TR-041

Simulator of XP' S

Moritoshi Aso

January, 1984

©1984, ICOT

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

Simulator of XP'S

TR-041

麻生盛敏
（新世代コンピュータ技術開発機構）

Simulator of XP'S

Moritoshi Asoh

ICOT Research Center
Technical Report TR-041, February 15 1984

ABSTRACT

We proposed an Extended OR-Parallel Prolog System, its computational model and the XP'S-Interpreter (ICOT Technical Report TR-023).
This system (called "XP'S") features the following facilities :

(1) OR-Parallel Reduction,
(2) Modularization (or Multi-World Expression),
(3) Preceded Implication (Control for searching AND-OR tree),
(4) Extended Guard (Control for searching AND-OR tree),
(5) Synchronization of Parallel Processes.

This paper describes the simulator of XP'S and the analytic result of several programs.

Simutator of XP'S

麻 生 盛 敏
（新世代コンピュータ技術開発機構）

１．はじめに

知識情報処理では与えられた問題と知識から問題解決手順を試行錯誤的に発見する推論機能が重要な役割を果す。このような機能を言語自体に備えたPROLOGが最近脚光を浴びており、これを直接実行する計算機システムを実現することはより高度な知識情報処理システムを構築するための第一歩となろう。

我々はすでに［１］において次に述べるような機能を備えた、拡張OR並列PROLOGシステム-XP'S- (EXtended OR-Parallel Prolog System) とその計算モデルについて提案し、そのインタプリータを開発した。

（１） OR並列推論処理機能
（２） モジュール化機能
（３） 推論の優先度制御機能（Preceded Implicationの導入）
（４） 拡張ガード機能
（５） 並列プロセスの同期制御機能

ここでは、新しく開発したこのXP'SのSimulator と、これを用いて解析した２，３のプログラムの解析結果について述べる。

２．拡張OR並列PROLOGシステム-XP'S-の概要

２．１ 計算モデル

PROLOGプログラムの実行過程は空なゴール文で終るゴール文の列の導出過程であり、その基本的実行サイクルを図示したのが図１である。
XP'Sではこれを図２に示すようなプロセス・モデルとして実現している。
但し、ここでいうプロセスとは従来のプロセスの概念とは別の概念であり、並列に解かれるゴール文に対応したものである。

また、図３はそのアーキテクチャの概念図を示したものであり、各Unification UnitはNetwork を介して適当なProcess Poolにあるレディ状態のプロセスをロードし、このプロセスが解くべきゴールの最初のリテラルとUnifiable なルールをClause Pool

より選択し、つづいてこれをもとに新しいプロセスを並列にForkする。
Forkされた子プロセスはNetwork を介して適当なProcess Poolに戻される。
子プロセスから (Success)メッセージが戻ってくるたびに、ウェイト状態の親プロセスをもとに新しくプロセスがつくられる。この時の状態を図示したのが図２である。
　拡張ガードや並列プロセスの同期制御機能は図３のGate Pool で実行される。

２．２　シンタックス

　XP'Sは現在の逐次型PROLOGを完全にそのサブセットとして含み、これに以下のような種々の機能を導入したシステムである。ここではそのシンタックスと特徴をプログラム例を用いて説明する。
なお、例として用いたプログラムはXP'Sインタプリータ上で実行できる。
但し、本シミュレータでは（３），（５），および（６）の機能はサポートしていないのでModule2,およびModule3 のプログラムは実行できない。

（１）モジュール定義

　XP'Sのプログラムは例１に示すようにdefineおよびend により定義されたいくつかのモジュールからなる。

（２）ゴールフレーム

　従来のゴールG はG=G1,G2,……,Gn のようにゴールリテラルGiの列であったが、XP'SではこれにG を解くべきモジュール名を合せて、 goal_frame(Module_name, G) として与える。

（３）プロセス生成述語

　これは一種の組込み述語であり、あるモジュール内から他のモジュール（同じモジュールでもよい）に新しいゴールフレーム（初期プロセス）を生成する場合に、例１の test3 モジュールのように用いる。

（４）Preceded Implication

　XP'Sでは従来のPROLOGの“:-”オペレータに優先度をつけたオペレータ（これをPreceded Implication と呼ぶことにする）を導入し、OR並列処理の制御を可能にしている。現在のシミュレータでは高優先度と低優先度の２レベルをサポートしており、例１の“::- ”が高優先度に相当する（［１］では低優先度であった）。

（５）拡張ガード

　XP'Sでは従来のカットオペレータに相当する機能を持った拡張ガード“!N”(N≧1)をサポートしている。これはCONCURRENT PROLOG におけるガードの拡張でもある。
拡張ガードの“N ”の意味はAND-OR木のOR分岐の所で高々 N個の節の非決定的なOR選択を許すと解釈する他に、その節に負荷された重みを表わすと解釈することもできる。
XP'Sインタプリータはどちらのモードでも実行できる。

（6）並列プロセス同期用述語

XP'Sでは従来のセマフォアの概念に準じた同期用述語“sync”を導入している。
これにより、例１の test3モジュールのような生産者・消費者問題のような並列事象間の同期を扱う問題もプログラムできる。
ヘッド部がproducer(M) とconsumerの２つのルールは先に述べたプロセスとして実現され、sync(identifier ,N^) （送信側）と sync(identifier , OUT?) （受信側）との間で同期がとられる。

```
/* Module 1 */  define test1.
   factorial(X,Y) ::- fact(0,X,Y), display(Y).
   fact(X,X,1).
   fact(X,Y,Y)    :-  Y is X+1.
   fact(X,Y,Z)    ::- Y>X+1, MID is (X+Y)/2, fact(X,MID,Z1), fact(MID,Y,Z2),
                      Z is Z1*Z2.
   end.
/* Module 2 */  define test2.
   go(X)  :- p(X,Y), display(Y).
   p(X,1) :- a(X), b(X), !1.
   p(X,2) :- a(X), !2, b(X).
   p(X,3) :- a(X), !3, b(X).
   p(X,4) :- a(X), b(X), !4.
   a(1).     a(2).     b(1).
   end.
/* Module 3 */  define test3.
   go :- producer(0).
   go :- consumer.
   producer(M) :- M<10, N is M+1, sync(identifier, N^),
                  generate_PROCESS(test3, producer(N)).
   consumer    :- sync(identifier, OUT?), OUT<10, display(OUT),
                  generate_PROCESS(test3, consumer).
   end.
```

例１．　XP'Sプログラムの例

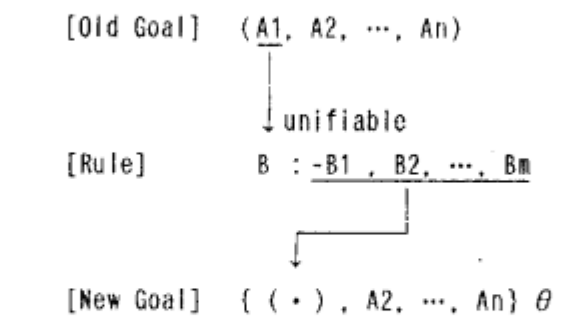


図１．　PROLOGの基本的実行サイクル

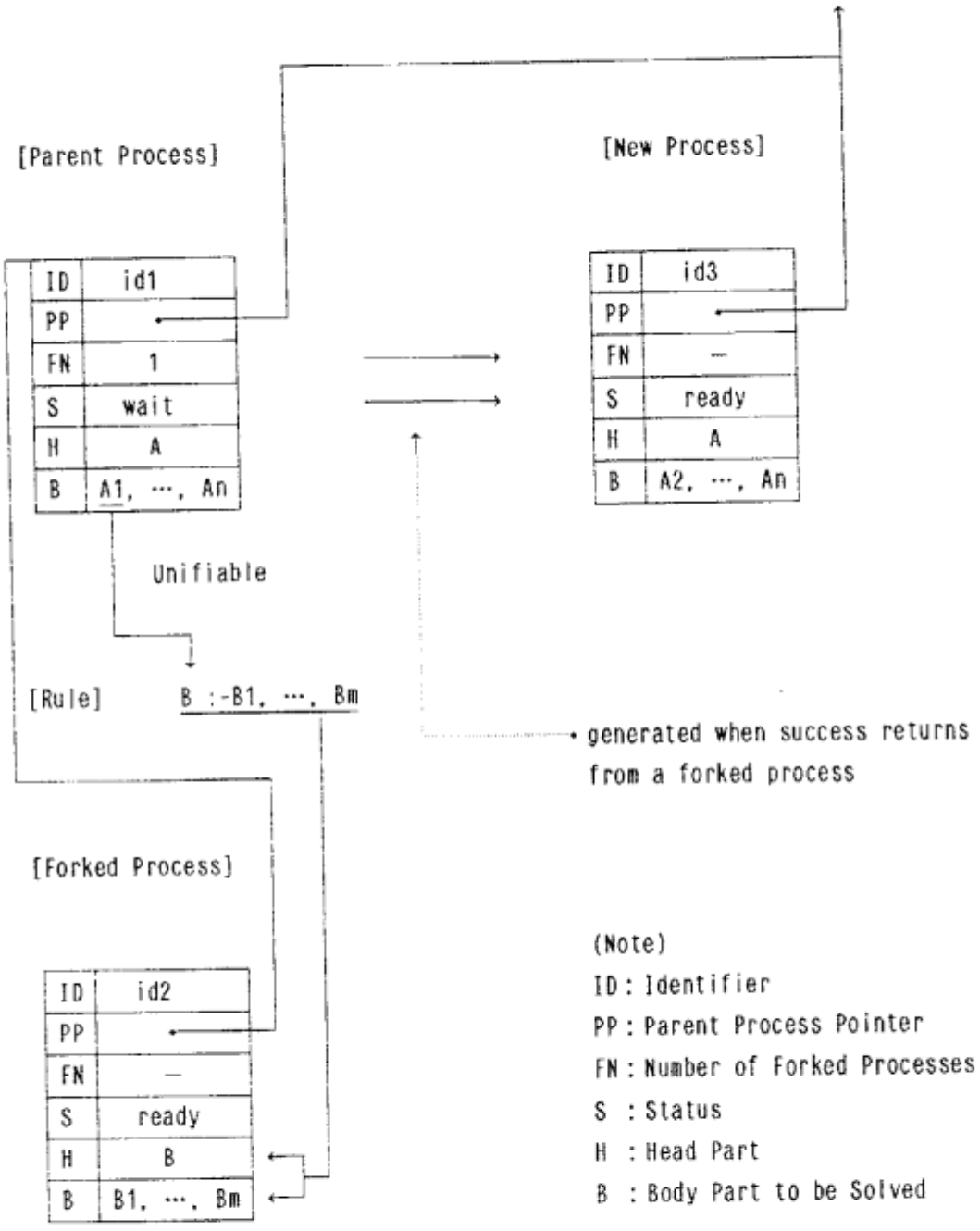


図2. XP'Sにおけるプロセス・モデル

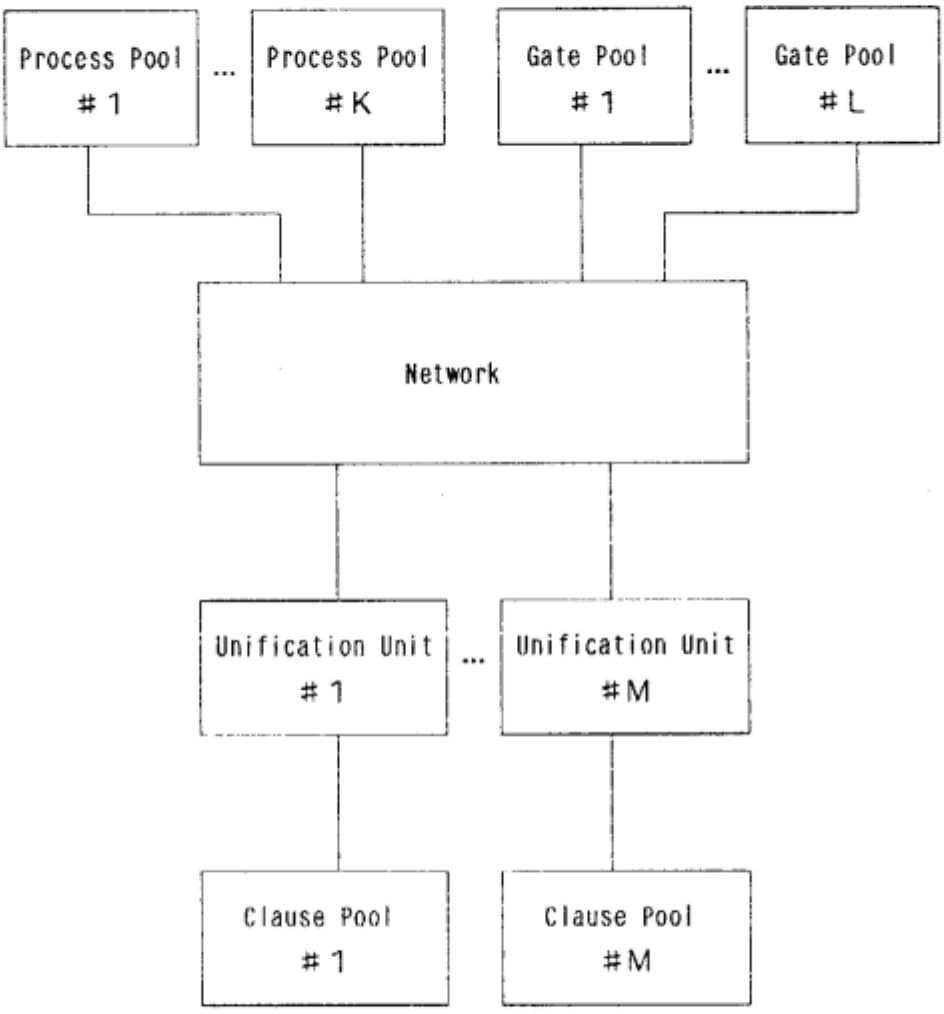


図3．　アーキテクチャの概念図

3．XP'Sシミュレータのモジュールの構成と機能

XP'SシミュレータはDEC-10 PROLOG 上に試作されており、次の４つのデータ構造と５つのプログラム・モジュールから構成される。

(1) データ構造

① Common Area
② Process Pool
③ Channel
④ Clause Pool

(2) プログラム・モジュール

① Control Module (XPS, XPS.TRANSLATOR, XPS.INITIALISER, XPS.DISPATCHER)
② PPU Module (XPS.PPU-MODULE)
③ Channel Module (XPS.CHANNEL-MODULE)
④ UU Module (XPS.UU-MODULE, XPS.BUILT-IN)
⑤ Analyze Module (XPS.ANALYZER)

【注】 PPU : Process Pool Unit
UU : Unification Unit

カッコ内は実際のプログラム・ファイル名を示す。
これらの関係を図示したのが図４である。
以下、それぞれの基本的な構成と機能を説明する。

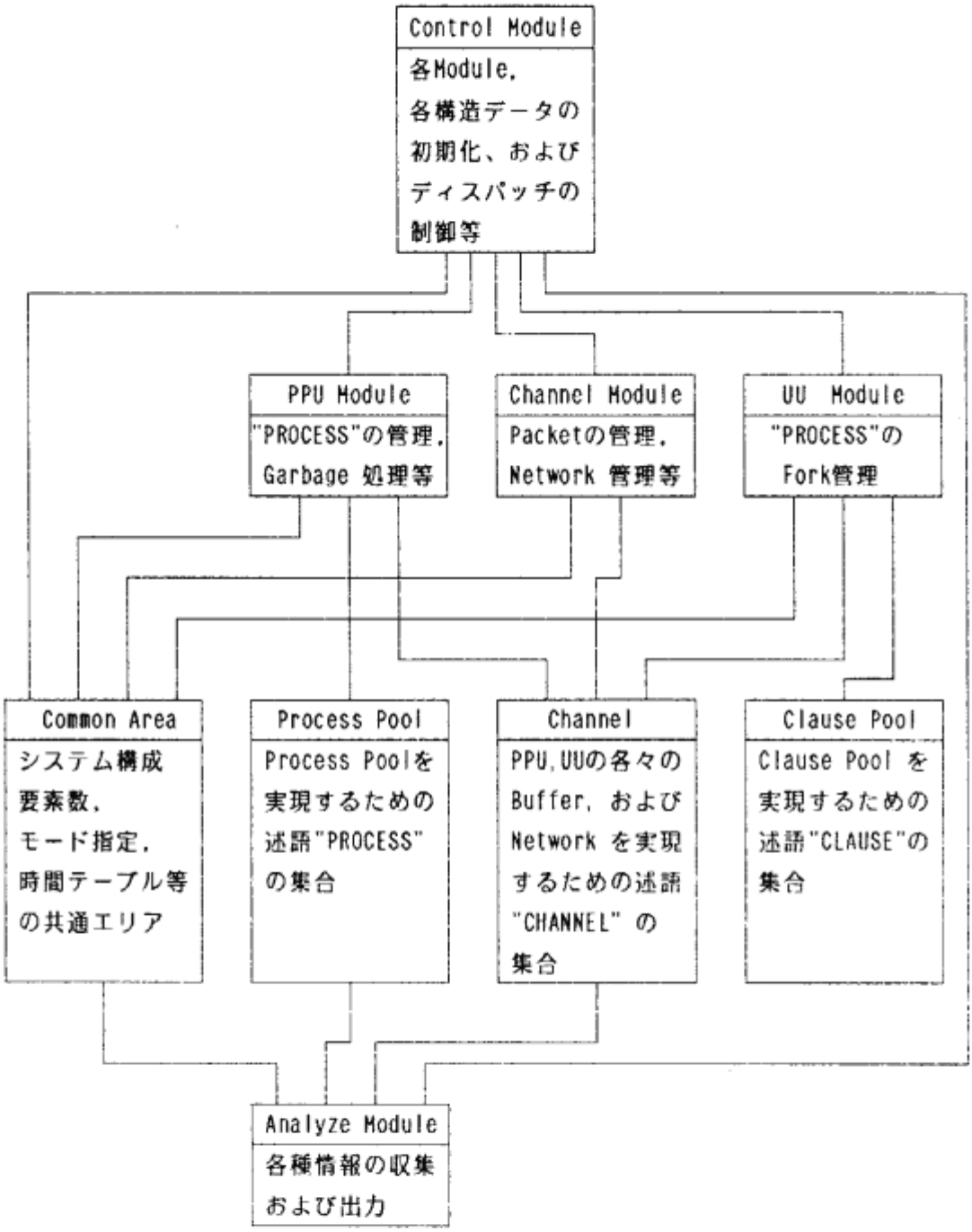


図４．　XP'Sシミュレータの構成図

３．１　データ構造

前述の４つのデータ構造について説明する。

(1) Common Area

本データ構造は各モジュール共通な次の述語からなる。

(a) ｐｐｕ＿Ｎｏ(PPN)

Process Pool Unit の数を表わし、初期設定時にControl Moduleが設定する。
他のModuleは参照のみ可。

(b) ｕｕ＿Ｎｏ(UN)

Unification Unitの数を表わし、初期設定時にControl Moduleが設定する。
他のModuleは参照のみ可。

(c) ｍｏｄｅｐ(Mode)

シミュレータの実行モードを表わし、初期設定時にControl Moduleが設定する。
他のModuleは参照のみ可。

① Mode=analyze ： 本モードの時、Analyze Moduleは収集データを解析し出力する。

② Mode=display ： 本モードの時、Analyze Moduleは収集データの中間情報を出力する。

③ Mode=step ： 本モードの時、各Unification UnitでUnification を実行するごとにStopしその時の情報を出力する。

④ Mode=debug1 ： 本モードの時、図４の "PROCESS"と"CHANNEL" 両述語のListを出力する。

⑤ Mode=(abort/T)： 本モードの時、Ｔで指定された時間後にAbort する。

(d) time_table(ppu/uu/net , UN, PT, CT, S)

時間管理を行うための述語であり、各引数の意味は以下の通りである。

① ppu/uu/net : PPU/UU/Networkユニットの区別を表わす。
② UN : 各ユニットの番号を表わす（Network はUN=0）。
③ PT : 各ユニットの処理時間を表わす。
④ CT : 各ユニットの次に動作可能な時刻を表わす。
⑤ S : 各ユニットの状態(b=busy,i=idle) を表わす。

本述語は初期設定時にControl Moduleが設定し(CT=0,S=i)、その後各ユニットがそれぞれのTime Tableを更新する。

(e) process_No(PN)

各"PROCESS" に割当てたプロセス番号の最新の値を保持する。

初期設定時にControl Moduleが設定し(PN=0)、PPU Moduleで新しい"PROCESS" を生成する時にこれを更新し使用する。

(2) Process Pool

XP'Sシミュレータでは、第２章で述べたプロセスを次のような述語として表わす。Process Poolはこの述語の集合である。

☆ process(ID, PP, FN, S , Head, (Body))
☆ ID= [P , PN, PPN , MN, OL, AL]
① P : Priority
② PN : Process Number
③ PPN : Process Pool Number
④ MN : Module Name
⑤ OL/AL : OR/AND Level
⑥ PP : Parent Process Pointer, PP=(PNparent, PPNparent)
⑦ FN : Forked Process Number
⑧ S : Status, S=ready/wait
⑨ Head : Head Part to be returned
⑩ Body : Body Part to be solved

本データの更新は全てPPU Moduleが行う。
但し、初期プロセスの登録のみはControl Moduleが行う。
また、Analyze Moduleは情報収集のためにこれを参照する。

