Memoirs of the Faculty of Engineering, Okayama University, Vol.24, No. 1, November 1989

An Expert System for the Scheduling of a Flexible Assembly Line for Multi Item Products

Yasuhiro KAJIHARA* and Hirokazu OSAKI**

(Received October 16, 1989)

SYNOPSIS

An expert system, in which preconditions and rules are expressed in logical formulas, is developed to support the scheduling of an automated job shop type multi-item assembly line. This system has the following characteristics to apply any case of schedulings: (1)Forward scheduling orbackward scheduling can be made. (2)The criterion on the input order of products, the dispatching process at each assembly station, and the selection of products from a buffer can be selected from several priority criteria. (3)Layout, number and velocity of vehicles, and the capacity of each buffer can be changed.

1. INTRODUCTION

Flexible automation in today's assembly lines has been increased to cope with the diversity of products and redesigns. Therefore, the role of experts who schedule the automated assembly lines have became very important[1].

We developed an expert system to support these experts. In this system, the knowledges for the flexible assembly line scheduling are classified into facts, relations among facts, rules, and inference mechanism in order to allow the experts to alternate these knowledges with

* Graduate school of natural science and technology
** Department of mechanical engineering

ease. These rules are represented in first-order predicate logic[2], and coded with Prolog language.[3]

2. THE COMPONENTS OF THE SYSTEM

The knowledges for the scheduling of an flexible assembly line can be classified into facts, relationships, rules and an inference mechanism. Facts mean basic elements comprising an assembly line. Relationships mean relations among basic elements. The rules infere relations. The inference mechanism determines how to use the rules to make the schedule of the flexible assembly line.

2.1 The fact and notation

It is assumed that the assembly line is composed of the following five basic elements:

(1)Product

g_i :i th product $(i=1,\cdots,k)$

es_i :The earliest operation start time of the i th product

d_i :Due date of the i th product

f_{il} :l th functional unit of i th product $(l=1,\cdots,s_i)$
The function of a part is changed by fastening it to other parts. A set of parts connected by one fastening method is called a 'functional unit.'[4]

S_i :The number of functional units of the i th product

(2)Station

I/O :Input/output station

m_j :j th assembly station $(j=1,\cdots,nm)$

p_j :Location of the m_j assembly station

sf_j :The function of the m_j assembly station
(screw, welding, insert etc..)

(3)Buffer

b_j :The buffer at the m_j assembly station

q_j :The capacity of the b_j buffer

(4)Path

p_l :l th block of the path $(l=1,\cdots,n)$
The path among assembly stations is divided in blocks, and they are serially numbered

(5)Vehicle

c_v :v th vehicle $(v=1,\cdots,nv)$
The direction of the vehicle is predetermined

2.2 Relation

To schedule an assembly line design, five following relations are needed, which are induced and changed dinamically by rules except for the precedence relations of functional units.

(1) Precedence relation among functional units

SP_{il} :The set of immediate predecessors of f_{il}

FP_{il} :The set of immediate followers of f_{il}

(2)The relation between a functional unit and an assembly station

op_{il} :The assembly station which operates the functional unit f_{il}

et_{il} :Time required for the operation of the functional unit f_{il}

(3) Operation

The operation means the relation between a product and an assembly station.

$on(m_j, g_i)$:j th assembly station is operating the i th product at this time

ft_j :The planned finishing time of the operation of the j th assembly station

$wait(m_j, g_i)$:An operation for the i th product by the j th assembly station has been finished

(4)Queue

b_j :Number of products at the j th buffer

qt_j :Remaining operation time which is the sum of the operation time of products waiting at the j th buffer

(5)Movement

$f(c_v, g_i, b_l, p_j)$:The vehicle c_v is at the block b_l on the path and carrying a product g_i to the assembly station m_j

2.3 Rules

The rules for the assembly line scheduling are classified according to basic elements into three groupes, that is, rules related to products, vehicles, and assembly stations.

2.3.1 Rules related to products.

To scedule, three procedures are required in regard to the products: (1)Determination of the input order of products. (2)Determination of the assembly sequence of functional units. (3)Selection of a vehicle which moves a product. These procedures are formulated as follows.

(1)Input rule

The first rule is the input rule which determines the input order of products. If elapsed time is 't' time unit, and there is no waiting product at the I/O station, represented as $wait(I/O, \phi)$, then select one product (g_i) from the products which are arranged in a row at the I/O station. This procedure is expressed in the equation (1).

$$(R1)\ \exists g_i\ \exists t[wait(I/O,\phi)\wedge s1(K1,g_i)\wedge(es_i<t) \rightarrow (put\ st_i\ as\ t)\wedge(put\ wait(I/O,\phi)\ as\ wait(I/O,g_i))] \quad (1)$$

Where K1 is the number of the priority rule[5] which is selected from following four priority rules (DD, HD, SO, LO), and $s1(K1,g_i)$ represents a fact that the product g_i is chosen by a priority rule K1.

(i)DD : selects a product with the earliest due date, if (K1=1).

(ii)HO : selects a product with the highest total operation time, if (K1=2).

(iii)SO : selects an assembly station that has the smallest operation ratio, and selects the product with the highest operation time for the selected assembly station, if (K1=3).

(iv)LO : selects an assembly station that has the lowest remaining operation time, and selects the product with the highest operation time for the selected assembly station, if (K1=4).

(2)Dispatching rule

When a functional unit (f_{il}) of the product (g_i) is finished at an assembly station (m_j) or selected by the input rule, then a next functional unit(f_{il}) is selected by the dispatching rule.

The next functional unit should satisfy the following conditions :

(a)If forward scheduling is used, the next functional unit f_{il} of product g_i should satisfy $(SP_{il} \cap FL_i = SP_{il})$.[3]

(b)If backward scheduling is used, then $(FP_{il} \cap FL_i = FP_{il})$ should be true.

If the next functional unit f_{il} which will be operated by the assembly station(m_{j+1}) is expressed as $go(g_i, m_{j+1}, f_{il})$, then this procedure is expressed in equation (2).

$$(R2)\ \exists m_j\ \exists m_{j+1}\ \exists g_i[wait(m_j,g_i)\wedge (q_{j+1}<q^{*}_{j+1})\wedge s2(K2_j,f_{il})\wedge(op_{il}=m_{j+1}) \rightarrow put\ go(m_j,g_i,f_{il-1})\ as\ go(m_{j+1},g_i,f_{il})] \quad (2)$$

Where $K2_j$ is the number of the priority rule which is selected from following three priority rules (NS, HO, LO), and $s2(K2_j,f_{il})$ represents a fact that the functional unit f_{il} is chosen by the priority rule(NS, HO or LO).

