Memoirs of the Faculty of Engineering, Okayama University, Vol.27, No. 2, pp. 1 - 9 , March 1993

An Expert System for Determining Precedence Relation in Assembly

Yasuhiro KAJIHARA* and Hirokazu OSAKI*

(Received January 29 , 1993)

SYNOPSIS

Precedence relation in assembly has been determined by experience only. Now, an expert system is developed for determining such precedence relation. The conjugate states of the units of a product are shown in face frames and unit frames following the frame model. Seven rules are formulated. They consider geometrical interference of units, and make the precedence relation enable operation time to be shorter, and the number of JIG and substandards to be smaller. These rules are programized by computer language(PROLOG).

1. INTRODUCTION

Designing of assembly lines is frequently needed in many production systems and for the designing of assembly lines precedence relation of assembly tasks is needed.[1] However, the precedence relation has been determined by experience of process designers. They determine the precedence relation in consideration of many factors such as geometrical interference among parts (overlapping), tools, fastening methods of parts.

We apply the method of knowledge engineering to express experience of the process designers in seven rules. An expert system is developed in order to support the process designers in the determination of precedence relation, which is determined after considering several factors, such as reduction of tools, JIGs, assembly time and substandards.

2. REPRESENTATION OF ASSEMBLY TASKS

A product is composed of units. A unit means a part group that was fastened by one or more fastening methods (screw, insertion, bolt nut etc.).

* Department of Mechanical Engineering

(a) Assembly face

In a drawing, each face of a product is represented in 6 symbols that are shown in Fig. 1 by JIS.[2] Hereafter, these 6 faces are called assembly faces.

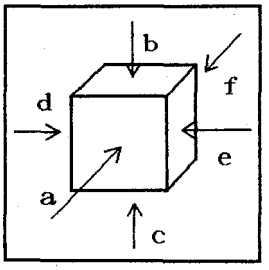

Fig.1 Assembly faces

(b) Representation of assembly tasks

A product is taken apart by removing a unit one by one. Whenever a unit is removed, fastening condition (scene) of new units are seen. The frame model has been proposed to express a scene like the fastening condition of a unit.[3]

We use the frame model in order to express each scene, and we name each scene a face frame. Also, a unit frame is used to express characteristics of each unit. Each frame is represented in the next symbol.

(i)Face frame

FACE(N_i,F_i,UN_i,U_i,B_i) : Face frame N_i has an upper class face frame UN_i. Unit U_i is fastened to a base part B_i on assembly face F_i.

(ii)Unit frame

UNIT(U_i,P_i,M_i,H_i,A_i) : Unit U_i is composed of parts P_i, and fastened to a base part by fastening method M_i. Height of U_i on a base part is H_i. If unit U_i moves (slides or rotates) A_i ='Y' is made. A_i='N', if unit U_i is fixed. A face frame represents task needed for the assembly of a unit. Therefore, we represent an assembly task by the face frame.

2.3 Rules for determining precedence relation

Rules are classified into two groups. One considers overlapping of face frames. The other considers particular goals such as shorter operation time, and a smaller number of JIG and substandards. There are three rules that consider overlapping of face frames.

(i) Rule for overlapping parts(R1)

This rule determines the precedence relation between two face frames when one face frame covers the other. The face frame of a unit U_i is defined by FACE(N_i,F_i,UN_i,U_i,B_i). The face frame of a unit U_j is defined by FACE(N_j,F_j,UN_j,U_j,B_j). When the face frame N_j is included in the upper frame UN_i of the face frame N_i ($N_j \in UN_i$), then the face frame N_j covers a part or all of the face frame N_i. Therefore, the face frame N_i is made the predecessor of the face frame N_j. We represent this precedence relation by a symbol PRECEDE(N_i,N_j). This rule is represented by eq.(1).

$$\forall N_i \forall N_j[\mathrm{FACE}(N_i,F_i,UN_i,U_i,B_i) \wedge \mathrm{FACE}(N_j,F_j,UN_j,U_j,B_j) \wedge (N_j \in UN_i) \rightarrow (\mathrm{put\ PRECEDE}(N_i,N_j))] \quad (1)$$

Where, a symbol (put A) means to set A.