(3) Channel

本データ構造はPPU,UU各UnitのI/O-BufferとNetwork を表わす述語"CHANNEL" の集合である。"CHANNEL" の構造は以下の通りである。

☆ channel(UN, No, IO, T , DATA)

① UN : Unit Name , UN=ppu/net/uu

② No : Unit Number

- UN=ppu/uu のとき, No=(そのUnit No)
- UN-netのとき, No=(送信先Unit No)

③ IO : Input/Output

- UN=ppu/uu のとき, IO=in/out (IN/OUT-Buffer の区別)
- UN=netのとき, IO=ppu/uu (送信方向の区別)

④ T : Time, 本データの発生した時刻
但し、UN=netのときは(Time,Delay)のペアとする。

⑤ DATA: 送受信データ
データの形式は次の通りである。

◎ PPU Output-Buffer(UN=ppu,IO=out)から
UU Input-Buffer(UN-uu,IO=in)まで,
・DATA=data1(ID, Subgoal)

◎ UU Output-Buffer(UN-uu,IO=out)から
PPU Input-Buffer(UN=ppu,IO-in)まで,
・OR-Fork のResult-Packet
DATA=data1(ID, Subgoal, FN)
FN>0 : OR-Fork 成功時
FN-0 : OR-Fork 失敗時
・Child-Process のGenerating-Packet
DATA=data1(New_ID, PP, Head, (Body))

◎ PPU Output-Buffer(UN=ppu,IO=out)から
PPU Input-Buffer(UN=ppu,IO=in)まで,
・DATA=data2(PP, Head, B)
B=true : Child-Process のBody部が解けた時
B=fail : Child-Process のBody部が失敗した時
B=up : Parent-ProcessのFNの更新時

【注】 ID,PP,Head,Body については,(2)と同じである。

本データはPPU,Channel,UUの各Moduleからそれぞれ更新される。
また、Analyze Moduleは情報収集のためにこれを参照する。

(4) Claus Pool

Claus PoolはXP'Sのプログラムであり、１つの節（Clause）は次のような述語として表現される。本データ構造はこの述語の集合である。

☆ clausep([P,MN], Head, Body)

① P : Priority
② MN : Module Name
③ Head : Head Part of Clause
④ Body : Body Part of Clause

本データは初期設定時にControl Moduleが作成し、UU Module がこれを参照する。

３．２ プログラム・モジュールの基本機能

XP'Sシミュレータを構成する５つのプログラム・モジュールの関係は図４に示した通りである。これらモジュールの基本機能について以下に説明する。

(1) Control Module

本モジュールは、シミュレータの初期化，各モジュールへの制御のディスパッチ等を行う。主な機能は以下の通りである。

① ユーザ・プログラムの入力およびその変換(Clause Poolの作成）。
② Common Area の初期設定。
③ 初期プロセス（ゴール）の入力。
④ シミュレータ各モジュールへの制御のディスパッチ。

(2) PPU Module

本モジュールは、Process Pool Unit をシミュレートする。
主な機能は以下の通りである。

① "ready" 状態の"PROCESS" をProcess PoolよりLoadする。
② Loadした"PROCESS" よりReducible なSubgoal を選択し、
"CHANNEL" (PPU Output-Buffer に相当する) として格納する。
③ Subgoal を選択した後の"PROCESS" の状態を"wait"とし、
元のProcess Poolへ戻す。
④ Channel より"CHANNEL" (これはUUもしくは他のPPU からのResult-Packet に相当する) をLoadし、それぞれに応じた処理を行う。
◎ OR-Fork に伴う処理
・ FN>0 : Succeed in OR-Fork
・ FN=0 : Fail in OR-Fork
◎ 子プロセスからのResult-Packet の処理
・ B=true : Succeed in Child-Process
・ B=fail : Fail in Child-Process
・ B=up : Count up Forked Process Number
⑤ Garbage の処理。

(3) Channel Module

本モジュールは、PPU I/O-Buffer←→Network ←→UU I/O-Buffer をシミュレートする。 主な機能は以下の通りである。

① channel(net , No, IO, (T, Delay), DATA) というデータに関して
Delay = 0ならば、これをchannel(IO, No, in, T , DATA) と変換し、
Delay ≠ 0ならば、New-Delay=Delay − 1とする。
② channel(UN, No, out , T , DATA) をLoadし、
channel(net , No' , ppu/uu, (T' , Delay), DATA) と変換する。

上記の操作はあるPPU/UUのOutput-Buffer からデータをLoadし、Network 上にのせ、Network 上にすでにあるデータはあらかじめ設定されたDelay Timeをcount downし、Delay Time=0となったデータを目的のPPU/UU Input-Buffer に格納する操作である。
上記の操作を全てのPPU,UUについて行う。

【注】 "CHANNEL" の構造は 3.1 (3)を参照。

(4) UU Module

本モジュールは、Unification Unitをシミュレートする。

主な機能は以下の通りである。

① Channel よりchannel(uu, No, in, T , DATA) をLoadする。

② ①でLoadしたDATA=data1(ID , Subgoal)をもとにして、
Subgoal とUnifiable なHeadをもつClauseをClause Pool より選択し、
Unification を実行する。

③ ②で得られた結果をchannel(uu, No, out , T', DATA')というResult-Packet
してChannel へ戻す。

◎ Unification に成功したとき,
DATA'=data1(ID , Subgoal , FN) および
DATA'=data1(New_ID, PP, Head, (Body))
但し、FN>0である。

◎ Unification に失敗したとき,
DATA'=data1(ID , Subgoal , 0)

【注】 "CHANNEL" の構造は 3.1 (3)を参照。

(5) Analyze Module

本モジュールは、各種統計情報の収集と出力を行う。

これらの情報は以下の通りである。

① 各Process PoolにおけるStatus別の"PROCESS" の数。

② Channel 上のデータの数。

③ 各Unification UnitにおけるOR並列度。

④ 各Unitの稼働率。

以上、（１）～（５）の各モジュールの機能をPROLOG風に記述すると次のようになる。

```
/*============================< XPS SIMULATOR >==============================*/
                                                          %- Top Module -%
 %- XPS SIMULATOR MODULE                                  -%
 %-  1. XPS.CONTROLER : (1). XPS.(This Module)           -%
 %-                     (2). XPS.TRANSLATOR              -%
 %-                     (3). XPS.INITIALISER             -%
 %-                     (4). XPS.DISPATCHER              -%
 %-  2. XPS.PPU_MODULE     (Process Pool Unit)           -%
 %-  3. XPS.CHANNEL_MODULE (Network & I/O Buffer)        -%
 %-  4. XPS.UU_MODULE      (Unification Unit)            -%
 %-  5. XPS.ANALYZER                                     -%
 %-  6. XPS.BUILT_IN       (Built-in Checker)            -%
 %-  7. XPS.UTILITIES                                    -%

 %-------------------------------------------------------------------------%

/*01*/ menu                  %- Top Level.
       :-!,repeat,
           display(MENU),
           read(MENU_No),    %- 1-3.
           execute_SELECTED_MODULE, fail.
                             %- 1. --> Go to XPS.TRANSLATOR  & Return.
                             %- 2. --> Go to XPS.INITIALISER & Not return.
                             %- 3. --> Exit.

/*============================< XPS TRANSLATOR >=============================*/
                                                          %- from XPS. -%
/*01*/ translator
       :-!,read(Input_&_Output_File_Name),
           open(Input_&_Output_File_Name),
           translate,                          %- Translate File.
           close(Input_&_Output_File_Name),!.  %--> Return to XPS.

/*02*/ translate
       :-!,repeat,
           read(Input_Statement),
           translate(Input_Statement),!.

/*03*/ translate(end_of_file)                  %- Check End of File.
       :-!.
/*04*/ translate((define WORLD))               %- Check Module Definition.
       :-!,repeat,
           read(Next_Input_Statement),
           translate(WORLD,Next_Input_Statement),!.
/*05*/ translate(_)                           %- Reject Other Statement.
       :-!,fail.
/*06*/ translate(_,end_of_file)               %- Check End of File.
       :-!.
/*07*/ translate(_,end)                       %- Check End of Module.
       :-!,translate,!.                      %- Translate Next Module.
/*08*/ translate(WORLD,Input_Statement)
       :-!,execute_TRANSLATION(Input_Statement,Output_Statement),
                   %- (HEAD::-BODY) --> clausep([2,WORLD],HEAD,(BODY)).
                   %- (HEAD:- BODY) --> clausep([1,WORLD],HEAD,(BODY)).
                   %-    FACT      --> clausep([1,WORLD],FACT,(true)).
           write(Output_Statement), fail.

/*=========================================================================*/
```

```
/*============================< XPS INITIALISER >============================*/
                                                                  %- from XPS. -%
/*01*/ initialiser
       :-!,reset,
           set_UNIT_No,
           set_TIME_TABLE,
           set_MODE,
           set_CLAUSE_POOL,
           activate_dispatcher,!.      %--> Go to XPS.DISPATCHER & Not return.

/*02*/ reset
       :-!,system_initialise.

/*03*/ set_UNIT_No
       :-!,read(PPU_&_UU_No),
           set_PPU_&_UU_No,!.

/*04*/ set_TIME_TABLE
       :-!,read(Processing_Time_of_Each_Unit),
           set_TIME_TABLE_of_PPU_UU_&_NET,!.

/*05*/ set_MODE
       :-!,set_Analyze_Mode_if_nessary,
           set_Display_Mode_if_nessary,
           set_Step____Mode_if_nessary,
           set_Debug___Mode_if_nessary,
           set_Abort___Cycle,!.

/*06*/ set_CLAUSE_POOL
       :-!,load_Program_File,!.

/*============================< XPS DISPATCHER >============================*/
                                                      %- from XPS.INITIALISER -%
/*01*/ activate_dispatcher
       :-!,repeat,
           set_Initial_Process,        %- Set Query as Initial Process.
           dispatcher,
           fail.                       %- Repeat forever.

/*02*/ dispatcher
       :-!,repeat,
           update_System_Time(TIME),
           channel_module(TIME),       %--> Activate Channel Module.
           ppu_module(TIME),           %--> Activate PPU Module.
           uu_module(TIME),            %--> Activate UU Module.
           analyzer(TIME),             %--> Activate Analyze Module.
           stop_condition(TIME),!.     %- Check Stop Condition.

/*03*/ stop_condition(TIME)            %- Check No Processes.
       :-  call(process(_,_,_,_,_,_)),
          (T is (TIME mod Abort_Cycle), T=0, abort ; fail) ;
           true,!.

/*=========================================================================*/
```

```
/*============================< XPS PPU MODULE >=============================*/
                                                        %- from XPS.DISPATCHER -%
/*01*/ ppu_module(TIME)
       :-!,repeat(Unit_No),                    %- Pick up New Unit No.
           ppu_unit(Unit_No,TIME),             %- Processing of Each PPU Unit.
           ppu_No(Unit_No),!.                  %--> Return to XPS.DISPATCHER.

/*02*/ ppu_unit(UN,TIME)
       :- (check_TIME_TABLE,               %- Check Time-Table & Branch, if Busy.
          (check_Processible_Data_in_PPU(UN,TIME), %- Branch, if No Data.
           garbage_processing(UN,TIME),
           ready_process_processing(UN,TIME),
           input_buffer_processing(UN,TIME),
           correct_TIME_TABLE_to_BUSY  ;
           correct_TIME_TABLE_to_IDLE) ;
           true),!.

/*03*/ input_buffer_processing(No,TIME)
       :- (get_Processible_Data(DATA),
           in_case_of(DATA,TIME) ;         %- Each cases :
                                           %-  1. Fail in OR-Fork.
                                           %-  2. Succeed in OR-Fork.
                                           %-  3. Fail in Subgoal.
                                           %-  4. Succeed in Subgoal
                                           %-       (Create Copy-Process).
                                           %-  5. Create Child-Process.
           true),!.                        %- No Processible Data Case.

/*04*/ ready_process_processing(No,TIME)
       :- (get_Ready_Process_&_change_STATUS_to_WAIT(PROCESS),
           select_Subgoal(PROCESS,SUBGOAL),
           put_Subgoal_into_Output_Buffer(SUBGOAL) ;
           true),!.                        %- No Ready Process.

/*05*/ garbage_processing(No,TIME)
       :- (bagof(X,garbage_condition(No,X),SET),   %- Get Garbage-Processes.
           correct_return_No(SET,TIME) ;
           true),!.                                %- No Garbage-Processes.

/*=========================================================================*/
```

```
/*==========================< XPS CHANNEL MODULE >============================*/
                                                        %- from XPS.DISPATCHER -%
/*01*/ channel_module(TIME)
       :- (check_Processible_Data_in_CHANNEL(TIME),  %- Branch, if No Data.
           ppu_out_buffer_process(TIME),
           uu_out_buffer_process(TIME),
           network_process(TIME),
           correct_TIME_TABLE_to_BUSY ;
           correct_TIME_TABLE_to_IDLE),!.    %--> Return to XPS.DISPATCHER.

/*02*/ network_process(_)
       :- (bagof(DATA_on_NET,DATA_on_NET,Data_Set),  %- Get Data on Net.
           data_processing_on_NET(Data_Set) ;
           true),!.                                  %- No Data.

/*03*/   data_processing_on_NET([])
         :-!.
/*04*/   data_processing_on_NET([TOP|RES])
         :-  get_Packet_&_count_down_Delay_&_if_Delay=0_put_into_PPU/UU(TOP),
             data_processing_on_NET(RES).

/*05*/ ppu_out_buffer_process(TIME)
       :-!,repeat(UN),                                 %- Pick up New Unit No.
           ppu_and_uu_data_processing(ppu,UN,TIME),
           ppu_No(UN),!.

/*06*/ uu_out_buffer_process(TIME)
       :-!,repeat(UN),                                 %- Pick up New Unit No.
           ppu_and_uu_data_processing(uu,UN,TIME),
           uu_No(UN),!.

/*07*/   ppu_and_uu_data_processing(U,UN,TIME)
         :- (get_Data_from_Output_Buffer_&_put_on_NET(TIME) ; true),!.

/*==========================================================================*/
```

```
/*============================< XPS UU MODULE >============================*/
                                                        %- from XPS.DISPATCHER -%

/*01*/ uu_module(TIME)
       :-!,repeat(UN),                          %- Pick up New Unit No.
           uu_unit(UN,TIME),                    %- Processing of Each UU Unit.
           uu_No(UN),!.                         %--> Return to XPS.DISPATCHER.

/*02*/ uu_unit(UN,TIME)
       :- (check_TIME_TABLE,                    %- Check Time-Table & Branch, if Busy.
          (check_Processible_Data_in_UU(UN,TIME), %- Branch, if No Data.
           generate_packet(UN,TIME),
           correct_TIME_TABLE_to_BUSY  ;
           correct_TIME_TABLE_to_WAIT) ;
           true),!.

/*03*/ generate_packet(UN,TIME)
       :- (get_Processible_Data_&_pick_up_Subgoal(Subgoal),
          (built_in(Subgoal),                           %- Check Built-in Predicate.
          (call(Subgoal),                               %- Execute Subgoal(Built-in).
           Candidate=[true]  ;                          %- Succeed in Execution.
           Candidate=[fail]) ;                          %- Fail in Execution.
           select_Candidate(W,Subgoal,Candidate)),  %- Not Built-in Case.
           generate_Result_Packet(UN,Subgoal,Candidate),
           true),!.                                     %- No Processible Data Case.

/*05*/   select_Candidate(W,Subgoal,Candidate)
         :- (bagof(X,Subgoal^unifiable(W,Subgoal,X),Candidate) ;
             suspend_check(W,Subgoal),Candidate=[suspend] ;
             Candidate=[fail]),!.

/*==========================================================================*/
```

```
/*============================< XPS ANALYZER >==============================*/
                                                        %- from XPS.DISPATCHER -%
/*01*/ analyzer(TIME)
       :- (modep(analyze),                  %- Analyze, if analyze-mode.
          (modep(debug),
           display('>>> PROCESS LIST <<<'),listing(process),
           display('>>> SUSPEND LIST <<<'),listing(sp_list),
           display('>>> CHANNEL LIST <<<'),listing(channel) ;
           true),
           analyze_unit(ppu,TIME),
           analyze_unit(uu,TIME),
           analyze_unit(net,TIME) ;
           true),!.                         %--> Return to XPS.DISPATCHER.

/*02*/ analyze_unit(Unit,TIME)
       :-!,repeat(No),                      %- Pick up New Unit No.
           analyze_unit(Unit,No,TIME),      %- Analyze Each Unit.
           unit_No(Unit,No),!.

/*03*/ analyze_unit(ppu,No,T)               %- Analyze PPU.
       :-!,analyze_PROCESSING_TIME,
           analyze_READY_____PROCESSES,
           analyze_WAIT______PROCESSES,
           analyze_GARBAGE___PROCESSES,
           analyze_SUSPENDED__SUBGOALS,
           analyze_DATA_in_INPUT__BUFFER,
           analyze_DATA_in_OUTPUT_BUFFRR,!.

/*04*/ analyze_unit(uu,No,T)               %- Analyze UU.
       :-!,analyze_PROCESSING_TIME,
           analyze_OR_FORK____No,
           analyze_DATA_in_INPUT__BUFFER,
           analyze_DATA_in_OUTPUT_BUFFRR,!.

/*05*/ analyze_unit(net,_,T)               %- Analyze NETWORK.
       :-!,analyze_PROCESSING_TIME,
           analyze_PPU_to_UU__DATA,
           analyze_UU__to_PPU_DATA,
           analyze_PPU_to_PPU_DATA,!.

/*=========================< BUILT IN PREDICATES >==========================*/
                                                        %- from XPS.UU_MODULE -%
/*01*/ built_in(X)
       :- check_BUILT_IN_PREDICATE(X),!. %--> Return to XPS.UU_MODULE.

/*=========================================================================*/
```

4．解析例

本章では、次の２つのテスト・モジュールを本シミュレータで実行した時の解析結果について述べる。

```
/* TEST MODULE 1 */
/*01*/ define factorial.
/*02*/  go :- fact(0,5,X),call((system__time(T),write({T,X}),ttynl)).
/*03*/  fact(X,X,1).
/*04*/  fact(X,Y,Y) :- Y is X+1.
/*05*/  fact(X,Y,Z) :- Y>X+1, MID is (X+Y)/2,
                       fact(X,MID,Z1),fact(MID,Y,Z2),
                       Z is Z1*Z2.
/*06*/ end.
/*07*/ define append.
/*08*/  go :- append(X,Y,[1,2,3,4,5]),
              call((system__time(T),write({T,[X,Y]}),ttynl)).
/*09*/  append([],L,L).
/*10*/  append([CAR|L1],L2,[CAR|L3]) :- append(L1,L2,L3).
/*11*/ end.
```

```
/* TEST MODULE 2 : In case of Preceded Implication */
/*01*/ define factorial.
/*02*/  go :- fact(0,5,X),call((system__time(T),write({T,X}),ttynl)).
/*03*/  fact(X,X,1).
/*04*/  fact(X,Y,Y) :-  Y is X+1.
/*05*/  fact(X,Y,Z) ::- Y>X+1, MID is (X+Y)/2,
                       fact(X,MID,Z1),fact(MID,Y,Z2),
                       Z is Z1*Z2.
/*06*/ end.
/*07*/ define append.
/*08*/  go :- append(X,Y,[1,2,3,4,5]),
              call((system__time(T),write({T,[X,Y]}),ttynl)).
/*09*/  append([],L,L).
/*10*/  append([CAR|L1],L2,[CAR|L3]) ::- append(L1,L2,L3).
/*11*/ end.
```

但し、ここではシミュレータの動作確認も兼ねているので、各Unitの処理時間のパラメータ設定値は解析結果がなるべく早く出るように（本シミュレータはPROLOGでプログラムされており実行時間はそれほど早くない）小さな値を設定した。
従って、この意味で必ずしも現実の値を反映した値とはいえない。
また、表１～４に解がでるまでの時間と実行終了までの時間をそれぞれグラフにしたものを示す。

【注】 (1) 表１は、TEST MODULE 1,2 の"factorial" の実行結果である。
① TEST1 はTEST MODULE 1 の"factorial" 実行結果であり、
PPU/UU台数（横軸）の各組合せに対して、
解が出るまでの時間（縦軸）をグラフにしたものである。
② TEST2 はTEST MODULE 2 の"factorial" 実行結果であり、
PPU/UU台数（横軸）の各組合せに対して、
解が出るまでの時間（縦軸）をグラフにしたものである。
③ TEST3 はTEST MODULE 1 の"factorial" 実行結果であり、
PPU/UU台数（横軸）の各組合せに対して、
実行終了までの時間（縦軸）をグラフにしたものである。
④ TEST4 はTFST MODULE 2 の"factorial" 実行結果であり、
PPU/UU台数（横軸）の各組合せに対して、
実行終了までの時間（縦軸）をグラフにしたものである。

(2) 表２は、TEST MODULE 1 の"append"の実行結果である。
① TEST12は、(PPUの台数)=１，(UUの台数)=１または２の場合である。
② TEST22は、(PPUの台数)=２，(UUの台数)=１または２の場合である。
③ TEST31は、(PPUの台数)=３，(UUの台数)=１の場合である。
④ TEST41は、(PPUの台数)=４，(UUの台数)=１の場合である。