(i) NS : Selects a functional unit which is operated by the nearest assembly station, if $(K2_j=1)$.

(ii) HO : Selects the functional unit with the highest operation time, if $(K2_j=2)$.

(iii)LO : Selects an assembly station that has the lowest remaining operation time, and selects the functional unit with the highest operation time for the selected assembly station, if $(K2_j=3)$.

(3)The rule for summoning a vehicle

When an operation for product(g_i) is finished at any assembly station(m_j), a vehicle $(c_v)$is summoned to move the product. In this case, we select a vehicle (c_v) arriving at any assembly station(m_{j+1}) and change its destination to m_j. This procedure is expressed in equation (3).

$$\text{(R3)}\quad \exists c_v \exists g_i[\text{wait}(m_j, g_i) \wedge f(cv, \phi, p_{j+1}, p_{j+1}) \rightarrow \text{put } f(cv, \phi, p_{j+1}, p_{j+1}) \text{ as } f(cv, \phi, p_{j+1}, p_j)] \quad (3)$$

where, p_j is the location of the assembly station m_j.

2.3.2 Rules related to the vehicles

Three rules are necessary to control the vehicles:

(1)The rule for movement of vehicles

When a vehicle is at block b_l, this vehicle is moved to the next block b_{l+1} after one time unit has passed.

This procedure is expressed in equation (4).

$$\text{(R4)}\quad \forall c_v[f(c_v, X, b_l, p_j) \wedge (X=\phi \vee X=g_i) \wedge (b_l \neq p_j) \rightarrow \text{put } f(c_v, X, b_l, p_j) \text{ as } f(c_v, g_i, b_{l+1}, p_j)] \quad (4)$$

(2)The rule for vehicles loading

When a vehicle(c_v) has been summoned and arriving at an assembly station (m_j), then a product(g_i) at the assembly station(m_j) is moved onto the vehicle(c_v). This procedure is expressed in equation (5).

$$\text{(R5)}\quad \exists g_i \exists c_v[f(c_v, \phi, p_j, p_j) \wedge \text{wait}(m_j, g_i) \wedge \text{go}(g_i, m_{j+1}, f_{il}) \rightarrow \text{put } f(c_v, \phi, p_j, p_j) \text{ as } f(c_v, g_i, p_j, p_{j+1})] \quad (5)$$

(3)The rule for vehicles unloading.

When a vehicle(c_v) has been carrying a product(g_i) and arriving at an assembly station(m_j), the product is moved from the vehicle to a location which is empty at the buffer of the assembly station. This procedure is expressed in equation (6).

$$\text{(R6)}\quad \exists c_v[f(c_v, g_i, p_j, p_j) \wedge t \wedge bu_{jr}(\phi, 0) \rightarrow (\text{put } f(c_v, g_i, p_j, p_j) \text{ as } f(c_v, \phi, p_j, p_j)) \wedge (\text{put } bu_{jr}(\phi, 0) \text{ as } bu_{jr}(g_i, t))] \quad (6)$$

Where, if a product g_i is put on the 'r' th place at b_j at elapsed time 't', then this fact is denoted as to be $bu_{jr}(g_i, t)$. If the 'r' th place is empty, then this fact is denoted as to be $bu_{jr}(\phi, 0)$.

2.3.3 Rules related to the assembly stations

Two rules are necessary to contorol the assembly stations:

(1)The rule for operation of the assembly station

If an assembly station(m_j) is idle and there is a queue at its buffer, then a product(g_i) is selected from the queue by a selection rule(FI, DD, LO or HO) and moved to the assembly station(m_j). This procedure is expressed in equation (7).

$$\text{(R7)}\ \exists g_i \exists m_j[on(m_j,\phi)\wedge s3(K3_j,g_i)\wedge bu_{jr}(g_i,at_i)\wedge go(g_i,m_j,f_{ii}) \rightarrow (\text{put } on(m_j,\phi) \text{ as } on(m_j,g_i))\wedge(\text{put } ft_j \text{ as } t+et_{ii})\wedge (\text{put } bu_{jr}(g_i,at_i) \text{ as } bu_{jr}(\phi,0))] \quad (7)$$

Where, $K3_j$ is the number of the priority rule for the products at the buffer. Four proiruty rules (FI, DD, LO, and HO) are as follows:

(i) FI : Selects the first incoming product, if ($K3_j$=1).

(ii) DD : Selects the product with shortest due date, if ($K3_j$=2).

(iii)LO : Selects the product with lowest operation time for the assembly station, if ($K3_j$=3).

(iv)HO : Selects the product with highest operation time for the assembly station, if ($K3_j$=4).

(2)The rule to stop the operation at the assembly station

If an assembly station(m_j) is operating a product(g_i), and a planned finishing time becomes equal to an elapsed time(t), then the operation is finished and the relation between m_j and g_i is denoted as to be wait(m_j,g_i). The assembly station then becomes idle and the state of m_j is denoted as to be on(m_j, ϕ). This procedure is expressed in equation(8).

$$\text{(R8)}\ \exists t \exists m_j[on(m_j,g_i)\wedge(t=ft_j) \rightarrow (\text{put } wait(m_j,\phi) \text{ as } wait(m_j,g_i))\wedge (\text{put } on(m_j,g_i) \text{ as } on(m_j,\phi))] \quad (8)$$

2.4 The inference mechanism

To schedule an assembly line, it is necessary to construct a inference mechanism that determines how to use above mentioned eight rules. Three procedures are needed for the scheduling:

(a)If there is a product g_i which satisfy the conditions of the imput rule(R1) or the rule to stop the operation of the station(R8), then it is needed to select a next functional unit by the dispatching rule (R2) and to summon a vehicle by the rule for summoning avehicle(R3). This procedure is expressed in the equation $\{(R1 \vee R8) \wedge R2 \wedge R3\}$.

(b)If a vehicle is on the path, the vehicle is moved by one brock by the rule for movement of vehickes(R4). When a vehicle is just arriviug at any station, the loading for the vehicle or unloading is done by the rule(R5, R6). There is no procedence relation ammong these three reles, that is $(R4 \vee R5 \vee R6)$.

