      (lewis 3.0 14 600 42.0 7.5 s45c
             T.665  Fs.667)
(959) (fail 0 36)
      (lewis 3.0 14 600 42.0 7.5 s45c
             T.665  Fs.667)
(857) (redo 0 2)
      (gear_shape 3.0 14 42.0 30.0)
(857) (fail 0 2)
      (gear_shape 3.0 14  D.661  B.669)
(625) (redo 0 3)
      (gear_shape 3.0 56 168.0 30.0)
(625) (exit 0 3)
      (gear_shape 4.0 56 224.0 40.0)
(857) (call 0 2)
      (gear_shape 4.0 14  D.661  B.669)
(857) (exit 0 2)
      (gear_shape 4.0 14 56.0 40.0)
(959) (call 0 36)
      (lewis 4.0 14 600 56.0 7.5 s45c
             T.665  Fs.667)
(959) (exit 0 36)
      (lewis 4.0 14 600 56.0 7.5 s45c
             1.217499E+03  4.348210E+02)
(1328) (call 0 54)
       (lewis 4.0 56 150.0 224.0 7.5
              Mat.2950 T.2952 Fs.2954)
(1328) (exit 0 54)
       (lewis 4.0 56 150.0 224.0 7.5 sc49
              4.869996E+03 4.348208E+02)
```

図2(b) リダクション過程のトレース

3. プログラミング手法

本設計システムでは、システムのモジュール性、多目的性、可用性をさらに高めるために以下に述べるプログラミング手法が有効である。

〔設計対象の抽象化と概念階層の利用〕

機械システムは近似的にいくつかの要素(部品)に分解され、設計公式もそれに対応して用意されている。従って、各要素は使用される環境、相互作用する相手の影響を抽象化することによって一般的に概念化できる場合が多い。次の例は、歯車減速機設計における抽象化の例である。

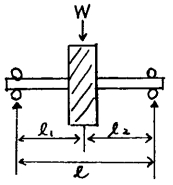
例1. 歯面圧強さを検証する手続き hertz の抽象化

一組の平歯車の設計において、縦弾性係数の調和平均、歯数の調和平均を歯車の環境を代表するものと考えると主動歯車、従動歯車の区別をなくすことができ、hertz を歯車の拘束条件として表現できる。

例2. 軸設計の抽象化

軸設計のフレームでは、ねじりモーメント、作用力をスロットにし環境を抽象化する。このことによってこのフレームは歯車軸やポンプ軸等の設計で

共用できる．歯車軸の設計では歯車のフレームが決定したスロット値を用いる。また、軸の固有振動数を表わす拘束条件の一例は図3で示されるが、軸径、回転体の質量、軸の長さ、回転体の位置、軸受の種数を固有振動数のフレームのスロットにすることによって固有振動数を求める手続きは抽象化される．軸システム設計ではこのフレームを上位概念として扱うことによって可用性が保証される．



中央に質量を持つ両端支持軸
軸受はころがり軸受

$$\omega_{nb} = \sqrt{gk/W}$$

$$k = 3EIl/(l_1^2 l_2^2)$$

$$I = \pi d^4(1-\nu^4)/64$$

$$\nu = d_1/d$$

図3 軸の固有振動系の拘束条件[2)]

次に、抽象化された設計対象をADLのフレームが持っている階層関係（部分－全体階層、上位－下位概念階層）を用いて階層化する．図5は歯車減速機のフレームの一部とその概念階層を表わしている．歯車システムは、形状のみを考慮する歯車システム、曲げ強さを考慮した歯車システム、歯面圧強さを考慮した歯車システム、曲げと歯面圧強さを考慮した歯車システムに抽象化され、図のように階層化される．

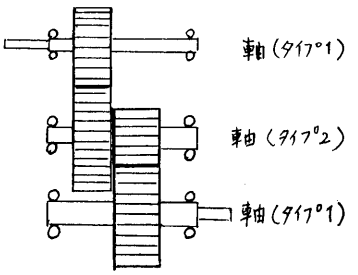


図4 二段歯車減速機

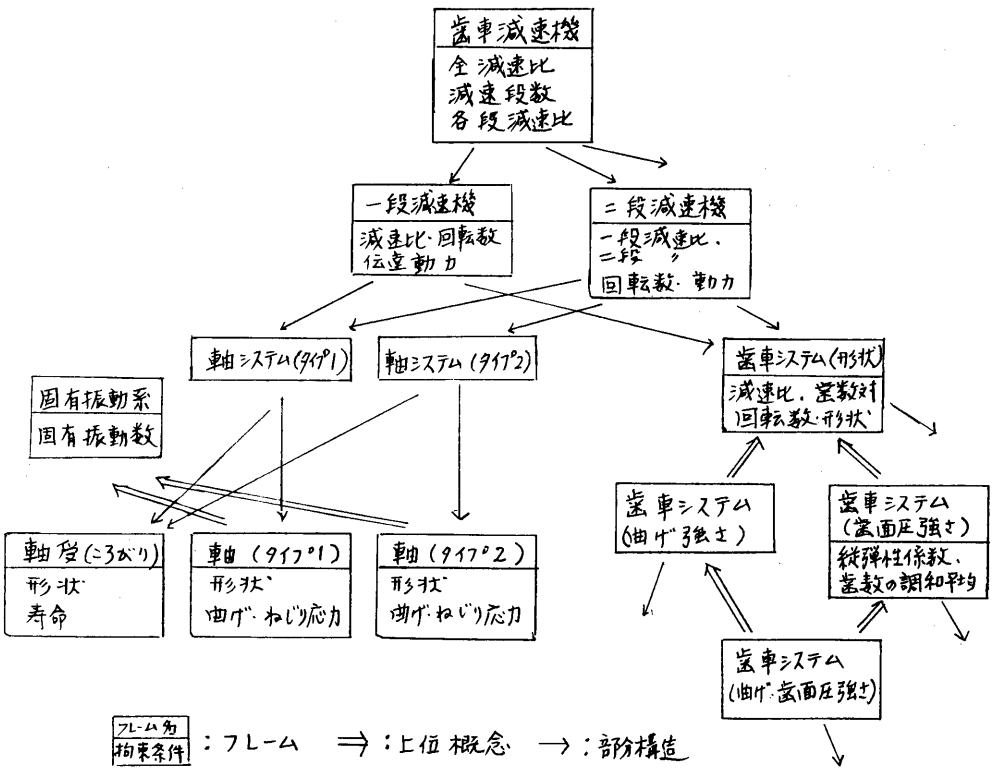


図5 歯車減速機のフレームの概念階層

同様に歯車も、歯車(形状)，歯車(曲げ強さ)，歯車(歯面圧強さ)，歯車(曲げ、歯面圧強さ)に抽象化され、階層化される。(図5では省略)
歯車システムの各々のフレームは、対応する歯車フレームを部分構造として持っている。これらの階層化によって設計プログラムは、所要レベルの設計に柔軟に対応でき、可用性が高められている。図6に階層化された歯車システムのフレームと歯車フレームを示す。
ako スロットで上位フレームを指定することにより、上位フレームのスロット、拘束条件が継承される。