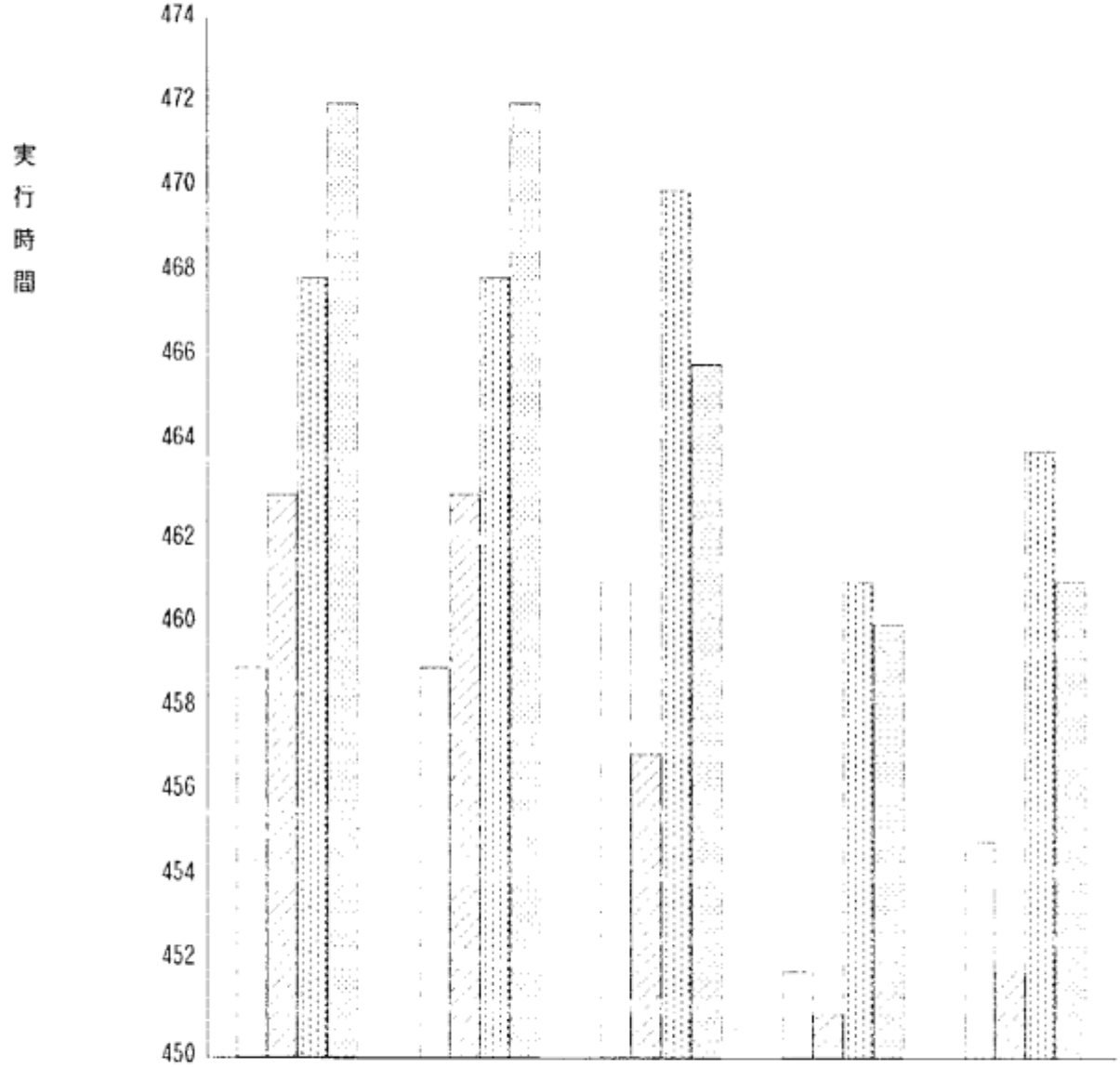
(3) 表３は、TEST MODULE 2 の"append"の実行結果である。
① TEST11は、(PPUの台数)=１，(UUの台数)=１の場合である。
② TEST21は、(PPUの台数)=２，(UUの台数)=１の場合である。
③ TEST31は、(PPUの台数)=３，(UUの台数)=１の場合である。

(4) 表４は、上記(2) と(3) の実行結果の比較である。
但し、比較は (PPU/UUの台数)=1/1 と 3/1の場合についてのみ行なった。
これは単にグラフ作成上の問題のためである。

表1．　Ｆａｃｔｏｒｉａｌの実行時間

（解が出るまでの時間と実行終了までの時間）

単位：サイクル



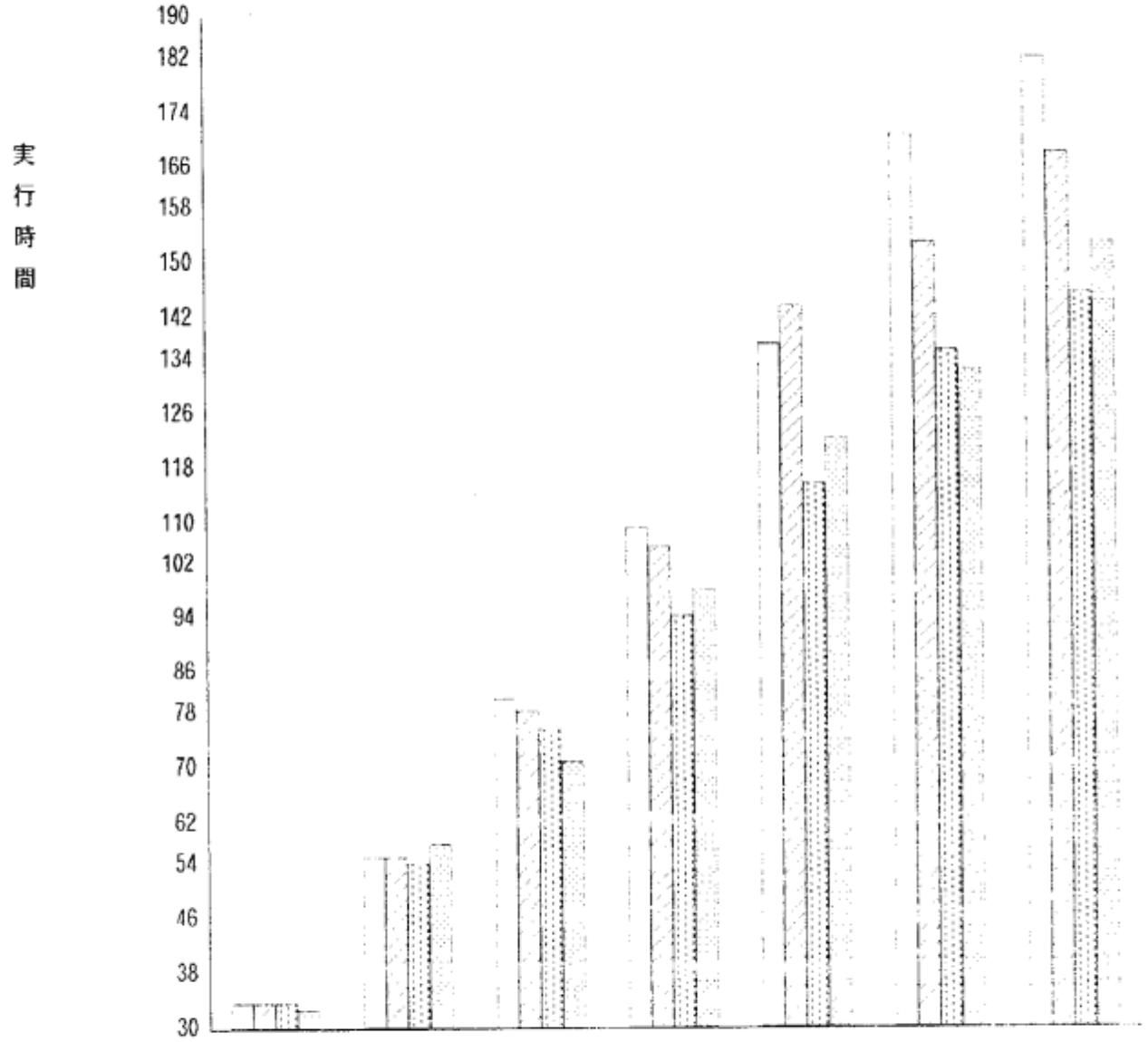
	P1／U1	P1／U2	P2／U1	P2／U2	P2／U3
TEST1	459	459	461	452	455
TEST2	463	463	457	451	452
TEST3	468	468	470	461	464
TEST4	472	472	466	460	461

Ｕｎｉｔ台数

表2．　Append（Module1）の実行時間

（解が出るまでの時間と実行終了までの時間）

単位：サイクル



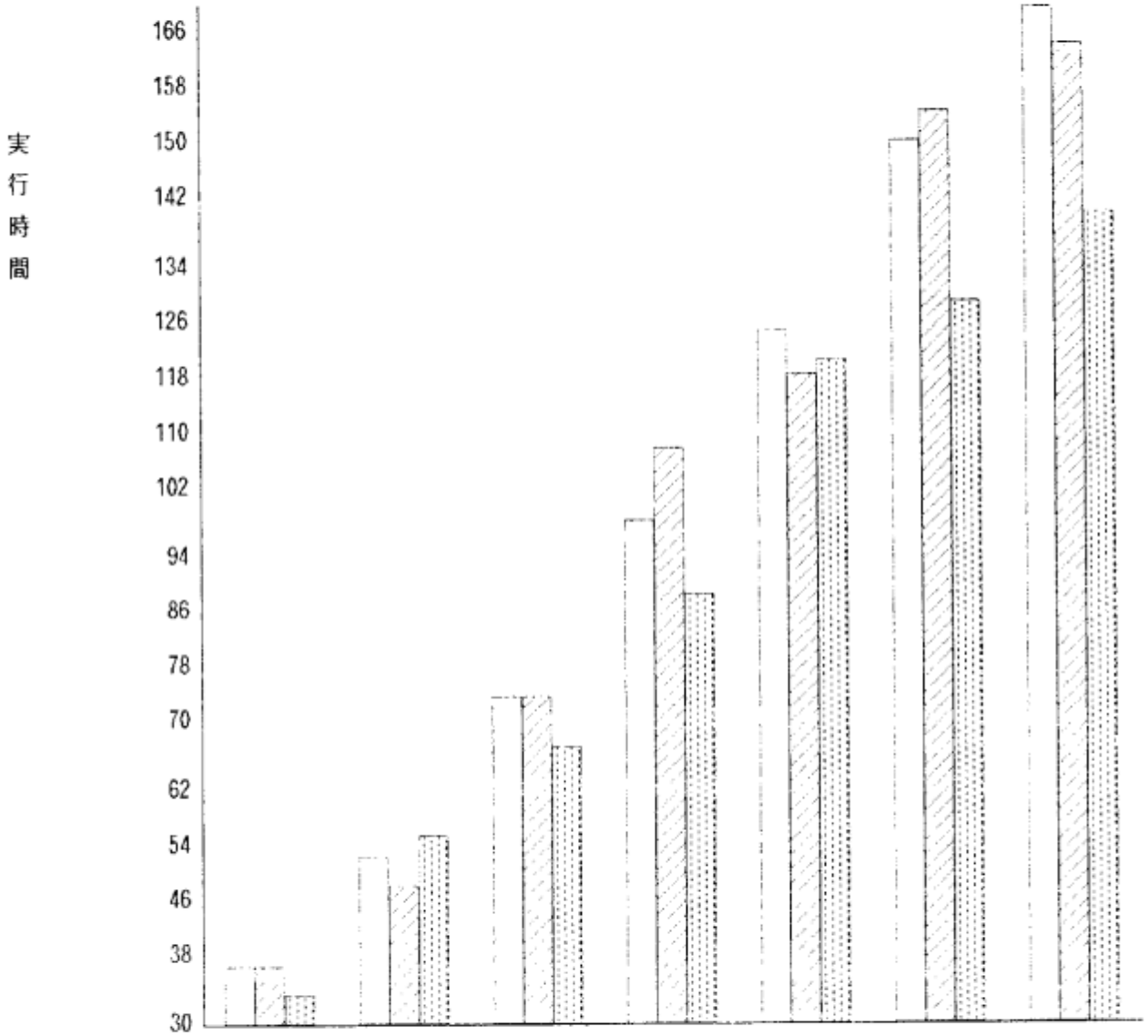
	1番目の解	2番目の解	3番目の解	4番目の解	5番目の解	6番目の解	実行終了時
TEST12	34	57	82	109	138	171	183
TEST22	34	57	80	106	144	154	168
TEST31	34	56	77	95	116	137	146
TEST41	33	59	72	99	123	134	154

解の順番

表3．　Ａｐｐｅｎｄ（Ｍｏｄｕｌｅ２）の実行時間

（解が出るまでの時間と実行終了までの時間）

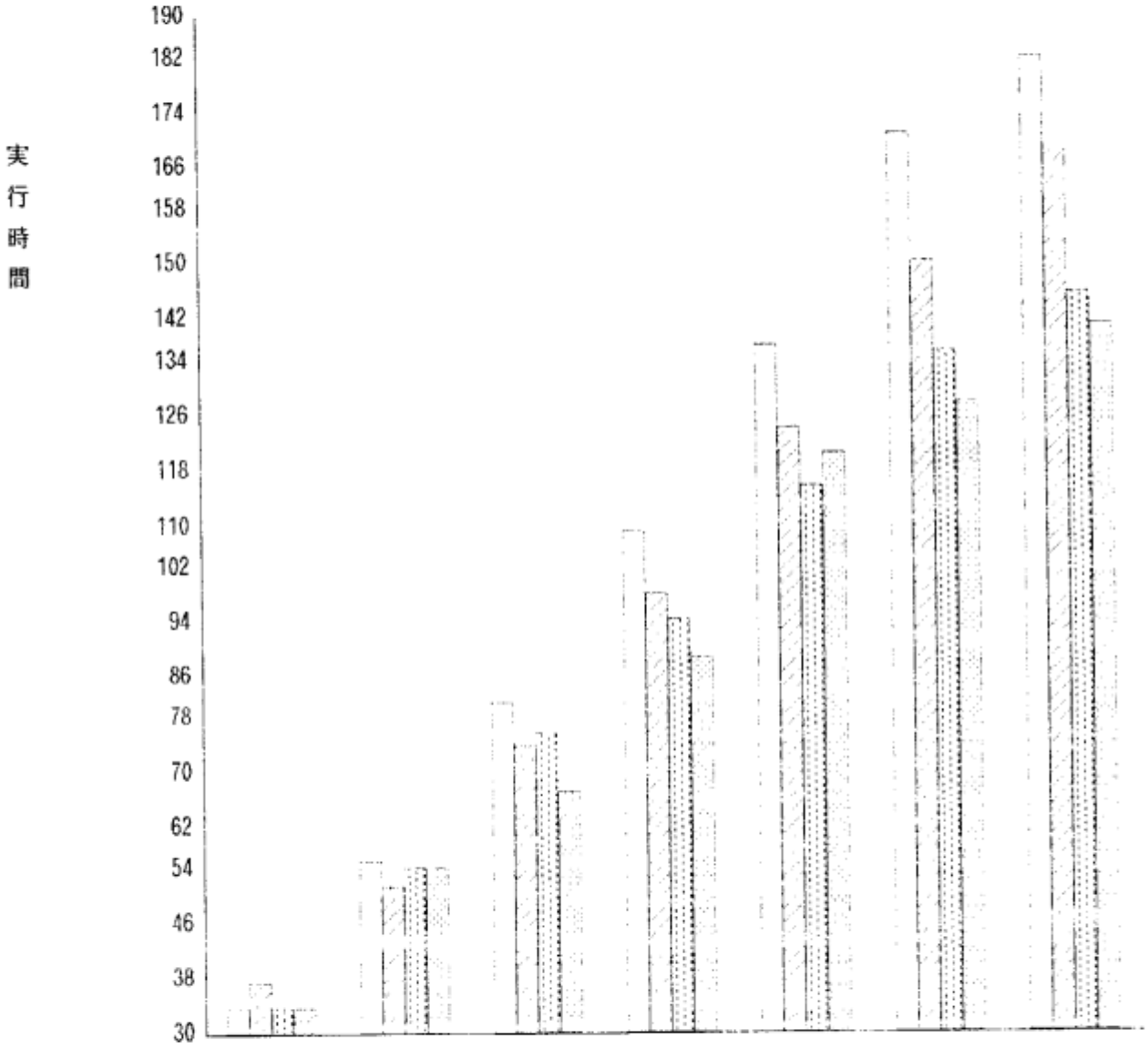
単位：サイクル



	1番目の解	2番目の解	3番目の解	4番目の解	5番目の解	6番目の解	実行終了時
TEST11	38	53	75	99	125	151	169
TEST21	38	49	75	109	119	155	164
TEST31	34	56	68	89	121	129	141

表4．　Appendの実行時間の比較
（Test　Module1と2の比較）

単位：サイクル



	1番目の解	2番目の解	3番目の解	4番目の解	5番目の解	6番目の解	実行終了時
□ M1−1／1	34	57	82	109	138	171	183
M2−1／1	38	53	75	99	125	151	169
M1−3／1	34	56	77	95	116	137	146
M2−3／1	34	56	68	89	121	129	141

解の順番

表をみるとわかるように、

① 各Unitの台数効果
（Network を含めたPPU/UU台数の組合せによる効果の違い等。
但し、プロセスの各PPU への格納戦略は現在ランダムに行なっているので、このデータだけからは一概に結論ずけられない。）
② Preceded Implicationの有効性

が示されているが、その効果の割合はそれほど高くない。
これはシミュレータのパラメータの設定値等によるものと考えられるが、
結論は今後の詳細な評価をまって出すことにする。
各Unitの稼働率，入出力データの平均などを含めたより詳細な解析データを 6.3に示す。

5．おわりに

以上、本レポートでは拡張OR並列PROLOGシステムのソフトウェア・シミュレータとその解析結果について述べた。
今後は AND並列の導入等の機能拡張と改良、具体的応用分野の検討、
および本シミュレータによる詳細な評価等を行っていく予定である。
最後に日頃御指導いただく村上国男第一研究室長はじめ第一研究室諸氏に深謝する。

［参考文献］

[1] 麻生，尾内：“拡張OR並列PROLOGシステム-XP'S-”，Technical Report TR-023，August 1983

6．付録

6.1, 2, 3, に、それぞれXP'S-Simulator(Version-1.0) の Program・List、
シミュレータを実行した時の Logging・List、およびその実行結果である
解析データを示す。