(c)If a product is loading on a vehicle at the station m_j by the rule R5, then it is needed to move a product to the station from it buffer by the rule R7.

These procedure is done by every unit time. Therefore, the inference mechanism is expressed in eq(9).

$$\forall t[f(t) \rightarrow ((R1 \vee R8) \wedge R2 \wedge R3 \vee s) \wedge (R4 \vee R5 \vee R6 \vee s) \wedge ((R5 \wedge R7) \vee s) \wedge f(t+1)] \quad (9)$$

Where, 's' is a rule which will success at every time.

3. PROGRAM

Facts, relations and rules for the scheduling of an assembly station are coded in computer language 'Prolog'. Program list is shown in Table 1-(a). In the program, following arguments are used.

3.1 Argument List

Argument List	Definition
due_date(K,D)	K : the product type
	D : the due date of the product type K
order(K,N)	N : the number of order
ope_time(K,[(F,M,T),…]).	F : functional unit
	M : assembly station which operates the functional unit F
	T : operation time for the functional unit F
station(M,KK)	KK: set of functional units which can be operated by the assembly station M
queue(M,[(B1,G1,AT1),… (BN,GN,ATN)])	Bi: i th location at the buffer of the assembly station M
	Gi: a product which is put at the i th location of the buffer
	ATi:time when a product Gi is put in the buffer

buffer(M,P)		P : location of the buffer of the assembly station M
input(1,G)	:	Input rule. G is the selected product by the input rule.
dispatch0(G,M)	:	Dispatching rule. If a product G is operated by an assembly station M, this rule selects the next functional unit FU which will be operated by the same assembly station M.
dispatch1(G,M)	:	Dispatching rule. If a product G is operated by an assembly station M, this rule selects a functional unit which will be operated next by another assembly station M1.
dispatch2(G,M)	:	Dispatching rule. If a product G is operated by an assembly station M, and the buffers at assmbly stations which operate functional units of the product G are full, then select a route which combines some assembly stations serially and cyclically. In this case, the vehicle can pass by a station only once and goes from one assembly station to the next carrying a product which will be operated by this next station.
opearation	:	The rule for operation of the assembly station M.
stop_ope	:	The rule to stop operation at the assembly station M.
movement	:	The rule for movement of the vehicle V.
unloading(60)	:	The rule for unloading the vehicle which is carrying a finished product G and arriving at the I/O station.
unloading(M)	:	The rule for unloading the vehicle V which is arriving at an assembly station M.
loading(M)	:	The rule for loading the vehicle V which is arriving at an assembly station M.
summoning	:	The rule for summoning a vehicle V.
time	:	This rule increase an elapsed time T by one time unit.
start	:	This rule is the inference mechanism.

3.2 Application example

This expert system was applied to schedule the job shop type assembly line for the seven types of products.

(1)Assembly line

The assembly line is composed of one I/O station, five assembly stations and two vehicles as shown in Fig 1. Each assembly station has a buffer storage. The capacity of each buffer storage is four products. Seven different types of products are assembled in this line.

The preconditions of the assembly line are corded by using above mentiond arguments as shown in Table 1-(b).

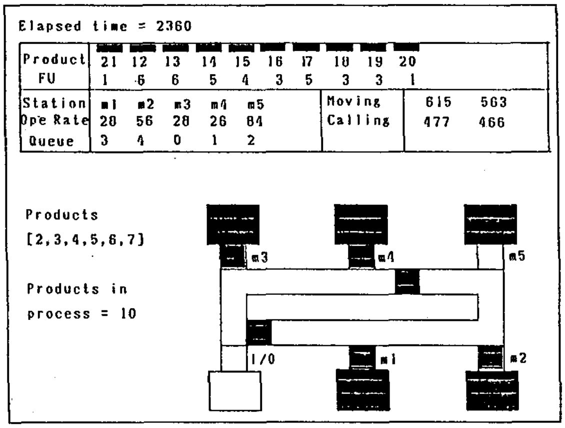

Fig.1 Screen display

(2)Given data of products

Following data on products are given.

(a)Due date, Order, functional units, the assembly station for each functional unit (Table 2)

(b)The operation time of each functional unit (Table 3)

(c)The precedence relations among functional units (Fig 2)

These given data are corded as shown in Table 1-(b).

(3)Selection of priority rules

The inference mechanism is started with the command 'start'.

The priority rules are shown in the CRT, and the selection of all priority rules for the scheduling are asked diagnostically. The selection of priority rules for the input rule is shown in Fig 3. In this example, the due date rule (K1=3) is selected as the priority rule for the input rule. The priority rule H0

Table 2 The operation time of functional units

Product	Functional unit						NOP	DD
	f1	f2	f3	f4	f5	f6		
g1	32	32	30	34	36	120	14	3000
g2	16	18	15	17	18	67	63	5000
g3	16	18	30	34	40	67	12	7000
g4	32	40	15	48	18	40	30	9000
g5	40	40	20	15	20	80	42	4000
g6	20	44	24	20	34	100	35	6000
g7	28	50	32	28	15	120	20	8000

NOP : Number of ordered products
DD : Due date

Table 3 The assembly station for the functional unit f_{ij}

Product	Functional unit					
	f1	f2	f3	f4	f5	f6
g1	m2	m2	m4	m3	m1	m5
g2	m2	m2	m4	m3	m1	m5
g3	m2	m2	m4	m3	m1	m5
g4	m2	m2	m4	m4	m1	m5
g5	m1	m5	m2	m2	m4	m3
g6	m1	m1	m2	m2	m4	m3
g7	m1	m1	m2	m2	m4	m3

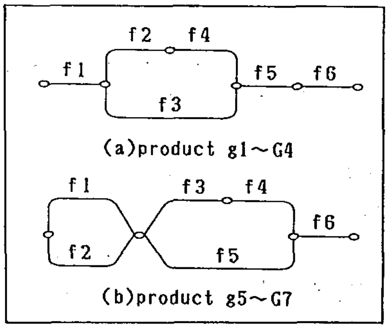

Fig.2 The precedence relation

is selected for the dispatching rule. $(K2_j=2)$ The priority rule PI is selected to choose a product at the buffer. $(K3_j=1)$

(4)Screen display

When all the priority rules for the scheduling were selected, then simulation starts and some parameters, such as the operation ratio of assembly stations and vehicles and queue length, will be displayed on the CRT dynamically. (Fig 1)

(5)Scheduling results

Following scheduling results were made by this expert system. The flow time was 11652 time units. The operation ratio of assembly station m1 is 50%, m2 is 64%, m3 is 52%, m4 is 40% and m5 is 52%. These results were displayed on the CRT.