(ii)Rule for the higher part (R2)

When two units which differ in height are fastened to the same base part, the higher unit may impede operation of a robot in fastening the lower unit. Therefore, the lower unit is made the predecessor of the higher unit. The face frame N_i is defined by $FACE(N_i,F_i,UN_i,U_i,B_i)$ and the face frame N_j is defined by $FACE(N_j,F_j,UN_j,U_j,B_j)$.

If the unit U_i and U_j are fastened to the same base part($B_i=B_j$) and U_i is higher than U_j, then the face frame N_i is made the predecessor of the face frame N_j. Otherwise the face frame N_j is made the predecessor of N_i. This rule is represented by eq.(2).

$$\forall N_i \forall N_j[FACE(N_i,F_i,UN_i,U_i,B_i) \wedge FACE(N_j,F_j,UN_j,U_j,B_j) \wedge (B_i=B_j) \wedge (N_j \notin UN_i) \wedge (N_i \notin UN_j) \wedge UNIT(U_i,P_i,M_i,H_i,A_i) \wedge UNIT(U_j,P_j,M_j,H_j,A_j) \wedge ((H_i<H_j) \rightarrow (\text{put } PRECEDE(N_i,N_j)); (\text{put } PRECEDE(N_j,N_i)))] \quad (2)$$

Here, a symbol (A → B;C) means that B is implemented if A is true. Otherwise C is implemented.

(iii)Rule for assembly faces (R3)

This rule gives precedence relation to a unit which is fastened to several assembly faces of a base part. A unit, which is fastened to several assembly faces of the base part, holds the following equation.

$$\exists N_i \exists N_j[FACE(N_i,F_i,UN_i,U_i,B_i) \wedge FACE(N_j,F_j,UN_j,U_j,B_j) \wedge (U_i=U_j) \wedge (F_i \neq F_j)]$$

This unit is inserted in the base part first. After that the unit is fastened by other fastening methods such as screws or bolts. Therefore, a face frame with the fastening method of insertion is made the predecessor. When, a fastening method of insertion is represented by a symbol 'ins', then this rule is represented by eq.(3).

$$\forall N_i \forall N_j[FACE(N_i,F_i,UN_i,U_i,B_i) \wedge FACE(N_j,F_j,UN_j,U_j,B_j) \wedge (U_i=U_j) \wedge (F_i \neq F_j) \wedge UNIT(U_i,P_i,M_i,H_i,A_i) \wedge UNIT(U_j,P_j,M_j,H_j,A_j) \wedge ((M_i=\text{'ins'}) \rightarrow (\text{put } PRECEDE(N_i,N_j)); (\text{put } PRECEDE(N_j,N_i)))] \quad (3)$$

Now, we formulate three rules that are used to attain particular goals in assembly.

(iv) Rule for movable parts(R4)

When a unit moves (rotates or slides) on the base part and the fastening position of a unit is not proper, products of inferior quality occur. This rule is used to decrease products of inferior quality caused by aberration of a position.

A base part Bi of a face frame N_i is defined by $FACE(N_i,F_i,UN_i,U_i,B_i)$. A base part B_j of a face frame N_j is defined by $FACE(N_j,F_j,UN_j,U_j,B_j)$. The part set P_j of the unit U_j is defined by the unit frame $UNIT(U_j,P_j,M_j,H_j,A_j)$. When the face frame Ni is fastened to Nj, ($B_i \in P_j$) is held. If unit U_i is movable, (A_j='Y') is held. In this case, fastening position of both units is made correct by making the face frame N_i the predecessor of the unit frame N_j. This rule is represented by eq.(4).

$$\forall N_i \forall N_j[\ FACE(N_i,F_i,UN_i,U_i,B_i) \wedge FACE(N_j,F_j,UN_j,U_j,B_j) \wedge UNIT(U_j,P_j,M_j,H_j,A_j) \wedge (B_i \in P_j) \wedge (A_j = 'Y') \rightarrow (\text{put } PRECEDE(N_i,N_j))] \quad (4)$$

(v) Rule for frames having the same fastening method (R5)

If the fastening method of units differ, tools are changed. This rule is used to decrease the number of changing tools.