6．1　Program・List

次のページより、XP'S-Simulator(Version-1.0) の Program・Listを示す。

```
/*============================< XPS SIMULATOR >============================*/
 %                                Programmed  by  Moritoshi ASOU.  1983/11/18.
 %- XPS MODULE                                                        -%
 %-  1. XPS.CONTROLER : (1). XPS.(This Module)    -%
 %-                     (2). XPS.TRANSLATOR       -%
 %-                     (3). XPS.INITIALISER      -%
 %-                     (4). XPS.DISPATCHER       -%
 %-  2. XPS.PPU_MODULE     (Process Pool Unit)    -%
 %-  3. XPS.CHANNEL_MODULE (Network & I/O Buffer) -%
 %-  4. XPS.UU_MODULE      (Unification Unit)     -%
 %-  5. XPS.ANALYZER                              -%
 %-  6. XPS.BUILT_IN       (Built-in Checker)     -%
 %-  7. XPS.UTILITIES                             -%

 :- op(300,fx,define).       %- Define Program Module.
 :- op(1200,xfx,'::-').      %- Preceded Implication.
 :- op(450,xf,'?').          %- Read Only Annotation for Concurrent Prolog.
 :- op(1050,xfy,'//').       %- Parallel AND Operator for Concurrent Prolog.

 %- Consult XPS MODULE.
 %--> In this program, there is no public definition on "bagof",
 %-   so some error occur when compile mode. ( See "<--<<<" Mark,etc. )
 %- ['xps.translator', 'xps.initialiser',    'xps.dispatcher',
 %-  'xps.ppu_module', 'xps.channel_module', 'xps.uu_module',
 %-  'xps.analyzer',   'xps.built_in',       'xps.utilities'].

/*01*/ menu                  %- Top Level.
       :-!,repeat,
           display('INPUT No'),
           ttynl,
           display(' ** 1 : XPS.TRANSLATOR'),
           ttynl,
           display(' ** 2 : XPS.INITIALISER'),
           ttynl,
           display(' ** 3 : EXIT'),
           ttynl,
           display(' No='),
           ttyflush,
           ttyget(No),       %- Read selected No.
           run(No),          %- Execute this selected module.
           fail.

/*02*/ run(49)
       :-!,translator,!.     %--> Go to XPS.TRANSLATOR & Return.
/*03*/ run(50)
       :-!,initialiser,!.    %--> Go to XPS.INITIALISER & Not return.
/*04*/ run(51)
       :-!,halt.

/*============================================================================*/
```

```
/*============================< XPS TRANSLATOR >==============================*/
 :- public translator/0.                                          %- from XPS. -%
 %- fastcode.
 %- compactcode.

/*01*/ translator
       :-!,display(' INPUT  FILE NAME [ FORMAT : ''******.***''. ] = '),
           ttyflush,
           read(Input_File_Name),
           display(' OUTPUT FILE NAME [ FORMAT : ''******.***''. ] = '),
           ttyflush,
           read(Output_File_Name),
           see(Input_File_Name),          %- Open Input File.
           tell(Output_File_Name),        %- Open Output File.
           translate,                     %- Translate File.
           seen,                          %- Close Input File.
           told,                          %- Close Output File.
           display(Input_File_Name),
           display(' translated. '),
           ttynl,!.                       %--> Return to XPS.

/*02*/ translate                          %- Translate File.
       :-!,repeat,
           read(Input_Statement),
           translate(Input_Statement),!.

/*03*/ translate(end_of_file)
       :-!.
/*04*/ translate((define WORLD))          %- Check Module Definition.
       :-!,repeat,
           read(Input_Statement),
           translate(WORLD,Input_Statement),!.
/*05*/ translate(_)                       %- Reject Other Statement.
       :-!,fail.

/*06*/ translate(_,end_of_file)
       :-!.
/*07*/ translate(_,end)                   %- Check End of Module.
       :-!,translate,!.                   %- Repeat.
/*08*/ translate(WORLD,Input_Statement)   %- Execute Translation.
       :-!,replace(WORLD,Input_Statement,Output_Statement),
           write(Output_Statement),
           put(46),                       %- '46' = '.'
           put(31),!,                     %- '31' = 'CR'
           fail.

/*============================< XPS REPLACER >================================*/

/*01*/ replace(ID,(HEAD::-BODY),clausep([2,ID],HEAD,(BODY)))
       :-!.                    %- (HEAD:-BODY) --> clausep([2,ID],HEAD,(BODY)).
/*02*/ replace(ID,(HEAD:-BODY),clausep([1,ID],HEAD,(BODY)))
       :-!.                    %- (HEAD:-BODY) --> clausep([1,ID],HEAD,(BODY)).
/*03*/ replace(ID,FACT,clausep([1,ID],FACT,(true)))
       :-!.                    %-    FACT      --> clausep([1,ID],FACT,(true)).

/*============================================================================*/
```

```
/*============================< XPS INITIALISER >============================*/
 :- public initialiser/0.                                        %- from XPS. -%
 %- fastcode.
 %- compactcode.

/*01*/ initialiser
       :-!,reset,
           set_unit_No,
           set_time_table,
           set_mode,
           set_clause_pool,
           activate_dispatcher,!.        %--> Go to XPS.DISPATCHER & Not return.

/*02*/ reset
       :-!,abolish(ppu_No,1),            %- Reset ppu_No(PPN).
           abolish(uu_No,1),             %- Reset uu_No(UN).
           abolish(system_time,1),       %- Reset system_time(T).
           abolish(system_memo,7),       %- Reset system_memo(U,No,Item,NAMT/4).
           abolish(time_table,5),        %- Reset time_table(U,U#,PT,CT,S).
           abolish(modep,1),             %- Reset modep(MODE).
           abolish(process_No,1),        %- Reset process_No(PN).
           abolish(process,6),           %- Reset process(ID,PP,FN,S,H,(B)).
           abolish(sp_list,4),           %- Reset sp_list(PPN,CID,V,LIST).
           abolish(clausep,3),           %- Reset clausep(ID,H,(B)).
           abolish(channel,5),           %- Reset channel(U,No,IO,T,DATA).
           trimcore,!.                   %- Garbage Collection.

/*03*/ set_unit_No
       :-!,display(' [ PPU#, UU# ]. = '),ttyflush,read([PPU,UU]),
           assertz(ppu_No(PPU)),         %- Set ppu_No.
           assertz(uu_No(UU)),!.         %- Set uu_No.

/*04*/ set_time_table
       :-!,set_time_table(ppu),
           set_time_table(uu),
           set_time_table(net),!.

/*05*/   set_time_table(U)
         :-!,repeat(UN),                 %--> Go to XPS.UTILITIES & Return.
             set_time_table(U,UN),
             unit_No(U,UN),!.            %--> Go to XPS.UTILITIES & Return.

/*06*/     set_time_table(U,UN)
           :-!,display(' Processing Time of '),display(U),
              (U=net ; display(' #'),display(UN)),
               display('  (0<PT<10) = '),ttyflush,
               ttyget(PT),               %- Read Processing Time of Each Unit.
               New_PT is PT-48,          %-  and Set Time-Table.
               assertz(time_table(U,UN,New_PT,0,i)),!.
```

```
/*07*/ set_mode
       :-!,display(' Analyze Mode   (Y/N) = '),ttyflush,ttyget(YorN1),
          ((YorN1=89 ; YorN1=121),      %- Read Yes or No,
           assertz(modep(analyze)) ;    %-  and Set Mode when Yes.
           true),
           display(' Interim Report (Y/N) = '),ttyflush,ttyget(YorN2),
          ((YorN2=89 ; YorN2=121),      %- Read Yes or No.
           assertz(modep(display)) ;    %-  and Set Mode when Yes.
           true),
           display(' Step    Mode   (Y/N) = '),ttyflush,ttyget(YorN3),
          ((YorN3=89 ; YorN3=121),      %- Read Yes or No.
           assertz(modep(step)) ;       %-  and Set Mode when Yes.
           true),
           display(' Debug   Mode   (Y/N) = '),ttyflush,ttyget(YorN4),
          ((YorN4=89 ; YorN4=121),      %- Read Yes or No.
           assertz(modep(debug1)) ;     %-  and Set Mode when Yes.
           true),
           display(' Abort   Cycle  (T>0) = '),ttyflush,read(T),
           assertz(modep(abort/T)),!.   %- Set Abort Cycle.

/*08*/ set_clause_pool
       :-!,repeat,
           load_program_file,
           next(true),!.                %--> Go to XPS.UTILITIES & Return.

/*09*/ load_program_file
       :-!,display(' PROGRAM FILE NAME [ FORMAT : ''******.***''. ] = '),
           ttyflush,
           read(Program_File),
           consult(Program_File),!.     %- Set Clause Pool.

/*=============================================================================*/
```

```
/*============================< XPS DISPATCHER >============================*/
 :- public activate_dispatcher/0.                    %- from XPS.INITIALISER -%
 %- fastcode.
 %- compactcode.

/*01*/ activate_dispatcher
       :-!,prompt(_,'>>'),              %- Set New Prompt.
           repeat,
           set_initial_process,         %- Set Query as Initial Process.
           dispatcher,
           fail.                        %- Repeat forever.

/*02*/ set_initial_process
       :-!,display('INPUT GOAL [ FORMAT : goal_frame(WORLD,(GOAL)). ]'),ttynl,
           read(goal_frame(WORLD,GOAL)),
           new_No(New_PN),              %--> Go to XPS.UTILITIES & Return.
           new_No(New_PN1),             %--> Go to XPS.UTILITIES & Return.
           ID=[2,New_PN1,1,WORLD,0,0], %- Set ID & Initial Process.
           assertz(process(ID,(New_PN,1),(_,0),ready,query,GOAL)),!.

/*============================< DISPATCHER >============================*/

/*01*/ dispatcher
       :-!,repeat,
           trimcore,                    %- Garbage Collection.
           time(TIME),                  %--> Go to XPS.UTILITIES & Return.
     /*A*/ channel_module(TIME),        %--> Activate Channel Module.
     /*B*/  ppu_module(TIME),           %--> Activate PPU Module.
     /*C*/   uu_module(TIME),           %--> Activate UU Module.
       %-->   This sequence has an effect on the result of analysis.
           analyzer(TIME),              %--> Activate Analyze Module.
           stop_condition(TIME),!.      %- Check Stop Condition.

/*02*/ stop_condition(T)                %- Check Ready Processes.
       :-  PROCESS=process(_,_,_,_,_,_),
           call(PROCESS),
           T1 is (T mod 10),
          (T=\=0, modep(abort/T2), T3 is (T mod T2), T3=0,
          (modep(display) ; display('TIME = '),display(T),ttynl),!,
           display('### INPUT "dispatcher." TO CONTINUE ###'),ttynl,
           put(7),put(7),put(7),abort ;
          (modep(display) ; T1=0,display('TIME = '),display(T),ttynl ; true),!,
           fail).
/*03*/ stop_condition(T)                %- Stop & New Query.
       :-  display('Final Time = '),display(T),ttynl,
          (modep(analyze),
           display('### Result of Analysis ###'),
           listing(system_memo),ttynl,
           abolish(system_memo,7) ;     %- Reset system_memo(UN,No,Item,NAMT/4).
           true),
           abolish(process_No,1),       %- Reset process_No(PN).
           abolish(process,6),          %- Reset process(ID,PP,FN,S,H,(B)).
           abolish(sp_list,4),          %- Reset sp_list(PPN,CID,V,LIST).
           abolish(channel,5),          %- Reset channel(U,No,IO,T,DATA).
           trimcore,                    %- Garbage Collection.
           put(7),put(7),put(7),put(7),put(7),!.

/*=========================================================================*/
```

```
/*============================< XPS PPU MODULE >============================*/
 :- public ppu_module/1.                                    %- from XPS.DISPATCHER -%
 %- fastcode.
 %- compactcode.

/*01*/ ppu_module(TIME)
       :-!,repeat(UN),
           ppu_unit(UN,TIME),                    %- Processing of Each PPU Unit.
           ppu_No(UN),!.                         %--> Return to XPS.DISPATCHER.

/*02*/ ppu_unit(UN,TIME)
       :- (time_table(ppu,UN,PT,CT,_),                    %- Check Time-Table.
           TIME>=CT,                                      %- Branch, if Busy.
          (check_processible_data_in_ppu(UN,TIME),        %- Branch, if No Data.
     /*A*/ garbage_processing(UN,TIME,0,N1),
     /*B*/  ready_process_processing(UN,TIME,N1,N2),
     /*C*/   input_buffer_processing(UN,TIME,N2,N3),
       %-->   This sequence has an effect on the result of analysis.
           PT1 is PT*N3,
           correct_time_table(ppu,UN,PT1,b); %--> Go to XPS.UTILITIES & Return.
           correct_time_table(ppu,UN,1,i)) ; %--> Go to XPS.UTILITIES & Return.
           true),!.

/*03*/   check_processible_data_in_ppu(UN,TIME)     %- Data in Input-Buffer.
         :-  channel(ppu,UN,in,T,_),T=<TIME,!.
/*04*/   check_processible_data_in_ppu(UN,TIME)     %- Ready Process.
         :-  process([_,_,UN|_],_,_,ready,_,_),!.
/*05*/   check_processible_data_in_ppu(UN,TIME)     %- Garbage Process.
         :-!,process([_,_,UN|_],_,(FN1,FN2),wait,_,_),
             integer(FN1),integer(FN2),FN1==FN2,!.

/*06*/ input_buffer_processing(No,TIME,N1,N2)
       :- (channel(ppu,No,in,T,DATA),T<TIME,
           retract(channel(ppu,No,in,T,DATA)),        %- Select Processible Data.
           in_case_of(DATA,TIME,N1,N2) ;
           N1=N2),!.                                  %- No Processible Data Case.

/*07*/   in_case_of(data1(ID,_,0),_,N1,N2)           %- Fail in OR-Fork(FN=0).
         :-!,retract(process(ID,PP,_,wait,H,B)),     %- Change Parent Process
             assertz(process(ID,PP,(0,0),wait,H,(fail))),  %- into Garbage.
             N2 is N1+1.
/*08*/   in_case_of(data1(ID,_,FN),_,N1,N2)          %- Succeed in OR-Fork(FN>0).
         :-!,retract(process(ID,PP,(_,FN1),wait,H,B)),     %- Set Forked-No to
             assertz(process(ID,PP,(FN,FN1),wait,H,B)),    %-   Parent Process.
             N2 is N1+1,!.
/*09*/   in_case_of(data1(_,(PN,PPN),_,(fail)),_,N1,N2) %- Fail in Subgoal.
         :-!,retract(process([P,PN,PPN|ID],PP,(FN1,FN2),wait,H,B)),
             FN3 is FN2+1,                             %- Count up Return-No.
             assertz(process([P,PN,PPN|ID],PP,(FN1,FN3),wait,H,B)),
             N2 is N1+1,!.
/*10*/   in_case_of(data1(_,(PN,PPN),G,(true)),T,N1,N3) %- Create Copy-Process.
         :-!,retract(process([P,PN,PPN|ID],PP,(FN1,FN2),wait,H,B)),
             FN3 is FN2+1,                             %- Count up Return-No.
             correct_forked_No_of_grand_parent(PPN,PP,T,H,N1,N2),
             assertz(process([P,PN,PPN|ID],PP,(FN1,FN3),wait,H,B)),
             generate_new_process([P,PPN|ID],PP,G,(true),H,B,T,N2,N3),!.
/*11*/   in_case_of(data1([P,_|ID],(PN,PPN),H,B),_,N1,N2) %- Create Child-Proc.
         :-!,generate_new_process([P|ID],(PN,PPN),H,B,_,_,_,N1,N2),!.

                                %- 06 -%
```

```
/*12*/   in_case_of(data2((PN,PPN),_,(fail)),_,N1,N2)   %- Fail in Subgoal.
         :-!,retract(process([P,PN,PPN|ID],PP,(FN1,FN2),wait,H,B)),
             FN3 is FN2+1,                              %- Count up Return-No.
             assertz(process([P,PN,PPN|ID],PP,(FN1,FN3),wait,H,B)),
             N2 is N1+1,!.
/*13*/   in_case_of(data2((PN,PPN),G,(true)),T,N1,N3)   %- Create Copy-Process.
         :-!,retract(process([P,PN,PPN|ID],PP,(FN1,FN2),wait,H,B)),
             FN3 is FN2+1,                              %- Count up Return-No.
             correct_forked_No_of_grand_parent(PPN,PP,T,H,N1,N2),
             assertz(process([P,PN,PPN|ID],PP,(FN1,FN3),wait,H,B)),
             generate_new_process([P,PPN|ID],PP,G,(true),H,B,T,N2,N3),!.
/*14*/   in_case_of(data2((PN,PPN),_,up),_,N1,N2)  %- Count up Forked-No.
         :-!,retract(process([P,PN,PPN|ID],PP,(FN0,FN2),wait,H,B)),
            (var(FN0),abort,! ;                         %-->> For detect BUG!!
             FN1 is FN0+1),                             %- Count up Forked-No.
             assertz(process([P,PN,PPN|ID],PP,(FN1,FN2),wait,H,B)),
             N2 is N1+1,!.

/*15*/     correct_forked_No_of_grand_parent(PPN,PP,T,query,N,N)
           :-!.   %- Parent-Process is Query, so not nessary to correct.
/*16*/     correct_forked_No_of_grand_parent(PPN,(PN,PPN),T,_,N1,N2)
           :-!,retract(process([P,PN,PPN|ID],PP,(FN0,FN2),wait,H,B)),
               FN1 is FN0+1,                            %- Count up Forked-No.
               assertz(process([P,PN,PPN|ID],PP,(FN1,FN2),wait,H,B)),
               N2 is N1+1,!.
                  %- Grand-Parent-Process is in the same PPU.
/*17*/     correct_forked_No_of_grand_parent(PPN,PP,T,_,N1,N2)
           :-!,time_table(ppu,PPN,PT,_,_),          %- Pick up Processing Time.
               N2 is N1+1, T1 is T+(PT*N2),
               assertz(channel(ppu,PPN,out,T1,data2(PP,dummy,up))),!.
                  %- Used dummy to build up data2 to 3 arguments.
                  %- Grand-Parent-Process is in the different PPU.

/*18*/     generate_new_process(_,_,Subgoal,(true),query,Subgoal,_,N,N)
           :-!.   %- Succeed in Subgoal & Query, and not generate Copy-Process.
/*19*/     generate_new_process([_,PPN|_],(PN,PPN),Subgoal,(true),Head,Subgoal,
                                T,N1,N2)
           :-!,in_case_of(data2((PN,PPN),Head,(true)),T,N1,N2),!.
                  %- Succeed in Subgoal & Parent-Process,
                  %-  and then to Grand-Parent-Process in the same PPU.
/*20*/     generate_new_process([_,PPN|_],PP,Subgoal,(true),Head,Subgoal,
                                T,N1,N2)
           :-!,time_table(ppu,PPN,PT,_,_),          %- Pick up Processing Time.
               N2 is N1+1, T1 is T+(PT*N2),
               assertz(channel(ppu,PPN,out,T1,data2(PP,Head,(true)))),!.
                  %- Succeed in Subgoal & Parent-Process,
                  %-  and then to Grand-Parent-Process in the different PPU.
/*21*/     generate_new_process([P|ID],PP,Subgoal,(true),Head,Body,_,N1,N2)
           :-!,and_reduction(Subgoal,Body,New_Body),
               new_No(No),                          %--> Go to XPS.UTILITIES & Return.
               assertz(process([P,No|ID],PP,(_,0),ready,Head,New_Body)),
               N2 is N1+1,!.
                  %- Succeed in Subgoal and then create Copy-Process.
/*22*/     generate_new_process([P|ID],PP,Head,Body,_,_,_,N1,N2)
           :-!,new_No(No),                          %--> Go to XPS.UTILITIES & Return.
               assertz(process([P,No|ID],PP,(_,0),ready,Head,Body)),
               N2 is N1+1,!.                        %- Create Child-Process.
```

```
/*23*/        and_reduction(G,(G,G1),G1)
              :-!.
/*24*/        and_reduction(G,(G1,G2),(New_G1,G2))
              :-!,and_reduction(G,G1,New_G1),!.

/*25*/ ready_process_processing(No,TIME,N1,N2)
       :- ((retract(process([2,PN,No|ID],PP,FN,ready,H,B)),
            P=2 ;                                    %- Get Ready Process-2.
            retract(process([1,PN,No|ID],PP,FN,ready,H,B)),
            P=1),                                    %- Get Ready Process-1.
           assertz(process([P,PN,No|ID],PP,FN,wait,H,B)), %- Change into WAIT.
           time_table(ppu,No,PT,_,_),                %- Pick up Processing Time.
           N2 is N1+1,
           T is TIME+(PT*N2),                        %- T is New Arrival-Time.
           select_subgoal(B,Subgoal),                %- Select & Fork Subgoal.
           assertz(channel(ppu,No,out,T,data1([P,PN,No|ID],Subgoal))) ;
           N2=N1),!.                                          %- No Ready Process.

/*26*/   select_subgoal((G1,G2),G3)
         :-!,select_subgoal(G1,G3),!.
/*27*/   select_subgoal(G,G)
         :-!.

/*28*/ garbage_processing(No,TIME,N1,N2)
       :- (bagof(X,garbage_condition(No,X),SET),   %- Get Garbage-Processes.
           correct_return_No(SET,TIME,N1,N2) ;
           N2=N1),!.                               %- No Garbage-Processes.

/*29*/   garbage_condition(N,X)                    %- ERROR in COMPILED  <--<<<
         :-  X=process([_,_,N|_],_,(FN1,FN2),wait,_,_),!,call(X),
             integer(FN1),integer(FN2),FN1==FN2.

/*30*/   correct_return_No([],_,N,N)
         :-!.
/*31*/   correct_return_No([TOP|RES],T,N1,N3)
         :-!,TOP=process([_,PN,No|_],PP,_,_,H,_),
             retract(TOP),                         %- Garbage!!
             correct_return_No(No,PP,T,H,N1,N2),
             correct_return_No(RES,T,N2,N3),!.

/*32*/     correct_return_No(PPN,PP,T,query,N,N)
           :-!.   %- This Process is Query, so not nessary to correct Parent.
/*33*/     correct_return_No(PPN,(PN,PPN),T,_,N1,N2)
           :-!,retract(process([P,PN,PPN|ID],PP,(FN1,FN2),wait,H,B)),
               FN3 is FN2+1,                       %- Count up Forked-No.
               assertz(process([P,PN,PPN|ID],PP,(FN1,FN3),wait,H,B)),
               N2 is N1+1,!.
                  %- Parent-Process is in the same PPU.
/*34*/     correct_return_No(PPN,PP,T,H,N1,N2)
           :-!,time_table(ppu,PPN,PT,_,_),         %- Pick up Processing Time.
               N2 is N1+1,
               T1 is T+(PT*N2),
               assertz(channel(ppu,PPN,out,T1,data2(PP,H,fail))),!.
                  %- Used dummy to build up data2 to 3 arguments.
                  %- Parent-Process is in the different PPU.

/*=============================================================================*/
```

```
/*=========================< XPS CHANNEL MODULE >============================*/
 :- public channel_module/1.                              %- from XPS.DISPATCHER -%
 %- fastcode.
 %- compactcode.

/*01*/ channel_module(TIME)
       :- (check_processible_data(TIME),      %- Branch, if No Data.
     /*A*/ ppu_out_buffer_process(TIME),
     /*B*/  uu_out_buffer_process(TIME),
     /*C*/   network_process(TIME),
       %-->   This sequence has an effect on the result of analysis.
           correct_time_table(net,1,1,b) ;   %--> Go to XPS.UTILITIES, and then
           correct_time_table(net,1,1,i)),!. %-->  Return to XPS.DISPATCHER.

/*02*/ check_processible_data(_)
       :-  channel(net,_,_,_,_),!.
/*03*/ check_processible_data(TIME)
       :-  channel(ppu,_,out,T,_),T<TIME,!.
/*04*/ check_processible_data(TIME)
       :-  channel(uu,_,out,T,_),T<TIME,!.

/*05*/ network_process(_)
       :-!,DATA=channel(net,_,_,_,_),
          (bagof(DATA,DATA,Data_Set),
           data_processing_on_net(Data_Set) ;
           true),!.

/*06*/   data_processing_on_net([])
         :-!.
/*07*/   data_processing_on_net([TOP|RES])
         :-  retract(TOP),                        %- Get One Packet.
             packet_process(TOP),!,
             data_processing_on_net(RES).

/*08*/     packet_process(channel(net,No,IO,(T,Delay),DATA))
           :- (Delay=0,                           %- Put into PPU/UU, if Delay=0.
               assertz(channel(IO,No,in,T,DATA)) ;
               New_Delay is Delay-1,
               assertz(channel(net,No,IO,(T,New_Delay),DATA))),!.

/*09*/ ppu_out_buffer_process(TIME)
       :-!,repeat(UN),
           ppu_and_uu_data_processing(ppu,UN,TIME),
           ppu_No(UN),!.

/*10*/ uu_out_buffer_process(TIME)
       :-!,repeat(UN),
           ppu_and_uu_data_processing(uu,UN,TIME),
           uu_No(UN),!.

/*11*/   ppu_and_uu_data_processing(U,UN,TIME)
         :-!,(channel(U,UN,out,T,DATA),
              T<TIME,                                %- Branch, if No Data.
              retract(channel(U,UN,out,T,DATA)),     %- Get from Output-Buffer.
              get_on_network(TIME,channel(U,UN,out,T,DATA)) ;
              true),!.
```

```
/*12*/      get_on_network(TIME,channel(ppu,_,out,_,DATA))
            :- (DATA=data1(_,_),                    %- PPU --> UU Case.
                uu_No(UN),
                random(UN,No),                      %--> Go to XPS.UTILITIES & Return.
                IO=uu,! ;
                DATA=data2((_,No),_,_),             %- PPU --> PPU Case.
                IO=ppu),
                time_table(net,1,PT,_,_),           %- Pick up Processing Time.
                T1 is TIME+PT,                      %- T1 is Data-Arrival-Time.
                assertz(channel(net,No,IO,(T1,PT),DATA)),!.