The assembly sequence of functional units of each product is recorded in a specific file named 'seq.pl' as shown in Fig 4. In Fig 4, the first row shows the assembly sequence (f1,f3,f2,f4,f5,f6) of the product g1. And the following rows represent the same meaning.

The finished time of each product is recorded in the file 'fin.pl' as shown in Fig 5. In Fig 5, the first row shows the fact that the assembly of g4 was started at 1 time unit and finished at 554 time unit. Same meaning is represented in the consecutive rows.

The cumulative operating time of each assembly station is recorded in the file 'ope_time.pl' as shown in

```
-- Select a priority rule to
   select a product at I/O station --
    1: Earliest Due Date
    2: Highest total operation time
    3: Lowest operation ratio
    4: Lowest remaining operation time
     : 1.
```

Fig.3 The selection of priority rules

```
/* assembly sequence */
  1,4,[7,6,5,4,2,3,1],
  2,4,[7,6,5,4,2,3,1],
  3,4,[7,6,5,4,2,3,1],
  4,4,[7,6,5,4,2,3,1],
  5,4,[7,6,5,4,2,3,1],
  6,4,[7,6,5,4,2,3,1],
  7,4,[7,6,5,4,2,3,1],
  8,4,[7,6,5,4,2,3,1],
  9,5,[7,6,4,3,5,1,2],
  10,5,[7,6,4,3,5,1,2],
  11,3,[7,6,5,3,4,2,1],
  12,3,[7,6,5,3,4,2,1],
        :
```

Fig.4 assembly sequence of functional units

```
/* Start & finished time */
  1,4,  1, 554,
  2,4,  2, 602,
  3,4,  3, 658,
  4,4, 13, 702,
  5,4, 19, 763,
  6,4, 25, 860,
  7,4, 44, 906,
  8,4, 76,1020,
  9,5,108, 744,
  10,5,116, 822,
  11,3,555,1115,
  12,3,603,1181,
        :
```

Fig.5 Start and finished time of each product

Fig 6. In Fig 6, the first low shows the fact that the cumulative operating time of m1 is 116 time unit when the elapsed time is 554 time unit,and m2 is 550 time unit, m3 is 80 time unit,m4 is 304 time unit and m5 is 80 time unit.

```
/*Cumulative ope. time */
554,116,556, 80,304, 80
602,134,616, 80,352, 80
658,152,662,160,400,160
702,170,696,160,448,160
744,188,746,240,496,200
763,188,764,240,544,234
822,224,816,280,574,308
860,224,816,308,604,348
906,304,816,336,664,382
1020,394,851,444,729,490
        :
```

Fig.6 The cumulative operating time of assembly stations

REFERENCES

(1) Carrie A.:Simulation of manufacturing system, John Wiley,U.S.(1988)

(2) Bullers W. et al.,"Artificial Intelligence in Manufacturing Planning and controll",AIIE,vol12,No.4,pp.351-363(1980)

(3) Sibayama E. et al.,"Prolog KABA reference manual",(1986),(in Japanese)

(4) Osaki H. et al.:Seisan system gihou,Kyouritu Syuppan(1981), (in Japanese)

(5) Panwalker S,"A Survey of Scheduling Rules",Ope.Res.,25-1,pp.45-61 (1977)