An fastening method M_i of a face frame N_i is defined by $UNIT(U_i,P_i,M_i,H_i,A_i)$ and M_j of a face frame N_j is defined by $UNIT(U_j,P_j,M_j,H_j,A_j)$. When M_i and M_j are the same, the face frame N_i and N_j are assembled consecutively. This rule is represented by eq.(5).

$$\forall N_i \forall N_j[\ UNIT(U_i,P_i,M_i,H_i,A_i) \wedge UNIT(U_j,P_j,M_j,H_j,A_j) \wedge (M_i=M_j) \rightarrow (\text{put } CONSECT(N_i,N_j))] \quad (5)$$

Here, the symbol CONSECT(A,B) means that both face frames A and B are fastened consecutively.

(vi) Rule for assembly faces (R6)

If several units are fastened to different assembly faces of the same base part, the attitude of the base part is changed. This rule is used to decrease the number of the attitude change of the base part.

A base part of a unit U_i is defined by $FACE(N_i,F_i,UN_i,U_i,B_i)$ and a base frame of a unit U_j is defined by $FACE(N_j,F_j,UN_j,U_j,B_j)$. When the base parts and the assembly faces of the unit U_i and U_j are the same($B_i=B_j$, $F_i=F_j$), then both units are assembled consecutively. This rule is represented by eq.(6).

$$\forall N_i \forall N_j[\ FACE(N_i,F_i,UN_i,U_i,B_i) \wedge FACE(N_j,F_j,UN_j,U_j,B_j) \wedge (B_i=B_j) \wedge (F_i=F_j) \rightarrow (\text{put } CONSECT(N_i,N_j))] \quad (6)$$

(vii)Rule for consecutive face frames (R7)

When the two units hold a condition of $CONSECT(U_i,U_j)$, then the predecessors and followers of both units should be the same. The predecessor of the unit N_i is defined by $PRECEDE(N_a,N_i)$, and followers are defined by $PRECEDE(N_i,N_b)$. Therefore, the predecessor and follower of the unit U_i are made the predecessor and follower of the unit U_j. The same procedure is applied to the unit Uj. This rule is represented by eq.(7).

$$\forall N_i \forall N_j[\ CONSECT(N_i,N_j) \wedge PRECEDE(N_a,N_i) \rightarrow (\text{put } PRECEDE(N_a,N_j))]$$
$$\forall N_i \forall N_j[\ CONSECT(N_i,N_j) \wedge PRECEDE(N_i,N_b) \rightarrow (\text{put } PRECEDE(N_j,N_b))] \quad (7)$$

2.4 The inference mechanism

An inference mechanism is necessary to determine precedence relation by using the above rules. Rules R1, R2 and R3 are applied first, because they can be applied to any assembly products. But there is no precedence relation among these rules. Rules R4, R5 and R6 are applied depending on objects.

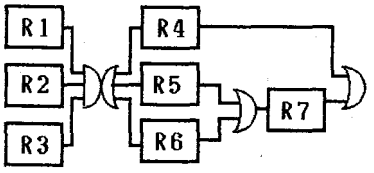

Fig.2 The inference mechanism

If quality improvement is aimed, R4 is applied. If reduction of the cost of equipments is aimed, R5 is applied. If reduction of work time is aimed, R6 is applied. Rules R4, R5 and R6 may be combined and applied. R7 is applied after either R5 or R6 is applied. This procedure is shown in Fig.2 with reasoning symbols.

3. PROGRAM

In this method, rules for determination of precedence relation are written in predicate logic. This system is coded in the computer language PROLOG that is able to use a predicate logic formula as a program. This system is composed of a data base representing assembly tasks by using frame models, knowledge base of rules, and inference mechanism that controls the application order of each rule.