/*13*/      get_on_network(TIME,channel(uu,_,out,_,DATA))
            :- (DATA=data1([P,PN,PPU,W,OL,AL],PP,H,B), %- Child Process Case.
               (var(PPU),                           %- PPU is undefined.
                ppu_No(UN),
                random(UN,No) ;                     %--> Go to XPS.UTILITIES & Return.
                No=PPU),                            %- Already defined.
                New_DATA=data1([P,PN,No,W,OL,AL],PP,H,B),! ;
                DATA=data1([_,_,No,_,_,_],_,_),          %- Forked Result Case.
                New_DATA=DATA),
                time_table(net,1,PT,_,_),           %- Pick up Processing Time.
                T1 is TIME+PT,                      %- T1 is Data-Arrival-Time.
                assertz(channel(net,No,ppu,(T1,PT),New_DATA)),!.

/*=============================================================================*/
```

```
/*============================< XPS UU MODULE >============================*/
 :- public uu_module/1.                                    %- from XPS.DISPATCHER -%
 %- fastcode.
 %- compactcode.

/*01*/ uu_module(TIME)
       :-!,repeat(UN),
           uu_unit(UN,TIME),                    %- Processing of Each UU Unit.
           uu_No(UN),!.                         %--> Return to XPS.DISPATCHER.

/*02*/ uu_unit(UN,TIME)
       :- (time_table(uu,UN,PT,CT,_),                  %- Check Time-Table.
           TIME>=CT,                                   %- Branch, if Busy.
          (check_processible_data_in_uu(UN,TIME),      %- Branch, if No Data.
           generate_packet(UN,TIME),
           correct_time_table(uu,UN,PT,b) ;  %--> Go to XPS.UTILITIES & Return.
           correct_time_table(uu,UN,1,i)) ;  %--> Go to XPS.UTILITIES & Return.
           true),!.

/*03*/   check_processible_data_in_uu(UN,TIME)
         :-!,channel(uu,UN,in,T,_),T<TIME,!.

/*04*/ generate_packet(UN,TIME)
       :- (channel(uu,UN,in,T,data1(ID,Subgoal)),
           T<TIME,                              %- Select Processible Data.
           retract(channel(uu,UN,in,T,data1(ID,Subgoal))),
           ID=[_,_,_,W,_,_],                    %- Pick up Module Name.
          (built_in(Subgoal),                   %- Check Built-in Predicate.
          (call(Subgoal),                       %- Execute Subgoal(=Built-in).
           Candidate=[true]  ;                  %- Succeed in Execution.
           Candidate=[fail]) ;                  %- Fail in Execution.
           select_candidate(W,Subgoal,Candidate)),  %- Not Built-in Case.
           time_table(uu,UN,PT,_,_),            %- Pick up Processing Time.
           T1 is TIME+PT,
           generate_result_packet(UN,T1,ID,Subgoal,Candidate,_),
           trace_UU(ID,Subgoal) ;               %--> Go to XPS.UTILITIES & Return.
           true),!.                             %- No Processible Data Case.

/*05*/   select_candidate(W,Subgoal,Candidate)
         :- (bagof(X,Subgoal^unifiable(W,Subgoal,X),Candidate) ;
             suspend_check([_,W],Subgoal),Candidate=[suspend] ;
             Candidate=[fail]),!.
%-->     If not support Concurrent Prolog & no Preceded Implication, .....
%-->     :-!,CLAUSE=clausep(W,Subgoal,_),
%-->        (bagof(CLAUSE,CLAUSE,Candidate) ;
%-->         Candidate=[fail]),!.

/*06*/     unifiable(W,Subgoal,X)              %- ERROR in COMPILED  <--<<<
           :- (X=clausep([2,W],Head,_) ; X=clausep([1,W],Head,_)),
               call(X),
               unify(by_pass,Subgoal,Head).

/*07*/     suspend_check(W,Subgoal)            %- For Concurrent Prolog.
           :-!,clausep(W,Head,_),\+(\+(unify(suspend,Subgoal,Head))),!.
```

```
/*08*/      unify(_,X,Y)
            :- (var(X) ; var(Y)),!,X=Y.
/*09*/      unify(suspend,X?,Y)              %- For Concurrent Prolog.
            :-  var(X),!.
/*10*/      unify(suspend,X,Y?)              %- For Concurrent Prolog.
            :-  var(Y),!.
/*11*/      unify(Suspend,X?,Y)              %- For Concurrent Prolog.
            :-!,nonvar(X),unify(Suspend,X,Y).
/*12*/      unify(Suspend,X,Y?)              %- For Concurrent Prolog.
            :-!,nonvar(Y),unify(Suspend,X,Y).
/*13*/      unify(_,[],[])
            :-!.
/*14*/      unify(Suspend,[X|Xs],[Y|Ys])
            :-!,unify(Suspend,X,Y),unify(Suspend,Xs,Ys),!.
/*15*/      unify(Suspend,X,Y)
            :-  X=..[Pred|Xs],Y=..[Pred|Ys],
                unify(Suspend,Xs,Ys),!.

/*16*/    generate_result_packet(UN,T,ID,Subgoal,[fail],_)
          :-!,assertz(channel(uu,UN,out,T,data1(ID,Subgoal,0))),!.
/*17*/    generate_result_packet(UN,T,ID,Subgoal,[suspend],_)
          :-!,assertz(channel(uu,UN,out,T,data1(ID,Subgoal,(-1)))),!.
/*18*/    generate_result_packet(UN,T,ID,Subgoal,[true],_)
          :-!,ID=[_,PN,PPN,W,OL,AL],           %- Old ID.
              OL1 is OL+1, AL1 is AL+1,        %- Level Up.
              PP=(PN,PPN),                     %- Create Parent Pointer.
              New_ID=[1,_,PPN,W,OL1,AL1],      %- Create New-ID.
              assertz(channel(uu,UN,out,T,data1(ID,Subgoal,1))),
              assertz(channel(uu,UN,out,T,data1(New_ID,PP,Subgoal,(true)))),!.

/*19*/    generate_result_packet(_,_,_,_,[],_)
          :-!.
/*20*/    generate_result_packet(UN,T,ID,Subgoal,[TOP|RES],By_Pass)
          :- (By_Pass==by_pass ;               %- If not by_pass,
              length([TOP|RES],FN),            %- Create Forked Result Packet.
              assertz(channel(uu,UN,out,T,data1(ID,Subgoal,FN)))),
              TOP=clausep([P,_],Head,Body),
              copy(Subgoal,COPY),              %- Copy Subgoal to avoid Unifying.
              unify(by_pass,COPY,Head),!,      %- Unify after Copy.
%-->      If not support Concurrent Prolog, .....
%-->          COPY=Head,                       %- Unify after Copy.
              ID=[_,PN,PPN,W,OL,AL],             %- Old ID.
              OL1 is OL+1, AL1 is AL+1,        %- Level Up.
              PP=(PN,PPN),                     %- Create Parent Pointer.
             ((Body=(true) ; Body=(fail)),     %- Branch, if Clause is 'Rule'.
              New_ID=[P,_,PPN,W,OL1,AL1] ;     %- Create New-ID.
              New_ID=[P,_,_,W,OL1,AL1]),       %- Create New-ID.
              assertz(channel(uu,UN,out,T,data1(New_ID,PP,Head,Body))),!,
              generate_result_packet(UN,T,ID,Subgoal,RES,by_pass),!.

/*=================================================================*/
```

```
/*============================< XPS ANALYZER >=============================*/
 :- public analyzer/1.                                  %- from XPS.DISPATCHER -%
 %- fastcode.
 %- compactcode.

/*01*/ analyzer(TIME)
       :- (modep(analyze),                %- Analyze, if analyze-mode.
          (modep(debug1),                                    %- For Debug.
           display('>>> PROCESS LIST <<<'),listing(process),ttynl,
           display('>>> SUSPEND LIST <<<'),listing(sp_list),ttynl,
           display('>>> CHANNEL LIST <<<'),listing(channel),ttynl ;
           true),
           analyze_unit(ppu,TIME),
           analyze_unit(uu,TIME),
           analyze_unit(net,TIME) ;
           true),!.                       %--> Return to XPS.DISPATCHER.

/*02*/ analyze_unit(Unit,TIME)
       :-!,repeat(No),
           analyze_unit(Unit,No,TIME),    %- Analyze Each Unit.
           unit_No(Unit,No),!.            %--> Go to XPS.UTILITIES & Return.

/*03*/ analyze_unit(ppu,No,T)                                 %- Analyze PPU.
       :-!,time_table(ppu,No,PT,_,S),                        %- Check Status.
           lengthx(process([_,_,No|_],_,_,ready,_,_),_,RP), %- Count Processes.
           lengthx(process([_,_,No|_],_,(FN1,FN2),wait,_,_),_,(WP,GP)),
           lengthx(sp_list(No,_,_),_,SG),                    %- Count Sus-Subgoal.
           lengthx(channel(ppu,No,in,_,_),T,ID),             %- Count Input Data.
           lengthx(channel(ppu,No,out,_,_),T,OD),            %- Count Output Data.
           operating_time(ppu,No,S,S1,OT),
           compute(ppu,No,ready,RP,ARP,MRP,TRP),
           compute(ppu,No,wait,WP,AWP,MWP,TWP),
           compute(ppu,No,garbage,GP,AGP,MGP,TGP),
           compute(ppu,No,suspend,SG,ASG,MSG,TSG),
           compute(ppu,No,input,ID,AID,MID,TID),
           compute(ppu,No,output,OD,AOD,MOD,TOD),
          (modep(display),
           display('*** PPU #'),display(No),display(' ***'),ttynl,
           display('    CURRENT    TIME       = '),display(T),ttynl,
           display('    PROCESSING TIME       = '),display(PT),ttynl,
           display('    CURRENT    STATUS     = '),write(S1/OT),ttynl,
           display('    READY      PROCESSES  = '),write(RP/ARP/MRP/TRP),ttynl,
           display('    WAIT       PROCESSES  = '),write(WP/AWP/MWP/TWP),ttynl,
           display('    GARBAGE    PROCESSES  = '),write(GP/AGP/MGP/TGP),ttynl,
           display('    SUSPENDED  SUBGOALS   = '),write(SG/ASG/MSG/TSG),ttynl,
           display('    DATA in INPUT  BUFFER = '),write(ID/AID/MID/TID),ttynl,
           display('    DATA in OUTPUT BUFFRR = '),write(OD/AOD/MOD/TOD),ttynl,
           ttynl ; true),!.
```

```
/*04*/ analyze_unit(uu,No,T)                                    %- Analyze UU.
       :-!,time_table(uu,No,PT,_,S),                           %- Check Status.
           lengthx(channel(uu,No,out,T,fork_No),T,FN),         %- Count Fork-No.
           lengthx(channel(uu,No,in,_,_),T,ID),                %- Count Input Data.
           lengthx(channel(uu,No,out,_,_),T,OD),               %- Count Output Data.
           operating_time(uu,No,S,S1,OT),
           compute(uu,No,fork,FN,AFN,MFN,TFN),
           compute(uu,No,input,ID,AID,MID,TID),
           compute(uu,No,output,OD,AOD,MOD,TOD),
          (modep(display),
           display('*** UU #'),display(No),display(' ***'),ttynl,
           display('    CURRENT    TIME       = '),display(T),ttynl,
           display('    PROCESSING TIME       = '),display(PT),ttynl,
           display('    CURRENT    STATUS     = '),write(S1/OT),ttynl,
           display('    OR-FORK    No.        = '),write(FN/AFN/MFN/TFN),ttynl,
           display('    DATA in INPUT  BUFFER = '),write(ID/AID/MID/TID),ttynl,
           display('    DATA in OUTPUT BUFFRR = '),write(OD/AOD/MOD/TOD),ttynl,
           ttynl ; true),!.

/*05*/ analyze_unit(net,_,T)                                    %- Analyze NETWORK.
       :-!,time_table(net,1,PT,_,S),                           %- Check Status.
           lengthx(channel(net,_,uu,_,data1(_,_)),_,PU),       %- Count Data on Net.
           lengthx(channel(net,_,ppu,_,data1),_,UP),           %- (P->U, U->P, P->P)
           lengthx(channel(net,_,ppu,_,data2(_,_,_)),_,PP),
           operating_time(net,1,S,S1,OT),
           compute(net,1,p_u,PU,APU,MPU,TPU),
           compute(net,1,u_p,UP,AUP,MUP,TUP),
           compute(net,1,p_p,PP,APP,MPP,TPP),
          (modep(display),
           display('*** NETWORK ***'),ttynl,
           display('    CURRENT    TIME       = '),display(T),ttynl,
           display('    PROCESSING TIME       = '),display(PT),ttynl,
           display('    CURRENT    STATUS     = '),write(S1/OT),ttynl,
           display('    PPU --> UU  DATA      = '),write(PU/APU/MPU/TPU),ttynl,
           display('    UU  --> PPU DATA      = '),write(UP/AUP/MUP/TUP),ttynl,
           display('    PPU --> PPU DATA      = '),write(PP/APP/MPP/TPP),ttynl,
           ttynl ; true),!.

/*06*/   lengthx(process([_,_,No|_],_,(FN1,FN2),wait,_,_),_,(WP,GP))
         :- (X=process([_,_,No|_],_,_,wait,_,_),
             bagof(X,X,SET1),length(SET1,WGP),                 %- Count WAIT-PRO.
            (bagof(Y,garbage_condition(No,Y),SET2),
                    %-->    \__**-->> Go to XPS.PPU_MODULE & Return.
             length(SET2,GP) ; GP=0),                          %- Count GARBAGE-PRO.
             WP is WGP-GP ;
             WP=0,GP=0),!.

/*07*/   lengthx(sp_list(No,_,_),_,SG)                         %- Count Sus-Subgoal.
         :- (bagof(SPL,ID^sp_list(No,ID,SPL),SPLs),
             lengthy(SPLs,SG) ;
             SG=0),!.

/*08*/     lengthy([],0)
           :-!.
/*09*/     lengthy([TOP|RES],N)
           :-!,length(TOP,N1),lengthy(RES,N2),N is N1+N2,!.
```

```
/*10*/   lengthx(channel(uu,No,out,T,Fork_No),T,FN)        %- Count Fork-No.
         :-  Fork_No==fork_No,
            (channel(uu,No,out,T,data1(_,_,FN)) ; FN=0),!.

/*11*/   lengthx(channel(net,_,ppu,_,data1),_,UPD)         %- Count (U->P) Data.
         :- (X=channel(net,_,ppu,_,data1(_,_,_)),
             bagof(X,X,SET1),length(SET1,UPD1) ;           %- data1/3 type.
             UPD1=0),
            (Y=channel(net,_,ppu,_,data1(_,_,_,_)),
             bagof(Y,Y,SET2),length(SET2,UPD2) ;           %- data1/4 type.
             UPD2=0),
             UPD is UPD1+UPD2,!.

/*12*/   lengthx(channel(UN,No,IO,_,_),T,IOD)              %- Count Data in
         :- (UN==ppu ; UN==uu),!,                          %- PPU/UU I/O Buffer.
            (bagof(X,io_condition(UN,No,IO,T,X),SET),length(SET,IOD) ;
             IOD=0),!.

/*13*/     io_condition(UN,No,IO,T1,X)              %- ERROR in COMPILED  <--<<<
           :-  X=channel(UN,No,IO,T2,_),call(X),T1>=T2.    %- Time Check.

/*14*/   lengthx(X,_,L)                                    %- Count Other Cases.
         :- (bagof(X,X,SET),length(SET,L) ; L=0),!.

/*15*/   operating_time(UN,No,S,S1,Ave1)
         :- (retract(system_memo(UN,No,operating,N,Ave,dummy,Tot)) ;
             N=0, Ave=0, Tot=0),
             N1 is N+1,
            (S=b, S1='BUSY', Cur=1 ; S1='IDLE', Cur=0),
             Tot1 is Tot+Cur, divide(Tot1/N1,Ave1),
             assertz(system_memo(UN,No,operating,N1,Ave1,dummy,Tot1)),!.

/*16*/   compute(uu,No,fork,Cur,Ave1*I1,Max1,Tot1)
         :- (retract(system_memo(uu,No,fork,N1/N2,Ave*I,Max,Tot)) ;
             N1=0, N2=0, Ave=0, I=0, Max=0, Tot=0 ),
             N4 is N2+1,
            (Cur=0,N3=N1 ; N3 is N1+1),divide(N4/N3,I1),
             Tot1 is Tot+Cur, divide(Tot1/N4,Ave1),
            (Cur>Max, Max1=Cur ; Max1=Max),
             assertz(system_memo(uu,No,fork,N3/N4,Ave1*I1,Max1,Tot1)),!.
/*17*/   compute(UN,No,Item,Cur,Ave1,Max1,Tot1)
         :- (retract(system_memo(UN,No,Item,N,Ave,Max,Tot)) ;
             N=0, Ave=0, Max=0, Tot=0 ),
             N1 is N+1, %--> When except '0' in AVE.  (Cur=0,N1=N ; N1 is N+1),
             Tot1 is Tot+Cur, divide(Tot1/N1,Ave1),
            (Cur>Max, Max1=Cur ; Max1=Max),
             assertz(system_memo(UN,No,Item,N1,Ave1,Max1,Tot1)),!.

/*18*/     divide(X/Y,Z)
           :- (Y=0,Z='0.00' ;
               Z1 is X/Y,
               Z2 is (X mod Y)*10/Y,
               Z3 is (((X mod Y)*10) mod Y)*10/Y,
               name(Z1,Z4),name(Z2,Z5),name(Z3,Z6),
               append(Z4,[46|Z5],Z7),append(Z7,Z6,Z8),name(Z,Z8)),!.

/*=================================================================*/
```

```
/*============================< BUILT IN PREDICATES >============================*/
 :- public built_in/1.                                    %- from XPS.UU_MODULE -%
 :- mode   built_in(+).
 %- fastcode.
 %- compactcode.

/* TOP */  built_in(X) :- built_in1(X),!.   %--> Return to XPS.UU_MODULE.

/*============================< CATEGORIZATION >=============================*/

% [01]   Input / Output

%      (1)  Reading-in Programs

            built_in1(consult(_)).          built_in1(reconsult(_)).
            built_in1([_|_]).

%      (2)  File Handling

            built_in1(see(_)).              built_in1(seeing(_)).
            built_in1(seen).                built_in1(tell(_)).
            built_in1(telling(_)).          built_in1(told).
            built_in1(close(_)).            built_in1(fileerrors).
            built_in1(nofileerrors).        built_in1(rename(_,_)).
            built_in1(log).                 built_in1(nolog).
            built_in1(end_of_file).

%      (3)  Input and Output of Terms

            built_in1(read(_)).             built_in1(write(_)).
            built_in1(display(_)).          built_in1(writeq(_)).
            built_in1(print(_)).

%      (4)  Character Input / Output

            built_in1(nl).                  built_in1(get0(_)).
            built_in1(get(_)).              built_in1(skip(_)).
            built_in1(put(_)).              built_in1(tab(_)).
            built_in1(ttynl).               built_in1(ttyflush).
            built_in1(ttyget0(_)).          built_in1(ttyget(_)).
            built_in1(ttyskip(_)).          built_in1(ttyput(_)).
            built_in1(ttytab(_)).           built_in1(ttywait).

% [02]   Arithmetic

            built_in1(_+_).                 built_in1(_-_).
            built_in1(_*_).                 built_in1(_/_).
            built_in1(_ mod _).             built_in1(-_).
            built_in1(_/\_).                built_in1(_\/_).
            built_in1(\(_)).                built_in1(_<<_).
            built_in1(_>>_).                built_in1(!(_)).
            built_in1($(_)).                built_in1(_ is _).
            built_in1(_=:=_).               built_in1(_=\=_).
            built_in1(_<_).                 built_in1(_>_).
            built_in1(_=<_).                built_in1(_>=_).
```

```
% [03]   Comparison of Terms

          built_in1(_==_).                  built_in1(_\==_).
          built_in1(_@<_).                  built_in1(_@>_).
          built_in1(_@=<_).                 built_in1(_@>=_).
          built_in1(compare(_,_,_)).        built_in1(sort(_,_)).
          built_in1(keysort(_,_)).


% [04]   Convenience

          built_in1((_,_)).                 built_in1((_;_)).
          built_in1(true).                  built_in1(fail).
          built_in1(_=_).                   built_in1(length(_,_)).

% [05]   Extra Control

          built_in1(!).                     built_in1(\+_).
          built_in1((_->_)).                built_in1(repeat).

% [06]   Information about the state of the Program

          built_in1(listing).               built_in1(listing(_)).
          built_in1(numbervars(_,_,_)).     built_in1(ancestors(_)).
          built_in1(subgoal_of(_)).         built_in1(current_atom(_)).
          built_in1(current_functor(_,_)). built_in1(current_predicate(_,_)).

% [07]   Meta-Logical

          built_in1(var(_)).                built_in1(nonvar(_)).
          built_in1(atom(_)).               built_in1(integer(_)).
          built_in1(atomic(_)).             built_in1(functor(_,_,_)).
          built_in1(arg(_,_,_)).            built_in1(_=..._).
          built_in1(name(_,_)).             built_in1(call(_)).

% [08]   Modification of the Program

          built_in1(assert(_)).             built_in1(asserta(_)).
          built_in1(assertz(_)).            built_in1(clause(_,_)).
          built_in1(retract(_)).            built_in1(abolish(_,_)).

% [09]   Internal Database

          built_in1(recorded(_,_,_)).       built_in1(recorda(_,_,_)).
          built_in1(recordz(_,_,_)).        built_in1(erase(_)).
          built_in1(instance(_,_)).         built_in1(assert(_,_)).
          built_in1(asserta(_,_)).          built_in1(assertz(_,_)).
          built_in1(clause(_,_,_)).

% [10]   Sets

          built_in1(setof(_,_,_)).          built_in1(bagof(_,_,_)).
          built_in1(_^_).
```

```
% [11]  Compiled Program

          built_in1(compile(_)).          built_in1(incore(_)).
          built_in1(revive(_,_)).
          built_in1((mode _)).            %- Not an Evaluable Predicate.
          built_in1((public _)).          %- Not an Evaluable Predicate.
          built_in1(fastcode).            %- Not an Evaluable Predicate.
          built_in1(compactcode).         %- Not an Evaluable Predicate.

% [12]  Debugging

          built_in1(unknown(_,_)).        built_in1(debug).
          built_in1(nodebug).             built_in1(trace).
          built_in1(leash(_)).            built_in1(spy _).
          built_in1(nospy _).             built_in1(debugging).

% [13]  Definite Clause Grammars

          built_in1((_-->_)).             built_in1(expand_term(_,_)).
          built_in1(phrase(_,_)).

% [14]  Environmental

          built_in1(halt).                built_in1('NOLC').
          built_in1('LC').                built_in1(op(_,_,_)).
          built_in1(current_op(_,_,_)).   built_in1(break).
          built_in1(abort).               built_in1(save(_)).
          built_in1(save(_,_)).           built_in1(restore(_)).
          built_in1(reinitialise).        built_in1(maxdepth(_)).
          built_in1(depth(_)).            built_in1(gcguide(_,_,_)).
          built_in1(gc).                  built_in1(nogc).
          built_in1(trimcore).            built_in1(statistics).
          built_in1(statistics(_,_)).     built_in1(prompt(_,_)).
          built_in1(version).             built_in1(version(_)).
          built_in1(plsys(_)).            built_in1(core_image).
          built_in1(run(_,_)).            built_in1(tmpcor(_,_,_)).

/*============================================================================*/
```

```
/*============================< XPS UTILITIES >==============================*/
 %- fastcode.                                              %- from EACH MODULE -%
 %- compactcode.
 :- public append/3, bagof_1/3, copy/2, correct_time_table/4, new_No/1, next/1,
           random/2, repeat/1,  time/1, trace_UU/2,           unit_No/2.

/*01*/ append([],L,L)
       :-!.
/*02*/ append([CAR|L1],L2,[CAR|L3])
       :-!,append(L1,L2,L3).

/*03*/ bagof_1(X,Y,Z)
       :-!,bagof(X,Y,Z),! ; Z=[],!.

/*04*/ copy(OLD,NEW)
       :-!,abolish(system_work,1),
           assert(system_work(OLD)),
           retract(system_work(NEW)),!.

/*05*/ correct_time_table(UN,No,UP,STATUS)
       :-!,retract(time_table(UN,No,PT,T,_)),
           T1 is T+UP,
           assertz(time_table(UN,No,PT,T1,STATUS)),!.

/*06*/ new_No(New_No)
       :-!,(retract(process_No(Old_No)) ; Old_No is 0),
           New_No is Old_No+1,
           asserta(process_No(New_No)),!.

/*07*/ next(COMMAND)
       :-!,display(' NEXT FILE ?  (Y/N) = '),
           ttyflush,
           ttyget(Y_or_N),!,
           next(Y_or_N,COMMAND),!.
/*08*/ next(89,_)
       :-!,fail.
/*09*/ next(121,_)
       :-!,fail.
/*10*/ next(_,COMMAND)
       :-!,COMMAND.

/*11*/ random(R,N)
       :-!,(retract(random_seed(S)) ; S is 13),
           N is (S mod R)+1,
           New_S is (125*S+1) mod 4096,
           asserta(random_seed(New_S)),!.

/*12*/ repeat(1).
/*13*/ repeat(M)
       :-!,repeat(N),M is N+1.

/*14*/ time(T)
       :-!,(retract(system_time(T)) ; T is 0),
           T1 is T+1,
           asserta(system_time(T1)),!.
```

```
/*15*/ trace_UU(ID,Subgoal)
       :- (modep(step),ID=[_,P,Q|_],
           process(ID,PP,FN,S,H,B),
           channel(uu,No,out,T,data1(ID,Subgoal,FN1)),
           X=data1(_,(P,Q),Subgoal,_),
          (bagof(X,channel(uu,No,out,T,X),SET) ; SET=[]),
           display('### PARENT PROCESS ###'),ttynl,
           display('    '),display(process(ID,PP,FN,S,H,B)),ttynl,
           display('###   INPUT DATA   ###'),ttynl,
           display('      '),display(data1(ID,Subgoal)),ttynl,
           display('###   OUTPUT DATA  ###'),ttynl,
           display('      '),display(data1(ID,Subgoal,FN1)),ttynl,ttynl,
           display_data1(SET),
           display(' NEXT ? (Y/N) = '),ttyflush,
           ttyget(YorN),
          (next(YorN,retract(modep(step))) ; true) ;
           true),!.

/*16*/   display_data1([])
         :-!,ttynl.
/*17*/   display_data1([TOP|RES])
         :-!,display('      '),display(TOP),ttynl,
             display_data1(RES).

/*18*/ unit_No(ppu,PPU)
       :-!,ppu_No(PPU),!.
/*19*/ unit_No(uu,UU)
       :-!,uu_No(UU),!.
/*20*/ unit_No(net,1)
       :-!.

/*=============================================================================*/
```