```
(frame gear
 (Obj (rpm N)
      (module M)
      (no_of_teeth Z)
      (pitch_circle_diameter D)
      (material Mat)
      (tooth_width B))
 (proc (gear_shape M Z? D B)))

(frame gear_lewis
 (Obj (ako gear)
      (rpm N)
      (power_kw L)
      (module M)
      (no_of_teeth Z)
      (pitch_circle_diameter D)
      (material Mat)
      (torque T)
      (pitch_circle_force Fs))
 (proc (lewis M Z N D L Mat T Fs)))

(frame gear_hertz
 (Obj (ako gear)
      (module M)
      (pitch_circle_diameter D)
      (rpm N)
      (power_kw L)
      (material Mat)
      (e_mean Em)
      (z_mean Zm)
      (hardning Yes_no)
      (hardness Hb))
 (proc
  (hertz M? Em? Zm? D? N? L? Mat
         Yes_no Hb)))

(frame gear_lewis_hertz
 (Obj (ako gear_lewis gear_hertz))
 (proc))
```

```
(frame gearsys
 (Obj (reduction_ratio U)
      (driving_gear G1)
      (drived_gear G2))
 (proc
  (fget (G1 (ako gear)
            (module M)
            (no_of_teeth Z1)
            (rpm N1)
            (tooth_width B)
            (material Mat1))
        (G2 (ako gear)
            (module M)
            (no_of_teeth Z2)
            (rpm N2)
            (tooth_width B)
            (material Mat2)))
  (N2 = N1 * U)
  (reduction_ratio U Z1 Z2)))