Table 1-(a) Program of the scheduling expert system

```
/* (1)Rules */
/* The rule to increase elapsed time */
time:-retract(time(T)),T1 is T+1,asserta(time(T1)),
      d_cursor(10,0),write(T1),!.

/* Input rule */
input(1,G):-time(T),¥+(wait(60,_)),prod_in(N),N<10,
    prod_sum(S),G is S+1,prod_type(K),
    retract(type_sum(K,R)),R1 is R+1,kosu(K,R1),
    asserta(type_sum(K,R1)),asserta(date(G,K,st(T),fin(0))
    ),asserta(func_fin(G,[])),N1 is N+1,
    asserta(prod_in(N1)),retract(prod_in(N)),
    d_cursor(8,20),write(N1),assertz(wait(60,G)),
    retract(prod_sum(S)),asserta(prod_sum(G)),
    w_prod(G),w_func(G,K),!.
w_func(G,K):-ope_time(_,K,L),member((_,M,_),L),
    M1 is M+60,asserta(func(M1,G)),fail.
w_func(G,K):-!.
del(K,[K|L],L). del(K,[A|L],L1):-del(K,L,L2),L1=[A|L2].
kosu(K,R1):-order(K,R1),products(LL),del(K,LL,L),
    retract(products(LL)),asserta(products(L)),
    d_cursor(0,16),write('           '),
    d_cursor(0,16),write(L),!.
kosu(K,R1):-!.
w_prod(G):-S is G mod 10,
    (S=:=0 ->X is 48;X is 12+(S-1)*4),d_cursor(X,3),
       write(G),color(G,C),
    (S=/=0 ->X1 is 83+(S-1)*32;X1 is 371),
     g_boxfill(X1,27,X1+24,34,C),!.

/* Dispatchng rule */
dispatch:-time(T),X is T mod 4,X=:=0,dispatch(G,M),!.
dispatch:-!.
/* Select the same station */
dispatch0(G,M):-wait(M,G),M>60,¥+(next_func(G,_,_)),
    date(G,KK,st(TS),fin(TF)),precdnce(KK,LP),
    func_fin(G,FL),ope_time(G,KK,L),member((FU,K,WT),L),
    ¥+member(FU,FL),station(M,KL),member(K,KL),
    possible(FU,LP,FL),retract(func_fin(G,FL)),
    asserta(func_fin(G,[FU|FL])),displ(G,FU),
    asserta(next_func(G,M,WT)),!,remain_func(M,G),
    retract(wait(M,G)),store(M,G,WT,0),!.
displ(G,FU):-!,S is G mod 10,
    (S=:=0 ->X is 48;X is 12+(S-1)*4),d_cursor(X,4),
    write(FU),!.
remain_func(M,K):-d_cursor(20,10),write(func(M,K)),
    retract(func(M,K)),!.

/* Select an other station */
dispatch1(G,M1):-capacity(MAX),station(M1,[K]),
    q_length(M1,Q1),Q1<MAX,s2(M1,LM),member(KK,LM),
    date(G,KK,st(TS),fin(TF)),wait(M,G),
    ¥+(next_func(G,_,_)),precdnce(KK,LP),func_fin(G,FL),
    ope_time(G,KK,L),member((FU,K,WT),L),¥+member(FU,FL),
    possible(FU,LP,FL),retract(func_fin(G,FL)),
    asserta(func_fin(G,[FU|FL])),displ(G,FU),
    asserta(next_func(G,M1,WT)),q_decrese(M),
    q_increse(M1),remain_func(M1,G),!.
q_decrese(M):- M=/=60,retract(q_length(M,Q)),
    Q1 is Q-1,asserta(q_length(M,Q1)),w_queue(M,Q1),
    d_cursor(0,13),write(M),write(' '),write(Q),
    write(' '),write(Q1),!.
q_decrese(60):-!.
q_increse(M):- M=/=60,retract(q_length(M,Q)),Q1 is Q+1,
    asserta(q_length(M,Q1)),w_queue(M,Q1),d_cursor(0,14),
    write(M),write(' '),write(Q),write(' '),write(Q1),!.
q_increse(M):-!.

/* loop */
dispatch2(G,M):-wait(M,G),q_length(M,Q),¥+next_func(G,_,_)
    ,loop(M,M,[],LR),reverse(LR,L1),d_cursor(0,11),
    write(L1),write('                              '),!,
    take_f((G1,S,M1,FU,WT),L1),retract(func_fin(G1,FL)),
    asserta(func_fin(G1,[FU|FL])),displ(G1,FU),
    asserta(next_func(G1,M1,WT)),wait(M2,G1),ckk1(M1,G1),
    fail.
ckk1(M1,G1):-remain_func(M1,G1),!.
loop(M,S,LF,LR):-
    wait(S,G),date(G,KK,st(TS),fin(TF)),precdnce(KK,LP),
    func_fin(G,FL),ope_time(G,KK,L),wait(B,G1),¥+(B=S),
    ¥+next_func(G1,_,_),
   (B=M; ¥+member((_,_,B,_,_),LF)),station(B,[K]),
    member((FU,K,WT),L),¥+member(FU,FL),possible(FU,LP,FL)
    ,(M=B ->LR = [(G,S,B,FU,WT)|LF];
        loop(M,B,[(G,S,B,FU,WT)|LF],LR)),!.
reverse([],[]).
reverse([A|X],Z):-reverse(X,Xr),append(Xr,[A],Z).
possible(FU,LP,FL):-
   (member((FU,no,SP),LP);member((FU,and,SP),LP),
     inter_sec(SP,FL,SP)),!.

/* The rule to stop the operation */
stop_ope(G,M):-time(T),fin_time(M,T1),T>=T1,¥+(wait(M,G1))
    ,buffer(M,B1),retract(fin_time(M,T1)),retract(on(M,G))
    ,assertz(wait(M,G)),locate(G,B1),locate_del(G,M),
    retract(next_func(G,M,WT)),!.

/* The rule for vehicles unloading */
unloading(60):-agv(V,G,1,1),¥+G=[],retract(prod_in(N)),
    N1 is N-1,asserta(prod_in(N1)),retract(agv(V,G,1,1)),
    asserta(agv(V,[],1,1)),ope_rate,time(T),
    date(G,KK,st(T1),fin(0)),retract(next_func(G,60,1)),
    ck1(G,KK,T),ck2(G,KK),d_cursor(8,20),write(N1),!.
loading(M):-agv(V,[],B,B),buffer(M,B),wait(M,G),
    retract(call(V,G)),next_func(G,M1,_),buffer(M1,B1),
    retract(agv(V,[],B,B)),asserta(agv(V,G,B,B1)),
    retract(wait(M,G)),date(G,KK,st(T1),fin(0)),
    (wait(M,G1) -> locate(G,B);locate_del(0,B)),!.
unloading(M):-agv(V,G,B,B),¥+G=[],B=/=1,buffer(M,B),
    queue(M,L,T1,Q1),retract(agv(V,G,B,B)),
    asserta(agv(V,[],B,B)),next_func(G,M,WT),
    store(M,G,WT,1),!.
ck1(G,KK,T):-retract(date(G,KK,st(T1),fin(0))),
    tella('c:date.jxw'),write(G),write(','),write(KK),
write(','),write(T1),write(','),write(T),write(','),nl,
    told,!.
ck2(G,KK):-func_fin(G,FL),tella('c:sequence.jxw'),write(G)
    ,write(','),write(KK),write(','),write(FL)    ,
    write(','),nl,told,retract(func_fin(G,FL)),!.
w_queue(M,S):-(M=/=60),X is (M-60)*4+8,Y is 8,
    d_cursor(X,Y),write(S),write(' '),!.
w_queue(M,S):-!.

/* Calculation of operation ratio */
ope_rate:-time(T),work_sum(M,TT),(TT>T ->TT1 is T;
```

```
    TT1 is TT),M=/=60,E is TT1/(T/50)*2,X is 12+(M-61)*4,
    d_cursor(X,7),write('  '),d_cursor(X,7),write(E),
    retract(ope_rate(M,EX)),asserta(ope_rate(M,E)),fail.
ope_rate:-!.

/* The rule for summoning a vehicle */