6．2　Logging・List

次のページより、シミュレータを実行した時の Logging・Listを示す。

```
############ LOGGING LIST ############

XP'S Simulator  version 1.0
Programmed  by  Moritoshi Asou,  1983/11/18.

| ?- ['xps.simulator'].

xps.simulator consulted   11291 words     14.60 sec.

yes
| ?- menu.                          <-- Top Level Command.
INPUT No
 ** 1 : XPS.TRANSLATOR
 ** 2 : XPS.INITIALISER
 ** 3 : EXIT
 No=1                              <-- Select Menu No. & Input File Name.
 INPUT  FILE NAME [ FORMAT : '******.***'. ] = 'xps_test.in'.
 OUTPUT FILE NAME [ FORMAT : '******.***'. ] = 'xps_test.out'.
xps_test.in translated.
INPUT No
 ** 1 : XPS.TRANSLATOR
 ** 2 : XPS.INITIALISER
 ** 3 : EXIT
 No=2                              <-- Select Menu No. & Input Each Processing Time.
 [ PPU#, UU# ]. = [2,3].
 Processing Time of ppu #1  (0<PT<10) = 2
 Processing Time of ppu #2  (0<PT<10) = 3
 Processing Time of uu #1  (0<PT<10) = 2
 Processing Time of uu #2  (0<PT<10) = 3
 Processing Time of uu #3  (0<PT<10) = 4
 Processing Time of net  (0<PT<10) = 1
 Analyze Mode   (Y/N) = y      <-- Set Each Mode & Input Program File.
 Interim Report (Y/N) = y
 Step    Mode   (Y/N) = y
 Debug   Mode   (Y/N) = y
 Abort   Cycle  (T>0) = 100.
 PROGRAM FILE NAME [ FORMAT : '******.***'. ] = 'xps_test.out'.

xps_test.out consulted   705 words      0.83 sec.
 NEXT FILE ?  (Y/N) = n
INPUT GOAL [ FORMAT : goal_frame(WORLD,(GOAL)). ]
>>goal_frame(append,go).       <-- Input Query(Goal) : ?- append(X,Y,[1,2,3]).
```

```
>>> PROCESS LIST <<<          <-- Display when Analyze & Debug Mode.
process([2,2,1,append,0,0],(1','1),(_1','0),wait,query,go).

>>> SUSPEND LIST <<<          <-- Display when Analyze & Debug Mode.
>>> CHANNEL LIST <<<          <-- Display when Analyze & Debug Mode.
channel(ppu,1,out,2,data1([2,2,1,append,0,0],go)).

*** PPU #1 ***                <-- Display when Analyze & Interim Mode.
    CURRENT    TIME       = 0
    PROCESSING TIME       = 2
    CURRENT    STATUS     = BUSY/1.00
    READY      PROCESSES  = 0/0.00/0/0
    WAIT       PROCESSES  = 1/1.00/1/1
    GARBAGE    PROCESSES  = 0/0.00/0/0
    SUSPENDED  SUBGOALS   = 0/0.00/0/0
    DATA in INPUT  BUFFER = 0/0.00/0/0
    DATA in OUTPUT BUFFRR = 0/0.00/0/0

*** PPU #2 ***                <-- Display when Analyze & Interim Mode.
    CURRENT    TIME       = 0
    PROCESSING TIME       = 3
    CURRENT    STATUS     = IDLE/0.00
    READY      PROCESSES  = 0/0.00/0/0
    WAIT       PROCESSES  = 0/0.00/0/0
    GARBAGE    PROCESSES  = 0/0.00/0/0
    SUSPENDED  SUBGOALS   = 0/0.00/0/0
    DATA in INPUT  BUFFER = 0/0.00/0/0
    DATA in OUTPUT BUFFRR = 0/0.00/0/0

*** UU #1 ***                 <-- Display when Analyze & Interim Mode.
    CURRENT    TIME       = 0
    PROCESSING TIME       = 2
    CURRENT    STATUS     = IDLE/0.00
    OR-FORK    No.        = 0/(0.00*0.00)/0/0
    DATA in INPUT  BUFFER = 0/0.00/0/0
    DATA in OUTPUT BUFFRR = 0/0.00/0/0

*** UU #2 ***                 <-- Display when Analyze & Interim Mode.
    CURRENT    TIME       = 0
    PROCESSING TIME       = 3
    CURRENT    STATUS     = IDLE/0.00
    OR-FORK    No.        = 0/(0.00*0.00)/0/0
    DATA in INPUT  BUFFER = 0/0.00/0/0
    DATA in OUTPUT BUFFRR = 0/0.00/0/0

*** UU #3 ***                 <-- Display when Analyze & Interim Mode.
    CURRENT    TIME       = 0
    PROCESSING TIME       = 4
    CURRENT    STATUS     = IDLE/0.00
    OR-FORK    No.        = 0/(0.00*0.00)/0/0
    DATA in INPUT  BUFFER = 0/0.00/0/0
    DATA in OUTPUT BUFFRR = 0/0.00/0/0

*** NETWORK ***               <-- Display when Analyze & Interim Mode.
    CURRENT    TIME       = 0
    PROCESSING TIME       = 1
    CURRENT    STATUS     = IDLE/0.00
    PPU --> UU  DATA      = 0/0.00/0/0
    UU  --> PPU DATA      = 0/0.00/0/0
    PPU --> PPU DATA      = 0/0.00/0/0
```

```
    :
    :
### PARENT PROCESS ###          <-- Display & Stop when Step Mode.
    process([2,2,1,append,0,0],,(1,1),,(_1139,0),wait,query,go)
###   INPUT DATA   ###
      data1([2,2,1,append,0,0],go)
###   OUTPUT DATA  ###
      data1([2,2,1,append,0,0],go,1)

      data1([1,_1156,_1157,append,1,1],,(2,1),go,,(append(_1158,_1159,[1,2,3]),
call(,(system_time(_1160),,(write({}(,(_1160,[_1158,_1159]))),ttynl)))))

 NEXT ? (Y/N) = n
    :
    :
{35,[[],[1,2,3]]}             <-- First  Solution.
    :
{59,[[1],[2,3]]}              <-- Second Solution.
    :
{85,[[1,2],[3]]}              <-- Third  Solution.
    :
### INPUT "dispatcher." TO CONTINUE ###
[ Execution aborted ]         <-- Aborted if (SYSTEM_TIME mod ABORT_CYCLE)=0.

| ?- dispatcher.              <-- Input "dispatcher." to continue.
    :
{110,[[1,2,3],[]]}            <-- Fourth Solution.
    :
Final Time = 121
### Result of Analysis ###    <-- Display when Analyze Mode.
system_memo(ppu,1,operating,120,'0.76',dummy,92).
system_memo(ppu,1,ready,120,'0.22',1,27).
system_memo(ppu,1,wait,120,'2.92',5,351).
system_memo(ppu,1,garbage,120,'0.15',1,18).
system_memo(ppu,1,suspend,120,'0.00',0,0).
system_memo(ppu,1,input,120,'0.39',2,47).
system_memo(ppu,1,output,120,'0.07',1,9).
system_memo(ppu,2,operating,120,'0.25',dummy,30).
system_memo(ppu,2,ready,120,'0.01',1,2).
system_memo(ppu,2,wait,120,'0.40',1,48).
system_memo(ppu,2,garbage,120,'0.02',1,3).
system_memo(ppu,2,suspend,120,'0.00',0,0).
system_memo(ppu,2,input,120,'0.09',1,11).
system_memo(ppu,2,output,120,'0.05',1,6).
system_memo(uu,1,operating,120,'0.01',dummy,2).
system_memo(uu,1,fork,1/120,'0.01'*'120.00',2,2).
system_memo(uu,1,input,120,'0.00',1,1).
system_memo(uu,1,output,120,'0.05',3,6).
system_memo(uu,2,operating,120,'0.11',dummy,14).
system_memo(uu,2,fork,5/120,'0.05'*'24.00',2,7).
system_memo(uu,2,input,120,'0.04',1,5).
system_memo(uu,2,output,120,'0.17',3,21).
system_memo(uu,3,operating,120,'0.10',dummy,12).
system_memo(uu,3,fork,3/120,'0.02'*'40.00',1,3).
system_memo(uu,3,input,120,'0.02',1,3).
system_memo(uu,3,output,120,'0.07',2,9).
system_memo(net,1,operating,120,'0.49',dummy,59).
system_memo(net,1,p_u,120,'0.07',1,9).
system_memo(net,1,u_p,120,'0.17',1,21).
system_memo(net,1,p_p,120,'0.06',1,8).
```

```
--> [NOTE]
  (01) operating : Operating Time Ratio.
  (02) ready     : Ave./Max./Tot. Number of Ready   Processes.
  (03) wait      : Ave./Max./Tot. Number of Wait    Processes.
  (04) garbage   : Ave./Max./Tot. Number of Garbage Processes.
  (05) suspend   : Ave./Max./Tot. Number of Suspended Subgoal.
  (06) input     : Ave./Max./Tot. Number of Data in Input  Buffer of Each Unit.
  (07) output    : Ave./Max./Tot. Number of Data in Output Buffer of Each Unit.
  (08) fork      : Ave./Max./Tot. Number of OR-Forked Processes.
  (09) p_u       : Ave./Max./Tot. Number of Packet from PPU to UU.
  (10) u_p       : Ave./Max./Tot. Number of Packet from UU  to PPU.
  (11) p_p       : Ave./Max./Tot. Number of Packet from PPU to PPU.

yes
| ?-
```

```
############ SOURCE PROGRAM ############

/*01*/ define append.
/*02*/  go :- append(X,Y,[1,2,3]),
              call((system_time(T),write({T,[X,Y]}),ttynl)).
/*03*/  append([],L,L).
/*04*/  append([CAR|L1],L2,[CAR|L3]) ::- append(L1,L2,L3).
/*05*/ end.

############ OBJECT PROGRAM ############

clausep([1,append],go,(append(_346,_367,[1,2,3]),call((system_time(_421),
                       write({_421,[_346,_367]}),ttynl)))).
clausep([1,append],append([],_342,_342),true).
clausep([2,append],append([_342|_358],_379,[_342|_415]),
                   append(_358,_379,_415)).
```