(frame gearsys_hertz
 (Obj (ako gearsys)
      (power_kw L)
      (driving_gear G1)
      (drived_gear G2))
 (proc
  (fget (G1 (ako gear_hertz)
            (no_of_teeth Z1)
            (power_kw L)
            (material Mat1)
            (e_mean Em)
            (z_mean Zm))
        (G2 (ako gear_hertz)
            (no_of_teeth Z2)
            (power_kw L)
            (material Mat2)
            (e_mean Em)
            (z_mean Zm)))
  (gearsys_em_zm Mat1? Mat2? Z1? Z2?
                 Em Zm)))

(frame gearsys_lewis
 (Obj (ako gearsys)
      (power_kw L)
      (driving_gear G1)
      (drived_gear G2))
 (proc
  (fget
   (G1 (ako gear_lewis) (power_kw L))
   (G2 (ako gear_lewis) (power_kw L)))))

(frame gearsys_lewis_hertz
 (Obj (ako gearsys_lewis gearsys_hertz)
      (driving_gear G1)
      (drived_gear G2))
 (proc
   (fget (G1 (ako gear_lewis_hertz))
         (G2 (ako gear_lewis_hertz)))))
```

図6 階層化された歯車システムのフレーム

〔ボトムアッププログラミング〕

ADLで記述された機械設計システムでは、設計知識は対象単位に抽象化、階層化されるので、ボトムアッププログラミングが容易である。

設計対象が与えられると、まず、最小単位の部分構造(部品)に分解し、その部分を、使用される環境を抽象化してフレームを作成し、次に、それらの部分構造に新しい属性、拘束条件を付加して抽象化された構造物を作り、部分ー全体階層を作る。この手順を繰返して、ボトムアップに設計システムを構成することができる。この方法により作られたフレームは、そのフレームが呼ばれる文脈に依存しないので、システムは高度にモジュール化、多目的化されたものとなる。例えば、歯車減速機の設計システムにおいては、まず歯車、軸、軸受等のフレームを作り、これを用いて階層化された歯車システム、軸システムのフレームを作る。ポンプの設計システムを構成する場合軸と軸受のフレームはそのまま使用可能である。

〔生成・検証法〕

機械設計では生成・検証の技法(解又はその一部を仮定し、方程式を解き不等式を用いて検証する方法)は従来から多用されている。その理由は、規格にある材料や部品を使用して多くの拘束条件の妥協点を求めることによって機械を構成することが多いからである。ADLでは、仮定の生成・伝播・検証はすべてリダクション・ルールによって表現でき、生成・検証法の表現は容易である。例えば、図1の歯車システムの設計では、reduction_ratioにおいて歯数対を、gear_shapeにおいてモジュールを仮定し、lewisにおいて歯車材料を仮定して歯の曲げ強さを検証している。またこの手法は拘束条件解法を強化し、システムの可用性を高めるためにも有効である。

〔非線型性の削減〕

設計公式の多くは非線型な部分を含んでいるが、上に述べた生成検証法を用いることによって非線型を減らすことができる。例えば、歯車のモジュールを仮定することによってモジュールについて方程式を解くことを避けることができる。

4. 解探索効率化の手法

生成・検証法を多用する設計システムでは、解探索においてバックトラックが非常に多くなる。このためADLの拘束条件リダクション手続きでもコスト評価による効率改善を行っているが、さらに効率化するためADLの利用者は次のような手法を用いることができる。

〔priority の利用〕

機械設計には、基本設計、詳細設計などの設計段階がある。また基本設計においても明らかに設計順序が定まっている場合もある。これらの階層を表現するため、設計システムでは個々の拘束条件にpriorityを与えて拘束条件を階層化し、解探索の効率化を計る。

〔generatorの利用〕

生成・検証法は設計者に理解されやすいが、バックトラックをともなうため問題が大きくなれば解探索に要する時間も急増する。しかし、設計者は一般にどのような解が望ましいか、どのように範囲に解が存在するか知っていることが多いので、この知識をgenerator に反映させ、解の探索範囲を小さくする。図7(a)はgeneratorを外付けにしたルールによって行う方法、

同図(b)は拘束条件を追加し、生成した仮定を直ちに検証する方法である。