summoning:-agv(V,[],B,B),¥+call(V,_),wait(M,G),
    next_func(G,M1,WT),¥+call(_,G),buffer(M,B1),
    buffer(M0,B),asserta(call(V,G)),w_move(M0,M,M1),
    (B=:=B1 ->true;retract(agv(V,[],B,B)),
    asserta(agv(V,[],B,B1))),!.
summoning:-!.
w_move(M0,M,M1):-(retract(from_to(M0,M,E));
    retract(from_to(M,M0,E));E is 0),E1 is E+1,
    asserta(from_to(M0,M,E1)),(retract(from_to(M,M1,H));
    retract(from_to(M1,M,H));H is 0),
    H1 is H+1,asserta(from_to(M,M1,H1)),!.

/* The rule for operation of the assembly station */
operation:-queue(M,L,TL,Q),TL>0,¥+on(M,_),s3(M,G),
    next_func(G,M,WT),time(T),T1 is T+WT,
    assertz(fin_time(M,T1)),assertz(on(M,G)),
    locate(G,M),buffer(M,B),locate_del(G,B),
    retract(work_sum(M,TT)),TT1 is TT+WT,
    assertz(work_sum(M,TT1)),!.
operation:-!.

/* The rule for movement of vehicle */
movement:-for(V,1,2),movement(V),fail.
movement:-!.
movement(V):-agv(V,G,N,M),N=/=M,block(N,M,N1),
    retract(agv(V,G,N,M)),(G=[] -> G1 is 0;G1=G),
      (buffer(M1,N) ->move_st(V);locate_del(G1,N)),
      (buffer(M2,N1) ->(G=[] ->arrive_φ(V),usage(V);
          arrive(V),usage(V),locate(G1,N1));
          locate(G1,N1)),assertz(agv(V,G,N1,M)),!.
movement(V):-!.
usage(V):-moveφ_sum(V,TH),X is 50+(V-1)*6,
    d_cursor(X,7),write(TH),move_sum(V,TH1),
    X1 is 50+(V-1)*6,d_cursor(X1,6),write(TH1).
usage(V):-!.

/* Vehicle's operation rate */
move_st(V):-time(T),asserta(st(V,T)),!.
arrive_φ(V):-time(T),retract(st(V,T1)),
    retract(moveφ_sum(V,T2)),TH is T2+T-T1,
    asserta(moveφ_sum(V,TH)),!.
arrive(V):-time(T),retract(st(V,T1)),
    retract(move_sum(V,T2)),TH is T2+T-T1,
    asserta(move_sum(V,TH)),!.
usage:-moveφ_sum(V,TH),X is 50+(V-1)*6,d_cursor(X,7),
    write(TH),fail.
usage:-move_sum(V,TH),X is 50+(V-1)*6,d_cursor(X,6),
    write(TH),fail.
usage:-!.

/* Common program */
inter_sec([],Y,[]):-!.
inter_sec([X|L],Y,R):-inter_sec(L,Y,R1),
    (member(X,Y) -> R=[X|R1];R=R1).
member(A,[A|_]). member(A,[_|L]):-member(A,L).
for(I,I,M) :-I=<M.
for(I,N,M) :-N<M,N1 is N+1,for(I,N1,M).
take_f(E,[E|S]).  take_f(E,[S1|S2]):-take_f(E,S2).
append([],Y,Y).  append([A|X],Y,[A|Z]):-append(X,Y,Z).
change([X|L],X,Y,[Y|L]).
change([S|L],X,Y,[S|L1]):-change(L,X,Y,L1).

/* Displaying the location of vehicles */
locate(G,N):-N<60,color(G,C),I is (N-1)/11,
    I1 is (N-1) mod 11,X is 201+I1*25,X1 is X+23,
    Y is 326-I*25,Y1 is Y+23,g_boxfill(X,Y,X1,Y1,C),!.
locate(G,N):-N>=60,color(G,C),I is N mod 3,I1 is N/63,
    X is 189+I*125,X1 is X+48,Y is 351-I1*165,
    Y1 is Y+38,g_boxfill(X,Y,X1,Y1,C),!.
color(0,7).  color(G,C):-C1 is G mod 6,C is C1+1.

locate_del(G,N):-N<60,I is (N-1)/11,
    I1 is (N-1) mod 11,X is 201+I1*25,X1 is X+23,
    Y is 326-I*25,Y1 is Y+23,g_boxfill(X,Y,X1,Y1,0),!.
locate_del(G,N):-N>=60,I is N mod 3,I1 is N/63,
    X is 189+I*125,X1 is X+48,Y is 351-I1*165,
    Y1 is Y+38,g_boxfill(X,Y,X1,Y1,0),!.

/* CRT initiarize */
grafics_int :-g_screen(3,0,0,1),g_origin(0,2),d_clear,
g_cls,crt,!.
crt:-for(J,1,2),for(I,1,3),X is 188+(I-1)*125,
    X1 is X+50,X2 is X+12,X3 is X2+25,Y is 185+(J-1)*165,
    Y1 is Y+40,Y2 is Y1+25-(J-1)*90,g_box(X2,Y1,X3,Y2,7,0)
    ,g_boxfill(X,Y,X1,Y1,0),g_box(X,Y,X1,Y1,7,0),fail.
crt:-g_box(200,250,475,325,7,0),g_box(225,275,450,300,7,0)
    ,fail.
crt:-d_cursor(0,3),write('  product'),d_cursor(0,4),
    write('fuc. unit'),d_cursor(0,6),write(' station')
    ,d_cursor(0,7),write('ope. rate'),d_cursor(0,8),
    write('  queue '),d_cursor(1,20),write('prod in. ='),
    d_cursor(40,6),write('move sum'),d_cursor(40,7),
    write('moveφ sum'),d_cursor(0,10),write('loop='),
    g_box(8,25,620,72,7,0),g_box(8,72,620,118,7,0),
    g_box(8,118,620,134,7,0),g_line(8,118,620,118,8,0),
    g_line(75,25,75,134,7,0),g_box(300,72,380,134,7,0),
    d_cursor(24,21),
    write('     I/O             m1              m2'),
    d_cursor(24,15),
    write('     m3              m4              m5'),
    d_cursor(10,6),write('  m1  m2  m3  m4  m5'),
    d_cursor(0,15),write('products'),products(L),
    d_cursor(0,16),write('              '),d_cursor(0,16),
    write(L),!.

/* (2)Inference mechanism */
start:-wad,in,func,buffer,cap,grafics_int,fact,repeat,
al,prod_in(0),w_agv.
al:-time,time(T),(input(1,G),dispatch4(G,M);true),
    (stop_ope(G1,M1),(dispatch0(G1,M1);dispatch4(G1,M1));
        true),
    summoning,
    (unloading(60);unloading(M2);loading(M3);true ),
    operation,
    movement,!.
al:-!.

dispatch4(G,M):-(dispatch1(G1,M);true),dispatch1(G,M1),
    dispatch1(G1,M),!.
dispatch4(G,M):-dispatch2(G,M),!.  dispatch4(G,M):-!.
```

```
/* (3) Rules for the selection of priority rules */
wad:-d_clear,g_cls,d_cursor(20,5),
    write('--- Select foreward sch. or backward sch. ---')
    ,d_cursor(25,7),write('1:  Foreward scheduling'),
    d_cursor(25,8), write('2:  Backward scheduling  '),
    d_cursor(25,9),read(N),
    (N=1,reconsult('b:¥ward¥1.jxw');
    N=2,reconsult('b:¥ward¥2.jxw')).
buffer:-d_clear,g_cls,d_cursor(10,5),
    write('--- Select a priority rule to take out a
product from a buffer ---'),
    d_cursor(25,7),write('1:First in first out '),
    d_cursor(25,8),write('2:Shortest due date'),
  d_cursor(25,9),write('3:Lowest imminent operation time')
    ,d_cursor(25,9),write('4:  Highest imminent operation
time'),d_cursor(25,11),read(N),
    (N=1,reconsult('b:¥buffer¥1.jxw');