６．３　解析データ

次のページより、XP'S-Simulator(Version-1.0) を実行して得た解析データを示す。
実行したプログラムは第４章に示した２つのテスト・モジュールである。

```
 [ PPU#, UU# ]. = [1,1].
 Processing Time of ppu #1  (0<PT<10) = 2
 Processing Time of uu #1  (0<PT<10) = 2
 Processing Time of net  (0<PT<10) = 1
 PROGRAM FILE NAME [ FORMAT : '******.***'. ] = 'xps_test1.out'.
>>goal_frame(append,go).
{34,[[],[1,2,3,4,5]]}
{57,[[1],[2,3,4,5]]}
{82,[[1,2],[3,4,5]]}
{109,[[1,2,3],[4,5]]}
{138,[[1,2,3,4],[5]]}
{171,[[1,2,3,4,5],[]]}
Final Time = 183
### Result of Analysis ###
system_memo(ppu,1,operating,184,'0.84',dummy,155).
system_memo(ppu,1,ready,184,'0.41',1,76).
system_memo(ppu,1,wait,184,'4.46',8,821).
system_memo(ppu,1,garbage,184,'0.21',1,40).
system_memo(ppu,1,suspend,184,'0.00',0,0).
system_memo(ppu,1,input,184,'0.50',2,92).
system_memo(ppu,1,output,184,'0.07',1,13).
system_memo(uu,1,operating,184,'0.14',dummy,26).
system_memo(uu,1,fork,13/184,'0.09'*'14.15',2,18).
system_memo(uu,1,input,184,'0.07',1,13).
system_memo(uu,1,output,184,'0.29',3,54).
system_memo(net,1,operating,184,'0.35',dummy,65).
system_memo(net,1,p_u,184,'0.07',1,13).
system_memo(net,1,u_p,184,'0.16',1,31).
system_memo(net,1,p_p,184,'0.00',0,0).
```

```
 [ PPU#, UU# ]. = [1,2].
 Processing Time of ppu #1  (0<PT<10) = 2
 Processing Time of uu #1  (0<PT<10) = 2
 Processing Time of uu #2  (0<PT<10) = 2
 Processing Time of net  (0<PT<10) = 1
 PROGRAM FILE NAME [ FORMAT : '*******.***'. ] = 'xps_test1.out'.
>>goal_frame(append,go).
{34,[[],[1,2,3,4,5]]}
{57,[[1],[2,3,4,5]]}
{82,[[1,2],[3,4,5]]}
{109,[[1,2,3],[4,5]]}
{138,[[1,2,3,4],[5]]}
{171,[[1,2,3,4,5],[]]}
Final Time = 183
### Result of Analysis ###
system_memo(ppu,1,operating,184,'0.84',dummy,155).
system_memo(ppu,1,ready,184,'0.41',1,76).
system_memo(ppu,1,wait,184,'4.46',8,821).
system_memo(ppu,1,garbage,184,'0.21',1,40).
system_memo(ppu,1,suspend,184,'0.00',0,0).
system_memo(ppu,1,input,184,'0.50',2,92).
system_memo(ppu,1,output,184,'0.07',1,13).
system_memo(uu,1,operating,184,'0.05',dummy,10).
system_memo(uu,1,fork,5/184,'0.03'*'36.80',2,7).
system_memo(uu,1,input,184,'0.02',1,5).
system_memo(uu,1,output,184,'0.11',3,21).
system_memo(uu,2,operating,184,'0.08',dummy,16).
system_memo(uu,2,fork,8/184,'0.05'*'23.00',2,11).
system_memo(uu,2,input,184,'0.04',1,8).
system_memo(uu,2,output,184,'0.17',3,33).
system_memo(net,1,operating,184,'0.35',dummy,65).
system_memo(net,1,p_u,184,'0.07',1,13).
system_memo(net,1,u_p,184,'0.16',1,31).
system_memo(net,1,p_p,184,'0.00',0,0).
```

```
 [ PPU#, UU# ]. = [2,1].
 Processing Time of ppu #1  (0<PT<10) = 2
 Processing Time of ppu #2  (0<PT<10) = 2
 Processing Time of uu #1  (0<PT<10) = 2
 Processing Time of net  (0<PT<10) = 1
 PROGRAM FILE NAME [ FORMAT : '******.***'. ] = 'xps_test1.out'.
>>goal_frame(append,go).
{34,[[],[1,2,3,4,5]]}
{57,[[1],[2,3,4,5]]}
{80,[[1,2],[3,4,5]]}
{106,[[1,2,3],[4,5]]}
{144,[[1,2,3,4],[5]]}
{154,[[1,2,3,4,5],[]]}
Final Time = 168
### Result of Analysis ###
system_memo(ppu,1,operating,169,'0.88',dummy,149).
system_memo(ppu,1,ready,169,'0.27',1,47).
system_memo(ppu,1,wait,169,'3.62',6,613).
system_memo(ppu,1,garbage,169,'0.17',1,30).
system_memo(ppu,1,suspend,169,'0.00',0,0).
system_memo(ppu,1,input,169,'0.94',3,159).
system_memo(ppu,1,output,169,'0.10',1,18).
system_memo(ppu,2,operating,169,'0.26',dummy,44).
system_memo(ppu,2,ready,169,'0.01',1,2).
system_memo(ppu,2,wait,169,'0.59',2,101).
system_memo(ppu,2,garbage,169,'0.02',1,5).
system_memo(ppu,2,suspend,169,'0.00',0,0).
system_memo(ppu,2,input,169,'0.10',1,17).
system_memo(ppu,2,output,169,'0.08',1,14).
system_memo(uu,1,operating,169,'0.15',dummy,26).
system_memo(uu,1,fork,13/169,'0.10'*'13.00',2,18).
system_memo(uu,1,input,169,'0.07',1,13).
system_memo(uu,1,output,169,'0.31',3,54).
system_memo(net,1,operating,169,'0.53',dummy,91).
system_memo(net,1,p_u,169,'0.07',1,13).
system_memo(net,1,u_p,169,'0.18',1,31).
system_memo(net,1,p_p,169,'0.11',2,19).
```

```
 [ PPU#, UU# ]. = [2,2].
 Processing Time of ppu #1  (0<PT<10) = 2
 Processing Time of ppu #2  (0<PT<10) = 2
 Processing Time of uu #1  (0<PT<10) = 2
 Processing Time of uu #2  (0<PT<10) = 2
 Processing Time of net  (0<PT<10) = 1
 PROGRAM FILE NAME [ FORMAT : '******.***'. ] = 'xps_test1.out'.
>>goal_frame(append,go).
{34,[[],[1,2,3,4,5]]}
{57,[[1],[2,3,4,5]]}
{80,[[1,2],[3,4,5]]}
{106,[[1,2,3],[4,5]]}
{144,[[1,2,3,4],[5]]}
{154,[[1,2,3,4,5],[]]}
Final Time = 168
### Result of Analysis ###
system_memo(ppu,1,operating,169,'0.88',dummy,149).
system_memo(ppu,1,ready,169,'0.27',1,47).
system_memo(ppu,1,wait,169,'3.62',6,613).
system_memo(ppu,1,garbage,169,'0.17',1,30).
system_memo(ppu,1,suspend,169,'0.00',0,0).
system_memo(ppu,1,input,169,'0.94',3,159).
system_memo(ppu,1,output,169,'0.10',1,18).
system_memo(ppu,2,operating,169,'0.26',dummy,44).
system_memo(ppu,2,ready,169,'0.01',1,2).
system_memo(ppu,2,wait,169,'0.59',2,101).
system_memo(ppu,2,garbage,169,'0.02',1,5).
system_memo(ppu,2,suspend,169,'0.00',0,0).
system_memo(ppu,2,input,169,'0.10',1,17).
system_memo(ppu,2,output,169,'0.08',1,14).
system_memo(uu,1,operating,169,'0.05',dummy,10).
system_memo(uu,1,fork,5/169,'0.04'*'33.80',2,8).
system_memo(uu,1,input,169,'0.02',1,5).
system_memo(uu,1,output,169,'0.14',3,24).
system_memo(uu,2,operating,169,'0.09',dummy,16).
system_memo(uu,2,fork,8/169,'0.05'*'21.12',2,10).
system_memo(uu,2,input,169,'0.04',1,8).
system_memo(uu,2,output,169,'0.17',3,30).
system_memo(net,1,operating,169,'0.53',dummy,91).
system_memo(net,1,p_u,169,'0.07',1,13).
system_memo(net,1,u_p,169,'0.18',1,31).
system_memo(net,1,p_p,169,'0.11',2,19).
```

```
 [ PPU#, UU# ]. = [3,1].
 Processing Time of ppu #1  (0<PT<10) = 2
 Processing Time of ppu #2  (0<PT<10) = 2
 Processing Time of ppu #3  (0<PT<10) = 2
 Processing Time of uu #1  (0<PT<10) = 2
 Processing Time of net  (0<PT<10) = 1
 PROGRAM FILE NAME [ FORMAT : '******.***'. ] = 'xps_test1.out'.
>>goal_frame(append,go).
{34,[[],[1,2,3,4,5]]}
{56,[[1],[2,3,4,5]]}
{77,[[1,2],[3,4,5]]}
{95,[[1,2,3],[4,5]]}
{116,[[1,2,3,4],[5]]}
{137,[[1,2,3,4,5],[]]}
Final Time = 146
### Result of Analysis ###
system_memo(ppu,1,operating,147,'0.65',dummy,97).
system_memo(ppu,1,ready,147,'0.19',1,28).
system_memo(ppu,1,wait,147,'2.31',3,340).
system_memo(ppu,1,garbage,147,'0.12',1,18).
system_memo(ppu,1,suspend,147,'0.00',0,0).
system_memo(ppu,1,input,147,'0.28',2,42).
system_memo(ppu,1,output,147,'0.05',1,8).
system_memo(ppu,2,operating,147,'0.17',dummy,26).
system_memo(ppu,2,ready,147,'0.00',1,1).
system_memo(ppu,2,wait,147,'0.40',1,59).
system_memo(ppu,2,garbage,147,'0.00',1,1).
system_memo(ppu,2,suspend,147,'0.00',0,0).
system_memo(ppu,2,input,147,'0.05',1,8).
system_memo(ppu,2,output,147,'0.05',1,8).
system_memo(ppu,3,operating,147,'0.53',dummy,78).
system_memo(ppu,3,ready,147,'0.04',1,6).
system_memo(ppu,3,wait,147,'1.58',4,233).
system_memo(ppu,3,garbage,147,'0.08',1,12).
system_memo(ppu,3,suspend,147,'0.00',0,0).
system_memo(ppu,3,input,147,'0.24',2,36).
system_memo(ppu,3,output,147,'0.13',1,20).
system_memo(uu,1,operating,147,'0.17',dummy,26).
system_memo(uu,1,fork,13/147,'0.12'*'11.30',2,18).
system_memo(uu,1,input,147,'0.09',1,14).
system_memo(uu,1,output,147,'0.39',3,58).
system_memo(net,1,operating,147,'0.63',dummy,93).
system_memo(net,1,p_u,147,'0.08',1,13).
system_memo(net,1,u_p,147,'0.21',1,31).
system_memo(net,1,p_p,147,'0.15',1,23).
```

```
 [ PPU#, UU# ]. = [4,1].
 Processing Time of ppu #1  (0<PT<10) = 2
 Processing Time of ppu #2  (0<PT<10) = 2
 Processing Time of ppu #3  (0<PT<10) = 2
 Processing Time of ppu #4  (0<PT<10) = 2
 Processing Time of uu #1  (0<PT<10) = 2
 Processing Time of net  (0<PT<10) = 1
 PROGRAM FILE NAME [ FORMAT : '******.***'. ] = 'xps_test1.out'.
>>goal_frame(append,go).
{33,[[],[1,2,3,4,5]]}
{59,[[1],[2,3,4,5]]}
{72,[[1,2],[3,4,5]]}
{99,[[1,2,3],[4,5]]}
{123,[[1,2,3,4],[5]]}
{134,[[1,2,3,4,5],[]]}
Final Time = 154
### Result of Analysis ###
system_memo(ppu,1,operating,155,'0.67',dummy,105).
system_memo(ppu,1,ready,155,'0.01',1,2).
system_memo(ppu,1,wait,155,'1.85',3,288).
system_memo(ppu,1,garbage,155,'0.03',1,5).
system_memo(ppu,1,suspend,155,'0.00',0,0).
system_memo(ppu,1,input,155,'0.74',5,115).
system_memo(ppu,1,output,155,'0.12',1,19).
system_memo(ppu,2,operating,155,'0.16',dummy,26).
system_memo(ppu,2,ready,155,'0.00',1,1).
system_memo(ppu,2,wait,155,'0.36',1,57).
system_memo(ppu,2,garbage,155,'0.00',1,1).
system_memo(ppu,2,suspend,155,'0.00',0,0).
system_memo(ppu,2,input,155,'0.05',1,8).
system_memo(ppu,2,output,155,'0.05',1,8).
system_memo(ppu,3,operating,155,'0.67',dummy,104).
system_memo(ppu,3,ready,155,'0.13',1,21).
system_memo(ppu,3,wait,155,'1.34',3,209).
system_memo(ppu,3,garbage,155,'0.18',1,28).
system_memo(ppu,3,suspend,155,'0.00',0,0).
system_memo(ppu,3,input,155,'0.36',2,57).
system_memo(ppu,3,output,155,'0.20',1,32).
system_memo(ppu,4,operating,155,'0.28',dummy,44).
system_memo(ppu,4,ready,155,'0.01',1,2).
system_memo(ppu,4,wait,155,'0.56',2,87).
system_memo(ppu,4,garbage,155,'0.03',1,6).
system_memo(ppu,4,suspend,155,'0.00',0,0).
system_memo(ppu,4,input,155,'0.10',1,17).
system_memo(ppu,4,output,155,'0.09',1,14).
system_memo(uu,1,operating,155,'0.16',dummy,26).
system_memo(uu,1,fork,13/155,'0.11'*'11.92',2,18).
system_memo(uu,1,input,155,'0.09',2,15).
system_memo(uu,1,output,155,'0.38',3,60).
system_memo(net,1,operating,155,'0.78',dummy,122).
system_memo(net,1,p_u,155,'0.08',2,13).
system_memo(net,1,u_p,155,'0.20',1,31).
system_memo(net,1,p_p,155,'0.38',2,60).
```

```
 [ PPU#, UU# ]. = [5,1].
 Processing Time of ppu #1  (0<PT<10) = 2
 Processing Time of ppu #2  (0<PT<10) = 2
 Processing Time of ppu #3  (0<PT<10) = 2
 Processing Time of ppu #4  (0<PT<10) = 2
 Processing Time of ppu #5  (0<PT<10) = 2
 Processing Time of uu #1  (0<PT<10) = 2
 Processing Time of net  (0<PT<10) = 1
 PROGRAM FILE NAME [ FORMAT : '******.***'. ] = 'xps_test1.out'.
>>goal_frame(append,go).
{33,[[],[1,2,3,4,5]]}
{60,[[1],[2,3,4,5]]}
{89,[[1,2],[3,4,5]]}
{103,[[1,2,3],[4,5]]}
{139,[[1,2,3,4],[5]]}
{165,[[1,2,3,4,5],[]]}
Final Time = 180
### Result of Analysis ###
system_memo(ppu,1,operating,181,'0.40',dummy,73).
system_memo(ppu,1,ready,181,'0.00',1,1).
system_memo(ppu,1,wait,181,'1.43',2,260).
system_memo(ppu,1,garbage,181,'0.01',1,3).
system_memo(ppu,1,suspend,181,'0.00',0,0).
system_memo(ppu,1,input,181,'0.21',3,39).
system_memo(ppu,1,output,181,'0.04',1,9).
system_memo(ppu,2,operating,181,'0.81',dummy,148).
system_memo(ppu,2,ready,181,'0.16',1,30).
system_memo(ppu,2,wait,181,'2.06',5,373).
system_memo(ppu,2,garbage,181,'0.18',1,34).
system_memo(ppu,2,suspend,181,'0.00',0,0).
system_memo(ppu,2,input,181,'1.30',5,237).
system_memo(ppu,2,output,181,'0.25',1,46).
system_memo(ppu,3,operating,181,'0.00',dummy,0).
system_memo(ppu,3,ready,181,'0.00',0,0).
system_memo(ppu,3,wait,181,'0.00',0,0).
system_memo(ppu,3,garbage,181,'0.00',0,0).
system_memo(ppu,3,suspend,181,'0.00',0,0).
system_memo(ppu,3,input,181,'0.00',0,0).
system_memo(ppu,3,output,181,'0.00',0,0).
system_memo(ppu,4,operating,181,'0.11',dummy,20).
system_memo(ppu,4,ready,181,'0.00',1,1).
system_memo(ppu,4,wait,181,'0.32',1,59).
system_memo(ppu,4,garbage,181,'0.01',1,2).
system_memo(ppu,4,suspend,181,'0.00',0,0).
system_memo(ppu,4,input,181,'0.03',1,7).
system_memo(ppu,4,output,181,'0.03',1,6).
system_memo(ppu,5,operating,181,'0.20',dummy,38).
system_memo(ppu,5,ready,181,'0.00',1,1).
system_memo(ppu,5,wait,181,'0.67',1,123).
system_memo(ppu,5,garbage,181,'0.00',1,1).
system_memo(ppu,5,suspend,181,'0.00',0,0).
system_memo(ppu,5,input,181,'0.06',1,12).
system_memo(ppu,5,output,181,'0.06',1,12).
system_memo(uu,1,operating,181,'0.14',dummy,26).
system_memo(uu,1,fork,13/181,'0.09'*'13.92',2,18).
system_memo(uu,1,input,181,'0.07',1,13).
system_memo(uu,1,output,181,'0.30',3,56).
system_memo(net,1,operating,181,'0.75',dummy,137).
system_memo(net,1,p_u,181,'0.07',1,13).
system_memo(net,1,u_p,181,'0.17',1,31).
system_memo(net,1,p_p,181,'0.33',2,60).
```

```
 [ PPU#, UU# ]. = [1,1].
 Processing Time of ppu #1  (0<PT<10) = 2
 Processing Time of uu #1  (0<PT<10) = 2
 Processing Time of net  (0<PT<10) = 1
 PROGRAM FILE NAME [ FORMAT : '******.***'. ] = 'xps_test1.out'.
>>goal_frame(factorial,go).
{459,120}
Final Time = 468
### Result of Analysis ###
system_memo(ppu,1,operating,469,'0.73',dummy,347).
system_memo(ppu,1,ready,469,'0.30',1,142).
system_memo(ppu,1,wait,469,'4.09',7,1919).
system_memo(ppu,1,garbage,469,'0.31',1,150).
system_memo(ppu,1,suspend,469,'0.00',0,0).
system_memo(ppu,1,input,469,'0.20',1,98).
system_memo(ppu,1,output,469,'0.07',1,37).
system_memo(uu,1,operating,469,'0.15',dummy,74).
system_memo(uu,1,fork,28/469,'0.07'*'16.75',2,37).
system_memo(uu,1,input,469,'0.07',1,37).
system_memo(uu,1,output,469,'0.25',3,120).
system_memo(net,1,operating,469,'0.37',dummy,176).
system_memo(net,1,p_u,469,'0.07',1,37).
system_memo(net,1,u_p,469,'0.15',1,74).
system_memo(net,1,p_p,469,'0.00',0,0).
```

```
 [ PPU#, UU# ]. = [1,2].
 Processing Time of ppu #1  (0<PT<10) = 2
 Processing Time of uu #1  (0<PT<10) = 2
 Processing Time of uu #2  (0<PT<10) = 2
 Processing Time of net  (0<PT<10) = 1
 PROGRAM FILE NAME [ FORMAT : '******.***'. ] = 'xps_test1.out'.
>>goal_frame(factorial,go).
{459,120}
Final Time = 468
### Result of Analysis ###
system_memo(ppu,1,operating,469,'0.73',dummy,347).
system_memo(ppu,1,ready,469,'0.30',1,142).
system_memo(ppu,1,wait,469,'4.09',7,1919).
system_memo(ppu,1,garbage,469,'0.31',1,150).
system_memo(ppu,1,suspend,469,'0.00',0,0).
system_memo(ppu,1,input,469,'0.20',1,98).
system_memo(ppu,1,output,469,'0.07',1,37).
system_memo(uu,1,operating,469,'0.07',dummy,36).
system_memo(uu,1,fork,12/469,'0.03'*'39.08',2,16).
system_memo(uu,1,input,469,'0.03',1,18).
system_memo(uu,1,output,469,'0.11',3,54).
system_memo(uu,2,operating,469,'0.08',dummy,38).
system_memo(uu,2,fork,16/469,'0.04'*'29.31',2,21).
system_memo(uu,2,input,469,'0.04',1,19).
system_memo(uu,2,output,469,'0.14',3,66).
system_memo(net,1,operating,469,'0.37',dummy,176).
system_memo(net,1,p_u,469,'0.07',1,37).
system_memo(net,1,u_p,469,'0.15',1,74).
system_memo(net,1,p_p,469,'0.00',0,0).
```

```
 [ PPU#, UU# ]. = [2,1].
 Processing Time of ppu #1  (0<PT<10) = 2
 Processing Time of ppu #2  (0<PT<10) = 2
 Processing Time of uu #1  (0<PT<10) = 2
 Processing Time of net  (0<PT<10) = 1
 PROGRAM FILE NAME [ FORMAT : '******.***'. ] = 'xps_test1.out'.
>>goal_frame(factorial,go).
{461,120}
Final Time = 470
### Result of Analysis ###
system_memo(ppu,1,operating,471,'0.40',dummy,193).
system_memo(ppu,1,ready,471,'0.06',1,30).
system_memo(ppu,1,wait,471,'2.57',4,1211).
system_memo(ppu,1,garbage,471,'0.08',1,42).
system_memo(ppu,1,suspend,471,'0.00',0,0).
system_memo(ppu,1,input,471,'0.12',1,60).
system_memo(ppu,1,output,471,'0.07',1,36).
system_memo(ppu,2,operating,471,'0.51',dummy,244).
system_memo(ppu,2,ready,471,'0.13',1,63).
system_memo(ppu,2,wait,471,'1.47',3,693).
system_memo(ppu,2,garbage,471,'0.18',1,85).
system_memo(ppu,2,suspend,471,'0.00',0,0).
system_memo(ppu,2,input,471,'0.15',1,72).
system_memo(ppu,2,output,471,'0.09',1,46).
system_memo(uu,1,operating,471,'0.15',dummy,74).
system_memo(uu,1,fork,28/471,'0.07'*'16.82',2,37).
system_memo(uu,1,input,471,'0.10',2,51).
system_memo(uu,1,output,471,'0.25',3,120).
system_memo(net,1,operating,471,'0.53',dummy,252).
system_memo(net,1,p_u,471,'0.07',2,37).
system_memo(net,1,u_p,471,'0.15',1,74).
system_memo(net,1,p_p,471,'0.09',1,45).
```

```
 [ PPU#, UU# ]. = [2,2].
 Processing Time of ppu #1  (0<PT<10) = 2
 Processing Time of ppu #2  (0<PT<10) = 2
 Processing Time of uu #1  (0<PT<10) = 2
 Processing Time of uu #2  (0<PT<10) = 2
 Processing Time of net  (0<PT<10) = 1
 PROGRAM FILE NAME [ FORMAT : '******.***'. ] = 'xps_test1.out'.
>>goal_frame(factorial,go).
{452,120}
Final Time = 461
### Result of Analysis ###
system_memo(ppu,1,operating,462,'0.41',dummy,193).