The program list is shown in Table 1. In the program, following arguments are used.

3.1 Argument list	Definition
face(N,F,UN,U,B)	:N(Name of face frame), U(Unit), F(Assembly face), B(Base part), M(Fastening method)
unit(U,P,M,H,A)	:U(Unit), P(Set of parts), M(Fastening method), H(Height), A('y';Moving, 'n';fix)
main	: Rule for start of the inference
a(r1)	: Rule for overlapping parts(R1)
a(r2)	: Rule for the higher part(R2)
a(r3)	: Rule for assembly faces(R3)
a(r4)	: Rule for movable parts(R4)
a(r5)	: Rule for frames having the same fastening method(R5)
a(r6)	: Rule for assembly faces (R6)
a(r7)	: Rule for consecutive face frames (R7)

3.2 Application example

This system was applied to determine precedence relation of an engine part (*carburetor*).

(1) Representation of assembly tasks

An assembly product (carburetor) was composed of 10 units which are shown in Fig.3. All units were removed one by one. Units U7, U8, U9 were removed from assembly face b. Units U7 and U10 were removed from assembly face c. Units U1, U6 and U9 were removed from assembly face d. Units U2,U3, U4, U5, U6, U9 were removed from assembly face e. The unit U9 was fastened to the 3 assembly faces. The units U6 and U7 were fastened to the 2 assembly faces. Accordingly, the assembly tasks of all units were represented by 14 face frames. An example of face frames is shown in Fig.4.

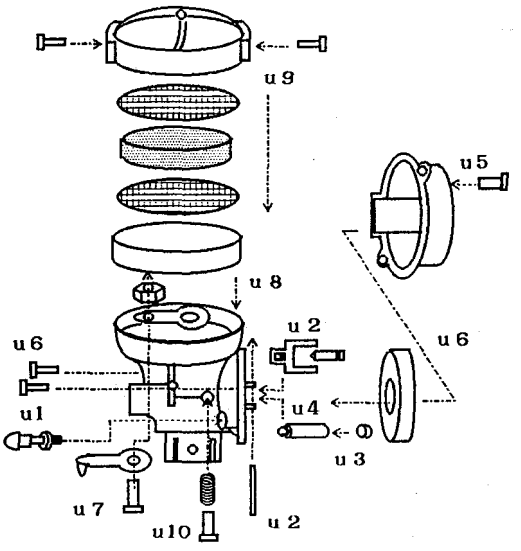

Fig.3 The sssembly product

(2) Determination of precedence relation

First, rules R1, R2 and R3 were applied, and precedence relation was determined as shown in Fig.5. Fig.5 means that units U7, U8 and U9 are assembled consecutively. In the assembly face c, units U7 and U10 have no predecessor. In the assembly face e, units U2, U3 and U5 have no predecessor.

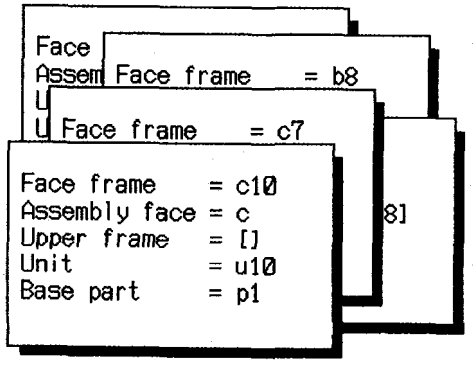

Fig.4 Face frames

Next, rules R1, R2, R3 and R5 were applied to reduce total assembly time and installation cost of equipments. Furthermore, rule R7 was applied, and the precedence relation was modified as shown in Fig.6. The number of face frames which are assembled in parallel decreased. The precedence relation which was led by all the rules(R1～R7)

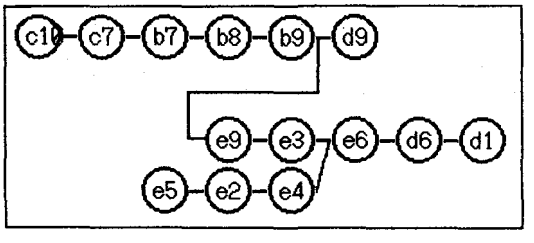

Fig.5 Precedence relation by R1, R2 and R3

became the same as Fig.6.