     N=2,reconsult('b:¥buffer¥2.jxw');
     N=3,reconsult('b:¥buffer¥3.jxw');
     N=4,reconsult('b:¥buffer¥4.jxw')).
func:-d_clear,d_cursor(20,5),write('--- Select a priority
rule for dispatching ---'),
    d_cursor(25,7), write('1:Nearest station   '),
    d_cursor(25,8), write('2:Highest operation time '),
    d_cursor(25,9),write('3:Lowest opearation time '),
    d_cursor(25,11),read(N),
    (N=1,reconsult('b:¥func¥1.jxw');
     N=2,reconsult('b:¥func¥2.jxw');
     N=3,reconsult('b:¥func¥3.jxw')).
in:-d_clear,d_cursor(10,5),write('--- Select a priority
rule to select a product at I/O station ---'),
    d_cursor(25,7), write('1:  Earliest due date'),
    d_cursor(25,8),write('2:Highest total operation time')
    ,d_cursor(25,9),write('3:Lowest operation ratio'),
    d_cursor(25,10),
    write('4:Lowest remaining operation time'),
    d_cursor(25,13),read(N),
    (N=1,reconsult('b:¥input¥1.jxw');
    N=2,reconsult('b:¥input¥2.jxw');
    N=3,reconsult('b:¥input¥3.jxw');
    N=4,reconsult('b:¥input¥4.jxw')).
cap:-d_clear,d_cursor(15,7),
    write('How many buffer capacity is ? '),
    d_cursor(15,8),read(CP),assert(capacity(CP)),!.
w_agv:-tell('c:fromto.jxw'),listing(from_to),told.

/* (4)Priority rules */
/*  Due date urle   */
prod_type(KK):-due_date(L),take_f(KK,L),products(LL),
    member(KK,LL),!.
due_date([1,5,2,6,7,3,4]).

/*  Lowest total operation time rule   */
prod_type(KK):-order(L),take_f(KK,L),products(LL),
    member(KK,LL),!.
order([1,7,6,5,3,4,2]).

/*  Operation ratio rule  */
prod_type(G):-sort([61,62,63,64,65],L),take_f(M,L),
    ope_rate(M,E),station(M,K),products(LL),
    selection(LL,K,G),!.
sort([],[]).  sort([X|L],S):-sort(L,L1),ins(X,L1,S).
ins(X,[Y|L],[Y|M]):-ope_rate(X,EX),ope_rate(Y,EY),
    EY<EX,!,ins(X,L,M).
ins(X,L,[X|L]).
selection(LL,K,G):-order(K,L),take_f(G,L),member(G,LL),!.
selection(LL,K,0):-!.
order([1],[7,6,3,5,1,2,4]). order([2],[4,1,7,6,5,2,3]).
order([3],[7,6,5,1,3,2,4]). order([4],[4,6,1,3,5,2,7]).
order([5],[1,2,3,4,5,6,7]). order([0],[2,4,3,5,6,7,1]).

/*  Remaining operation time rule  */
prod_type(G):-sort([61,62,63,64,65],L),take_f(M,L),
    ope_rate(M,E),station(M,K),products(LL),
    selection(LL,K,G),!.
sort([],[]).  sort([X|L],S):-sort(L,L1),ins(X,L1,S).
ins(X,[Y|L],[Y|M]):-queue(X,LX,TX,QX),queue(Y,LY,TY,QY),
    TY=<TX,!,ins(X,L,M).   ins(X,L,[X|L]).
selection(LL,K,G):-order(K,L),take_f(G,L),member(G,LL),!.
selection(LL,K,0):-!.
order([1],[7,6,3,5,1,2,4]). order([2],[4,1,7,6,5,2,3]).
order([3],[7,6,5,1,3,2,4]). order([4],[4,6,1,3,5,2,7]).
order([5],[1,2,3,4,5,6,7]). order([0],[2,4,3,5,6,7,1]).

/* dispatching  rule */
/*  Largest imminent operation time rule  */
s2(61,[4,2,1,5,3,6,7]).  s2(62,[3,2,5,6,7,1,4]).
s2(63,[4,2,3,1,5,6,7]).  s2(64,[7,2,5,3,1,6,4]).
s2(65,[7,6,5,4,3,2,1]).  s2(60,[1,7,6,5,3,4,2]).

/* buffer */
/*  First in first out rule      */
store(B,G,T,C):-(C=0 ->X is 1;time(X)),queue(B,L,T1,Q1),
    member((S,0,0),L),date(G,K,_,_),
    change(L,(S,0,0),(S,G,X),L1),retract(queue(B,L,T1,Q1))
    ,T2 is T1+T  ,Q2 is Q1+1,asserta(queue(B,L1,T2,Q2)),
    q_length(B,LL),w_queue(B,LL),!.

s3(B,G):- queue(B,L,T1,Q1),smallest(L,(S,G,T)),
    change(L,(S,G,T),(S,0,0),L1),retract(queue(B,L,T1,Q1))
    ,next_func(G,B,WT),T2 is T1-WT,Q2 is Q1-1,
    asserta(queue(B,L1,T2,Q2)),q_length(B,LL),
    w_queue(B,LL),!.
smallest([(S,G,T)],(S,G,T1)):-(T=0 ->T1 is 32000;T1 is T).
smallest([(S,G,T)|L],(S1,G1,M1)):-smallest(L,(S2,G2,M2)),
    ((T=0;T>M2) ->S1 is S2,G1 is G2,M1 is M2;
       S1 is S,G1 is G,M1 is T).
/*  Lowest imminent operation time rule   */
store(B,G,T,C):-(C=0 ->X is 1;time(X)),queue(B,L,T1,Q1),
    member((S,0,0),L),change(L,(S,0,0),(S,G,T),L1),
    retract(queue(B,L,T1,Q1)),T2 is T1+T,Q2 is Q1+1,
    asserta(queue(B,L1,T2,Q2)),w_queue(B,Q2),!.
s3(B,G):-queue(B,L,T1,Q1),smallest(L,(S,G,T)),
    change(L,(S,G,T),(S,0,0),L1),retract(queue(B,L,T1,Q1))
    ,next_func(G,B,WT),T2 is T1-WT,Q2 is Q1-1,
    asserta(queue(B,L1,T2,Q2)),w_queue(B,Q2),!.
smallest([(S,G,T)],(S,G,T1)):-(T=0 ->T1 is 32000;T1 is T).
smallest([(S,G,T)|L],(S1,G1,M1)):-smallest(L,(S2,G2,M2)),
    ((T=0;T>M2) ->S1 is S2,G1 is G2,M1 is M2;S1 is S,
       G1 is G,M1 is T).
/*  Due date rule       */
store(B,G,T,C):-(C=0 ->D is 1;due_date(G,D)),
    queue(B,L,T1,Q1),member((S,0,0),L),
    change(L,(S,0,0),(S,G,D),L1),retract(queue(B,L,T1,Q1))
    ,T2 is T1+T,Q2 is Q1+1,asserta(queue(B,L1,T2,Q2)),
```

```
    w_queue(B,Q2),!.
s3(B,G):-queue(B,L,T1,Q1),smallest(L,(S,G,T)),
    change(L,(S,G,T),(S,0,0),L1),retract(queue(B,L,T1,Q1))
    ,next_func(G,B,WT),T2 is T1-WT,Q2 is Q1-1,
    asserta(queue(B,L1,T2,Q2)),w_queue(B,Q2),!.
work_sum(1,0). work_sum(2,0). work_sum(3,0).
work_sum(4,0). work_sum(5,0). work_sum(6,0).
ope_rate(1,0). ope_rate(2,0). ope_rate(3,0).
ope_rate(4,0). ope_rate(5,0). ope_rate(6,0).
time(0). prod_in(0). prod_sum(0).
block(1,M,13):-!. block(6,M,18):-!. block(11,M,22):-!.
block(55,M,43):-!. block(50,M,38):-!. block(45,M,34):-!.
block(33,M,44):-!. block(23,M,12):-!.
block(N,M,N1):-1=<N,N=<22,(N=17,M=6,N1 is 6;N=12,M=1,
   N1 is 1;N=22,M=11,N1 is 11;N=22,N1 is 33;N1 is N+1),!.
block(N,M,N1):-34=<N,N=<44,(N=44,M=55,N1 is 55;N=39,M=50,
   N1 is 50;N=34,M=45,N1 is 45;N=34,N1 is 23;N1 is N-1),!.
buffer(60,1):-!. buffer(61,6):-!. buffer(62,11):-!.
buffer(63,45):-!. buffer(64,50):-!. buffer(65,55):-!.
```