system_memo(ppu,1,ready,462,'0.06',1,30).
system_memo(ppu,1,wait,462,'2.57',4,1188).
system_memo(ppu,1,garbage,462,'0.09',1,42).
system_memo(ppu,1,suspend,462,'0.00',0,0).
system_memo(ppu,1,input,462,'0.12',1,60).
system_memo(ppu,1,output,462,'0.07',1,36).
system_memo(ppu,2,operating,462,'0.52',dummy,244).
system_memo(ppu,2,ready,462,'0.13',1,63).
system_memo(ppu,2,wait,462,'1.44',3,669).
system_memo(ppu,2,garbage,462,'0.18',1,85).
system_memo(ppu,2,suspend,462,'0.00',0,0).
system_memo(ppu,2,input,462,'0.14',1,67).
system_memo(ppu,2,output,462,'0.09',1,46).
system_memo(uu,1,operating,462,'0.07',dummy,36).
system_memo(uu,1,fork,13/462,'0.03'*'35.53',2,17).
system_memo(uu,1,input,462,'0.03',1,18).
system_memo(uu,1,output,462,'0.12',3,56).
system_memo(uu,2,operating,462,'0.08',dummy,38).
system_memo(uu,2,fork,15/462,'0.04'*'30.80',2,20).
system_memo(uu,2,input,462,'0.04',1,19).
system_memo(uu,2,output,462,'0.13',3,64).
system_memo(net,1,operating,462,'0.51',dummy,239).
system_memo(net,1,p_u,462,'0.08',2,37).
system_memo(net,1,u_p,462,'0.16',2,74).
system_memo(net,1,p_p,462,'0.09',1,45).
```

```
 [ PPU#, UU# ]. = [2,3].
 Processing Time of ppu #1  (0<PT<10) = 2
 Processing Time of ppu #2  (0<PT<10) = 2
 Processing Time of uu #1  (0<PT<10) = 2
 Processing Time of uu #2  (0<PT<10) = 2
 Processing Time of uu #3  (0<PT<10) = 2
 Processing Time of net  (0<PT<10) = 1
 PROGRAM FILE NAME [ FORMAT : '******.***'. ] = 'xps_test1.out'.
>>goal_frame(factorial,go).
{455,120}
Final Time = 464
### Result of Analysis ###
system_memo(ppu,1,operating,465,'0.41',dummy,193).
system_memo(ppu,1,ready,465,'0.06',1,30).
system_memo(ppu,1,wait,465,'2.57',4,1196).
system_memo(ppu,1,garbage,465,'0.09',1,42).
system_memo(ppu,1,suspend,465,'0.00',0,0).
system_memo(ppu,1,input,465,'0.12',1,60).
system_memo(ppu,1,output,465,'0.07',1,36).
system_memo(ppu,2,operating,465,'0.52',dummy,244).
system_memo(ppu,2,ready,465,'0.13',1,63).
system_memo(ppu,2,wait,465,'1.45',3,677).
system_memo(ppu,2,garbage,465,'0.18',1,85).
system_memo(ppu,2,suspend,465,'0.00',0,0).
system_memo(ppu,2,input,465,'0.15',1,70).
system_memo(ppu,2,output,465,'0.09',1,46).
system_memo(uu,1,operating,465,'0.03',dummy,16).
system_memo(uu,1,fork,8/465,'0.02'*'58.12',2,11).
system_memo(uu,1,input,465,'0.01',1,8).
system_memo(uu,1,output,465,'0.07',3,33).
system_memo(uu,2,operating,465,'0.07',dummy,34).
system_memo(uu,2,fork,11/465,'0.03'*'42.27',2,14).
system_memo(uu,2,input,465,'0.04',2,19).
system_memo(uu,2,output,465,'0.10',3,48).
system_memo(uu,3,operating,465,'0.05',dummy,24).
system_memo(uu,3,fork,9/465,'0.02'*'51.66',2,12).
system_memo(uu,3,input,465,'0.02',1,13).
system_memo(uu,3,output,465,'0.08',3,39).
system_memo(net,1,operating,465,'0.52',dummy,242).
system_memo(net,1,p_u,465,'0.07',2,37).
system_memo(net,1,u_p,465,'0.15',2,74).
system_memo(net,1,p_p,465,'0.09',1,45).
```

```
 [ PPU#, UU# ]. = [3,2].
 Processing Time of ppu #1  (0<PT<10) = 2
 Processing Time of ppu #2  (0<PT<10) = 2
 Processing Time of ppu #3  (0<PT<10) = 2
 Processing Time of uu #1  (0<PT<10) = 2
 Processing Time of uu #2  (0<PT<10) = 2
 Processing Time of net  (0<PT<10) = 1
 PROGRAM FILE NAME [ FORMAT : '******.***'. ] = 'xps_test1.out'.
>>goal_frame(factorial,go).
{458,120}
Final Time = 467
### Result of Analysis ###
system_memo(ppu,1,operating,468,'0.17',dummy,83).
system_memo(ppu,1,ready,468,'0.02',1,12).
system_memo(ppu,1,wait,468,'2.04',4,957).
system_memo(ppu,1,garbage,468,'0.03',1,18).
system_memo(ppu,1,suspend,468,'0.00',0,0).
system_memo(ppu,1,input,468,'0.05',1,28).
system_memo(ppu,1,output,468,'0.02',1,13).
system_memo(ppu,2,operating,468,'0.46',dummy,216).
system_memo(ppu,2,ready,468,'0.10',1,51).
system_memo(ppu,2,wait,468,'1.27',4,597).
system_memo(ppu,2,garbage,468,'0.13',1,62).
system_memo(ppu,2,suspend,468,'0.00',0,0).
system_memo(ppu,2,input,468,'0.15',2,71).
system_memo(ppu,2,output,468,'0.09',1,46).
system_memo(ppu,3,operating,468,'0.35',dummy,164).
system_memo(ppu,3,ready,468,'0.08',1,41).
system_memo(ppu,3,wait,468,'0.75',3,353).
system_memo(ppu,3,garbage,468,'0.10',1,49).
system_memo(ppu,3,suspend,468,'0.00',0,0).
system_memo(ppu,3,input,468,'0.11',1,54).
system_memo(ppu,3,output,468,'0.07',1,36).
system_memo(uu,1,operating,468,'0.07',dummy,36).
system_memo(uu,1,fork,13/468,'0.03'*'36.00',2,17).
system_memo(uu,1,input,468,'0.03',1,18).
system_memo(uu,1,output,468,'0.11',3,56).
system_memo(uu,2,operating,468,'0.08',dummy,38).
system_memo(uu,2,fork,15/468,'0.04'*'31.20',2,20).
system_memo(uu,2,input,468,'0.04',1,19).
system_memo(uu,2,output,468,'0.13',3,64).
system_memo(net,1,operating,468,'0.57',dummy,269).
system_memo(net,1,p_u,468,'0.07',2,37).
system_memo(net,1,u_p,468,'0.15',2,74).
system_memo(net,1,p_p,468,'0.12',1,58).
```

```
 [ PPU#, UU# ]. = [1,1].
 Processing Time of ppu #1  (0<PT<10) = 2
 Processing Time of uu #1  (0<PT<10) = 2
 Processing Time of net  (0<PT<10) = 1
 PROGRAM FILE NAME [ FORMAT : '******.***'. ] = 'xps_test2.out'.
>>goal_frame(append,go).
{38,[[],[1,2,3,4,5]]}
{53,[[1],[2,3,4,5]]}
{75,[[1,2],[3,4,5]]}
{99,[[1,2,3],[4,5]]}
{125,[[1,2,3,4],[5]]}
{151,[[1,2,3,4,5],[]]}
Final Time = 169
### Result of Analysis ###
system_memo(ppu,1,operating,170,'0.91',dummy,155).
system_memo(ppu,1,ready,170,'0.44',1,76).
system_memo(ppu,1,wait,170,'5.20',9,885).
system_memo(ppu,1,garbage,170,'0.32',2,56).
system_memo(ppu,1,suspend,170,'0.00',0,0).
system_memo(ppu,1,input,170,'1.62',5,277).
system_memo(ppu,1,output,170,'0.07',1,13).
system_memo(uu,1,operating,170,'0.15',dummy,26).
system_memo(uu,1,fork,13/170,'0.10'*'13.07',2,18).
system_memo(uu,1,input,170,'0.07',1,13).
system_memo(uu,1,output,170,'0.31',3,54).
system_memo(net,1,operating,170,'0.38',dummy,66).
system_memo(net,1,p_u,170,'0.07',1,13).
system_memo(net,1,u_p,170,'0.18',1,31).
system_memo(net,1,p_p,170,'0.00',0,0).
```

```
 [ PPU#, UU# ]. = [2,1].
 Processing Time of ppu #1  (0<PT<10) = 2
 Processing Time of ppu #2  (0<PT<10) = 2
 Processing Time of uu #1  (0<PT<10) = 2
 Processing Time of net  (0<PT<10) = 1
 PROGRAM FILE NAME [ FORMAT : '******.***'. ] = 'xps_test2.out'.
>>goal_frame(append,go).
{38,[[],[1,2,3,4,5]]}
{49,[[1],[2,3,4,5]]}
{75,[[1,2],[3,4,5]]}
{109,[[1,2,3],[4,5]]}
{119,[[1,2,3,4],[5]]}
{155,[[1,2,3,4,5],[]]}
Final Time = 164
### Result of Analysis ###
system_memo(ppu,1,operating,165,'0.90',dummy,149).
system_memo(ppu,1,ready,165,'0.27',1,45).
system_memo(ppu,1,wait,165,'3.93',7,649).
system_memo(ppu,1,garbage,165,'0.16',1,28).
system_memo(ppu,1,suspend,165,'0.00',0,0).
system_memo(ppu,1,input,165,'1.22',4,202).
system_memo(ppu,1,output,165,'0.10',1,18).
system_memo(ppu,2,operating,165,'0.26',dummy,44).
system_memo(ppu,2,ready,165,'0.01',1,3).
system_memo(ppu,2,wait,165,'0.66',2,110).
system_memo(ppu,2,garbage,165,'0.03',1,5).
system_memo(ppu,2,suspend,165,'0.00',0,0).
system_memo(ppu,2,input,165,'0.11',2,19).
system_memo(ppu,2,output,165,'0.08',1,14).
system_memo(uu,1,operating,165,'0.15',dummy,26).
system_memo(uu,1,fork,13/165,'0.10'*'12.69',2,18).
system_memo(uu,1,input,165,'0.07',1,13).
system_memo(uu,1,output,165,'0.32',3,54).
system_memo(net,1,operating,165,'0.54',dummy,90).
system_memo(net,1,p_u,165,'0.07',1,13).
system_memo(net,1,u_p,165,'0.18',1,31).
system_memo(net,1,p_p,165,'0.11',1,19).
```

```
 [ PPU#, UU# ]. = [3,1].
 Processing Time of ppu #1  (0<PT<10) = 2
 Processing Time of ppu #2  (0<PT<10) = 2
 Processing Time of ppu #3  (0<PT<10) = 2
 Processing Time of uu #1  (0<PT<10) = 2
 Processing Time of net  (0<PT<10) = 1
 PROGRAM FILE NAME [ FORMAT : '******.***'. ] = 'xps_test2.out'.
>>goal_frame(append,go).
{34,[[],[1,2,3,4,5]]}
{56,[[1],[2,3,4,5]]}
{68,[[1,2],[3,4,5]]}
{89,[[1,2,3],[4,5]]}
{121,[[1,2,3,4],[5]]}
{129,[[1,2,3,4,5],[]]}
Final Time = 141
### Result of Analysis ###
system_memo(ppu,1,operating,142,'0.68',dummy,97).
system_memo(ppu,1,ready,142,'0.14',1,20).
system_memo(ppu,1,wait,142,'2.32',4,330).
system_memo(ppu,1,garbage,142,'0.12',1,18).
system_memo(ppu,1,suspend,142,'0.00',0,0).
system_memo(ppu,1,input,142,'0.28',2,41).
system_memo(ppu,1,output,142,'0.05',1,8).
system_memo(ppu,2,operating,142,'0.18',dummy,26).
system_memo(ppu,2,ready,142,'0.00',1,1).
system_memo(ppu,2,wait,142,'0.39',1,56).
system_memo(ppu,2,garbage,142,'0.00',1,1).
system_memo(ppu,2,suspend,142,'0.00',0,0).
system_memo(ppu,2,input,142,'0.05',1,8).
system_memo(ppu,2,output,142,'0.05',1,8).
system_memo(ppu,3,operating,142,'0.54',dummy,78).
system_memo(ppu,3,ready,142,'0.04',1,6).
system_memo(ppu,3,wait,142,'1.52',4,217).
system_memo(ppu,3,garbage,142,'0.09',1,14).
system_memo(ppu,3,suspend,142,'0.00',0,0).
system_memo(ppu,3,input,142,'0.35',3,50).
system_memo(ppu,3,output,142,'0.14',1,20).
system_memo(uu,1,operating,142,'0.18',dummy,26).
system_memo(uu,1,fork,13/142,'0.12'*'10.92',2,18).
system_memo(uu,1,input,142,'0.10',2,15).
system_memo(uu,1,output,142,'0.38',3,54).
system_memo(net,1,operating,142,'0.65',dummy,93).
system_memo(net,1,p_u,142,'0.09',2,13).
system_memo(net,1,u_p,142,'0.21',1,31).
system_memo(net,1,p_p,142,'0.16',2,23).
```

```
 [ PPU#, UU# ]. = [1,1].
 Processing Time of ppu #1  (0<PT<10) = 2
 Processing Time of uu #1  (0<PT<10) = 2
 Processing Time of net  (0<PT<10) = 1
 PROGRAM FILE NAME [ FORMAT : '******.***'. ] = 'xps_test2.out'.
>>goal_frame(factorial,go).
{463,120}
Final Time = 472
### Result of Analysis ###
system_memo(ppu,1,operating,473,'0.73',dummy,347).
system_memo(ppu,1,ready,473,'0.30',1,142).
system_memo(ppu,1,wait,473,'4.05',7,1918).
system_memo(ppu,1,garbage,473,'0.32',1,155).
system_memo(ppu,1,suspend,473,'0.00',0,0).
system_memo(ppu,1,input,473,'0.18',1,87).
system_memo(ppu,1,output,473,'0.07',1,37).
system_memo(uu,1,operating,473,'0.15',dummy,74).
system_memo(uu,1,fork,28/473,'0.07'*'16.89',2,37).
system_memo(uu,1,input,473,'0.07',1,37).
system_memo(uu,1,output,473,'0.25',3,120).
system_memo(net,1,operating,473,'0.37',dummy,176).
system_memo(net,1,p_u,473,'0.07',1,37).
system_memo(net,1,u_p,473,'0.15',1,74).
system_memo(net,1,p_p,473,'0.00',0,0).
```

```
 [ PPU#, UU# ]. = [1,2].
 Processing Time of ppu #1  (0<PT<10) = 2
 Processing Time of uu #1  (0<PT<10) = 2
 Processing Time of uu #2  (0<PT<10) = 2
 Processing Time of net  (0<PT<10) = 1
 PROGRAM FILE NAME [ FORMAT : '*******.***'. ] = 'xps_test2.out'.
>>goal_frame(factorial,go).
{463,120}
Final Time = 472
### Result of Analysis ###
system_memo(ppu,1,operating,473,'0.73',dummy,347).
system_memo(ppu,1,ready,473,'0.30',1,142).
system_memo(ppu,1,wait,473,'4.05',7,1918).
system_memo(ppu,1,garbage,473,'0.32',1,155).
system_memo(ppu,1,suspend,473,'0.00',0,0).
system_memo(ppu,1,input,473,'0.18',1,87).
system_memo(ppu,1,output,473,'0.07',1,37).
system_memo(uu,1,operating,473,'0.07',dummy,36).
system_memo(uu,1,fork,15/473,'0.04'*'31.53',2,19).
system_memo(uu,1,input,473,'0.03',1,18).
system_memo(uu,1,output,473,'0.12',3,60).
system_memo(uu,2,operating,473,'0.08',dummy,38).
system_memo(uu,2,fork,13/473,'0.03'*'36.38',2,18).
system_memo(uu,2,input,473,'0.04',1,19).
system_memo(uu,2,output,473,'0.12',3,60).
system_memo(net,1,operating,473,'0.37',dummy,176).
system_memo(net,1,p_u,473,'0.07',1,37).
system_memo(net,1,u_p,473,'0.15',1,74).
system_memo(net,1,p_p,473,'0.00',0,0).
```

```
 [ PPU#, UU# ]. = [2,1].
 Processing Time of ppu #1  (0<PT<10) = 2
 Processing Time of ppu #2  (0<PT<10) = 2
 Processing Time of uu #1  (0<PT<10) = 2
 Processing Time of net  (0<PT<10) = 1
 PROGRAM FILE NAME [ FORMAT : '******.***'. ] = 'xps_test2.out'.
>>goal_frame(factorial,go).
{457,120}
Final Time = 466
### Result of Analysis ###
system_memo(ppu,1,operating,467,'0.57',dummy,269).
system_memo(ppu,1,ready,467,'0.15',1,72).
system_memo(ppu,1,wait,467,'3.42',5,1600).
system_memo(ppu,1,garbage,467,'0.20',1,95).
system_memo(ppu,1,suspend,467,'0.00',0,0).
system_memo(ppu,1,input,467,'0.16',1,79).
system_memo(ppu,1,output,467,'0.08',1,38).
system_memo(ppu,2,operating,467,'0.31',dummy,148).
system_memo(ppu,2,ready,467,'0.05',1,24).
system_memo(ppu,2,wait,467,'0.64',2,300).
system_memo(ppu,2,garbage,467,'0.07',1,34).
system_memo(ppu,2,suspend,467,'0.00',0,0).
system_memo(ppu,2,input,467,'0.09',1,43).
system_memo(ppu,2,output,467,'0.07',1,34).
system_memo(uu,1,operating,467,'0.15',dummy,74).
system_memo(uu,1,fork,28/467,'0.07'*'16.67',2,37).
system_memo(uu,1,input,467,'0.10',2,51).
system_memo(uu,1,output,467,'0.25',3,120).
system_memo(net,1,operating,467,'0.49',dummy,232).
system_memo(net,1,p_u,467,'0.07',2,37).
system_memo(net,1,u_p,467,'0.15',1,74).
system_memo(net,1,p_p,467,'0.07',1,35).
```

```
 [ PPU#, UU# ]. = [2,2].
 Processing Time of ppu #1  (0<PT<10) = 2
 Processing Time of ppu #2  (0<PT<10) = 2
 Processing Time of uu #1  (0<PT<10) = 2
 Processing Time of uu #2  (0<PT<10) = 2
 Processing Time of net  (0<PT<10) = 1
 PROGRAM FILE NAME [ FORMAT : '******.***'. ] = 'xps_test2.out'.
>>goal_frame(factorial,go).
{451,120}
Final Time = 460
### Result of Analysis ###
system_memo(ppu,1,operating,461,'0.58',dummy,269).
system_memo(ppu,1,ready,461,'0.15',1,72).
system_memo(ppu,1,wait,461,'3.41',5,1573).
system_memo(ppu,1,garbage,461,'0.20',1,95).
system_memo(ppu,1,suspend,461,'0.00',0,0).
system_memo(ppu,1,input,461,'0.17',1,81).
system_memo(ppu,1,output,461,'0.08',1,38).
system_memo(ppu,2,operating,461,'0.32',dummy,148).
system_memo(ppu,2,ready,461,'0.05',1,24).
system_memo(ppu,2,wait,461,'0.62',2,286).
system_memo(ppu,2,garbage,461,'0.07',1,34).
system_memo(ppu,2,suspend,461,'0.00',0,0).
system_memo(ppu,2,input,461,'0.09',1,43).
system_memo(ppu,2,output,461,'0.07',1,34).
system_memo(uu,1,operating,461,'0.07',dummy,36).
system_memo(uu,1,fork,14/461,'0.03'*'32.92',2,18).
system_memo(uu,1,input,461,'0.03',1,18).
system_memo(uu,1,output,461,'0.12',3,58).
system_memo(uu,2,operating,461,'0.08',dummy,38).
system_memo(uu,2,fork,14/461,'0.04'*'32.92',2,19).
system_memo(uu,2,input,461,'0.04',1,19).
system_memo(uu,2,output,461,'0.13',3,62).
system_memo(net,1,operating,461,'0.47',dummy,221).
system_memo(net,1,p_u,461,'0.08',2,37).
system_memo(net,1,u_p,461,'0.16',2,74).
system_memo(net,1,p_p,461,'0.07',1,35).
```

```
 [ PPU#, UU# ]. = [2,3].
 Processing Time of ppu #1  (0<PT<10) = 2
 Processing Time of ppu #2  (0<PT<10) = 2
 Processing Time of uu #1  (0<PT<10) = 2
 Processing Time of uu #2  (0<PT<10) = 2
 Processing Time of uu #3  (0<PT<10) = 2
 Processing Time of net  (0<PT<10) = 1
 PROGRAM FILE NAME [ FORMAT : '******.***'. ] = 'xps_test2.out'.
>>goal_frame(factorial,go).
[452,120]
Final Time = 461
### Result of Analysis ###
system_memo(ppu,1,operating,462,'0.58',dummy,269).
system_memo(ppu,1,ready,462,'0.15',1,72).
system_memo(ppu,1,wait,462,'3.41',5,1578).
system_memo(ppu,1,garbage,462,'0.20',1,95).
system_memo(ppu,1,suspend,462,'0.00',0,0).
system_memo(ppu,1,input,462,'0.17',1,80).
system_memo(ppu,1,output,462,'0.08',1,38).
system_memo(ppu,2,operating,462,'0.32',dummy,148).
system_memo(ppu,2,ready,462,'0.05',1,24).
system_memo(ppu,2,wait,462,'0.62',2,289).
system_memo(ppu,2,garbage,462,'0.07',1,34).
system_memo(ppu,2,suspend,462,'0.00',0,0).
system_memo(ppu,2,input,462,'0.09',1,43).
system_memo(ppu,2,output,462,'0.07',1,34).
system_memo(uu,1,operating,462,'0.03',dummy,16).
system_memo(uu,1,fork,7/462,'0.02'*'66.00',2,10).
system_memo(uu,1,input,462,'0.01',1,8).
system_memo(uu,1,output,462,'0.06',3,31).
system_memo(uu,2,operating,462,'0.07',dummy,34).
system_memo(uu,2,fork,13/462,'0.03'*'35.53',2,16).
system_memo(uu,2,input,462,'0.03',1,18).
system_memo(uu,2,output,462,'0.11',3,52).
system_memo(uu,3,operating,462,'0.05',dummy,24).
system_memo(uu,3,fork,8/462,'0.02'*'57.75',2,11).
system_memo(uu,3,input,462,'0.03',2,14).
system_memo(uu,3,output,462,'0.08',3,37).
system_memo(net,1,operating,462,'0.48',dummy,223).
system_memo(net,1,p_u,462,'0.08',2,37).
system_memo(net,1,u_p,462,'0.16',2,74).
system_memo(net,1,p_p,462,'0.07',1,35).
```