Three workers (male students) determined precedence relation of the assembly product after practices of the assembly. Two workers determined the same precedence relation in Fig.5. The other determined the same precedence relation in Fig.6. It is conceivable that the first two workers considered geometrical interference among units, and the other worker considered reduction of assembly time.

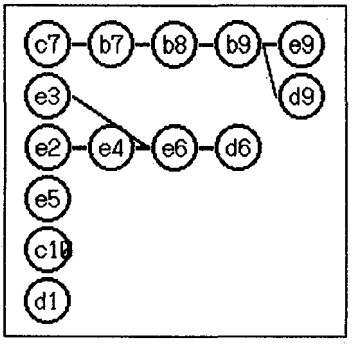

Fig.6 Precedence relation by R1～R3, R5 and R7

This result means that different precedence relation is determined depending on the factors which are considered in production management. This system enables us to simulate various precedence relation and helps to clarify the aims of process designers.

4. CONCLUSION

An expert system for determining precedence relation in assembly is developed in this research. This system has following characteristics.

(1)Tasks needed for assembly of a product are represented in the face frames and unit frames following the frame model.

(2)Seven rules are formulated considering geometrical interference of units, and particular objects such as reduction of assembly time and installation cost.

(3)Precedence relation of an engine part was determined by this system. It was shown that the precedence relation differs depending on the intended management objects.

REFERENCES

(1) Dar-EL.E.M., MALB-A Heuristic Technique for Balancing Large Single-Model Assembly Lines, AIIE trans. pp.343-355(1973)

(2) Editing comittee of Basis Drawing Manual, "JIS kihon seizu manual", Japan Standard association,p111(1984)