Table 1-(b) Given data on the assembly line

```
station(60,[0]). station(61,[1]). station(62,[2]).
station(63,[3]). station(64,[4]). station(65,[5]).
queue(60,[],0,0).
queue(61,[(1,0,0),(2,0,0),(3,0,0),(4,0,0),(5,0,0)],0,0).
queue(62,[(1,0,0),(2,0,0),(3,0,0),(4,0,0),(5,0,0)],0,0).
queue(63,[(1,0,0),(2,0,0),(3,0,0),(4,0,0),(5,0,0)],0,0).
queue(64,[(1,0,0),(2,0,0),(3,0,0),(4,0,0),(5,0,0)],0,0).
queue(65,[(1,0,0),(2,0,0),(3,0,0),(4,0,0),(5,0,0)],0,0).
q_length(1,0). q_length(2,0). q_length(3,0).
q_length(4,0). q_length(5,0). q_length(6,0).
agv(1,[],1,1). move_sum(1,0). moveφ_sum(1,0).
agv(2,[],1,1). move_sum(2,0). moveφ_sum(2,0).

products([1,5,2,6,7,3,4]).
due_date(1,1). due_date(2,3). due_date(3,6).
due_date(4,7). due_date(5,2). due_date(6,4).
due_date(7,5).
order(1,14). order(2,63). order(3,12).
order(4,30). order(5,42). order(6,35). order(7,20).
type_sum(1,0). type_sum(2,0). type_sum(3,0).
type_sum(4,0). type_sum(5,0). type_sum(6,0).
type_sum(7,0).
precdnce(K,[(1,no,[]),(2,and,[1]),(3,and,[1]),(4,and,[2]),
    (5,and,[3,4]),(6,and,[5]),(7,and,[6])]):- 1=<K,K=<4.
precdnce(K,[(1,no,[]),(2,no,[]),(3,and,[1,2]),(4,and,[3]),
    (5,and,[1,2]),(6,and,[4,5]),(7,and,[6])]):- 5=<K,K=<7.
ope_time(G,1,[(1,2,32),(2,2,32),(3,4,30),(4,3,34),
    (5,1,36),(6,5,120),(7,0,1)]).
ope_time(G,2,[(1,2,16),(2,2,18),(3,4,15),(4,3,17),
    (5,1,18),(6,5,67),(7,0,1)]).
ope_time(G,3,[(1,2,16),(2,2,18),(3,4,30),(4,3,34),
    (5,1,40),(6,5,67),(7,0,1)]).
ope_time(G,4,[(1,2,32),(2,2,40),(3,4,15),(4,4,48),
    (5,1,18),(6,5,40),(7,0,1)]).
ope_time(G,5,[(1,1,40),(2,5,40),(3,2,20),(4,2,15),
    (5,4,20),(6,3,80),(7,0,1)]).
ope_time(G,6,[(1,1,20),(2,1,44),(3,2,24),(4,2,20),
    (5,4,34),(6,3,100),(7,0,1)]).
ope_time(G,7,[(1,1,28),(2,1,50),(3,2,32),(4,2,28),
    (5,4,15),(6,3,120),(7,0,1)]).
```