```
:-(solve
   (member M (4.0 5.0 6.0))
   (fget (gs1 (ako gearsys)
              (reduction_ratio 0.25)
              (power_kw 7.5)
              (driving_gear g1))
         (g1  (ako gear)
              (rpm 600)
              (module M?)
              (material s45c)))).
```

(a) 仮定生成ルールの外付け

```
:-(solve
   (test D? M?)
   (fget (gs1 (ako gearsys)
              (reduction_ratio 0.25)
              (power_kw 7.5)
              (driving_gear g1))
         (g1  (ako gear)
              (rpm 600)
              (pitch_circle_diameter D)
              (material s45c)))).
(test ?D ?M) :- (D>70.0),(4.0<M),
                (M<6.0).
```

(b) 拘束条件の追加

図7 generatorの利用

[CRS呼出し(solve)の階層化]

一つの設計問題は拘束条件集合で表わされ、その解はCRSのリダクション手続きにより求められる。しかし大きな機械システムの設計では拘束条件集合が大きくなり解探索空間が広がるので、解探索に要する時間も急増するという難点がある。そこで、拘束条件集合を部分構造のような意味のあるいくつかの集合に分け、階層化し、各々の集合をリダクションする手法を用いる。これはフレームの拘束条件にsolveを書くことによって実現される。同レベルのsolveには入力条件を記述し、実行を制御する。階層化されたsolveは子から親へ、兄弟はメカ条件を満足した集合から、決定的に実行される。図8は一段歯車減速機のフレームにおけるsolveの階層化の例である。

```
(frame one_step_reduction_sys
 (Obj (reduction_ratio U)
      (in_power_kw L1)
      (out_power_kw L2)
      (in_rpm N1)
      (out_rpm N2)
      (condition Cond)
      (shaft_length Sl)
      (gearsys Gs1)
      (shaftsys1 Ss1)
      (shaftsys2 Ss2))
 (proc
  (solve (wait T1 Fs1 Sl N1 Cond)
     (fget (Ss1 (ako shaftsys_type1)
                (shaft S1)
                (bearing_1 B1)
                (bearing_2 B2))
           (S1 (ako shaft_type1)
               (torsional_moment T1)
               (force Fs1)
               (length Sl)
               (length_a Sla)
               (length_b Slb))
           (B1 (ako bearing)
               (rpm N1)
               (condition Cond))
           (B2 (ako bearing)
               (rpm N1)
               (condition Cond))))
  (solve (wait T3 Fs2 Sl N2 Cond)
     (fget (Ss2 (ako shaftsys_type1)
                :(中略)
                )))
  (solve (wait L1)
     (fget (Gs1 (ako gearsys)
                (reduction_ratio U)
                (power_kw L1)
                (driving_gear G1)
                (drived_gear G2))
           (G1 (ako gear)
               (rpm N1)
               (torque T1)
               (pitch_circle_force Fs1))
           (G2 (ako gear)
               (rpm N2)
               (torque T2)
               (pitch_circle_force Fs2))))
  (L2 = L1 * 9.800000E-01 )
  (T3 = T2 * 9.800000E-01 )))
```

図8 solveの階層化の例

5. おわりに

筆者らが開発した設計記述言語ADLを用いて歯車減速機の設計プログラムを試作した。本プログラムは、本文で述べた種々のプログラミング手法を用いることによって、多目的性、モジュール性の高いプログラムとなることを確認できた。また効率面でも一応の成果を得ている。今後は、グラフィック機能や対話機能等を拡張し、より総合

的な設計支援システムを開発し機械設計の教育に試用してゆく予定である。なお本システムは、九大大型計算機センター及び情報処理教育センターで稼動している。

〔参考文献〕

1）小川　潔：機械設計システムのプログラミング，P.206，森北出版（1977）.

2）小川　潔：機械設計システム，P.240，森北出版（1973）.

3）須藤敏男：機械設計(9)歯車減速機の設計製図，P.259，パワー社(1960).

4）長澤，古川，荒牧：Prologを基礎とした設計システム記述言語ADL，本研究会資料（1983）.