(3) Winston P.H., The Psychology of Computer Vision, McGraw-Hill,(1975), Y. Shirai(trans. in Japanese), Sangyo Toshyo, pp.237-261(1987)

```
/* ------------------------------------------
   Table 1 Program of an expert system for
           determining precedence relation
           System will start by "main."
  ------------------------------------------ */

main:-w_ini,g_cls,d_clear,repeat,
       w_decision2(w0,c,[kansu,2,1,12],[L]),
       a(L),L=end.

a(end).
kansu(frames,3).      kansu(r1,5).
kansu(r2,6).          kansu(r3,7).
kansu(r4,8).          kansu(r5,9).
kansu(r6,11).         kansu(r7,12).
kansu(display,10).    kansu(end,13).
ans(yes,1).           ans(no,2).

/* Arrow diagram of precedence relation*/
make_set:-senkou_set(X,Y),
       retract(senkou_set(X,Y)),fail.
make_set:-face(N,F,UU,U,B),s(N),fail.
s(N):-set(N,[]),!.
make_set:-listing(senkou_set).
set(N,L):-preced(X,N),
       ¥+(member(X,L)),
       set(N,[X|L]).
set(N,L):- assert(senkou_set(N,L)),!.
senkou_main:-make_set,frame(L),for(I,1,10),
       hyouji_senkou(I,L),fail,!.
senkou_main:- for(I,1,10),nl,write(I),
       iti_senkou(I,N,K),
       write(' '),write(N),
       write('  '),write(K),fail.
senkou_main:-d_clear,gurafu_senkou,!.
q(X,U):-(X='1',U is 1;  X='2',U is 2;
       X='3',U is 3;   X='4',U is 4;
       X='5',U is 5;   X='6',U is 6;
       X='7',U is 7;   X='8',U is 8;
       X='9',U is 9;   X='10',U is 10),!.

a(display):-repeat,g_cls,d_clear,
       gurafu_senkou,d_cursor(4,0),
       write('(X)   1    2    3    4    5    6
  7    8    9    10 '),
       d_cursor(6,3),write(1),
       d_cursor(6,5),write(2),
       d_cursor(6,7),write(3),
       d_cursor(0,7),write('(Y)'),
       d_cursor(6,9),write(4),
       d_cursor(6,11),write(5),
       d_cursor(6,13),write(6),
       w_open(w6),
       w_cursor(0,0),w_write('Frame='),
       w_read(N),w_nl,
       w_write('X     ='),w_read(X),w_nl,
       w_write('Y     ='),w_read(Y),w_nl,
       q(X,X1),q(Y,Y1),
       iti_senkou(I,N,K),
       retract(iti_senkou(I,N,K)),
       asserta(iti_senkou(X1,N,Y1)),
       w_write('continue? '),w_read(Ans),
       Ans=n.

hyouji_senkou(1,L):- bango_new,
       member(N,L),senkou_set(N,[]),
       iti_senkou(1,N),
       fail.
hyouji_senkou(X,L):- X>1,
       bango_new,member(N,L),
       senkou_set(N,S),¥+(S=[]),
       member(E,S),X1 is X-1,
       iti_senkou(X1,E,K1),
       iti_senkou(X,N),
       fail.
iti_senkou(I,N):- ¥+(iti_senkou(X,N,Y)),
       bango(K),retract(bango(K)),
       K1 is K+1,assert(bango(K1)),
       assert(iti_senkou(I,N,K1)),!.
gurafu_senkou:-d_clear,
       iti_senkou(I,N,K),
       X is I*5+4,Y is K*2+1,
       d_cursor(X,Y),
       X1 is X*8,Y1 is Y*16,write(N),
       g_circle(X1,Y1-8,14,14,7),
       g_circle(X1,Y1-8,13,13,7),fail.
gurafu_senkou:- iti_senkou(I,N,K),
       senkou_set(N,L),¥+(L=[]),
       member(E,L),
       iti_senkou(I1,E,KE),
       XE is (I1*5+4)*8,
       YE is (KE*2+1)*16-8,
       X is (I*5+4)*8,
       Y is (K*2+1)*16-8,
       g_line(X-16,Y,XE+16,YE,7,0),
       g_line(X-16,Y+1,XE+16,YE+1,7,0),
       fail.
gurafu_senkou:-d_cursor(0,0),!.

for(I,I,M):-I=<M.
for(I,N,M):- N<M,N1 is N+1,for(I,N1,M).

bango(0).
bango_new:-retract(bango(_)),assert(bango(0)),!.

/* windows */
w_table(w0,[10,1,50,18]).
w_table(w1,[20,1,60,13]).
w_table(w2,[25,3,30,4]).
w_table(w3,[25,7,30,10]).
w_table(w4,[25,3,30,4]).
w_table(w5,[25,7,30,7]).
w_table(w6,[10,15,25,7]).

w_ini:-ttyget(R),mouse(off),g_cls,d_clear,
       g_screen(3,0,0,1),d_scroll(_,24),
```

```
        mouse(on).
position(X,Y):-repeat,mouse(A,B,X,Y),B = -1,!.

a(frames):- g_cls,w_color(1,2,5),
        face(N,F,UU,U,B),
        position(X1,Y1),
        X is X1/8,Y is Y1/16,
        asserta(w_table(N,[X,Y,25,8])),
        w_open(N),
        w_write('Face frame   = '),
        w_write(N),w_nl,
        w_write('Assembly face= '),
        w_write(F),w_nl,
        w_write('Upper frame  = '),
        w_write(UU),w_nl,
        w_write('Unit         = '),
        w_write(U),w_nl,
        w_write('Base part    = '),
        w_write(B),w_nl,
        fail.
member(E,[E|L]).  member(E,[X|L]):-member(E,L).

a(r1):-face(N1,F1,UU1,U1,B1),
        face(N2,F2,UU2,U2,B2),
        ¥+(N1=N2),member(N2,UU1),
        ¥+preced(N1,N2),¥+preced(N2,N1),
        assert(preced(N1,N2)),fail.
a(r1):-listing(preced).

a(r2):-face(N1,F1,UU1,U1,B1),
        face(N2,F2,UU2,U2,B2),
        ¥+(N1=N2),B1=B2,
        member(N2,UU1),
        ¥+member(N1,UU2),
        unit(N1,F1,M1,H1,A1),
        unit(N2,F2,M2,H2,A2),
        ¥+preced(N1,N2),¥+preced(N2,N1),
        (H1>H2 -> assert(preced(N2,N1));
                  assert(preced(N1,N2))),
        fail.
a(r2):-listing(preced).

a(r3):-face(N1,F1,UU1,U1,B1),
        face(N2,F2,UU2,U2,B2),
        ¥+(N1=N2), U1=U2, ¥+(F1=F2),
        unit(N1,P1,M1,H1,A1),
        unit(N2,P2,M2,H2,A2),
        ¥+preced(N1,N2),¥+preced(N2,N1),
        (M1=ins,assert(preced(N1,N2));
        ¥+(M1=ins), ¥+(M2=ins), true;
        M2=ins,assert(preced(N2,N1)) ),
        ¥+consect(N1,N2),¥+consect(N2,N1),
        assert(consect(N1,N2)),
        fail.
int_ck:-retract(ck(_)),fail.
int_ck:-assert(ck(0)).
a(r3).

a(r4):-face(N1,F1,UU1,U1,B1),
        face(N2,F2,UU2,U2,B2),
        ¥+(N1=N2),F1=F2,
        unit(N2,P2,M2,H2,A2),A2=y,
        member(B1,P2),
        ¥+preced(N1,N2),¥+preced(N2,N1),
        asserta(preced(N1,N2)),
        fail.
a(r4).

a(r5):- unit(N1,P1,M1,H1,A1),
        unit(N2,P2,M2,H2,A2),
        ¥+(N1=N2),M1=M2,
        ¥+consect(N1,N2),
        ¥+consect(N2,N1),
        asserta(consect(N1,N2)),
        fail.
a(r5).

a(r6):-face(N1,F1,UU1,U1,B1),
        face(N2,F2,UU2,U2,B2),
        ¥+(N1=N2),F1=F2,
        ¥+preced(N1,NB),¥+preced(NC,N2),
        asserta(consect(N1,N2)),fail.
a(r6).

a(r7):- consect(N1,N2),
        ¥+preced(N1,NB),¥+preced(NC,N2),
        assert(preced(N1,N2)),fail.
a(r7):- consect(N2,N1),
        ¥+preced(N1,NB),¥+preced(NC,N2),
        assert(preced(N1,N2)),fail.

/* Face frames */
face(d1,d,[],u1,p1).     face(e2,e,[e4],u2,p1).
face(e3,e,[e6],u3,p19).  face(e4,e,[e6],u4,p1).
face(e5,e,[],u5,p22).    face(e6,e,[],u6,p1).
face(d6,d,[],u6,p1).     face(b7,b,[b8],u7,p1).
face(c7,c,[],u7,p1).     face(b8,b,[b9],u8,p1).
face(b9,b,[],u9,p4).     face(d9,d,[],u9,p4).
face(e9,e,[],u9,p4).     face(c10,c,[],u10,p1).

/* Unit farmes */
unit(e6,[p22,p21,p1],ins,40,n).
unit(d6,[p15,p1],screw,40,n).
unit(c7,[p11,p12,p1],bolt,35,n).
unit(b7,[p3,p2,p1],bolt,40,n).
unit(b8,[p4,p5],screw,58,n).
unit(e9,[p10,p4],screw,58,n).
unit(d9,[p10,p4],screw,58,n).
unit(b9,[p8,p9,p7,p6,p4],ins,60,n).
unit(e5,[p23,p24,p22],bolt,40,n).
unit(e3,[p19,p20],bolt,40,n).
unit(e4,[p19,p1],bolt,40,n).
unit(e2,[p17,p18,p1],ins,40,n).
unit(d1,[p15,p1],screw,40,n).
unit(c10,[p14,p13,p1],screw,45,n).
```